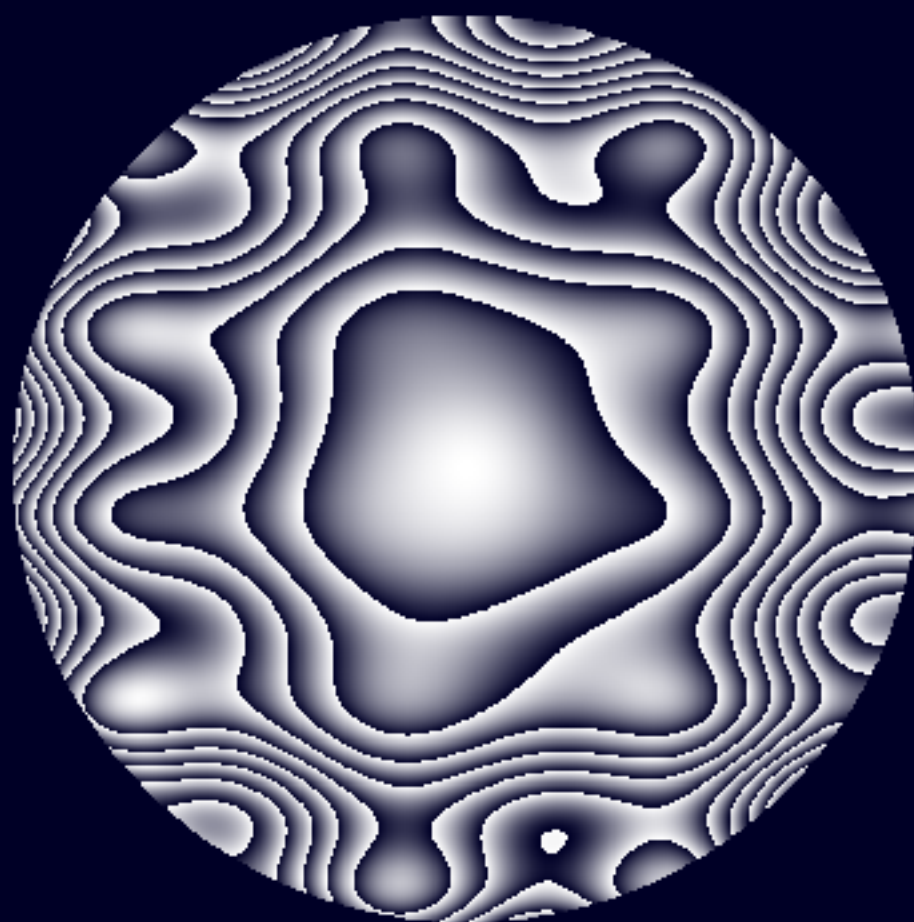


SURVO

**An Integrated Environment
for Statistical Computing
and Related Areas**



Seppo Mustonen

SURVO
An Integrated Environment
for Statistical Computing and Related Areas

by
Seppo Mustonen
Professor of Statistics
University of Helsinki

Published 1992, Survo Systems Ltd, Helsinki, Finland

Copyright © 1992 by the author

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author.

The contents of this paper are furnished for informational use only, are subject to change without notice, and should not be construed as a commitment by the author. The author assumes no responsibility or liability for any errors or inaccuracies that may appear in this paper. The software described in this paper is furnished under license and may be used or copied only in accordance with the terms of this license.

The functions described in this book require the SURVO 84C version 4.04 or later with appropriate options.

This paper was composed and written using SURVO 84C. The original of the paper was created as PostScript files by the PRINT operation of SURVO 84C. Proofs were printed on the QMS-PS 810 PostScript printer.

The final copies were printed by Helsinki University Printing House, Helsinki, Finland.

ISBN 951-96634-0-1

This document appeared in 1992.

In this version various comments about new features are included.



Obsolete passages (typically replaced by better alternatives in SURVO MM) are displayed in gray.

Information about new features is also obtained by clicking red keywords working as links to the Web Edition of SURVO MM Help system.

Example: **HELP?**

PREFACE

The process leading to the current version of Survo and to this book was started over 30 years ago. Already then, when making statistical programs, I had discussions with my friends about the future of data processing. We had more or less specific dreams about how the things related to statistical computing should be arranged. In those discussions, the problems of the integration of various activities and the interactivity between the user and the computer had an important role. However, computers at that time were only able to satisfy our wishes to a slight degree.

The emergence of minicomputers in the 1960's and personal computers in the 1980's dramatically changed the situation. Since then, we have had ample means for making old dreams true.

Throughout the different versions of the Survo system, I have tried to follow those early visions. Each generation of the progress and the various practical problems encountered have, of course, given stimuli for new creations.

For example, the idea of an editorial approach arose in connection with music printing. In order to help our son who is a musician, I wrote, in 1980, a program for the notation of music by means of a plotter connected to a minicomputer. To enter a composition into this program, the manuscript had to be coded in a special language as a series of code words. In order to edit this code, I wrote an editor program, too. A set of compositions by Olli Mustonen for solo violin was printed by this program and published later in 1983.

LOOK!

Only after this experiment, I learned that the same editor could host statistical data and applications as well. In short, the editor became the basis for the integration of all activities. It was also important that in the editorial approach the fundamental element is one byte (character, digit) and not a cell (number, word, formula) as in spreadsheet programs. This makes the environment flexible since both text processing and spreadsheet computing activities are available on the same platform.

I had believed earlier that interactivity in data processing was the same as a conversational or menu-based interface. The editorial approach opened my eyes to appreciate interactivity as a more general concept. In the Survo environment, we can now speak about interactivity on three different levels.

Interactivity between the user and the system is, of course, the most obvious one. In Survo, this means an interplay on equal terms in contrast to many menu-based programs where the system tends to take a leading role by compelling the user to follow predetermined paths only.

Another form of interactivity appears between the programmer and the system. Currently, Survo is a medium for its own programming, too. Within a Survo session, one can write new program modules in C in the edit field, compile and link them as new Survo operations and, finally, test and apply

them with sample data and commands in the same edit field. This feature has had a tremendous impact on the development of the system by speeding up programming and debugging. Similarly, Survo has become an interactive environment for an application programmer by providing various means of its own, like the sucro language and the matrix interpreter.

The third and, for me, the most rewarding form of interactivity has been the cooperation with active users. Many people have made contributions to Survo by their suggestions and criticism. My task has been to evaluate and to filter these impulses and to make the technical improvements and extensions that were found necessary. The list of these 'benevolent interactors' is too long to include here. However, they are remembered and I am deeply grateful for their cooperation during the last three decades.

This book is a completely revised version of the Survo 84 User's Guide which appeared five years ago in 1987. I am pleased to express my gratitude to Esa Ala-Uotila, Arto Kallinen, Markku Korhonen, Anna-Riitta Niskanen, Jyrki Nummela, Jari Pakkanen, Juha Puranen, Marjut Schreck, and Lauri Tarkkonen for checking the manuscript and for many helpful comments. Survo itself has also been a valuable tool in preparing the manuscript and the final camera-ready copy.

Last but not least, I would like to thank my wife Marja-Liisa and our daughter and son, Elina and Olli, for their help and empathy for my work during so many years.

SM

Hituniemi,
26 September, 1992

Contents

| | |
|---|----|
| 1. Introduction | 1 |
| 2. An Overview of Survo | 4 |
| 2.1 Entering Survo and exit from it | 4 |
| 2.2 Edit field | 5 |
| 2.3 Control of the edit field | 5 |
| 2.4 Operations and commands | 6 |
| 2.5 Work schemes | 7 |
| 2.6* Subfields | 9 |
| 2.7* Operation sequences | 10 |
| 2.8* Tutorial mode and sucros | 11 |
| 2.9 Inquiry system | 12 |
| 2.10* Calling other programs and the operating system | 14 |
| 2.11 List of operations | 15 |
| 3. Text processing | 18 |
| 3.1 General principles | 19 |
| 3.2* Printing a complete document | 21 |
| 3.3 Control of the line length | 22 |
| 3.4 Moving and copying text and tables | 24 |
| 3.5 Clearing the edit field | 27 |
| 3.6 Searching and replacing | 28 |
| 3.7 Moving data between edit fields and text files | 31 |
| 3.8 Output files | 33 |
| 3.9* LIST operations | 33 |
| 4. Table management | 35 |
| 4.1 Editing tables in the edit field | 35 |
| 4.2 Sorting of tables | 38 |
| 4.3 Making new columns | 40 |
| 4.4* Interpolation | 43 |
| 4.5* Transposing | 45 |
| 5. Arithmetics and spreadsheet computing | 46 |
| 5.1 Editorial computing: arithmetics | 47 |
| 5.2 Editorial computing: functions | 49 |
| 5.3* Editorial computing: control statements | 52 |
| 5.4 Computation schemes | 55 |
| 5.5 Numerical conversions | 57 |

| | | |
|-----------|--|------------|
| 5.5.1 | Physical measurements | 58 |
| 5.5.2 | Currencies | 60 |
| 5.5.3 | Number systems | 61 |
| 5.5.4* | Decimal numbers to fractions | 62 |
| 5.5.5 | Prime factors of an integer | 64 |
| 5.6 | Touch mode | 64 |
| 5.6.1 | Touch mode: computing with single numbers | 65 |
| 5.6.2 | Touch mode: touch chains | 67 |
| 5.6.3* | Touch mode: editing touch chains | 70 |
| 6. | Data base management | 75 |
| 6.1 | Data lists | 76 |
| 6.2 | Data tables | 79 |
| 6.3 | Data files | 82 |
| 6.4 | Data saving and editing | 90 |
| 6.4.1 | FILE SHOW operation | 91 |
| 6.4.2 | FILE EDIT operation | 96 |
| 6.5 | Selecting fields and field attributes | 98 |
| 6.6 | Selecting observations | 102 |
| 6.7 | Sorting of data files | 104 |
| 6.8 | Copying and merging of data files | 105 |
| 6.9 | Text files | 107 |
| 6.10 | Transformation of variables | 115 |
| 6.11 | Generating data by simulation | 124 |
| 6.12 | Generating new variables by classification | 125 |
| 6.13 | Same transformation for several variables | 128 |
| 6.14 | Aggregation of observations | 128 |
| 7. | Statistical operations | 131 |
| 7.1 | General principles | 132 |
| 7.1.1 | Representation of statistical data | 132 |
| 7.1.2 | Size of statistical data | 132 |
| 7.1.3 | Selection of variables and observations | 133 |
| 7.1.4 | Computing accuracy | 134 |
| 7.1.5 | Treating of missing values | 134 |
| 7.1.6* | Checking scale types | 134 |
| 7.1.7 | Printout of results | 135 |
| 7.1.8* | Results in matrix files | 137 |
| 7.1.9 | Accuracy and extent of results | 138 |
| 7.2 | Basic statistics | 139 |
| 7.2.1 | STAT operation | 139 |
| 7.2.2 | CORR operation | 142 |
| 7.2.3 | TAB operation | 145 |
| 7.2.4 | HISTO operation | 150 |
| 7.3 | Testing small samples | 162 |

| | | |
|-----------|--|------------|
| 7.3.1 | Comparing two or more independent samples | 163 |
| 7.3.2 | Pairwise comparisons, rank correlations | 166 |
| 7.3.3 | Testing for normality | 167 |
| 7.4 | Linear regression analysis | 168 |
| 7.4.1 | LINREG operation | 168 |
| 7.4.2 | Regression diagnostics | 171 |
| 7.4.3 | Semiparametric smoothing | 175 |
| 7.5 | ESTIMATE operation | 178 |
| 7.5.1 | Simple example | 178 |
| 7.5.2 | Analytical derivatives | 180 |
| 7.5.3 | Computational methods | 181 |
| 7.5.4 | Initial estimates | 183 |
| 7.5.5 | Constants in the model | 184 |
| 7.5.6 | Weighting of observations | 185 |
| 7.5.7 | Estimation criteria | 187 |
| 7.5.8 | Residuals and predicted values | 188 |
| 7.5.9 | Maximum likelihood estimates | 189 |
| 7.5.10 | Special applications | 191 |
| 7.6 | Management and analysis of multiway tables | 197 |
| 7.6.1 | Structure of multiway tables | 197 |
| 7.6.2 | Modifications and transformations | 198 |
| 7.6.3 | Log-linear models and analysis of variance | 202 |
| 7.7 | Generalized linear models | 206 |
| 7.8 | Multivariate analysis | 209 |
| 7.8.1 | Matrix files | 209 |
| 7.8.2 | Linear combinations of variables | 213 |
| 7.8.3 | MATRUN operations | 216 |
| 7.8.4 | Principal components | 218 |
| 7.8.5 | Factor analysis | 220 |
| 7.8.6 | Canonical correlations and variables | 228 |
| 7.8.7 | Multiple discriminant analysis | 230 |
| 7.8.8 | Cluster analysis | 231 |
| 7.9 | Time series analysis | 234 |
| 7.9.1 | SER operations | 234 |
| 7.9.2 | Auto- and crosscorrelations | 237 |
| 7.9.3 | FORECAST operation | 238 |
| 8. | Graphics | 241 |
| 8.1 | General specifications | 243 |
| 8.2 | Frames, texts and lines | 247 |
| 8.3 | Bar and pie charts | 250 |
| 8.4 | Scatter diagrams and plots of time series | 256 |
| 8.4.1 | Common specifications | 257 |
| 8.4.2 | Contour ellipses and trend lines | 264 |
| 8.4.3 | Time series | 266 |

| | | |
|------------|---|------------|
| 8.4.4 | Multiple time series | 268 |
| 8.4.5 | Normal probability plots | 269 |
| 8.5 | Curves and families of curves | 271 |
| 8.5.1 | Simple curves | 271 |
| 8.5.2 | Parametric representation | 272 |
| 8.5.3 | Families of curves | 274 |
| 8.5.4* | Data values as varying parameters | 276 |
| 8.5.5 | Integral functions | 284 |
| 8.5.6 | Specifications in curve plotting | 284 |
| 8.6 | Contour plots | 287 |
| 8.7 | Plots of multivariate data | 291 |
| 8.7.1 | Enhancements in standard plots | 291 |
| 8.7.2 | Data matrix as a raster image | 294 |
| 8.7.3 | Draftsman's display | 297 |
| 8.7.4 | Chernoff's faces | 298 |
| 8.7.5 | Andrews' function plots | 301 |
| 8.7.6 | Profile and star symbol plots | 303 |
| 8.8* | Device-dependent features | 306 |
| 8.9 | Plotting on a PostScript printer | 308 |
| 8.10 | Plotting on the screen | 311 |
| 8.11 | Graphics examples | 313 |
| 9. | Printing of reports | 337 |
| 9.1 | Text printing | 338 |
| 9.2 | Control lines and words | 339 |
| 9.3 | Control and shadow characters | 340 |
| 9.4 | Multipage documents | 345 |
| 9.5 | Pictures | 346 |
| 9.6 | Justifying the right edge of the text | 348 |
| 9.7 | Footnotes | 349 |
| 9.8 | Automatic index | 350 |
| 9.9 | Tables | 350 |
| 9.10 | Box graphics | 354 |
| 9.11* | Mathematical expressions | 356 |
| 9.12* | PostScript code in the PRINT list | 358 |
| 10. | Matrix interpreter | 360 |
| 10.1 | Matrices in the edit field | 361 |
| 10.2 | Moving data to matrix files | 364 |
| 10.3 | MAT operations | 365 |
| 10.4 | Examples | 372 |
| 10.4.1 | Matrix inversion | 372 |
| 10.4.2 | Correlation matrix | 373 |
| 10.4.3 | Partial correlations | 375 |
| 10.4.4 | Linear regression analysis by orthogonalization | 377 |

| | | |
|-------------------|---------------------------------------|------------|
| 10.4.5 | Principal components | 379 |
| 10.4.6 | Canonical correlations | 380 |
| 10.4.7 | Multiple discriminant analysis | 382 |
| 10.5 | MATRUN operations | 384 |
| 10.6 | Partitioned matrices | 386 |
| 10.7 | Operations on polynomials | 388 |
| 10.8 | Linear programming | 390 |
| 11. | Control operations | 393 |
| 11.1 | Changing dimensions of the edit field | 393 |
| 11.2 | Moving the cursor in the edit field | 393 |
| 11.3 | Time control | 394 |
| 11.4 | Selecting the data disk | 394 |
| 11.5* | Code conversions | 395 |
| 11.6 | Changing the inquiry system | 396 |
| 11.7 | Screen colors | 396 |
| 11.8 | Calling other programs from Survo | 397 |
| 11.9 | MS-DOS commands | 397 |
| 11.10 | DIR command | 398 |
| 12. | Sucros | 399 |
| 12.1 | Using ready-made sucros | 400 |
| 12.2 | Making a sucro: an example | 402 |
| 12.3 | Code words of the sucro language | 406 |
| 12.4 | Sucro memory | 409 |
| 12.5 | Probe statements | 414 |
| 12.6 | Activation of sucros | 418 |
| 12.7 | Conditional statements | 420 |
| 12.8 | User interaction | 423 |
| 12.9 | Arithmetics | 427 |
| 12.10 | Sucro files | 430 |
| 12.11 | Types of sucros | 432 |
| 12.12 | Error control in sucros | 441 |
| Appendices | | |
| 1. | Keys and their functions | 444 |
| 2. | Survo system parameters | 451 |
| 3. | Sucro codes and statements | 459 |
| 4. | Installation of SURVO 84C | 465 |
| 5. | Survo keywords | 471 |
| References | | 483 |
| INDEX | | 485 |

1. Introduction

Survo is an integrated interactive system for statistical analysis, computing, graphics and report generating. It also includes unique features related to spreadsheet computing, matrix algebra and computer aided teaching. It provides tools for making of application programs in various special areas. All functions of Survo are based on the **editorial approach** developed by the author in 1979. The center of the activities in Survo is an **edit field** that at all times is partially visible on the screen. The edit field is maintained by the **Survo Editor**.

The user works with Survo by typing text in the edit field and by activating various operations and commands written among the text. In many applications, it is convenient to create **work schemes** including several extra **specifications**, also written in the text and in arbitrary order.

The data and the results of various operations and application schemes (like plotting schemes and matrix programs) are displayed in the same edit field when required. For more extensive data sets and tables of results, Survo provides its own file representations. Survo also communicates with text (ASCII) files.

From the user's point of view, Survo is one huge program which is controlled along certain general principles. Technically, however, Survo is a collection of several independent programs (modules) which are called by the Survo editor according to the user's activations. The user hardly notices the shifting of programs, but sees the system as one integrated environment without any need to know its internal structure.

As a collection of programs, Survo is open for additional modules made by experienced users according to certain rules. These rules and different tools for making modules are described in a separate document "*Programming SURVO 84 in C*" (Mustonen 1989). After a new module has been programmed and compiled, the commands and operations defined in it can be used as any standard Survo operations.

The open structure of Survo allows calling any other program and using it while staying in Survo. After finishing the job with the other program, the current Survo session will be continued again. Because the commands of the operating system can also be employed in this way, Survo can be considered an extension of the operating system.

The Survo system may be compared to any extensive text processing program. However, when using Survo as a word processor, the other activities are readily available, too.

Survo is also a tool for making new application programs. It provides several ready-made structures and user-friendly "languages" for such tasks. The Survo **matrix interpreter** and working modes like **tutorial** and **touch mode** are examples of such an approach. The tutorial mode permits recording of Survo

sessions as **sucros** (Survo macros). A special sucro language for writing sucros for teaching and expert applications has been developed. This is reported in "*Sucros in SURVO 84C*" (Mustonen 1988).

To a great extent, Survo is a selfcontained system providing different working modes needed e.g. in statistical research and planning. Naturally it cannot do everything, but it will be continuously extended to new areas of application.

Basically Survo is intended for professional users, but it is an easy system even for a beginner since everything is based on simple text editing. Speaking about "ease" in this context may be misleading. If a system is made easy and friendly just for a beginner, after a short learning period it may turn out to be very frustrating for a user who already knows its characteristics.

A good system should be like a musical instrument that requires a lot from its player before yielding its best. If, for example, the violin were invented in recent days, many people would object to its poor "user interface". However, the violin is far more advanced than mechanical, simple musical instruments since it gives scope for true skills and even for virtuosity.

If one knows the main ideas and working methods of Survo, there is no need to read manuals and user's guides. The best and always up-to-date source of information is the system's own inquiry and help facility, which is readily available during any Survo session.

Another way to get acquainted with the system is to watch tutorials recorded during normal Survo sessions. One can produce such teaching programs on any topic during the work by turning on the **tutorial mode**. This permits saving of all actions selected by the user. Ready-made Survo work schemes have been collected on separate diskettes.

The purpose of this book is to provide basic information about Survo. Also advanced topics are treated to some extent. Those chapters are denoted by (*) and can be omitted on first reading. The only prerequisite is that the user is familiar with the fundamentals of the operating system MS-DOS and its standard utilities.

Finally, it should be emphasized that no information on paper can give a true picture of a dynamic system like Survo. Therefore it would be excellent if the reader could use and test the system while studying this text.

Background of the system

Many of the ideas and principles appearing in Survo have been adopted from the earlier versions. The first in line was SURVO 66 originated by the author in 1966 and implemented on Elliott 803. One explanation for the name SURVO is the word "survey", since the first Survo was primarily planned for analysis of survey data. It can also be derived from the Finnish verb "*survoa*" which means "*compress*". The SURVO 66 jobs were controlled by a simple command language. The original SURVO 66 was further developed at the University of Tampere and is now known by the name of SURVO/71.

In 1976, the first interactive version, SURVO 76 was initiated by the author. It was completed in 1984 by him and his research group (Department of Statistics, University of Helsinki). Originally, SURVO 76 was made in a conversational (menu-based) form. The editorial approach was introduced in 1979. SURVO 76 was running on the Wang 2200 minicomputer.

The work on SURVO 84 started in 1984 on the basis of SURVO 76 by using the interpretative Basic language. This was the first microcomputer version and could be run on the Wang PC only.

The current SURVO 84C system was originated in 1985. From the user's viewpoint it is much like SURVO 84 and it is also highly compatible with SURVO 76. But the latest version is far more efficient and it allows wider applications since it is programmed in the C language. SURVO 84C can be run on MS-DOS microcomputers.

From 1990 on, a reduced version SURVOS has been available. It is intended mainly for teaching applications in secondary and vocational schools.

Different versions

The current Survo system has two versions, the complete one, SURVO 84C and the reduced one, SURVOS. Any function in SURVOS is working similarly in SURVO 84C but not conversely. The SURVOS version is designed for teaching applications and does not include e.g. more sophisticated statistical methods and support for high-quality graphics and desktop publishing. It also limits the size of the data to be processed.

Both SURVO 84C and SURVOS have extended versions for local area networks.

This document describes properties of the standard SURVO 84C version. The functions supported in SURVOS are indicated in the list of Survo functions and keywords given as Appendix 5.



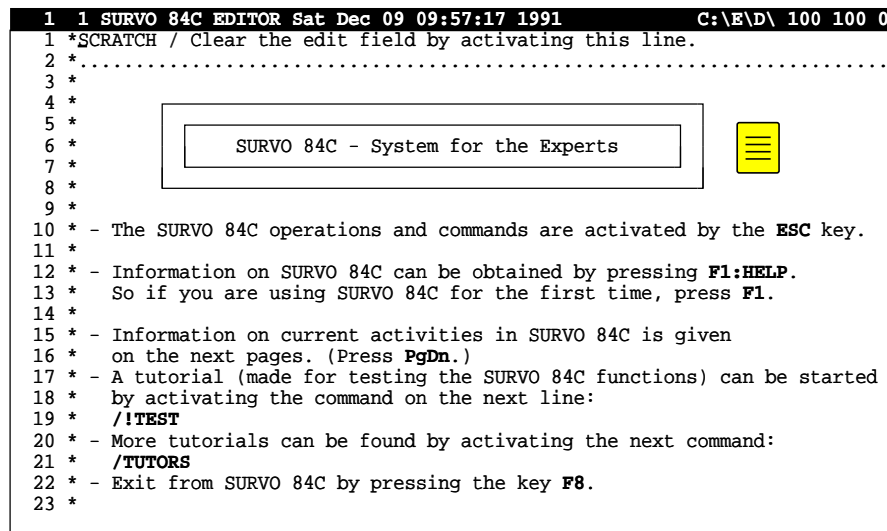
**The current version SURVO MM was initiated in 2000.
It works practically on all Windows platforms.**

SURVO MM?

2. An Overview of Survo

2.1 Entering Survo and exit from it

Survo is usually entered from the MS-DOS command level by the command **SURVO**. A Survo session will then be started and typically we will get the following display on the screen¹:



```

1 1 SURVO 84C EDITOR Sat Dec 09 09:57:17 1991 C:\E\D\ 100 100 0
1 *SCRATCH / Clear the edit field by activating this line.
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 * - The SURVO 84C operations and commands are activated by the ESC key.
11 *
12 * - Information on SURVO 84C can be obtained by pressing F1:HELP.
13 *   So if you are using SURVO 84C for the first time, press F1.
14 *
15 * - Information on current activities in SURVO 84C is given
16 *   on the next pages. (Press PgDn.)
17 * - A tutorial (made for testing the SURVO 84C functions) can be started
18 *   by activating the command on the next line:
19 *   /*!TEST
20 * - More tutorials can be found by activating the next command:
21 *   /*TUTORS
22 * - Exit from SURVO 84C by pressing the key F8.
23 *

```

Each user can have an entry point of his/her own. The default display above is suitable for a beginner who would like to get acquainted with the Survo system. An advanced user may create a more appropriate entry display (edit file **START**) leading directly to current applications. In general, the whole Survo environment (colors, various system parameters) may be tailored according to the needs of the user.


In any case, after the entry we are always working with the Survo Editor which is the basis for all functions. In this editorial mode, the cursor (initially blinking in the first location) is moved by the arrow keys, for example, and text can be typed on any line as with any text editor.

If you are using Survo for the first time, please, follow the instructions given in the entry display. It pays first to get familiar with the help system of Survo (instruction on line 12) and to run some of the tutorials (suggested on lines 17-21).

Exit from Survo always takes place by pressing the **F8** key.

¹ If Survo starts from a menu, this display (START field) is obtained from it as alternative 1.

2.2 Edit field



The work area for the Survo Editor is an *edit field* which is entirely located in the central memory of the computer. The edit field has typically 100 lines and 100 columns. Each line is preceded by a control symbol for special notations; this control symbol is always initially ‘*’ on each line. 

During the work, the user can maintain any number of edit fields. However, at a time only one of them is active (in central memory and partially visible on the screen). The others are in edit files on disks, but they can be scanned in a temporary window (by a `SHOW` operation) and/or loaded partially to the current edit field.

Usually one is working with various edit fields one after another by saving the current field after editing (`SAVE` operation) and loading another as a whole (`LOAD`). Thus it is simple to change the active field when necessary.

In its basic form, one edit field corresponds to about two pages of normal text. However, in no application it is necessary to identify one edit field with a page or two in some report. When printing documents consisting of several pages (like this text), the general `PRINT` operation of Survo will automatically take care of proper page division (obeying the wishes presented by the user, of course). When defining the printout, the user simply tells what are the edit fields and chapters in them that belong to the document.

The edit field (as well as the edit files) normally contains text and tables written by the user, various Survo operations (commands, work schemes etc.) and their results. Representation of various data structures in the same space formed by the edit field is essential and gives exciting possibilities for combining different activities in a creative manner.

When working with larger data sets, the space given in the edit field is not enough for the data itself. Although the dimensions of the edit field may be expanded (up to 600 lines with 100 columns, for example, by the `REDIM` command), it is not wise to create very large edit fields. For big data sets and tables Survo supplies special data files.  

Survo also supports some other data representations and permits information from other files to be loaded to the edit field. Even text files created by other systems can be processed in Survo by using operations like `SHOW` and `LOADP` first. Furthermore, Survo data and results can be copied to general text (ASCII) files by `SAVEP` and `FILE LOAD` operations, for example.

**REDIM?
RESIZE?
WINDOW?
FONT?**

2.3 Control of the edit field

If the first line (`SCRATCH`) is activated in the entry display, all the text will be erased and an empty edit field will appear as:


```

1 1 SURVO 84C EDITOR Sat Dec 06 17:18:01 1986 C:\E\D\ 100 100 0
1 *
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *

```

On the header line, some basic information is given like date and time, the data disk drive designation (C:\E\D\), and the size of the edit field ('100 100' means 100 lines and 100 columns).

The user can now start writing text as on a standard typewriter. The **ENTER** key moves the cursor to the next line. A new line is initialized even automatically when the visible line becomes full. Correspondingly, when the last visible line has been filled, the visible part of the edit field automatically scrolls upwards giving space for a new line at the bottom.


It is always possible to move the cursor in the field and in the text by the arrow keys or the **PgUp** (previous page) and **PgDn** (next page) keys. 

Simple editing takes place by typing over previous text and by using **INSERT** and **DELETE** keys for inserting and deleting texts, respectively. To make room for new empty lines between current text lines, the **LINE INS** (alt-INSERT) key has to be pressed. The **LINE DEL** (alt-DELETE) key deletes the current line entirely.

A complete description of the keys used in Survo operations is given in Appendix 1. When using Survo, the keyword **KEYS?** when activated also yields information on this topic.

In all editing functions, the foremost principle is to keep them as simple as possible. The best way to learn these basic actions is to practice them at the computer.

2.4 Operations and commands

Because Survo includes hundreds of activities that all are invoked from the editor, it would be too complicated to do everything by special keys and key combinations. Therefore, each more advanced operation is carried out by writing a command on any free line and by activating it by the **ESC** key. 

ACTIV?

Assume that we have written in the edit field:

```

10 1 SURVO 84C EDITOR Sat Dec 06 17:37:46 1986 C:\E\D\ 120 80 0
1 *
2 *This text has been written in the edit field, and we want
3 *to have a printout of it on the printer.
4 *It can be done by the command PRINT L1,L2 where L1 is the
5 *first and L2 the last edit line to be printed.
6 *
7 *PRINT 2,5_
8 *
9 *
10 *
11 *

```

Lines 2 to 5 will now be printed on the printer by the PRINT 2,5 command on line 7. To activate it, we move the cursor to line 7 (if it is not already there) and press the `ESC` key. When a line becomes activated, it is temporarily displayed in reversed video (as shown above).

Many commands and operations refer to edit lines (lines of the edit field) like 2 and 5 in the previous PRINT operation. Instead of line numbers, line labels (of one character like A,B,x,y,+,-) written in the control column can be used. The control column is originally filled with asterisks '*'. To reach it, move the cursor to the first column and keep the left arrow key pressed down long enough. In normal typing functions the control column remains intact.

Because the operations (commands) are written into the text as any other information, it is easy to edit them and activate again. Thus the commands do not disappear from the the screen (edit field) after they have been completed. As with any other text, the user can scratch the commands by using the `ctrl-END` key, for example, or overwrite them by text or other commands.

2.5 Work schemes

In demanding Survo applications, various commands and operations together with various extra **specifications** are written as **work schemes**. To some extent, these schemes resemble programs.

In a typical work scheme, not only the activated operation but also the specifications appearing in the edit field may have particular influence. Each operation has its own specification words. If a specification is not given in the edit field, when the operation (work scheme) is activated by `ESC`, a default value is automatically entered. By giving the specification, the user can change the default values.

When planning Survo operations and program modules, much attention has been paid to the selection of specifications and their default values. Good solutions, in this respect, lessen the burden of the user and make the system more intelligent. They form an essential part of the interface between the user and Survo.

All specifications have the form <specification word>=<values>. For example, when making pictures on the screen by a GPLLOT operation the size of the graph is controlled by the SIZE specification. SIZE=600,320 indicates

that the width of the graph will be 600 pixels and the height 320 pixels. On PostScript printers (operation PLOT with specification DEVICE=PS), the unit is 0.1 mm instead of pixels.

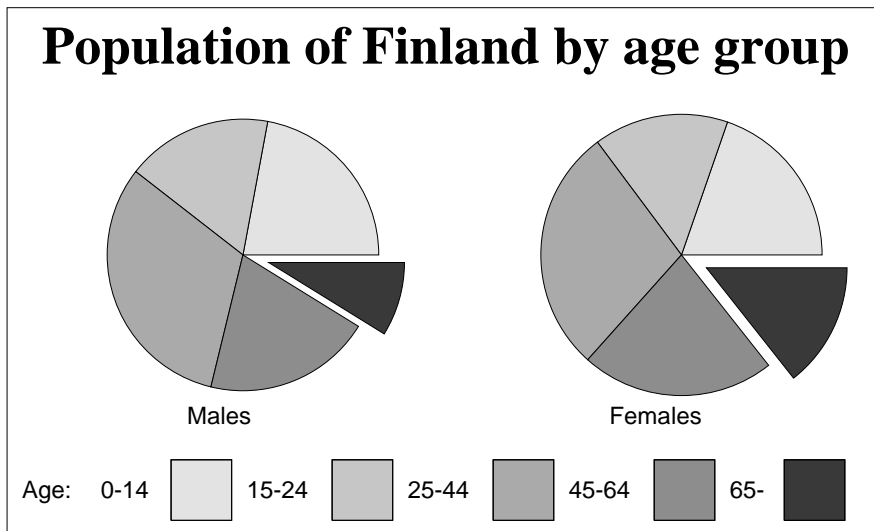
Below, a typical work scheme with the result is displayed. Note the free setting of specifications (no strict order) and the possibility to alter any detail in the scheme before a new activation.

```

14 1 SURVO 84C EDITOR Fri Dec 06 14:10:40 1991 C:\E\D\ 100 100 0
1 *
2 *
3 *   Population in Finland (1000)
4 *
5 *DATA FINLAND
6 *Sex      0-14 15-24 25-44 45-64 65-
7 *Males    506   399   727   458   202
8 *Females  484   381   693   547   353
9 *
10 *HEADER=[Times(20)],Population_of_Finland_by_age_group
11 *GPLOT FINLAND
12 *SIZE=600,320           size of the picture
13 *TYPE=PIE              pie chart
14 *LEGEND=Age:          text in front of the legend
15 *SHADING=1,2,3,4,7P  shadings (colors), P=pull out sector
16 *XDIV=0,1,0          no vertical margins
17 *
18 *
19 *
20 *
21 *
22 *
23 *

```

When the G~~PLOT~~ operation on line 11 is activated, the following graph is produced on the screen:



2.6* Subfields

Work schemes which are placed in the same edit field may disturb each other if they are using the same specifications in different ways. To avoid confusion, the schemes may be isolated from each other by typing a border line between them. A border line has the form `*` (i.e. ‘*’ in the control column followed by at least 10 dots). If the entire border line is filled (as is usual) with dots, the editor displays it as a thicker stripe that clearly reveals the boundary between the work schemes. All operations activated between these two border lines will adhere to specifications in this limited area only. The area is called a *subfield*.



BORDER?

In some cases it is desirable to have several work schemes with the same specifications. Those specifications have to be written in the first subfield of the current edit field, and this subfield must contain the keyword `*GLOBAL*` positioned anywhere in that subfield. When a work scheme in the edit field is activated, the specifications are primarily searched for in the local subfield and secondarily in the `*GLOBAL*` subfield. If neither local nor global specification is found, a default value is employed.



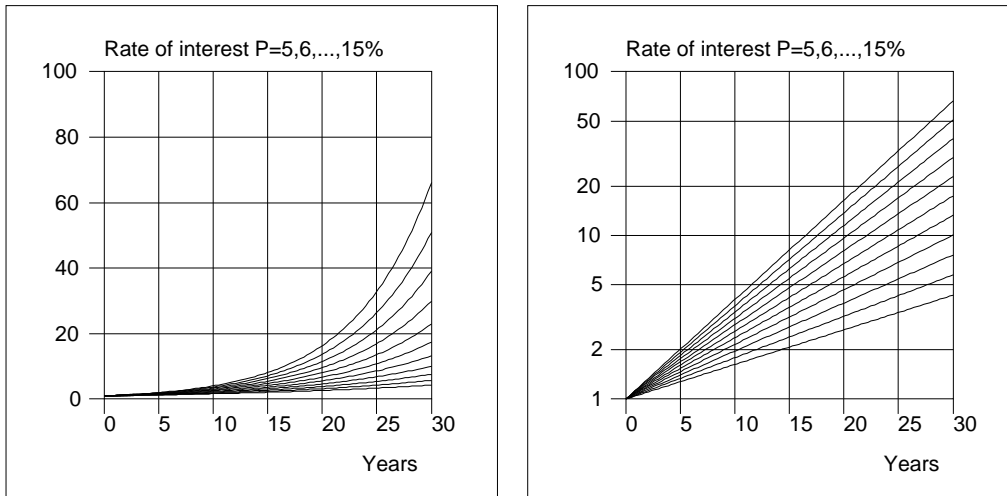
Below, two families of curves are plotted with their own PLOT schemes but using partially same specifications given in the `*GLOBAL*` subfield.

```

12  1 SURVO 84C EDITOR Fri Dec 06 16:13:43 1991      C:\E\D\ 100 100 0
1  *
2  *
3  *Compound interest on linear and logarithmic scales
4  * *GLOBAL* P=5,15,1 SIZE=650,650 PEN=[Swiss(8)]
5  *      GRID=XY HEADER=
6  *      XSCALE=0(5)30 XLABEL=Years
7  *      YLABEL=Rate_of_interest_P=5,6,...,15%
8  * .....
9  *PLOT Y(x)=(1+P/100)^x / DEVICE=PS,INTER1.PS
10 *YSCALE=0(20)100
11 * .....
12 *PLOT Y(x)=(1+P/100)^x / DEVICE=PS,INTER2.PS
13 *YSCALE=*log(y),1,2,5,10,20,50,100
14 * .....
15 *
16 *PRINT 17,18
17 - picture INTER1.PS,*-90,100
18 - picture INTER2.PS,*+600,100
19 *
20 *
21 *
22 *
23 *

```

The PLOT schemes on lines 9 and 12 save their results in PostScript files INTER1.PS and INTER2.PS, respectively. The PRINT operation on line 16 yields the following pair of graphs:



Since the commands, work schemes and other pieces of control information reside in the edit field and are easy to edit, there is no need to abbreviate keywords as happens in many command languages. By using various copying and editing facilities offered by the editor, it is possible to avoid writing the same words over and over. It pays to learn to avoid unnecessary work and to use old material for new applications. Furthermore, by saving pertinent edit fields one may create work schemes (or programs) that form a basis for new altered and improved applications.

2.7* Operation sequences

A series of operations and work schemes can be activated with one touch by placing the operations needed on consecutive edit lines. If the first operation is activated by pressing `[F2:PREFIX]` and then `[ESC]`, then after the execution of the first operation the cursor will automatically be moved to the next line and the operation on that line is activated. The operations are carried out as long as there are feasible operation lines. Usually an empty line is used to interrupt the operation sequence.

It is not necessary that the operations belonging to the operation sequence are written on consecutive lines, since jumps to other lines may be performed by a `GOTO` command. For example, `GOTO 24,30,15` changes the display in the edit field so that edit line 24 will be the first visible one and the cursor will be located on the 15th position of the 30th line. Thus any place in the current edit field can be reached during a sequence of operations.

Likewise jumps to other edit fields may be done by using a `LOAD` command. For example, `LOAD PART2,1,10` replaces the current edit field by another (`PART2`) from the data disk and places the cursor there on the 10th line so that the first edit line is the first visible line.

The next display shows a operation sequence that saves a 3x4 matrix **A** in a matrix file, computes **AA'** and its inverse matrix and finally displays the result in the edit field.

```

1 1 SURVO 84C EDITOR Sun Dec 07 14:26:33 1986 C:\E\D\ 100 100 0
24 *
25 *Computing the inverse matrix of AA'
26 *
27 *MATRIX A ///
28 *      12.5   4.2  11.0  -8.1
29 *      0.0   -1.3  5.6   2.4
30 *      5.2  10.4  -9.3   0.3
31 *
32 *MAT SAVE A
33 *MAT B=MMT(A) / *B~A*A' S3*3
34 *MAT B=INV(B) / *B~INV(A*A') 3*3
35 *MAT LOAD B,37
36 *
37 *MATRIX B
38 *INV(A*A')
39 *///
40 *      1      2      3
41 * 1      0.00347 -0.00662 -0.00200
42 * 2      -0.00662 0.06308 0.01857
43 * 3      -0.00200 0.01857 0.00998
44 *
45 *
46 *

```

Our operation sequence is on edit lines 32-35 and contains **MAT** operations only. The situation after the completion of the sequence is displayed. The changes and results due to the operations are indicated in a gray shading. The parameter 37 of the **MAT LOAD** operation on line 35 refers to the first line of the results.

The automatic execution of an operation sequence may be halted by pressing (i.e. a full stop). Since the same key is used for interrupts in longer individual operations (like **PLOT**), it is better to stop a sequence of operations by pressing twice and to wait for a moment.

2.8* Tutorial mode and sucros

In addition to operation sequences, Survo provides a possibility of recording sessions in a tutorial mode. In this mode, all the actions (key touches) of the user will be saved in a special tutorial file (with name extension **.TUT**). When the saving stage is over, the whole task may be repeated automatically. The original key touches are simulated by the tutorial file and it acts like a ghost-user carrying out all the actions exactly as they were initially performed.

The tutorials can be edited afterwards by loading the contents of the tutorial file in the edit field by a **TUTLOAD** command and by saving it after modifications by a **TUTSAVE** command. Time delays, displaying of key touches, prompts, etc. can be added in this way.

Since tutorial mode is available in all Survo actions, it offers a unique possibility for teaching various things, not only related to Survo operations and

methods of use in general but also for any potential application of Survo. It would be most valuable that advanced Survo users could demonstrate various working methods by saving their Survo sessions as tutorials.

The normal Survo installation includes a separate section of tutorials which are readily available from the standard entry display.

In addition to teaching programs, also various expert applications are saved in tutorial mode. Such applications are called sucros (Survo macros). The most advanced sucros are programmed by writing them in the sucro language. The TUTSAVE and TUTLOAD commands are used for interpreting code written in the sucro language.

For example, typing of text "Survo system" is defined as a sucro 1 as follows:

```

10 1 SURVO 84C EDITOR Sat Dec 07 16:35:06 1991 C:\E\D\ 100 100 0
1 *
2 *TUTSAVE 1
3 *Survo system{end}
4 *

```

By activating the command TUTSAVE 1 on line 2, a sucro file 1.TUT is created on the current Survo data disk/path and the sucro code until the code word {end} is saved in it. Thereafter the sucro 1 called either by a command /1 (a slash precedes the sucro name) or in the case of a one-character name simply by the key combination **F2** **M** **1** where the last key corresponds to the name. Hence, we are able to type the text "Survo system" in the edit field whenever needed by pressing those three keys.

This example represented a sucro as its simplest form. In addition to plain text typing, a typical sucro contains various control codes and statements for conditional processing and user interaction. The range of sucros is wide from elementary macros to large teaching programs and expert applications.

2.9 Inquiry system

The inquiry system is an essential part of Survo and takes about a quarter of the space required for Survo on the system disk. The inquiry system is also continuously maintained by the Survo Editor because files belonging to it are ordinary edit files. Thus when the Survo system is revised and extended, all changes are also recorded in the inquiry system. Hence, it provides always fresh information to the user.

An inquiry may be initiated, when Survo is in use, either by pressing the **HELP** key (F1) or by activating (by **ESC**) a keyword with a question mark (e.g. PRINT?).

If **HELP** is pressed, the inquiry system usually tells about the operation last attempted. If a keyword with a '?' is activated, information concerning it or some related topic will be given. If several alternatives have the same forefront as the keyword, the inquiry system will list them and allows the user to have a choice. For example, inquiry (command) A? lists all keywords beginning with

'A' and ? alone gives all possible keywords. During an inquiry, the keyword may be changed by pressing [?] and entering a new one.

The inquiry system can also be activated as a hypertext. By indicating any word by the cursor and by pressing keys [F2:PREFIX] and [F1:HELP] information on this word is displayed. While staying in the help mode, the cursor can be moved to indicate any word in the help text (or even in the field) and more information on the new word is obtained by pressing the [ESC] key or [F2] [F1] .

Often the inquiry can be started without any particular idea by using some general term (like EDITOR? for example). Then it is possible to clarify the intention by going through menus displayed on the screen, step by step. In this sense, Survo is strongly menu-oriented. In case of erroneous selections it is possible to undo them by pressing [BACKSPACE] (the system remembers the 100 last steps). Exit from inquiries takes always place by pressing [ENTER] .

The inquiry system displays information in a temporary window and nothing will change in the edit field. Thus the work after a query may be continued without any harm. The user may, however, copy the texts shown by the inquiry system to the edit field by pressing key [+]. This book contains many sections adopted from the inquiry system in a revised form.

Often an inquiry is related to some Survo operation and its structure. When now the inquiry is stopped by [ENTER], a model for the operation which was last commented will be printed on the inquiry line if there is free space. This model will serve as a basis for the application.

For instance, inquiry SHOW? produces the following display:

```

6 1 SURVO 84C EDITOR Sun Dec 07 17:58:02 1991 C:\EVD\ 100 100 0
24 * Often an inquiry is related to some Survo operation and its structur
25 *now the inquiry is stopped by ENTER, a model for the operation, which wa
26 *last commented, will be printed on the inquiry line if there is free spa
27 *model will serve as the basis for the application.
28 *
29 *For instance, inquiry SHOW? produces the following display:
30 *
31 *SHOW?_
-----
SHOW <file name>,<first line to be shown (optional)>
shows portions of edit files and text files in a temporary window
below the current line (SHOW line). The text may be scrolled in that
window and selected lines may be copied in the edit field from
the line below the SHOW line onwards, provided that those lines are
empty.

SHOW without any parameters displays the contents of the current out-
put file set by the OUTPUT operation.
SHOW * displays the current edit field itself.
Press ENTER!

```

If **ENTER** is now pressed, the system returns to the normal editing mode and the inquiry line (31) changes to:

```

7 1 SURVO 84C EDITOR Sun Dec 07 17:58:08 1991 C:\E\D\ 100 100 0
24 * Often an inquiry is related to some Survo operation and its structur
25 *now the inquiry is stopped by ENTER, a model for the operation, which wa
26 *last commented, will be printed on the inquiry line if there is free spa
27 *model will serve as the basis for the application.
28 *
29 *For instance, inquiry SHOW? produces the following display:
30 *
31 *SHOW <file name>,<first line to be shown (optional)>
32 *

```

Even an experienced user needs the services of the inquiry system continuously. Survo covers so many different activities that it is not reasonable to try to remember them all. The inquiry system helps in recalling the necessary information.

Since it is assumed that the reader has access to Survo, in many places in this text we are referring to information which can be found through inquiries. In those cases, a suitable keyword will be given in the form (CHANGE?) and the reader is advised to consult the inquiry system.

2.10* Calling other programs and the operating system

As a family of programs, Survo allows other programs (not belonging to it) to be run while staying in Survo. This can be done by using the **CHILD** command of Survo. For example, **CHILD COMMAND** invokes the command interpreter of the MS-DOS operating system and **CHILD** without any parameters calls another copy of Survo itself. The other programs which are called in this way can be run, if the operating system can find them (a suitable MS-DOS **PATH** command has been given) and there is enough free memory (after MS-DOS, Survo Editor and eventual resident programs). The Survo Editor with the standard 100*100 edit field needs about 160 KB of the CPU memory.

The MS-DOS commands can also be given directly from Survo by typing them in the edit field like any Survo command but inserting a prompt symbol '>' in front of the command.

For example, the MS-DOS command **VER** will give the following display when activated in Survo:



LINKS?

F1?

```

4 1 SURVO 84C EDITOR Tue Dec 07 18:05:53 1991 C:\E\D\ 100 100 0
20 *
21 *   The MS-DOS commands can also be given directly from Survo by typing
22 *them in the edit field like any Survo command but inserting a prompt
23 *symbol '>' in front of the command.
24 *   For example, the MS-DOS command VER will give the following display
25 *when activated in Survo:
26 *
27 *
28 *>VER
-----
IBM DOS Version 5.00
Press any key!

```

After pressing any key, the normal Survo display will be resumed.

2.11 List of operations

The current version of SURVO 84C covers over 200 different operations (commands) and other actions. The list below is only a general account of those activities and does not give any details. Typical keywords and operation names are given in parentheses. Call the inquiry system with these keywords for detailed information. The functions indicated by '*' are optional and do not belong to the standard version of SURVO 84C.

1. Control operations

- Redimensioning of the edit field (REDIM, INIT)
- Selecting the data disk (DISK)
- Selecting the output device/file (OUTPUT)
- Code conversions (CONVERT, CODES)
- Calling other programs/systems from Survo (CHILD)
- General file management (by MS-DOS commands)
- Time (TIME, WAIT)
- Moving the cursor in operation sequences (GOTO)
- Changing the system parameters (SETUP)
- Screen colors (COLOR)
- Changing the inquiry system (QPATH)
- Listing directory entries (DIR)

2. Text and table management

- Clearing the edit field (CLEAR, SCRATCH, ERASE)
- Text typing, editing and saving/loading
(Function keys, SAVE, LOAD)
- Text management
(INSERT, DELETE, TRIM, MOVE, COPY, CHANGE, SHOW, and keys like F4:BLOCK and F2:WORDS)
- Searches in edit fields (FIND, REPLACE)
- Moving data between text files and the edit field
(LOADP, SAVEP, SHOW)
- Table management (FORM, SORT, SET, COUNT)
- Table arithmetics (C ja L operations;
see also mathematical operations!)

- Report management and printing (PRINT)
- Desktop publishing (PRINT, POSTSCRIPT)
- Hypertext applications (HYPER)

3. Data file management (FILE operations)

- Creating a data file (FILE CREATE)
- Data activation and protection
(FILE ACTIVATE, FILE_ACT key)
- Data saving and editing by filling a form on the screen
(FILE EDIT, FILE SHOW)
- Searches in data files (FILE EDIT, FILE SHOW)
- Loading the structure of the data file to the edit field
(FILE STATUS)
- Updating the data file structure (FILE UPDATE)
- Copying parts of data files to the edit field and to text
files or to the printer (FILE LOAD)
- Moving tables in text files to data files (FILE SAVE)
- Moving a data file (or a data set in the edit field)
to another data file (FILE COPY)
- Data sorting (FILE SORT)
- Aggregation of observations (FILE AGGRE)
- Data transformations by formulas and rules given by the
user (VAR, CLASSIFY)
- Generating data by simulation (VAR)
- *Data saving and editing by user-defined forms (FEDIT by *M.Korhonen*)
- *Combining several related Survo data files (MFCOPY by *M.K.*)
- *Transforming multiple responses (MRESP by *M.K.*)

4. Statistical computing and analysis (STATIS)

- Variable transformations (VAR)
- Standardized and normalized variables (VAR)
- Simulated data (VAR)
- Conditional processing (VARS, MASK, IND, CASES, SELECT)
- Scale type checking (SCALES)
- Basic statistics and univariate summaries (STAT)
mean, standard deviation, skewness, kurtosis,
geometric, harmonic, and power means,
order statistics, autocorrelation, entropy,
frequency distribution with automatic classification
- Means, standard deviations and correlations (CORR)
- Frequency distributions, histograms and fitting univariate
distributions (standard and user-defined), Chi²-test
(HISTO)
- Multiway tables of frequencies, means and standard
deviations (TAB)
- Editing of multiway tables (TAB operations)
- Log-linear models (TABFIT)
- Constructing multiway tables for frequencies, sums, means,
and other descriptive statistics (MTAB by *M.Korhonen*)
- Sample statistics and comparison tests (COMPARE)
ex. t test, F test, Mann-Whitney, Kruskal-Wallis,
Wilcoxon
rank correlations (Spearman, Kendall)
tests for normality (Shapiro-Wilk, D'Agostino etc.)
Fisher's randomization principle applied by simulation
- Linear regression analysis (LINREG)
- Nonlinear regression analysis (ESTIMATE)
- Nonlinear estimation (ESTIMATE)
- Regression diagnostics (REGDIAG)
- Semiparametric data smoothing (SMOOTH)
- Multivariate analysis by the matrix interpreter
- Generalized linear models (GENREG)
- Canonical analysis (CANON)
- Linear combinations of variables (LINCO)
- Factor analysis (FACTA)

- Rotation in factor analysis (ROTATE)
- Semiparametric data smoothing (SMOOTH)
- Moving averages, differences, cumulative sums (SER)
- Auto- and cross-correlations (XCORR)
- Time series forecasting (FORECAST)
- *ARMA and SARMA models by means of the Kalman filter (by *J.Boucelham*)
- *Multiple comparisons of means, general analysis of variance and covariance (ANOVA by *M.Korhonen*)
- *Multivariate analysis of variance and covariance, generalized repeated measurements and multiple comparisons of means (ANOVA by *M.K.*)
- *Estimating missing values of data (PAD by *L.Sadeniemi*)
- *Estimating correlations in the case of missing values (CORRM,CORRML by *L.Sadeniemi*)
- *Discriminant analysis and classification of data (DISCR by *M.Korhonen*)

5. Graphics (PLOT, GPLOT)

- Bar charts (9 different types)
- Pie charts
- Histograms (HISTO, GHISTO)
- Correlation diagrams
- Time series, line graphs
- Scale transformations, probability plots
- Analytic curves, families of curves
- Integral functions
- Contour plots
- Matrix diagrams
- Andrews' function plots
- Chernoff's faces
- Star and profile symbol plots
- Draftsman's displays (simultaneous scatter plots)

6. Mathematical operations (MATH)

- Editorial arithmetics
- Arithmetics in touch mode
- Functions related to probability and statistics
- Spreadsheet computing (C,L operations, touch mode)
- Conversion between measurement units, number systems, currencies, etc.
- Operations on polynomials with real and complex coefficients, roots of algebraic equations (POL operations)
- Symbolic derivatives of functions (DER)
- Linear programming (SIMPLEX)

7. Matrix interpreter (MAT operations)

- Saving matrices to matrix files (MAT SAVE)
- Loading matrix files to the edit field (MAT LOAD, LOADM)
- Basic arithmetics with matrices (+,-,*,' ,INV etc.)
- Normalizations
- Column sums, sums of squares etc.
- Element by element transformations
- Scalars in matrix operations
- Matrix decompositions (Cholesky, Gram-Schmidt, eigenvalues and -vectors, singular values)
- Linear equations (MAT SOLVE)
- Least squares problems (MAT SOLVE)
- Partitioned matrices
- Super matrices (+,-,*,' ,INV)
- Automatic control for matrix names, column and row labels
- Matrix programs

8. Teaching and user support

- Inquiry system (HELP)
- Sucros, tutorial mode (TUTOR, TUTSAVE, TUTLOAD)

3. Text processing

Since the role of the Survo Editor is prominent in all working functions related to standard word processing, the editor should provide the same facilities which are typical for specialized text processing systems.

The basic functions needed for simple text editing were already discussed when describing the edit field and its control. If one is able to use a standard typewriter, there should be no difficulties in writing text and making small corrections in it. In Appendix 1, detailed descriptions of various keys used in text editing are given. See also KEYS?.

Useful keys in these activities are:

| | |
|--------------------|--------------------------------------|
| INSERT | Toggle insert mode |
| DELETE | Delete one character |
| ERASE (ctrl-END) | Erase colors/text to the end of line |
| HOME | Cursor to start of line/page/field |
| NEXT (PgDn) | Next page |
| PREV (PgUp) | Previous page |
| DISK (F4) | Select data disk drive/path |
| FORMAT (F5) | Select display attributes, colors |
| MERGE (F6) | Merge/split lines |
| REF (F7) | Set/recall a reference point |
| WORDS (alt-F2) | Move words |
| COPY LINE (alt-F3) | Copy a line |
| BLOCK (alt-F4) | Move a rectangular block (chapter) |
| SEARCH (alt-F5) | Search for a string |
| CODE (alt-F7) | Select/type a special character |
| LINE INS (alt-F9) | Insert a line |
| LINE DEL (alt-F10) | Delete a line |

Some of these keys will be described more thoroughly in later chapters. Next we shall tell about Survo commands that are used for more demanding manipulation of textual data and which also control large reports and printouts of them. It should be noted that many expedients useful in text processing are portrayed in later chapters. For instance, many operations for table management are helpful in word processing, too.

Some tasks related to text processing can be carried out by key sucros. For example, the key sequence **F2:PREFIX** **M** **X** changes the location of the current word with the next one on the line. A list of sucro tools for text editing as well as for other purposes is obtained by activating the sucro command /SUCROS (see also Appendix 1).

3.1 General principles

At first it is good to make clear, how to proceed, when a report including several pages should be produced as a part of a Survo job.

Although it is possible to keep two or three pages of typical text in one edit field, everything cannot be held in one place. Thus one is normally working with several edit fields and it is wise to label them consistently, for example by indexing them like TEXT1, TEXT2, TEXT3, etc.

It is best to start with a SAVE command on the first edit line. For example, the first edit field for this chapter could begin as follows:

```

11  1 SURVO 84C EDITOR Sun Dec 15 17:09:06 1991      D:\P2\TEXT\ 150 100 0
1  *SAVE TEXT1_
2  *LOAD TEXT2
3  *
4  *Survo - Interactive Environment
5  *DEF A,6,END
6  *
7  *
8  (3.  Text processing
9  *
10 *      ince the role of the Survo Editor is prominent in all working func
11 - [big_letter(S)]
12 *      related to standard word processing. The editor should provide the
13 )facilities which are typical for specialized text processing systems.
14 *      The basic functions needed for simple text editing were already disc
15 *when describing the edit field and its control. If one is able to use a
16 *typewriter, there should be no difficulties in writing text and making s
17 *corrections in it. In Appendix 1, detailed descriptions of various keys
18 *text editing are given. See also KEYS?.
19 (      Useful keys in these activities are:
20 *
21 T              T
22 *  INSERT      Toggle insert mode
23 *  DELETE      Delete one character

```

On line 1, we have a SAVE command which can be activated by **ESC** and it saves the entire edit field to the edit file TEXT1 on the data path D:\P2\TEXT\ . The selected data disk/path designation appears on the header line of the screen and it is changed by a DISK <path> command when necessary.

The LOAD TEXT2 command on line 2 erases the current contents of the edit field and loads the edit file TEXT2 to the edit field. This simplifies scanning through reports consisting of several edit fields.

Nothing obliges the user to place SAVE and LOAD commands just on these lines. Other conventions are possible, too. It is, however, always easy to reach the first lines, for example, by pressing the **HOME** key a couple of times. Scanning through fields is very convenient in this way.

It is also worthwhile to observe the DEF A,6,END definition on line 5. It specifies the lines from 6 to the last written line (END) in the current edit field as a chapter labelled by A.

When dealing with reports and lists including many edit fields on disk, it is possible to refer to parts of the report by using names of edit files and chapters in them.


For example, the whole text starting from line 6 in this edit field is called by a control line

- chapter A in TEXT1

in the PRINT operation which is used for the printing of this paper.

This naming convention permits the same chapter to appear in several documents. It may also occur many times in the same document. Because there is no obstacle to defining several chapters in the same field, some of them may be parts of others.

The most common situation is, however, the one described above; excluding a few first lines, the whole text in the field forms one chapter, and it can always be labelled with A, for example.


The initial size of the edit field is usually 100 lines and 100 columns plus the control column. These dimensions may be altered by a REDIM command. For example, 

```
REDIM 120,80
```


sets the number of lines to 120 and number of columns to 80. REDIM gives an error message if the current text exceeds the new limits.

If various display effects (colors, blinking etc.) are used, a special *shadow (attribute)* line is created for each edit line having such attributes. The default maximum number of shadow lines is 20. It can be altered by an extra parameter in the REDIM operation. For example REDIM 120,80,40 sets the maximum number of shadow lines in the current edit field to 40.

A faster tool for redimensioning is the INIT operation with equivalent parameters but it clears the entire edit field. INIT is mostly employed for initialization of the work space.

The maximum size of an edit field is 64KB (i.e. 65536 characters) with the control column and shadow lines. Thus a field with 600 lines and 100 columns may be maintained. In general, it is rational to avoid very large edit fields. 

The maximum number of columns is 252, but it is not usually reasonable to extend it from the default value 100. It is also sensible to avoid too narrow fields because it makes text management more difficult. It is profitable to keep a margin of 10-20 characters at the end of edit lines.

The number of edit lines visible on the screen is 23. In VGA monitors, this number can be extended to 48 by activating the *sucro* command /48. The original number is resumed by the command /23. 

A 100x100 edit field is usually wide enough if one selects a right approach for the task at hand. Only very large (broad) tables may offer some problems. A typical solution is to save them in Survo data files (FILE?) or in matrix files (MAT?) and pick them piecewise into the edit field (by FILE LOAD, MAT LOAD or LOADM) when needed for reporting etc.

When text and tables are written in the edit field or loaded to the field, it is not usually good to fill it completely, but at least the last 20 lines or so should be left empty for later additions. Of course, the number of lines can easily be increased by REDIM. If, however, the edit field TEXT1, for example, is

completely occupied and TEXT2 is already in use, it is natural to write additional text to a new edit field, say TEXT1B. Now when dealing with the document in question, it can be defined to consist of chapters A in edit files TEXT1, TEXT1B, TEXT2, etc. in this order.

For a general description of a document, a separate edit field is usually reserved. In our example it could be called TEXT. When the files are on diskettes, the INDEX file is also suitable for this purpose. [It is good to create one edit file INDEX on each diskette to keep track of all other Survo files.](#)

3.2* Printing a complete document

The TEXT field controlling the printout of this chapter could have a structure like this:

```

14 1 SURVO 84C EDITOR Wed Dec 25 14:38:35 1991      D:\P2\TEXT\ 100 100 0
1  *SAVE TEXT
2  *LOAD TEXT1
3  *
4  - [PSAVE][M][Times(9)]
5  *?????          Text processing          ###
6  - [PRESTORE]
7  *
8  - [PSAVE][M][Times(9)]
9  *###           Survo - Integrated Environment  ?????
10 - [PRESTORE]
11 *
12 *
13 *PRINT CUR+1,END
14 - include STYLE.DV2
15 - [page_number(18)][M]
16 - replace ?????
17 - header_lines 4,7,8,11
18 - chapter A in TEXT1
19 - chapter A in TEXT2
20 - chapter A in TEXT3
21 - chapter A in TEXT4
22 - chapter A in TEXT5
23 - chapter A in TEXT6

```

On line 13, we have a PRINT operation which prints this document consisting of chapters A in edit files TEXT1, TEXT2, ... For a detailed description of the PRINT operation, see PRINT?.

In this context we make only some remarks on the example above.

The PRINT command on line 13 has parameters CUR+1,END which are the first and last edit line, respectively, to be processed by the Survo PRINT module when line 13 is activated. CUR is the index of the current cursor line. Thus in this case CUR+1 takes the value 14. END refers to the last non-empty line of the field.

Before going through the lines 14-END, a special text file, the printing *device driver* (in this case the standard PostScript driver PS.DEV of Survo) will be processed first. This file gives all basic information to the PRINT module about the printer in question.

Only after this 'learning period' will the lines (14-END) be processed.

The lines with a ‘*’ in the control column will be printed as such while lines with a ‘-’ in the control column hold various control information only.

The first line in the PRINT list above is the control line

- include STYLE.DV2

implying a secondary driver STYLE.DV2 to be included, too. This driver has been designed for our present application and it sets the style of the document. For example, it defines various fonts, line widths and line spacings to be used in this book. Each of these settings could be given explicitly as control lines in the PRINT list. However, a predefined style file simplifies maintenance of the document.

Line 15 sets the page number for the first page and the principal font by [M] which is defined as 10.5 point Times in STYLE.DV2 . The header lines for the pages are specified by line 17, setting lines 4-7 for odd pages and 8-11 for even pages. On these header lines, the place of the page number is indicated by ### and that of the chapter index (here usually of type 3.2) by ????? . Line 16 gives an empty string to be currently substituted in the place of ????? .

The code words [PSAVE] and [PRESTORE] (defined in PS.DEV) and appearing in header lines take the control of saving and restoring the current graphics state. Then, for example, different fonts used in the main text and on the header lines will not be mixed.

Finally, the lines from 18 onwards give the text chapters to be included from other edit fields.

3.3 Control of the line length

The normal writing width of the lines in the edit field is 72, for it is the length of the visible line. If a longer or shorter line length is preferred, it is not necessary to use it at first. It is always better to write text in a free format and adjust the line length only after the writing and proofreading stage. Then it is very easy to achieve the desired line length by using various TRIM operations.



RESIZE?

Similarly, chapters which should ultimately be indented can originally be written normally from the left edge of the edit field and moved later to the right by using an INSERT command. However, it is easy to reset the first position of the line in writing by indicating that position with the cursor and pressing the keys [PREFIX] [ENTER] . Thereafter the [ENTER] key moves the cursor to the selected position on the next line until a new start position is selected with [PREFIX] [ENTER] .

TRIM operations are used for line length adjustment and justification. For example, TRIM 11, 50, 60 edits the text on lines 11-50 to line length 60 without splitting words. If the new condition of the text requires more lines (new line length is smaller), the text after line 50 in the current edit field will be

moved downwards to make space for the modified one. Correspondingly, when less lines are needed, the text after will be lifted upwards. A variant of TRIM is TRIM3 which also hyphenates (splits) the words according to the rules of Finnish. TRIM3 behaves also fairly well in English.

It is easy to fix possible hyphenation errors manually. Most conveniently this is done by using the key sucro H. It moves the rest of last word on the line to the beginning of the next line and vice versa.

Consider the following situation:

```
34 1 SURVO 84C EDITOR Thu Dec 26 12:45:21 1991      D:\P2\TEXT\ 100 100 0
25 *
26 *The last part of the word 'statistics'
27 *should be moved to this line.
28 *
```

When the cursor is pointing to the substring `tics'` we press the keys `PREFIX` `M` `H` and the text will be changed into

```
34 1 SURVO 84C EDITOR Thu Dec 26 12:45:22 1991      D:\P2\TEXT\ 100 100 0
25 *
26 *The last part of the word 'statis=
27 *tics' should be moved to this line.
28 *
```

Similarly, if the cursor is now moved to the last character of `tics'` and the same key sucro H is activated again, the original display will be restored.

Since TRIM is a common operation, also abbreviated forms are provided. The command words TRIM and TRIM3 can be written as T and T3. If the target for TRIM is the paragraph just after the TRIM line up to the next empty line, the line numbers may be omitted. For instance, T3 50 adjusts the next paragraph to line length 50. Even the form T or T3 is allowed and means adjusting to the default length 72 (maximum length of a visible line).

The TRIM commands above connect and disconnect lines freely. However, if a line does not start from the left edge of the current paragraph, but there are spaces before the first word, the start of the line remains as it is.

TRIM and TRIM3 do not justify the right edge of the text. This takes place by TRIM2 (T2), and its only task is to insert spaces properly between words on scanty lines. This is valid also in printouts on paper when a monospaced font (like typewriter font Courier) is in use.

***Trimming in proportional pitch**

When working with type sets with a proportional pitch, still other forms of TRIM are required for printing.

In PostScript printing, correct alignment of the right edge will be achieved in two steps, before activating the PRINT operation, all text chapters must be trimmed by a sucro command /TRIMP of the form

```
/TRIMP L1,L2,C,F(S)
```

where L1-L2 are the lines to be trimmed, C is the desired line length in picas



WORDS?

(1 pica = 6 Points = 1/12 inch) and F is the font type and S is the font size in Points. The font types F available are Times, Swiss, AvantGarde, Bookman, NewCentury, Palatino, and ZapfChancery (only initials required).

If the line labels L1,L2 are omitted, the text below the command line until the first empty line will be trimmed. For example, the text paragraphs of this document are processed by

```
/TRIMP 55,Times(10.5) .
```

After /TRIMP, the right edge of the text on the screen is not even since we have only fixed pitch available on the screen. However, if a PRINT operation with a control word of the form

```
- [trim(C)] / C is the line width as above
```

is now activated, the right edge of all chapters treated earlier by the /TRIMP surco command will be straight.

To change the width, use another value C in - [trim(C)]. To abandon the right edge alignment, use - [trim(0)].

3.4 Moving and copying text and tables



The commands INSERT, DELETE, COPY, MOVE, and CHANGE are available for various moves in the edit field. Also certain keys are useful in those tasks. LINEDEL? LINEINS?

INSERT and DELETE commands



INSERT adds one or more spaces in the same place on several lines and thus moves text to the right.

Example: When INSERT is activated in the situation below, (please, note the position of the cursor on line 19, position 22),

```
22 1 SURVO 84C EDITOR Sat Dec 27 15:26:26 1986 D:\P2\TEXT\ 100 100 0
18 *
19 *INSERT 20,24,10
20 * Country 0-14 15-24 25-44 45-64 65-
21 * Sweden 841 571 1188 930 585
22 * Denmark 564 385 731 537 308
23 * Finland 506 399 727 468 202
24 * Norway 474 316 534 442 251
25 *
```

the lines 20-24 will be moved from the point indicated by the cursor 10 steps to the right:

```
22 1 SURVO 84C EDITOR Sat Dec 27 15:28:43 1986 D:\P2\TEXT\ 100 100 0
18 *
19 *INSERT 20,24,10
20 * Country 0-14 15-24 25-44 45-64 65-
21 * Sweden 841 571 1188 930 585
22 * Denmark 564 385 731 537 308
23 * Finland 506 399 727 468 202
24 * Norway 474 316 534 442 251
25 *
```

If the line numbers (20,24 above) are omitted, INSERT will operate on suc-

cessive lines until the first empty line (here 25). Alternatively, we could have activated `INSERT 10` as well.

`INSERT` is an example of a command where the location of the cursor on the current line affects the result. Usually, the vertical position of the cursor has no influence.

The `DELETE` command has an opposite task since it deletes characters from a selected position on several lines. `DELETE` may erase text from the edit field without any warning.

The rules for `DELETE` are the same as those for `INSERT`. Thus, if in the last display above, the word `INSERT` is replaced by `DELETE` and the cursor is moved back to the position 22 as the line is again activated, the original situation before activation of `INSERT` would be restored.

COPY command and key

The `COPY` command copies one or more lines to a new place in the edit field. Old lines may then be overwritten. For example, `COPY 11, 20, 50` copies lines 11-20 so that the copy starts from line 50. Lines 11-20 will be preserved. To copy a single line, it is easier to use the `COPY` (alt-F3) key. When this key is pressed, the user is prompted to enter the number (or label) of the line to be copied from the current cursor position onwards.

The `COPY` command as an alternative form

```
COPY L1, L2 TO <text file>
```

that copies lines L1-L2 to a text file. If `TO <text file>` is omitted, the Survo output file selected by the command `OUTPUT <file>` is used. (See `OUTPUT?`).

MOVE command

The `MOVE` command is able to make more complicated transfers in the edit field. Any rectangular part of the edit field can be copied to another place by first indicating its left upper corner by some special character (say %), the right low corner by another (say &) and finally the new point for the left upper corner by a third character (say #). Now, if `MOVE %&#` is activated, the indicated area will be copied. One must select the characters so that they are unique in the current edit field; otherwise `MOVE` refuses to work.

Another form of `MOVE` is


```
MOVE L1, L2, C1, C2 FROM <text or edit file> TO L, C
```

It copies a rectangular array from a text or edit file to the current edit field. The array to be copied consists of lines L1-L2 and columns C1-C2. C1 and C2 are optional (default: entire lines). The left upper corner of the array will be copied to line L and column C in the edit field (C being optional with default value 1).



CLIPBOARD?

BLOCK and WORDS keys

A usually simpler method for copying rectangular blocks is to press key  **[BLOCK]** (alt-F4) and follow instructions given on the bottom line. The block to be copied is defined by indicating two of its opposite corners. The block will be seen painted in light blue color. The block thus defined can be copied to any number of places in the same edit field by indicating the position of the left-upper corner. The last block defined and copied is also automatically saved as a temporary file. Thus whenever later, one can copy it from that file by indicating the new place by the cursor and by pressing **[BLOCK]** **four** times.

When using the **[BLOCK]** key in insert mode, a suitable amount of empty lines will be inserted for the copy of the block.

Similarly, any part of a text chapter can be moved (or in insert mode copied) to another place by the **[WORDS]** (alt-F2) key. The beginning and the end of text is indicated by **[WORDS]**. The selected text appears in light blue color. Then the new position of the text (even inside another sentence) is indicated by **[WORDS]** and the text will be moved (or copied) to that place. Thereafter the text paragraphs in question are automatically formatted (by a TRIM command given as an autotrim line in the Survo system file SURVO.APU).

MERGE key

Still another useful key in moving parts of the text is **[MERGE]** (F6). It splits the current line into two lines from the position of the cursor (provided that the next line is empty) or if the current line to the right from the cursor is empty, the next line will be connected to the current line (if there is enough free space).

CHANGE command

The **CHANGE** command has two forms. One of them is used for exchange of two paragraphs in the edit field and another for exchange of columns of tables (**CHANGE?**).

If two paragraphs in the edit field should be exchanged, the **CHANGE** command is applied as follows:

```

10 1 SURVO 84C EDITOR Sat Dec 27 15:40:03 1986      D:\P2\TEXT\ 100 100 0
16 *
17 *Exchanging paragraphs:
18 X Here begins paragraph A,
19 * which should be replaced
20 X by paragraph C.
21 *      Here is chapter B, which should
22 *      stay in its place.
23 Y Here begins chapter C, which should be
24 Y replaced by chapter A.
25 *
26 *CHANGE XY
27 *

```

The first and last lines of the chapters to be exchanged are denoted in the control column by their own symbols (here X and Y).

If CHANGE XY on line 26 is activated, we get the display:

```

10 1 SURVO 84C EDITOR Sat Dec 27 15:44:14 1986 D:\P2\TEXT\ 100 100 0
16 *
17 *Exchanging paragraphs:
18 Y Here begins chapter C, which should be
19 Y replaced by chapter A.
20 * Here is chapter B, which should
21 * stay in its place.
22 X Here begins paragraph A,
23 * which should be replaced
24 X by paragraph C.
25 *
26 *CHANGE XY_
27 *

```

If the same CHANGE is activated anew, the original order of the chapters will be restored.

CHANGE can also be employed for moving a single chapter by denoting the new place (an empty line) by Y and the first and last lines of the chapter by X's.

3.5 Clearing the edit field

The operations SCRATCH, CLEAR, and ERASE are meant for various clearing actions of the edit field. DELETE, which was described earlier, is useful when clearing columns of tables.

SCRATCH (without parameters) clears the whole edit field from the current line onwards and restores '*' in the control column in these lines.

CLEAR with two line labels clears the selected lines from the position of the cursor to the right. Thus, if CLEAR 11, 20 is activated and the cursor is in the first position (under letter C), the lines 11-20 are completely erased.

CLEAR has also another form, which is illustrated by the next pair of displays:

```

13 1 SURVO 84C EDITOR Sat Dec 27 16:01:43 1986 D:\P2\TEXT\ 100 100 0
21 *
22 *CLEAR M,B,23_
23 * XXXXXXXX          XXX
24 M Country  0-14 15-24 25-44 45-64 65-
25 A Sweden   841   571  1188   930  585
26 * Denmark  564   385   731   537  308
27 * Finland  506   399   727   468  202
28 B Norway   474   316   534   442  251
29 *

```

Here the extra parameter 23 refers to so-called *mask line*, which accurately (up to one character) indicates the columns to be cleared. After CLEAR has carried out its task, we have

```

13 1 SURVO 84C EDITOR Sat Dec 27 16:04:43 1986 D:\P2\TEXT\ 100 100 0
21 *
22 *CLEAR M,B,23_
23 * XXXXXXXX XXX
24 M      0-14 15-24 25-44 45-64
25 A      841  571 1188  930
26 *      564  385  731  537
27 *      506  399  727  468
28 B      474  316  534  442
29 *

```

The **[BLOCK]** (alt-F4) key may also be used for clearing rectangular areas in the edit field.

ERASE is a clearing command which erases selected characters in the edit field. For example,

ERASE ()

will erase all parentheses and

ERASE +- .1234567890

all numbers. **ERASE** is often applied after **MOVE** to erase the special indicator characters.

3.6 Searching and replacing



Words and other expressions appearing in the text often have to be found and even replaced. For various searches in the current edit field, commands **FIND** and **REPLACE** are available. Also the **[SEARCH]** (alt-F5) key and the key combination **[PREFIX]** **[J]** are useful in certain searches.

FIND?
SEARCH?

Searches in Survo data files and lists consisting of several edit files are described later.

FIND command

FIND <text>

finds all the occurrences of <text> from the current edit field. After activation of **FIND**, the system will show the first case found and the user can point out (instructions will be displayed) how to proceed. For example, by pressing **[N]** the next occurrence of <text> will be displayed. The search may be interrupted by **[ENTER]** and the cursor remains to indicate the last case found. By pressing **[C]** all the cases are searched for automatically and only the number of occurrences will be displayed.

REPLACE command

REPLACE <old_text>, <new_text>

finds all the occurrences of <old_text> below the activated **REPLACE** line and replaces them when needed by <new_text>. After activation, the system will show the first case found and the user can point out (instructions are seen on the screen) how to proceed. For example, the **[R]** key replaces and continues the search, but **[N]** only continues the search without replacement. The user may thus select those cases which should be replaced. The spaces in the

text strings have to be substituted by characters `_`. Example:

```
REPLACE SURVO_84C,SURVO_84C_system
```

Another way is to put the texts in quotation marks as

```
REPLACE "SURVO 84C","SURVO 84C system"
```

Also `REPLACE` displays the number of cases found before returning to the normal editorial mode.

```
REPLACE <old_text>,<new_text>,<C or N>
```

does the replacement without any prompts or interrupts. If `N` is used as an extra parameter, the number of replacements will also be put in the sucro memory (see `SUCROS?`). For example,

```
REPLACE Survo,Survo,N
```

counts the frequency of words `Survo` below the current line and puts that frequency in the sucro memory.

SEARCH key

`SEARCH` (`alt-F5`) initiates a stepwise search in the edit field. The user is prompted to enter the characters of the search string one after another and the first occurrence of the given string is shown immediately in each stage. The process is interrupted until the given string is not found anymore or by pressing `ENTER`.

Word completing

The key combination `PREFIX` `J` initiates a search for an incomplete word just before the cursor and completes it by the first matching word found in the current edit field.

Example: Assume that the user has written

```
21 1 SURVO 84C EDITOR Sat Dec 28 16:55:14 1991 D:\P2\TEXT\ 100 100 0
30 *
31 *Ouagadougou is the capital of Burkina Faso.
32 *The population of Ou_
33 *
```

By pressing `PREFIX` `J` the system will complete the word as

```
21 1 SURVO 84C EDITOR Sat Dec 28 16:55:15 1991 D:\P2\TEXT\ 100 100 0
30 *
31 *Ouagadougou is the capital of Burkina Faso.
32 *The population of Ouagadougou_
33 *
```

and the user can continue typing. The search for possible continuations is restricted to the edit field in the natural order of edit lines.

This search technique is really effective in situations where certain information according to a given keyword must be found from a table in the edit field. Assume that we are studying population statistics

```

8 1 SURVO 84C EDITOR Sat Dec 28 17:19:26 1991 D:\P2\TEXT\ 170 80 0
1 *
2 *Median age (M.age), percentage of urban population (Urban),
3 *population density (Density) and annual growth (Growth)
4 *in 150 countries of the world.
5 *
6 * Nether_
7 *
8 * Country M.age Urban Density Growth
9 *
10 * Finland 34.3 66.9 15 0.30
11 * Denmark 35.8 85.9 119 0.14
12 * Iceland 28.5 89.6 2 0.91
13 * Ireland 26.5 60.9 50 0.99
14 * Norway 34.4 57.2 13 0.38
15 * Sweden 37.5 89.2 18 0.08
16 * Great Britain 35.5 91.7 228 0.04
17 * Albania 22.1 39.3 106 1.93
18 * Greece 34.4 65.9 73 0.57
19 * Italy 35.5 71.7 193 0.35
20 * Malta 32.0 85.4 1117 0.71
21 * Portugal 29.7 33.5 112 0.79
22 * Spain 31.1 77.4 77 0.77
23 * Yugoslavia 31.6 46.3 91 0.72

```

and want to find values for the Netherlands. On line 6, we have typed `Nether_`. Now, by pressing `PREFIX` `J` we get the word completed (since Netherlands and its data appear somewhere later in the table) and a prompt on the bottom line as follows:

```

8 1 SURVO 84C EDITOR Sat Dec 28 17:19:26 1991 D:\P2\TEXT\ 170 80 0
1 *
2 *Median age (M.age), percentage of urban population (Urban),
3 *population density (Density) and annual growth (Growth)
4 *in 150 countries of the world.
5 *
6 * Netherlands
7 *
8 * Country M.age Urban Density Growth
9 *
10 * Finland 34.3 66.9 15 0.30
11 * Denmark 35.8 85.9 119 0.14
12 * Iceland 28.5 89.6 2 0.91
13 * Ireland 26.5 60.9 50 0.99
14 * Norway 34.4 57.2 13 0.38
15 * Sweden 37.5 89.2 18 0.08
16 * Great Britain 35.5 91.7 228 0.04
17 * Albania 22.1 39.3 106 1.93
18 * Greece 34.4 65.9 73 0.57
19 * Italy 35.5 71.7 193 0.35
20 * Malta 32.0 85.4 1117 0.71
21 * Portugal 29.7 33.5 112 0.79
22 * Spain 31.1 77.4 77 0.77
23 * Yugoslavia 31.6 46.3 91 0.72
Next word by the space bar. Cancel by DELETE. Accept by ENTER!

```

By pressing the space bar 4 times, we get the 4 data values for the Netherlands to the line 6 as follows:

```

8 1 SURVO 84C EDITOR Sat Dec 28 17:19:26 1991 D:\P2\TEXT\ 170 80 0
1 *
2 *Median age (M.age), percentage of urban population (Urban),
3 *population density (Density) and annual growth (Growth)
4 *in 150 countries of the world.
5 *
6 * Netherlands 33.0 76.3 354 0.40
7 *
8 * Country M.age Urban Density Growth
9 *
10 * Finland 34.3 66.9 15 0.30
11 * Denmark 35.8 85.9 119 0.14
12 * Iceland 28.5 89.6 2 0.91

```

Hence we have the possibility of picking up a piece of data from a table without knowing the original position of that data.

3.7 Moving data between edit fields and text files


As described earlier, even large reports can be maintained in separate parts saved in various edit fields. Usually there is no particular interest in joining all the pieces together in Survo. In some applications, however, information from other sources is required in the edit field. For example, we may need text files (ASCII files) made in other systems to be moved to Survo and vice versa. In this kind of communication, commands like `SAVE`, `LOAD`, `SAVEP`, `LOADP`, and `SHOW` are useful.

The `SAVE` and `LOAD` commands have been introduced earlier. `SAVE` stores the entire edit field with its control column and other attributes to an edit file on the current data disk. `LOAD` substitutes the current edit field by the contents of an edit file.

Since the moves of the edit fields in this way are rather fast, no partial savings and loadings are generally required.

The edit files (with the default extension `.EDT` in the name) processed by `SAVE` and `LOAD` are not standard text files since they have to contain information about the edit field structure and about various display attributes stored as shadow lines.

SAVEP and LOADP commands

In communication between the edit field and text files, the `SAVEP` and `LOADP` commands take care of corresponding tasks. 

A part of an edit field may be saved as a text file by a `SAVEP` operation. **SAVEP?**
For example,

```
SAVEP 11,20,A:PART2.TXT
```

saves the text on edit lines 11-20 as a text file `PART2.TXT` on the disk `A:`. The possible previous contents of `PART2.TXT` will be destroyed. The control column and the display attributes for the text are not saved.

```
SAVEP PART2.TXT
```

saves all the lines below the active line as a text file `PART2.TXT` on the cur-

rent Survo data disk/path. The trailing empty lines at the end of the edit field are not saved.

SAVEP also permits writing and editing of program files in the editor. Similarly, various data and results can be moved from Survo to other systems provided that they are able to read text files.

LOADP has an opposite task. For example,

```
LOADP A:PART2.TXT,31
```

loads the contents of the text file PART.TXT on disk A: into the current edit field so that its first line will be copied as the edit line 31. If the line number is missing, the copied lines will appear below the activated LOADP line.

SHOW operation

The SHOW operation has partially a similar function, but it allows the text file to be scanned in a temporary window before copying lines into the edit field. SHOW can be applied both for edit files and general text files.

```

11 1 SURVO 84C EDITOR Sat Dec 29 12:28:51 1991      D:\P2\TEXT\ 100 100 0
45 *SHOW operation
46 *The SHOW operation has partially a similar function, but it allows
47 *the text file to be scanned in a temporary window before copying lines
48 *into the edit field. SHOW can be applied both for edit files and gene-
49 *ral text files.
50 *
51 *SHOW TEXT8_
1  *SAVE TEXT8
2  *LOAD TEXT9
3  *
4  *Survo - Integrated Environment
5  *DEF A,6,END
6  *
7  *3.7 Moving data between edit fields and text files
8  *
9  *As described earlier, even large reports can be maintained in separate
10 *parts saved in various edit fields. Usually there is no particular in-
12 *terest in joining all the pieces together in Survo. In some applica-
13 *tions, however, information from other sources is required in the edit
14 *field. For example, we may need text files (ASCII files) made in other
15 *systems to be moved to Survo and vice versa. In this kind of communica-
16 *tion, commands like SAVE, LOAD, SAVEP, LOADP and SHOW are useful.
ENTER=Exit N=Next P=Prev E=End L=Load S=Search

```

Above, we have a situation after the SHOW operation on line 51 has been activated. In the temporary window below the SHOW line, the first 16 lines of the edit file TEXT8 are displayed. The file may be scrolled in this window by keys **[N]** and **[P]**. Selected lines may be copied to the current edit field below the SHOW line (if there are empty lines) by pressing **[L]**. When pressing **[ENTER]**, the temporary window will be shut and the control returns to the normal editing mode.

Besides data moves, SHOW makes it convenient to view two edit or text files on the screen simultaneously. When scanning a general text file, it is also possible to search for any text by the **[S]** key.


3.8 Output files

The results of various Survo operations can be directed to the printer if the user so desires. For many reasons, this is not usually a good practice. It is far more convenient to use either the edit field or special output files for immediate results and edit them before the actual printout.


Thus, for example, all statistical operations of Survo write their results to the current edit field or to an output file which is a general text file. Especially for large output tables, the output file should be preferred. *Often, a file on the virtual disk (ramdrive) is a good choice for an output file.*

The default output file is selected automatically by the Survo system (see SYSTEM?). The output file is reselected during the work by the


OUTPUT <name_of_output_file>

command. By activating OUTPUT without parameters, the name of the current output file is written to the edit field. 

Later, it is easy to scan output files and load pertinent information to the edit field by the SHOW operation.

An output file which stores results in text form is not always sufficient for numerical results obtained in numerical and statistical computations. In particular, when some result should be used as input in subsequent analysis, there may appear loss of precision due to excessive rounding. Therefore most of the numerical results are saved in double precision as intermediate files (matrix files). These output files will be described later (OUTCNTRL?).  **OUTCNTRL?**


3.9* LIST operations

LIST operations are intended for processing of long lists and reports. Such lists consist of several chapters specified by DEF definitions in different edit files.  **LIST?**

The current version of Survo includes only two LIST operations. In the nearest future, certain additional operations for sorting of textual data and for statistical summaries of such data will be implemented.

A list or a report to be processed by LIST operations is to be defined in the current edit field in the following way:

```
LIST <name_of_the_list>: <chapter_1>,<edit_field_1>
                        <chapter_2>,<edit_field_2>
                        ...
                        END
```



Example:

```
LIST REPORT1: A,INTRO1 A,INTRO2 A,INTRO3
              A,CH1 A,CH2 A,CH3 A,CH4 A,CH5 *,REF END
```

'*' in place of a chapter name means the entire edit file. The list above can

also be given in an abbreviated form

```
LIST REPORT1: A,INTRO1-3 A,CH1-5 *,REF END
```

Edit files can given with complete pathnames. A common path for all fields is entered by `PATH=<path>`. Example: `PATH=D:\REPORT\`.

LIST SHOW operation

```
LIST SHOW <name_of_list>
```

opens a temporary window below the current (`LIST SHOW`) line and displays the contents of the edit fields belonging to the list. The lines (and columns) may be scrolled in that window in the same way as the edit field is scrolled on the screen. The user sees the text belonging to the list as one contiguous stream of lines. The changes of the fields are indicated by extra header lines telling the edit file and chapter.

Various searches of strings over the entire list are initiated by the `SEARCH` (`alt-F5`) key. More information while staying in `LIST SHOW` is obtained by the `HELP` (`F1`) key. The `S` key tells about search options.

```
LIST SHOW *
```

works with the list that has been used most recently by `LIST SHOW`.

In the `LIST SHOW` mode, the text in the list (edit fields) cannot be edited. However, by means of a `/LIST` sucro, an interplay between the `LIST SHOW` mode and the standard editorial mode can be arranged. The sucro command `/LIST <name_of_list>`

calls the corresponding `LIST SHOW` operation. On exit from `LIST SHOW`, the last edit field shown by `LIST SHOW` is loaded automatically and the cursor indicates the same place as it was pointing at in `LIST SHOW`. The user can edit the field and save it normally. By activating `/LIST *` (this could be defined as a macro), `LIST SHOW` is called again and it immediately gives the same display as in the previous call. In this way, even very long reports are maintained easily.

LIST REPLACE operation

```
LIST REPLACE <name_of_list>,L1,L2
```

replaces all the occurrences of the text given on the edit line `L1` by that given on line `L2`. Even shadow characters are observed. However, `LIST REPLACE` is not able to create new shadow lines. Hence, cases where `L2` has shadows but `L1` not are not allowed. For cases where trailing spaces should be observed, a generalized form is available. Its syntax is

```
LIST REPLACE <name_of_list>,L1,L2,len1,len2
```

where `len1` is the length of the old string and `len2` that of the new one.

4. Table management

Tables appear in various formats in many applications. In this chapter we shall consider management of standard tables having regular columns. The tables may be written in the edit field or they can be loaded from other sources.

Our main interest is focused on construction, editing and sorting of tables. Simple arithmetic operations on tables is described to some extent, too.

A considerable part of functions related to tables will be introduced in later chapters. Then we shall consider

- more advanced computing (editorial arithmetics, touch mode) (**MATH?**),
- transformation of variables (columns) (**VAR?**)
- statistical operations on tables (**STATIS?**),
- computing and analyzing multiway tables (**TAB?**),
- matrix operations (**MAT?**).

All this amounts that Survo offers many approaches to so-called spreadsheet computing. Our procedures are different from those applied in most spreadsheet programs since the edit field is a more general concept than the common spreadsheet. One basic difference is that in spreadsheet computing the smallest operand is typically one number, word or expression, but in editorial mode the operations can be addressed up to one character. This more accurate pointing facility greatly simplifies combining of various activities. There are no different work sheets and command areas; everything is done directly in the same environment. The idea of the whole editorial approach originated from this principle.



TAB?
TABFIT?
TABTEST?
/TABULATE
BURT?

4.1 Editing tables in the edit field

When typing a new table which consists of several columns, it is not necessary to keep the columns properly aligned. Thus a table may be entered like any text as follows:

```

13 1 SURVO 84C EDITOR Sun Dec 28 11:49:25 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 *Finland      12.5   0.15  54.7  7.6   2.7
7 *Sweden      12.9  0.30  58.3  7.9   2.9
8 *Norway       9.4   0.19  43.5  3.1   1.8
9 *Denmark     11.8  0.41  113.9 10.4  1.7
10 *England     1.8   3.49  113.7 5.1   1.4
11 *Ireland     0.2   3.73  124.5 3.8   1.9
12 *Holland     9.2   0.58  75.5  9.7   2.7
13 *Switzerland 9.1   0.25  73.5  44.9  2.1
14 *France      5.2   0.10  44.5  104.3 2.5
15 *Italy       3.6   0.06  13.6  106.6 2.0
16 *Spain       2.5   0.03  43.6  73.2  2.7
17 *Portugal   2.2   0.03  27.5  89.3  0.9
18 *
19 *XXXXXXXXXXXX 12.1   1.12  123.1 123.1  1.1
20 *FORM 6,17,19_
21 *
22 *
23 *

```

The numbers are now easily repositioned to aligned columns by using a FORM operation which refers to lines in question (6,17) and to a so-called mask line (19) which can be any free line. The mask line is a model of a typical adjusted line and tells where and in what form the alphanumeric (XXXXXXXXXXXX) and numeric (12.1) columns will be cast.

By activating FORM on line 20, the following display will be achieved:

```

13 1 SURVO 84C EDITOR Sun Dec 28 11:55:05 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 * Finland      12.5   0.15  54.7  7.6   2.7
7 * Sweden      12.9  0.30  58.3  7.9   2.9
8 * Norway       9.4   0.19  43.5  3.1   1.8
9 * Denmark     11.8  0.41  113.9 10.4  1.7
10 * England     1.8   3.49  113.7 5.1   1.4
11 * Ireland     0.2   3.73  124.5 3.8   1.9
12 * Holland     9.2   0.58  75.5  9.7   2.7
13 * Switzerland 9.1   0.25  73.5  44.9  2.1
14 * France      5.2   0.10  44.5  104.3 2.5
15 * Italy       3.6   0.06  13.6  106.6 2.0
16 * Spain       2.5   0.03  43.6  73.2  2.7
17 * Portugal   2.2   0.03  27.5  89.3  0.9
18 *
19 *XXXXXXXXXXXX 12.1   1.12  123.1 123.1  1.1
20 *FORM 6,17,19_
21 *
22 *
23 *

```

Another way to enter data in tabular form is to employ the **TAB** key. **TAB** moves the cursor to the next tab position which is defined by the T characters on the first edit line having a 'T' in its control column. If no T line is found, columns 11,21,31,41,... are the default tab positions. After **TAB** is pressed, numbers to be typed will be correctly aligned to form straight columns.

Thus the previous table could also be entered in the following way:

```

29 1 SURVO 84C EDITOR Sun Dec 28 12:18:47 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee   Tea      Beer   Wine   Spirits
5 T              T        T        T      T      T
6 * Finland      12.5    0.15    54.7   7.6    2.7
7 * Sweden       12.9    0.30    58.3   7.9    2.9
8 * Norway       9.4     0.19    43.5   3.1    1.8
9 * Denmark      11.8    0.41    113.9  10.4   1.7
10 * England     1.8     3.49    113.7  5.1    1.4
11 * Ireland     0.2     3.73    124.5  3.8    1.9
12 * Holland     9.2     0.58    75.5   9.7    2.7
13 * Switzerland 9.1     0.25    73.5   44.9   2.1
14 * France      5.2     0.10_
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *

```

Above, the line 5 (with a 'T' in the control column) defines the tab positions. To enter the next number (beer consumption for France) the **TAB** key is pressed and the cursor advances to the next tab position and the number (44.5) is typed normally. Then **TAB** is pressed again and the next number (104.3) is typed. After the last column, **TAB** gives a beep sound and the cursor advances to the start of the next line.

Some of the text editing operations are suitable for table management as well. The INSERT and DELETE commands, for example, are useful in making room for new columns and in deletion of old ones.

When one needs to exchange columns, an alternative form of the CHANGE command is supplied:

```

15 1 SURVO 84C EDITOR Sun Dec 28 12:41:02 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee   Tea      Beer   Wine   Spirits
5 *
6 * Finland      12.5    0.15    54.7   7.6    2.7
7 * Sweden       12.9    0.30    58.3   7.9    2.9
8 * Norway       9.4     0.19    43.5   3.1    1.8
9 * Denmark      11.8    0.41    113.9  10.4   1.7
10 * England     1.8     3.49    113.7  5.1    1.4
11 * Ireland     0.2     3.73    124.5  3.8    1.9
12 * Holland     9.2     0.58    75.5   9.7    2.7
13 * Switzerland 9.1     0.25    73.5   44.9   2.1
14 * France      5.2     0.10    44.5   104.3  2.5
15 * Italy        3.6     0.06    13.6   106.6  2.0
16 * Spain       2.5     0.03    43.6   73.2   2.7
17 * Portugal    2.2     0.03    27.5   89.3   0.9
18 *
19 *          XXXXXXXXXXXXX          YYYYYYY
20 *CHANGE 4,19,19_
21 *
22 *
23 *

```

Also this operation refers to lines to be processed (4,19) and to a mask line

(19). The task of the mask line is to indicate the columns (within an accuracy of one character) by X and Y masks. After activation of **CHANGE** we have:

```

15 1 SURVO 84C EDITOR Sun Dec 28 12:44:12 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Spirits  Beer  Wine  Coffee  Tea
5 *
6 * Finland      2.7    54.7   7.6 12.5   0.15
7 * Sweden       2.9    58.3   7.9 12.9   0.30
8 * Norway       1.8    43.5   3.1  9.4    0.19
9 * Denmark      1.7   113.9  10.4 11.8   0.41
10 * England     1.4   113.7   5.1  1.8    3.49
11 * Ireland     1.9   124.5   3.8  0.2    3.73
12 * Holland     2.7    75.5   9.7  9.2    0.58
13 * Switzerland 2.1    73.5  44.9  9.1    0.25
14 * France      2.5    44.5  104.3 5.2    0.10
15 * Italy        2.0    13.6  106.6 3.6    0.06
16 * Spain       2.7    43.6   73.2 2.5    0.03
17 * Portugal    0.9    27.5   89.3 2.2    0.03
18 *
19 *              YYYYYYY                XXXXXXXXXXXXX
20 *CHANGE 4,19,19_
21 *
22 *
23 *

```

On purpose, we selected the range of lines to include the mask line (19), too. The advantage of this procedure is that the masks will also be exchanged properly and thus the original setting of columns may be reached by activating the same **CHANGE** again.

4.2 Sorting of tables

SORT is the command for sorting lines of a table in the edit field. Larger tables and data which cannot be kept in the edit field can be moved to Survo data files and sorted by the **FILE SORT** operation.

We shall now sort the data set of our previous examples with respect to beer consumption:

```

13 1 SURVO 84C EDITOR Sun Dec 28 12:52:00 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 * Finland      12.5   0.15  54.7  7.6   2.7
7 * Sweden       12.9   0.30  58.3  7.9   2.9
8 * Norway       9.4    0.19  43.5  3.1   1.8
9 * Denmark      11.8   0.41  113.9 10.4  1.7
10 * England      1.8    3.49  113.7  5.1   1.4
11 * Ireland      0.2    3.73  124.5  3.8   1.9
12 * Holland      9.2    0.58  75.5  9.7   2.7
13 * Switzerland  9.1    0.25  73.5  44.9  2.1
14 * France       5.2    0.10  44.5  104.3 2.5
15 * Italy         3.6    0.06  13.6  106.6 2.0
16 * Spain        2.5    0.03  43.6  73.2  2.7
17 * Portugal     2.2    0.03  27.5  89.3  0.9
18 *
19 *
20 *SORT 6,17,19_
21 *
22 *
23 *

```

In the SORT command on line 20, the lines to be sorted (6,17) and the mask line (19) are given. The mask 11111 indicates the beer column and since it consists of digits, a numeric sort will be carried out. Activation of SORT leads to:

```

13 1 SURVO 84C EDITOR Sun Dec 28 12:55:41 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 * Italy         3.6    0.06  13.6  106.6 2.0
7 * Portugal     2.2    0.03  27.5  89.3  0.9
8 * Norway       9.4    0.19  43.5  3.1   1.8
9 * Spain        2.5    0.03  43.6  73.2  2.7
10 * France       5.2    0.10  44.5  104.3 2.5
11 * Finland      12.5   0.15  54.7  7.6   2.7
12 * Sweden       12.9   0.30  58.3  7.9   2.9
13 * Switzerland  9.1    0.25  73.5  44.9  2.1
14 * Holland      9.2    0.58  75.5  9.7   2.7
15 * England      1.8    3.49  113.7  5.1   1.4
16 * Denmark      11.8   0.41  113.9 10.4  1.7
17 * Ireland      0.2    3.73  124.5  3.8   1.9
18 *
19 *
20 *SORT 6,17,19_
21 *
22 *
23 *

```

We have now the countries in ascending order of beer consumption. To achieve descending order, SORT is replaced by -SORT. Thus a '-' is to be inserted in front of SORT and the command is reactivated. It is also simple to change the sort key by moving the mask on line 19. For example, if a list of countries in alphabetic order is needed, a mask of the form XXXXXXXXXX is typed to cover the name column and SORT is activated again.

SORT allows a nested sorting according to several sort keys simultaneously. Then the primary sort key is denoted by A letters, the secondary one by B letters, etc.

Besides pure sorting, the `Sort` command may be employed in some other tasks, too. For example, a subset of the data in the table is extracted by first making a new column of 1's and 0's, where the 1's indicate the lines to be selected. When a `Sort` command using this column as a sort key is activated, the selected lines will appear as a unified sub-table.



TOUCH?

*Filters for sorting

In alphabetic sorting, the order of various characters is determined by the file `SortCode.Bin`. This file can be edited by using the `Codes Load` (and `Codes Save`) operation (see `Codes?`). It can also be replaced by another `.Bin` file by entering a specification `Filter=<name_of_file>` when `Sort` is activated. The default path for `.Bin` files is `.\Sys`.

For example, `Filter=SortLow.Bin` defines the order `AaBbCc...ÄäÖö` where 'A' and 'a' are considered distinct and 'A' precedes 'a'.

4.3 Making new columns

New columns in tables may be formed in several ways. The commands `Set`, `Count`, `C+`, `C-`, `C*`, and `C/`, for instance, are suitable for this purpose. Some more advanced means (like operations `Var`, `Classify`, and techniques provided by Touch mode) give more possibilities for complicated transformations in tables.

Often a new column has to be written in the middle of the current table. Space for it can be made by using the `Insert` command. Typing in the middle of a table is simplified by defining the starting position of a new line temporarily by keys `PREFIX` and `ENTER`.

Another alternative is to type the new column as the first one and move it later to its final place by the `Change` command or by the `BLOCK` key.

The idea of `Set` and `Count` commands can be seen from the next display which tells the situation after both have been activated (the results of the operations are shown here in gray shading).

```

14 1 SURVO 84C EDITOR Sun Dec 28 13:22:19 1986 D:\P2\TEXT\ 180 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 * Finland      1 12.5  0.15  54.7  7.6   2.7   * 1
7 * Sweden       2 12.9  0.30  58.3  7.9   2.9   * 1
8 * Norway       3 9.4   0.19  43.5  3.1   1.8   * 1
9 * Denmark      4 11.8  0.41  113.9 10.4  1.7   * 1
10 * England      5 1.8   3.49  113.7 5.1   1.4   * 1
11 * Ireland      6 0.2   3.73  124.5 3.8   1.9   * 1
12 * Holland      7 9.2   0.58  75.5  9.7   2.7   * 1
13 * Switzerland  8 9.1   0.25  73.5  44.9  2.1   * 1
14 * France       9 5.2   0.10  44.5  104.3 2.5   * 1
15 * Italy        10 3.6   0.06  13.6  106.6 2.0   * 1
16 * Spain       11 2.5   0.03  43.6  73.2  2.7   * 1
17 * Portugal    12 2.2   0.03  27.5  89.3  0.9   * 1
18 *
19 *
20 *SET 6,17,19
21 *
22 *COUNT 6,17,21,12
23 *

```

The task of the SET operation is simply to copy the non-space characters of the mask line to selected lines. It is handy when making columns with a constant value.

COUNT has several forms. The main function is to create columns with values increasing or decreasing by a constant increment. The mask line (here 21) indicates the position and the format of the new column (12 means an integer of two digits).

The parameters of COUNT tell precisely the initial value and the increment. The default for both the initial value and the increment is 1. Thus COUNT 6,17,21 forms the sequence 1,2,3,... starting from line 6 and ending to line 17 according to the mask line 21. By adding a fourth parameter, the initial value is changed.

For example, COUNT 6,17,21,1976 creates the sequence 1976, 1977, 1978, ... as a new column. Then the mask on line 21 should be widened to form 1234, for example.

The fifth parameter gives the increment. For example,

```
COUNT 6,17,21,100,10
```

produces numbers 100, 110, 120, 130, 140, ...

To make various periodic or cyclic columns, a sixth parameter is added. The sign of this parameter determines the type of periodicity. The following example illustrates this possibility. Our task is to generate a time column for a quarterly time series with notations 1986:1, 1986:2, 1986:3, 1986:4, 1987:1, etc.

```

12 1 SURVO 84C EDITOR Sun Dec 28 13:42:46 1986 D:\P2\TEXT\ 100 100 0
1 *
2 *
3 *COUNT A,B,X,1986,1,-4
4 *COUNT A,B,Y,1,1,4
5 *SET A,B,Z
6 *
7 X 1234
8 Y 1
9 Z :
10 *
11 A 1986:1
12 * 1986:2
13 * 1986:3
14 * 1986:4
15 * 1987:1
16 * 1987:2
17 * 1987:3
18 * 1987:4
19 * 1988:1
20 * 1988:2
21 * 1988:3
22 * 1988:4
23 B 1989:1

```

The three operations on lines 3-5, when succesively activated, have produced the display above.

The commands C+, C-, C*, C/, and C% form new columns by addition, subtraction, multiplication, division, and taking percentages.

For example, to compute the ratio between coffee and tea consumption, the following C/ command is activated. (The results are shaded here).

```

11 1 SURVO 84C EDITOR Sun Dec 28 14:15:22 1986 D:\P2\TEXT\ 120 82 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country Coffee Tea Beer Wine Spirits
5 *
6 * Finland 12.5 0.15 54.7 7.6 2.7 83.333
7 * Sweden 12.9 0.30 58.3 7.9 2.9 43.000
8 * Norway 9.4 0.19 43.5 3.1 1.8 49.474
9 * Denmark 11.8 0.41 113.9 10.4 1.7 28.780
10 * England 1.8 3.49 113.7 5.1 1.4 0.516
11 * Ireland 0.2 3.73 124.5 3.8 1.9 0.054
12 * Holland 9.2 0.58 75.5 9.7 2.7 15.862
13 * Switzerland 9.1 0.25 73.5 44.9 2.1 36.400
14 * France 5.2 0.10 44.5 104.3 2.5 52.000
15 * Italy 3.6 0.06 13.6 106.6 2.0 60.000
16 * Spain 2.5 0.03 43.6 73.2 2.7 83.333
17 * Portugal 2.2 0.03 27.5 89.3 0.9 73.333
18 *
19 * XXXX YYY Y 123.123
20 *C/ 6,17,19
21 *
22 *
23 *

```

The mask line shows the dividend (XXXX), the divisor (YYYY) and the place and the format for the result (123.123).

In the addition command C+ and in the multiplication command C*, the number of operands (indicated by X's) may be more than 2. In the C% command, the ratio of the X column and the Y column is computed in percentages.

In all these C commands, some elementary functions can be included. For example, if the operation on line 20 had been written in the form

```
C-log 6,17,19
```

the difference of the logarithms of Coffee and Tea

```
log(Coffee)-log(Tea)
```

would have been computed as a new column.

Permitted functions in C operations are log, exp, sqrt, sin, cos, tan, arctan, int, and abs.

To compute sums of rows etc., commands L+, L-, L*, L/, and L% are available. Their structure resembles that of C operations. (See L+?).

Many computations with the tables are simplest to perform in touch mode, although it takes some time and effort to learn this technique. (Touch mode will be explained later in 5.6.) For example, computing a sum from numbers in a column is accomplished in touch mode in 5 key strokes.



WORDS?
CHARS?

4.4* Interpolation

```
INTERP L1,L2,L3,L4,K
```

where K is a label of a mask line of the form

```
XXXXX  XXXX  YY.YYY  XX  YYY.YY
```

interpolates columns denoted by Y masks by linear regression analysis using X columns as regressors. If there are no X columns or only one X column appears, the Y columns are interpolated by polynomial regression. In this case, the degree m of the polynomial is given by the specification DEGREE=m. Default is DEGREE=1.

The source data with complete X and Y values is given on lines L1-L2. The interpolated (and extrapolated) Y values will be computed on lines L3-L4 using given X values on the same lines.

In polynomial regression without X columns, L-L1+1 where L is the current line number, is the basic regressor. In this case it is typical that L3=L2+1.

The next example illustrates how INTERP works. We have an incomplete table of second and third powers of integers N as follows:

```

17 1 SURVO 84C EDITOR Sat Jan 04 18:37:29 1992 D:\P2\TEXT\ 120 80 0
1 *
2 * N N^2 N^3
3 M XX YYYY YYYY
4 A 1 1 1
5 * 2 4 8
6 * 3 9 27
7 * 4 16 64
8 B 5 25 125
9 C 6
10 * 7
11 D 8
12 *
13 *INTERP A,B,C,D,M_ / DEGREE=2
14 *

```

When INTERP on line 13 is activated, we get a second degree approximation for the Y columns for the N values 6,7, and 8:

```

17 1 SURVO 84C EDITOR Sat Jan 04 18:37:30 1992 D:\P2\TEXT\ 120 80 0
1 *
2 * N N^2 N^3
3 M XX YYYY YYYY
4 A 1 1 1
5 * 2 4 8
6 * 3 9 27
7 * 4 16 64
8 B 5 25 125
9 C 6 36 199
10 * 7 49 293
11 D 8 64 404
12 *
13 *INTERP A,B,C,D,M_ / DEGREE=2
14 *

```

Of course, the squares become exact but the cubics are false, due to a quadratic extrapolation. By entering DEGREE=3, also the last column will be completed correctly.

```

17 1 SURVO 84C EDITOR Sat Jan 04 18:37:47 1992 D:\P2\TEXT\ 120 80 0
1 *
2 * N N^2 N^3
3 M XX YYYY YYYY
4 A 1 1 1
5 * 2 4 8
6 * 3 9 27
7 * 4 16 64
8 B 5 25 125
9 C 6 36 216
10 * 7 49 343
11 D 8 64 512
12 *
13 *INTERP A,B,C,D,M_ / DEGREE=3
14 *

```

The same results would have been obtained even if the first column (N values) or the X mask is omitted. Then the orders 1,2,... of the lines would be used as the values of the regressor.

Statistical operations like LINREG and FORECAST offer more advanced means for problems of this kind.

4.5* Transposing

TRANSP L1,L2,L

transposes a table (i.e. exchanges rows and columns) on the edit lines L1-L2 and writes the transposed table from the line L onwards. The original table must be a rectangular array and it can contain columns of both numbers and words.



TRANSPPOSE?

Our table on European countries is transposed as follows:

```

15 1 SURVO 84C EDITOR Sat Jan 04 19:00:07 1992 D:\P2\TEXT\ 50 150 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 * Finland      12.5   0.15  54.7  7.6   2.7
6 * Sweden       12.9   0.30  58.3  7.9   2.9
7 * Norway       9.4    0.19  43.5  3.1   1.8
8 * Denmark     11.8   0.41  113.9 10.4  1.7
9 * England      1.8    3.49  113.7 5.1   1.4
10 * Ireland     0.2    3.73  124.5 3.8   1.9
11 * Holland     9.2    0.58  75.5  9.7   2.7
12 * Switzerland 9.1    0.25  73.5  44.9  2.1
13 * France      5.2    0.10  44.5  104.3 2.5
14 * Italy        3.6    0.06  13.6  106.6 2.0
15 * Spain       2.5    0.03  43.6  73.2  2.7
16 * Portugal    2.2    0.03  27.5  89.3  0.9
17 *
18 *REDIM 50,150 / This gives enough space for columns.
19 *TRANSP 4,16,20_
20 *

```

TRANSP on line 19 processes the lines 4-16 and writes the transposed table from line 20 onwards:

```

15 1 SURVO 84C EDITOR Sat Jan 04 19:00:08 1992 D:\P2\TEXT\ 50 150 0
14 * Italy        3.6    0.06  13.6  106.6  2.0
15 * Spain       2.5    0.03  43.6  73.2  2.7
16 * Portugal    2.2    0.03  27.5  89.3  0.9
17 *
18 *REDIM 50,150
19 *TRANSP 4,16,20_
20 *Country Finland Sweden Norway Denmark England Ireland Holland Switzerland
21 *Coffee 12.5 12.9 9.4 11.8 1.8 0.2 9.2 9.1
22 *Tea 0.15 0.30 0.19 0.41 3.49 3.73 0.58 0.25
23 *Beer 54.7 58.3 43.5 113.9 113.7 124.5 75.5 73.5
24 *Wine 7.6 7.9 3.1 10.4 5.1 3.8 9.7 44.9
25 *Spirits 2.7 2.9 1.8 1.7 1.4 1.9 2.7 2.1
26 *

```

5. Arithmetics and spreadsheet computing

Survo provides two different working modes for computing with numbers and tables. Furthermore, the statistical operations and general mathematical tools (like the matrix interpreter) give support in advanced applications. In this section, however, we are mainly considering typical elementary calculations.

Editorial computing permits calculating values for various expressions which have been typed in the edit field. Even more complicated formulae or *computation schemes* (consisting of formulae with instructions) may be entered and activated with given starting values.

In addition to basic arithmetics, various mathematical and statistical functions are readily available. New functions may also be defined by the user either very easily in the edit field or by programming them in the C language.

The functions for editorial computing are also available in the VAR operation (described later) for tasks related to spreadsheet computing. The main applications of VAR are transformations of variables in statistical data sets.

As an extension to editorial computing, various **numerical conversions** can be performed directly in the edit field. These conversions include physical measurements (i.e. lengths, areas, volumes, weights, time, etc.), currency, integers from a base to another, ASCII characters, Roman numerals, decimal numbers to fractions, and integers to prime factors.

Touch mode is another unique approach in Survo for calculations. In this mode, various computations are carried out by moving the cursor to touch any number in the edit field and the number is activated by pressing any of the keys $+$ $-$ $*$ or $/$ which correspond to standard arithmetical operations. Several numbers can be activated in this way and the resulting numerical expression will appear at each stage on the bottom line of the screen.

To print the current result in the edit field, the cursor is moved to indicate the desired position for output and the key $=$ is pressed.

Touch mode gives powerful tools for spreadsheet computing, too. Any systematic computing sequence consisting of several steps can be first defined simply by carrying out calculations in touch mode for the first case. After this definition stage the sequence may be automatically repeated for remaining cases by a single activation. These computation sequences, *touch chains*, can also be saved and used later when needed.

Touch mode provides the same mathematical and statistical library functions as editorial computing.

Editorial computing is always available in editorial mode. Touch mode is entered by pressing key $\boxed{\text{TOUCH}}$ (F3). Return from touch mode to editorial mode takes place simply by pressing $\boxed{\text{ENTER}}$.



MATH?
ARIT?
MATARIT?

5.1 Editorial computing: arithmetics

Arithmetic expressions are typed in the edit field according to normal mathematical notation.

For example, to calculate the arithmetic mean of numbers 12, 17, and 25 we enter:

```
22 1 SURVO 84C EDITOR Sat Jan 03 17:08:54 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *
3 *      (12+17+25)/3=_
4 *
5 *
```

Now, when the cursor is blinking immediately after =, we press the activation key **[ESC]**. Because there is no command on the current line, the Survo Editor studies the character just before the cursor position. If it is = as in this case, the editor assumes that the user wants to calculate something and calls the editorial computing module which analyzes the current expression, computes its value and writes the result in the edit field. Finally the control is transferred back to the editor and we may continue the work.

In this case we immediately obtain the following display:

```
22 1 SURVO 84C EDITOR Sat Jan 03 17:08:54 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *
3 *      (12+17+25)/3=18
4 *
5 *
```

If the geometric mean of the same numbers is then desired, we continue by typing, for example,

```
26 1 SURVO 84C EDITOR Sat Jan 03 17:22:15 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *
3 *      (12+17+25)/3=18
4 *      (12*17*25)^(1/3)=_
5 *
```

and activate by pressing **[ESC]** again. Then we shall have

```
26 1 SURVO 84C EDITOR Sat Jan 03 17:22:15 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *
3 *      (12+17+25)/3=18
4 *      (12*17*25)^(1/3)=17.2130062073
5 *
```

Since after each activation we are back in editorial mode, it is easy to round and edit the results, change the numbers, expressions and activate again...

When the same numbers are used for several computations or when more general expressions are wanted, we may also use symbolic notation and type

```
31 1 SURVO 84C EDITOR Sat Jan 03 17:45:43 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *      X=12 Y=17 Z=25
3 *      Arithmetic mean is (X+Y+Z)/3=_
4 *      Geometric mean is (X*Y*Z)^(1/3)=
5 *
```


After activating both expressions we get the following display.

```
34 1 SURVO 84C EDITOR Sat Jan 03 17:45:43 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *      X=12 Y=17 Z=25
3 * Arithmetic mean is (X+Y+Z)/3=18
4 * Geometric mean is (X*Y*Z)^(1/3)=17.2130062073
5 *
```

In the preceding example we activated the expressions separately. To get updated values of several expressions simultaneously, we use characters `. =` instead of mere `=` in each and activate only one of them. Then the other expressions (having `. =`) will also be recomputed.

Thus, our example can be rewritten in the form:

```
35 1 SURVO 84C EDITOR Sat Jan 03 18:02:35 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *      X=12 Y=17 Z=25
3 * Arithmetic mean is (X+Y+Z)/3.=18
4 * Geometric mean is (X*Y*Z)^(1/3).=17.2130062073
5 *
```

Observe that we have also changed the value of Z and hence the current values of the means displayed above cannot hold true anymore. However, if we now activate, for example, the latter expression, the former one will also be updated as follows:

```
35 1 SURVO 84C EDITOR Sat Jan 03 18:02:35 1987 D:\P2\ARIT\ 100 100 0
1 *
2 *      X=12 Y=17 Z=23
3 * Arithmetic mean is (X+Y+Z)/3.=17.3333333333
4 * Geometric mean is (X*Y*Z)^(1/3).=16.7411775028
5 *
```

Numerical accuracy

Numerical computations are performed in Survo in double precision (i.e. within 15-16 correct digits). The default accuracy when displaying results in editorial computing is about 14 digits. The accuracy can be altered by an ACCURACY specification. Suitable values for ACCURACY are from 4 to 16.

The next display shows how ACCURACY works.

```
5 1 SURVO 84C EDITOR Tue Jan 14 11:15:58 1992 D:\P2\ARIT\ 100 100 0
1 *
2 *1/3=0.33333333333333
3 *.....
4 *ACCURACY=4
5 *1/3=0.3333
6 *.....
7 *ACCURACY=16
8 *1/3=0.33333333333333
9 *
```



MULTIPLE?

5.2 Editorial computing: functions

In editorial computing, the standard mathematical functions like sqrt, exp, log, sin, etc. are always available. In addition to these predetermined functions, library functions, which are saved on disk can be used. Their availability depends on the current installation. The user may create new library functions by programming them in C (See FUNC? N). Furthermore, new functions for temporary applications are most easily defined in the edit field.

Standard functions

| Function | | Examples |
|-----------|--|--|
| int(x) | integer part | int(5.35)=5 int(-5.35)=-6 |
| abs(x) | absolute value | abs(-3.7)=3.7 |
| sgn(x) | 1, if x>0 0, if x=0 -1, if x<0 | sgn(+3.7)=1 sgn(0)=0 sgn(-3.7)=-1 |
| ind(x) | 1, if x>0 0, if x<=0 | ind(3.7)=1 ind(-3.7)=0 |
| max() | maximum value | max(2,5,1)=5 |
| min() | minimum value | min(2,5,1)=1 |
| sqrt(x) | square root | sqrt(1/2)=0.70710678118655 |
| log(x) | natural logarithm | log(2)=0.69314718055995 |
| exp(x) | exponent function | exp(1)=2.718281828459 |
| sin(x) | sine function | pi=3.14159265358979 sin(pi/4)=0.70710678118655 |
| cos(x) | cosine function | cos(pi/6)=0.86602540378444 |
| tan(x) | tangent function | tan(pi/4)=1 |
| arcsin(x) | arc sine | 6*arcsin(0.5)=3.1415926535898 |
| arccos(x) | arc cosine | 3*arccos(0.5)=3.1415926535898 |
| arctan(x) | arc tangent | 4*arctan(1)=3.1415926535898 |
| C(m,n) | binomial coefficient | C(10,5)=252 |
| GCD(m,n) | greatest common divisor | GCD(144,270)=18 |
| mod(m,n) | m modulo n | mod(20,7)=6 |
| rnd(x) | random number on (0,1) | rnd(0)=0.49105834960938 rnd(0)=0.75527954101563 |
| probit(x) | inverse of standard normal distribution | probit(0.5)=0.00000010100668 N.G(0,1,0.5)=0 |

FUNCTIONS?
XFUNC?



SHORTF?



RANDOM?
New generators

RNDTEST?

Remarks:

1. In rnd(x), if x>0, a fixed seed number x is used but if x=0, the seed number is selected according to current clock time.
2. The probit function is fast but not very accurate. A better approximation is given by the library function N.G(0,1,x).

The following library functions belong to the standard Survo installation. All of them are saved in the subdirectory `.\F` of the Survo system.

Library functions

LIBRARY FUNCTIONS?

| Function | | Examples |
|--|----------------------|--|
| <code>fact(N)</code> | N factorial N! | <code>fact(fact(3))=720</code> |
| <code>fact.L(N)</code> | <code>log(N!)</code> | <code>fact.L(1000)=5912.1281784882</code> |
| Binomial distribution bin(N,P): | | |
| <code>Bin.f(N,P,x)</code> | probability for x | <code>Bin.f(4,1/2,0)=0.0625</code> |
| <code>Bin.F(N,P,x)</code> | distribution func. | <code>Bin.f(4,1/2,1)=0.25</code> |
| <code>Bin.G(N,P,p)</code> | inverse distr.f. | <code>Bin.F(4,1/2,1)=0.3125</code> <code>Bin.G(4,1/2,0.3125)=1</code> |
| Poisson distribution Poisson(a): | | |
| <code>Poisson.f(a,x)</code> | probability for x | <code>Poisson.f(10,10)=0.12511003572113</code> |
| <code>Poisson.F(a,x)</code> | distribution func. | <code>Poisson.F(10,10)=0.58303975019299</code> |
| <code>Poisson.G(a,p)</code> | inverse distr.f. | <code>Poisson.G(10,0.583)=10</code> |
| Normal distribution N(m,s²): | | |
| <code>N.f(m,s²,x)</code> | density function | <code>N.f(0,1,0)=0.39894228040143</code> |
| <code>N.F(m,s²,x)</code> | distribution func. | <code>N.F(0,1,2)=0.97724986805182</code> |
| <code>N.G(m,s²,p)</code> | inverse distr.f. | <code>N.G(0,1,0.97724986805182)=2</code> |
| t distribution with n degrees of freedom: | | |
| <code>t.f(n,x)</code> | density function | <code>t.f(10,0)=0.38910838396603</code> |
| <code>t.F(n,x)</code> | distribution func. | <code>t.F(10,2)=0.96330598261463</code> |
| <code>t.G(n,p)</code> | inverse distr.f. | <code>t.G(10,0.96330598261463)=2</code> |
| χ^2 distribution with n degrees of freedom: | | |
| <code>Chi2.f(n,x)</code> | density function | <code>Chi2.f(10,10)=0.08773368488393</code> |
| <code>Chi2.F(n,x)</code> | distribution func. | <code>Chi2.F(10,10)=0.55950671493479</code> |
| <code>Chi2.G(n,p)</code> | inverse distr.f. | <code>Chi2.G(10,0.55950671493479)=10</code> |
| F distribution with m,n degrees of freedom: | | |
| <code>F.f(m,n,x)</code> | density function | <code>F.f(3,5,1)=0.36117447894228</code> |
| <code>F.F(m,n,x)</code> | distribution func. | <code>F.F(3,5,1)=0.53514521000637</code> |
| <code>F.G(m,n,p)</code> | inverse distr.f. | <code>F.G(3,5,0.53514521000637)=1</code> |

Functions **N**, **t**, **Chi2**, and **F** have been programmed by Timo Patovaara.

The program of each library function has to be loaded each time from the disk when the function is called. Hence, if the same function is called several times, it is advantageous to copy the program file from the library directory `.\F` to the ramdrive (virtual disk) and call the function with a complete path-name.

For example, the program file for the functions related to normal distribution is `N.EXE`. If this file has been copied to disk `D:`, the distribution function `N.F` may now be called also as `D:N.F`.

If a disk cache memory (like `SMARTDRV.SYS`) is installed, a corresponding enhancement is automatically provided.

It is typical that several related library functions are saved in the same file in directory `.\F`. All functions belonging to a certain file (say `ABC`) have names of type `ABC.DEF`. The user obtains information about the functions belonging to file `ABC` by activating `ABC.X(1)=_` where `X` prompts the `ABC` file to respond by listing all its functions and their definitions; no function value will then be returned.

For example, inquiry for the binomial distribution (`Bin`) leads to a following temporary display:

```

13  1 SURVO 84C EDITOR Sat Jan 10 14:53:55 1987          D:\P2\ARIT\ 120  80  0
 1 *
 2 *
 3 *   Bin.X(1)=_
Binomial distribution Bin(N,P): N integer>0, 0<P<1
Bin.f(N,P,x) probability for x (x=0,1,2,...,N)
Bin.F(N,P,x) distribution function
Bin.G(N,P,p) inverse distribution function, 0<p<1
Press any key!
 9 *
```

Thus it is easy to examine library functions in `.\F` by first listing all the function files (by `>DIR .\F*.EXE`, for example) and then making inquiries for interesting alternatives by the procedure mentioned above.

More library functions may be created for any application by programming them in C. The procedures required for this are described separately (See `FUNC?`).

Temporary functions

For temporary purposes, functions may easily be defined during the work. These functions can be called only in the edit field where they have been defined. It is simple, however, to copy definitions from a field to another and use them in various contexts. Temporary functions are used mostly in connection with computation schemes.

A temporary function is written in the edit field as any expression in the following form

```
<name_of_function>( <list_of_parameters> ) :=<expression>
```

In our example on arithmetic and geometric means, temporary functions could have been defined as follows:

```

21 1 SURVO 84C EDITOR Sun Jan 04 18:27:00 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 * Arithmetic mean AM(X,Y,Z):=(X+Y+Z)/3
4 * Geometric mean GM(X,Y,Z):=(X*Y*Z)^(1/3)
5 *
6 * AM(12,17,25).=
7 * GM(12,17,25).=
8 *
9 * A=11.5 B=14.7 C=16.1
10 * AM(A,B,C).=
11 * GM(A,B,C).=
12 * AM(A+1,B+1,C+1).=
13 * GM(A+1,B+1,C+1).=_
14 *

```

The definitions of function **AM** and **GM** appear on lines 3 and 4. On lines 6-13, several expressions using these functions have been written.

To compute these expressions, it is enough (since all of them are tailed by `.=`) to activate just one of them and we shall have

```

21 1 SURVO 84C EDITOR Sun Jan 04 18:27:00 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 * Arithmetic mean AM(X,Y,Z):=(X+Y+Z)/3
4 * Geometric mean GM(X,Y,Z):=(X*Y*Z)^(1/3)
5 *
6 * AM(12,17,25).=18
7 * GM(12,17,25).=17.2130062073
8 *
9 * A=11.5 B=14.7 C=16.1
10 * AM(A,B,C).=14.1
11 * GM(A,B,C).=13.9619801763
12 * AM(A+1,B+1,C+1).=15.1
13 * GM(A+1,B+1,C+1).=14.971612979
14 *

```

5.3* Editorial computing: control statements

Although editorial computing is no real substitute for a general programming language, it is worthwhile to have some control structures which are helpful in more advanced applications.

If statement

Conditional expressions are written as follows:

```

6 1 SURVO 84C EDITOR Sun Jan 04 19:11:19 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 * B=if(rnd(0)<0.3)then(1)else(2)
4 *
5 * B=_ (B will be 1 with probability 0.3 and 2 with probability 0.7)
6 *
7 *

```

In this example B will get either of the two values (1 and 2) depending on the condition $\text{rnd}(0) < 0.3$. If the condition is true, i.e. the number sampled randomly from the interval (0,1) is less than 0.3, B will get value 1. Otherwise it will be 2.

Thus, when you activate B on line 5 repeatedly, you will see its value fluctuating randomly. Please, note that in this case the **if** statement is not necessary, since B could also be defined more elegantly as $B = \text{int}(1.3 + \text{rnd}(0))$.

In the **if** statement, the condition is always a relation between two expressions, say A and B, with the following alternatives:

$A < B$, $A \leq B$, $A = B$, $A \geq B$, $A > B$, and $A \neq B$.

For more examples on **if** statements, see the VAR operation.

'For' statement

Editorial computing includes also a **for** statement for computing sums, products, maxima and minima of expressions.

For example, the N factorial (which already exists as the library function $\text{fact}(N)$) could be defined as a temporary function

```
17 1 SURVO 84C EDITOR Thu Jan 08 10:46:48 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 * factorial(N):=for(I=1)to(N)product(I)
4 *
5 * factorial(10)=3628800
6 * factorial(factorial(3))=720
7 * factorial(0)=1
8 *
```

Similarly the sum $S = 1 + 1/2 + 1/3 + 1/4 + \dots + 1/3000$ is computed as

```
6 1 SURVO 84C EDITOR Thu Jan 08 10:52:28 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 * S:=for(I=1)to(3000)sum(1/I)
4 *
5 * S=8.58374989
6 *
```

In the next display, the well-known alternating series by Leibniz (1673) for $\pi/4$ is defined as a temporary function using the **for** statement. Also some partial sums are calculated.

```
12 1 SURVO 84C EDITOR Sat Jan 18 13:46:56 1992 D:\P2\ARIT\ 100 100 0
1 *
2 * pi/4=1-1/3+1/5-1/7+1/9-1/11+...+(-1)^(I-1)/(2*I-1)+...
3 *
4 * S(N):=for(I=1)to(N)sum((-1)^(I-1)/(2*I-1))
5 * 4*S(100)=3.1315929035586
6 * 4*S(1000)=3.1405926538398
7 * 4*S(10000)=3.14149265359
8 *
```

Many sources (like Wells 1985) comment on this series i.e. by saying that "the series is remarkable for its simplicity, but it is hopelessly inefficient as a means for calculating π , because so many hundreds of terms must be calculated to obtain even a few digits of π ."

This is literally true as seen from our display, but at the same time one should observe that the series as such is suffering of a systematic bias of size $1/N$ since for $N=100$ the second decimal, for $N=1000$ the third decimal, and for $N=10000$ the fourth decimal is out of its place by one, but then follows 4, 5, and 6 right digits, respectively. Thus, by making this simple correction, already for $N=100$ we obtain 7 correct digits!

The general structure of the **for** statement is

`for (I=I1) to (I2) <operation> (<expression>)`

where permitted operation words and their functions are

| | |
|----------------------|----------------------------------|
| <code>sum</code> | sum of values of expression, |
| <code>product</code> | product of values of expression, |
| <code>max</code> | maximum of values of expression, |
| <code>maxi</code> | index I of the maximum value, |
| <code>min</code> | minimum of values of expression, |
| <code>mini</code> | index I of the minimum value. |

As an example of using operation words `max` and `maxi`, we compute the maximal probability in binomial distribution and the mode of the same distribution as the index that gives this maximum:

```

41 1 SURVO 84C EDITOR Thu Jan 08 11:27:57 1987 D:\P2\ARIT\ 100 100 0
60 *
61 * Mode of the Binomial distribution Bin(N,P):
62 *
63 * Pmax(N,P):=for(I=0)to(N)max(Bin.f(N,P,I))
64 * Mode(N,P):=for(I=0)to(N)maxi(Bin.f(N,P,I))
65 *
66 * Let N=10 and P=0.2
67 * Then Pmax(N,P)=.301989888 Mode(N,P)=2
68 *

```

In some cases, the expression concerned in the **for** statement may be rather complicated and its value is easier to compute stepwise or iteratively.

For example, in the series expansion of the exponential function $\exp(X) = 1 + X + X^2/2! + X^3/3! + \dots + X^N/N! + \dots$

the general term $A(X,N) = X^N/N!$ can be written as

$A(X,N) = A(X,N-1) * X/N$, for $N=1, 2, \dots$

In situations like this, an extended version of the **for** statement is often useful and leads to faster execution. We define the sum of N first terms as

`expn(X,N) := for (I=0) to (N) term(T=1) sum(T*X/I)`.

There `term(T=1)` defines an auxiliary variable T and initiates it to value 1. Similarly, `sum(T*X/I)` means that before summing, the current value of T is updated multiplying by X/I . However, for the first I value ($I=0$) the initial value of T ($T=1$) is used as such.

The display below shows the definition of `expn ()` and some applications of it.

```

19 1 SURVO 84C EDITOR Sat Jan 10 15:50:27 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *      Serial expansion of exp(X):
3 *      expn(X,N):=for(I=0)to(N)term(T=1)sum(T*X/I)
4 *
5 *      x=5
6 *      expn(x,1).=6
7 *      expn(x,2).=18.5
8 *      expn(x,5).=91.4166666667
9 *      expn(x,10).=146.3806010251
10 *      expn(x,15).=148.4029173714
11 *      expn(x,20).=148.4131470674
12 *      expn(x,30).=148.4131591026
13 *      exp(x).=148.4131591026
14 *

```

**Recursive functions

Editorial computing allows even recursive definitions for numeric functions, although it is hardly an economical approach for practical calculations. The N factorial could be defined recursively as

```

17 1 SURVO 84C EDITOR Sat Jan 10 16:42:37 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 *      factorial(N):=if(N=1)then(1)else(N*factorial(N-1))
4 *
5 *      factorial(10)=3628800
6 *      factorial(factorial(3))=720
7 *      factorial(0)=_ (This leads to stack overflow! Why?)
8 *

```

Also binomial coefficients could be defined recursively:

```

12 1 SURVO 84C EDITOR Sat Jan 10 16:52:50 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *
3 *      Binomial coefficients:
4 *      BC(N,M):=if(M=0)then(1)else(if(M=N)then(1)else(BC(N-1,M-1)+BC(N-1,M)))
5 *
6 *      BC(5,2)=10
7 *      BC(12,6)=924
8 *      BC(14,7)=3432 (takes 29 seconds on a 25 MHz PC)
9 *

```



ARITED?

5.4 Computation schemes

It is always up to the user how to organize the computations when working with the Survo Editor. In many applications, all the formulas and operations needed for reaching a specific goal can be expressed as a **computation scheme** which to some extent resembles a computer program. One clear distinction is, however, that in a computation scheme there is no specific order of statements.

Each activation in a Survo work scheme leads always to a search process where the editor and other programs called for help are looking for the information needed for carrying out the task.

In fact, all the previous examples of editorial computing have been computation schemes in a modest sense. A true computation scheme, however, usually contains instructions and comments to help the user in applying the scheme.

For example, testing of a correlation coefficient by using Fisher's z transformation could be represented as a computation scheme as follows:

```

33 1 SURVO 84C EDITOR Sun Jan 11 12:22:39 1987      D:\P2\ARIT\ 100 100 0
47 * Testing the correlation coefficient
48 * The sample correlation coefficient is r and the sample size n.
49 * To test the hypothesis that in the population the unknown
50 * correlation coefficient rho is r0 against the alternative rho>r0,
51 * we form the test statistic
52 *      U=sqrt(n-3)*(Fisher(r)-Fisher(r0))
53 * where
54 *      Fisher(r):=0.5*log((1+r)/(1-r))
55 * is Fisher's transformation of the correlation coefficient.
56 *
57 * If the null hypothesis is true, U is approximately N(0,1).
58 * Hence we reject the hypothesis, if P=1-N.F(0,1,U)
59 * is less than the risk level (say 0.05).
60 *
61 * Assume now that n=25, r=0.85 and r0=0.7
62 *
63 * Then U.=1.8238788825   and P.=0.03408519244644
64 *
65 * Thus if P<0.05, we reject the hypothesis that correlation
66 * coefficient in the population is r0.=0.7
67 * .....
68 * Instructions: Insert your own values on line 61 and
69 *      activate P.= on line 63, for example.

```

As another example we consider the solving of an equation of second degree:

```

10 1 SURVO 84C EDITOR Sat Jan 10 18:03:11 1987      D:\P2\ARIT\ 100 100 0
24 * .....
25 *
26 * The roots of the equation A*Z^2+B*Z+C=0 are computed as follows:
27 *
28 * Let the roots be Z1=X1+i*Y1 and Z2=X2+i*Y2.
29 * The sign of the discriminant D=B*B-4*A*C determines whether
30 * the roots are complex or not.
31 *
32 * Thus X1=if(D>=0)then((-B+sqrt(D))/(2*A))else(-B/(2*A))
33 *      X2=if(D>=0)then((-B-sqrt(D))/(2*A))else(-B/(2*A))
34 *      Y1=if(D>=0)then(0)else(sqrt(-D)/(2*A))
35 *      Y2=if(D>=0)then(0)else(-sqrt(-D)/(2*A))
36 *
37 * Let A=1, B=-9 and C=14.
38 * Then (activate any of the items below)
39 *      X1.=7          Y1.=0
40 *      X2.=2          Y2.=0
41 *      D.=25
42 *
43 * .....

```

Often it is profitable to collect several related computation schemes and other work schemes together into the same edit field. Such schemes have to be separated into distinct subfields by border lines (like lines 24 and 43 in the preceding example). If the schemes have common definitions for constants and temporary functions etc., a global subfield (indicated by the keyword *GLOBAL* somewhere in that subfield) is employed to store them (see GLOBAL?).

The precedence order of definitions (specifications) is:

1. current line
2. current subfield
3. *GLOBAL* subfield (must appear as the first subfield)
4. default value



SPECIF?
SPECS?
BORDER?

No default values are admitted in editorial computing. Even universal constants like $\pi=3.14159265358979\dots$ must be given in the current job.

5.5 Numerical conversions

As an extension of editorial computing, various numerical conversions are performed by activating expressions like

```
13 1 SURVO 84C EDITOR Sun Jan 12 14:01:43 1992 D:\P2\ARIT\ 100 100 0
1 *
2 *440(yard:meter)=_
3 *
```

giving 440 yards as meters:

```
13 1 SURVO 84C EDITOR Sun Jan 12 14:01:43 1992 D:\P2\ARIT\ 100 100 0
1 *
2 *440(yard:meter)=402.336
3 *
```

The general form of a conversion is

<number> (<unit1>:<unit2>)=_

In place of <number>, no formal expressions are accepted. If <number> is missing, '1' is used as default. For example,

```
13 1 SURVO 84C EDITOR Sun Jan 12 14:02:23 1992 D:\P2\ARIT\ 100 100 0
1 *
2 * 1(yard:meter)=0.9144 (yard:meter)=0.9144
3 *
```

Multiple activations occur along the rules of editorial computing by using '=' instead of '='. For example, activation in

```
13 1 SURVO 84C EDITOR Mon Oct 26 08:55:34 1992 D:\P2\ARIT\ 100 100 0
1 * ACCURACY=7
2 *(pound:kg).=_ 1000(grain:g).= (ounce:grain).=
3 *
```

gives

```
13 1 SURVO 84C EDITOR Mon Oct 26 08:55:34 1992 D:\P2\ARIT\ 100 100 0
1 * ACCURACY=7
2 *(pound:kg).=0.4535854 1000(grain:g).=64.797917 (ounce:grain).=437.5
3 *
```

If conversions between unsuitable units like (kg:m)= is attempted, an error message 'Conversion from kg to m is impossible!' is displayed. When appropriate, the prefixes of the International System like 'mega' and 'micro' are recognized. Examples:

```

13 1 SURVO 84C EDITOR Sun Jan 12 14:05:01 1992 D:\P2\ARIT\ 100 100 0
1 *
2 * (megaton:ton)=1000000
3 * (nanometer:Ångström)=1
4 * (femtosecond:sec)=1e-015
5 *

```

Valid prefixes and their effects in powers of 10

| | | | |
|-------|-----|-------|----|
| atto | -18 | hecto | 2 |
| femto | -15 | k | 3 |
| pico | -12 | kilo | 3 |
| nano | -9 | M | 6 |
| micro | -6 | mega | 6 |
| m | -3 | giga | 9 |
| milli | -3 | tera | 12 |
| centi | -2 | peta | 15 |
| deci | -1 | exa | 18 |
| deca | 1 | | |

The names of measurement units and their relations to other units are defined in alphabetic order as a table in the edit file **MEASURES** in the subdirectory **.\SYS** of the Survo system path. The structure of the table is described in the same edit file. One can add new items to that table when necessary.

Besides physical measurements, Survo conversions apply to currencies, number systems, ASCII codes, Roman numerals, decimal numbers (into fractions), and integers (to prime factors), etc. In sequel, information on each category is given mostly by examples.

5.5.1 Physical measurements

Length:

```

12 1 SURVO 84C EDITOR Mon Oct 26 08:57:20 1992 D:\P2\ARIT\ 150 100 0
5 *
6 *(m:meter).=1 Base length unit
7 *(dm:m).=0.1 (decimeter:m).=0.1 (cm:m).=0.01 (centimeter:m).=0.01
8 *(mm:m).=0.001 (millimeter:m).=0.001 (dmm:mm).=0.1
9 *(km:m).=1000 (kilometer:m).=1000 (Ångström:m).=0.0000000001
10 *(light-year:megakm).=9460529.503
11 *(inch:cm).=2.54 (ft:inch).=12 (feet:inch).=12
12 *(yard:ft).=3 (fathom:ft).=6 (rod:ft).=16.5 (hand:inch).=400
13 *(chain:yard).=22 (furlong:chain).=10 (mile:furlong).=8 (mile:yard).=1760
14 *(nautical_mile:yard).=2025.3718285214 (league:mile).=3
15 *Measures in typesetting:
16 *(inch:Point).=72 (pica:Point).=6 (point:Point).=0.996264
17 *

```

Area:

```

19 1 SURVO 84C EDITOR Sun Jan 12 18:29:00 1992 D:\P2\ARIT\ 150 100 0
19 *
20 *(m2:square_meter).=1 Base area unit
21 *(are:m2).=100 (a:m2).=100
22 *(hectare:m2).=10000 (ha:m2).=10000
23 *(square_kilometer:m2).=1000000 (km2:m2).=1000000
24 *
25 *(acre:m2).=4046.8564224 (acre:square_rod).=160
26 *(square_yard:square_inch).=1296
27 *

```

Volume:

```

12 1 SURVO 84C EDITOR Sun Jan 12 19:33:22 1992 D:\P2\ARIT\ 150 100 0
29 *
30 *(L:liter).=1 Base volume unit ACCURACY=7
31 *(cubic_decimeter:L).=1 (dm3:L).=1 (dL:L).=0.1
32 *(cubic_meter:L).=1000 (m3:L).=1000
33 *(cubic_millimeter:L).=0.000001 (mm3:L).=0.000001
34 *
35 *(cubic_feet:L).=28.316847 (feet3:cubic_feet).=1
36 *(cubic_feet:cubic_inch).=1728000 (bushel:inch3).=1848000
37 *(dry_pint:inch3).=33600.3 (dry_quart:dry_pint).=2
38 *(liquid_pint:inch3).=28875 (pint:liquid_pint).=1
39 *(peck:inch3).=537604.8 (gill:inch3).=7218.75
40 *(gallon:inch3).=231000 (gallon:pint).=8
41 *

```

Mass and weight:

```

16 1 SURVO 84C EDITOR Mon Jan 13 12:12:40 1992 D:\P2\ARIT\ 150 100 0
43 *
44 *(kg:kilogram).=1 Base mass (weight) unit ACCURACY=7
45 *(g:kg).=0.001 (ton:kg).=1000 (carat:g).=0.2
46 *
47 *(ounce:kg).=0.0283491 (oz:ounce).=1 (oz_av:oz).=1 (oz_ap:oz).=1.0971429
48 *(pound:lb).=1 (lb:oz).=16 (lb_av:lb).=1 (lb_t:oz).=13.165714
49 * (lb_ap:oz_ap).=12 (lb:scruple).=350
50 *(short_ton:lb).=2000 (long_ton:lb).=2240
51 *(oz_ap:pennyweight).=20 (pennyweight:scruple).=1.2
52 *
53 *(kp:kg).=1 mass/weight relation in this context
54 *(g:dyn).=980.665 (newton:dyn).=100000
55 *

```

Work, power and pressure:

```

10 1 SURVO 84C EDITOR Mon Jan 13 12:33:26 1992 D:\P2\ARIT\ 150 100 0
57 *
58 *Work:
59 *(Cal:J).=4.1868 (cal:J).=4.1868 (calorie:J).=4.1868 (erg:J).=0.0000001
60 *(eV:J).=1.602e-019 (GeV:J).=1.6023 (J:J).=1 (joule:J).=1
61 *(kCal:J).=4186.8 (kcal:J).=4186.8 (keV:J).=1.602e-016 (kJ:J).=1000
62 *(kWh:J).=3600000 (MeV:J).=0.000000000000016
63 *
64 *Power:
65 *(W:watt).=1 (kW:W).=1000 (horsepower:kW).=0.7355 (hp:horsepower).=1
66 *
67 *Pressure:
68 *(Pa:pascal).=1
69 *(bar:Pa).=100000 (at:Pa).=98066.5
70 *(mmHg:Pa).=133.3223684 (atm:mmHg).=760.0000012001 (torr:mmHg).=1
71 *(at:atm).=0.96784110535406
72 *

```

Time, angle and velocity:

```

10 1 SURVO 84C EDITOR Tue Jan 14 17:53:36 1992 D:\P2\ARIT\ 150 100 0
74 *
75 *(sec:s).=1 Base time unit
76 *(min:sec).=60 (hour:min).=60 (day:hour).=24 (year:day).=365.24219
77 *7405(sec:hms).=02:03:25 10:35:55(hms:h).=10.598611111111
78 *(degree:s).=3600
79 *42.5(degree:hms).=42:30:00 180(degree:radian).=3.1415926535898
80 *
81 *1000(km/h:m/s).=277.777777777778
82 *(Mach:m/s).=340.277777777778 (Mach:speed_of_sound).=1
83 *(speed_of_light:m/s)=299792500
84 *(mph:km/h).=1.609344 ( (mile:kilometer).=1.609344 )
85 *

```

5.5.2 Currencies

Conversions of monetary units are performed by using their international three-letter symbols (USD=US Dollar, GBP=British Pound, DEM=German Mark, FIM=Finnish Mark, etc.) as follows:

```

14 1 SURVO 84C EDITOR Fri Jan 17 09:53:38 1992 D:\P2\ARIT\ 150 100 0
87 *
88 * 100(USD:FIM)=437.8
89 * 437.8(FIM:USD)=99.635867091488
90 * (not 100 since buying rate is less than selling rate)
91 *
92 * (GBP:USD)=1.7567137005007
93 * (GBP:DEM)=2.8318291877614
94 *

```

Since exchange rates are fluctuating, they must be updated when necessary. The current rates are kept in the edit field CURRENCY of the subdirectory .\SYS of the Survo system path. When loaded into the edit field it looks like

```

1 1 SURVO 84C EDITOR Fri Jan 17 09:59:17 1992 D:\P2\ARIT\ 50 72 0
2 *SAVE .\SYS\CURRENCY
3 *
4 *Use the UPDATE command (on line 9) for updating exchange rates
5 *in .\SYS\MEASURES.EDT .
6 *If new items (lines, countries) are introduced, the corresponding
7 *lines have to be first inserted in .\SYS\MEASURES.EDT .
8 *
9 * Helsinki, January 16, 1992
10 *UPDATE CUR+1,END,4,.\SYS\MEASURES.EDT
11 *FIM X C 1 1 Finland
12 *USD X C 4.378 4.394 USA
13 *CAD X C 3.790 3.810 Canada
14 *GBP X C 7.719 7.761 Gr.Britain
15 *IEP X C 7.228 7.268 Ireland
16 *SEK X C 0.745 0.749 Sweden
17 *NOK X C 0.690 0.694 Norway
18 *DKK X C 0.7002 0.7032 Denmark
19 *ISK X C 0.0734 0.0784 Iceland
20 *DEM X C 2.7158 2.7258 Germany
21 *NLG X C 2.4124 2.4214 Holland
22 *BEC X C 0.1318 0.1324 Belgium
23 *CHF X C 3.0547 3.0667 Switzerland
24 *FRF X C 0.7947 0.8007 France

```

This table is based on the official information published daily by the Finnish Bank. Therefore the Finnish Mark is used as a base unit (both selling and buying rates are 1). This base unit and the order and selection of currencies in the table can be edited along the user's preferences.

When some alterations is done in the CURRENCY field, the corresponding lines in the MEASURES file must be edited similarly. This takes place most easily by activateing the UPDATE command (on line 9 in the CURRENCY field).

In applications where fresh exchange rates are needed constantly, one can make a sucro for automatic updating.

5.5.3 Number systems

The initial values and the results in Survo calculations are given always in our standard decimal system. In some applications, for example in those related to programming, numbers written in other bases than 10 are needed. Especially, binary (2), octal (8), and hexadecimal (16) representation of numbers is common in such applications.

The next example illustrates conversion between number systems and appropriate notations.

```

17 1 SURVO 84C EDITOR Fri Jan 17 10:34:49 1992 D:\P2\ARIT\ 100 100 0
1 *
2 * Converting integers from a base to another:
3 *
4 * 1988(10:2)=11111000100 1988 from base 10 to base 2
5 * 11111000100(2:10)=1988 11111000100 from base 2 to base 10
6 * 254(10:8)=376 254 is 376 as an octal (base 8) number
7 * 254(10:16)=FE 200 is FE as a hexadecimal number
8 * 7C4(16:10)=1988
9 * FFFF(hex:dec)=65535 hex and dec are allowed instead of 16 and 10
10 * 777(oct:bin)=111111111 oct and bin are allowed instead of 8 and 2
11 *
12 * Max. base is 36. In bases >10, digits are
13 * 0,1,2,3,4,5,6,8,9,A,B,C,D,...,Z .
14 *

```

As more special forms of conversion, integers can be represented as Roman numerals and spelled in words as follows:

```

24 1 SURVO 84C EDITOR Fri Jan 17 20:21:02 1992 D:\P2\ARIT\ 100 100 0
16 *
17 *Converting Roman numerals:
18 *
19 * 1988(10:Roman)=MCMLXXXVIII 1988(10:roman)=mcmclxxxviii
20 * MCMLXXXVIII(Roman:10)=1988 MXXIV(roman:2)=1000000000
21 *
22 *Spelling in words:
23 * 1(10:words)=one
24 * 17(10:words)=seventeen
25 * 1992(10:words)=one thousand nine hundred ninety-two
26 * 4500000(10:words)=four million five hundred thousand
27 * 1000000000(10:words)=one milliard
28 *1000000000100(10:words)=one billion one hundred
29 *1111111111111(2:words)=eight thousand one hundred ninety-one
30 *

```

Also conversion from and to ASCII characters is possible:

```

51 1 SURVO 84C EDITOR Fri Jan 17 20:24:25 1992 D:\P2\ARIT\ 100 100 0
32 *
33 *Converting ASCII characters:
34 *
35 * A(ascii:10)=65 A(ascii:8)=101 A(ascii:16)=41
36 * 65(10:ascii)=A 101(8:ascii)=A 41(16:ascii)=A
37 *

```

5.5.4* Decimal numbers to fractions

A ratio of two integers is simple to compute as a decimal number. The opposite task, to convert a given decimal number into a ratio of two integers is more complicated. Especially a good approximation as a ratio of two *small* integers is not trivial to discover.

The next example shows how such approximations with different degrees of accuracy are found in Survo:



INTREL?

```

26 1 SURVO 84C EDITOR Sat Jan 18 10:34:41 1992 D:\P2\ARIT\ 100 100 0
39 *
40 * Approximations of pi:
41 *
42 * 3.14159265(1:ratio).=3/1 (0.14159265)
43 * 3.14159265(4:ratio).=16/5 (-0.05840735)
44 * 3.14159265(5:ratio).=22/7 (-0.00126449285714)
45 * 3.14159265(11:ratio).=355/113 (-0.00000027035398)
46 *
47 * The last ratios are well-known from the history of mathematics.
48 *

```

For example, in `3.14159265(5:ratio)` parameter '5' refers to the accuracy of approximation. The number in parentheses (`-0.00126449285714`) is the difference of the decimal number and the ratio.

Method of approximation

Our technique in these approximations is based on the theory of musical sounds. When a person with a "musical ear" listens to an interval of two tones, he/she usually recognizes it as one of the basic intervals (fifth, fourth, major third, minor third, etc.) although it may be far from a pure interval. For example, if the frequency ratio is 1.5312 (this was the howling "Wolf" interval of the ancient mean-tone tuning on keyboard instruments), it is perceived as a dissonant fifth having the ideal ratio $3/2=1.5$.

In general, assume that we have an interval of tones with frequencies f and g hz and the ratio $x=f/g$ is approximately n/m where m and n are small integers. The impurity of the interval is revealed by an regular pulsating (beats) having the rate of $ng - mf$ hz. The relative value of the beat with respect to ng is then $(n - mx)/n$. Identifying the interval amounts to selection of integers m and n so that beating is not too rapid and the integers m and n remain as small as possible.

The author formulated this goal in 1972 as an optimization problem:

For given x and c , minimize the function

$$\text{diss}(c,x,m,n)=n+10^c \cdot (mx/n - 1)^2$$

with respect to integers m and n . The constant c describes the accuracy of the ear. In practice c may vary from about 3 (poor) to about 5 (superb). A high c value means that the ear does not tolerate false sounds and, on the other hand, is able to detect very complicated musical intervals (ratios n/m).

Hence, the general interpretation of the conversion from a decimal number x to ratio n/m is in terms above

$$x(c:\text{ratio})=n/m \quad (x-n/m).$$

As an illustration we consider a neutral third:

```

38 1 SURVO 84C EDITOR Sat Jan 18 14:44:49 1992 D:\P2\ARIT\ 100 100 0
50 *
51 * Neutral third is the geometric mean of minor (6/5) and
52 * major third (5/4), i.e. sqrt(5/4*6/5)=sqrt(3/2).
53 *
54 * sqrt(3/2)= 1.2247448713916(3:ratio)=5/4 (-0.0252551286084)
55 *           1.2247448713916(5:ratio)=11/9 (0.00252264916938)
56 *

```

A poor musical ear hears it as a major third but a very good ear recognizes it

as a rather pure interval 11/9. In section 8.11, a graph of dissonance functions is given.

5.5.5 Prime factors of an integer

The next display gives all the essentials for factoring integers in the edit field.

```

38 1 SURVO 84C EDITOR Sat Jan 18 15:19:15 1992 D:\P2\ARIT\ 100 100 0
58 *
59 * Prime factors of an integer:
60 *
61 * 30(10:factors)=2*3*5      37(10:factors)=37
62 * 144(dec:factors)=2^4*3^2
63 * 10000000000000000000(bin:factors)=2^20
64 * 10000(7:factors)=7^4
65 *
66 * Max. number to be factored is 2^32-1=4294967295:
67 * 4294967295(dec:factors)=3*5*17*257*65537
68 *
69 * 4127127127(dec:factors)=7213*572179  572179(dec:factors)=572179
70 * 65520^2-1= 4292870399(dec:factors)=65519*65521 (both primes!)
71 *

```

5.6 Touch mode



TOUCH?

It is not easy to describe the ideas of touch mode on paper since it has its own dynamic nature which is most easily comprehended by doing. Therefore the simplest way to become acquainted with touch mode is by testing it with small examples and using ready-made tutorials.

While working with the Survo Editor, the touch mode is entered by pressing the **TOUCH** (F3) key. In touch mode, the standard display of the current edit field will be seen. Only the top line and the bottom line are changed and text related to touch mode will be displayed on those lines. In touch mode, you can move the cursor, scroll the screen and even type more text (in lower case letters) and numbers in the edit field. There are only some upper case letters and control keys getting an alternate function. Whenever you want to return to the normal editing mode, you press the **ENTER** key and all the text and results obtained in touch mode and written in the edit field will be preserved.

The main idea of the touch mode is to move the cursor in the edit field to touch various numbers, activate them by pressing keys like **+** **-** ***** **/** to indicate the arithmetical operation to be performed and finally write results obtained in the edit field by pressing the **=** key.

After a short learning period (be patient), you will find out that touch mode is an amazingly efficient technique for computing both with individual numbers and arrays of numbers. It is so efficient because it is so straightforward. One should have the feeling that everything is done with minimum effort.

5.6.1 Touch mode: computing with single numbers

Assume that we have entered touch mode by pressing **TOUCH** (F3) in the following situation:

```

5  1 SURVO 84C Touch Mode Sat Jan 17 14:27:53 1987      TOUCH  100 100
1  *
2  *Population statistics:
3  *In 1985 the number of males in Finland was estimated to be 2374000 and
4  *the number of females 2532000. Thus the total population of Finland
5  *was -
6  *
7  *
8  *
9  *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
TOUCH MODE (To stop, press ENTER)

```

The total population $2374000+2532000=4906000$ will now be computed as follows: The cursor is first moved to touch the number 2374000. It may touch the number at any position (for example, 2374000). If now the **+** key is pressed, we shall have the display:

```

61 1 SURVO 84C Touch Mode Sat Jan 17 14:27:53 1987      TOUCH  100 100 +
1  *
2  *Population statistics:
3  *In 1985 the number of males in Finland was estimated to be 2374000 and
4  *the number of females 2532000. Thus the total population of Finland
5  *was
6  *
....
....
21 *
22 *
23 *
+2374000

```

The number activated by **+** is also presented on the bottom line. In touch mode, this line always keeps record on the expression to be evaluated.

We continue by moving the cursor to touch 2532000 and press **+** again:

```

29 1 SURVO 84C Touch Mode Sat Jan 17 14:27:59 1987      TOUCH  100 100 +
1  *
2  *Population statistics:
3  *In 1985 the number of males in Finland was estimated to be 2374000 and
4  *the number of females 2532000. Thus the total population of Finland
5  *was
6  *
....
....
21 *
22 *
23 *
+2374000+2532000.

```

The bottom line shows the sum of the numbers touched so far. Finally the only

thing to do is to move the cursor to indicate the place of the result and press the $\boxed{=}$ key:

```

11 1 SURVO 84C Touch Mode Sat Jan 17 14:28:05 1987 TOUCH 100 100 =
1 *
2 *Population statistics:
3 *In 1985 the number of males in Finland was estimated to be 2374000 and
4 *the number of females 2532000. Thus the total population of Finland
5 *was 4906000
6 *
....
....
21 *
22 *
23 *
TOUCH MODE (To stop, press ENTER)

```

The expression has been evaluated and printed at the place indicated by the cursor. Please note that the cursor points at the first integer position. To return to the normal editing mode, we press the $\boxed{\text{ENTER}}$ key.

When calculating in touch mode, the numbers are touched by keys $\boxed{+}$ $\boxed{-}$ $\boxed{*}$ $\boxed{/}$. These keys correspond to standard arithmetical operations.

For example, if we want to compute $56.3 - 24 / 1.5$ with numbers 56.3, 24, and 1.5 already appearing in the edit field, we enter touch mode by $\boxed{\text{TOUCH}}$, pick 56.3 by $\boxed{+}$, then 24 by $\boxed{-}$ and 1.5 by $\boxed{/}$. Finally we indicate the place for the result and press $\boxed{=}$.

Also some other keys have special functions in touch mode. Pressing of $\boxed{\text{S}}$ evaluates the current expression and displays it on the bottom line but does not print it into the edit field. The result after pressing $\boxed{\text{S}}$ can be treated like a number which has been activated by the $\boxed{+}$ key.

Using of $\boxed{\text{S}}$ permits computing expressions which otherwise would require parentheses. In touch mode we need no parentheses. For example, to compute $(123.9 + 87.3) / 4$ with numbers 123.9, 87.3, and 4, we enter the touch mode, touch 123.9 by $\boxed{+}$ and 87.3 by $\boxed{+}$. Then by pressing $\boxed{\text{S}}$ the current expression $+123.9 + 87.3$ on the bottom line will be replaced by 211.2 and we get the final expression (in the form $211.2 / 4$) by activating 4 by $\boxed{/}$.

The numerical format of the results to be printed in the edit field is selected by pressing $\boxed{\text{F}}$. If $\boxed{\text{F}}$ is pressed in touch mode, a prompt *Touch a format by F!* is temporarily displayed on the bottom line. The user has to move the cursor to touch a number which represents the desired format. For example, 12.123 indicates a format with a two-digit integer part and three decimal places. After touching such an image number with $\boxed{\text{F}}$ *Format 12.123 is selected!* is shown on the bottom line. The cursor automatically retains the original position and operations in touch mode can be continued. Whenever results are printed by pressing $\boxed{=}$, the format is determined by the last one selected by $\boxed{\text{F}}$.

If no format has been selected, the results are printed with maximum accuracy. To select this free format again after a fixed format, touch an empty place (space) by $\boxed{\text{F}}$.

Touch mode supports the same (mathematical, statistical etc.) functions which are available in editorial computing. To compute univariate functions, first the argument is activated by $\boxed{+}$ or computed in touch mode as an expression. Then the $\boxed{@}$ key is pressed which leads to a prompt *Go to touch a function name by '@'!* on the bottom line. Then such a name (like sqrt, exp, sin, etc.) is touched by $\boxed{@}$. The value of the function is then evaluated and displayed on the bottom line like any number which has just been activated by $\boxed{+}$.

In library functions with several arguments, the first argument is always (as in univariate functions) the current expression. The other arguments have to be saved in memory locations 1,2,3,... by the \boxed{M} key.

More information about various touch mode operations can be obtained while staying in touch mode by the \boxed{HELP} (F1) key. See also TOUCH? .

5.6.2 Touch mode: Touch chains

Strongest application areas for touch mode are various tasks related to spreadsheet computing. In addition to single numbers, we can operate in touch mode with regular arrays having many columns and rows. The general principle is to define the calculation desired first for one case (row, column etc.) in touch mode and then let the system repeat the same task for other cases automatically.

The touch chains defined in such a way are like macro operations. They can be saved on disk and edited later by using the TCHLOAD and TCHSAVE commands. In most cases, however, touch chains are defined anew each time they are needed.

The definition of a touch chain is started in touch mode by moving the cursor to the start position of the first (typical) case to be processed. Then the definition stage is entered by pressing the function key $\boxed{F4}$. As an indication of touch chain definition, the word TOUCH on the top line of the screen is replaced by DEF.

All steps required for the first case will then be performed by using touch mode in a normal way and the system will temporarily save all the key strokes. Hence, if the definition is now terminated by the \boxed{EXIT} (F8) key, the touch chain just defined may be automatically repeated for any other case by first moving the cursor to its starting point and by pressing the \boxed{ESC} key.

Another way to interrupt the definition stage is to press the \boxed{ESC} key directly. This immediately leads to a repetitive execution of the chain until an empty place with no number is touched. Execution of a touch chain can also be interrupted manually by pressing the $\boxed{.}$ key.

As an example of a touch chain, we consider calculation of a five term moving average for a time series appearing in the edit field on consecutive lines.

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 D
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 *
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
TOUCH MODE (To stop, press ENTER)

```

To save space in this document, we have decreased the number of visible lines of the edit field in our displays.

In the display above we have already entered the touch mode by the **TOUCH** (F3) key. Furthermore, we have moved the cursor to the first case (on line 6) and pressed the **F4** key. Then DEF will be seen on the top line. We are ready to define the touch chain for the five-term moving average.

In the definition stage we have to calculate the sum of the first five observations $18.7+19.5+18.8+17.9+17.9$, divide it by constant 5 and print the result to the right of the current column on the line of the central observation.

As the last step of the definition stage, we must also indicate the starting point of the next case (i.e. second number 19.5 on line 7).

Hence, to compute the sum, we press the following keys

+ **↓** **+** **↓** **+** **↓** **+** **↓** **+**

and reach the following situation:

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 +
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 *
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
+18.7+19.5+18.8+17.9+17.9

```

On the bottom line is the sum of the 5 first observations. The cursor indicates the last number activated by **+**. Before dividing by 5 we have to evaluate the sum by pressing **S** and the bottom line will be replaced by the sum 92.8. To divide this by 5 we press the **C** (C=constant) key and obtain the following display:

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 C
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 *
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
Go to touch a number as a constant by +,-,* or / !

```

To conform the prompt appearing on the bottom line we now move the cursor to any free place (here to line 5), type the number 5 there, move the cursor one step backwards (to touch 5) and press \square in order to divide with this constant:

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 /
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 * 5
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
92.8/5

```

We have now the first average as an expression on the bottom line. It is printed to the line 8 (which corresponds to the central observation) by moving the cursor to the correct position and then by pressing \square .

```

20 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 =
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 * 5
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8 18.56
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
TOUCH MODE (To stop, press ENTER)

```

As the last step in the definition stage, we move the cursor to indicate the starting point of the next case (line 7, column 13)

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 DEF 100 80 u
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 *
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8 18.56
9 * 1947 17.9
10 * 1948 17.9
11 * 1949 17.3
12 * 1950 15.8
13 * 1951 15.6
14 * 1952 16.3
TOUCH MODE (To stop, press ENTER)

```

and press the `[ESC]` key. This initiates an automatic execution of the touch chain from the current point. At first the second average $(19.5+18.8+17.9+17.9+17.3)/5=18.28$ is computed and printed on line 9 below the first result 18.56.

The system will continue the calculation as long as there are numbers in the edit field in the Temperature column or until we press `[.]`. After the process has terminated, we shall have the display:

```

13 1 SURVO 84C Touch Mode Sat Jan 17 17:16:40 1987 TOUCH 100 80
1 *
2 *Mean temperature (C) in Helsinki, July 1944-
3 *
4 * Year Temperature
5 *
6 * 1944 18.7
7 * 1945 19.5
8 * 1946 18.8 18.56
9 * 1947 17.9 18.28
10 * 1948 17.9 17.54
11 * 1949 17.3 16.9
12 * 1950 15.8 16.58
13 * 1951 15.6 16.42
14 * 1952 16.3 16.5
TOUCH MODE (To stop, press ENTER)

```

While staying in touch mode, the last touch chain defined can be reactivated by selecting a new starting point and pressing `[ESC]`. In this case we could go to the new 5-term average column and activate by `[ESC]`. Then one more column would appear to the right as 5-term averages of 5-term averages.

The chain may also be saved on disk by typing the name of the chain (a complete pathname like `d:average5` in lowercase letters) and touching it by `[T]`. Similarly any touch chain saved earlier by `[T]` may be called first by typing its pathname and touching it by `[L]`. The called chain is then activated again by `[ESC]`. All the touch chains on disk have the extension `.TCH` in their filenames.

5.6.3* Touch mode: Editing touch chains

A touch chain which has been saved under touch mode on disk by the `[T]` key can be called to the edit field and edited by the Survo Editor.

For example, the touch chain `AVERAGE5` which was defined and saved in the preceding chapter, is now be called (in editorial mode) by the `TCHLOAD`

D:AVERAGE5 command with the following result:

```

19  1 SURVO 84C EDITOR Sun Jan 18 14:22:11 1987 D:\P2\ARIT\ 120 80 0
1  *
2  *Mean temperature (C) in Helsinki, July 1944-
3  *
4  *TCHLOAD D:AVERAGE5
5  *+ d + d + d + d + S C5 / u2 r7 = 17 u CTNUE
6  *
7  * Year   Temperature
8  *
9  * 1944      18.7
10 * 1945      19.5
11 * 1946      18.8   18.56
12 * 1947      17.9   18.28
13 * 1948      17.9   17.54
14 * 1949      17.3   16.9
15 * 1950      15.8   16.58
16 * 1951      15.6   16.42
17 * 1952      16.3   16.5

```

After activating the TCHLOAD command, the touch chain AVERAGE5 can be seen on the next line(s) as a sequence of code words. In this program code the key touches used in the definition stage are coded in the following way.

The numbers and letters are represented as such but various special keys are converted into a more legible form.

The arrow keys are coded as

| | |
|---|-------------|
| d | down arrow |
| u | up arrow |
| r | right arrow |
| l | left arrow |

Consecutive strokes are abbreviated as r7 (7 steps to the right).

Other codes in use are:

| | |
|---------|--|
| C5 | Enter constant 5. |
| @log | Compute natural logarithm, similarly @sqrt, @exp, @sin, etc. |
| F12.123 | Enter format 12.123 for results. |
| CTNUE | Key ESC (at the end of the chain) |
| END | Key EXIT (F8) (at the end of the chain) |
| P | Store the current cursor position. When P is pressed again, the cursor returns to the stored position. |
| R | Repeat the current steps of the chain as long as possible. Thereafter continue definition. |
| M | Evaluate the current expression and save it in one of the memory locations 0,1,2,...,9. |
| K | Activate one of the memory locations. |

After editing, the touch chain may be saved again by changing TCHLOAD to TCHSAVE and activating by **ESC**. The edited chain may then be called in touch mode by touching its name by the **L** key.

It is always better to define a touch chain by doing, not by typing some pe-

cular code words. However, in complicated chains it is good to have the possibility of making improvements without repeating the whole definition stage again. Therefore Survo provides TCHLOAD and TCHSAVE commands for touch chain editing.

Here we can also use the legible code for examining some typical touch chains on paper. For example, these simple chains are usually defined again each time they are needed:

Sum of numbers in a column which ends at an empty line:

+ d R = CTNUE

Column sums in a table (columns 10 steps apart from each other):

P + d R = P r10 CTNUE

The previous chain can also be written for a general numeric table whose columns have different widths as

P + d R = P rn CTNUE

where 'rn' means a jump from current number to the next number to the right. When defining such a chain in touch mode, 'rn' will be achieved by pressing first the **PREFIX** key and then the right arrow.

Row sums in a table with a variable column width:

P + rn R = P d CTNUE

In the following example we compute the ratio of the 'Coffee' and 'Tea' columns from a small table below. This task was already carried out by the C/ command.

In the following display the results and the touch chain used can be seen:

```

14 1 SURVO 84C EDITOR Sun Jan 18 15:24:09 1987 D:\P2\ARIT\ 120 80 0
1 *
2 *Consumption of various beverages in 12 European countries
3 *
4 * Country      Coffee  Tea    Beer  Wine  Spirits
5 *
6 * Finland      12.5   0.15  54.7  7.6   2.7   12.123
7 * Sweden       12.9   0.30  58.3  7.9   2.9   83.333
8 * Norway       9.4    0.19  43.5  3.1   1.8   43.000
9 * Denmark      11.8   0.41  113.9 10.4  1.7   49.474
10 * England      1.8    3.49  113.7 5.1   1.4   28.780
11 * Ireland      0.2    3.73  124.5 3.8   1.9   0.516
12 * Holland      9.2    0.58  75.5  9.7   2.7   0.054
13 * Switzerland 9.1    0.25  73.5  44.9  2.1   15.862
14 * France       5.2    0.10  44.5  104.3 2.5   36.400
15 * Italy         3.6    0.06  13.6  106.6 2.0   52.000
16 * Spain        2.5    0.03  43.6  73.2  2.7   60.000
17 * Portugal     2.2    0.03  27.5  89.3  0.9   83.333
18 *
19 * TCHLOAD RATIO
20 *P + r7 / r34 F12.123 = P d CTNUE
21 *
22 *
23 *

```

The situation above is attained by starting a touch chain definition from the 'Coffee' value 12.5 on line 6. The code words on line 20 describe the actions taken during the definition which has been initiated by **F4**.

First **P** is pressed to locate the starting point. Then the number (12.5) is

activated by $\boxed{+}$ and the cursor is moved to the next column (r7). The second number (0.15) is activated by $\boxed{/}$ giving 12.5/0.15 as the current expression. To indicate the place of the result the cursor is moved 34 steps to the right (r34). Then the format is entered by first pressing \boxed{F} and then (according to a prompt) the format required is typed (here on line 5) and activated by \boxed{F} again. Finally the result is printed by $\boxed{=}$ and the cursor is returned to the starting position by \boxed{P} . In order to continue from the second case, the cursor is moved one step downwards (d) and the automatic repetition follows by pressing the \boxed{ESC} key (code word CTNUE).

After execution, the touch chain has been saved on disk by typing its name (ratio) somewhere and activating it by \boxed{T} while still staying in touch mode. After returning to editorial mode by \boxed{ENTER} , the chain has been loaded by activating the TCHLOAD command on line 19.

As the last example on touch mode applications, we show how Pascal's triangle consisting of the binomial coefficients $C(n,m)$ can be constructed by starting from a line of the form

1 1

(it is the line 5 below corresponding to $n=1$) and by using the well-known iterative formula

$$C(n,m)=C(n-1,m)+C(n-1,m-1), \quad C(n,0)=C(n,n)=1, \quad n=2,3,4, \dots$$

```

15 1 SURVO 84C EDITOR Sun Jan 18 15:37:48 1987          D:\ARIT\ 100 100 0
1 *
2 *TCHLOAD PASCAL
3 *P + rn + d = u R d = P + d = CTNUE
4 *
5 *      1      1
6 *      1      2      1
7 *      1      3      3      1
8 *      1      4      6      4      1
9 *      1      5      10     10     5      1
10 *     1      6      15     20     15     6      1
11 *     1      7      21     35     35     21     7      1
12 *     1      8      28     56     70     56     28     8      1
13 *     1      9      36     84     126    126    84     36     9      1
14 *     1     10     45    120    210    252    210    120    45    10     1
15 *     1     11    55    165    330    462    462    330    165    55    11     1
16 *     1     12    66    220    495    792    924    792    495    220    66    12
17 *     1     13    78    286    715   1287   1716   1716   1287    715    286    78
18 *     1     14    91    364   1001  2002  3003  3432  3003  2002  1001    364
19 *     1     15   105   455   1365  3003  5005  6435  6435  5005  3003   1365
20 *           16   120   560   1820  4368  8008 11440 12870 11440  8008   4368
21 *
22 *
23 *

```



GIF animation

When starting the definition of the touch chain PASCAL, all the lines in the edit field except line 5 have been empty and the starting position has been the first '1' on line 5.

During the definition stage, line 6 (having 1 2 1) has been generated. When the definition stage has terminated, the cursor has been on line 6 and pointing at the first '1' on that line.

All remaining lines from 7 onwards are automatically generated after pressing the **ESC** key.

The automatic execution is stopped on line 20 because then the right edge of the triangle reaches the right edge of the edit field. Observe that the current width of the edit field is 100.

After this first run of the touch chain, it has been saved on disk by typing its name `pascal` and activating it by **T**.

Finally we have returned to editorial mode by **ENTER** and the command `TCHLOAD PASCAL` on line 2 has been activated giving the sequence of touch steps on line 3. By studying the code words on line 3, you can try to define the same chain again. (Please, remember that 'rn' is PREFIX <right arrow> which means 'jump to the next number on the right'.)

6. Data base management

One of the most important application areas of Survo is analysis of statistical data. Structures and procedures in data management are designed mainly for maintenance of statistical data sets. The design principles are, however, so general that many data types encountered in nonstatistical applications can be handled as well.

A data base in Survo is always conceived as a rectangular array having one or more fields (variables) as columns and a set records (observations, cases) as rows. Survo supports a few different representations for such data bases and the practical arrangement of fields and records in them may vary. The user may always select the alternative which is the most suitable for applications at hand.

Possible representations for source data are:

1. Lists of data values in the edit field,
2. Data tables (or data matrices) in the edit field,
3. Data files on disk.

All these forms of data also have variants of their own. Survo provides tools for transforming the representation from one form to another. Also text files (ASCII files) can be converted to Survo data bases and vice versa.

The statistical operations usually accept data in any of the forms mentioned above. However, in more demanding tasks, types 2 and 3 are preferred. Especially in methods which generate new variables to be included in the data set, type 3 should be used.

In types 1 and 2, the size of the data set is limited by the size of the edit field. However, it is not reasonable to waste the operational space offered by the edit field to very large data sets. Typically, in data lists, the number of fields is from 1 to 3 and the number of records is less than, say 300. Similarly, in data tables, a feasible number of fields varies from 1 to 10 and number of records up to, say 500.

In data files (type 3) there are practically no upper limits for the number of fields and cases. In theory, the maximum number of fields is about 5000 and the number of records is anything within the disk capacity allowed by the computer. However, it is hardly reasonable to create data files having over 1000 fields.

Many statistical modules of Survo produce intermediate results which can be represented as new data sets or matrices. Many of those results are saved in full numerical precision as matrix files and they can be processed further by other statistical operations or by the matrix (MAT) operations of Survo. To some extent, even matrix files are valid as source data sets.

The general tools of the editorial mode (i.e. commands and operations in text and table management) are supporting management of data bases in many ways. All these means provide fluent conversion of data from one form to another. Any piece of information, once saved in the system, should be reached easily in each application and connected to some other data structure.

Especially for management of large data bases (type 3), Survo provides a set of **FILE** operations. These operations are used e.g. for saving, editing, browsing, sorting, and combining of data files. Still more operations (like **VAR**, **CLASSIFY**, and **TRANSFORM**) are available for systematic and conditional transformation of variables (fields) in data tables and files.

In data lists, the fields are always numerical. In other forms of Survo data bases, either numerical or alphanumerical (string) fields may appear. Also string fields can be used in numerical computation and analysis when their contents are interpretable as numbers. Other uses of string fields are identification and classification of records. A special form of the **VAR** operation is available for concatenation and modification of string fields.

6.1 Data lists

The simplest data representation is merely a list of adjacent data values in the edit field. A more general form accepts more than one variable.

In the next display, a small data set of one variable **X** is given and a **CORR** operation has been performed on it:

```

9 1 SURVO 84C EDITOR Mon Jan 19 17:18:48 1987 D:\P2\DATA\ 100 100 0
1 *
2 *
3 *DATA X: 1.2 3 -2.4 0 1.7 3.2 2.5 2.7 -0.3 -0.9 1 1.9 END
4 *
5 *CORR X, 6-
6 *X N=12
7 *Variable Mean Std.dev.
8 *X 1.133333 1.731176
9 *
10 *

```

Please, note that the list of data values is always terminated by an empty edit line or **END** after the last value. Since line 4 is empty, **END** on line 3 is not absolutely necessary.

The list starts by the word **DATA**. Then follows a name terminated by a colon (as **X:** above). The data values have to be separated from the name as well as from each other by one or more spaces or commas. The list may occupy several lines.

The text on lines 6-8 is output from the **CORR** operation activated on line 5 (**CORR?**). Since the data set has one variable, only the mean and the standard deviation of variable **X** have been computed.

In the next example, we have two one-variate data sets and they have been compared (pairwise) by the **COMPARE** operation:

```

24 1 SURVO 84C EDITOR Mon Jan 19 17:48:14 1987 D:\P2\DATA\ 100 100 0
1 *
2 *(Conover: Practical Nonparametric Statistics, Wiley 1971, p.208)
3 *Twelve sets of identical twins were given psychological tests to
4 *determine whether the first-born of the twins tends to be more
5 *aggressive than the other. The results were as follows, where the
6 *higher score indicates more aggressiveness.
7 *
8 *
9 *DATA First: 1 2 3 4 5 6 7 8 9 10 11 12
10 *DATA Second: 86 71 77 68 91 72 77 91 70 71 88 87 END
11 *
12 *COMPARE First,Second,14 / TEST=Pairwise
13 *
14 *Paired comparisons:
15 *Samples: N=12 First Second Difference
16 *Mean 79.08333 77.16667 -1.916667
17 *Standard deviation 8.887768 10.37333 7.153617
18 *Paired t=-0.928 (P=0.1866 one-sided t test df=11)
19 *Wilcoxon signed ranks test=-0.756 (P=0.2247 normal approximation)
20 *Critical levels by simulation:
21 * Differences Signed rank
22 *Critical level 0.19684 0.23807 N=17100
23 *Standard error 0.00304 0.00326

```

This example illustrates the possibility of having data lists within the text in a most natural way. (For results, see COMPARE?).

The same data sets could have been connected to one two-variate data list as follows. Observe the form of the COMPARE call, too.

```

38 1 SURVO 84C EDITOR Mon Jan 19 18:08:56 1987 D:\P2\DATA\ 100 100 0
1 *
2 *(Conover: Practical Nonparametric Statistics, Wiley 1971, p.208)
3 *Twelve sets of identical twins were given psychological tests to
4 *determine whether the first-born of the twins tends to be more
5 *aggressive than the other. The results were as follows, where the
6 *higher score indicates more aggressiveness.
7 *
8 *DATA Twins:(First,Second) 86,88 71,77 77,76 68,64 91,96 72,72
9 * 77,65 91,90 70,65 71,80 88,81 87,72 END
10 *
11 *COMPARE Twins(First),Twins(Second),12 / TEST=CORRELATION
12 *Rank correlations:
13 *Samples: N=12 Twins(First) Twins(Second)
14 *Mean 79.08333 77.16667
15 *Standard deviation 8.887768 10.37333
16 *Product moment correlation R=0.7344 (P=0.9967)
17 *Spearman's Rho=0.7355
18 *Kendall's Tau=0.5606 (Nc=49 Nd=12 P=0.9944 normal approximation)
19 * Exact P=0.9973
20 *Critical levels by simulation:
21 * R Rho
22 *Critical level 0.99556 0.99613 N=14200
23 *Standard error 0.00056 0.00052

```

The data list `Twins` is typed on lines 8 and 9. After the word `DATA` we have the name of the data terminated by a colon and then the names of the variables in parentheses. The values are then typed in pairs. Commas and spaces can be used freely. Here the observations are separated by spaces and variables by commas. The number of data values must be divisible by the number of variables. If not, an error message is given when any operation is activated for the data set.

It is up to the user how to organize the listing of the values. For example, in periodic time series, the following setup is handy:

```

24 1 SURVO 84C EDITOR Mon Jan 19 18:54:23 1987 D:\P2\DATA\ 120 100 0
1 *
2 *(Source: Abraham and Ledolter: Statistical Methods for Forecasting)
3 *Monthly average residential electricity usage in Iowa City
4 *(in kilowatt-hours), January 1971 to October 1979
5 *
6 *           Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
7 *DATA IOWA:
8 * 1971: 454 421 389 368 460 386 501 606 539 381 396 452
9 * 1972: 470 485 456 414 388 502 543 642 617 454 429 473
10 * 1973: 504 454 424 406 384 466 668 662 672 426 415 442
11 * 1974: 483 431 402 399 374 430 666 663 501 392 400 501
12 * 1975: 543 473 451 444 384 489 763 711 598 406 415 440
13 * 1976: 523 502 439 420 387 453 630 637 576 411 455 512
14 * 1977: 530 507 436 407 392 531 710 658 500 414 418 520
15 * 1978: 535 503 464 414 383 472 676 622 652 474 422 501
16 *
17 * 1979: 536 506 477 422 385 438 546 642 566 399 - -
18 *
19 *FORECAST IOWA,IOWA,-,20_
20 *
21 *
22 *
23 *

```

The monthly values of each year are typed on one line. The text in the beginning of each line up to the first colon ':' is considered a comment only. Thus the labels 1971: 1972: etc. are skipped when the data is processed. Here line 16 is empty and excludes the last year (1979) from the data list IOWA.

Thus when the FORECAST operation on line 19 is activated, it employs the 8 eight first years (96 observations) as an estimation period and gives a prediction for the year 1979 as follows (see FORECAST?):

```

24 1 SURVO 84C EDITOR Mon Jan 19 18:55:22 1987 D:\P2\DATA\ 120 100 0
17 * 1979: 536 506 477 422 385 438 546 642 566 399 - -
18 *
19 *FORECAST IOWA,IOWA,-,20_
20 *Holt-Winters' Additive Seasonal Forecast: Data IOWA, Variable IOWA
21 *Period=12 obs. (judged from data) Estimation on observations 1-96
22 *Outliers:55,81,7 (+more to be found)
23 *MSE=941.061643 a(level)=-0.000 a(seasonal)=0.000 a(slope)=0.243
24 *Autocorrelations of residuals: r1=+0.10 r2=+0.01 r3=-0.04 r4=-0.16
25 * r5=-0.02 r6=+0.04 r7=+0.10 r8=+0.02 r9=-0.02 r10=+0.08 r11=-0.14
26 * r12=-0.15
27 *Obs.# Forecast
28 * 97 531.10
29 * 98 502.01
30 * 99 456.26
31 * 100 429.35
32 * 101 413.35
33 * 102 491.72
34 * 103 668.90
35 * 104 665.11
36 * 105 620.81
37 * 106 445.12
38 * 107 444.61
39 * 108 512.24

```

Although the data lists may include several variables and observations up to the capacity of the edit field, their use is not recommended for large data sets. It is easy to edit data lists in the edit field, but it is not possible to let the statistical operations write derived variables (like predicted values and residuals of a model) back to the data set. Furthermore, data lists support numeric variables only. No textual data (like names of observations) are accepted.

6.2 Data tables

In data lists, the observations were arranged in rows. Data tables do that columnwise and give usually a better appearance for many-dimensional data sets. There are three different levels in data table representation.

On the primary level, a data table is as follows: (We use the earlier data set of twins.)

```

14 1 SURVO 84C EDITOR Mon Jan 19 19:51:43 1987 D:\P2\DATA\ 150 80 0
1 *
2 *DATA Twins
3 * First Second
4 * 86 88
5 * 71 77
6 * 77 76
7 * 68 64
8 * 91 96
9 * 72 72
10 * 77 65
11 * 91 90
12 * 70 65
13 * 71 80
14 * 88 81
15 * 87 72
16 *
17 *CORR Twins,18
18 *Means, std.devs and correlations of Twins N=12
19 *Variable Mean Std.dev.
20 *First 79.08333 8.887768
21 *Second 77.16667 10.37333
22 *Correlations:
23 * First Second
24 * First 1.0000 0.7344
25 * Second 0.7344 1.0000

```

A data table on the primary level consists of several consecutive lines. The first of them (here line 2) is always of the form

DATA <name of the data set>

and the next gives the names of the variables as contiguous strings. The rest of the lines up to the first empty line (here 16) hold the observations. Thus for each variable there is a column of data values.

Each data value (numeric or a string) must be contiguous (no spaces between characters).

In other forms of data tables, line numbers (labels) are given for the first and last data line as well as for the line which contains the names of the variables. Then it is no longer necessary to have a specific order of various lines. Also a part of a table in the edit field can be defined as a data set.


```

18 1 SURVO 84C EDITOR Tue Jan 20 13:04:09 1987 D:\P2\DATA\ 150 80 0
1 *
2 *
3 *DATA COUNTRIES,A,B,N
4 *DATA NORDIC,A,C,N_
5 *Consumption of various beverages in 12 European countries
6 *
7 N Country      Coffee  Tea    Beer  Wine  Spirits
8 *
9 A Finland      12.5   0.15  54.7  7.6   2.7
10 * Sweden      12.9   0.30  58.3  7.9   2.9
11 * Norway       9.4    0.19  43.5  3.1   1.8
12 C Denmark     11.8   0.41  113.9 10.4  1.7
13 * England      1.8    3.49  113.7  5.1   1.4
14 * Ireland      0.2    3.73  124.5  3.8   1.9
15 * Holland      9.2    0.58  75.5   9.7   2.7
16 * Switzerland  9.1    0.25  73.5  44.9  2.1
17 * France       5.2    0.10  44.5  104.3 2.5
18 * Italy         3.6    0.06  13.6  106.6 2.0
19 * Spain        2.5    -     43.6  -     2.7
20 B Portugal    2.2    0.03  27.5  89.3  0.9
21 *
22 *
23 *

```

In this display, two data sets COUNTRIES and NORDIC are defined on lines 3 and 4, respectively. COUNTRIES includes lines from A to B (from 9 to 20) and NORDIC is a part of it consisting of lines from A to C (from 9 to 12). In both data sets, the names of variables (columns) are given on line N (7).

In the DATA definitions, line numbers could have been employed instead of symbolic, one-character line labels. For example, in this case DATA COUNTRIES,9,20,7 is same as DATA COUNTRIES,A,B,N. In practice, however, it is reasonable to favor line labels, since inserting or deleting lines above the data table does not invalidate the DATA definition. Labels are always relative to data, numbers are not.

In both preceding forms of a data table, the number of data values on each line (for COUNTRIES on lines from A to B) must be the same (here 6) and the same as the number of names on the header line (N). The columns need not be correctly aligned (straight), but it is always clearer to make them aligned, for example, by using the FORM command or a tab line.

Missing observations are denoted by '-' as seen on line 19 of the preceding example. Handling of missing observations always depends on the statistical operation in use and will be explained later.

The forms of data tables already presented as well as the data lists are suitable for all statistical operations which only read data and compute results but which neither alter any data values nor create any new variables (like residuals or predicted values of a model).

There is, however, a third alternative which enables writing to a data table, although the best choice in those applications is often a data file representation.

If we enter still one extra line label (*a mask line*) as the last parameter of the DATA definition, we shall have the possibility of creating new variables (columns) in a data table.

In the next display, the COUNTRIES data has been defined with such a mask line (M):

```

40 1 SURVO 84C EDITOR Thu Jan 22 19:36:38 1987 D:\P2\DATA\ 150 80 0
1 *
2 *
3 *DATA COUNTRIES,A,B,N,M
4 *Consumption of various beverages in 12 European countries
5 *
6 N Country      Coffee  Tea    Beer  Wine  Spirits  Ratio
7 M AAAAAAAAAAA  12.1   1.12  123.1 123.1  1.1    123.123
8 *
9 A Finland      12.5   0.15  54.7  7.6   2.7
10 * Sweden      12.9   0.30  58.3  7.9   2.9
11 * Norway      9.4    0.19  43.5  3.1   1.8
12 * Denmark     11.8   0.41  113.9 10.4  1.7
13 * England     1.8    3.49  113.7 5.1   1.4
14 * Ireland     0.2    3.73  124.5 3.8   1.9
15 * Holland     9.2    0.58  75.5  9.7   2.7
16 * Switzerland 9.1    0.25  73.5  44.9  2.1
17 * France      5.2    0.10  44.5  104.3 2.5
18 * Italy        3.6    0.06  13.6  106.6 2.0
19 * Spain       2.5    -     43.6  -     2.7
20 B Portugal    2.2    0.03  27.5  89.3  0.9
21 *
22 *VAR Ratio=Coffee/Tea TO COUNTRIES_
23 *

```

The task of the mask line (line 7 labelled by M) is to indicate the locations and sometimes also the formats of the columns. There must be as many contiguous mask strings as there are names of columns (on line N).

Please note that we have added an extra column labelled by Ratio and with mask 123.123 for computing of a new column as the ratio Coffee/Tea. This time we are using a VAR operation (described later) for that task and giving the following result:

```

40 1 SURVO 84C EDITOR Thu Jan 22 19:37:53 1987 D:\P2\DATA\ 150 80 0
1 *
2 *
3 *DATA COUNTRIES,A,B,N,M
4 *Consumption of various beverages in 12 European countries
5 *
6 N Country      Coffee  Tea    Beer  Wine  Spirits  Ratio
7 M AAAAAAAAAAA  12.1   1.12  123.1 123.1  1.1    123.123
8 *
9 A Finland      12.5   0.15  54.7  7.6   2.7    83.333
10 * Sweden      12.9   0.30  58.3  7.9   2.9    43.000
11 * Norway      9.4    0.19  43.5  3.1   1.8    49.474
12 * Denmark     11.8   0.41  113.9 10.4  1.7    28.780
13 * England     1.8    3.49  113.7 5.1   1.4     0.516
14 * Ireland     0.2    3.73  124.5 3.8   1.9     0.054
15 * Holland     9.2    0.58  75.5  9.7   2.7    15.862
16 * Switzerland 9.1    0.25  73.5  44.9  2.1    36.400
17 * France      5.2    0.10  44.5  104.3 2.5    52.000
18 * Italy        3.6    0.06  13.6  106.6 2.0    60.000
19 * Spain       2.5    -     43.6  -     2.7    -
20 B Portugal    2.2    0.03  27.5  89.3  0.9    73.333
21 *
22 *VAR Ratio=Coffee/Tea TO COUNTRIES_
23 *

```

Since the value of 'Tea' in Spain is missing, the value of 'Ratio' is missing, too.

Hence, in this form of a data table, writing of new columns is possible by statistical and other Survo modules working on statistical data sets. Since the masks determine the locations of the data values, the missing values can also

be left empty (a ‘-’ is not necessary). For example, before activating the VAR operation the `Ratio` column was totally missing.

Because it is permitted to write new values over old columns (for example, by the VAR operation), one should be very careful when using this form of data representation.

6.3 Data files



Keeping data sets in data files is the most advanced alternative in Survo. Although immediate access to data values and good visibility are strong merits of data lists and tables, there are other virtues which make the file representation superior.

MAT?
MATD?

Data files are naturally the only possibility when handling large data sets (having, say, over 20 variables and 1000 observations). Since Survo data files may include in addition to numeric variables also textual data, many tasks typical of general data base management can be successfully accomplished. The Survo data files are random access files and the numeric data is represented in compressed binary form. This reduces execution times of statistical operations. The default extension in the names of Survo data files is `.SVO`.



By various `FILE` operations it is simple to manipulate data files, convert data from ASCII files to Survo data files and vice versa. It is also easy to scan data files on the screen, edit them and load parts of them to the current edit field in a form of a data list or a data table or even in a user-defined format.

A natural way to create a data file is to write a representative part of the data set in the edit field in a form of a data list or a data table and use the `FILE COPY` operation.

For example, the `FILE COPY COUNTRIES,EURO1` command in the next exhibit creates a new data file `EURO1.SVO` on the current data path (here `D:\P2\DATA` displayed on the top line of the screen) and saves all records from the data table `COUNTRIES` in it. If, however, `EURO1.SVO` already exists on path `D:\P2\DATA`, the data in `COUNTRIES` would be appended to the current contents of `EURO1.SVO`, provided that its structure is compatible with `COUNTRIES`. If the structures do not match, an error message will be given and no data values are saved.

```

26 1 SURVO 84C EDITOR Sun Jan 25 12:16:39 1987 D:\P2\DATA\ 100 100 0
1 *
2 *
3 *DATA COUNTRIES,A,B,N
4 *Consumption of various beverages in 12 European countries
5 *
6 N Country      Coffee  Tea    Beer  Wine  Spirits
7 *
8 A Finland      12.5   0.15  54.7  7.6   2.7
9 * Sweden       12.9   0.30  58.3  7.9   2.9
10 * Norway      9.4    0.19  43.5  3.1   1.8
11 * Denmark     11.8   0.41  113.9 10.4  1.7
12 * England     1.8    3.49  113.7  5.1   1.4
13 * Ireland     0.2    3.73  124.5  3.8   1.9
14 * Holland     9.2    0.58  75.5   9.7   2.7
15 * Switzerland 9.1    0.25  73.5  44.9  2.1
16 * France      5.2    0.10  44.5  104.3 2.5
17 * Italy        3.6    0.06  13.6  106.6 2.0
18 * Spain       2.5    -     43.6  -     2.7
19 B Portugal    2.2    0.03  27.5  89.3  0.9
20 *
21 *FILE COPY COUNTRIES,EURO1
22 *
23 *

```

As a proof of a successful move of the data values, we activate a FILE SHOW EURO1 command as follows:

```

16 1 SURVO 84C EDITOR Mon Apr 20 09:09:01 1992 D:\P2\DATA\ 100 100 0
17 * Italy        3.6    0.06  13.6  106.6 2.0
18 * Spain       2.5    -     43.6  -     2.7
19 B Portugal    2.2    0.03  27.5  89.3  0.9
20 *
21 *FILE COPY COUNTRIES,EURO1
22 *FILE SHOW EURO1
23 *

```

This activation replaces the current edit field window temporarily by another which displays the first records of the data file EURO1:

```

16 1 SURVO 84C EDITOR Mon Apr 20 09:09:25 1992 D:\P2\DATA\ 100 100 0
File EURO1 N=12 Finland Country Finland
1 Country      CoffeeTea Beer Wine Spirits
1Finland      12.50.15 54.7 7.6 2.7
2Sweden       12.90.30 58.3 7.9 2.9
3Norway      9.40.19 43.5 3.1 1.8
4Denmark     11.80.41113.9 10.4 1.7
5England     1.83.49113.7 5.1 1.4
6Ireland     0.23.73124.5 3.8 1.9
7Holland     9.20.58 75.5 9.7 2.7
8Switzerland 9.10.25 73.5 44.9 2.1
9France      5.20.10 44.5104.3 2.5
10Italy      3.60.06 13.6106.6 2.0
11Spain      2.5 - 43.6 - 2.7
12Portugal   2.20.03 27.5 89.3 0.9
13
14
15
16
17
18
19
20
21
To stop, press EXIT! (F1=HELP)

```

The records are displayed, in principle, in the same way as in the data table COUNTRIES in the edit field. In this FILE SHOW mode, one can scroll the visible part of the data file to any direction, type new values and edit the data.

The great advantage of larger files is that there are no such limits for the number of fields (columns) or records (lines) in this setup as we have in the edit field. The functions of **FILE SHOW** will be described later more thoroughly.

By pressing the **EXIT** (F8) key, the **FILE SHOW** display disappears and the normal editorial mode is resumed.

It is not necessary to know any details about data file structure in order to create files and to use them. The data file **EURO1** can now be used instead of the data table **COUNTRIES** in statistical and other operations.

In principle, there are no severe limits to the size of a data file. Currently, the maximum number of variables is about 5000, but the largest practical number of fields (variables) is about 1000. For the number of records (observations, cases) the only limit is the disk memory capacity.

A Survo data file has to be saved entirely on one disk. Any disk available on the installation can be used as a data disk. The current disk (path) for Survo data files and other files generated in a Survo session is displayed on the top line of the screen (as **D:\P2\DATA** in the preceding example). However, a complete pathname always overrides the default path. For example, in the preceding example, the command

```
FILE COPY COUNTRIES,C:\STAT\EURO2
```

would save data to the file **EURO2** on the path **C:\STAT**.

The current data path is altered by the **DISK** command or by the **DISK** (F4) key.

The following operations are available for Survo data file management:



FILE CREATE

Creating a data file by entering the structure of the file as a **FIELDS** list in the edit field.

FILE?

FILE ACTIVATE

Selecting active variables (fields). Also selecting protected fields and giving scale types for statistical variables. Defining various sets of variables. The **FILE ACT** key (alt-F6) is a more straightforward alternative for the **FILE ACTIVATE** command.



FILE SHOW

Saving, editing and scanning of a data file. The data is displayed in a window on the screen so that one record takes one line and the active fields appear as columns. Several options for searches, moves, etc.

MASK?

FILE EDIT

Saving and editing of a data file. **FILE EDIT** displays all active fields of one record at a time in its own window.



MEDIT?

FILE STATUS

Loading the structure of a data file to the edit field in a form of a **FIELDS** list. This allows the possibility of editing the structure by the **FILE UPDATE** operation or creating a new data file with a similar structure by **FILE CREATE**.

FILE UPDATE

Updating the data file structure by using a **FIELDS** list. Also new fields (variables) of selected type may be defined.

FILE LOAD

Loading the active part of a data file in the edit field as a data table, a data list or in a format specified by the user. The output of **FILE LOAD** may also be directed to a text file.

FILE SAVE

Copying a text file (ASCII) or a part of it to a Survo data file. If the data file does not exist, a **FIELDS** list gives the structure or if it is missing, the new data file will be automatically created according to the structures of the fields in the text file.

FILE COPY

Copying a data list, table or file to another data file. If the target file does not exist, it will be automatically created according to the structure of the source data. If the target file exists, the source data will be appended to the end of the target data file. However, if a **MATCH=<variable>** is given, the new data will be joined to the target data according to the **MATCH** variable. **FILE COPY** is the main tool for merging of data files in various ways.

FILE SORT

Sorting observations in a data file according to a sort key defined by several numeric and string variables in a given order. The sorted observations will be saved in a new data file with the same structure.

FILE INIT

Adding empty observations to the end of a data file.

FILE REDUCE

Deleting last fields and records from a data file.

FILE DEL

Deleting the entire data file.

VAR

Making transformed variables in any data file or in a data table with a mask line. Several variables may be transformed and new variables defined at the same time. The transformations are given as equations in the edit field. The functions of editorial computing are available. Also simulated data according to various distributions and statistical models can be generated.

CLASSIFY

Transformation of variables by classifying and rearranging old values.

TRANSFORM

Making the same selected transformation for a set of variables.

FILE AGGRE

Combining observations by computing sums or means of active variables.

FIELDS list

The FIELDS list describes the structure of a data file. It is loaded into the edit field when the FILE STATUS operation is activated. For the data file generated in our preceding example we obtain

```

12 1 SURVO 84C EDITOR Sun Jan 25 15:40:15 1987 D:\P2\DATA\ 100 100 0
13 * England 1.8 3.49 113.7 5.1 1.4
14 * Ireland 0.2 3.73 124.5 3.8 1.9
15 * Holland 9.2 0.58 75.5 9.7 2.7
16 * Switzerland 9.1 0.25 73.5 44.9 2.1
17 * France 5.2 0.10 44.5 104.3 2.5
18 * Italy 3.6 0.06 13.6 106.6 2.0
19 * Spain 2.5 - 43.6 - 2.7
20 B Portugal 2.2 0.03 27.5 89.3 0.9
21 *
22 *FILE COPY COUNTRIES,EURO1
23 *FILE STATUS EURO1
24 * Copy of data matrix COUNTRIES
25 *FIELDS: (active)
26 * 1 SA_ 11 Country
27 * 2 NA_ 4 Coffee (####.#)
28 * 3 NA_ 4 Tea (#.##)
29 * 4 NA_ 4 Beer (####.#)
30 * 5 NA_ 4 Wine (####.#)
31 * 6 NA_ 4 Spirits (#####.#)
32 *END
33 *SURVO 84C data file EURO1: record=58 bytes, M1=11 L=64 M=6 N=12
34 *

```

The `FILE STATUS` command produces the output on lines from 23 to 32. The `FIELDS` list is here on lines from 24 to 31. The first line always starts with the word `FIELDS` and the list ends to a line starting with `END`. Between these lines, all active fields are described as follows. Each field (variable) has a line of its own starting with the index of the field. This index ranges here from 1 to 6 since there are altogether 6 variables.

The next item on the line is a string describing the type and status of the variable. The string `'SA_'` for the first variable `'Country'` (line 25) tells that `'Country'` is a string variable (S), its activity status is `'A'` and the field is not protected (`_`).

The third column in the `FIELDS` list gives the length of the field in bytes. For example, the system has reserved 11 bytes for the name of the country, since the longest name in the original data set which was used as a basis for this data file was `'Switzerland'`.

Finally there is the name of the field which is always a contiguous string (without spaces). The names of the numeric fields can include formats of type (`####.#`). In this case the formats have been automatically selected by the `FILE COPY` operation which was used for the creation of `EURO1` file. These formats are used, for example, when values are loaded into the edit field by the `FILE LOAD` command.

`FILE STATUS` has also reported the basic structure parameters of the data file after the `END` line (on line 32). It tells that each observation takes 58 bytes (`record=58 bytes`) and there is space for (`M1=11`) fields. At the moment (`M=6`) fields and (`N=12`) observations has been saved. The current 6 variables take only $11+4+4+4+4+4=31$ bytes. Thus the `FILE COPY` operation has reserved $58-31=27$ extra bytes for possible new variables to be defined later. The difference $M1-M=5$ is the maximum number of such new variables.

The `L` parameter (`L=64`) gives the longest permitted name for a variable in this file. Please, note however, that only 8 first bytes of names are used as reference when calling variables in various Survo operations. The rest of the name string is free description which helps the user to recognize the meaning of the variable.

The line preceding the `FIELDS` line (line 23) gives an overall narration of the data file. When creating a data file by the `FILE CREATE` scheme, any number of lines may be used for such a free description. When editing the structure of the file (by `FILE UPDATE`) it is not possible to change the number of text lines, but their contents can be altered.

The structural parameters `'record'`, `'M1'` and `'L'` as well as the types and the lengths of the fields cannot be changed during the lifetime of the data file. However, when needed, one can easily define a new file with a better structure and copy the data values to it.

On the other hand, the names and status descriptions of the fields can be

changed within the data file by using the FILE UPDATE operation. We can also define new variables in this way.

In our example, the setup may be modified as follows and the new structure becomes valid by activating FILE UPDATE as follows:

```

18 1 SURVO 84C EDITOR Sun Jan 25 16:24:29 1987 D:\P2\DATA\ 100 100 0
12 * England 1.8 3.49 113.7 5.1 1.4
13 * Ireland 0.2 3.73 124.5 3.8 1.9
14 * Holland 9.2 0.58 75.5 9.7 2.7
15 * Switzerland 9.1 0.25 73.5 44.9 2.1
16 * France 5.2 0.10 44.5 104.3 2.5
17 * Italy 3.6 0.06 13.6 106.6 2.0
18 * Spain 2.5 - 43.6 - 2.7
19 B Portugal 2.2 0.03 27.5 89.3 0.9
20 *
21 *FILE COPY COUNTRIES,EURO1
22 *FILE UPDATE EURO1
23 * Consumption of various beverages in 12 European countries
24 *FIELDS:
25 * 1 SA_ 11 Country Name of the country
26 * 2 NA_ 4 Coffee kg (####.#) {0,15}
27 * 3 NA_ 4 Tea kg (###.#) {0,5}
28 * 4 NA_ 4 Beer l (###.#)
29 * 5 NA_ 4 Wine l (###.#)
30 * 6 NA_ 4 Spirits l (#####.#)
31 * 7 N 8 Ratio Coffee/Tea
32 *END
33 *
34 *

```

To see the difference, the alterations are shaded. We have typed a new narration of the file on line 23. Similarly more text in field descriptions is written on lines 25-30.

The permitted range of a numeric variable can be entered in braces. Thus {0,15} on line 26 restricts all subsequent values to be saved as 'Coffee' to the interval from 0 to 15 in the FILE SHOW and FILE EDIT operations.

Finally, we have defined a new variable 'Ratio' on line 7. The text 'Coffee/Tea' is a plain comment and does not imply anything about the current values of 'Ratio'. In fact, after the FILE UPDATE operation, the new variables are initiated to have missing values. We have to activate a command like

```
VAR Ratio=Coffee/Tea TO EURO1
```

to obtain sensible values for this new variable.

The field length of a numeric variable is always 1,2,4 or 8 corresponding to following packed representations:

- 1 integer from 0 to 254
- 2 integer from -32768 to 32766
- 4 single precision decimal number
6-7 significant digits
- 8 double precision decimal number
15 significant digits

When the data file is automatically created (as in our preceding example), the

Survo system selects the types of the numeric variables according to their numeric behavior in the source data. Since in our examples all numeric variables were decimal numbers with 2-4 digits precision, type 4 was selected.

In FILE UPDATE the new 'Ratio' variable was defined to be a double precision (8) variable.

The string variables are not packed and their length typically varies from 1 to 64 bytes. Even longer string variables could be defined, but, for example, the FILE EDIT operation does not support string variables which are longer than 64 bytes.

FILE CREATE operation

The most general way to make a Survo data file is to use the FILE CREATE operation with a FIELDS list. The EURO1 file could have been created by the following scheme:

```

27 1 SURVO 84C EDITOR Sun Jan 25 18:29:31 1987 D:\P2\DATA\ 100 100 0
1 *
2 *FILE CREATE EURO1,60,12,64
3 * Consumption of various beverages in 12 European countries
4 *FIELDS:
5 * 1 S 11 Country Name of the country
6 * 2 N 4 Coffee kg (####.#) {0,15}
7 * 3 N 4 Tea kg (###.#) {0,5}
8 * 4 N 4 Beer l (###.#)
9 * 5 N 4 Wine l (###.#)
10 * 6 N 4 Spirits l (#####.#)
11 * 7 N 8 Ratio Coffee/Tea
12 *END
13 *

```

The rules for FILE CREATE are same as for FILE UPDATE, but we have more possibilities of specifying the structure and capacity of the file. Also types of the fields can be selected freely.

Although no parameters except the name of the file are required, it is typical to give three extra parameters, at least. In the command

```
FILE CREATE EURO1,60,12,64
```

60 is the record length in bytes, 12 is the maximum numbers of fields allowed and 64 is the maximum length of names of variables.

After the FILE CREATE operation, the file is empty. Records have to be taken from text files by FILE SAVE or copied from data lists and tables by FILE COPY or entered manually by FILE SHOW or FILE EDIT.

As a more genuine example of data file creating we present the following scheme:

```

30 1 SURVO 84C EDITOR Sat Jan 25 12:46:27 1992 D:\P2\DATA\ 100 100 0
1 *
2 *FILE CREATE FINLAND,128,30,64_
3 * The Finnish towns and communes in alphabetic order
4 * Data mainly from years 1978-80.
5 *
6 *FIELDS:
7 * 1 SA 16 Commune Name of commune
8 * 2 SA 3 Province [UUS,TUR,AHV,HAM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
9 * 3 NA 4 Popul Estimated population in 1980 (#####)
10 * 4 NA 4 Births Live births in 1978 (#####)
11 * 5 NA 8 Area Land area square km in 1979 (#####.##)
12 * 6 SA 1 Agri Population in agriculture (10%)
13 * 7 SA 1 Industry Population in industry (10%)
14 * 8 SA 1 Service Population in other occupations (10%)
15 * 9 NA 4 Houses New houses and apartments in 1978 (#####)
16 * 10 NA 4 Tax Communal tax rate in 1979 (##.##) {12,20}
17 * 11 NA 2 Income Income per inhabitant in 1979 (#####)
18 *END
19 *

```

The statistics to be presented in this data file are more or less out of date. However, this is one of the data sets employed in various Survo documents, demonstrations, and teaching programs for a long time. Although the data could easily be updated, we stick to this traditional version.

If there are plenty of variables, it is not sensible to define them all when the file is created. The FILE UPDATE operation with an extended FIELDS list is always available for defining more variables. Especially if the data values will be entered manually, a smaller amount of variables at a time is easier to handle. On the other hand, the user can always select a subset of variables (see FILE ACTIVATE) for current operations. Then the other (passive) fields do not cause any harm.

6.4 Data saving and editing

As seen from the previous examples, statistical and other numeric data can be entered in several ways. In case of smaller data sets, many people would like to write the data values directly to the edit field in the form of a data list or a data table (matrix). All text processing and table management facilities are available for their editing. Finally, it is simple to move such data sets to data files, if needed, by the FILE COPY command.

For maintaining large data bases we need, however, special tools. These are provided by the FILE SHOW and FILE EDIT operations.

FILE SHOW opens a temporary window where the each record takes one line and the active fields appear as columns. Data is then entered and edited in the same way as in the edit field. The good point is that in this working mode there are no limits for the number of lines or columns as in the edit field. At a time, FILE SHOW displays 21 records (or 46 records in the 50 line display mode activated by the sucro command /48) and as many fields as columns as there is space on the screen. By arrow keys etc., it is easy to scroll the visible part of the data file in any direction.

FILE EDIT offers similar possibilities for data saving and editing. It shows



RESIZE?

only one record at a time but all the active fields of it. When there are less active fields than lines in the window, the fields are shown on consecutive lines. Otherwise the list of fields is organized by rows.

An alternative tool for advanced data management is the optional **FEDIT** operation by M.Korhonen described in SURVO 84C Contributions No.5 (1992). In **FEDIT**, the values in a Survo data file are displayed on forms defined by the user. The size of the form is not limited by the screen. Several forms may correspond to one observation in a Survo data file. Besides standard fields forms may involve selection fields. Validity checks for field values can be defined.

6.4.1 FILE SHOW operation

We introduce the functions of **FILE SHOW** in a natural order, starting from data input and editing. The data file **FINLAND** (created at the end of section 6.3) will be used as a leading example. Activation of the command

```
FILE SHOW FINLAND
```

erases the current edit field window and replaces it by a **FILE SHOW** window. Assuming that no data have been saved in the file so far, the display is



```

16 1 SURVO 84C EDITOR Wed Jan 25 13:19:30 1992 D:\P2\DATA\ 100 100 0
File FINLAND N=0
1 Commune ProPopul BirthsArea AISHousesTax Income
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
To stop, press EXIT! (F1=HELP)

```

FILE SHOW?

The voidness of the data file is indicated by the notation **N=0** on the second line. All the fields for the 21 first records are empty. The cursor is blinking in the first position of the first field (**Commune**) of the first record. Every second field has a different background colour. This enables putting the fields side by side without space consuming delimiters.

To enter data values, the user has to give himself a **writing permission** by **pressing the function key [F3]**. Since **FILE SHOW** is often used for data browsing only without no intention to alter the data, this lessens the danger of

changing the data values by accident. **F3** is needed only once during each FILE SHOW activation.

To a great extent, data values are typed and the cursor is moved by the same keys and key combinations as in the edit field. The tab positions are same as the field limits.

Information about key codes and other options is obtained in the FILE SHOW mode by the **HELP** (F1) key. At first it gives a list of keys in the form:

```

16 1 SURVO 84C EDITOR Wed Jan 25 15:02:43 1992 D:\P2\DATA\ 100 100 0
2
3
4
Key codes in FILE SHOW:
F1 Help      F2 PREFIX      F3 Permit editing
F4 Copy a field          alt-F4      Block definition etc.
F5 Copy a record        alt-F5      Search for records
                          PREFIX alt-F5 Search for fields
F6 Field description  PREFIX F6 Continuous display of field descriptions
F7 Set/Recall a reference point      alt-F7      Append field to the tutstack
F8 Exit
F9 Insert mode on/off          alt-F9      Insert a record (line)
F10 Delete character or block mode  alt-F10     Delete a record

PgDn  Next page          PgUp  Previous page
TAB   Next field
END   Last field / Scroll to the right
HOME  First field / Scroll to the left / First page
PREFIX ENTER Set the return column
PREFIX F7 Sound on/off
S = More information on FILE SHOW

```

FSHOWKEY?

Thereafter other functions may be inquired by pressing **S**. In fact, all information provided by the inquiry system of Survo is available in a normal way. However, when the inquiry is interrupted by **ENTER**, the previous FILE SHOW display is resumed.

Hence, we start data saving by **F3** and type data values as follows:

```

16 1 SURVO 84C EDITOR Wed Jan 25 13:19:30 1992 D:\P2\DATA\ 100 100 0
File FINLAND N=9 Kerava Tax 16.00
1 Commune ProPopul BirthsArea AISHousesTax Income
1Helsinki UUS 483057 5851 176.87027 419215.00 26875
2Espoo UUS 133556 2063 308.87027 196414.50 24665
3Vantaa UUS 129669 2187 240.02036 179315.00 21317
4Hyvinkää UUS 37204 527 321.23044 58115.00 17651
5Kerava UUS 23300 390 30.70045 45216.00 19613
6Järvenpää UUS 23129 374 41.38045 39715.50 18000
7Tuusula UUS 22036 326 215.86045 26714.75 18505
8Nurmijärvi UUS 21875 322 361.56135 25613.75 17861
9Kirkkonummi UUS 19372
10
11
12
13
14
15
16
17
18
19
20
21
To stop, press EXIT! (F1=HELP)

```

In this situation, 8 complete records and a part of the 9th have been typed. The cursor is moved (just as an illustration) to the 'Tax' field of the 5th record

'Kerava'. These facts are confirmed on the second display (top) line. This line keeps an account of the current file status by giving the number of records (N=9), the name of the record (Kerava), and the name and the value of the current field indicated by the cursor. In fact, in Survo data files we have no special field for the names of the records. However, in many operations, like FILE SHOW here, the first field (if its a string field) is considered a 'name' and used as such. Here the name of the current record is displayed in the middle of the top line even if the name field itself is not visible (being a passive field or due to scrolling to the right).

The names of the fields appear on the line below the top line. They are truncated to the width of the field. To see the complete name of the current column, press **F6**. The name is then displayed on the bottom line of the window. To see these names continuously during the FILE SHOW session, press keys **PREFIX F6**.



It is best to learn the technique of data saving and editing by doing. Take data of your own and try to proceed according to hints given above. Certain useful facts are documented below.

In the sequel, when using the FINLAND data we assume that all the 464 records have been saved and they are sorted in alphabetic order according to the name of the commune.

Formats of the fields

In FILE SHOW, the fields are displayed in formats given in the field descriptions in the form (###.##) . To edit these formats, use FILE STATUS and FILE UPDATE. If no format is given for a particular field, the default formats are

```

###    for 1 byte numeric field,
##### for 2 byte numeric field,
#####.### for 4 and 8 byte numeric fields.

```

Missing values appear always as empty fields. If the data value exceeds the limits given by the current format, an empty field terminated by '*' will be displayed instead. To see the true value of such a field, touch it by the cursor. Then the value will be displayed on the top line.

Setting limits for values

When editing numerical fields, extra limits may be set by the limit specifications of the form {lower_limit,upper_limit} in the field descriptions. These limits are set and altered by FILE UPDATE. In FINLAND, the 'Tax' field has such limits, namely {12, 20}.

The values of a string field are limited to a predetermined set by giving the permitted values as a list of the form [value1,value2, ...] in the field description. This option is used in the field 'Province' of the FINLAND file.

In the FILE SHOW mode, the field descriptions and hence the possible lim-

its or permitted values appear on the bottom line constantly by pressing the keys `PREFIX` `F6` or temporarily by `F6`.

Inserting, deleting and moving of records

The current record is deleted from the data file by `LINE DEL` (`alt-F10`) key. An empty record is inserted after the current record by the `LINE INS` (`alt-F9`) key.

A block of consecutive records is marked by using the `BLOCK` (`alt-F4`) key repeatedly. The marked block is indicated by *blinking* numbers in the first (order) column. A marked block is moved to another place in the data file by indicating the new position of the first record of the block by the cursor (to move it just in front of record # *N*, indicate record # *N*) and pressing `BLOCK`. After the move, no block is marked. The records belonging to a marked block are deleted by the `ctrl-END` key.

For typing of values occurring in consecutive records or simply for copying existing records and fields, keys `F4` and `F5` are available. `F4` copies the current field and `F5` the entire record. If a marked block (by `BLOCK`) exists, the first record of it is used as the source. If there is no marked block, the previous field (or record) just above is copied. If the current field is not empty, `F4` has no effect. Similarly, if the current record is not empty, `F5` has no effect.

Searches

Records meeting given conditions are sought by placing the cursor in the appropriate field and by pressing the `SEARCH` (`alt-F5`) key. The search takes place according to the values of the current field or # of the record. Permitted conditions (search keys) are, for example,

`<100 >=25.5 <>1` (not equal to 1).

In string fields, the search key is a sequence of characters and the search is succesful when a case starting with this sequence is found. In addition, a partial string (in the middle of the field) can be searched for, by using a search word of the form `*<string>`. The search is sequential from the current record onwards. The search is stopped when the first succesful record has been found and this record will be displayed. The search can then be continued simply by pressing `ESC`. If only the number of succesful cases is wanted, press `PREFIX` `ESC`.

The sequential search is replaced in string fields by the much faster binary search if a specification of the form `SORT:<sort_field>` appears in the general file description (between `FILE CREATE` and `FIELDS` lines). It is then assumed that the file has been sorted in alphabetic order with respect to the field in question. In fact, the `FILE SORT` operation adds this specification automatically to the sorted file.

In the `FILE SHOW` mode, also fields (variables, columns) can be searched for by their names. This is relevant when there are very many active fields. To initiate such a search, press `PREFIX` `alt-F5`.

***Sound effects**

PREFIX **F7** turns on/off the sound effects. The sounds will be heard when the cursor in a horizontal move crosses the border between fields or in vertical move touches various characters. In vertical direction, the sound is based on the characters touched by the cursor. However, if the insert mode is on, the value of the entire field determines the pitch. In the latter case, the tone range is selected in accordance with the field limits in braces. If such limits are not given for the current field, the pitches are selected adaptively according to the range of values encountered.

This technique enables an audible control for data saving and browsing. The sound effects are defined in the `SOUND.BIN` file.



MEDIT?

****File SOUND.BIN**

`SOUND.BIN` is located in the `.\SYS` subdirectory and it is a binary file of 256 characters. It is edited by the `CODES LOAD` and `CODES SAVE` commands.

For each printable character, the `SOUND.BIN` file gives a value 0 - 255 which is the pitch of that character on a twelve-tone scale. When the cursor is moved either up or down, each character touched will generate the tone defined in `SOUND.BIN`.

The value for standard pitch (a = 440 hz) is given as the second byte in `SOUND.BIN`. Default is 65. The first byte in `SOUND.BIN` is the duration of each sound.

In the default setting of `SOUND.BIN`, the letters give the chromatic scale starting from A, a=440 hz and digits 0,1,2,3,... ascending C major triad tones. This simplifies detecting of outliers in numerical columns. Moving over field borders in horizontal direction gives same tones as the digits 0,1,2,3,... in vertical direction.

Any column in the `FILE SHOW` display can also be "played" automatically by turning the sound on (by **PREFIX** **F7**) and starting by **PREFIX** **↓**. The tempo is adjusted during the play by the **+** and **-** keys. The original tempo is given as the third byte in `SOUND.BIN`.



PLAY?

***Other specialities**

The following keys have the same tasks as they have in the edit field:

F7 sets and recalls a reference point in the data file.

PREFIX **ENTER** sets the return column thus simplifying typing data in one column.

PREFIX **↓** moves the cursor primarily to the last line and secondarily to the last record in the file. However, if the sound is on, **PREFIX** **↓** starts playing of the current column (see Sound effects).

The active fields appearing as columns in `FILE SHOW` display are selected by the first mask column of the data file. Another mask column may be selected by activating `FILE SHOW` of the form (see also `FACTIV?`)

FILE SHOW <data_file>, <#_of_the_mask_column> .

To go directly to the line **after** the last record (line) (instead of the first one), activate

FILE SHOW <data_file>+

and one can start typing new records at once. Similarly, activation of

FILE SHOW <data_file>-

leads immediately to the last record.



6.4.2 FILE EDIT operation

FILE SHOW?

FILE EDIT works, in principle, in a similar way as FILE SHOW. The main difference is the structure of the display. When a command of the form

FILE EDIT <data_file>

is activated, the normal display is replaced by another window which gives a table of all active fields and the values of the first observation in them as follows. For the file FINLAND we have the following opening:



```

10 1 SURVO 84C EDITOR Sun Jan 26 11:33:18 1992 D:\P2\DATA\ 100 100 0
Data file FINLAND Record # 1 Alahärmä N=464
Commune Alahärmä
Province VAA
Popul 5358
Births 78
Area 348.23
Agri 4
Industry 2
Service 2
Houses 64
Tax 17.00
Income 12711
To stop, press EXIT!
```

As in FILE SHOW, data values are entered and edited just by typing text in the cells reserved for the active fields in this display. In the FILE EDIT mode, the cursor is moving within these cells only. Thus when one cell has filled, the cursor jumps to the next one or if the current cell is the last one, to the first cell of the next case. Also when the **ENTER** key is pressed, the cursor goes to the next cell.

Any text is corrected simply by overtyping or by using the **INSERT**, **DELETE** and **ERASE** keys. The cursor control keys (arrow keys) can be used in a normal way and the records are scanned rapidly by the **PgDn** and **PgUp** keys.

All text and numbers are saved as soon they are entered. However, if a value does not fit in the range of the field type in question or in the range of the form {lower_limit, upper_limit} given in the field description, an error message is displayed and the previous value is retained.

Missing values are represented as empty cells (mere blanks or spaces). Originally, all cells are empty and correspond to missing values.

Saving and editing is interrupted by pressing the **EXIT** (F8) key. Then the file edit screen disappears, the data file will be closed and the standard edit field display is resumed.

The **FILE EDIT** operation offers also other activities which are useful in file management. The options are not so multifarious as those of **FILE SHOW**.

The list of **FILE EDIT** options are displayed by pressing the **HELP** key after **FILE EDIT** has been activated. Currently, the following help window is obtained:

```

10 1 SURVO 84C EDITOR Sun Jan 26 11:34:41 1992 D:\P2\DATA\ 100 100 0
Key codes in FILE EDIT:
EXIT Return back to SURVO 84 EDITOR
NEXT Next observation (case, record)
PREV Previous observation
ENTER Next variable (field)
REF Description of the current variable

SEARCH Search for observations according to the current field
or # of observation. Permitted conditions are, for example
<100 >=25.5 <>1 (not equal to 1)
If the file is sorted (in ascending order) with respect to
some field and SORT:<name_of_sort_field> is given on the
text lines in the file, cases are sought for by a binary search.
ESC Search for the next case (according to rule given by SEARCH)

DEL_LINE Delete one or more observations from the file
INS_LINE Move one or more observations to a new place
COPY Copy one observation to replace the current one
Arrow keys have their normal functions

Press any key!

```

The **SEARCH** (alt-F5) key has the same function as in **FILE SHOW**. For example, if we are going to type new observations to the end of a data file which already contains, say, 47 observations, the procedure is as follows. We enter the file edit mode by a **FILE EDIT** operation and see the number of current observations on the second line in the form **N=47**. Then the first new observation (#48) is reached by pressing the **SEARCH** key, and when the prompt

Record to be found ? _

appears, we enter 47 (# of last observation) and get the values of the last observation. If now the **PgDn** key is pressed, the empty cells of observation #48 will be displayed and we are ready to type more data values.

In **FILE SHOW**, the corresponding task is carried out also more simply by pressing the key combination **PREFIX** **↓** twice.

In `FILE EDIT`, ‘+’ and ‘-’ after the file name have the same effects as in `FILE SHOW`; the end of the file will be shown. Similarly `mask #n` selects the active fields by

`FILE EDIT <data>,n` (`n=1` is default).

The extended field names are shown by entering a second extra parameter 1.


Example: `FILE EDIT FINLAND,1,1`.


6.5 Selecting fields and field attributes

The `FILE SHOW` and `FILE EDIT` operations handle **active** fields (variables) only. In many other tasks, too, it is advantageous to have the possibility of working with a particular subset of variables. In Survo we have several alternatives in making such choices.

The active fields can be selected by the

`FILE ACTIVATE <data_file>`

command. Since this function is needed very often, there is a shorter alternative. The `FILE ACT` (`alt-F6`) key does the same task with the current data file. If no data file during the Survo session has been used earlier, a prompt for the file will appear. When using the `FILE ACT` key, it is also possible to change the file by pressing *the key twice (quickly)*. 

In file activation, (long) field names and types are listed on the screen. If there are many variables (over 20), the list is split into several pages (max 20 fields each) and it is easy to scan the list by the `PgDn` and `PgUp` keys. 

The cursor is now moved in a shaded area having typically three one-character columns. We call these columns **mask columns**, because various selections are marked in them.

RESIZE?

Assume now that we have saved all 464 observations in the `FINLAND` file and pressed the `FILE ACT` (`alt-F6`) key. The following display will appear:

```

10 1 SURVO 84C EDITOR Tue Jan 26 14:27:17 1992 D:\P2\DATA\ 100 100 0
Data file FINLAND M= 11 Col= 1 M(active)= 4 M(protected)= 0
1 S A- Commune Name of commune
2 S -- Province [UUS,TUR,AHV,HÄM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
3 4 -- Popul Estimated population in 1980 (#####)
4 4 A- Births Live births in 1978 (#####)
5 8 -- Area Land area square km in 1979 (#####.##)
6 S A- Agri Population in agriculture (10%)
7 S -- Industry Population in industry (10%)
8 S -- Service Population in other occupations (10%)
9 4 -- Houses New houses and apartments in 1978 (#####)
10 4 A- Tax Communal tax rate in 1979 (##.##) {12,20}
11 2 -- Income Income per inhabitant in 1979 (#####)

Denote passive variables by - and active by A,X,Y etc. Stop by EXIT!

```

Thus on each line the complete name of the field in question is seen. This is very helpful when making various choices, especially if there are a great number of fields. Observe also that the type of the field (S,1,2,4,8) is given as a column after the index of the field.

Shading indicates the mask columns where the cursor can be moved by the arrow keys. When something is typed in a certain position, the cursor automatically moves one step **downwards**, which helps the selection process.

Now we have denoted most of the fields by '-' in the first mask column (activation column) and only 4 of them have remained active.

There are two more mask columns in the shaded area. We can go to the second one by pressing the 'right' arrow. Then the display will be altered to the following form:

```

10 1 SURVO 84C EDITOR Tue Jan 26 14:30:15 1992 D:\P2\DATA\ 100 100 0
Data file FINLAND M= 11 Col= 2 M(active)= 4 M(protected)= 6
1 S AP Commune Name of commune
2 S -P Province [UUS,TUR,AHV,HÄM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
3 4 -P Popul Estimated population in 1980 (#####)
4 4 AP Births Live births in 1978 (#####)
5 8 -P Area Land area square km in 1979 (#####.##)
6 S AP Agri Population in agriculture (10%)
7 S -I Industry Population in industry (10%)
8 S -- Service Population in other occupations (10%)
9 4 -- Houses New houses and apartments in 1978 (#####)
10 4 A- Tax Communal tax rate in 1979 (##.##) {12,20}
11 2 -- Income Income per inhabitant in 1979 (#####)

Denote protected variables by P. Stop by EXIT!

```

The second mask column is intended for marking those fields by 'P' which should be protected. The system is never allowed to write any results in

protected fields. Similarly, for example, in the file editing modes (FILE SHOW and FILE EDIT), it is not possible to alter values of protected fields, even if they are active and thus displayed. Such protected fields will appear in the FILE EDIT window with red 'P' labels and in the FILE SHOW window with differently coloured column labels.

The third mask column has no effect in file management. Its task is to indicate the scale types of statistical variables. Thus when proceeding yet one step to the right we shall have:

```

10 1 SURVO 84C EDITOR Tue Jan 26 14:40:35 1992 D:\P2\DATA\ 100 100 0
Data file FINLAND M= 11 Col= 3 M(active)= 4 M(protected)= 6
1 S AP- Commune Name of commune
2 S -PM Province [UUS,TUR,AHV,HAM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
3 4 -PF Popul Estimated population in 1980 (#####)
4 4 APF Births Live births in 1978 (#####)
5 8 -PR Area Land area square km in 1979 (#####.##)
6 S API Agri Population in agriculture (10%)
7 S --I Industry Population in industry (10%)
8 S --I Service Population in other occupations (10%)
9 4 --F Houses New houses and apartments in 1978 (#####)
10 4 A-I Tax Communal tax rate in 1979 (##.##) {12,20}
11 2 --I Income Income per inhabitant in 1979 (#####)

Enter scale types: N=nominal, O=order, I=interval, R=ratio, --no

```

The notations in the third mask column enable the statistical modules to make decisions about which variables are statistical (or computable). They can also give warnings if the variables selected do not meet the scaling requirements of certain statistical methods.

The possible notations in the scale type column will be explained later in connection with statistical operations (See 7.1.6 and SCALES?).

Since the selections in the mask columns are saved as a part of the Survo data file, the user can keep control on the data without a constant need to specify selections anew when various operations are activated.

In fact, even more mask columns can be used for such purposes. The default number of mask columns in each data file is 7. The three first of them are the above-mentioned activation, protection, and scale type columns. The number of mask columns selected may be set even greater when the data file is created (See FILE? CREATE).

It is typical to define special subsets of variables in those optional mask columns. For example, variables for a linear regression analysis could be specified as follows in the last (#7) mask column. We enter the file activation mode (by the **FILE ACT** key) and press the **ESC** key. This widens the shaded area to cover all seven mask columns.

```

10 1 SURVO 84C EDITOR Tue Jan 26 14:53:33 1992 D:\P2\DATA\ 100 100 0
Data file FINLAND M= 11 Col= 7 M(active)= 4 M(protected)= 6
1 S AP- - Commune Name of commune
2 1 -PN - Province [UUS,TUR,AHV,HÄM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
3 S -PF - Popul Estimated population in 1980 (#####)
4 S APF - Births Live births in 1978 (#####)
5 4 -PR X Area Land area square km in 1979 (#####.##)
6 4 API - Agri Population in agriculture (10%)
7 4 --I - Industry Population in industry (10%)
8 4 --I - Service Population in other occupations (10%)
9 4 --F - Houses New houses and apartments in 1978 (#####)
10 4 A-I Y Tax Communal tax rate in 1979 (##.##) {12,20}
11 4 --I X Income Income per inhabitant in 1979 (#####)

To stop, press EXIT!

```

We have moved the cursor to the last mask column and declared most of the variables passive by '-'. The variables of the model are denoted by X's and Y's.

We may now return to the normal editing mode and activate a LINREG operation which gives the following results (appearing from line 4 onwards):

```

17 1 SURVO 84C EDITOR Tue Jan 26 14:58:52 1992 D:\P2\DATA\ 100 100 0
1 *
2 *LINREG FINLAND,3 / MASK=#7 RESULTS=0
3 *Linear regression analysis: Data FINLAND, Regressand Tax N=464
4 *Variable Repr. coeff. Std. dev. t beta
5 *Area 0.000156 0.000030 5.285 0.210
6 *Income -0.000146 0.000012 -11.67 -0.464
7 *constant 17.97796 0.167445 107.4
8 *Variance of regressand Tax=0.950961110 df=463
9 *Residual variance=0.681360444 df=461
10 *R=0.5353 R^2=0.2866
11 *

```

The specification MASK=#7 (on line 2) defines the 7th mask column in the FINLAND data file to give the variables belonging to the model. Variables with a 'Y' mask are considered regressands and variables with 'X' are regressors.

If no MASK specification were given, the first mask column would have been used. In this example it would lead to an error message, since the first mask column has plain A's and does not define any regression model.

VARS and MASK specifications

There are also other means for selection of variables. For example, the same regression model could have been defined by the VARS specification

```
VARS=Tax(Y), Area(X), Income(X)
```

or by entering an explicit mask

```
MASK=----X----YX
```

which gives the mask symbols for each variable separately in the order they appear in the data file. The default mask symbols in VARS are A's. Thus

VAR\$=Tax,Area,Income is same as MASK=----A----AA .

VAR\$=ALL sets all the variables active in the current data set.

VAR\$=ALL, -Commune, -Tax means that all variables except Commune and Tax are to be selected.

If both VAR\$ and MASK are given, VAR\$ determines the selection. The influence of VAR\$ and MASK is only temporary and they do not alter the activation status in the data file. The VAR\$ and MASK specifications are valid also when selecting fields in data lists and tables.

6.6 Selecting observations

In many applications we need means for defining various subsets of observations (cases) as well. The essential tools for restricting observations are specifications IND, CASES, and SELECT.

All statistical operations working on source data sets (with the only exception: COMPARE) observe the limitations set by these specifications.

For example, from the data file FINLAND, to load values for variables 'Province', 'Popul', 'Tax', and 'Income' for those cases which are either in province 'UUS' or 'TUR' and have a population larger than 50000, we use the FILE LOAD operation as follows:

```

18 1 SURVO 84C EDITOR Mon Jan 27 09:01:28 1992 D:\P2\DATA\ 100 100 0
24 *
25 *VAR$=Commune,Popul,Tax,Income
26 *CASES=Province:UUS,TUR IND=Popul,50000,500000
27 *FILE LOAD FINLAND
28 *DATA FINLAND*,A,B,C
29 C Commune      Popul      Tax      Income
30 A Espoo        133556  14.50   24665
31 * Helsinki    483057  15.00   26875
32 * Pori        79261   17.00   16722
33 * Turku       164081  15.00   19154
34 B Vantaa      129669  15.00   21317
35 *

```



IND?
CASES?
SELECT?

The IND specification has three forms which correspond to following conditions (X is a name of a field, a,b numeric values):

| | Condition |
|-----------|-----------|
| IND=X | X=1 |
| IND=X,a | X=a |
| IND=X,a,b | a<=X<=b |

The CASES specification has the form

CASES=STRVAR:ABC,DEF,GHI

implying that only cases where the string variable STRVAR has a value ABC or DEF or GHI will be processed.

If IND and CASES appear simultaneously (as in the preceding example), only observations which conform to both conditions are active.

***SELECT specification**

The **SELECT** specification allows combining of several conditions of type **IND** and **CASES** in a form of a general Boolean expression. Any conceivable logical condition can be expressed by means of the **SELECT** specification.

SELECT=A+B+C+ ... (read: A or B or C or ...)

selects observations satisfying at least one of the alternative conditions A, B, C, ... where each condition (say A) is of the form

A=A1*A2*A3* ... (read: A1 and A2 and A3 and ...).

Each of the conditions A1, A2, A3, ... must be given as a specification of the form

A1=<variable>, <lower_limit>, <upper_limit> (as in **IND**)

or

A1=<string_variable>:<case1>,<case2>, ... (as in **CASES**).

Any words can be used in place of A1, A2, A3, ..., B1, B2, B3, ..., etc.

The **SELECT** specification will be considered only after the observation has passed the potential **IND** and **CASES** conditions.

For example, the following **FILE LOAD** scheme finds those records from **FINLAND** where

18000<=Income<=40000 and Service=5,6,7,8 or 9

and

either **Province is OUL or LAP and Popul>=10000**

or **Province is UUS or KYM or TUR and Popul>=20000**

```

18 1 SURVO 84C EDITOR Mon Jan 27 09:23:37 1992 D:\P2\DATA\ 100 100 0
37 *
38 *IND=Income,18000,40000 CASES=Service:5,6,7,8,9
39 *SELECT=North*Over10000+South*Over20000
40 *North=Province:OUL,LAP South=Province:UUS,KYM,TUR
41 *Over10000=Popul,10000,50000 Over20000=Popul,20000,50000
42 *VARS=Commune,Popul,Tax,Income
43 *FILE LOAD FINLAND
44 *DATA FINLAND*,D,E,F
45 F Commune Popul Tax Income
46 D Espoo 133556 14.50 24665
47 * Helsinki 483057 15.00 26875
48 * Järvenpää 23129 15.50 18000
49 * Kerava 23300 16.00 19613
50 * Kouvola 30906 15.00 18953
51 * Oulu 93230 16.25 18061
52 * Rovaniemi 29559 17.00 18099
53 * Turku 164081 15.00 19154
54 * Tuusula 22036 14.75 18505
55 E Vantaa 129669 15.00 21317
56 *

```

To describe a complicated set of conditions permanently, it is worthwhile to create special indicator (dummy) variables, for example, by a **VAR** operation with appropriate conditional specifications. Thereafter the same conditions are easily expressed by means of these indicator variables.

6.7 Sorting of data files

The **FILE SORT** operation sorts a data file according to a sort key given as a list of fields and saves the sorted records in a new data file with the same structure as the original one.

The optional **IND**, **CASES** and **SELECT** specifications limit the set of records to be sorted.

For example,

```
FILE SORT FINLAND BY Province, -Popul TO FINLAND2
```

sorts the communes in **FINLAND** file using 'Province' as the primary sort key and then within each province the cases are sorted in descending order with respect to 'Popul'. Without a '-' in front of 'Popul' the communes would have been sorted in ascending order. The sorted records will be saved in **FINLAND2**. The previous contents of **FINLAND2** is lost.

Up to 10 sort keys are allowed and their order in the **FILE SORT** command determines the precedence order. Both string and numeric fields can be used. If ties occur, the original order is preserved. Also, a part of a string field may be applied by indicating the first and last byte in the form

```
<name_of_the_field>[ <first_byte>: <last_byte> ].
```

For example, in **FINLAND**, notation **Province[1:3]** is same as **Province** since the length of the field is 3.

If string fields appear as a part of the sort key, the characters are sorted according to the order specified by a **FILTER=<code_file>** specification. A complete pathname must be given for the code file. Default is

```
FILTER=.\SYS\SORTCODE.BIN
```



which equates lower case letters to upper case ones.

One can study existing code files (extension **.BIN**) and create new code files by using the **CODES SAVE** and **CODES LOAD** commands (**CODES?**).

The **FILE SORT** operation works in following steps. At first the original data set is scanned and the sort keys are formed in the central memory. If the memory requirements of the sort keys exceed the space available in the central memory, **FILE SORT** performs internal sorts for subsets of data, saves the sorted keys in intermediate files and merges these files in one or several passes. Finally, the original data set will be moved to the sorted file according to the last merged file of the sort keys. Thus, in principle there are no limits for the size of the data to be sorted.

The number of intermediate files merged in one pass is 4 by default. This number can be altered by the specification **FILEMAX=<#_of_files>**. Maximum is 12.

As an option, only a limited number of the records in the sorted data are moved to the new data file. By giving `NSORT=k`, only `k` first of the sorted records are moved.

This is an useful feature when `FILE SORT` is used for taking random samples. Then a random variable, say `R`, is computed by `VAR R=rnd(0)` and the data is sorted with respect of `R` by

```
FILE SORT <data_file> BY R TO <new_data_file>
NSORT=<samplesize> .
```



SAVESORT?

6.8 Copying and merging of data files

Survo data files are copied and merged in different ways by the `FILE COPY` operation. First of all, `FILE COPY` is used simply for copying a given source data file (or a data list or a data table) to another (new) target data file.

If the target file `DATA1` already exists,

```
FILE COPY DATA2,DATA1
```

copies the records (possibly limited by `IND`, `CASES` and `SELECT`) of `DATA2` to the end (as new records) of `DATA1`. Only active fields of `DATA2` are copied and `DATA1` must have fields with the same names. If not, an error message is displayed and nothing is copied. The types and the lengths of the corresponding fields need not be same. Hence, `FILE COPY` is also a handy tool for changing types of the fields.

If `DATA1` does not exist at all, a new data file is created with a structure corresponding to the active fields of `DATA2`. In this case, a considerable amount of extra space is reserved for possible new fields in `DATA1`. This feature makes possible to extend the capacity of a data file by setting all its fields active, copying the contents to a new file, and finally, deleting the old version and renaming the new one with the original name.

By using `FILE COPY` repeatedly, several data sets can be merged. In the next example, data from the largest towns in provinces `UUS`, `TUR`, and `HÄM` (and in this order) are copied to a new file `FIN2` by using `FILE COPY` three times (lines 62-64). The file thus created is loaded into the edit field by the `FILE LOAD` command on line 67.

```

15 1 SURVO 84C EDITOR Sat Feb 01 17:16:58 1992 D:\P2\DATA\ 100 100 0
58 *
59 *FILE DEL FIN2 / This command deletes FIN2.SVO if it exists.
60 *VARS=Commune,Province,Popul,Tax,Income
61 *IND=Popul,50000,500000
62 *FILE COPY FINLAND,FIN2 / CASES=Province:UUS
63 *FILE COPY FINLAND,FIN2 / CASES=Province:TUR
64 *FILE COPY FINLAND,FIN2 / CASES=Province:HÄM
65 *.....
66 *Border line above invalidates the CASES (and other) specifications.
67 *FILE LOAD FIN2
68 *DATA FIN2*,G,H,I
69 I Commune Pro Popul Tax Income
70 G Espoo UUS 133556 14.50 24665
71 * Helsinki UUS 483057 15.00 26875
72 * Vantaa UUS 129669 15.00 21317
73 * Pori TUR 79261 17.00 16722
74 * Turku TUR 164081 15.00 19154
75 * Lahti HÄM 94845 15.85 17468
76 H Tampere HÄM 165453 15.00 18546
77 *

```

MATCH specification

By using a **MATCH** specification, data files are merged by extending records of DATA1 by the corresponding (matching) records of DATA2.

MATCH=<match_field>

implies that merging is to be controlled by a given field which must be defined in both files. Then the first active record of DATA2 is compared for the value <match_field> with the first record of DATA1. If the values are the same, the values of the active variables in DATA2 are joined to the first record of DATA1. If not, the second case of DATA1 is tested and so on, until a match occurs. Thereafter the second active record of DATA2 is compared with the next records of DATA1.

All active fields thus copied from DATA2 to DATA1 must be defined also in DATA1.

By entering **MATCH=#**, the index (number) of the record serves as a match field. This is, maybe, the most usual case.

When a match field is used, **FILE COPY** assumes that each record to be copied from the source file is also present in the target file. If not, an error message will be given. Such odd records (without counterparts in the target file) are ignored by entering a specification **ODD=<variable>**. Then, for odd records, **FILE COPY** will write value 1 in the **ODD** variable of the source file and common records will be copied according to **MATCH**. It is the user's responsibility to initialize the **ODD** variable to 0 by a **VAR** operation before **FILE COPY**. After **FILE COPY**, the **ODD** records can be easily recognized and, for example, copied after the common records in the target file (by **FILE COPY** without **MATCH**). The **ODD** variable is omitted by entering **ODD=NULL** or by omitting **ODD** and giving a **MODE=1** specification.

If common records are not in the same order in both files, one must give either **ODD=<variable>,2** or **MODE=2** specification. In this case, the process may

be much slower.

Assume that we want to copy values of the 'Area' field from FINLAND to FIN2. Since that field does not exist in FIN2 before, it must be defined at first. In the following display this is done by a VAR operation (on line 80). Then the records are copied by a FILE COPY operation. To match the cases, we use the MATCH=Commune specification. Since the communes in FINLAND are in alphabetic order but in FIN2 not, the MODE=2 specification must be added, too.

```

15 1 SURVO 84C EDITOR Sat Feb 01 17:49:12 1992 D:\P2\DATA\ 100 100 0
79 *
80 *VAR Area:8=MISSING TO Fin2
81 *.....
82 *VARS=Area MATCH=Commune
83 *MODE=2 (because the records are in different order)
84 *FILE COPY FINLAND,FIN2
85 *.....
86 *FILE LOAD FIN2
87 *DATA FIN2*,J,K,L
88 L Commune          Pro   Popul   Tax  Income          Area
89 J Espoo            UUS  133556 14.50 24665          308.870
90 * Helsinki        UUS  483057 15.00 26875          176.870
91 * Vantaa           UUS  129669 15.00 21317          240.020
92 * Pori             TUR   79261 17.00 16722          476.150
93 * Turku           TUR  164081 15.00 19154          234.860
94 * Lahti           HAM   94845 15.85 17468          133.490
95 K Tampere         HAM  165453 15.00 18546          523.700
96 *

```

In previous forms of FILE COPY, it has been assumed that each record in the source data is copied only once and then to the first matching record of the target file.

In some cases, it is required that for each record in the target file we have to copy the first matching record from the source data. Then it is typical that one source record is copied to many records in the target file. For example, we want to join aggregated data to each record separately.

This kind of performance is achieved by the specification MODE=3. Then also possible IND, CASES, and SELECT specifications refer to the target file and not to the source file as in previous modes of FILE COPY. An example will be given in the section 6.14.

With a MFCOPY operation (a separate option by M.Korhonen, reported in SURVO 84C Contributions No.5, 1992) several *related* Survo data files can be combined into a new data file. The relation of between the data files is indicated by key variables.

6.9 Text files

Although text (ASCII) files cannot be used in typical Survo operations as such, it is easy to convert them into Survo data files. On the other hand, Survo data files can be transformed into text files when needed. The Survo output files are text files in themselves.

The **FILE SAVE** operation converts text files (or parts of them) into Survo data files. Since the other general software systems should be able to produce data as text files, **FILE SAVE** is a link from other programs to Survo.

For a text file to be converted, some systematic structure is essential. The file should have lines terminating to a line feed character. [The maximum length of a line is 2559 characters.](#)

It is assumed that each record is located on one line or on several consecutive lines. If more than one line is assigned to one record, the corresponding lines in different records must have the same structure (i.e. the data values cannot float freely between lines). The fields may appear in free formats or in a fixed format each. In the first case, one or more spaces is the default delimiter between fields. Also other characters can be defined for such a purpose. In the fixed format case, neighbouring fields may follow each other without delimiters. Even mixtures of both setups are allowed. Possible header lines, not belonging to the data set, and nonsystematic trailing lines must be indicated in advance.

An ideal case is a neat table of records having a constant number of fields on each line and preceded by a similar line of column (field) labels. Such a text file is automatically converted by **FILE SAVE** to a decent Survo data file without a need to give any extra specifications.

As an illustration, we consider a small data set on 4 Nordic countries. We have typed it into the edit field and saved it as a text file **NORD.TXT**.

```

21 1 SURVO 84C EDITOR Sun Feb 02 13:10:20 1992 D:\P2\DATA\ 100 100 0
25 *
26 *SAVEP 27,31,NORD.TXT
27 *Country      Coffee   Tea      Beer     Wine     Spirits
28 *Finland      12.5    0.15    54.7    7.6     2.7
29 *Sweden       12.9    0.30    58.3    7.9     2.9
30 *Norway       9.4     0.19    43.5    3.1     1.8
31 *Denmark      11.8    0.41    113.9   10.4    1.7
32 *

```

The simplest way of converting this data set into a Survo data file would have been to label it in the edit field with a **DATA** definition and use **FILE COPY** at once. However, now we assume that the text file **NORD.TXT** is the original representation of the data set (thus ignoring the table in the edit field). Then, the **FILE SAVE** operation does the job of transforming the text file **NORD.TXT** into a Survo data file **NORD.SVO** immediately as follows:



FSAVE?

```

17 1 SURVO 84C EDITOR Mon Apr 20 09:16:58 1992 D:\P2\DATA\ 100 100 0
32 *
33 *FILE SAVE NORD.TXT,NORD
34 *FILE LOAD NORD
35 *DATA NORD*,C,D,E
36 E Country Coffee Tea Beer Wine Spirits
37 C Finland 12.5 0.15 54.7 7.6 2.7
38 * Sweden 12.9 0.30 58.3 7.9 2.9
39 * Norway 9.4 0.19 43.5 3.1 1.8
40 D Denmark 11.8 0.41 113.9 10.4 1.7
41 *
42 *FILE STATUS NORD
43 * Copied from text file NORD.TXT
44 *FIELDS: (active)
45 * 1 SA_ 7 Country
46 * 2 NA_ 4 Coffee (####.#)
47 * 3 NA_ 4 Tea (#.##)
48 * 4 NA_ 4 Beer (###.#)
49 * 5 NA_ 4 Wine (##.#)
50 * 6 NA_ 4 Spirits (#####.#)
51 *END
52 *SURVO 84C data file NORD: record=53 bytes, M1=11 L=64 M=6 N=4
53 *

```

To check the result of FILE SAVE, the data file NORD.SVO has been loaded into the edit field by FILE LOAD (lines 34-40) and its structure is revealed by FILE STATUS (lines 42-52).

One can see that the FILE SAVE operation has detected the types of the columns and noticed that the first line in the text file apparently contains the column labels (because no entry in it is numeric but the number of them matches with the number of columns). If the label line were missing, FILE SAVE would use X1, X2, ... ,X6 as field names.

Also the formats of the numeric fields have been recognized and appear in the field name extensions. When selecting these formats, the number of digits in the integer part as well as in the decimal part and the length of the field name is considered.

In less ideal cases, one must indicate the line of the first record (by a FIRST specification) and eventually the last record (by LAST). If a label line exists somewhere in the text file, it is given by a NAMES specification. The default values are (in accordance with the ideal case) FIRST=1 and LAST=<# of last line>. If the NAMES specification is missing, field names in the Survo data file will be X1, X2, These names can afterwards be replaced by better ones by taking the structure of the file into the edit field by FILE STATUS, by editing them as one likes, and finally by resaving the file structure by FILE UPDATE.

Correct values for specifications FIRST, LAST, and NAMES are selected most easily by using a SHOW operation which lists the text lines with their line numbers in a temporary window. No line numbers are needed in the text file itself.

To illustrate this situation, we create another text file NORD2.TXT by a SAVEP command:

```

17 1 SURVO 84C EDITOR Sun Feb 02 13:15:44 1992 D:\P2\DATA\ 100 100 0
1 *
2 *SAVEP NORD2.TXT
3 *Consumption of various beverages in 4 Nordic countries
4 *
5 * Country      Coffee  Tea    Beer  Wine  Spirits
6 *
7 * Finland      12.5   0.15  54.7  7.6   2.7
8 * Sweden       12.9   0.30  58.3  7.9   2.9
9 * Norway       9.4    0.19  43.5  3.1   1.8
10 * Denmark     11.8   0.41  113.9 10.4  1.7
11 *
12 *

```

The text file NORD2.TXT will contain 8 lines. Of them, lines from 5 (edit line 7) onwards are data lines. Thus this information can now be converted to a Survo data file NORD2 (with the extension .SVO) by the command

```
FILE SAVE NORD2.TXT,NORD2 / FIRST=5 NAMES=3 .
```

The resulting Survo data file NORD2.SVO has exactly the same structure and contents as NORD.SVO in the previous example.

These examples were quite artificial. In practice there is no need to move data from the edit field to data files via text files. By entering a proper DATA definition in the edit field, the FILE COPY does the same job directly.

FILE SAVE with a FIELDS list

One record in a text file may occupy several lines provided that each of them has a fixed structure in all records (observations). Also fields without intervening spaces and commas may occur.

In such cases, the FILE SAVE requires a FIELDS list of a special type and the target file (Survo data file) must have been previously created by the FILE CREATE operation.

We illustrate this alternative by the following quotation from an edit field:

```

16 1 SURVO 84C EDITOR Wed Jan 28 16:26:41 1987 D:\P2\DATA\ 100 100 0
1 *
2 *SAVEP 3,8,TEST2.TXT
3 *Sat 24-01-1987 17856785 17.1
4 *-45:6 C
5 *Mon 26-01-1987 23657432 234.5
6 *34/8 A
7 *Wed 28-01-1987 95622656 56
8 *28:6 C
9 *
10 *FILE CREATE DATA2
11 * Copy of TEST2.TXT
12 *FIELDS:
13 * 1 S 4 Year 1987
14 * 2 S 2 Month 01
15 * 3 S 2 Day 24
16 * 4 S 3 Weekday Sat
17 * 5 N 1 A1 17
18 * 6 N 1 A2 85
19 * 7 N 1 A3 67
20 * 8 N 1 A4 85
21 * 9 N 4 B 17.1
22 *10 N 2 C1 -45
23 *11 N 2 C2 6
24 *12 S 1 D C
25 *END
26 *
27 *FILE SAVE TEST2.TXT,DATA2
28 *FIELDS:
29 * 1 Weekday
30 * 2 Day -
31 * 3 Month -
32 * 4 Year
33 * 5 A1 2
34 * 6 A2 2
35 * 7 A3 2
36 * 8 A4
37 * 9 B LF
38 *10 C1 :/
39 *11 C2
40 *12 D
41 *END
42 *
43 *FILE LOAD DATA2
44 *DATA DATA2*,A,B,C
45 C Year Mo Da Wee A1 A2 A3 A4 B C1 C2 D
46 A 1987 01 24 Sat 17 85 67 85 17.100 -45 6 C
47 * 1987 01 26 Mon 23 65 74 32 234.500 34 8 A
48 B 1987 01 28 Wed 95 62 26 56 56.000 28 6 C
49 *

```

The original data set consists of 3 observations taking 2 lines each. They are seen on edit lines 3-8 from which they have been saved into the text file TEST2.TXT by the SAVEP command on line 2.

Assume now that the text file TEST2.TXT is our true original data set and we have to convert it to a Survo data file DATA2.

We start by creating the data file by a FILE CREATE scheme with a FIELDS list. This scheme is shown on edit lines 10-25. As an explanation, we have listed the values of the first observation on lines 13-24 to show how the fields in this data file will correspond to the original data set.

After creating the data file with the proper structure, we enter a FILE SAVE scheme with a new FIELDS list. This is located on edit lines from 27 to 41 and shows the mapping from the 12 fields of the text file to the 12 fields of the data file.

In the FIELDS list for the fields of the text file, a field name of the data file

is given. Optionally, if the field does not terminate by a space or a comma, either the length of the field as an integer (like 2 for A1, A2 and A3) or a string of terminating characters is given. For example, C1 may terminate either by : or by / .

If a line feed is the terminating character (as for B), it is denoted by LF.

Not all fields have to be copied. If some field of the text file should be omitted, it is denoted in the FIELDS list by putting a '-' instead of a field name. In this example we did not have such fields.

As a proof of a successful conversion we have loaded the contents of DATA2 back to the edit field by a FILE LOAD operation on line 43.

Although all possible representations cannot be covered by the means mentioned in this example, this extended form of FILE SAVE will help in most cases. In complicated situations, we can always load parts of any text file to the edit field (by LOADP or SHOW) and edit the data in various ways before copying it to a data file.

Fields in the text file consisting of spaces and/or '-'s are treated as missing values. This convention can be overridden by a specification

MISSING=<string>

giving the code word <string> for a missing value. For example, **MISSING=.** implies all fields with at least two consecutive '.'s to be missing values.

The default mode (**MODE=1**) considers each space a delimiter in the text file also when other symbols are given as delimiters. Thus fields consisting of several words are not treated correctly. In such cases a specification **MODE=2** should be given. Then also empty fields terminating with LF will be accepted.

MATCH specification

As in FILE COPY, mapping from the source (text file) to the target (Survo data file) with respect to records can be controlled by a MATCH specification. If MATCH is omitted, the records will be copied to the Survo data file as new records.

MATCH=<field name>

specifies the field in the Survo data file whose contents should be equal to the corresponding field of the text file. The MATCH field has to be given as one of the items in the FIELDS list. Typically, the name of the record serves as a match field. The order of the records in the text file and in the data file must be the same. Use either a string field or an integer-valued numeric field as a MATCH field. Non-integer fields are not safe as MATCH fields because their values may be rounded. By giving **MATCH=#**, the order of the record will be used as a match variable.

Code conversions

In all forms of FILE SAVE, code conversions may be performed by an extra specification FILTER=<conversion_file>. For making code conversion files, see CODES? . Unwanted characters (after conversion) are skipped by SKIP=<list_of_characters>. Example: SKIP=, "

*Saving text data

Words or characters in a text file can be saved in a Survo data file as values of a string field by entering a specification either of form

```
FORMAT=WORD, <variable>, "delimiters"
```

or

```
FORMAT=CHAR, <variable>, <length> .
```

If FORMAT is given, the FIELDS list is not used at all.

In the former case, the text in the text file is split into "words" limited by line feeds and by any of the characters given as "delimiters". Default setting for delimiters is " , . : ; ? ! - " . The words are appended to the end of the data file as values of <variable>.

In the latter case, code sequences of <length> characters are the words to be saved as values of <variable>. Default is <length>=1. Spaces (decimal code 32) are skipped.

By using the FILTER specification with an appropriate conversion file, selected codes may be merged or skipped (by mapping them to spaces).

The next example is an application of FILE SAVE of this type.

```
14 1 SURVO 84C EDITOR Fri Feb 20 19:30:27 1987 D:\P2\DATA\ 100 100 0
1 *
2 *FILE CREATE ABCDE,1,1
3 *File for codes A,B,C,D,E
4 *FIELDS:
5 *1 SA_N 1 Code
6 *END
7 *
8 *SAVEP 9,11,TEST.TXT
9 *CBAABAACBACDABDDCABBBCDDBACDDBCDABBBCDAADDCBDBAADDCCBDDADCCDDDBCCBDCDAADC
10 *DCECEDACBBADCDBEBBCDEAACBDDDEDAADDECCBDDDEEAADCCBDDCCAAADCCDDAAAEAA
11 *DDEAAADCCAAABCDDAAADCCBBECCDDBBCCDDECCBDDDDDCDCDDDE
12 *
13 *FILE SAVE TEST.TXT,ABCDE
14 *FORMAT=CHAR,Code
15 *
16 *STAT ABCDE,17
17 *Basic statistics: ABCDE N=199
18 *Variable: Code
19 *Nominal scale
20 *entropy=2.19525 (94.5%)
21 *Code          f          %          *=2 obs.
22 *A              40      20.1 *****
23 *B              32      16.1 *****
24 *C              50      25.1 *****
25 *D              62      31.2 *****
26 *E              15       7.5 *****
27 *
```

Our task has been to make a summary of data originally listed on edit lines 9-11. In practice, the data set may be much larger and it is readily in a text

file.

Here we create first a Survo data file ABCDE with a single one byte string variable 'Code'. 'N' in notation 'SA_N' on line 5 implies that 'Code' is measured on a nominal scale.

SAVEP 9,11,TEST.TXT on line 8 saves the source data in a text file TEST.TXT.


FILE SAVE on line 13 with the FORMAT specification on line 14 reads the text in TEST.TXT byte by byte and saves the bytes to the data file ABCDE as consecutive values of variable 'Code'.

The result is checked here by a STAT operation. It gives the frequency distribution of code symbols.

Conversion of data files to text

The FILE LOAD operation (which has already been applied in previous examples) is the main tool for transforming data in the Survo data files back to text form. FILE LOAD moves data either to the current edit field or to any text file.

FILE LOAD <Survo data file>,L

loads the active fields of the data file into the edit field from the edit line L onwards. If L is omitted, the next edit line is used. IND, CASES and SELECT specifications are also available. To indicate the fields to be loaded, a VARS or a MASK specification is used. In the default setting, the result is a valid data table with the source file name trailed by '*'. 

FLOAD?

However, if the name of the data file is preceded by a '-', the header lines are not printed in front of the data values.

An alternative form is

FILE LOAD <Survo data file> TO <name of a text file>

moving the active fields of the data file to a text file. Also the header lines giving the name of the data file and the names of the fields will be moved. If the text file does not exist, a new file is created. If the text file already exists, the converted data will be appended after the current end of the text file.

If <Survo data file> is preceded by a '-', the header lines are not moved. This is useful in situations where distinct parts of a data file are copied into a single text file.

The default representation of the output is a data matrix (table). The fields will be printed according to the images of the form (##.###) given in the field names (descriptions). If such an image is not given to a certain field, the format is selected according to the type of the field.

Other formats are chosen by a FORMAT specification. It has 3 different forms:

1. `FORMAT=ORDER: <field_1>, <field_2>, ...`
simply reorders the fields in the output.
2. `FORMAT=LIST`
gives the output in the form of a data list.
3. `FORMAT=<name_of_format>`
tells that a special (multiline) format defined in the current edit field is to be used. This format has to be given in the form:

```

FORMAT <name_of_format>
    one or several lines containing free text and field
    descriptions of the form
    name_of_field: ###.## or [name_of_field]
END

```

If `###.##` is missing, the default format for the field is used. In the latter case (name in brackets []) the default format is always used and the value overwrites the name of the field.

An application of the last type of `FORMAT` is presented in the next display.

```

20 1 SURVO 84C EDITOR Fri Feb 14 12:39:14 1992 D:\P2\DATA\ 100 100 0
1 *
2 *FILE LOAD FINLAND,9_ / FORMAT=PLAN1 IND=Popul,100000,500000
3 *FORMAT PLAN1
4 *[Commune] Province [Province]
5 *Population [Popul] Tax rate [Tax]
6 *-----
7 *END
8 *
9 *Espoo Province UUS
10 *Population 133556 Tax rate 14.50
11 *-----
12 *Helsinki Province UUS
13 *Population 483057 Tax rate 15.00
14 *-----
15 *Tampere Province HÄM
16 *Population 165453 Tax rate 15.00
17 *-----
18 *Turku Province TUR
19 *Population 164081 Tax rate 15.00
20 *-----
21 *Vantaa Province UUS
22 *Population 129669 Tax rate 15.00
23 *-----

```

6.10 Transformation of variables

The main tool for making various mathematical transformations is the `VAR` operation. It works on all data files and also on data tables with a mask line. Other transformation operations are `CLASSIFY` and `TRANSFORM`.

For derived variables (like residuals and predicted values of models) there are options in various statistical modules. For general linear combinations, a



**VARSTAT?
POWERS?**

special LINCO operation is often the best alternative. Also SER operations for time series transformations and SMOOTH for semiparametric smoothing of a data series are useful tools in data transformation. These operations will be presented later in connection with statistical analysis.

An optional MRESP operation by M.Korhonen (documented in SURVO 84C Contributions No.5) is intended to help the use of multiple responses. The operation generates a dichotomous variable for each possible response.

The VAR operation has two basic forms. The simpler one is

```
VAR <variable>=<function of other variables> TO <data>
```

allowing only one variable to be computed as a function of other variables in the data. <variable> may be an old variable already present in <data> or it may be a new one. In the latter case, it is automatically created provided that there is space for new variables. If not, an error message is displayed.

The type of a new variable can be given by writing VAR in the form

```
VAR <variable>:T=<function of other variables> TO <data>
```

where T is 1,2,4,8 or S#. In the last alternative, # is the length of the string field. The default type is 4.

An alternative form of VAR is

```
VAR <var1>,<var2>,... TO <data>
```

where one or more variables (old and new) are computed. The expressions telling the values of <var1>,<var2>,... have to be given in the edit field as specifications of the form

```
<var1>=<function of other variables in data>
```

```
<var2>=<function of other variables in data>
```

...

In all forms of VAR, the mathematical expressions are written according to the rules of editorial computing. The expressions may include constants and temporary functions defined in the edit field. All standard and library functions can be used as well. Even the control structures (if-then-else, for) of editorial computing are available. (See ARIT?)

When a VAR operation is activated, it starts by examining the variables to be computed and creates the new ones if needed. Then the values of these output variables are computed and saved for all active observations. The observations to be processed can be limited by the IND, CASES and SELECT specifications.

If missing values appear in some observations, the output values depending on such values will also be saved as missing values. The new variables which do not get values for all observations (due to IND and CASES, for example), remain indefinite in passive observations.

Therefore, it is a good practice to initialize a new variable by

```
VAR NEWVAR=MISSING TO DATA1
```

for example, before a partial transformation. Above MISSING is a special notation for a missing value.

Another solution is to use conditional statements like

```
VAR LX=if(X<=0)then(MISSING)else(log(X)) TO DATA1
```

Besides MISSING there are two other predefined variables which can be employed in VAR operations. N is the number of observations in the current data set and ORDER is the index of the current observation of the data set thus varying from 1 to N.

For example,

```
VAR Cumfreq=(ORDER-0.5)/N TO SAMPLE3
```

computes the relative cumulative frequency for (a sorted) SAMPLE3.

A variable may be transformed into itself, for example, as

```
VAR Test=if(Test=MISSING)then(3)else(Test) .
```

For time series data, both lags and leads may be used for (input) variables appearing in expressions. For example, X[-12] is the value of variable X lagged by 12 and X[+1] is the value of X one observation ahead. X[0] is the same as X. Referring by lags and leads to observations which are not available leads to missing values in output variables.

To compute a five term moving average AVE5 of X, either

```
VAR AVE5=(X[-2]+X[-1]+X[X[1]+X[2]])/5 TO DATA1
```

or

```
VAR AVE5=for(I=-2)to(2)sum(X[I]/5) TO DATA1
```

could be used. The SER operations provide faster and more general tools for such transformations (See SER?).

To minimize the processing time and to save space in large applications, it is good to restrict active fields to those appearing in the VAR operation. To reduce the processing time, library functions (like the cumulative normal distribution function N.F() which is called from disk for each observation) can be copied from the directory .\F to the ramdisk (say D:) and written as D:N.F(0,1,X). A suitable cache memory facility does the same thing automatically.

The following tiny setup exemplifies application of VAR. If we activate VAR on line 14,

```

26 1 SURVO 84C EDITOR Fri Jan 30 19:31:41 1987 D:\P2\DATA\ 100 100 0
1 *
2 *
3 *DATA TEST,A,B,N,M
4 N X Y Z U P Ave5 X2
5 M AA AAAA 12.1 123.123 1.12345 12.123 1
6 A 3 4
7 * 5 12
8 * 1 1
9 * 2 4
10 * 0 2.3
11 B 12 4.5
12 *
13 *
14 *VAR Z,U,P,Ave5,X2 TO TEST_
15 *
16 * Z=X+Y U=norm(X,Y) norm(A,B):=sqrt(A*A+B*B)
17 * P=N.F(0,1,X/10)
18 * Ave5=for(I=-2)to(2)sum(X[I]/5)
19 * X2=if(X<3)then(X)else(3)
20 *
21 *
22 *
23 *

```

the following results are obtained:

```

26 1 SURVO 84C EDITOR Fri Jan 30 19:31:47 1987 D:\P2\DATA\ 100 100 0
1 *
2 *
3 *DATA TEST,A,B,N,M
4 N X Y Z U P Ave5 X2
5 M AA AAAA 12.1 123.123 1.12345 12.123 1
6 A 3 4 7.0 5.000 0.61791 - 3
7 * 5 12 17.0 13.000 0.69146 - 3
8 * 1 1 2.0 1.414 0.53983 2.200 1
9 * 2 4 6.0 4.472 0.57926 4.000 2
10 * 0 2.3 2.3 2.300 0.50000 - 0
11 B 12 4.5 16.5 12.816 0.88493 - 3
12 *
13 *
14 *VAR Z,U,P,Ave5,X2 TO TEST_
15 *
16 * Z=X+Y U=norm(X,Y) norm(A,B):=sqrt(A*A+B*B)
17 * P=N.F(0,1,X/10)
18 * Ave5=for(I=-2)to(2)sum(X[I]/5)
19 * X2=if(X<3)then(X)else(3)
20 *
21 *
22 *
23 *

```

In this case, the object of transformation is a data table with a mask line. One can directly check the results which are shaded for illustration.

As a more practical example, we perform certain operations on the FINLAND data file. We would like to know the communes having the highest birth rate. We start by computing a new variable $Brthrate=1000*Births/Popul$ (i.e. the number of births for 1000 people) and sort the data in descending order with respect to it. Finally we load the first communes of the sorted data to the edit field. The next display tells all the actions needed:

```

15 1 SURVO 84C EDITOR Fri Feb 14 19:45:28 1992 D:\P2\DATA\ 100 100 0
43 *
44 *VAR Brthrate=1000*Births/Popul TO FINLAND
45 *FILE SORT FINLAND BY -Brthrate TO FIN2
46 *
47 *FILE LOAD FIN2 / IND=ORDER,1,12 VARS=Commune,Province,Brthrate
48 *DATA FIN2*,A,B,C
49 C Commune Pro Brthrate
50 A Jämsä KES 25.825
51 * Luoto VAA 23.232
52 * Kaskinen VAA 23.085
53 * Raahe OUL 22.821
54 * Oulunsalo OUL 21.957
55 * Perho VAA 21.802
56 * Utsjoki LAP 20.777
57 * Uusikaupunki TUR 19.422
58 * Oulu OUL 19.318
59 * Oulainen OUL 19.298
60 * Kempele OUL 19.276
61 B Ranua LAP 18.804
62 *

```

The three operations (on lines 44,45 and 47) have been activated in this order. The border line 46 is a precaution. If for some reason the same operations were activated anew (after changing something in the setup), the specifications IND and VARS intended for FILE LOAD only would not have any effect in preceding operations.

In general, it is wise to use border lines to avoid confusions between various operations and work schemes.

If we now continue by looking at the status of the FINLAND file by a FILE STATUS FINLAND command written on line 63,

```

20 1 SURVO 84C EDITOR Fri Feb 14 19:51:08 1992 D:\P2\DATA\ 100 100 0
59 * Oulainen OUL 19.298
60 * Kempele OUL 19.276
61 B Ranua LAP 18.804
62 *
63 *FILE STATUS FINLAND
64 * The Finnish towns and communes in alphabetic order
65 * Data mainly from years 1978-80.
66 * SORT:Commune
67 *FIELDS: (active)
68 * 1 SAP 16 Commune Name of commune
69 * 2 SA- 3 Province [UUS,TUR,AHV,HÄM,KYM,MIK,KAR,KUO,KES,VAA,OUL,LAP]
70 * 12 NA- 4 Brthrate ~1000*Births/Popul
71 *END
72 *SURVO 84C data file FINLAND: record=128 bytes, M1=30 L=64 M=12 N=464
73 *

```

the specification VARS=Commune,Province,Brthrate on line 47 determines the active fields in this operation, too.

Please note also that the VAR operation has included the expression ~1000*Births/Popul of the new variable 'Brthrate' in its name extension (line 70). The character ~ in front of this comment indicates that it is written by the Survo system and not by the user. The system is allowed to overwrite comments in field names starting by the character ~ but not others. Thus, if you are using some field for temporary transformations again and again, the extended field name always tells the expression of the last transformation. This facility simplifies keeping an account of transformed variables.

As a more complicated transformation, we define an auxiliary variable 'Urban' in FINLAND getting value 1 when a commune has characteristics typical for a town or a city and 0 otherwise:

```

27 1 SURVO 84C EDITOR Sat Feb 15 16:59:55 1992 D:\P2\DATA\ 100 100 0
1 *
2 *VAR Urban:1=if(Popul/Area>100)then(1)else(Other) TO FINLAND
3 *Other=if(Popul<10000)then(0)else(Other2)
4 *Other2=if(Agri>2)then(0)else(1)
5 *
6 *STAT FINLAND,7 / VARS=Urban
7 *Basic statistics: FINLAND N=464
8 *Variable: Urban ~if(Popul/Area>100)then(1)else(Other)
9 *min=0 in obs.#1 (Alahärmä)
10 *max=1 in obs.#6 (Anjalankoski)
11 *mean=0.204741 stddev=0.403948 skewness=1.461864 kurtosis=0.134892
12 *Urban f % *=8 obs.
13 * 0 369 79.5 *****
14 * 1 95 20.5 *****
15 *
16 *FILE REDUCE FINLAND,12,464
17 *

```

On lines 2-4 we have a conditional VAR operation. It computes a new variable 'Urban' of numerical type 1. A commune is considered urban only if its population density is over 100 inhabitants in square kilometer or its total population is at least 10000 and proportion of people in agriculture is at most 20 per cent.

The STAT operation (on line 6) gives the statistical summary for 'Urban' indicating that about one fifth of the communes could be considered urban according to these criteria.

At this stage, the number of fields in FINLAND has grown to 13 since there are 11 original fields and 2 more ('Brthrate', 'Urban') generated by VAR operations. Both fields and records can be deleted from the end of the file simply by a FILE REDUCE command. Here FILE REDUCE FINLAND,12,464 just deletes the last field 'Urban', but leaves other fields and records intact.

In order to delete fields and records which are not at the end of the file, one has to set these fields passive and use a FILE COPY operation with suitable conditions for records.

*Using data from other data sets

Fields from other Survo data sets can be used as input information in VAR transformations. Such secondary data sets to be employed by VAR are given by a specification

```
INDATA=<data_1>,<data_2>,...
```

where <data_1>,<data_2>,... are Survo data sets of any type. Number of them is limited to 12. The user is, of course, responsible for similarity of data sets with respect to the order of the records.

A too small value of FILES in the MS-DOS system file CONFIG.SYS also restricts the number of simultaneously open files. Thus, if VAR is giving a message 'Data file ... not found!', the cause for this can also be a too small FILES value in CONFIG.SYS.

In VAR operations, a field in <data_i> is denoted by $D_i : \langle \text{fieldname} \rangle$, $i=1, 2, \dots$. As before, lags and leads are indicated in brackets []. For example,

```
VAR Change=Price-D1:Price TO DAT87 / INDATA=DAT86
```


computes 'Change' to data DAT87 as a difference of 'Price' between DAT87 and DAT86.

Special VAR transformations

Certain data-dependent transformations are often needed in statistical analysis.

```
VAR <y_variable>=#<function>( <expression> ) TO <data>
```

makes such special transformations of <expression> to <y_variable>. In most cases <expression> is simply a variable. The following #<functions> are available:

| | | |
|---------|--|---|
| #RANK | (ranks of values, tied observations with average rank) |  |
| #NRANK | (ranks of values, tied observations with lowest rank) | |
| #NORMAL | (normalized values with original mean and variance) | |
| #STD | (standardized values with mean=0, std.dev=1) | |
| #TRUNCP | (outliers specified by P are replaced by missing values) | |
| #WINSP | (outliers specified by P are replaced by border values) | |
| #TRUNCL | (outliers specified by LEVEL are replaced by missing values) | |
| #WINSL | (outliers specified by LEVEL are replaced by border values) | VARR? |

In #TRUNCP and #WINSP, $100(1-P)\%$ of the observations are considered outliers. P is given by the specification $P=\langle \text{value} \rangle$ and default is 0.95.

In #TRUNCL and #WINSL, an observation X is an outlier if

$$\text{abs}(X - \text{mean}) > \text{level} * \text{stddev}.$$

Level is given by the specification $\text{LEVEL}=\langle \text{level} \rangle$ and default is 1.96.

For example,

```
VAR Y=#NORMAL(X) TO DATA1
```

computes normalized X values as a variable Y. The following display illustrates how the #STD and #NORMAL functions work in practice:

```

39 1 SURVO 84C EDITOR Sun Feb 16 09:18:29 1992 D:\P2\DATA\ 100 100 0
1 *
2 *DATA UNIFORM,A,B,N,M
3 N U S NORM NDEC
4 M 11 11.111111 11.111111 11.111111
5 A 1 -1.486301 -1.644854 -1.644854
6 * 2 -1.156012 -1.036433 -1.036433
7 * 3 -0.825723 -0.674490 -0.674490
8 * 4 -0.495434 -0.385320 -0.385320
9 * 5 -0.165145 -0.125661 -0.125661
10 * 6 0.165145 0.125661 0.125661
11 * 7 0.495434 0.385320 0.385320
12 * 8 0.825723 0.674490 0.674490
13 * 9 1.156012 1.036433 1.036433
14 B 10 1.486301 1.644854 1.644854
15 *
16 *VAR S=#STD(U) TO UNIFORM
17 *VAR NORM=#NORMAL(S) TO UNIFORM
18 *VAR NDEC=N.G(0,1,(U-0.5)/N) TO UNIFORM_
19 *

```

Originally, we have a 'sample' of 10 U values 1,2,3,...,10 in the data table UNIFORM. These values are standardized by the first VAR operation on line 16 giving the column S. The S values are then normalized by the next VAR operation and we get the column NORM. In fact, irrespective of the starting values U, the NORM values should be the p fractiles for $p=0.05, 0.15, 0.25, \dots, 0.95$ for the standard normal distribution as ascertained by a direct computation by the last VAR operation on line 18.

String manipulation

If a string variable holds numerical information, it can be treated like a numerical variable in numerical transformations.

To modify true textual information in string variables, a VAR operation of the form

```
VAR str(U,p,len)=str(V1,p1,len1)&str(V2,p2,len2)& ...
```

where U,V1,V2, ... are string variables, is available. Above, `str(U,p,len)` denotes a partial string of U starting from position 'p' and having length of 'len' bytes. Alternative notations are `str(U)` (the whole string field), `str(U,p)` (the whole field from 'p' onwards). '&' indicates concatenation (chaining) of the strings. On the right hand side, also constant strings of the form "ABC" can appear.

`pos(S,p,string)` is a useful auxiliary function giving the position of 'string' in the string variable S after the 'p'th position. `pos(S,string)` is the same as `pos(S,1,string)`. For example, if S is "ABC:123:X" then `pos(S,:)=4`, `pos(S,5,:)=8`, and `pos(S,Y)=0` since Y does not appear at all.

The next example illustrates how to operate with text fields:

```

32 1 SURVO 84C EDITOR Sat Feb 15 18:58:09 1992 D:\P2\DATA\ 100 100 0
55 *
56 *DATA TEST,a,b,n,m
57 m AAAAAA AAAAAAAA AAAA AAAAAAAA A
58 n Code1 Code2 Year1 Code3 len
59 a 281037 AB/1967 1937 1967:AB 7
60 * 061217 A/1978 1917 1978:A 6
61 * 120558 ACD/1977 1958 1977:ACD 8
62 b 190939 D/1989 1939 1989:D 6
63 *
64 *VAR str(Year1)="19"&str(Code1,5) TO TEST
65 *VAR str(Code3)=str(Code2,pos1+1,4)&":"&str(Code2,1,pos1-1) TO TEST
66 * pos1=pos(Code2,/)
67 *VAR len=pos(Code2,sp)-1 TO TEST_
68 *

```

Above, 'sp' is the notation for the space character. Similarly ',' is denoted by 'comma'.

In FINLAND, the lengths of the names of the communes are computed as a one-byte numerical variable 'Namelen' as follows:

```

27 1 SURVO 84C EDITOR Sat Feb 15 19:06:47 1992 D:\P2\DATA\ 100 100 0
17 *
18 *VAR Namelen:1=pos(Commune,1,sp)-1 TO FINLAND
19 *STAT FINLAND,20 / VARS=Namelen
20 *Basic statistics: FINLAND N=464
21 *Variable: Namelen ~pos(Commune,1,sp)-1
22 *min=2 in obs.#59 (Ii)
23 *max=16 in obs.#291 (Pietarsaaren_mlk)
24 *mean=7.825431 stddev=2.172308 skewness=0.321878 kurtosis=-0.062531
25 *autocorrelation=0.1116
26 *lower_Q=6 median=8 upper_Q=9
27 *Namelen f % *=2 obs.
28 * 2 1 0.2 :
29 * 3 1 0.2 :
30 * 4 14 3.0 *****
31 * 5 62 13.4 *****
32 * 6 62 13.4 *****
33 * 7 74 15.9 *****
34 * 8 68 14.7 *****
35 * 9 74 15.9 *****
36 * 10 64 13.8 *****
37 * 11 26 5.6 *****
38 * 12 7 1.5 ***
39 * 13 6 1.3 ***
40 * 14 4 0.9 **
41 * 16 1 0.2 :
42 *
43 *FILE REDUCE FINLAND,12,464_
44 *

```

The VAR operation on line 18 seeks the position of the first space in the 'Commune' field and subtracts 1 from the result thus giving the length of the name. The STAT operation (line 19) gives the frequency distribution of 'Namelen' and other appropriate statistics. The FILE REDUCE command (line 43) removes the 'Namelen' field.

6.11 Generating data by simulation

The VAR operation can be employed also for generating data from various stochastic distributions. The core in such tasks is the `rnd` function which generates a random value in the interval (0,1). By using `rnd(x)` as an argument in an inverse distribution function, variates from that distribution are obtained.

A data file for such artificial observations is usually generated by a FILE CREATE scheme and the file is then initiated by a selected number of (missing) observations by a FILE INIT command.

The following display presents an experiment where 1000 observations from a bivariate normal distribution are produced:

```

16 1 SURVO 84C EDITOR Sun Feb 16 10:18:22 1992 D:\P2\DATA\ 120 80 0
1 *
2 *FILE CREATE BINORM,8,2
3 *FIELDS:
4 *1 N 4 X
5 *2 N 4 Y
6 *END
7 *
8 *FILE INIT BINORM,1000
9 *
10 *VAR X,Y TO BINORM
11 *X=probit(rnd(3))
12 *Y=X+probit(rnd(3))
13 *
14 *PLOT BINORM,X,Y
15 *XSCALE=-4(+1)4 YSCALE=-6(+1)6
16 *SIZE=1164,1164 DEVICE=PS
17 *

```

FILE CREATE on line 2 creates a data file BINORM with two single precision variables X,Y. Totally 4+4=8 bytes is reserved for one observation.

FILE INIT BINORM,1000 adds 1000 missing observations to BINORM. Since the file after creation is empty, we have now room for 1000 observations to feed in.

The VAR operation on line 10 computes X as a normal random deviate and Y as the sum X and another independent normal random deviate. Instead of the probit function, we could use the $N.G(0,1,X)$ function as well, but the former one is faster although not as accurate.

The argument of the `rnd` function (3) tells that the seed number 3 for the random number generator should be selected. The basic routines for pseudorandom integers are the C functions `rand` and `srand` in the *Microsoft C run-time library*. The argument of `rnd` can be any integer from 1 to 65535 giving a series of pseudorandom numbers from various seeds. Thus argument 3 gives the same numbers whenever it is used, but these are different from the series that the other seed numbers give.

To produce random numbers which are different in each trial without need to change the argument of `rnd`, the value 0 can be employed. `rnd(0)` will always be computed from a seed number which is dependent on current time and on the preceding random numbers.

The experiment is here concluded by plotting the sample generated by a

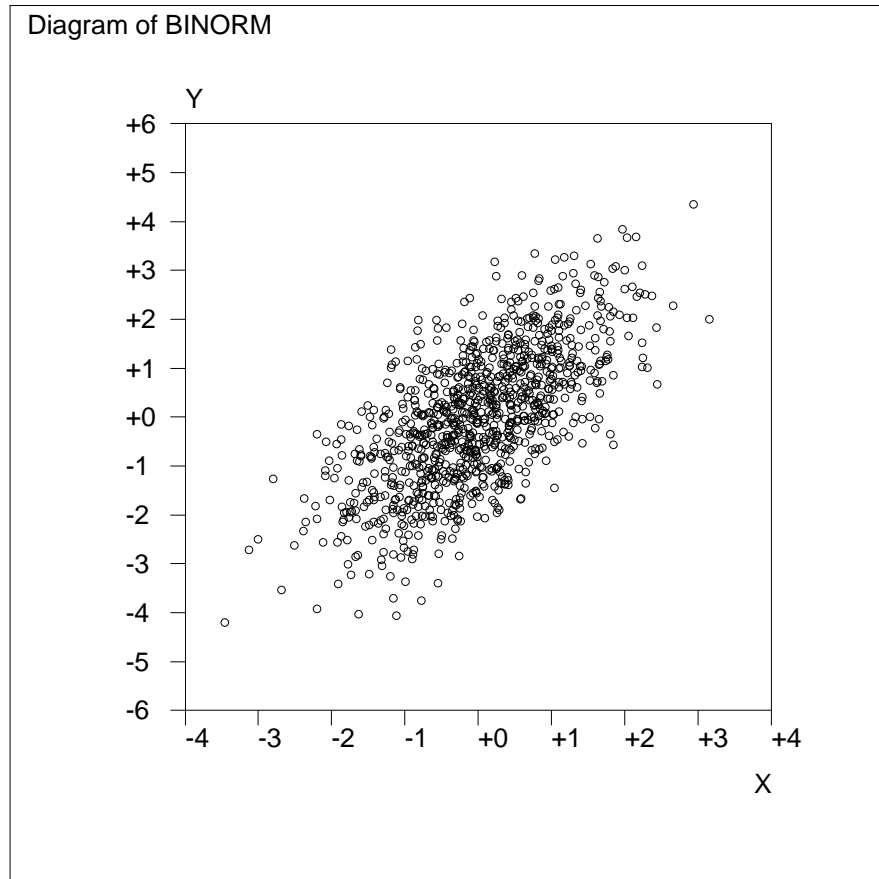


MNSIMUL?



RAND?
RNDTEST?
TRANSFORM?

PLOT scheme on lines 14-16.



6.12 Generating new variables by classification

In many applications simple rearrangements or classifications of values in existing variables are needed. In such cases a special operation **CLASSIFY** is usually more convenient than **VAR**.

CLASSIFY <data> ,<classification> ,<input_variable> ,<output_variable> computes the values of <output_variable> as a function of <input_variable> in Survo <data> according to a given <classification>. Specifications **IND**, **CASES** and **SELECT** are used for selecting observations.

Classification is defined in the edit field as follows:

```

CLASSIFICATION <classification>
<values_of_input_variable>: <value_of_output_variable>
<values_of_input_variable>: <value_of_output_variable>
...
END

```

<input_variable> and <output_variable> can be numeric or string variables. If <output_variable> does not exist (in a Survo data file), a new variable will be automatically created.

For example,

```

CLASSIFICATION NUMBERS
1,3,5,7,9: odd
0,2,4,6,8: even
END

```

maps values 1,3,5,7 and 9 to 'odd' and 0,2,4,6 and 8 to 'even'.

```

CLASSIFICATION C1
0,99,MISSING: MISSING
11 - 20:      1
21 - 40:      2
OTHERS:       3
END

```

interprets values 0, 99 and missing values of <input_variable> as missing values, values from 11 to 20 as 1, values from 21 to 40 as 2 and all other values as 3.

```

CLASSIFICATION C2
M1,M2,M31:  A
M3 - M6:    B
M7,N        C
D,E,F,G:    SAME
OTHERS:     MISSING
END

```

maps values of a string variable. For example, M315 -> A, but M302 -> B. Thus if the alternatives overlap, the first valid case is selected. Keyword SAME can be used to indicate persistent values. Above, values starting with 'D', 'E', 'F' or 'G' do not change in classification.

The following display illustrates application of the last classification.

```

31 1 SURVO 84C EDITOR Sun Mar 15 16:52:44 1987 D:\P2\DATA\ 120 80 0
1 *
2 *CLASSIFICATION C2
3 *M1,M2,M31: A
4 *M3 - M6: B
5 *M7,N C
6 *D,E,F,G: SAME
7 *OTHERS: MISSING
8 *END
9 *
10 *CLASSIFY SAMPLE,C2,CODE1,CODE2
11 *
12 *DATA SAMPLE,A,B,N,M
13 M AAAAAA AAAAAA
14 N CODE1 CODE2
15 A M567 B
16 * M307 B
17 * M319 A
18 * F257 F257
19 * N989 C
20 * M999
21 * M
22 * N C
23 B M63456 B

```



CLASSIFY?

The new CODE2 column produced by CLASSIFY has a gray shading.

The following options are possible when the input variable is a string. To indicate space characters in the classification list, any character, say `_`, can be selected by entering a specification `SPACE=_`. Then, for example,

```

CLASSIFICATION C3
___: 0
A_,_A_,_A: 1
AA_,_A_,_AA: 2
AAA: 3
END

```

gives the number of 'A's in a three-character string field. The true values of that string field are assumed to be

"", "A ", " A ", " A", "AA ", "A A", " AA", "AAA".

Wild characters are accepted by giving a specification `WILD=?`. For example, the line

```
??A: 1
```

in a classification list implies that any string having 'A' as its third element is mapped to 1. Similarly, partial matches are indicated by giving a specification `PARTIAL=*`. Then, for example, the line

```
*DOS: 2
```

in a classification list maps all cases where DOS appears as a substring to 2.

6.13 Same transformation for several variables

If the same numerical transformation should be performed for many variables simultaneously, the TRANSFORM operation is in most cases the right choice.

TRANSFORM <data> BY <function of X>

transforms all the active variables and observations (possibly limited by MASK, IND, CASES and SELECT specifications) by <function of X> where X refers to one of the active variables in turn.

TRANSFORM accepts the same functions as VAR. Also the **if** statement is allowed. Library functions and the control statement **for** are not accepted.

For example,

```
TRANSFORM DATA1 BY if(X=9)then(MISSING)else(X)
```

replaces all values 9 in active variables by missing values.

Although the same transformations can be made by VAR for each variable separately, use of TRANSFORM saves time and effort in systematic modifications of data values. For example,

```
TRANSFORM YEAR88 BY log(X)
```

replaces data values in data YEAR88 by their natural logarithms.

The transformed values can also be saved as other (possibly new) variables by using TRANSFORM in an extended form

```
TRANSFORM <data> BY <function of X> AS <letter>:<type>
```

where <letter> is a character to be placed in front of each transformed variable.

The extension :<type> where <type> is 1,2,4 or 8 is optional and gives the type of new numerical variables. Default is 4. For example,

```
TRANSFORM YEAR88 BY log(X) AS L / VARS=Sales,Costs
```

makes the logarithms of variables 'Sales' and 'Costs' as (new) variables 'LSales' and 'LCosts'.

6.14 Aggregation of observations

FILE AGGRE <Survo_data_file>,<new_aggregated_file>

combines records of a Survo data file according to values of a selected grouping variable. Only variables activated by 'A' will be processed. The records can be weighted by a selected variable activated by 'W'. Active records are indicated by CASES, IND and SELECT specifications when necessary.

Aggregation is controlled by an extra specification AGGRE having the form

```
AGGRE=<grouping variable>,<SUM or MEAN> .
```

The observations with the same value in the grouping variable are combined either by summing the values of variables (SUM) or by computing their mean values (MEAN). If missing values occur in any active variable, the observation in question is left out.

The aggregated observations are saved in a new Survo data file having the




TRANSFORM?
MARKOV?



AGGR?

same structure as the original data file has. If <new_aggregated_file> already exists, it is overwritten.

To save the frequencies (or sums of weights when a 'W' variable exists), a specification `FREQ=<name_of_variable>` is entered. This variable will be automatically created for the new aggregated file.

The maximum number of aggregates (new combined observations) `N(agg)` is limited approximately by  `N(agg)<150000/len`

where `len` is the field length of the grouping variable. The actual capacity depends on the internal memory available when the `FILE AGGRE` module is running.

The weighted means of three variables (`Tax`, `Income`, `Brthrate`) in the 12 provinces of Finland are computed by the `FILE AGGRE` operation in the next display. The essential parts of the resulting data file `PROVS` has finally been loaded to the edit field by a `FILE LOAD` operation.

Due to the `FREQ=Popul` specification, the sum of the weight variable 'Popul' will also be used for sums of weights. Thus 'Popul' in the new `PROVS` file gives the total population in the province.

```

16 1 SURVO 84C EDITOR Mon Feb 17 16:48:24 1992 D:\P2\DATA\ 100 100 0
45 *
46 *MASK=-W-----AAA-- / 'Popul' is the weight variable.
47 *FILE AGGRE FINLAND,PROVS
48 *AGGRE=Province,MEAN FREQ=Popul
49 *.....
50 *MASK=-AW-----AAA--
51 *FILE LOAD PROVS_
52 *DATA PROVS*,A,B,C
53 C Pro Popul Tax Income Brthrate
54 A VAA 430634 16.74 13579 14.367
55 * TUR 702007 15.68 15771 13.168
56 * OUL 414796 16.86 13119 16.654
57 * KYM 345102 15.84 15895 11.898
58 * MIK 209416 16.49 13450 11.398
59 * UUS 1119532 15.06 22952 13.606
60 * HAM 662387 15.41 16351 12.251
61 * AHV 22596 14.94 15686 11.861
62 * KAR 176680 16.49 12624 12.271
63 * LAP 195017 17.15 13634 14.614
64 * KES 242241 16.48 14283 13.697
65 B KUO 251686 16.98 13195 12.714
66 *

```

Since the aggregated file (`PROVS` in our example) inherits the field structure of the original file (`FINLAND`), the capacity of certain fields may be exceeded especially when the `SUM` option is used in the `AGGRE` specification. For example, the maximum value in the two-byte 'Income' field is 32766 and practically no sums are accepted.

In such cases, the normal procedure is to create a new version of the original data file with wider fields and to copy the data set to this extended environment with `FILE COPY`.

Sometimes the results of the aggregation should be moved back to the original

data file. This is done by using the FILE COPY operation with MODE=3.

For example, the difference of the Tax rate from its (weighted) provincial mean is computed for each commune as follows:

```

16 1 SURVO 84c EDITOR Mon Feb 17 17:28:43 1992 D:\P2\DATA\ 100 100 0
67 *
68 *FILE UPDATE PROVS
69 *FIELDS:
70 * 10 NA- 4 MTax Mean of the tax rate in 1979 (##.##)
71 *END
72 *
73 *VAR MTax=MISSING TO FINLAND
74 *
75 *FILE COPY PROVS,FINLAND / VARS=MTax MATCH=Province MODE=3
76 *
77 *VAR Taxdiff=Tax-MTax TO FINLAND
78 *.....
79 *CORR FINLAND,81_ / VARS=Taxdiff,Popul(W) CASES=Province:UUS
80 *FINLAND N=40 Weight=Popul
81 *Variable Mean Std.dev.
82 *Taxdiff -0.000000 0.542026
83 *

```

At first, the name of the Tax field in PROVS is changed to MTax (FILE UPDATE on lines 68-71) and the corresponding new variable is created by VAR on line 73. Then the provincial means MTax are copied to each commune in FINLAND by FILE COPY on line 75 and finally, the difference 'Taxdiff' is computed by VAR on line 77.

As a check, the weighted mean of 'Taxdiff' for the 40 communes in the province Uusimaa (UUS) is computed by a CORR operation on lines 79-82 and the result turns out to be 0 as expected.

7. Statistical operations

Tasks related to statistical analysis and statistical computing are performed in Survo by various statistical operations. Certainly there is no clear-cut difference between genuine statistical operations and others. Many activities described in other sections of this document are valuable in statistical computing as well. For example, the general calculating techniques (editorial computing and touch mode) and the transformation operations (`VAR`, `CLASSIFY`, `TRANSFORM`) are helpful in many tasks. Similarly, Survo graphics and matrix operations are essential tools in statistical applications. However, certain structural factors bind actual statistical procedures together in Survo.

The primary data values (samples etc.) are always represented as data lists or tables in the edit field or as data files on disk. Information from other sources, like text files, is easily converted into Survo data sets by the `FILE` operations described earlier.

In some statistical methods, however, the computations include several steps where the output of one operation should be studied carefully before continuing with the next operation. The common trend in the successive steps of the analysis is to compress the original data to sufficient statistics which often are represented as matrices with lower dimensions than the raw data.

For example, in standard multivariate analysis based on multinormality of original variables, the means, standard deviations, and correlations form sufficient statistics. Then matrix files consisting of those statistics can be used as a basis for computations as well. On the other hand, it is good to remember that there are often better computational techniques (based, for example, on orthogonalization of data) which do not rely on correlations at all.

In any case, a statistical system should provide tools for operating with various intermediate results in a matrix form as easily as with primary data. In Survo we have a large subsystem, the matrix interpreter, for these tasks. Many Survo operations related to linear models and multivariate analysis are making their matrix computations through the matrix interpreter. An advanced user may as well use the matrix interpreter directly by the `MAT` commands and chains.

In addition to standard data representation, there are other traits connecting various statistical methods in Survo. Although any statistical operation may have its own special requirements, it is important that the user can expect each operation to work according to the same style as the associate operations, at least to some extent. Thus, before describing the individual statistical operations currently available in Survo, we shall give an account of the general stylistic or structural similarities among various statistical operations.

7.1 General principles

7.1.1 Representation of statistical data

Most of the statistical operations use data sets only for their input. They give results as various summaries and output matrices and do not write any results back to the original data set. In these cases, any of the representations, data lists and tables in the edit field and data files on disk, discussed in the data management section is equally valid as an input.

However, in methods which create new derived variables, like residuals and predicted values of models, the use of data file representation is highly recommended. Also large data sets of, say, more than 10 variables and 300 observations should always be stored in files. It is easy to extract portions from them to the edit field by `FILE LOAD`, for example.

When using Survo for teaching of statistical data analysis, it is of course natural to keep data sets visible as data lists or tables in the edit field as long as possible.

Although it is not a primary concern of the user, it is useful to know that even those who program (in C) new statistical modules for Survo have no need to know which data representation is employed in a certain run. The programmer has access to both input and output data through programming tools which are independent of the data representation. This approach enables new structures of data to be included later without any need to change statistical program modules.

7.1.2 Size of statistical data

In any statistical system there are some theoretical and practical limits to the size of data sets. In Survo such limits are obviously related to the size of the edit field when data lists and data tables are employed. [The maximum size of the edit field is about 64000 elements \(600 lines with 100 columns or 1200 lines with 50 columns, for example\).](#) Thus a data table with 500 observations of 30 two-digit variables can be typed in the edit field. Similarly 1000 observations from 10 variables of max. 4 characters each could be used in that form. In practice, as remarked earlier, it is not wise to fill the whole edit field with pure data. The field is much more valuable space for commands, operations, work schemes, and explanatory text.

There are also good reasons to keep the size of the edit field limited since a large edit field occupies more valuable space in central memory which is needed for various program modules and for their work space during the Survo session. The default size of [100 lines and columns](#) is usually well enough in a typical Survo task, irrespective of the size of the data set.

Survo data files have no restrictions in relation to the edit field. The number

of variables can (in theory) be as high as about 5000, and the only limit for the number of observations is the capacity of the disk. In practice (on current PC's) it seems quite feasible to maintain Survo data bases with 500 variables and 10000 observations (i.e 5 million data values of 6-7 digits each). Such a file would require about $4*5=20$ megabytes on a hard disk.

Although one data file can be very large, it is seldom profitable to use extensive files as such. Working with a huge data base when only a small number of variables and observations are active may be time consuming and restrict the central memory available. It is better to make temporary work files (by FILE COPY) of selected variables and observations.

When scanning through the data in statistical operations, it is typical that only one observation at a time is processed in the central memory and pertinent information is collected for the current analysis. Thus, there is no need to reserve space for the whole data set and any number of cases can be processed without difficulty. The only critical factor is then the number of variables. There are, however, methods (like things related to time series) which require all the observations simultaneously in the central memory, but then the number of variables is never high.

Although hundreds and even thousands of variables may be defined in a single Survo data file, each statistical operation has more restrictions which are dependent on the memory size available for Survo at the current installation. One important general limit in the present implementation is that the maximum dimension of a square matrix in double precision is 90. Hence this is an absolute maximum number of active variables in applications related to linear models, multivariate analysis, and other methods working with matrices.

Survo allocates space dynamically for all work space needed in computations and checks whether there is enough memory available for the current application. If not, an error message like "Not enough memory!" is temporarily displayed and the present operation interrupted. Since the memory space available for Survo depends on many external factors in the installation, no exact limits can be given for any operation. For example, if there are resident programs in the memory which are loaded before Survo has been entered, the memory space may be smaller than without such a background.

7.1.3 Selection of variables and observations

With a few exceptions, all statistical operations working directly on primary data can use VARS and MASK specifications for selecting (activating) variables as well as CASES, IND and SELECT specifications for restricting the range of observations. These options have been described earlier in the section on data management. More examples of their application will be seen in examples related to various statistical operations.

Normally, any variable of the data set can be treated as a numeric variable in statistical operations. Even string fields (if they contain numeric values as text) can be processed as numeric variables.

7.1.4 Computing accuracy

Unless otherwise stated, all computations with statistical variables are carried out in double precision. Thus about 15 significant digits are used in all arithmetics and basic functions. Due to rounding errors, the final results cannot usually be so accurate.

In floating point calculations, the numeric range of variables is not very significant. However, it is wise to keep the numeric values roughly in the range from 0.001 to 1000, although values from about $1e-300$ to $1e300$ are permitted (as numeric type 8). Decent values around 1 improve computing accuracy in some applications and they are also nice in printouts and in scale notations of graphs, for instance.

7.1.5 Treating of missing values

The missing values are denoted by '-' in data lists and tables and by leaving the cell of the variable empty when feeding data in data files by **FILE SHOW** or **FILE EDIT**.

Each statistical operation is responsible for recognizing all missing values appearing in active variables and observations. How they proceed when missing values are encountered, is another problem and considered separately in each operation.

There are two general alternatives when handling missing values. The first alternative is to exclude all observations where at least one variable has a missing value. Another alternative is to replace missing values by plausible substitutes which are computed on the basis of other variables and observations.

The user can easily devise strategies for missing value replacement by using, for example, regression methods for predicting missing values and replacing them conditionally by the **VAR** operation.

Certain optional Survo modules for missing value replacement have been programmed by Leena Sadeniemi. These modules (**PAD**, **CORRM**, **CORRML**) are reported in **SURVO 84C Contributions No.5 (1992)**.



CORRMV?

7.1.6* Checking scale types

As explained earlier, the third mask column of a data file (edited by the **FILE ACTIVATE** operation or by the **FILE ACT** key) is reserved for indicating scale types of the variables. Originally, this column contains spaces, and all scale checks when such a data file is processed are omitted. Similarly, for data lists and tables in the edit field, scale checks are never performed.

It is, however, strongly advised to use the scale control option for statistical data, especially in data sets which contain plenty of variables and variables on different measuring levels. Then the system is able to detect unsuitable choices of variables, and thus, for example, it prevents computing of means for variables on a nominal scale. Even if the user clearly knows the restrictions imposed by different scale types, he/she may easily forget the types of vari-

ables in large applications.

The following codes are used in the mask column 3 when a data file is activated (by `FILE ACTIVATE` etc.).

| | | |
|---|-----------|---|
| - | no scale | (variable excluded from statistical operations) |
| D | Dichotomy | (two distinct numeric values) |
| N | Nominal | |
| O | Ordinal | (discrete) |
| o | Ordinal | (continuous) |
| S | Score | (discrete) |
| s | Score | (continuous) |
| I | Interval | (discrete) |
| i | Interval | (continuous) |
| R | Ratio | (discrete) |
| r | Ratio | (continuous) |
| F | Frequency | |

If the scale type is omitted (code is blank), no scale control will be performed and data values are accepted in all applications. In statistical operations, these scale type notations are usually interpreted as follows:

| <u>type of scale/variable</u> | <u>allowed scale notations</u> |
|-------------------------------|--------------------------------|
| nominal scale | N |
| ordinal scale | DOoSsIiRrF |
| score scale | DSsIiRrF |
| interval scale | DIiRrF |
| ratio scale | RrF |
| discrete variable | DNOSIRF |
| continuous variable | osir |

The statistical operations observe scale types when desired. The strictness of the scale type control depends on the system parameter `scale_check` in the system file `SURVO.APU` as follows:

- 0 no control
- 1 warning when a variable with improper scale is selected
- 2 warning and interrupt of operation

The default level of `scale_check` is 2 and normally there is no reason to change this selection.

7.1.7 Printout of results

The results of statistical operations appear in four different ways:

1. Temporary displays on the screen
2. Results in the edit field
3. Output files
4. Matrix files

Each operation is responsible for giving information about its progress by displaying text, numbers etc. which tell about the ongoing computing process. This information is displayed in a temporary window below the current command line.

On the basis of this display, the user should always have a feeling that something is really happening. To halt the work temporarily, one can press the space bar. Then a prompt '*Continue the operation Y/N?*' is displayed. After studying the current situation, the user makes the decision. Thus by pressing **[N]**, the current job is terminated and the system returns to the normal editing mode. By pressing **[Y]**, the process will continue normally.

The information in the temporary window is not saved at all. Parts of it may, however, appear in permanent printouts.

When scanning the observations, the statistical operations usually list the indices of active observations on the screen. To interrupt this listing (which in some applications may slow down the process) the user may press **[.]** once. Then work will be continued without listing the indices until **[.]** is pressed again. Not necessarily all statistical operations follow this convention.

The actual results in readable form are always printed in the edit field and/or in an output file. Most of the statistical operations have an optional parameter which gives the first line for the results in the edit field. For example, the CORR operation in the next display

```

19  1 SURVO 84C EDITOR Sun Feb 23 10:35:59 1992      D:\P2\STAT\ 150 100 0
 1 *
 2 *VARS=Tax,Brthrate,Income
 3 *CORR FINLAND,END+2_
 4 *
 5 *
 6 *
 7 *
 8 *
 9 *This is the last non-empty line before activation of CORR above.
10 *

```

gives its results from line $END+2=9+2=11$ onwards as follows:

```

19  1 SURVO 84C EDITOR Sun Feb 23 10:36:03 1992      D:\P2\STAT\ 150 100 0
 1 *
 2 *VARS=Tax,Brthrate,Income
 3 *CORR FINLAND,END+2_
 4 *
 5 *
 6 *
 7 *
 8 *
 9 *This is the last non-empty line before activation of CORR above.
10 *
11 *Means, std.devs and correlations of FINLAND  N=464
12 *Variable  Mean      Std.dev.
13 *Tax       16.22511    0.975172
14 *Brthrate  12.02225     3.459748
15 *Income    12737.80     3102.735
16 *Correlations:
17 *          Tax      Brthrat  Income
18 * Tax       1.0000   0.1016  -0.4933
19 * Brthrate   0.1016   1.0000   0.2549
20 * Income    -0.4933   0.2549   1.0000
21 *

```

In this case, we could use an absolute line number or a line label as well.

New results always overwrite existing text on the result lines without any warning. This situation is avoided, for example, by using line labels relative to **END**.

If the results overstep the limits of the current edit field, the excessive parts are simply omitted. Each statistical module is supposed to write its results within the typical visible line length in the edit field (currently 72 characters). If results in a table or matrix form appear, they are split into suitable parts.

If the line number or label parameter in a statistical operation is omitted, no results will generally appear in the current edit field.

A permanent and safe place for Survo results is an output file which is a standard text (ASCII) file. The results (without any limits imposed by the edit field) are always appended to the end of the output file. The default name (and path) of the output file is given by the system parameter `eout` in the file `SURVO.APU`. The output file can be easily changed during the Survo session by the

`OUTPUT <name of output file>`

command. By activating merely `OUTPUT` without any parameter, the name of the current output file is displayed. The output file is suppressed by

`OUTPUT - .`

As a special case, `OUTPUT PRN` selects the printer as the output file. Then all results will be printed directly on the printer. (This feature is invalid in typical PostScript printers because they cannot interpret direct stream of output characters.)

Since text on paper is hard to edit, it is usually better to use a true file for the output. Then activation of `SHOW` (without any parameter) always displays the contents of the current output file and selected lines may be loaded to the edit field for reporting.

7.1.8* Results in matrix files

The output file which stores results in text form is not always sufficient for numerical results obtained in statistical analyses. To preserve maximum numerical accuracy and to provide immediate access to the matrix interpreter (`MAT` and `MATRUN` operations), the statistical operations save their output matrices in matrix files, too.

Such output matrices are always saved on the current data disk using standard names with extension `.M` (instead of `.MAT` which is the extension for `MAT` files). They are, however, normal matrix files which can be processed by `MAT` operations and by many statistical (multivariate) operations as well.

Default names used for certain output matrices are given below and the user can change them easily by the `MS-DOS` command `REN`. For example,

`>REN A:* .M *.MAT` (even as a Survo command)

changes on disk `A`: all matrix result files (with extension `.M`) to matrix files (with extension `.MAT`).



MULTI?



Typical matrix files for statistical results:

| | |
|-----------------|---|
| CORR.M | correlation matrix (by operation CORR, for example) |
| MSN.M | matrix of 3 columns (CORR): column 1: means of variables column 2: standard deviations of variables column 3: number of observations |
| REG.M | column vector of regression coefficients (LINREG) |
| PCOMP.M | Principal components loadings (MATRUN PCOMP) |
| PCOEFF.M | Coefficients for principal component scores |
| PFACT.M | Factor matrix (MATRUN PFACT) |
| FACT.M | Factor matrix (FACTA) |
| AFACT.M | Rotated factor matrix |
| TFACT.M | Rotation matrix |
| FCOEFF.M | Factor score coefficients (MATRUN FCOEFF) |
| LCAN.M | Canonical correlations (CANON) |
| XCAN.M | Standardized loadings of canonical variates (set 1) |
| YCAN.M | Standardized loadings of canonical variates (set 2) |
| XTAB.M | Design matrix (TABFIT) |
| PCOV.M | Covariance estimates of parameters (TABFIT) |

An up-to-date version of this list is obtained during a Survo session by the inquiry MATRES?.



MATRES?

7.1.9 Accuracy and extent of results

As said earlier, all computations in Survo are carried out in double precision. When presenting numerical results, it is not sensible, however, to use 15 significant digits in normal printouts. To meet various requirements in output accuracy, Survo has a special system parameter **accuracy** (in SURVO.APU) which gives the typical number of significant digits in output. A regular value is **accuracy=7** and the permitted range from 4 to 16.

The statistical operations observe this system parameter and adjust their printouts accordingly. Of course, when selecting a correct number of decimals, other factors (like the nature of the statistics) must be considered, too. For example, the correlations are usually printed with **accuracy=3** decimals.

The extent of results given by statistical operations is selected by the **RESULTS=<output level>** specification where <output level> is an integer from 0 to 100. Level 0 implies minimal output and 100 maximal. The default value of the output level is set in the system file SURVO.APU as **results=70**.

The effect of the **RESULTS** specification is determined in each operation separately (see description of the operation in question). In some operations, a more detailed form of **RESULTS** may be available (e.g. permitting a list of statistics to be printed).

7.2 Basic statistics

7.2.1 STAT operation

STAT <data>,L

computes certain basic statistics and summaries for active variables in <data> and forms a frequency distribution for each of them according to an automatic classification.

L (optional) is the first line for the results. The **IND**, **CASES** and **SELECT** specifications limit the observations and the missing values are recorded for each variable separately. **VARS** and **MASK** specifications are used for selecting of variables.

The main application of **STAT** is to give an overall summary of the statistical behavior of each active variable. On the basis of the **STAT** results, decisions about more refined analysis can be done.

The basic statistics given by **STAT** include, for example, min, max, mean, stddev, skewness, kurtosis, first order autocorrelation, median, quartiles and entropy.

The selection of statistics depends, however, on the scale type of the variable. Certain statistics (like autocorrelation) are omitted if no information seems to be gained ($\text{autocorrelation} < 2/\sqrt{N}$).

More fractiles are computed (from the automatically grouped data) by the specification **FRACTILES=p1, p2, ...** where $0 < p < 1$, for $p=p1, p2, \dots$. Order statistics are not given if (due to an outlier etc.) most of the data is accumulated into a single class.

Special means of the form $((X1^k + X2^k + \dots + XN^k) / N)^{(1/k)}$ will be computed for positive variables X and for various values of the exponent $k=k1, k2, \dots$ by entering a specification **MEANS=k1, k2, ...**. Important special cases are

| | |
|-----------------|------------------|
| Quadratic mean | for k=2 or k=Q |
| Arithmetic mean | for k=1 or k=A |
| Geometric mean | for k=0 or k=G |
| Harmonic mean | for k=-1 or k=H. |

For example, all these means plus a power mean with the exponent $k=2.5$ are obtained by **MEANS=A, G, H, Q, 2.5**.

Maximum number of classes is given by the **CLASSMAX=<#_of_classes>** specification (default **CLASSMAX=30**). When making frequency distributions, **STAT** tries first to keep a record of each variable for each distinct value up to the limit **CLASSMAX**. If it is not possible, a classification with equal class widths is entered on the basis of values found so far. This classification will be further redefined, by doubling the class widths if values outside the current range are found. For grouped distributions with user-defined classes, see **HISTO** and **TAB**.

Below is a typical application of STAT for one selected variable. The results of STAT on line 25 appear on lines from 26 to 56.

```

16 1 SURVO 84C EDITOR Mon Mar 02 09:51:32 1992 D:\P2\STAT\ 150 100 0
24 *
25 *STAT FINLAND,26 / VARS=Brthrate
26 *Basic statistics: FINLAND N=464
27 *Variable: Brthrate ~1000*Births/Popul
28 *min=2.169197 in obs.#165 (Kumlinge)
29 *max=25.82496 in obs.#88 (Jämsä)
30 *mean=12.02225 stddev=3.459748 skewness=0.186248 kurtosis=0.81534
31 *lower_Q=9.884615 median=12.05645 upper_Q=14.15476
32 *up.limit f % *=2 obs. class width=1
33 * 3 3 0.6 *
34 * 4 3 0.6 *
35 * 5 4 0.9 **
36 * 6 9 1.9 ****
37 * 7 15 3.2 *****
38 * 8 23 5.0 *****
39 * 9 25 5.4 *****
40 * 10 39 8.4 *****
41 * 11 46 9.9 *****
42 * 12 62 13.4 *****
43 * 13 62 13.4 *****
44 * 14 51 11.0 *****
45 * 15 42 9.1 *****
46 * 16 30 6.5 *****
47 * 17 19 4.1 *****
48 * 18 14 3.0 *****
49 * 19 6 1.3 ***
50 * 20 4 0.9 **
51 * 21 1 0.2 :
52 * 22 2 0.4 *
53 * 23 1 0.2 :
54 * 24 2 0.4 *
55 * 25 0 0.0
56 * 26 1 0.2 :
57 *

```

As another example, we compute the frequencies of various characters in the preceding section (Data base management) of this document.

At first, all the text consisting of 23 edit files has been copied to a text file TEXT by using the PRINT operation. PRINT produces this text file as a by-product by entering a control line

```
- ascii_text TEXT
```

in the PRINT list (see PRINT?). The file has 130849 bytes. This is all the text plus possible leading spaces on each line and line feeds. The page headers and control characters and words needed for printing are not included.

The next exhibit gives the remaining steps of the task. To save space, we have rearranged the output of STAT in two columns. Since the code file SORTCODE.BIN is in use as a filter, the upper and lower case letters are equated. Spaces and line feeds are excluded.

```

20 1 SURVO 84C EDITOR Sat Mar 21 11:29:15 1992 D:\P2\STAT\ 100 100 0
1 *
2 *FILE CREATE CHARDATA,1,1
3 * Section Data base management from this book
4 *FIELDS:
5 *1 SA_N 1 Char
6 *END
7 *
8 *FILE SAVE TEXT,CHARDATA
9 *FORMAT=CHAR,Char FILTER=SORTCODE.BIN
10 *.....
11 *STAT CHARDATA,END+2
12 *CLASSMAX=100
13 *
14 *Basic statistics: CHARDATA N=96534
15 *Variable: Char
16 *Nominal scale
17 *entropy=4.840887 (80.4%)
18 *Char f % Char f % *=256 obs.
19 *| 10 0.0 : A 7413 7.7 *****
20 *" 38 0.0 : B 1391 1.4 *****
21 *# 305 0.3 * C 2725 2.8 *****
22 *% 20 0.0 : D 3477 3.6 *****
23 *& 6 0.0 : E 10134 10.5 *****
24 */ 110 0.1 : F 2694 2.8 *****
25 *( 424 0.4 * G 993 1.0 ***
26 *) 424 0.4 * H 2672 2.8 *****
27 ** 1123 1.2 **** I 6498 6.7 *****
28 *+ 32 0.0 : J 60 0.1 :
29 *, 975 1.0 *** K 332 0.3 *
30 *- 610 0.6 ** L 3788 3.9 *****
31 *. 2227 2.3 ***** M 1972 2.0 *****
32 */ 56 0.1 : N 5304 5.5 *****
33 *0 1353 1.4 ***** O 5369 5.6 *****
34 *1 1845 1.9 ***** P 1766 1.8 *****
35 *2 1095 1.1 **** Q 47 0.0 :
36 *3 716 0.7 ** R 4800 5.0 *****
37 *4 802 0.8 *** S 5623 5.8 *****
38 *5 684 0.7 ** T 7106 7.4 *****
39 *6 644 0.7 ** U 1932 2.0 *****
40 *7 598 0.6 ** V 1418 1.5 *****
41 *8 555 0.6 ** W 826 0.9 ***
42 *9 602 0.6 ** X 455 0.5 *
43 *: 318 0.3 * Y 1069 1.1 ****
44 *; 1 0.0 : Z 37 0.0 :
45 *< 128 0.1 : ^ 1 0.0 :
46 *= 299 0.3 * _ 122 0.1 :
47 *> 127 0.1 : ` 98 0.1 :
48 *? 15 0.0 : { 13 0.0 :
49 *[ 26 0.0 : } 13 0.0 :
50 *\ 186 0.2 : ~ 6 0.0 :
51 *] 26 0.0 :
52 *

```

The following display illustrates the use of the `FRACTILES` specification in connection with the `STAT` operation. We generate a sample of 1000 observations from a standard normal distribution and calculate some order statistics from it. `CLASSMAX=1000` guarantees that in this case no grouping of observations is needed and then the order statistics can be computed directly from the ordered sample.

For a comparison, some theoretical values have been computed afterwards (see the shaded area).

```

15 1 SURVO 84C EDITOR Sat Feb 21 16:29:52 1987 D:\P2\DATA\ 100 100 0
1 *
2 *FILE CREATE NORMAL,4,1
3 *FIELDS:
4 *1 N 4 X
5 *END
6 *
7 *FILE INIT NORMAL,1000
8 *
9 *VAR X=probit(rnd(0)) TO NORMAL
10 *
11 *STAT NORMAL,12 / CLASSMAX=1000 FRACTILES=0.01,0.05,0.5,0.95,0.99
12 *Basic statistics: NORMAL N=1000
13 *Variable: X ~probit(rnd(0))
14 *min=-2.988369 in obs.#138
15 *max=3.147558 in obs.#425
16 *mean=0.016743 stddev=1.023159 skewness=0.029168 kurtosis=-0.127374
17 *lower_Q=-0.698415 median=0.042735 upper_Q=0.720972
18 *fractile(0.01)=-2.27472 N.G(0,1,0.01)=-2.3263478740408
19 *fractile(0.05)=-1.635637 N.G(0,1,0.05)=-1.6448536269515
20 *fractile(0.5)=0.042735 N.G(0,1,0.75)=0.67448975019608
21 *fractile(0.95)=1.706025
22 *fractile(0.99)=2.325277
23 *X f % Values not equidistant!
24 *-2.988369 1 0.1 *
25 *-2.827283 1 0.1 *
26 *-2.736034 1 0.1 *
27 *etc.

```

Since `rnd(0)` is used in this experiment, the results will vary from one realization to another.

7.2.2 CORR operation

`CORR <data>,L`

computes the means, standard deviations and correlations of all active variables and observations in `<data>`. Specifications like `VARS`, `MASK`, `CASES`, `IND` and `SELECT` are available for selecting variables and observations.

Observations with missing values in at least one of the active variables are omitted. By an optional `CORRM` operation by L.Sadeniemi (reported in SURVO 84C Contributions No.5, 1992) correlations using all non-missing pairs of data values are computed. Similarly, the optional `CORRML` operation by the same author provides the maximum likelihood method for estimating of correlations in the case of missing values.

The results of `CORR` will appear from line `L` onwards. If `L` is missing, results are not displayed in the edit field. If an output file (see `OUTPUT?`) is selected, the results are written to the end of that file. Printing of results may be completely left out by entering `RESULTS=0`. In any case, the results (in full precision) are saved in matrix files

`CORR.M` correlations

`MSN.M` col.1=means col.2=stddevs col.3=number of observations
on the current data disk. Their names can be changed by the MS-DOS command `>REN`, for example.

Many operations in multivariate analysis and linear models are using matrix

files CORR.M and MSN.M as a basis for their work.

All the computations are carried out in double precision. When computing the moments, the values of the first observation are used as temporary means. If a variable is a constant, no error message is given, but all the correlations (including the diagonal element, which normally is 1) are set to 0 for that variable.

In the next display, means, standard deviations and correlations have been computed for a subset of communes and variables in the FINLAND file.

```

15 1 SURVO 84C EDITOR Mon Apr 06 16:04:05 1992 D:\P2\STAT\ 100 100 0
1 *
2 *CASES=Province:UUS,TUR,HÅM,KYM IND=Popul,10000,500000
3 *VARS=Tax,Income,Brthrate,Agri,Industry,Service
4 *CORR FINLAND,5
5 *Means, std.devs and correlations of FINLAND N=54
6 *Variable Mean Std.dev.
7 *Tax 15.44352 0.667086
8 *Income 17263.50 2700.938
9 *Brthrate 13.51775 2.151205
10 *Agri 0.444444 0.839287
11 *Industry 4.092593 1.032829
12 *Service 4.148148 1.172120
13 *Correlations:
14 *
15 * Tax Income Brthrat Agri Industr Service
16 * Tax 1.0000 -0.1974 -0.1245 -0.0975 0.1337 -0.0374
17 * Income -0.1974 1.0000 0.4093 -0.5588 -0.1123 0.5977
18 * Brthrate -0.1245 0.4093 1.0000 -0.5062 0.2274 0.1550
19 * Agri -0.0975 -0.5588 -0.5062 1.0000 -0.4619 -0.2792
20 * Industry 0.1337 -0.1123 0.2274 -0.4619 1.0000 -0.6506
21 * Service -0.0374 0.5977 0.1550 -0.2792 -0.6506 1.0000

```

In addition to results in the edit field, the correlations as well as means and standard deviations have been saved in matrix files (CORR.M and MSN.M) in full precision. For example, a 3*3 submatrix of CORR.M may be loaded more accurately by MAT LOAD as follows:

```

47 1 SURVO 84C EDITOR Mon Apr 06 16:12:19 1992 D:\P2\STAT\ 100 100 0
21 *
22 *MAT LOAD CORR.M(1:3,1:3),12.123456789012345,23
23 *MATRIX CORR.M
24 *R(FINLAND)
25 *///
26 *Tax 1.0000000000000000 -0.197427872311857 -0.124457062308030
27 *Income -0.197427872311857 1.0000000000000000 0.409269513809020
28 *Brthrate -0.124457062308030 0.409269513809020 1.0000000000000000
29 *

```

As another example, we generate 200 observations from a first-order autoregressive process and compute autocorrelations for lags 1,2,3,4,5.

A vector diagram is a suitable way to visualize the correlation matrix. See page 5 in <http://www.survo.fi/tmp/VectorDiagrams.pdf>


```

14 1 SURVO 84C EDITOR Sun Feb 22 11:06:33 1987 D:\STAT\ 120 80 0
1 *
2 *FILE CREATE AUTO1,24,6,64,7,200
3 * 200 observations from X(t)=0.8*X(t-1)+eps
4 *FIELDS:
5 *1 N 4 X
6 *END
7 *
8 *VAR X=if(ORDER=1)then(eps)else(0.8*X[-1]+eps) TO AUTO1
9 * eps=probit(rnd(1))
10 *.....
11 *VAR X1,X2,X3,X4,X5 TO AUTO1
12 *X1=X[-1] X2=X[-2] X3=X[-3] X4=X[-4] X5=X[-5]
13 *.....
14 *CORR AUTO1,15
15 *Means, std.devs and correlations of AUTO1 N=200
16 *# of missing observations =5
17 *Variable Mean Std.dev.
18 *X -0.106722 1.756378
19 *X1 -0.102504 1.753710
20 *X2 -0.102143 1.753462
21 *X3 -0.104000 1.756002
22 *X4 -0.106121 1.758365
23 *X5 -0.110815 1.764946
24 *Correlations:
25 *
26 * X X X1 X2 X3 X4 X5
27 * X 1.0000 0.8092 0.6211 0.4683 0.3556 0.2878
28 * X1 0.8092 1.0000 0.8088 0.6185 0.4665 0.3531
29 * X2 0.6211 0.8088 1.0000 0.8082 0.6184 0.4664
30 * X3 0.4683 0.6185 0.8082 1.0000 0.8087 0.6203
31 * X4 0.3556 0.4665 0.6184 0.8087 1.0000 0.8093
32 * X5 0.2878 0.3531 0.4664 0.6203 0.8093 1.0000

```

7.2.3 TAB operation

The main tool for making multiway tables of frequencies, sums, means, and standard deviations is the TAB operation. A more general alternative is the MTAB operation by M.Korhonen (see SURVO 84C Contributions No.5, 1992).

TAB <data>,L

makes cross-tabulations of <data> according to selected variables and saves the results in the output file (see OUTPUT) and from line L onwards in the current edit field if L is given.

Also means, sums and standard deviations of a cell variable can be computed (see CELL specification below).

The IND, CASES and SELECT specifications are used for specifying the set of active observations.

The variables controlling classification can be numeric or string variables and they are to be listed by the specification

VARIABLES=<column var>,<row var 1>,<row var 2>,...,<row var n>

or

VARIABLES=<column var>:<row var 1>,<row var 2>,...,<row var n>

The first alternative of the VARIABLES specification implies computing of one n-dimensional table.

The second one (with a colon ':' after <column var>) implies computing of n two-dimensional tables with a common <column var>.

A still more general alternative is

VARIABLES=<cvar 1>:<cvar 2>:...:<cvar m>:<rvar 1>,<rvar 2>,...

which allows m nested column variables for each row variable.

The classification of a numeric variable is given in the form

<name of var>=L1,U1,U2,U3,...

where L1=lower limit of the 1st class

U1=upper limit of the 1st class

U2=upper limit of the 2nd class etc.

For example, Age=0,6,12,21 specifies age groups 0-6,7-12 and 13-21. Any class can be given a name in parentheses after the upper limit. For example, Color=1,1(white),2(red),3(blue).

If the class widths are equal, classification may also be given in the form <name of var>=<up.limit_of_1st_class>(<class_width>)<last_class>.

For example, Age=10(5)25 is same as Age=5,10,15,20,25. In this abbreviated notation, no class names can be given.

The classification of a string variable is described by examples:

1. Assume that 'Weekday' is a string variable with values 'Su', 'Mo' etc.
A classification for 'Weekday' could be
Weekday=/Mo/Tu/We/Th/Fr(Workdays) , /Sa/Su(Weekend)
thus forming 2 classes, Workdays=(Mo, Tu, We, Th, Fr) and
Weekend=(Sa, Su).
2. Assume that 'Letter' is a string variable with values 'A', 'B', 'C' etc. To
classify letters to vowels and consonants, the following specification is
valid:
Letter=/A/E/I/O/U(vowel) , /-(consonant) .
Above, /- denotes all remaining alternatives.

The printout of a (frequency) table may include column and row sums, too. In addition to frequencies, various tables of percentages can be obtained. These options are selected by a RESULTS specification of the form

RESULTS=CSUMS,RSUMS,C%,R%,T%

with the following consequences:

| | |
|-------|---|
| CSUMS | sums of columns |
| RSUMS | sums of rows |
| C% | table of percentages in columns |
| R% | table of percentages in rows |
| T% | table of percentages from the grand total |

Any of the keywords in RESULTS may be omitted.

For each two-dimensional table, the standard χ^2 statistic is computed and printed.

To compute the mean and the standard deviation for a selected variable, a CELL specification may be included. It has the form

CELL=<cell variable> , <format for the results, ###.# for example> .

When a CELL variable is used, the RESULTS specification becomes ineffective.

Sums of the cell variable are computed (instead of means and standard deviations) by giving the CELL specification in the form

CELL=<cell variable> , <format> , <FSUM or SUM>

For FSUM, both frequencies and sums are printed. For SUM, sums only are printed.

Tables generated by the TAB operation can be edited by TABS, TABM, TABI, TABD, TABJ operations and analyzed by a TABFIT operation. These functions are described in the section 7.6.

Application of the TAB operation is shown by some examples which are related to each other. At first, a 3-dimensional table is computed from the

FINLAND data as follows:

```

14 1 SURVO 84C EDITOR Sat Apr 11 12:42:34 1992 D:\P2\STAT\ 100 100 0
1 *
2 *TAB FINLAND,9_
3 *VARIABLES=Province,Popul,Brthrate
4 *Province=/UUS/TUR/AHV/KYM(South),/HÄM/MIK/KUO/KES/VAA/KAR(Middle),&
5 * /-(North) (& = continued on next line)
6 *Popul=0,5000,20000,500000
7 *Brthrate=0,10,15,30
8 *
9 *TABLE FINLAND1 A,B,F N=464
10 A Province South Middle North
11 *Popul Brthrate *****
12 * 5000 10 50 42 4
13 * 15 48 42 17
14 * 30 6 11 16
15 * 20000 10 8 17 0
16 * 15 42 73 16
17 * 30 9 7 16
18 *500000 10 0 0 0
19 * 15 12 12 1
20 B 30 5 6 4
21 *

```

Due to the specification

VARIABLES=Province,Popul,Brthrate

on line 3, the first variable 'Province' becomes the only column classifier while 'Popul' and 'Brthrate' appear nested on rows.

Simply by editing the VARIABLES specification into the form

VARIABLES=Province:Popul,Brthrate

(one comma replaced by a colon), we get two 2-dimensional tables:

```

14 1 SURVO 84C EDITOR Sat Apr 11 12:49:23 1992 D:\P2\STAT\ 100 100 0
1 *
2 *TAB FINLAND,9_
3 *VARIABLES=Province:Popul,Brthrate
4 *Province=/UUS/TUR/AHV/KYM(South),/HÄM/MIK/KUO/KES/VAA/KAR(Middle),&
5 * /-(North)
6 *Popul=0,5000,20000,500000
7 *Brthrate=0,10,15,30
8 *
9 *TABLE FINLAND1 A,B,F N=464
10 A Province South Middle North
11 *Popul *****
12 * 5000 104 95 37
13 * 20000 59 97 32
14 B500000 17 18 5
15 *Chi_square=7.915 df=4 P=0.0948
16 *
17 *TABLE FINLAND2 C,D,F N=464
18 C Province South Middle North
19 *Brthrate *****
20 * 10 58 59 4
21 * 15 102 127 34
22 D 30 20 24 36
23 *Chi_square=67.47 df=4 P=0.0000

```

The 'Province' variable is now the common column variable in both tables. Since the tables are 2-dimensional, the χ^2 statistic is computed, too.

If we add still one more colon in the VARIABLES specification, the result once again assumes a 3-dimensional form but now with two nested column variables:

```

14 1 SURVO 84C EDITOR Sat Apr 11 12:52:31 1992 D:\P2\STAT\ 100 100 0
1 *
2 *TAB FINLAND,9_
3 *VARIABLES=Province:Popul:Brthrate
4 *Province=/UUS/TUR/AHV/KYM(South),/HÄM/MIK/KUO/KES/VAA/KAR(Middle),&
5 * /-(North)
6 *Popul=0,5000,20000,500000
7 *Brthrate=0,10,15,30
8 *
9 *TABLE FINLAND1 A,B,F N=464
10 A Province South Middle North
11 * Popul 5000 20000 500000 5000 20000 500000 5000 2000
12 *Brthrate *****
13 * 10 50 8 0 42 17 0 4
14 * 15 48 42 12 42 73 12 17 1
15 B 30 6 9 5 11 7 6 16 1
16 *

```

Observe, however, that this setup overrides the current display width.

To obtain marginal columns and rows as well as percentages in addition to frequencies, a RESULTS specification should be entered. In the next display we have used RESULTS=CSUMS,C% and obtained the following results:

```

14 1 SURVO 84C EDITOR Sat Apr 11 13:54:47 1992 D:\P2\STAT\ 100 100 0
1 *
2 *TAB FINLAND,9_
3 *VARIABLES=Province,Popul,Brthrate RESULTS=CSUMS,C%
4 *Province=/UUS/TUR/AHV/KYM(South),/HÄM/MIK/KUO/KES/VAA/KAR(Middle),&
5 * /-(North)
6 *Popul=0,5000,20000,500000
7 *Brthrate=0,10,15,30
8 *
9 *TABLE FINLAND1 A,B,FC N=464
10 A Province South Middle North
11 *Popul Brthrate *****
12 * 5000 10 50 42 4
13 * 15 48 42 17
14 * 30 6 11 16
15 * 20000 10 8 17 0
16 * 15 42 73 16
17 * 30 9 7 16
18 *500000 10 0 0 0
19 * 15 12 12 1
20 B 30 5 6 4
21 * sum 180 210 74
22 *
23 *TABLE FINLAND2 C,D,%C N=464
24 C Province South Middle North
25 *Popul Brthrate *****
26 * 5000 10 27.8 20.0 5.4
27 * 15 26.7 20.0 23.0
28 * 30 3.3 5.2 21.6
29 * 20000 10 4.4 8.1 0.0
30 * 15 23.3 34.8 21.6
31 * 30 5.0 3.3 21.6
32 *500000 10 0.0 0.0 0.0
33 * 15 6.7 5.7 1.4
34 D 30 2.8 2.9 5.4
35 * sum 100.0 100.0 100.0
36 *

```

By giving a CELL specification, conditional means and standard deviations of a selected variable can be computed in classes determined by the variables in the VARIABLES specification. The next display is an example of this:

```

14 1 SURVO 84C EDITOR Sat Apr 11 13:56:49 1992 D:\P2\STAT\ 100 100 0
1 *
2 *TAB FINLAND,9
3 *VARIABLES=Province,Popul CELL=Brthrate
4 *Province=/UUS/TUR/AHV/KYM(South),/HÄM/MIK/KUO/KES/VAA/KAR(Middle),&
5 * /-(North)
6 *Popul=0,5000,20000,500000
7 *Brthrate=0,10,15,30
8 *
9 *TABLE FINLAND1 A,B,FMS N=464 Mean and SD of Brthrate
10 A Province South Middle North
11 *Popul *****
12 * 5000 104 95 37
13 * Mean 9.880194 10.80283 14.22038
14 * SD 3.191409 3.791524 3.360766
15 * 20000 59 97 32
16 * Mean 12.65011 12.13710 15.33251
17 * SD 2.275356 2.509748 2.818451
18 *500000 17 18 5
19 * Mean 14.25483 14.38854 16.54821
20 B SD 1.952341 1.687932 1.874006
21 *

```

As another type of application we consider a simulation experiment where 10000 values of 3 normal random deviates X, Y, Z are generated in a data file BINORM. The X and Y variates are independent, but $Z = Y + 0.05 * X$ will be slightly dependent on X .

Dependencies X versus Y and X versus Z are examined by computing corresponding contingency tables:

```

17 1 SURVO 84C EDITOR Tue Feb 24 11:01:32 1987 D:\P2\STAT\ 100 100 0
1 *
2 *FILE CREATE BINORM,12,3,64,7,10000
3 *FIELDS:
4 *1 N 4 X
5 *2 N 4 Y
6 *3 N 4 Z
7 *END
8 *
9 *VAR X,Y,Z TO BINORM
10 *X=probit(rnd(1))
11 *Y=probit(rnd(1))
12 *Z=Y+0.05*X
13 *.....
14 *TAB BINORM,END+2
15 *
16 *VARIABLES=X:Y,Z RESULTS=RSUMS,CSUMS
17 *X=-5,-2,-1,0,1,2,5 Y=-5,-2,-1,0,1,2,5 Z=-5,-2,-1,0,1,2,5
18 *
19 *TABLE BINORM1 A,B,FCR N=10000
20 A X -2 -1 0 1 2 5 sum
21 *Y *
22 *-2 7 26 76 79 27 6 221
23 *-1 26 182 443 444 185 37 1317
24 * 0 72 451 1110 1195 486 74 3388
25 * 1 87 476 1154 1186 480 104 3487
26 * 2 38 191 452 458 188 43 1370
27 B 5 6 37 71 65 29 9 217
28 * sum 236 1363 3306 3427 1395 273 10000
29 *Chi_square=17.62 df=25 P=0.8580
30 *
31 *TABLE BINORM2 C,D,FCR N=10000
32 C X -2 -1 0 1 2 5 sum
33 *Z *
34 *-2 8 35 79 70 23 4 219
35 *-1 30 196 454 436 155 31 1302
36 * 0 79 469 1117 1188 467 76 3396
37 * 1 81 467 1155 1198 506 101 3508
38 * 2 36 166 435 464 211 49 1361
39 D 5 2 30 66 71 33 12 214
40 * sum 236 1363 3306 3427 1395 273 10000
41 *Chi_square=37.02 df=25 P=0.0574
42 *

```

7.2.4 HISTO operation

HISTO <data>, <variable>, L / for PostScript and Canon laser printers
or

GHISTO <data>, <variable>, L / for screen graphics

forms the frequency distribution and plots the histogram of <variable> in <data> according to a classification given by

<variable>=<lower limit> (<step>) <upper limit>

L (optional parameter) is the first line for the table of the frequency distribution. The IND, CASES and SELECT specifications are available for selecting observations.

Furthermore, the frequency distribution is saved as a text file **FREQ.F** on the data disk and it can be used later by a HISTO operation of the form HISTO data>F, <variable>, L (e.g. HISTO FINLAND>F, Tax, CUR+1).

In this case, HISTO is using statistics already accumulated in the **FREQ.F** file

and scanning through the data file is not required anymore. Especially histograms of large data files are replotted faster by this option thus permitting immediate fitting of various theoretical distributions to the same frequency data (see FIT below).

Since the `FREQ.F` is always overwritten by a new frequency distribution when `HISTO` (without reference to a frequency file) is reactivated, the user can alter the extension `.F` of `FREQ.F` to another say `.F1` by the MS-DOS command `REN` and use `HISTO data>F1,L.`

As an application, we are studying a data file `HELSINKI` consisting of the mean temperature (`Temp`) and the rainfall (`Rain`) in July in Helsinki for 135 consecutive years from 1844 to 1978.

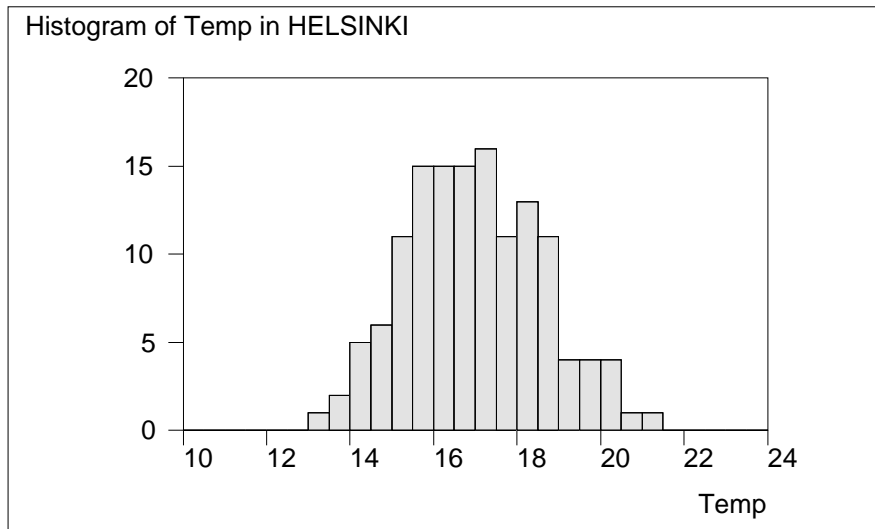
In the next exhibit, `HELSINKI` is introduced by `FILE STATUS` on lines 2-11:

```
21 1 SURVO 84C EDITOR Wed Feb 25 15:54:22 1992 D:\P2\STAT\ 100 100 0
1 *
2 *FILE STATUS HELSINKI
3 * Mean temperature (C) and rainfall (mm)
4 * Helsinki, July 1844-1978
5 *
6 *FIELDS: (active)
7 * 1 SA- 4 Year
8 * 2 NA- 4 Temp Mean temperature (C) in July (##.#)
9 * 3 NA- 1 Rain Rainfall (mm) in July (###)
10 *END
11 *SURVO 84C data file HELSINKI: record=128 bytes, M1=24 L=64 M=3 N=135
12 *
```

The frequency distribution of 'Temp' is then produced by the `HISTO` scheme on lines 13-15:

```
23 1 SURVO 84C EDITOR Wed Feb 25 15:55:08 1992 D:\P2\STAT\ 100 100 0
12 *
13 *HISTO HELSINKI,Temp,17
14 *Temp=10(0.5)24
15 *XSCALE=10(2)24 YSCALE=0(5)20 SIZE=1160,700 DEVICE=PS
16 *
17 *Frequency distribution of Temp in HELSINKI: N=135
18 *
19 *Class midpoint f % Sum %
20 * <=13.00 0 0.0 0 0.0
21 * 13.25 1 0.7 1 0.7
22 * 13.75 2 1.5 3 2.2
23 * 14.25 5 3.7 8 5.9
24 * 14.75 6 4.4 14 10.4
25 * 15.25 11 8.1 25 18.5
26 * 15.75 15 11.1 40 29.6
27 * 16.25 15 11.1 55 40.7
28 * 16.75 15 11.1 70 51.9
29 * 17.25 16 11.9 86 63.7
30 * 17.75 11 8.1 97 71.9
31 * 18.25 13 9.6 110 81.5
32 * 18.75 11 8.1 121 89.6
33 * 19.25 4 3.0 125 92.6
34 * 19.75 4 3.0 129 95.6
35 * 20.25 4 3.0 133 98.5
36 * 20.75 1 0.7 134 99.3
37 * 21.25 1 0.7 135 100.0
38 * >21.50 0 0.0 135 100.0
39 *Mean=16.98704 Std.dev.=1.623701
40 *
```


In addition to the numerical results on lines 17-39, the histogram is plotted by a PostScript printer (specification `DEVICE=PS`) in the following form:



The specifications `XSCALE`, `YSCALE`, `SIZE` and `DEVICE` (on line 15) are for plotting only and explained later in connection with the `PLOT` operation.

Other extra specifications of `HISTO` are, for example, `HOME`, `XDIV`, `YDIV`, `FRAME`, `GRID`, `TICK`, `HEADER`, `XLABEL`, `YLABEL` being mostly the same as those of the `PLOT` operation for scatter diagrams (`PLOT?`).

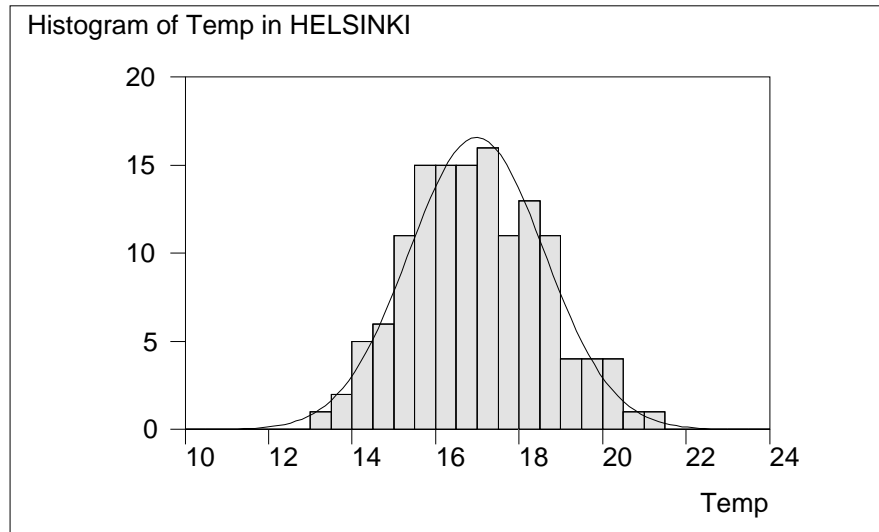
Various theoretical distributions may be fitted, and the corresponding frequency curve is plotted on the histogram by using a specification of the form `FIT=<name of the distribution>`. The following distributions are available:

| | |
|--|--|
| <code>NORMAL (mean,variance)</code> | <code>X is N(mean,variance)</code> |
| <code>LOGNORMAL (mean,variance)</code> | <code>log (X) is N(mean,variance)</code> |
| <code>UNIFORM (a,b)</code> | <code>X has uniform distribution on (a,b)</code> |
| <code>BINOMIAL (n,p)</code> | <code>X is Bin(n,p)</code> |
| <code>POISSON (mean)</code> | <code>X is Poisson(mean)</code> |

The user can also define other distributions.

If the parameters of the distribution are not given, the maximum likelihood estimates computed from the frequency distribution are used. For example, `FIT=NORMAL(17,2.56)` fits the normal distribution with given parameters whereas `FIT=NORMAL` uses normal distribution with parameters computed from the current sample.

When we add `FIT=NORMAL` to our previous example, the following graph and numerical results are obtained:



```

23 1 SURVO 84C EDITOR Sun Apr 12 08:14:52 1992 D:\P2\STAT\ 100 100 0
12 *
13 *HISTO HELSINKI,Temp,17_
14 *Temp=10(0.5)24 FIT=NORMAL
15 *XSCALE=10(2)24 YSCALE=0(5)20 SIZE=1160,700 DEVICE=PS,HEL2.PS
16 *
17 *Frequency distribution of Temp in HELSINKI: N=135
18 *
19 *Class midpoint f % Sum % e e f X^2
20 * <=13.00 0 0.0 0 0.0 1.0
21 * 13.25 1 0.7 1 0.7 1.2
22 * 13.75 2 1.5 3 2.2 2.3
23 * 14.25 5 3.7 8 5.9 4.0 8.5 8 0.0
24 * 14.75 6 4.4 14 10.4 6.5 6.5 6 0.0
25 * 15.25 11 8.1 25 18.5 9.4 9.4 11 0.3
26 * 15.75 15 11.1 40 29.6 12.4 12.4 15 0.6
27 * 16.25 15 11.1 55 40.7 14.9 14.9 15 0.0
28 * 16.75 15 11.1 70 51.9 16.3 16.3 15 0.1
29 * 17.25 16 11.9 86 63.7 16.3 16.3 16 0.0
30 * 17.75 11 8.1 97 71.9 14.8 14.8 11 1.0
31 * 18.25 13 9.6 110 81.5 12.2 12.2 13 0.0
32 * 18.75 11 8.1 121 89.6 9.2 9.2 11 0.3
33 * 19.25 4 3.0 125 92.6 6.3 6.3 4 0.8
34 * 19.75 4 3.0 129 95.6 3.9
35 * 20.25 4 3.0 133 98.5 2.2
36 * 20.75 1 0.7 134 99.3 1.2
37 * 21.25 1 0.7 135 100.0 0.5
38 * >21.50 0 0.0 135 100.0 0.4 8.3 10 0.4
39 *Mean=16.98704 Std.dev.=1.623701
40 *Fitted by NORMAL(16.987,2.6364) distribution
41 *Chi-square=3.596 df=9 P=0.9359
42 *

```

When FIT is given, the results will also include the fitted frequencies (columns e) and the common Chi-square test. In this test, consecutive classes are combined when necessary to meet the condition

expected frequency \geq min

where <min> is given by MINF=<min>. Default is MINF=5.

When using FIT for discrete distributions, the <variable> specification must be of the form <variable>=-0.5(1)<max.value+0.5>.

The next example illustrates fitting of a binomial distribution to a sample of 1000 observations. The sample is first generated from $\text{Bin}(10, 0.3)$ by summing in each observation 10 independent Bernoulli variates B_1, B_2, \dots, B_{10} . Bernoulli variates are generated simply as an integer part (int) of $p + \text{rnd}(1)$ since this is 1 with probability p and 0 with probability $1-p$.

```

17 1 SURVO 84C EDITOR Sun Apr 12 08:36:30 1992 D:\P2\STAT\ 100 100 0
1 *
2 *FILE CREATE BINOM,1,1,64,7,1000
3 * Sample from Bin(10,0.3)
4 *FIELDS:
5 *1 N 1 X
6 *END
7 *
8 *p=0.3
9 *VAR X=B1+B2+B3+B4+B5+B6+B7+B8+B9+B10 TO BINOM
10 *B1=int(p+rnd(1))
11 *B2=int(p+rnd(1))
12 *B3=int(p+rnd(1))
13 *B4=int(p+rnd(1))
14 *B5=int(p+rnd(1))
15 *B6=int(p+rnd(1))
16 *B7=int(p+rnd(1))
17 *B8=int(p+rnd(1))
18 *B9=int(p+rnd(1))
19 *B10=int(p+rnd(1))
20 *.....
21 *HISTO BINOM,X,25
22 *X=-0.5(1)10.5 XSCALE=-1:_,0(1)10,11:_ SIZE=1160,700 TICK=1,10
23 *FIT=Binomial(10) DEVICE=PS
24 *
25 *Frequency distribution of X in BINOM: N=1000
26 *
27 *Class midpoint f % Sum % e e f X^2
28 * 0.0 22 2.2 22 2.2 26.9 26.9 22 0.9
29 * 1.0 108 10.8 130 13.0 117.3 117.3 108 0.7
30 * 2.0 252 25.2 382 38.2 229.8 229.8 252 2.1
31 * 3.0 267 26.7 649 64.9 266.8 266.8 267 0.0
32 * 4.0 197 19.7 846 84.6 203.2 203.2 197 0.2
33 * 5.0 102 10.2 948 94.8 106.2 106.2 102 0.2
34 * 6.0 43 4.3 991 99.1 38.5 38.5 43 0.5
35 * 7.0 8 0.8 999 99.9 9.6
36 * 8.0 1 0.1 1000 100.0 1.6
37 * > 8.5 0 0.0 1000 100.0 0.2 11.3 9 0.5
38 *Mean=3.033000 Std.dev.=1.423345
39 *Fitted by BINOMIAL(10,0.3033) distribution
40 *Chi-square=5.138 df=6 P=0.5263
41 *

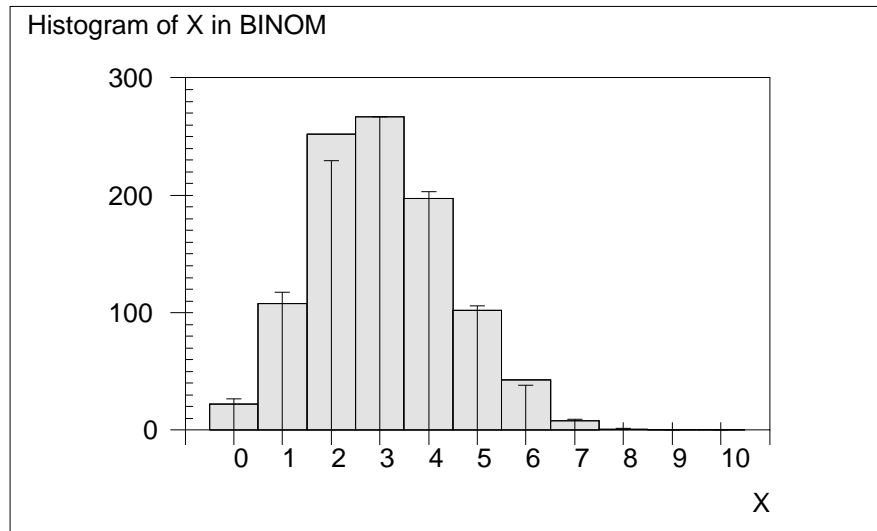
```

Specification $\text{FIT}=\text{Binomial}(10)$ implies a binomial distribution $\text{Bin}(10, p)$ to be fitted where p is to be estimated from the grouped data. In this case, the results (line 39) tell that the estimated p is 0.3033.

By using the **for** statement, the VAR operation (lines 9-19 above) could be replaced by

```
VAR X=for(I=1)to(10)term(T=0)sum(int(p+rnd(1))) TO BINOM.
```

Also the following graph is obtained:



*Distributions defined by the user

The user can specify any distribution by a **DENSITY** or a **PROBABILITY** definition given in the current edit field. **DENSITY** is used for continuous variables and **PROBABILITY** for discrete ones (with values 0,1,2,...).

A **FIT** specification refers to any such distribution in the same way as to predefined distributions (explained earlier).

The **DENSITY** definition has the form

```
DENSITY <name of distribution> (<list of parameters> )
      Y(X)=<density Y as a function of X and parameters>
```

The **PROBABILITY** definition has the form

```
PROBABILITY <name of distribution> (<list of parameters> )
      Y(X)=<probability of value X>
```

The rules in writing density and probability functions are the same as those of the **VAR** operation.

For example, the Beta (a,b) distribution is defined as:

```
DENSITY Beta(a,b)
Y(X)=if(X<=0)then(0)else(Y2)
Y2=if(X<1)then(X^(a-1)*(1-X)^(b-1))else(0)
```

Observe that the density function can be given without any normalizing constant since the computation procedure of **HISTO** automatically rescales the function so that the integral of $Y(X)$ is 1 on the range of classification defined by $\langle \text{variable} \rangle = \langle \text{lower limit} \rangle (\langle \text{step} \rangle) \langle \text{upper limit} \rangle$. The trapezoidal rule is used in numeric integration with step length $\langle \text{step} \rangle$. Since for non-normalized

densities, the integral has to be recomputed whenever any of the parameter values is changed, giving a normalized density speeds up the computation.

As an example of discrete distributions, the geometric distribution can be defined as:

```
PROBABILITY Geom(p)
Y(X)=if(X>=0)then(p*(1-p)^X)else(0)
```

In this case, the probabilities are correctly scaled. In general, they are rescaled so that their sum is 1 on the range of classification. The user is responsible for selecting a range wide enough for the purpose.

When referring to a user-defined distribution in the FIT specification, either the form `FIT=Beta(3,7)` (parameters given) or `FIT=Beta` (parameters are to be estimated from the current grouped data)

can be used. In the latter case, the parameter estimates are computed by the maximum likelihood method, using the polytope algorithm (Nelder, Mead 1965) in maximization of the likelihood function (See *Walsh: Methods of Optimization*, p.81-84).

The initial estimates may be given by an INIT specification `INIT=2,8` (for example). If INIT is not given, 0's are used as initial values. The initial step lengths for the optimization algorithm may be given by `STEP=<step1>, <step2>, ...` or `STEP=<step>` where `<step>` is the common step length for all parameters. Default is `STEP=0.1`.

The optimization process is displayed on the screen and it is the user's responsibility to interrupt it by . If the user does nothing, the process is terminated automatically after `MAXNF` function evaluations. Default is `MAXNF=32000`.

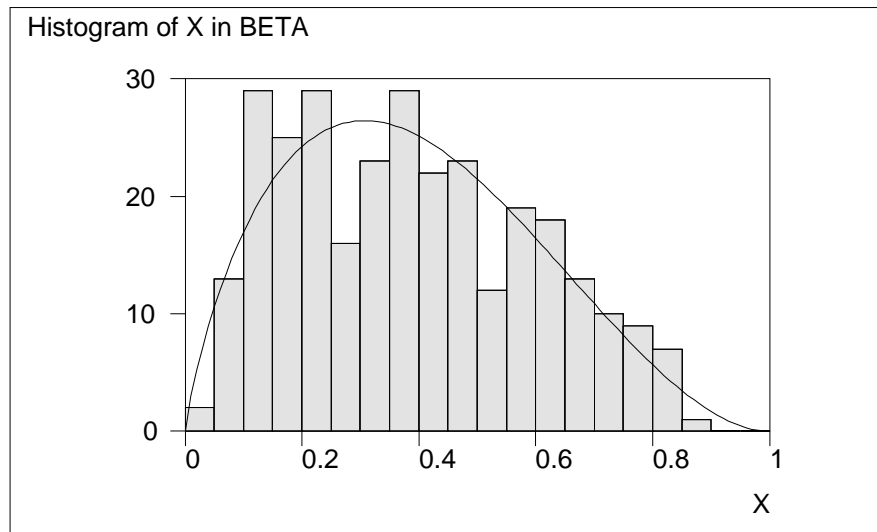
The following exhibit describes an application of HISTO for a user-defined distribution. We generate a sample BETA of 300 observations from `Beta(2,3)` distribution. Each observation is $X=U/(U+V)$ where U and V are independent χ^2 variables with 4 and 6 degrees of freedom, respectively. The χ^2 variables U and V are generated as sums of squares of independent standard normal variates.

```

22 1 SURVO 84C EDITOR Sun Apr 12 09:28:14 1992 D:\P2\STAT\ 100 100 0
1 *
2 *FILE CREATE BETA,4,1,64,7,300
3 * Sample from Beta(2,3) distribution
4 *FIELDS:
5 *1 N 4 X
6 *END
7 *
8 *VAR X=U/(U+V) TO BETA
9 *U=X1*X1+X2*X2+X3*X3+X4*X4
10 *V=X5*X5+X6*X6+X7*X7+X8*X8+X9*X9+X10*X10
11 *X1=probit(rnd(1))
12 *X2=probit(rnd(1))
13 *X3=probit(rnd(1))
14 *X4=probit(rnd(1))
15 *X5=probit(rnd(1))
16 *X6=probit(rnd(1))
17 *X7=probit(rnd(1))
18 *X8=probit(rnd(1))
19 *X9=probit(rnd(1))
20 *X10=probit(rnd(1))
21 *
22 * X is Beta(m,n) with m=2 n=3
23 * E(X) is m/(m+n)=0.4
24 * D(X) is sqrt(m*n/(m+n)/(m+n)/(m+n+1))=0.2
25 *.....

```

Using this sample of 300 observations we try to estimate the parameters, starting from initial values INIT=1, 5:



```

16 1 SURVO 84C EDITOR Sun Apr 12 09:35:22 1992 D:\P2\STAT\ 100 100 0
25 *.....
26 *
27 *HISTO BETA,X,37
28 *X=0(0.05)1 XSCALE=0(0.2)1 SIZE=1160,700 DEVICE=PS
29 *FIT=Beta
30 *
31 *DENSITY Beta(a,b)
32 * Y(X)=if(X<=0)then(0)else(Y2)
33 * Y2=if(X<1)then(X^(a-1)*(1-X)^(b-1))else(0)
34 *
35 *INIT=1,5
36 *
37 *HISTO: Estimated parameters of BETA:
38 *a=1.8331 (0.1474)
39 *b=2.8813 (0.2379)
40 *logL=-68.597884 # of function evaluations =79
41 *Correlations:
42 *          a      b
43 * a          1.000  0.817
44 * b          0.817  1.000
45 *Frequency distribution of X in BETA: N=300
46 *
47 *Class midpoint  f      %      Sum      %      e      e      f      X^2
48 *      0.025     2      0.7      2      0.7      5.8      5.8      2      2.4
49 *      0.075    13      4.3     15     5.0     13.9     13.9     13     0.1
50 *      0.125    29      9.7     44    14.7     19.3     19.3     29     4.9
51 *      0.175    25      8.3     69    23.0     22.9     22.9     25     0.2
52 *      0.225    29      9.7     98    32.7     25.1     25.1     29     0.6
53 *      0.275    16      5.3    114    38.0     26.2     26.2     16     4.0
54 *      0.325    23      7.7    137    45.7     26.4     26.4     23     0.4
55 *      0.375    29      9.7    166    55.3     25.7     25.7     29     0.4
56 *      0.425    22      7.3    188    62.7     24.4     24.4     22     0.2
57 *      0.475    23      7.7    211    70.3     22.5     22.5     23     0.0
58 *      0.525    12      4.0    223    74.3     20.3     20.3     12     3.4
59 *      0.575    19      6.3    242    80.7     17.8     17.8     19     0.1
60 *      0.625    18      6.0    260    86.7     15.1     15.1     18     0.6
61 *      0.675    13      4.3    273    91.0     12.3     12.3     13     0.0
62 *      0.725    10      3.3    283    94.3      9.5      9.5     10     0.0
63 *      0.775     9      3.0    292    97.3      6.9      6.9      9     0.6
64 *      0.825     7      2.3    299    99.7      4.5      4.5      7     0.0
65 *      0.875     1      0.3    300 100.0      2.6      2.6      1     0.0
66 *      >0.900     0      0.0    300 100.0      1.2      8.3      8     0.0
67 *Mean=0.389000 Std.dev.=0.208016
68 *Fitted by BETA(1.8331,2.8813) distribution
69 *Chi-square=18.02 df=14 P=0.2060
70 *

```

The estimated parameters as well as their standard errors (in parentheses) and correlations are given (lines 38-44).

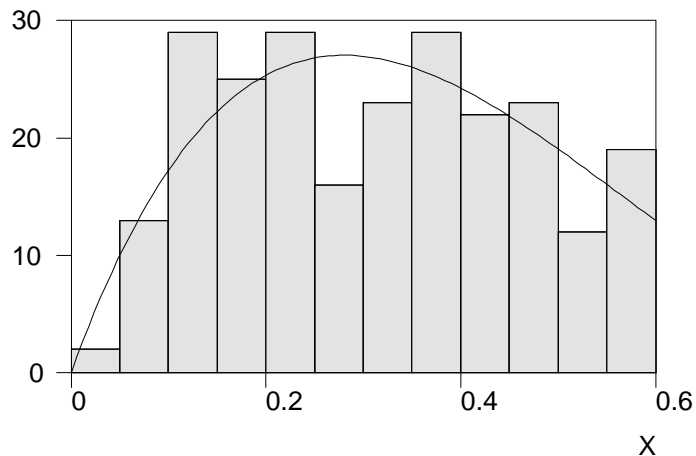
Since the density can be given without a normalizing coefficient, it is simple to estimate parameters even from truncated samples. In the next display, we have limited the range of permitted observations by giving a range from 0 to 0.6 (instead from 0 to 1) both in the X range and XSCALE specification. Then values of X above 0.6 will be excluded and the Beta density is renormalized to get the integral value 1 on interval (0,0.6). Of course, this kind of truncation makes the estimation of parameters more inefficient as seen from the following result:

```

16 1 SURVO 84C EDITOR Sun Apr 12 09:35:06 1992 D:\P2\STAT\ 100 100 0
25 *.....
26 *
27 *HISTO BETA, X, 37
28 *X=0(0.05)0.6 XSCALE=0(0.2)0.6 SIZE=1160,700 DEVICE=PS
29 *FIT=Beta
30 *
31 *DENSITY Beta(a,b)
32 * Y(X)=if(X<=0)then(0)else(Y2)
33 * Y2=if(X<1)then(X^(a-1)*(1-X)^(b-1))else(0)
34 *
35 *INIT=1,5
36 *
37 *HISTO: Estimated parameters of BETA:
38 *a=1.9838 (0.2372)
39 *b=3.5236 (0.6544)
40 *logL=-139.74232 # of function evaluations =108
41 *Correlations:
42 *
43 * a          a          b
44 * b          0.900     1.000
45 *Frequency distribution of X in BETA: N=242
46 *
47 *Class midpoint  f      %   Sum      %   e      e      f      X^2
48 * 0.025         2     0.8     2     0.8     5.2     5.2     2     2.0
49 * 0.075        13     5.4    15     6.2    13.8    13.8    13     0.0
50 * 0.125        29    12.0    44    18.2    19.9    19.9    29     4.2
51 * 0.175        25    10.3    69    28.5    23.9    23.9    25     0.0
52 * 0.225        29    12.0    98    40.5    26.2    26.2    29     0.3
53 * 0.275        16     6.6   114    47.1    27.0    27.0    16     4.5
54 * 0.325        23     9.5   137    56.6    26.6    26.6    23     0.5
55 * 0.375        29    12.0   166    68.6    25.2    25.2    29     0.6
56 * 0.425        22     9.1   188    77.7    23.1    23.1    22     0.1
57 * 0.475        23     9.5   211    87.2    20.5    20.5    23     0.3
58 * 0.525        12     5.0   223    92.1    17.6    17.6    12     1.8
59 * 0.575        19     7.9   242   100.0    14.5    14.5    19     1.4
60 *Mean=0.313223 Std.dev.=0.150604
61 *Fitted by BETA(1.9838,3.5236) distribution
62 *Chi-square=15.57 df=9 P=0.0765
63 *

```

Histogram of X in BETA



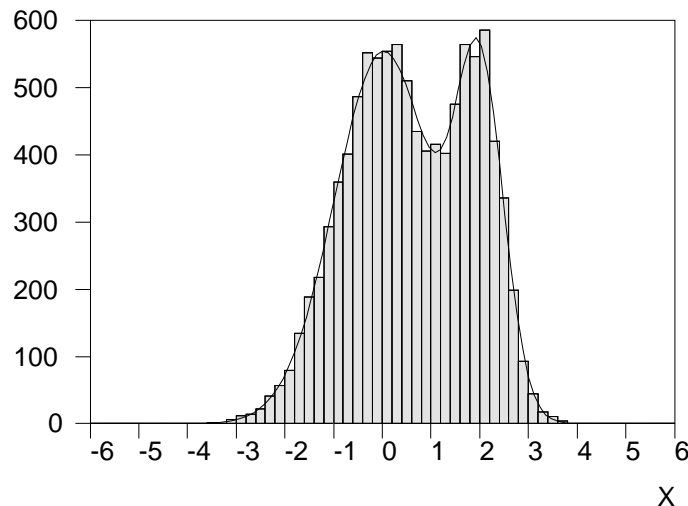
As a final example of HISTO, we shall study a mixture of two normal distributions of the form $p \cdot N(m_1, s_1^2) + (1-p) \cdot N(m_2, s_2^2)$. We shall generate 10000 observations with parameters $p=0.7$, $m_1=0$, $s_1=1$, $m_2=2$, $s_2=0.5$ and try to re-estimate them from starting values $INIT=0.5, -1, 1.5, 3, 1$.

```

20 1 SURVO 84C EDITOR Sun Apr 12 11:12:20 1992      D:\P2\STAT\ 100 100 0
1 *
2 *FILE CREATE SIMUDATA,4,1,64,7,10000
3 * Sample (N=10000) from a mixture of two normal distributions
4 *FIELDS:
5 *1 N 4 X
6 *END
7 *
8 *VAR X TO SIMUDATA
9 * X=if(rnd(1)<0.7)then(X1)else(X2)
10 * X1=probit(rnd(1))
11 * X2=0.5*probit(rnd(1))+2
12 *.....
13 *
14 *DENSITY MIXNORM(p,m1,s1,m2,s2)
15 *y(x)=c*(p/s1*exp(-0.5*((x-m1)/s1)^2)+(1-p)/s2*exp(-0.5*((x-m2)/s2)^2))
16 * c=0.39894226
17 *HISTO SIMUDATA, X, 22.
18 *X=-6(0.2)6 XSCALE=-6(1)6 YSCALE=0(100)600 DEVICE=PS
19 *FIT=MIXNORM INIT=0.5,-1,1.5,3,1
20 *SIZE=1160,800
21 *.....
22 *HISTO: Estimated parameters of MIXNORM:
23 *p=0.7003 (0.0125)
24 *m1=0.0301 (0.0296)
25 *s1=1.0077 (0.0193)
26 *m2=2.0095 (0.0180)
27 *s2=0.4906 (0.0137)
28 *logL=16043.335886 # of function evaluations = 311
29 *Correlations:
30 *
31 * p          m1      s1      m2      s2
32 * p          1.000  0.845  0.796  0.728 -0.707
33 * m1         0.845  1.000  0.802  0.673 -0.661
34 * s1         0.796  0.802  1.000  0.572 -0.591
35 * m2         0.728  0.673  0.572  1.000 -0.700
36 * s2        -0.707 -0.661 -0.591 -0.700  1.000

```

Histogram of X in SIMUDATA



```

20 1 SURVO 84C EDITOR Sun Apr 12 11:12:20 1992 D:\P2\STAT\ 100 100 0
36 *Frequency distribution of X in SIMUDATA: N=10000
37 *
38 *Class midpoint f % Sum % e e f X^2
39 * <=-3.6 0 0.0 0 0.0 1.1
40 * -3.5 2 0.0 2 0.0 1.2
41 * -3.3 1 0.0 3 0.0 2.4
42 * -3.1 6 0.1 9 0.1 4.5 9.3 9 0.0
43 * -2.9 12 0.1 21 0.2 8.2 8.2 12 1.7
44 * -2.7 14 0.1 35 0.4 14.3 14.3 14 0.0
45 * -2.5 22 0.2 57 0.6 24.0 24.0 22 0.2
46 * -2.3 41 0.4 98 1.0 38.7 38.7 41 0.1
47 * -2.1 57 0.6 155 1.6 59.9 59.9 57 0.1
48 * -1.9 79 0.8 234 2.3 89.1 89.1 79 1.2
49 * -1.7 135 1.4 369 3.7 127.6 127.6 135 0.4
50 * -1.5 189 1.9 558 5.6 175.6 175.6 189 1.0
51 * -1.3 218 2.2 776 7.8 232.5 232.5 218 0.9
52 * -1.1 293 2.9 1069 10.7 295.8 295.8 293 0.0
53 * -0.9 360 3.6 1429 14.3 362.0 362.0 360 0.0
54 * -0.7 401 4.0 1830 18.3 426.0 426.0 401 1.5
55 * -0.5 487 4.9 2317 23.2 482.0 482.0 487 0.1
56 * -0.3 552 5.5 2869 28.7 524.4 524.4 552 1.5
57 * -0.1 544 5.4 3413 34.1 548.6 548.6 544 0.0
58 * 0.1 554 5.5 3967 39.7 552.1 552.1 554 0.0
59 * 0.3 564 5.6 4531 45.3 535.0 535.0 564 1.6
60 * 0.5 510 5.1 5041 50.4 501.1 501.1 510 0.2
61 * 0.7 435 4.4 5476 54.8 458.7 458.7 435 1.2
62 * 0.9 406 4.1 5882 58.8 421.2 421.2 406 0.5
63 * 1.1 416 4.2 6298 63.0 405.3 405.3 416 0.3
64 * 1.3 402 4.0 6700 67.0 424.2 424.2 402 1.2
65 * 1.5 476 4.8 7176 71.8 476.3 476.3 476 0.0
66 * 1.7 564 5.6 7740 77.4 538.0 538.0 564 1.3
67 * 1.9 547 5.5 8287 82.9 570.4 570.4 547 1.0
68 * 2.1 586 5.9 8873 88.7 542.2 542.2 586 3.5
69 * 2.3 421 4.2 9294 92.9 450.6 450.6 421 1.9
70 * 2.5 336 3.4 9630 96.3 323.5 323.5 336 0.5
71 * 2.7 199 2.0 9829 98.3 199.7 199.7 199 0.0
72 * 2.9 93 0.9 9922 99.2 105.9 105.9 93 1.6
73 * 3.1 44 0.4 9966 99.7 48.4 48.4 44 0.4
74 * 3.3 18 0.2 9984 99.8 19.2 19.2 18 0.1
75 * 3.5 11 0.1 9995 100.0 6.8
76 * 3.7 4 0.0 9999 100.0 2.2
77 * 3.9 1 0.0 10000 100.0 0.7
78 * > 4.0 0 0.0 10000 100.0 0.4 10.0 16 3.6
79 *Mean=0.623320 Std.dev.=1.267161
80 *Fitted by MIXNORM(0.7003,0.0301,1.0077,2.0095,0.4906) distribution
81 *Chi-square=27.48 df=28 P=0.4921
82 *

```

7.3 Testing small samples

Statistical operations in Survo usually present, among other results, various test statistics and give their P values. Risk levels, etc. can also be computed afterwards, for example, by editorial computing which supplies standard distributions needed in common statistical inference.

A considerable part of testing situations related to comparisons of small samples is covered by a COMPARE operation. The tasks of COMPARE include comparing two or more independent samples, paired comparisons, tests based on rank correlations, testing for normality.

Comparison of two samples (independent or pairwise) is most easily performed by the `survo /COMPARE`. This `survo` presents questions about the mutual relation of the samples, the scale type of the variable, the distribution under null hypothesis and the form of the alternative hypothesis. On the basis of the answers given by the user, `/COMPARE` writes a short summary of current assumptions and activates a suitable form of the COMPARE operation. After the results have been obtained, `/COMPARE` studies them and gives an interpretation of the results as a provisional decision, i.e. it either accepts or rejects the null hypothesis.

Most of the methods in the COMPARE operation are non-parametric. Critical levels for tests are given exactly whenever possible or otherwise by approximations.

In many tests, the critical level will be determined by simulating the sampling process under the null hypothesis according to Fisher's randomization principle. The true critical level will appear as a limit for the proportion of those cases where the null hypothesis is rejected more clearly than in the samples observed. Also, the standard error of the simulated critical level is displayed.

The user can interrupt the simulation process displayed on the screen by a single keystroke. The time required for an appropriate convergence depends both on the size of the samples and the true risk level. Usually a crude result is obtained within a few seconds. If a fixed number of replicates is wanted, a `SIMUMAX=<# of replicates>` specification can be used.

The general structure of the COMPARE command is

```
COMPARE <sample1>,<sample2> , . . . ,L
```

where <sample1> , <sample2> , . . . are samples to be compared and L is an optional line label for results.

Each sample is given in the form <data> (<variable>). If <variable> is the

same word as <data> (as by default in simple data lists), it can be omitted.

For example, COMPARE DATA1(X),DATA1(Y) compares variables X and Y in data DATA1 and COMPARE Sample1,Sample2 compares variable Sample1 in data Sample1 with variable Sample2 in data Sample2.

To avoid confusions in selection of observations, the COMPARE operation does **not** allow IND, CASES and SELECT specifications. When, for example, samples from large data bases are to be compared, the normal procedure is to load those samples by FILE LOAD (with appropriate IND, CASES and SELECT specifications) to the edit field as data lists and the comparisons are made between the data lists.

The TEST specification tells the method of comparison. Its default value depends on the situation, e.g. on the number of samples.

7.3.1 Comparing two or more independent samples

If the TEST specification is not given, COMPARE assumes the samples to be independent and selects the tests accordingly. The following example illustrates this kind of a situation.

We activate COMPARE in the following situation:

```

21 1 SURVO 84C EDITOR Fri Mar 06 08:16:33 1987 D:\P2\STAT2\ 100 100 0
1 *
2 *Example from
3 *Conover: Practical Nonparametric Statistics, 2nd Edition, p.218
4 *
5 *The senior class in a particular high school had 48 boys. Twelve boys
6 *lived on farms and the other 36 lived in town. A test was devised to see
7 *if farm boys in general were more physically fit than the town boys.
8 *Each boy in the class was given a physical fitness test in which a low
9 *score indicates poor physical condition. The scores of the farm boys
10 *and the town boys are as follows.
11 *
12 *DATA FARM: 14.8 7.3 5.6 6.3 9.0 4.2 10.6 12.5 12.9 16.1 11.4 2.7 END
13 *DATA TOWN: 12.7 14.2 12.6 2.1 17.7 11.8 16.9 7.9 16.0 10.6 5.6 5.6
14 *          7.6 11.3 8.3 6.7 3.6 1.0 2.4 6.4 9.1 6.7 18.6 3.2
15 *          6.2 6.1 15.3 10.6 1.8 5.9 9.9 10.6 14.8 5.0 2.6 4.0 END
16 *
17 *COMPARE FARM,TOWN,18
18 *

```

Immediately after the activation, the screen is cleared and temporary information is displayed as follows:

```

21 1 SURVO 84C EDITOR Fri Mar 06 08:22:37 1987 D:\P2\STAT2\ 100 100 0
Independent samples          FARM          TOWN
Sample size                  12          36
Mean                        9.450000    8.650000
Standard deviation           4.283902    4.914265
Student's t=0.503 df=46 (P=0.6913 one-sided test)
Sum of ranks (R)             321         855
Mann-Whitney (U)            243         189
(P=0.7398 one-sided Mann-Whitney, normal approximation)
Critical levels by simulation:
      Mean      R or U
Critical level 0.68763 0.73119 N=5900
Standard error 0.00603 0.00577

To interrupt simulation press any key!

```

Basic statistics on the samples, the two-sample t test and the Mann-Whitney test is presented. The P value for the Mann-Whitney test is given first as a normal approximation.

On the last lines, information on the ongoing simulation process is displayed. The numbers in the shaded area are continuously updated and the user can interrupt the process by pressing any key.

In this setup, critical levels for the one-tailed Mann-Whitney test (R or U) and the Fisher-Pitman test (Mean) will be obtained by simulation. In both cases, the joint sample of $12+36=48$ observations is split randomly into two subsamples of 12 and 36 observations. For these subsamples, the rank sums (Mann-Whitney) and sums of original scores (Fisher-Pitman) are computed and compared to corresponding sums in the true FARM and TOWN samples. The randomization process will be repeated indefinitely and relative frequencies for replicates where the sums exceed the true sums are counted. The counts are updated on the screen after each 100th replicate. The relative frequencies thus formed will tend to critical values of one-tailed tests. To see the accuracy of the P value estimates, their standard errors are displayed, too.

The user may limit the number of replicates by a SIMUMAX specification. Default is SIMUMAX=100000. Below, the results after this automatic interrupt are displayed:

```

21 1 SURVO 84C EDITOR Fri Mar 06 09:25:52 1987 D:\P2\STAT2\ 100 100 0
16 *
17 *COMPARE FARM,TOWN,18_
18 *Independent samples FARM TOWN
19 *Sample size 12 36
20 *Mean 9.450000 8.650000
21 *Standard deviation 4.283902 4.914265
22 *Student's t=0.503 df=46 (P=0.6913 one-sided test)
23 *Sum of ranks (R) 321 855
24 *Mann-Whitney (U) 243 189
25 *(P=0.7398 one-sided Mann-Whitney, normal approximation)
26 *Critical levels by simulation:
27 * Mean R or U
28 *Critical level 0.69690 0.74115 N=100000
29 *Standard error 0.00145 0.00139
30 *

```

The next display illustrates the results given by the `sucro /COMPARE` in the same task. The assumptions made in the conversation with `/COMPARE` appear on lines 18-24 and the concluding summary on lines 36-39.

```

19 1 SURVO 84C EDITOR Thu Apr 16 14:30:48 1992 D:\P2\STAT2\ 100 100 0
16 *
17 */COMPARE FARM,TOWN_
18 *Assumptions:
19 * Samples FARM and TOWN are independent.
20 * Measurements are on the interval scale.
21 * No assumption on normality.
22 *Null hypothesis: Both samples are obtained from the same distribution.
23 *Alternative hypothesis:
24 * Sample TOWN distribution values lower (one-sided test).
25 *-----
26 *COMPARE FARM,TOWN,CUR+1 / SIMUMAX=0
27 *Independent samples FARM TOWN
28 *Sample size 12 36
29 *Mean 9.450000 8.650000
30 *Standard deviation 4.283902 4.914265
31 *Student's t=0.503 df=46 (P=0.6913 one-sided test)
32 *Sum of ranks (R) 321 855
33 *Mann-Whitney (U) 243 189
34 *(P=0.7398 one-sided Mann-Whitney, normal approximation)
35 *-----
36 *Summary:
37 * The decisions are based on Mann-Whitney-test.
38 * The critical level of the one-sided test is 0.2602 .
39 * Hypothesis "Samples from the same distribution" is not rejected.
40 *

```

The `TEST` specification indicates the test procedures to be selected. When comparing two independent samples, the following alternatives are available:

```

TEST=Mann-Whitney (Conover p.216)
TEST=Pitman (Conover p.328)
TEST=t (Conover p.225)
TEST=Smirnov (Conover p.369)

```

If `TEST` is not given, the t , Mann-Whitney and Pitman tests are presented. In the Smirnov test based on empirical distribution functions, the critical levels are determined by simulation.

When more than two samples are to be compared, the alternatives are

```

TEST=Kruskal-Wallis (Conover p.229)
TEST=F (Conover p.237)

```

If `TEST` is missing, both Kruskal-Wallis and F tests are performed.

7.3.2 Pairwise comparisons, rank correlations

If two samples with matched pairs are to be compared, the samples should be presented as separate, properly ordered data sets or as two variables within the same data set. Thus, the number of observations must be same in both samples. The samples are compared as follows:

```

24 1 SURVO 84C EDITOR Fri Mar 06 17:24:19 1987 D:\P2\STAT2\ 100 100 0
55 *
56 *Example from
57 *Conover: Practical Nonparametric Statistics, 2nd Edition, p.253
58 *
59 *Twelve sets of identical twins were given psychological tests to
60 *measure their aggressiveness. The emphasis is on examination of the
61 *degree of similarity between the first-born and second-born twins
62 *within the same set.
63 *
64 *DATA FIRST: 86 71 77 68 91 72 77 91 70 71 88 87 END
65 *DATA SECOND: 88 77 76 64 96 72 65 90 65 80 81 72 END
66 *
67 *COMPARE FIRST,SECOND,68 / TEST=PAIRED
68 *Paired comparisons:
69 *Samples: N=12          FIRST          SECOND          Difference
70 *Mean                  79.08333        77.16667        -1.916667
71 *Standard deviation    8.887768        10.37333        7.153617
72 *Paired t=-0.928 (P=0.1866 one-sided t test df=11)
73 *Wilcoxon signed ranks test=-0.756 (P=0.2247 normal approximation)
74 *Critical levels by simulation:
75 *          Differences Signed rank
76 *Critical level 0.20141  0.24269 N=7800
77 *Standard error 0.00454  0.00485
78 *

```

The specification TEST=PAIRED implies a matched-pairs comparison. The procedure is analogous to the case of independent samples. The Mann-Whitney test is replaced by the Wilcoxon signed ranks test and the Fisher-Pitman test by a randomization test based on score differences.

Rank correlations are obtained by using the specification TEST=CORRELATION:

```

24 1 SURVO 84C EDITOR Fri Mar 06 17:35:42 1987 D:\P2\STAT2\ 100 100 0
79 *.....
80 *
81 *COMPARE FIRST,SECOND,82 / TEST=CORRELATION
82 *Rank correlations:
83 *Samples: N=12          FIRST          SECOND
84 *Mean                  79.08333        77.16667
85 *Standard deviation    8.887768        10.37333
86 *Product moment correlation R=0.7344 (P=0.9967)
87 *Spearman's Rho=0.7355
88 *Kendall's Tau=0.5606 (Nc=49 Nd=12 P=0.9944 normal approximation)
89 *          Exact P=0.9973
90 *Critical levels by simulation:
91 *          R          Rho
92 *Critical level 0.99594 0.99624 N=17000
93 *Standard error 0.00049 0.00047
94 *

```

7.3.3 Testing for normality



COMPARE <sample>, #NORMAL, L

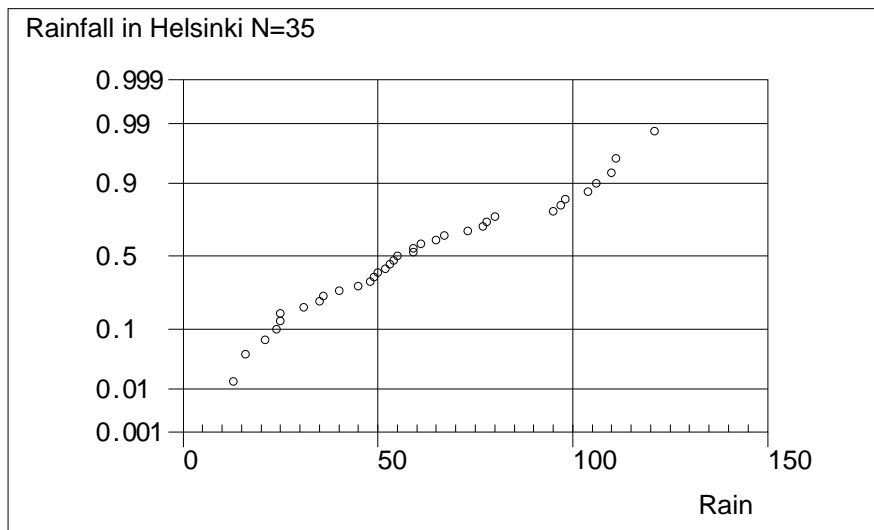
performs certain tests for normality in a given sample. If the sample size is at most 50, the Shapiro-Wilk test with a P value will be obtained. If the sample size is more than 50, the test of d'Agostino is presented. In addition, the Anderson-Darling test is computed in both cases.

In the next example, normality of 'Rain' in the HELSINKI data is tested for 35 last years. The data values are first loaded into the edit field as a data list 'Rain' and then COMPARE is called as follows:

```
27 1 SURVO 84C EDITOR Thu Apr 16 15:19:06 1992 D:\P2\STAT2\ 100 100 0
42 *
43 *FILE LOAD HELSINKI / VARS=Rain IND=ORDER,101,135 FORMAT=LIST
44 *DATA Rain: 59 98 35 95 25 49 48 40 45 121 78 16 104 77 73 55 110 65 53
45 *31 36 61 54 25 59 67 111 21 80 24 52 13 50 106 97 END
46 *
47 *COMPARE Rain, #NORMAL, CUR+1
48 *Tests for normality: Sample Rain N=35
49 *Mean=60.94286 Std.dev.=30.07730
50 *Skewness=0.322912 Kurtosis=-0.953200 (normal values=0)
51 *Median=55.00000
52 *Shapiro-Wilk W=0.950 (P=0.162)
53 *D'Agostino D=0.284624 Y=0.499 (0.2<P<0.8)
54 *Anderson-Darling A=0.470 (P=0.235)
55 *
```

Normality of the sample may be assessed also by plotting the ordered sample on a normal probability paper. Below, the 35 last observations of the HELSINKI data are sorted in ascending order by FILE SORT (on line 46) and the sorted data is plotted on the probability paper by a GPLOT operation.

```
27 1 SURVO 84C EDITOR Thu Apr 16 15:23:14 1992 D:\P2\STAT2\ 100 100 0
55 *
56 *FILE COPY Rain,RAIN2 / A data list cannot be sorted.
57 *FILE SORT RAIN2 BY Rain TO RAINSORT
58 *GPLOT RAINSORT,Rain,PROBIT / HEADER=Rainfall_in_Helsinki_N=35
59 *GRID=XY TICK=5
60 *
```



MNTEST?

7.4 Linear regression analysis

Survo offers several possibilities for linear regression analysis. The main operations are LINREG and REGDIAG described in this section. Also a related operation SMOOTH for semiparametric smoothing of data is included here.

The ESTIMATE operation intended primarily for nonlinear models can equally be used for linear regression analysis. Linear regression is performed implicitly in the PLOT operation for scatter diagrams where the TREND specification adds a linear trend to the graph. The INTERP operation described earlier is based on polynomial regression. As a general tool for linear systems, the matrix interpreter (MAT operations) is useful in various calculations related to linear models.

7.4.1 LINREG operation

LINREG data, L

performs a linear regression analysis on variables in 'data'. The regressands are indicated by masks Y's and the regressors by X's. Normally there is one regressand (Y) only. A constant term equal to 1 is automatically included in models.

An optional weight variable is included by the mask W. The sum of weights is always rescaled to N , the number of active observations.

Optional variables for output of residuals and predicted values of the model are pointed out by masks R and P, respectively. The first line for the results (L) is optional.

The VARS and MASK specifications are used to indicate X,Y,W,R and P variables. The observations are selected by optional IND, CASES and SELECT specifications.

If the results level is at least 70 (see RESULTS?), the means, standard deviations and correlations of variables included in the model will also be printed.

The regression coefficients are saved in the matrix file REG.M on the current data disk.

If several regressands (Y) occur, the model will be estimated for each of them. However, no residuals and predicted values are saved in this case. Similarly, no matrix file for the results will be created.

The LINREG operation computes first the means, standard deviations and correlations of all X,Y variables. When the number of variables and/or observations is large, time may be spared by computing these statistics once for all relevant variables by CORR. The correlations will then be found in the matrix file CORR.M and the means and the standard deviations in MSN.M. These files may be referred to by entering LINREG in the form

LINREG data>M,L

where data>M implies the CORR and MSN files with the extension .M to be employed as input for LINREG. In this form of LINREG, no further selection of observations (by IND, CASES) can occur; selections made by CORR are maintained. However, different models with partially same variables can be analyzed. If not all current X,Y variables are in CORR and MSN files, an error message will be given and computation is interrupted.

By altering the extension .M to another (say .M1) in current matrices, these can be used later by LINREG data>M1, although the original .M files have been overwritten by new CORR activations.

LINREG estimates the model by the ordinary (weighted) least squares method. The calculations are performed by inverting the correlation matrix of regressors by the Cholesky method. If serious multicollinearity among regressors occurs, an error message is given and the process is interrupted.

In the next exhibit, a small data set ($N=9$) of three variables (Y,X1,X2) is generated according to the model

$$Y=50+2*X1+5*X2+eps$$

where eps is $N(0,0.25)$. The VAR operation on line 7 computes the Y values (rounded to integers) for given X1,X2 values.

When the LINREG operation is activated, the masks on line M (3) tell the roles of variables in the analysis. Thus, in addition to results presented from line 17 onwards, the residuals of the model will appear as values of variable 'Res' due to the mask 'RR.RRR'. To see the effects of LINREG, the results are shaded.

```

15 1 SURVO 84C EDITOR Fri Apr 17 10:35:44 1992 D:\P2\STAT2\ 150 100 0
1 *
2 *DATA TEST,A,B,N,M
3 M YY XX XX RR.RRR
4 N Y X1 X2 Res
5 A 55 1 1 -1.056
6 * 62 1 2 0.611
7 * 67 1 3 0.278
8 * 59 2 1 0.444
9 * 64 2 2 0.111
10 * 69 2 3 -0.222
11 * 61 3 1 -0.056
12 * 67 3 2 0.611
13 B 71 3 3 -0.722
14 *
15 *VAR Y=50+2*X1+5*X2+0.5*probit(rnd(1)) TO TEST
16 *LINREG TEST,17 / RESULTS=50
17 *Linear regression analysis: Data TEST, Regressand Y N=9
18 *Variable Regr.coeff. Std.dev. t beta
19 *X1 2.500000 0.274986 9.091 0.422
20 *X2 5.333333 0.274986 19.39 0.900
21 *constant 48.22222 0.809537 59.57
22 *Variance of regressand Y=26.36111111 df=8
23 *Residual variance=0.453703704 df=6
24 *R=0.9935 R^2=0.9871
25 *DW=1.9796
26 *

```

The results include regression coefficients and their standard deviations, t

values (=Regr.coeff./Std.dev.) and standardized regression coefficients (beta=Regr.coeff.*stddev(X)/stddev(Y)). The multiple correlation (R), its square (R²) and the Durbin-Watson test statistics (DW) for the first-order autocorrelation of residuals are printed, too. The last statistics is given only when residuals or predicted values are computed.

*Stepwise regression

For stepwise and other more or less automatic model selection procedures, a special `surco /STEPREG` has been programmed by *Mikko Karpoja*. It starts by computing the means, standard deviations and correlations of selected variables for all active observations by `CORR`. On the basis of this information, the `LINREG` operation is called repeatedly with various alternative model specifications.

The `surco` is activated by the command

```
/STEPREG <data>, <rule>, MASK=<model as X's and Y>
```

Eight different rules for model selection are available. These rules cover backward elimination (`DESC`), forward selection (`INCL`), stepwise (`STEP`) and all possible regressions (`ALL`). In the last alternative, the maximum number of regressors is limited to 6.

More detailed information is obtained by activating `/STEPREG` without parameters.

The next example illustrates the use of `/STEPREG` in polynomial regression. At first, a sample of 50 observations is created by `FILE` and `VAR` operations. The 'right' model gives the variations of `Y` as a third degree polynomial of `X` and with an additive error term with variance $100^2=10000$.

```
54 1 SURVO 84C EDITOR Fri Apr 17 13:34:21 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *FILE CREATE POLYREG
3 * Test data for polynomial regression
4 * Y=10+15*X-2*X^2+0.1*X^3+eps, eps=100*N(0,1)
5 * X=1,2,...,50
6 *FIELDS:
7 *1 N 4 Y
8 *2 N 4 X
9 *3 N 4 X2
10 *4 N 4 X3
11 *5 N 4 X4
12 *END
13 *
14 *FILE INIT POLYREG,50
15 *VAR X=ORDER-25 TO POLYREG
16 *VAR X2,X3,X4 TO POLYREG
17 * X2=X*X X3=X2*X X4=X3*X
18 *VAR Y=10+15*X-2*X2+0.1*X3+100*probit(rnd(2)) TO POLYREG
19 *
```

Activation of `/STEPREG` with the selection rule `STEP` and with all powers of `X` up to 4 as potential regressors leads to the following result:

```

17 1 SURVO 84C EDITOR Fri Apr 17 13:45:18 1992 D:\P2\STAT2\ 100 100 0
19 *
20 */STEPREG POLYREG,STEP,MASK=YXXXX
21 *RESULTS=3
22 *LINREG POLYREG>M,CUR+3 /
23 *MASK=YXXX-
24 *Selected model :_
25 *Linear regression analysis: Data POLYREG>M, Regressand Y N=50
26 *Variable  Repr.coef.   Std.dev.   t      beta
27 *X          13.33533    2.292122  5.818  0.233
28 *X2        -1.945917    0.071559 -27.19 -0.440
29 *X3         0.100902    0.005616  17.97  0.723
30 *constant   8.620425
31 *Variance of regressand Y=695544.5788 df=49
32 *Residual variance=8749.361661 df=46
33 *R=0.9941 R^2=0.9882
34 *

```

During the selection process, the LINREG operation (on line 22) has been activated 15 times with different MASK specifications (on line 23). In this case, the true model was found. When making the same experiment with different seed numbers for the rnd function (line 18 in the first display), rnd(1) didn't give this model but it also included the 4th degree term. However, in most of the cases, for example with seed numbers 2,3,...,10, the procedure does not miss the true model. This example will be continued in section 7.4.3.

7.4.2 Regression diagnostics

REGDIAG data,L

estimates a linear regression model and computes various statistics for regression diagnostics. REGDIAG does not use correlations. It orthogonalizes the design matrix \mathbf{X} by the singular value decomposition and derives all results from that basis. By this approach, even strongly multicollinear situations may be examined and handled to a certain extent. For the techniques employed in REGDIAG, see e.g. Belsley, Kuh and Welch: *Regression Diagnostics*, Wiley 1980.

Since in REGDIAG the entire data set has to be kept in the central memory, the size of the maximum data set is limited to 8192 data values.

The model and the scope of the results are indicated as in the LINREG operation. REGDIAG permits, however, only one regressand (Y) at a time. Furthermore, the following masks (in addition to X,Y,R,P) can be used:

H = Diagonal of the hat matrix $\mathbf{X}(\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'$

S = Studentized residuals

C = (signed square roots of) Cook's distances

The constant term is omitted by setting the specification CONSTANT=0.

REGDIAG gives the condition number of the design matrix as the ratio of the largest and smallest singular value.

As an example, we study the following intercountry life-cycle savings data quoted from the aforementioned book (p.41-42). It offers an extra challenge for regression diagnostics since the source suffers from a printing error.

The data set has been saved in a data file with the following structure:

```

20 1 SURVO 84C EDITOR Tue Mar 17 11:38:35 1987 D:\P2\STAT2\ 120 80 0
1 *
2 *FILE STATUS SAVINGS
3 * Intercountry life-cycle savings data from
4 * Belsley,Kuh,Welsch(1980): Regression Diagnostics, p.41
5 * Each variable is an average over period 1960-1970
6 *FIELDS: (active)
7 * 1 SA_ 16 Country
8 * 2 NY_ 4 SR aggregate personal savings rate (##.##)
9 * 3 NX_ 4 POP15 percentage of population under 15 years (##.##)
10 * 4 NX_ 4 POP75 percentage of population over 75 years (##.##)
11 * 5 NX_ 8 DPI per-capita disposable income in USD (####.##)
12 * 6 NX_ 4 Rate percentage growth rate of DPI (##.##)
13 * 8 NH- 4 HAT1 Correct hat diagonal (#####)
14 * 11 NA- 4 HAT2 Incorrect hat diagonal (#####)
15 *END
16 *SURVO 84C data file SAVINGS: record=90 bytes, M1=16 L=64 M=11 N=50
17 *

```

and the data itself reads as follows:

```

18 1 SURVO 84C EDITOR Tue Mar 17 11:42:34 1987 D:\P2\STAT2\ 120 80 0
17 *
18 *FILE LOAD SAVINGS_
19 *DATA SAVINGS*,A,B,C
20 C Country SR POP15 POP75 DPI Rate HAT1 HAT2
21 A Australia 11.43 29.35 2.87 2329.68 2.87 0.0677 0.0672
22 * Austria 12.07 23.32 4.41 1507.99 3.93 0.1204 0.1199
23 * Belgium 13.17 23.80 4.43 2108.47 3.82 0.0875 0.0855
24 * Bolivia 5.75 41.89 1.67 189.13 0.22 0.0895 0.0892
25 * Brazil 12.88 42.19 0.83 728.47 4.56 0.0696 0.0673
26 * Canada 8.79 31.72 2.85 2982.88 2.43 0.1584 0.1579
27 * Chile 0.60 39.74 1.34 662.86 2.67 0.0373 0.0358
28 * Taiwan 11.90 44.75 0.67 289.52 6.51 0.0780 0.0781
29 * Colombia 4.98 46.64 1.06 276.65 3.08 0.0573 0.0578
30 * Costa Rica 10.78 47.64 1.14 471.24 2.80 0.0755 0.0738
31 * Denmark 16.85 24.42 3.93 2496.53 3.99 0.0627 0.0626
32 * Ecuador 3.59 46.31 1.19 287.77 2.19 0.0637 0.0633
33 * Finland 11.24 27.84 2.37 1681.25 4.32 0.0920 0.0800
34 * France 12.64 25.06 4.70 2213.82 4.52 0.1362 0.1277
35 * Germany(F.R.) 12.55 23.31 3.35 2457.12 3.44 0.0874 0.0830
36 * Greece 10.67 25.62 3.10 870.85 6.28 0.0966 0.0892
37 * Guatemala 3.01 46.05 0.87 289.71 1.48 0.0605 0.0613
38 * Honduras 7.70 47.32 0.58 232.44 3.19 0.0601 0.0614
39 * Iceland 1.27 34.03 3.08 1900.10 1.12 0.0705 0.0660
40 * India 9.00 41.31 0.96 88.94 1.54 0.0715 0.0674
41 * Ireland 11.34 31.16 4.19 1139.95 2.99 0.2122 0.1978
42 * Italy 14.28 24.52 3.48 1390.00 3.54 0.0665 0.0639
43 * Japan 21.10 27.01 1.91 1257.28 8.21 0.2233 0.1905
44 * Korea 3.98 31.74 0.91 207.60 5.81 0.0608 0.2145
45 * Luxembourg 10.35 21.80 3.73 2449.39 1.57 0.0863 0.0844
46 * Malta 15.48 32.54 2.47 601.05 8.12 0.0794 0.0776
47 * Norway 10.25 25.95 3.67 2231.03 3.62 0.0479 0.0479
48 * Netherlands 14.65 24.71 3.25 1740.70 7.66 0.0906 0.0854
49 * New Zealand 10.67 32.61 3.17 1487.52 1.76 0.0542 0.0521
50 * Nicaragua 7.30 45.04 1.21 325.54 2.48 0.0504 0.0508
51 * Panama 4.44 43.56 1.20 568.56 3.61 0.0390 0.0400
52 * Paraguay 2.02 41.18 1.05 220.56 1.03 0.0694 0.0665
53 * Peru 12.70 44.19 1.28 400.06 0.67 0.0650 0.0651
54 * Philippines 12.78 46.26 1.12 152.01 2.00 0.0643 0.0643
55 * Portugal 12.49 28.96 2.85 579.51 7.48 0.0971 0.0930
56 * South Africa 11.14 31.94 2.28 651.11 2.19 0.0651 0.0604
57 * South Rhodesia 13.30 31.92 1.52 250.96 2.00 0.1608 0.1375
58 * Spain 11.77 27.74 2.87 768.79 4.35 0.0773 0.0717
59 * Sweden 6.86 21.44 4.54 3299.49 3.01 0.1240 0.1226
60 * Switzerland 14.13 23.49 3.73 2630.96 2.70 0.0736 0.0733
61 * Turkey 5.13 43.42 1.08 389.66 2.96 0.0396 0.0401
62 * Tunisia 2.81 46.12 1.21 249.87 1.13 0.0746 0.0738
63 * United Kingdom 7.81 23.27 4.46 1813.93 2.01 0.1165 0.1155
64 * United States 7.56 29.81 3.43 4001.89 2.45 0.3337 0.3290
65 * Venezuela 9.22 46.40 0.90 813.39 0.53 0.0863 0.0866
66 * Zambia 18.56 45.25 0.56 138.33 5.14 0.0643 0.0636
67 * Jamaica 7.72 41.12 1.73 380.47 10.23 0.1408 0.1437
68 * Uruguay 9.24 28.13 2.72 766.54 1.88 0.0979 0.0915
69 * Libya 8.89 43.69 2.07 123.50 16.71 0.5315 0.5351
70 B Malaysia 4.71 47.20 0.66 243.60 5.08 0.0653 0.0674
71 *

```

The two extra variables (appearing as columns HAT1 and HAT2) do not belong to the original data. They will be commented upon later.

If now a linear regression model having SR as a regressand and the other variables (POP15,POP75,DPI,Rate) as regressors is fitted by the REGDIAG operation, we obtain the results:

```

19 1 SURVO 84C EDITOR Tue Mar 17 11:59:07 1987 D:\P2\STAT2\ 120 80 0
71 *.....
72 *VARS=SR(Y),POP15(X),POP75(X),DPI(X),Rate(X),HAT2(H)
73 *REGDIAG SAVINGS,74
74 *Regression diagnostics on data SAVINGS: N=50
75 *Regressand SR # of regressors=5 (Constant term included)
76 *Condition number of scaled X: k=31.685277
77 *Variable Regr.coeff. Std.dev. t
78 *Constant 21.531171 7.0000143 3.0759
79 *POP15 -0.3231888 0.1383109 -2.3367
80 *POP75 -0.8391659 1.0582703 -0.7930
81 *DPI -0.0001859 0.0009716 -0.1913
82 *Rate 0.4113508 0.2058051 1.9987
83 *Variance of regressand SR=20.07404584 df=49
84 *Residual variance=15.80904619 df=45
85 *R=0.5261 R^2=0.2768 Durbin-Watson=2.038
86 *

```

The results don't, however, agree with those given in the source. For example, $R^2=0.2768$ is smaller than the true one (0.33). To see the reason for this inconsistency, the diagonal of the hat matrix computed as variable HAT2 by REGDIAG can be studied more carefully. Of course, other measures used in regression diagnostics could be helpful, too.

Since the source gives the true hat diagonal HAT1 (presented in the data set above), it is easy to see that there is a dramatic disagreement in 'Korea' where HAT1=0.0608 and HAT2=0.2145. Thus something is wrong in POP15, POP75, DPI, Rate values of 'Korea'.

By comparing the values of 'Korea' to similar countries like 'Japan' and 'Taiwan', we came to the conclusion that POP15=31.74 should be corrected to POP15=41.74 (printing error in the most significant digit), and after this alteration we got results identical to those given in the source:

```

19 1 SURVO 84C EDITOR Tue Mar 17 12:25:26 1987 D:\P2\STAT2\ 120 80 0
86 *.....
87 *VARS=SR(Y),POP15(X),POP75(X),DPI(X),Rate(X),HAT1(H)
88 *REGDIAG SAVINGS,89
89 *Regression diagnostics on data SAVINGS: N=50
90 *Regressand SR # of regressors=5 (Constant term included)
91 *Condition number of scaled X: k=34.8683
92 *Variable Regr.coeff. Std.dev. t
93 *Constant 28.566233 7.3545026 3.8842
94 *POP15 -0.4611959 0.1446418 -3.1885
95 *POP75 -1.6914514 1.0835935 -1.5610
96 *DPI -0.0003370 0.0009311 -0.3620
97 *Rate 0.4096888 0.1961967 2.0882
98 *Variance of regressand SR=20.07404584 df=49
99 *Residual variance=14.46025813 df=45
100 *R=0.5818 R^2=0.3385 Durbin-Watson=1.934
101 *

```

The source gives the standard error of 'Constant' as 7.345 (versus our 7.3545), but other results match completely within the accuracy of presentation. Since in the source the printed values have been truncated, while our results are correctly rounded, some HAT1 values differ in the last decimal place.

We found 'Korea' to be the source of error by knowing the right hat diagonal in advance. It is obvious that the same conclusion could be achieved even without this information by computing all the diagnostic measures (from the erroneous data). The following exhibit summarizes those measures after using the previous REGDIAG with

VARS=SR(Y), POP15(X), POP75(X), DPI(X), Rate(X), HAT2(H)&
RES(R), STUDES(S), COOK(C)

```
18 1 SURVO 84C EDITOR Tue Mar 17 13:10:48 1987 D:\P2\STAT2\ 120 80 0
```

| C | Country | HAT2 | RES | STUDES | COOK |
|----|----------------|--------|---------|---------|--------------------------|
| 6 | Australia | 0.0672 | 1.0453 | 0.2694 | 0.0327 |
| 7 | Austria | 0.1199 | 0.4400 | 0.1167 | 0.0195 |
| 8 | Belgium | 0.0855 | 1.8688 | 0.4873 | 0.0672 |
| 9 | Bolivia | 0.0892 | -0.8967 | -0.2338 | -0.0331 |
| 10 | Brazil | 0.0673 | 3.9403 | 1.0267 | 0.1232 |
| 11 | Canada | 0.1579 | -0.5431 | -0.1472 | -0.0288 |
| 12 | Chile | 0.0358 | -7.9383 | -2.1098 | -0.1752 |
| 13 | Taiwan | 0.0781 | 2.7697 | 0.7216 | 0.0944 |
| 14 | Colombia | 0.0578 | -1.8037 | -0.4632 | -0.0518 |
| 15 | Costa Rica | 0.0738 | 4.5380 | 1.1915 | 0.1497 |
| 16 | Denmark | 0.0626 | 5.3318 | 1.3997 | 0.1601 |
| 17 | Ecuador | 0.0633 | -2.8231 | -0.7298 | -0.0853 |
| 18 | Finland | 0.0800 | -0.7693 | -0.1996 | -0.0266 |
| 19 | France | 0.1277 | 1.7042 | 0.4548 | 0.0785 |
| 20 | Germany(F.R.) | 0.0830 | 0.4052 | 0.1053 | 0.0143 |
| 21 | Greece | 0.0892 | -2.4011 | -0.6285 | -0.0886 |
| 22 | Guatemala | 0.0613 | -3.4632 | -0.8970 | -0.1027 |
| 23 | Honduras | 0.0614 | 0.6798 | 0.1746 | 0.0202 |
| 24 | Iceland | 0.0660 | -6.7859 | -1.8101 | -0.2100 |
| 25 | India | 0.0674 | 1.0084 | 0.2599 | 0.0316 |
| 26 | Ireland | 0.1978 | 2.3775 | 0.6634 | 0.1482 |
| 27 | Italy | 0.0639 | 2.3959 | 0.6185 | 0.0728 |
| 28 | Japan | 0.1905 | 6.7575 | 1.9466 | 0.4098 |
| 29 | Korea | 0.2145 | -8.8809 | -2.6890 | -0.5889 (erroneous case) |
| 30 | Luxembourg | 0.0844 | -1.1961 | -0.3112 | -0.0427 |
| 31 | Malta | 0.0776 | 3.3097 | 0.8643 | 0.1124 |
| 32 | Norway | 0.0479 | -0.8891 | -0.2267 | -0.0230 |
| 33 | Netherlands | 0.0854 | 1.0047 | 0.2615 | 0.0361 |
| 34 | New Zealand | 0.0521 | 1.8907 | 0.4842 | 0.0512 |
| 35 | Nicaragua | 0.0508 | 0.3810 | 0.0973 | 0.0102 |
| 36 | Panama | 0.0400 | -3.3854 | -0.8666 | -0.0794 |
| 37 | Paraguay | 0.0665 | -5.7038 | -1.5055 | -0.1772 |
| 38 | Peru | 0.0651 | 6.3234 | 1.6777 | 0.1941 |
| 39 | Philippines | 0.0643 | 6.3450 | 1.6830 | 0.1935 |
| 40 | Portugal | 0.0930 | -0.2592 | -0.0677 | -0.0098 |
| 41 | South Africa | 0.0604 | 1.0649 | 0.2735 | 0.0313 |
| 42 | South Rhodesia | 0.1375 | 2.5845 | 0.6959 | 0.1250 |
| 43 | Spain | 0.0717 | -0.0340 | -0.0088 | -0.0011 |
| 44 | Sweden | 0.1226 | -4.5570 | -1.2305 | -0.2045 |
| 45 | Switzerland | 0.0733 | 2.6990 | 0.7011 | 0.0887 |
| 46 | Turkey | 0.0401 | -2.6072 | -0.6651 | -0.0612 |
| 47 | Tunisia | 0.0738 | -3.2187 | -0.8384 | -0.1062 |
| 48 | United Kingdom | 0.1155 | -2.9475 | -0.7849 | -0.1274 |
| 49 | United States | 0.3290 | -1.7225 | -0.5246 | -0.1656 |
| 50 | Venezuela | 0.0866 | 3.3732 | 0.8855 | 0.1222 |
| 51 | Zambia | 0.0636 | 10.0344 | 2.7991 | 0.3040 |
| 52 | Jamaica | 0.1437 | -3.2073 | -0.8694 | -0.1597 |
| 53 | Uruguay | 0.0915 | -1.5482 | -0.4047 | -0.0580 |
| 54 | Libya | 0.5351 | -3.6347 | -1.3531 | -0.6433 |
| 55 | B Malaysia | 0.0674 | -3.0572 | -0.7929 | -0.0957 |
| 56 | * | | | | |

7.4.3 Semiparametric smoothing

SMOOTH <data>, <series>, <smoothened series>, <width>

smooths <series> in <data> using a technique based on FFT (Fast Fourier Transformation) with a window whose full width is of order <width> neighboring points and saves the smoothened values as <smoothened series>.

SMOOTH can be used for smoothing any variable Y in relation to another variable X by first sorting the data by X and then applying SMOOTH for Y .

SMOOTH removes any linear trend and uses then FFT to low-pass filter the data. The linear trend is reinserted at the end. $\langle \text{width} \rangle$ gives the "amount of smoothing", specified as the number of points over which the data should be smoothed (not necessarily an integer). The results of SMOOTH are generally in accord with the notion "draw a smooth curve through these scattered points".

The method is described more precisely in "*Numerical Recipes*" (1987) by Press, Flannery, Teukolsky, and Vetterling.

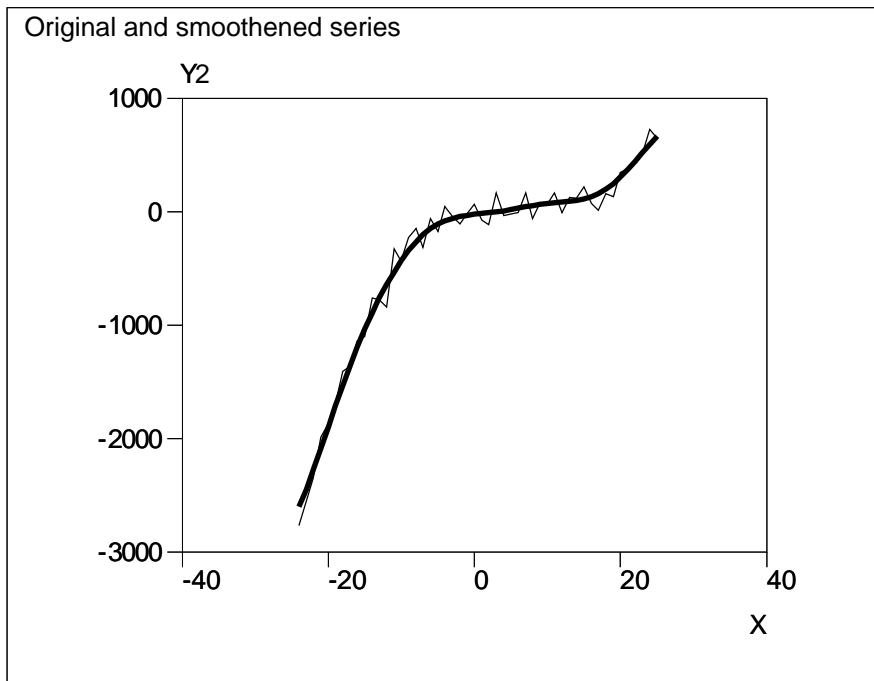
SMOOTH is especially suitable in curve fitting when the form of the model is not known. It competes amazingly well with parametric models, too. For example, in our example on stepwise regression, SMOOTH could be applied as follows:

```

19 1 SURVO 84C EDITOR Fri Apr 17 17:34:45 1992 D:\P2\STAT2\ 100 100 0
34 *
35 *GPLOT POLYREG,X,Y / LINE=1 OUTFILE=A
36 *
37 *VAR Y2=MISSING TO POLYREG
38 *SMOOTH POLYREG,Y,Y2,10
39 *GPLOT POLYREG,X,Y2 / LINE=[RED][line_width(1)],1 INFILE=A OUTFILE=B
40 *

```

The four activations in this scheme produce a graph showing the original series Y (as a function of X) and the smoothed series $Y2$:



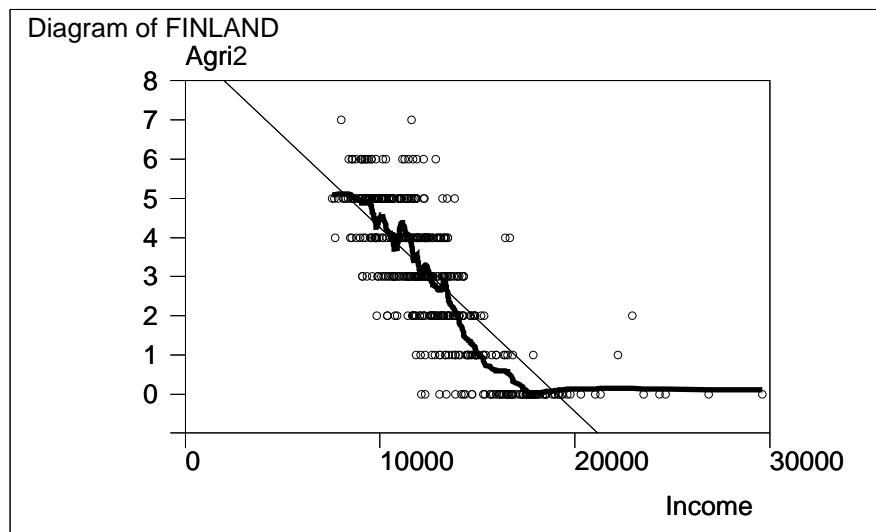
One can also try to plot the true cubic function (theoretical or estimated) in the same graph. The curve will be almost identical with this semiparametric fit.

As another example, we study the dependence between 'Income' (Income per inhabitant) and 'Agri' (population in agriculture in 10%) in the FINLAND data file. The next display shows a work scheme producing a graph where both a linear trend and the curve given by SMOOTH are plotted over the scatter plot.

```

25 1 SURVO 84C EDITOR Fri Apr 17 18:18:02 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *G PLOT FINLAND,Income,Agri / TREND=0
3 * YSCALE=-1(1)8 OUTFILE=A
4 *
5 *FILE SORT FINLAND BY Income TO FINL2
6 *VAR Agri2=MISSING TO FINL2
7 *SMOOTH FINL2,Agri,Agri2,25
8 *G PLOT FINL2,Income,Agri2_ / LINE=[line_width(5)],1
9 * YSCALE=-1(1)8 INFILE=A
10 *

```



One should observe the fact that the dependency is not linear and the trend line is biased. SMOOTH reveals that Agri decreases linearly only about to the point where Income=18000. Practically all communes with a higher personal income level are purely urban.

7.5 ESTIMATE operation

The **ESTIMATE** operation is a powerful tool for linear and nonlinear regression analysis. Also fairly general problems related to the maximum likelihood estimation can be solved by this operation. It permits the user to enter the model in normal mathematical notation. Before starting the computations, **ESTIMATE** analyzes the model function by evaluating its symbolic derivatives with respect to parameters to be estimated. The estimation and computation procedures are then selected according to the results of this analysis and on the basis of the user's specifications.

The data to be analyzed can be in any of the forms allowed by Survo. Since in true nonlinear cases the computing process is iterative and the data values have to be scanned repeatedly, **ESTIMATE** loads all active observations of variables appearing in the model to the central memory as a data matrix. In the current implementation, the maximum size of this data matrix is 8192 elements.

7.5.1 Simple example

The idea and use of **ESTIMATE** in standard applications is best described through an example. In the following display, a small data set and a linear regression model are presented in the form required by **ESTIMATE**.

```

28 1 SURVO 84C EDITOR Fri Mar 20 18:07:13 1987 D:\P2\STAT2\ 150 80 0
1 *
2 *DATA COUNTRIES,A,B,N
3 N Country      Coffee  Tea    Beer  Wine  Spirits
4 A Finland      12.5   0.15  54.7  7.6   2.7
5 * Sweden       12.9   0.30  58.3  7.9   2.9
6 * Norway       9.4    0.19  43.5  3.1   1.8
7 * Denmark      11.8   0.41  113.9 10.4  1.7
8 * England      1.8    3.49  113.7  5.1   1.4
9 * Ireland      0.2    3.73  124.5  3.8   1.9
10 * Holland      9.2    0.58  75.5  9.7   2.7
11 * Switzerland  9.1    0.25  73.5  44.9  2.1
12 * France       5.2    0.10  44.5  104.3 2.5
13 * Italy         3.6    0.06  13.6  106.6 2.0
14 * Spain        2.5    0.03  43.6  73.2  2.7
15 B Portugal     2.2    0.03  27.5  89.3  0.9
16 *
17 *MODEL Beer1
18 *log(Beer)=constant+coeff*log(Tea)
19 *
20 *ESTIMATE COUNTRIES,Beer1,21
21 *
22 *
23 *

```

When the **ESTIMATE** operation on line 20 is activated, it will produce the following results from line 21 onwards:

```

28 1 SURVO 84C EDITOR Fri Mar 20 18:07:18 1987 D:\P2\STAT2\ 150 80 0
16 *
17 *MODEL Beer1
18 *log(Beer)=constant+coeff*log(Tea)
19 *
20 *ESTIMATE COUNTRIES,Beer1,21_
21 *Estimated parameters of model Beer1:
22 *constant=4.488964 (0.156575)
23 *coeff=0.327629 (0.075271)
24 *n=12 rss=1.553654 R^2=0.65453 nf=7
25 *Correlations:
26 *      consta  coeff
27 * constant    1.000  0.687
28 *  coeff      0.687  1.000
29 *

```

During the activation, a temporary message

Model is linear with respect to parameters!

Optimization by the Newton-Raphson algorithm

Iteration 1:

appears on the screen telling about **ESTIMATE**'s decision on the basis of the formal analysis. In this linear case, the results are obtained in one iteration by the Newton-Raphson method. This corresponds to the standard solution of the normal equations. Among the results, the number of object function (sum of squared residuals) and derivative evaluations are given as **nf=7** on line 24.

Let us now describe more accurately what happened after **ESTIMATE** on line 20 was activated. In this operation we have three parameters: the name of the data set (**COUNTRIES**), the name of the model (**Beer1**), and the first line for the results (21).

As a data set, any data list or data file (limited by **CASES**, **IND** and **SELECT** specifications) could be used as well.

The model to be estimated has to be given by a **MODEL** definition. In this case, the model **Beer1** is defined on lines 17 and 18. The second line yields the model in the form **<regressand>=<model function>**. The model function is written according to mathematical notation similar to that used in the **VAR** operation. Operands and arguments in the model notation are either variables appearing in the data set or parameters to be estimated. Thus, if a certain word is not recognized as a variable, it is treated as a parameter.

In addition to **+**, **-**, *****, **/** and **^** operations, functions **sqrt**, **log**, **abs**, **int**, **exp**, **sin**, **cos**, **tan** and **arctan** are available.

'**Beer**' and '**Tea**' are considered in our example as variables and '**constant**' and '**coeff**' as parameters to be estimated.

The regressand, as **log(Beer)** here, may also be an expression of one or more variables. It is written according to the same rules as the model function. Also the regressand may include parameters to be estimated; see 7.5.10.

In the previous example, the model is converted by **ESTIMATE** to the form $\log(X1)=A1+A2*\log(X2)$.

Actually, a still more compact representation is selected, but the narration above gives the idea. The indices for variables (X) and parameters (A) are selected in the order of appearance.

After interpretation and notational transformation, the first and second partial derivatives with respect to parameters are evaluated and represented in the same notation as the model itself.

In our example, the first derivatives are 1 and $\log(X2)$. The second derivatives vanish and this clearly indicates that the model is linear with respect to parameters.

In order to speed up numerical computation of the model and derivatives, all the expressions are finally converted into an inverted Polish notation. A special routine will maintain the stack when evaluating numeric expressions.

7.5.2 Analytical derivatives

The procedure of forming analytical derivatives is recursive. Each step in this procedure consists of splitting the function to be analyzed into two parts like sum, difference, product or ratio of two functions and then applying basic derivation rules to the partition obtained. For example, to form the derivative of $f(x)=(x+a)*\log(x^2)$ with respect to x , the function $f(x)$ is interpreted as $f(x)=r(x)*s(x)$ where $r(x)=x+a$ and $s(x)=\log(x^2)$. In order to apply the derivation rule of a product, the derivatives of $r(x)$ and $s(x)$ are needed. In this case $r(x)=x+a$ is interpreted as sum of x and a . Similarly the derivative of $s(x)=\log(x^2)$ is obtained by the formula $D(\log(g(x)))=1/g(x)*D(g(x))$ thus requiring the derivative of $g(x)=x^2$ etc.

Each branch in this recursive process leads finally to a case where the derivative of x itself or a constant is needed. By combining these elementary derivatives by standard rules, the sought derivative will be obtained.

The formal derivation algorithm is automatically employed by the **ESTIMATE** operation. However, if the user wants to have analytical derivatives in the edit field, it is possible by the **DER** operation. **DER** employs the same algorithm as **ESTIMATE**. In the next exhibit, some applications of **DER** are given.

```

12 1 SURVO 84C EDITOR Sat Mar 21 09:14:23 1987 D:\P2\STAT2\ 120 80 0
1 *
2 *DER F(X)*sin(X) X
3 * Derivative of F(X)*sin(X)
4 * with respect to X is
5 * F'(X)*sin(X)+cos(X)*F(X)
6 *
7 *DER (x+a)^n x
8 * Derivative of (x+a)^n
9 * with respect to x is
10 * n*(x+a)^(n-1)
11 *
12 *DER (x+a)*log(x^2) x
13 * Derivative of (x+a)*log(x^2)
14 * with respect to x is
15 * log(x^2)+2*x/x^2*(x+a)
16 *
17 *DER X^X^X X
18 * Derivative of X^X^X
19 * with respect to X is
20 * X^X^X*(log(X^X)+X^X*(log(X)+1/X*X)/X^X*X)
21 *
22 *
23 *


```

The DER operation does not necessarily find the simplest possible representation of the derivative, but it is seldom any problem in applications. The great advantage is that the user is freed from the painstaking effort of giving the derivatives. If erroneous derivatives were entered, the optimization process of ESTIMATE would be seriously hurt.

7.5.3 Computational methods

The main estimation procedure of ESTIMATE is the ordinary least squares (OLS) method. (For alternatives see 7.5.7)

The iterative numerical algorithm needed for minimizing the residual sum of squares is selected by the user by the specification METHOD. We have three main alternatives:

| | | |
|----------|-------------------------|---|
| METHOD=N | Newton-Raphson |  |
| METHOD=D | Davidon-Fletcher-Powell | |
| METHOD=H | Hooke-Jeeves | |

See e.g. Walsh, G.R. (1975), *Methods of Optimization*, Wiley, New York.

If the METHOD specification is missing (as in our first example), ESTIMATE selects the computational algorithm according to the type of the model.

The Newton-Raphson method finds the optimum for a quadratic objective function (i.e. for a linear model) in one iteration round. This corresponds exactly to the conventional procedure of solving linear normal equations. Thus, when the second derivatives of the model function vanish, the model must be linear. Then ESTIMATE selects the Newton-Raphson method and performs one iteration only.

In genuine nonlinear cases (where at least one of the second derivatives is not zero) METHOD=D is selected. Although the Newton-Raphson is most efficient when it works, it is unreliable in more complicated models especially

when the initial values of parameters are poor. The Davidon-Fletcher-Powell (DFP) variable metric method seems to be one of the best numerical procedures for general unconstrained optimization. Furthermore, in linear models the DFP method will normally converge in m iterations where m is the number of parameters to be estimated.

The simple direct search method of Hooke and Jeeves (selected by `METHOD=H`) can be used for improving initial estimates and for very irregular models (e.g. when the model function is not differentiable). In this case, no analytical derivatives are formed.

In complicated models, the expressions of derivatives and especially of second derivatives can be very long and take much memory. To save space and to support larger models, the DFP method can also be selected by `METHOD=d`. Then no second derivatives are evaluated analytically.

In those cases where second derivatives are omitted (`METHOD=H` and `METHOD=d`) the approximate standard errors and correlations of estimated parameters will be computed by finite-difference approximations.

In more difficult problems, the initial step length used in one-dimensional searches should be selected carefully. A too-large step length can exceed the admissible region, and a too-short step length may slow down the optimization process or lead to a premature termination of the process.

If `METHOD=D` or `METHOD=H` is in use, the initial step length may be selected by `STEP=<step length>`. `STEP=1` is the default. If `METHOD=H` is used, the `STEP` specification may also be given in an augmented form `STEP=<initial step> , <final step>`. The default values correspond to `STEP=1, 0.00001`.

In the Newton-Raphson method (`METHOD=N`), the step length is automatically controlled and `STEP` has no effect.

7.5.4 Initial estimates

Initial values for the parameters to be estimated are not needed at all when dealing with linear models (by default `METHOD=N`). In nonlinear cases, however, good approximations for the final estimates are valuable.

Default for each initial value is always 0. Other initial values can be suggested simply by giving specifications of the form

`<parameter>=<initial value>`

in the edit field. Since the results are displayed in the same form, results from previous estimation attempts can be directly employed as starting values for subsequent `ESTIMATE` activations.

For example, in our first example we could generalize the model on line 18 to the following nonlinear form:

```

28 1 SURVO 84C EDITOR Sat Mar 21 12:43:49 1987 D:\P2\STAT2\ 120 80 0
16 *
17 *MODEL Beer1
18 *log(Beer)=constant+coeff*log(Tea+C*Coffee)
19 *METHOD=N
20 *ESTIMATE COUNTRIES,Beer1,21_
21 *Estimated parameters of model Beer1:
22 *constant=4.488964 (0.156575)
23 *coeff=0.327629 (0.075271)
24 *n=12 rss=1.553654 R^2=0.65453 nf=7
25 *Correlations:
26 *          consta  coeff
27 * constant    1.000  0.687
28 *  coeff      0.687  1.000
29 *

```

If `ESTIMATE` on line 20 is reactivated, the present values of 'constant' and 'coeff' on lines 22 and 23 will be used as initial estimates and since no initial value for the new parameter `C` is given, `C=0` will be used as such. Observe also that we have inserted `METHOD=N` on line 19 thus requiring the Newton-Raphson method still be used although the model is not linear anymore.

After 6 iterations, the previous results will be overwritten by

```

28 1 SURVO 84C EDITOR Sat Mar 21 12:43:49 1987 D:\P2\STAT2\ 120 80 0
16 *
17 *MODEL Beer1
18 *log(Beer)=constant+coeff*log(Tea+C*Coffee)
19 *METHOD=N
20 *ESTIMATE COUNTRIES,Beer1,21_
21 *Estimated parameters of model Beer1:
22 *constant=4.163718 (0.507366)
23 *coeff=0.518345 (0.250795)
24 *C=0.060901 (0.105947)
25 *n=12 rss=1.488258 R^2=0.66907 nf=84
26 *Correlations:
27 *          consta  coeff      C
28 * constant    1.000 -0.817 -0.971
29 *  coeff      -0.817  1.000  0.875
30 *  C          -0.971  0.875  1.000
31 *

```


7.5.5 Constants in the model

Numerical constants can, of course, be written as numbers in the model expression. Sometimes, however, it is useful to have symbolic notations, too. In that case, the value of the constant should be entered in the edit field as #<name of constant>=<value>.

Character '#' in front of the name of the constant indicates that a fixed value and not a parameter to be estimated is given. #pi=3.14159265 and #mean=370.23333 are examples of such constants. In the model expressions, the symbolic constants appear without any preceding '#'.

By using this facility, it is easy to fix and release any parameter in the model without changing the model notation.

In the next example, it is shown how to compute a mean of a variable and how to use it as a symbolic constant in another model. We are still using the data set COUNTRIES.

Now, in order to compute the arithmetic mean (and the standard deviation) of 'Tea', we enter the following model and ESTIMATE operation. Of course, Survo affords several other tools for the same task.

```

25 1 SURVO 84C EDITOR Sat Mar 21 13:24:34 1987 D:\P2\STAT2\ 120 80 0
32 *.....
33 *
34 *MODEL A1
35 *Tea=Tmean
36 *ESTIMATE COUNTRIES,A1,37
37 *Estimated parameters of model A1:
38 *Tmean=0.776667 (0.385194)
39 *n=12 rss=19.585467 R^2=0.00000 nf=4
40 *

```

Since the sum of $(Tea - Tmean)^2$ is minimized with respect to 'Tmean' when 'Tmean' is the arithmetic mean of the variable 'Tea', the model A1 'Tea=Tmean' can be used in ESTIMATE to get the desired result. In addition to mean (0.776667), we obtain the standard deviation of 'Tea' as the standard error of the estimate (0.385194).

To compute a quadratic model for 'Beer' with 'Tea' as the sole regressor, we introduce another model A2, where 'Tea' appears in a centered form 'Tea-Tmean'. To have 'Tmean' as a constant in this model, we insert '#' in front of 'Tmean' on line 38. Activation of ESTIMATE on line 43 then leads to the result:

```

25 1 SURVO 84C EDITOR Sat Mar 21 13:43:28 1987 D:\P2\STAT2\ 120 80 0
32 *.....
33 *
34 *MODEL A1
35 *Tea=Tmean
36 *ESTIMATE COUNTRIES,A1,37
37 *Estimated parameters of model A1:
38 *#Tmean=0.776667 (0.385194)
39 *n=12 rss=19.585467 R^2=0.00000 nf=4
40 *
41 *MODEL A2
42 *Beer=a+b*(Tea-Tmean)+c*(Tea-Tmean)^2
43 *ESTIMATE COUNTRIES,A2,44
44 *Estimated parameters of model A2:
45 *a=111.309375 (17.624)
46 *b=82.639477 (23.188702)
47 *c=-28.026505 (10.271105)
48 *n=12 rss=3195.017785 R^2=0.77213 nf=11
49 *Correlations:
50 *
51 * a          a          b          c
52 * b          1.000    0.935   -0.951
53 * c          0.935    1.000   -0.983
54 *          -0.951   -0.983    1.000

```

7.5.6 Weighting of observations

The observations can be weighted by writing a **WEIGHT** specification **WEIGHT=<weight function>**

where the weight function is a function of any variables appearing in the current data. Typically the weight is simply a variable. If **WEIGHT** is not given, **WEIGHT=1** serves as default. The weight function is expressed according to the same rules as the model function, but no unknown parameters are allowed.

Using **WEIGHT**, it is possible to estimate a model of the general type

$$g(\mathbf{X}, \mathbf{A}) = f(\mathbf{X}, \mathbf{A}) + \text{eps}/\sqrt{w(\mathbf{X})}$$

where \mathbf{X} and \mathbf{A} are variables and parameters, respectively,

| | |
|-----------------------------|--|
| $g(\mathbf{X}, \mathbf{A})$ | is the regressand (function), |
| $f(\mathbf{X}, \mathbf{A})$ | is the model function (regressor function), |
| $w(\mathbf{X})$ | is the weight function, |
| eps | is a normal error term with zero mean and unknown constant variance. |

To specify this kind of a model for the **ESTIMATE** operation, we define **MODEL** in the form $g(\mathbf{X}, \mathbf{A})=f(\mathbf{X}, \mathbf{A})$ and enter **WEIGHT=w(X)**.

If $g(\mathbf{X}, \mathbf{A})$ is independent of \mathbf{A} (this is the normal case), we obtain maximum likelihood estimates for parameters \mathbf{A} by the standard OLS criterion when the observations are independent.

If $g(\mathbf{X}, \mathbf{A})$ depends on \mathbf{A} , the estimation procedure does not take into account the Jacobian of the g transformation (see 7.5.10). To guarantee that the optimization problem is well-defined, the model is to be formulated so that the regressand is not too heavily dependent on \mathbf{A} .

To demonstrate application of `ESTIMATE` with a weight function, we make a simulation experiment. In the next display, 20 independent observations are generated according to model

$$Y = a + b \cdot \sin(c \cdot t) + \sqrt{t} \cdot \text{eps},$$

where $t=1, 2, \dots, 20$, $a=100$, $b=10$, $c=0.1$ and $\text{eps}=0.3 \cdot N(0,1)$. This is done by a VAR operation as follows:

```

39 1 SURVO 84C EDITOR Sat Apr 18 18:40:04 1992      D:\P2\STAT2\ 100 100 0
1 *
2 *VAR Y=a+b*sin(c*t)+sqrt(t)*eps TO TEST_
3 *   a=100 b=10 c=0.1 eps=0.3*probit(rnd(6))
4 *
5 *DATA TEST X,Y,Z,M
6 M AA  123.123
7 Z t    Y
8 X 1   100.750
9 * 2   102.581
10 * 3   102.930
11 * 4   103.295
12 * 5   103.847
13 * 6   106.334
14 * 7   106.341
15 * 8   106.829
16 * 9   108.618
17 * 10  107.438
18 * 11  110.013
19 * 12  107.918
20 * 13  107.720
21 * 14  110.411
22 * 15  110.455
23 * 16  108.913
24 * 17  108.765
25 * 18  109.504
26 * 19  110.367
27 Y 20  109.916
28 *

```

Using this artificial data set, we have tried to estimate the same model first without weighting the observations (lines 30-34 in the next display) and then by entering the correct weight function `WEIGHT=1/t` (lines 47-49):

```

25 1 SURVO 84C EDITOR Sat Apr 18 18:41:36 1992 D:\P2\STAT2\ 100 100 0
29 *.....
30 *a=101 b=9 c=0.11
31 *MODEL TRIG
32 *Y=a+b*sin(c*t)
33 *
34 *ESTIMATE TEST,TRIG,CUR+1
35 *Estimated parameters of model TRIG:
36 *a=100.318 (0.608228)
37 *b=9.53044 (0.712622)
38 *c=0.0938233 (0.00653369)
39 *n=20 rss=13.802410 R^2=0.91833 nf=92
40 *Correlations:
41 *
42 * a          a          b          c
43 * a          1.000    -0.907   -0.480
44 * b          -0.907    1.000    0.254
45 * c          -0.480    0.254    1.000
46 *.....
47 *a=101 b=9 c=0.11
48 *WEIGHT=1/t
49 *ESTIMATE TEST,TRIG,CUR+1
50 *Estimated parameters of model TRIG:
51 *a=100.043 (0.291701)
52 *b=9.8053 (0.491063)
53 *c=0.0970263 (0.00669982)
54 *n=20 rss=8.386239 R^2=0.96010 nf=96
55 *Correlations:
56 *
57 * a          a          b          c
58 * a          1.000   -0.637   -0.443
59 * b          -0.637    1.000   -0.159
60 * c          -0.443   -0.159    1.000

```

We see that correct weighting gives estimates with smaller standard errors.

7.5.7 Estimation criteria

The normal estimation criterion in the ESTIMATE operation is ordinary least squares (OLS). In our general setup model $g(\mathbf{X}, \mathbf{A}) = f(\mathbf{X}, \mathbf{A})$, weight $w(\mathbf{X})$ it implies minimization of sum of weighted squared deviations

$$w(\mathbf{X}) * (g(\mathbf{X}, \mathbf{A}) - f(\mathbf{X}, \mathbf{A}))^2$$

with respect to parameters \mathbf{A} .

By giving an extra specification CRITERION=L_p where p is any positive constant the estimates will be obtained by minimizing the sum of errors of type

$$w(\mathbf{X}) * \text{abs}(g(\mathbf{X}, \mathbf{A}) - f(\mathbf{X}, \mathbf{A}))^p.$$

CRITERION=L2 is default and corresponds to OLS.

CRITERION=L1 can also be given as CRITERION=ABS or MAD and it implies the sum of absolute deviations to be minimized.

The influence of the criterion selected is illustrated in the next display where a simple data set having a "serious outlier" is analyzed by the model $Y = a * X + b$ (true $a = 1$) and by using the L2 and L1 norms. In the latter case, the Hooke-Jeeves method is the only possible one because the objective function is not differentiable.

```

19 1 SURVO 84C EDITOR Sun Apr 19 09:15:14 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *MODEL YX
3 *Y=a*X+b
4 *DATA LIN:(X,Y) 1,1 2,2 3,3 4,4 5,7 6,6 7,7 8,8 9,9 10,10 END
5 *
6 *ESTIMATE LIN,YX,7
7 *Estimated parameters of model YX:
8 *a=0.987879 (0.07373)
9 *b=0.266667 (0.457485)
10 *n=10 rss=3.587879 R^2=0.95734 nf=6
11 *Correlations:
12 *
13 * a          a          b
14 * b          -0.886    1.000
15 *.....
16 *a=0.98 b=0.3
17 *CRITERION=L1 METHOD=H STEP=0.1,0.00000001
18 *ESTIMATE LIN,YX,19
19 *Estimated parameters of model YX:
20 *a=1
21 *b=4.76837E-008
22 *n=10 Norm L1: min=2.000000 nf=255
23 *

```

7.5.8 Residuals and predicted values

The residuals $g(\mathbf{X}, \mathbf{A}) - f(\mathbf{X}, \mathbf{A})$ and the predicted values $g(\mathbf{X}, \mathbf{A})$, $f(\mathbf{X}, \mathbf{A})$ can be computed as values of selected variables during the ESTIMATE operation.

The variables are selected for these tasks by optional masks

- R residuals $g(\mathbf{X}, \mathbf{A}) - f(\mathbf{X}, \mathbf{A})$
- G predicted values $g(\mathbf{X}, \mathbf{A})$
- F predicted values $f(\mathbf{X}, \mathbf{A})$

If the data set is located in the edit field as a data table, the extended form with a mask line must be used.

In the next display, the first example is reconsidered with the above options. All items written by ESTIMATE are shaded.

```

28 1 SURVO 84C EDITOR Sat Mar 21 18:10:33 1987 D:\P2\STAT2\ 120 80 0
1 *
2 *DATA COUNTRIES,A,B,N,M
3 M AAAAAAAAAA AAAA AAAA AAAAA AAAAA AAA RR.RR G.GG F.FF
4 N Country Coffee Tea Beer Wine Spirits RES PRED logB
5 A Finland 12.5 0.15 54.7 7.6 2.7 0.13 4.00 3.87
6 * Sweden 12.9 0.30 58.3 7.9 2.9 -0.03 4.07 4.09
7 * Norway 9.4 0.19 43.5 3.1 1.8 -0.17 3.77 3.94
8 * Denmark 11.8 0.41 113.9 10.4 1.7 0.54 4.74 4.20
9 * England 1.8 3.49 113.7 5.1 1.4 -0.16 4.73 4.90
10 * Ireland 0.2 3.73 124.5 3.8 1.9 -0.10 4.82 4.92
11 * Holland 9.2 0.58 75.5 9.7 2.7 0.01 4.32 4.31
12 * Switzerland 9.1 0.25 73.5 44.9 2.1 0.26 4.30 4.03
13 * France 5.2 0.10 44.5 104.3 2.5 0.06 3.80 3.73
14 * Italy 3.6 0.06 13.6 106.6 2.0 -0.96 2.61 3.57
15 * Spain 2.5 0.03 43.6 73.2 2.7 0.43 3.78 3.34
16 B Portugal 2.2 0.03 27.5 89.3 0.9 -0.03 3.31 3.34
17 *
18 *MODEL Beer1
19 *log(Beer)=constant+coeff*log(Tea)
20 *
21 *ESTIMATE COUNTRIES,Beer1,22_
22 *Estimated parameters of model Beer1:
23 *constant=4.488964 (0.156575)
24 *coeff=0.327629 (0.075271)
25 *n=12 rss=1.553654 R^2=0.65453 nf=6
26 *Correlations:
27 * consta coeff
28 * constant 1.000 0.687
29 * coeff 0.687 1.000
30 *

```

7.5.9 Maximum likelihood estimates

In addition to nonlinear regression, the `ESTIMATE` operation enables computation of maximum likelihood estimates for a user-defined univariate distribution. In this case, the `MODEL` definition is to be written in the form

```

MODEL <name of model>
LOGDENSITY=<logarithm of the density function>

```

The logarithm of the density function of a single observation has to be given. It is assumed that the data set is a random sample of the distribution in question.

In all other respects, `ESTIMATE` is used in the same way as in regression analysis.

As an example we try again to estimate the model

$$\log(\text{Beer}) = \text{constant} + \text{coeff} * \log(\text{Tea}) + \text{eps}$$

of our first example but rewrite it as a maximum likelihood problem for $\log(\text{Beer})$.

We make the standard assumption that the error term 'eps' is normally distributed with zero mean and unknown constant variance (denoted by 'var' in sequel).

In order to use `ESTIMATE`, we shall enter the logdensity of a normal distribution for 'log(Beer)' with mean 'constant+coeff*log(Tea)' and variance 'var'. This is expressed as model `NORMAL` on lines 17-18 in the next exhibit.

Since 'var' is a nuisance parameter (even for computational reasons), it is best to start the estimation by keeping it constant and setting #var=0.1 (on line 19).

After ESTIMATE on line 20 has been activated, we shall have the following display where the estimates obtained for 'constant' and 'coeff' are correct (due to the form of the normal density) but their standard errors are not.

```

29 1 SURVO 84C EDITOR Sat Mar 21 18:47:43 1987 D:\P2\STAT2\ 120 80 0
1 *
2 *DATA COUNTRIES,A,B,N
3 N Country Coffee Tea Beer Wine Spirits
4 A Finland 12.5 0.15 54.7 7.6 2.7
5 * Sweden 12.9 0.30 58.3 7.9 2.9
6 * Norway 9.4 0.19 43.5 3.1 1.8
7 * Denmark 11.8 0.41 113.9 10.4 1.7
8 * England 1.8 3.49 113.7 5.1 1.4
9 * Ireland 0.2 3.73 124.5 3.8 1.9
10 * Holland 9.2 0.58 75.5 9.7 2.7
11 * Switzerland 9.1 0.25 73.5 44.9 2.1
12 * France 5.2 0.10 44.5 104.3 2.5
13 * Italy 3.6 0.06 13.6 106.6 2.0
14 * Spain 2.5 0.03 43.6 73.2 2.7
15 B Portugal 2.2 0.03 27.5 89.3 0.9
16 *
17 *MODEL NORMAL
18 *LOGDENSITY=-0.5*((log(Beer)-constant-coeff*log(Tea))^2/var+log(var))
19 *METHOD=N #var=0.1
20 *ESTIMATE COUNTRIES,NORMAL,21
21 *Estimated parameters of model NORMAL:
22 *constant=4.488964 (0.125616)
23 *coeff=0.327629 (0.060388)
24 *n=12 log(L)=6.047239 nf=18
25 *Correlations:
26 *
27 * consta coeff
28 * constant 1.000 0.687
29 * coeff 0.687 1.000
29 *

```

To obtain an estimate for residual variance, we release 'var' by deleting '#' from line 19. When ESTIMATE is reactivated, it uses the obtained estimates for 'constant' and 'coeff' as starting values and overwrites the former results by

```

29 1 SURVO 84C EDITOR Sat Mar 21 19:08:44 1987 D:\P2\STAT2\ 120 80 0
16 *
17 *MODEL Beer1
18 *LOGDENSITY=-0.5*((log(Beer)-constant-coeff*log(Tea))^2/var+log(var))
19 *METHOD=N var=0.1
20 *ESTIMATE COUNTRIES,NORMAL,21
21 *Estimated parameters of model NORMAL:
22 *constant=4.488964 (0.142933)
23 *coeff=0.327629 (0.068713)
24 *var=0.129471 (0.052856)
25 *n=12 log(L)=6.265782 nf=70
26 *Correlations:
27 * consta coeff var
28 * constant 1.000 0.687 -0.000
29 * coeff 0.687 1.000 0.000
30 * var -0.000 0.000 1.000
31 *

```

As expected, the estimate of 'var' is uncorrelated with estimates of 'constant' and 'coeff'.

7.5.10 Special applications



ESTIMATE?

As mentioned earlier, the regressand function in **ESTIMATE** models can also include parameters to be estimated. However, the estimates obtained in this case (by weighted OLS) are not anymore ML estimates.

As a simple example we consider a model $(X-a)^2=b$ where X is a variable and a, b are parameters to be estimated. It is natural to expect that a is close to the mean of X and b is close to the variance of X . We apply this model to **COUNTRIES** by selecting 'Beer' as X .

```

25 1 SURVO 84C EDITOR Sun Apr 19 11:23:17 1992 D:\P2\STAT2\ 100 100 0
16 *
17 *MODEL ab
18 *(Beer-a)^2=b
19 *
20 *ESTIMATE COUNTRIES,ab,21
21 *Estimated parameters of model ab:
22 *a=72.796 (4.93548)
23 *b=1220.72 (344.901)
24 *n=12 rss=13663151.058956 nf=69
25 *Correlations:
26 *
27 * a          a      b
28 * b          0.207  1.000
29 *

```

To compare the results obtained with the true mean and variance of 'Beer', the latter statistics are computed by a **STAT** operation.

```

18 1 SURVO 84C EDITOR Sun Apr 19 11:24:48 1992 D:\P2\STAT2\ 100 100 0
29 *
30 *STAT COUNTRIES,31_ / VARS=Beer RESULTS=0
31 *Basic statistics: COUNTRIES N=12
32 *Variable: Beer
33 *min=13.6      in obs.#10 (Italy)
34 *max=124.5    in obs.#6 (Ireland)
35 *mean=65.56667 stddev=35.70262 skewness=0.404974 kurtosis=-1.154864
36 *autocorrelation=0.5985
37 *lower_Q=36.66667 median=56.66667 upper_Q=106.6667
38 *
39 *          stddev^2=1274.6770748644
40 *
41 *          mean+0.5*skewness*stddev=72.79598641594
42 *

```

It is seen that estimates for a and b do not match exactly to 'mean' (on line 35) and 'stddev^2' (computed afterwards on line 39). It can be shown that the OLS principle in this case leads to an estimate $a = \text{mean} + 0.5 * \text{skewness} * \text{stddev}$ and this fact is demonstrated on line 41 by editorial computing.

Box-Cox transformation

As another example, we shall study the effect of the Box-Cox power transformation in a certain special case where it is assumed that the model

$$(Y^c - 1) / c = aX + b + \text{eps}$$

is valid for an unknown value c . An artificial data set of 40 observations with $X=1,2, \dots, 40$, $a=-0.2$, $b=3$, $\text{eps}=0.3*N(0,1)$ and $c=0$

(i.e. $\log(Y) = aX + b + \text{eps}$) is generated as follows:

```

45 1 SURVO 84C EDITOR Sun Apr 19 11:45:18 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *FILE CREATE XY,8,2,64,7,40
3 *FIELDS:
4 *1 N 4 X
5 *2 N 4 Y
6 *END
7 *
8 *VAR X=ORDER TO XY
9 *VAR Y=exp(-0.2*X+3+0.3*probit(rnd(1))) TO XY
10 *

```

Now we estimate all three parameters, namely, the regression coefficients a, b and the power c of the Box-Cox transformation simultaneously. The initial estimate for c is 1.

```

20 1 SURVO 84C EDITOR Sun Apr 19 11:48:04 1992 D:\P2\STAT2\ 100 100 0
10 *
11 *MODEL TEST
12 *(Y^c-1)/c=a*X+b
13 *
14 * c=1 METHOD=N
15 *ESTIMATE XY,TEST,16
16 *Estimated parameters of model TEST:
17 *c=-0.000440942 (0.0234791)
18 *a=-0.199391 (0.0070945)
19 *b=2.90643 (0.112989)
20 *n=40 rss=4.216477 R^2=0.98049 nf=124
21 *Correlations:
22 *
23 * c          c          a          b
24 * a          0.758     1.000    -0.752
25 * b         -0.270    -0.752     1.000
26 *

```

Estimation of a circle

In this example, the location and radius of a circle are estimated from an artificial data set. This experiment (with time-dependent randomization of data) is available in Survo as a sacro /CIRCLE. In fact, the following listing is simply a realization of /CIRCLE with a specific seed number 1 for the random number generator. The listing should speak for itself:

```

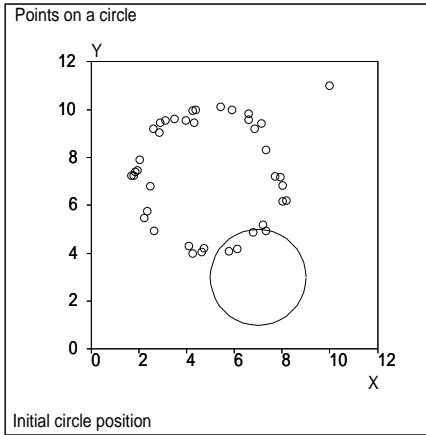
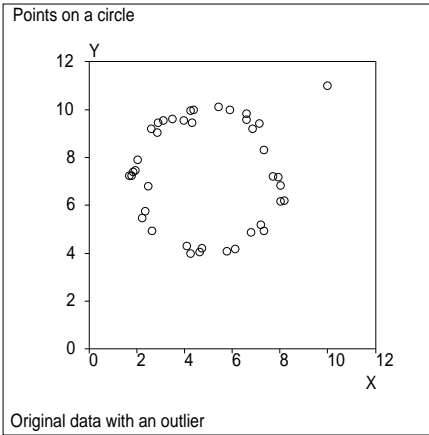
18 1 SURVO 84C EDITOR Fri Jul 03 17:41:48 1992      D:\P2\STAT2\ 180 100 0
1 *
2 *   Estimation of a circle                        *GLOBAL* parameters
3 *                                               MODE=VGA SIZE=479,479
4 *
5 *In this example we generate a sample of points (x,y) located
6 *roughly on the circumference of the circle
7 *   (x-X0)^2+(y-Y0)^2=R^2
8 *or
9 *   x=X0+R*cos(T), y=Y0+R*sin(T).
10 *
11 *The sample will also include one outlier and we shall investigate
12 *the influence of this outlier in parameter estimation.
13 *
14 *   Generating the sample
15 *
16 *We assume that X=X0+R*cos(T)+eps1 and Y=Y0+R*sin(T)+eps2 where
17 *the radius is R=3 and the coordinates of the center are X0=5 and Y0=7.
18 *The error terms are eps1=0.2*probit(rnd(1)) and eps2=0.2*probit(rnd(1))
19 *i.e. they both have a N(0,0.2^2) distribution.
20 *To scatter the points randomly around the center we assume that
21 *T has a uniform distribution, i.e. T=2*pi*rnd(1) where pi=3.14159265.
22 *
23 *Before generating sample values according to these specifications
24 *we create a data file CIRCLE for 40 observations:
25 *FILE CREATE CIRCLE,8,2,64,7,40
26 *FIELDS:
27 *1 N 4 X
28 *2 N 4 Y
29 *END
30 *
31 *Now 40 observations (X,Y) will be produced by the VAR operation:
32 *VAR X,Y TO CIRCLE
33 *

```

```

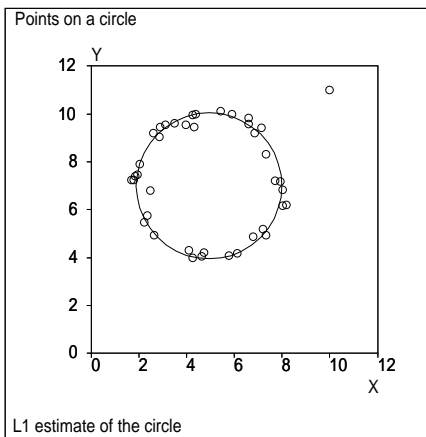
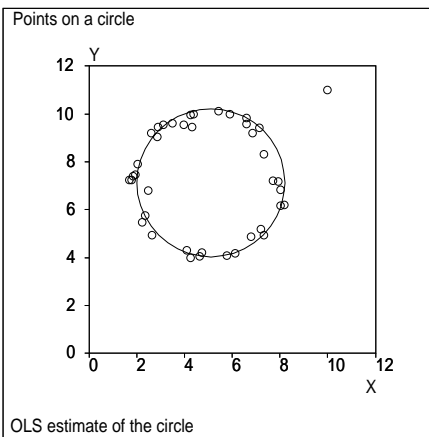
17 1 SURVO 84C EDITOR Fri Jul 03 17:44:32 1992      D:\P2\STAT2\ 180 100 0
33 *
34 *The sample is plotted by
35 *GPLOT CIRCLE,X,Y / SCALE=0(2)12
36 *   HEADER=Points_on_a_circle
37 *
38 *The last observation will be replaced by an outlier X=10, Y=11:
39 *FILE SHOW CIRCLE
40 *and the sample is plotted again:  OUTFILE=A (save picture in file A.SPX)
41 *.....
42 *   OLS estimation of the circle
43 *
44 *We now 'forget' the true values of parameters R,X0,Y0 and try to
45 *estimate them from the sample generated above.
46 *Estimation will be started from poor initial values R=2 X0=7 and Y0=3.
47 *The corresponding circle is plotted by
48 *GPLOT X(T)=X0+R*cos(T), Y(T)=Y0+R*sin(T) / T=0,2*pi,pi/20 INFILE=A
49 *   OUTFILE=A

```



```

41 1 SURVO 84C EDITOR Fri Jul 03 17:49:51 1992      D:\P2\STAT2\ 180 100 0
49 *
50 *The model to be estimated is defined by
51 *MODEL Cmodel
52 *sqrt((X-X0)^2+(Y-Y0)^2)=R
53 *
54 *To estimate parameters of this model from the data set CIRCLE
55 *we activate the following ESTIMATE operation:
56 *ESTIMATE CIRCLE,Cmodel,CUR+2
57 *
58 *.....
58 *Estimated parameters of model Cmodel:
59 *X0=5.11165 (0.130129)
60 *Y0=7.13113 (0.131058)
61 *R=3.09286 (0.09169)
62 *n=40 rrs=12.154939 nf=84
63 *Correlations:
64 *
65 *   X0      Y0      R
66 * X0      1.000  0.073  0.067
67 * Y0      0.073  1.000 -0.131
68 * R       0.067 -0.131  1.000
69 *
70 *The estimated circle is plotted by
71 *GPLOT X(T)=X0+R*cos(T), Y(T)=Y0+R*sin(T)
72 *
73 *           T=0,2*pi,pi/20 INFILE=A OUTFILE=A
    
```



```

41 1 SURVO 84C EDITOR Fri Jul 03 17:52:34 1992      D:\P2\STAT2\ 180 100 0
72 *
73 *The estimates may be slightly biased due to the drift caused by
74 *the outlier.
75 *To improve the estimates, a more robust estimation criterion MAD,
76 *Minimum Absolute Deviation or L1, instead of OLS (L2)
77 *can be applied. The previous OLS estimates are used as initial values.
78 *
79 *CRITERION=L1 (In fact, any Lp norm is accepted)
80 *ESTIMATE CIRCLE,Cmodel,CUR+2
81 *.....
82 *Estimated parameters of model Cmodel:
83 *X0=4.94783
84 *Y0=7.0057
85 *R=3.04986
86 *n=40 Norm L1: min=9.124031 nf=221
87 *
88 *The corresponding circle is:
89 *GPLOT X(T)=X0+R*cos(T), Y(T)=Y0+R*sin(T)-
90 *          T=0,2*pi,pi/20 INFILE=A
91 *

```

Solving equations

The ESTIMATE operation can be used for solving systems of equations by brute force. The linear equations

$$9x - 2y = 12$$

$$2x + 3y = 13$$

are solved as shown in the display

```

29 1 SURVO 84C EDITOR Sun Mar 22 11:47:00 1987      D:\P2\STAT2\ 100 100 0
1 *
2 *DATA EQUATIONS
3 *  A  B  C
4 *  9 -2 12
5 *  2  3 13
6 *
7 *MODEL LINEAR
8 *LOGDENSITY=-(A*X+B*Y-C)^2
9 *
10 *ESTIMATE EQUATIONS,LINEAR,11
11 *Estimated parameters of model LINEAR:
12 *X=2 (0.082242)
13 *Y=3 (0.210297)
14 *n=2 log(L)=-0.000000 nf=30
15 *Correlations:
16 *          X      Y
17 * X          1.000 0.361
18 * Y          0.361 1.000
19 *

```

and the nonlinear equations

$$x + y + z = 57$$

$$x^2 + y^2 + z^2 = 2595$$

$$x^3 + y^3 + z^3 = 93801$$

similarly by

```

26 1 SURVO 84C EDITOR Sun Apr 19 12:00:30 1992 D:\P2\STAT2\ 150 80 0
19 *.....
20 *DATA A: 1 END
21 *
22 *MODEL EQUATION
23 *LOGDENSITY=-90*(X+Y+Z-57)^2-9*(X^2+Y^2+Z^2-2595)^2-(X^3+Y^3+Z^3-93801)^2
24 *X=1 Y=2 Z=3
25 *ESTIMATE A,EQUATION,CUR+1
26 *Estimated parameters of model EQUATION:
27 *X=-11 (0.0120469)
28 *Y=25 (0.0121648)
29 *Z=43 (0.00470354)
30 *n=1 log(L)=-0.000000 nf=1884
31 *Correlations:
32 *           X           Y           Z
33 * X           1.000   0.709  -0.786
34 * Y           0.709   1.000  -0.993
35 * Z          -0.786  -0.993   1.000
36 *

```

In the latter case, LOGDENSITY is a weighted sum of 'squared residuals' and the data set A is a pure dummy. The weights (90, 9, 1) were selected in order to have a rough balance between the numerical errors in the three equations.

In both cases, standard errors and correlations of the 'estimates' have no importance. 'log(L)' is the least squares error and should be 0 for a true solution.

7.6 Management and analysis of multiway tables

Multiway tables consisting of structured rectangular arrays of numbers with nested textual labels are represented directly in the edit field. Such tables are obtained as results of the TAB operation which produces multiway tables of frequencies, means, etc. One can also import multiway tables from other sources and edit them to conform the style of the TAB output tables.

Survo provides several commands for management of multiway tables. These commands are used for transforming the structure of the table in different ways.

TABFIT is an operation for analysis of multiway tables. If the table contains frequencies, various log-linear models are estimated. If the elements are data values, analysis of variance is performed.

These functions are among oldest in the history of editorial approach. See, for example, Mustonen (1980) where management and analysis of multiway tables was used as an introductory example to the idea of editorial interface. The next presentation follows the same lines.

7.6.1 Structure of multiway tables

The next display illustrates how multiway tables are given in the edit field.

```

43 1 SURVO 84C EDITOR Sat Apr 25 10:57:00 1992 D:\P2\STAT2\ 100 100 0
1 *
2 * The following data are taken from a survey of the political attitudes
3 * of a sample of British electors which is reported by Butler and Stokes
4 * (1974). We examine the relationships among four variables: Vote, Sex,
5 * Class and Age.
6 * The variables are define and the observed frequencies are given in
7 * the form of a four-way table:
8 *
9 *TABLE ELECT,A,B,F
10 A Sex Male Female
11 * Vote Cons Labour Cons Labour
12 *Class Age ****
13 *
14 *upper >73 4 0 10 0
15 * 51-73 27 8 26 9
16 * 41-50 27 4 25 9
17 * 26-40 17 12 28 9
18 * <26 7 6 7 3
19 *
20 *lower >73 8 4 9 2
21 * 51-73 21 13 33 8
22 * 41-50 27 12 29 4
23 * 26-40 14 15 17 13
24 * <26 9 9 13 7
25 *
26 *work >73 8 15 17 4
27 * 51-73 35 62 52 53
28 * 41-50 29 75 32 70
29 * 26-40 32 66 36 67
30 B <26 14 34 18 33_
31 *

```

Observe that the table is represented in a nested form which is the most natural choice for general multiway tables. The structure of the table is defined by

the *classifiers* Age, Class, Vote, and Sex. The string of asterisks (***) on line 12 indicates the *row* classifiers (Age, Class) on the left side and *column* classifiers (Vote, Male) above.

To increase readability, blank lines (like 13,19,25 above) may be inserted between the data lines. The total span of the table is set by the **TABLE** definition line (here 9) giving the name of the table (**ELECT**), the first and last line (**A,B**) and the type of the elements (**F**). Here **F** stands for frequencies. The second major alternative is **X** for primary data values. The table definition line can appear anywhere in the edit field and it is a global definition. A natural place is, of course, just above the table itself.

7.6.2 Modifications and transformations

Multiway tables are maintained by the standard functions of text and table processing of the Survo Editor. These means are not powerful enough for automatic management of the hierarchical structure in multiway tables. Therefore, a special set of **TAB** commands is available.

By these commands, one can

- change the number of row classifiers (**TABS**),
- change positions of two classifiers (**TABM**),
- change positions of two classes of a classifier (**TABI**),
- delete a classifier (collapsing) (**TABD**),
- combine two classes of a classifier (**TABJ**),
- perform arithmetical operations on elements (**TAB+**, **TAB-**, **TAB***, **TAB/**).

Each **TAB** command produces a new multiway table in the current edit field. The name of the new table is created automatically from the name of the input table by inserting an extra character to the end of that name. This extra character is the fourth character of the operation name (like **S** from **TABS**).

Changing the number of row classifiers

TABS <table>, **K**, **L**

generates a new table from <table> by altering the number of row classifiers to **K**. **K** can have values 0,1,..., *n* where *n* is the total number of classifiers in <table>. **L** is the first line for the new table in the edit field.

As an example, we consider an artificial 4-way table:

```

37 1 SURVO 84C EDITOR Sat Apr 25 18:04:46 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *TABLE TEST,A,B,X
3 A      A A1      A2      A3
4 *      B B1 B2 B3 B1 B2 B3 B1 B2 B3
5 *C D *
6 *C1 D1  1  5  9 13 17 21 25 29 33
7 *      D2  2  6 10 14 18 22 26 30 34
8 *C2 D1  3  7 11 15 19 23 27 31 35
9 B      D2  4  8 12 16 20 24 28 32 36_
10 *

```

The column classifiers **A** and **B** both have 3 classes each while the classifiers

C and D are dichotomous. If we like to reorganize the table so that also C becomes a column classifier, this situation is reached by the TABS command with K=1:

```

18 1 SURVO 84C EDITOR Sat Apr 25 18:32:21 1992 D:\P2\STAT2\ 100 100 0
10 *
11 *TABS TEST,1,CUR+1
12 *TABLE TESTS C,D,F
13 C A A1 A2 A3
14 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
15 * C C1 C2 C1 C2 C1 C2 C1 C2 C1 C2 C1 C2 C1 C2 C1 C2
16 *D *
17 *D1 1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35
18 DD2 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36
19 *

```

The new table TESTS created by TABS is on lines 12-18. Observe that the first and the last line of the new table are indicated by symbolic labels (here C and D) selected automatically.

To have all classifiers on the row side, we select K=4 and obtain:

```

18 1 SURVO 84C EDITOR Sat Apr 25 18:40:03 1992 D:\P2\STAT2\ 100 100 0
10 *
11 *TABS TEST,4,CUR+1
12 *TABLE TESTS C,D,F
13 CA B C D *
14 *A1 B1 C1 D1 1
15 * D2 2
16 * C2 D1 3
17 * D2 4
18 * B2 C1 D1 5
19 * D2 6
20 * C2 D1 7
21 * D2 8
22 * B3 C1 D1 9
23 * D2 10
24 * C2 D1 11
25 * D2 12
26 *A2 B1 C1 D1 13
27 * D2 14
28 * C2 D1 15
29 * D2 16
30 * B2 C1 D1 17
31 * D2 18
32 * C2 D1 19

```

Changing positions of classifiers

TABM <table>,A,B,L

changes the positions of classifiers A and B as shown in the next example.


```

20 1 SURVO 84C EDITOR Mon Apr 27 19:51:39 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *TABLE TEST,A,B,F
3 A A A1 A2 A3
4 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
5 *C D *
6 *C1 D1 1 5 9 13 17 21 25 29 33
7 * D2 2 6 10 14 18 22 26 30 34
8 *C2 D1 3 7 11 15 19 23 27 31 35
9 B D2 4 8 12 16 20 24 28 32 36
10 *
11 *TABM TEST,A,D,CUR+1
12 *TABLE TESTM C,D,F
13 C D D1 D2
14 * B B1 B2 B3 B1 B2 B3
15 *C A *
16 *C1 A1 1 5 9 2 6 10
17 * A2 13 17 21 14 18 22
18 * A3 25 29 33 26 30 34
19 *C2 A1 3 7 11 4 8 12
20 * A2 15 19 23 16 20 24
21 D A3 27 31 35 28 32 36
22 *

```

Changing positions of two classes of a classifier

TABI <table>,C,C1,C2,L

changes the positions of classes C1 and C2 in classifier C. For example, the positions of A1 and A3 are changed as follows:

```

24 1 SURVO 84C EDITOR Mon Apr 27 20:00:00 1992 D:\P2\STAT2\ 100 100 0
10 *
11 *TABI TEST,A,A1,A3,CUR+1
12 *TABLE TESTI C,D,F
13 C A A3 A2 A1
14 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
15 *C D *
16 *C1 D1 25 29 33 13 17 21 1 5 9
17 * D2 26 30 34 14 18 22 2 6 10
18 *C2 D1 27 31 35 15 19 23 3 7 11
19 D D2 28 32 36 16 20 24 4 8 12
20 *

```

Deleting a classifier (collapsing)

TABD <table>,C,L

deletes a classifier C.

In a *frequency table* (F), the elements of the modified table are the marginal sums of the C classification; in this case, TABD means collapsing over the classifier C and the results is an $(n-1)$ -dimensional table.

In a *data table* (X), the C classification will be renamed as an N classification where the C class names are replaced by indices 1,2, etc. The modified table is still n -dimensional and has the old structure and contents; only the name C is replaced by N. If TABD is applied to a data table which already includes an N classification, the C classification will be joined in the N classification. The dimension of the table decreases by 1 but the elements remain the same.

The D class is eliminated in our original frequency table by

```

18 1 SURVO 84C EDITOR Sun May 10 12:14:58 1992 D:\P2\STAT2\ 100 100 0
10 *
11 *TABD TEST,D,CUR+1
12 *TABLE TESTD C,D,F
13 C A A1 A2 A3
14 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
15 *C *
16 *C1 3 11 19 27 35 43 51 59 67
17 DC2 7 15 23 31 39 47 55 63 71
18 *

```

See what happens to a corresponding table of data values (type X) when both D and C classifiers are deleted:

```

20 1 SURVO 84C EDITOR Sun May 10 12:18:42 1992 D:\P2\STAT2\ 100 100 0
20 *TABLE TEST2,a,b,X
21 a A A1 A2 A3
22 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
23 *C D *
24 *C1 D1 1 5 9 13 17 21 25 29 33
25 * D2 2 6 10 14 18 22 26 30 34
26 *C2 D1 3 7 11 15 19 23 27 31 35
27 b D2 4 8 12 16 20 24 28 32 36
28 *
29 *TABD TEST2,D,CUR+1
30 *TABLE TEST2D E,F,X
31 E A A1 A2 A3
32 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
33 *C N *
34 *C1 1 1 5 9 13 17 21 25 29 33
35 * 2 2 6 10 14 18 22 26 30 34
36 *C2 1 3 7 11 15 19 23 27 31 35
37 F 2 4 8 12 16 20 24 28 32 36
38 *
39 *TABD TEST2D,C,CUR+1
40 *TABLE TEST2DD G,H,X
41 G A A1 A2 A3
42 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
43 *N *
44 *1 1 5 9 13 17 21 25 29 33
45 *2 2 6 10 14 18 22 26 30 34
46 *3 3 7 11 15 19 23 27 31 35
47 H4 4 8 12 16 20 24 28 32 36
48 *

```

Combining two classes of a classifier

TABJ <F-table>,C,C1,C2,C0,L

combines in a frequency table (F) the classes C1 and C2 of the classifier C and calls the combined class by name C0. The class C0 takes the position of C1 and the C2 class will be cancelled.

The classes B1 and B3 are combined to class B13 as follows:

```

28 1 SURVO 84C EDITOR Sun May 10 12:34:03 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *TABLE TEST,A,B,F
3 A A A1 A2 A3
4 * B B1 B2 B3 B1 B2 B3 B1 B2 B3
5 *C D *
6 *C1 D1 1 5 9 13 17 21 25 29 33
7 * D2 2 6 10 14 18 22 26 30 34
8 *C2 D1 3 7 11 15 19 23 27 31 35
9 B D2 4 8 12 16 20 24 28 32 36
10 *
11 *TABJ TEST,B,B1,B3,B13,CUR+1
12 *TABLE TESTJ C,D,F
13 C A A1 A2 A3
14 * B B13 B2 B13 B2 B13 B2
15 *C D *
16 *C1 D1 10 5 34 17 58 29
17 * D2 12 6 36 18 60 30
18 *C2 D1 14 7 38 19 62 31
19 D D2 16 8 40 20 64 32
20 *

```

Arithmetical operations on multiway tables

TAB+ <table1>, <table2>, L

makes a new multiway table whose elements are sums of corresponding elements in tables <table1> and <table2>. Tables concerned must have the same structure.

Operations

TAB- <table1>, <table2>, L

TAB* <table1>, <table2>, L

TAB/ <table1>, <table2>, L

compute differences, products and ratios of elements.

In all above operations, <table2> may be replaced by a constant value. For example, **TAB/ SAMPLE1, 100, 23** divides all the elements of table **SAMPLE1** by 100, writes the modified table from line 23 onwards and labels it as **TABLE SAMPLE1/**.

7.6.3 Log-linear models and analysis of variance

TABFIT <table>, <name of model>, L

estimates the parameters of a log-linear model for frequency data given as a multiway table (F table) or estimates the parameters of an ANOVA model for data given as a multiway table (X table).

Both types of **TABFIT** models are special cases of GLM (generalized linear models) and treated here by the same program. More general cases using raw data as Survo data sets (in data files) are treated by the **GENREG** operation or by an optional **ANOVA** operation. The latter one has been programmed by Markku Korhonen.

The type of the multiway table (F or X) is given as the last parameter of the **TABLE** definition for <table>. It specifies whether a log-linear model (with

the log link) or a classical ANOVA model (with the identity link) is estimated.

<table> has to be written in the edit field and it has to be defined by a TABLE specification. The structure of <table> is that of tables produced by the TAB operation (See TAB?). Thus, multiway tables computed by TAB can be used as input for TABFIT without any changes. Similarly, results of table management operations TABD, TABM, etc. are valid input tables.

Parameter L is optional and gives the first line for the results in the edit field.

The model to be fitted has to be specified by a MODEL definition typed on any two consecutive lines in the edit field as follows:

```
MODEL <name of model>
      <list of terms to be fitted>
```

Each term in the model may consist either of one (main effect) or several (interaction between factors) factors. The factors are notated by the initials of the classifier names in the table.

The terms to be fitted are separated by a '+' operator. If the term is an interaction term of two or several classifiers, the factors must be separated by '.'. For example, A.B represents an interaction of A and B. Also an '*' operator can be used:

```
For example,  A*B=A+B+A.B
              A*B*C=A+B+C+A.B+A.C+B.C+A.B.C
```

Any existing term is removed from the model by the operator '-'. For example, A*B*C-A.B.C=A+B+C+A.B+A.C+B.C. The constant term is given as '1'. A typical model definition is

```
MODEL M1
1+S+B*K
```

where S,B and K are initials of classifiers of the multiway table.

The extent of results given by TABFIT is controlled by the RESULTS specification. If RESULTS=0, only minimal information (i.e. deviance, degrees of freedom and structure of the model) is printed. Otherwise also the estimates of the parameters and their standard errors are given. Furthermore, if a value >70 (like RESULTS=100) is given, the fitted values are displayed as a new multiway table having the same structure as the original table.

In all cases, the design matrix generated by TABFIT is saved as a matrix file XTAB.M and the covariance estimates of parameters as PCOV.M on the current data disk.

Example: Log-linear model

In the following display, the 4-way table of electors is considered. At first, a reduction to three dimensions is carried out by collapsing over the classifier 'Age'. Thereafter, a model M1 (on lines 42-43) is estimated by TABFIT on line 41. The results appear on lines 44-62.

```

23 1 SURVO 84C EDITOR Mon May 11 18:27:00 1992 D:\P2\STAT2\ 100 100 0
31 *
32 *TABD ELECT, Age, CUR+1
33 *TABLE ELECTD C,D,F
34 C Sex Male Female
35 * Vote Cons Labour Cons Labour
36 *Class ****
37 *upper 82 30 96 30
38 *lower 79 53 101 34
39 Dwork 118 252 155 227
40 *
41 *TABFIT ELECTD, M1, CUR+3 / RESULTS=100
42 *MODEL M1
43 *V*S+V*C
44 *Table ELECTD: Deviance=2.75698 df=4 P=0.5993
45 *Model: V*S+V*C
46 * estimate s.e. parameter
47 * 1 4.36569 0.08728 1
48 * 2 0.01117 0.10570 C2
49 * 3 0.42769 0.09634 C3
50 * 4 -0.89656 0.16022 V2
51 * 5 0.36039 0.19833 C2V2
52 * 6 1.64967 0.16744 C3V2
53 * 7 0.23242 0.08016 S2
54 * 8 -0.37323 0.11334 V2S2
55 *
56 *TABLE ELECTDFIT E,F,F
57 E Sex Male Female
58 * Vote Cons Labour Cons Labour
59 *Class ****
60 *upper 78.704 32.109 99.296 27.891
61 *lower 79.588 46.558 100.412 40.442
62 Fwork 120.708 256.334 152.292 222.666
63 *

```

TABLE sucros for analysis of variance

Data in multiway tables of type X can be analyzed by the TABFIT operation. However, several activations of TABFIT with different models are needed in order to obtain the standard ANOVA output.

To achieve such results more or less automatically, a set of TABLE sucros has been programmed. The most important of them is /TABLE-ANOVA which performs the necessary calculations for an n factor experiment ($n=2,3,4$). To see other functions in the TABLE sucro family, activate /TABLE-README .

By applying /TABLE-ANOVA to a four-dimensional table EXP, the following results are obtained:

```

17 1 SURVO 84C EDITOR Tue May 12 08:32:21 1992 D:\P2\STAT2\ 100 100 0
1 *
2 *TABLE EXP,A,B,X
3 A A 1 2 3
4 * B 1 2 1 2 1 2
5 * C D **
6 * 1 1 1.3 1.2 1.8 1.5 2.3 2.1
7 * 2 1.2 1.0 1.7 1.5 2.4 2.0
8 *
9 * 2 1 1.5 1.4 2.2 1.9 2.6 2.4
10 B 2 1.5 1.3 2.3 2.0 2.9 2.7
11 *
12 */TABLE-ANOVA EXP
13 *Source of Sum of Degrees of
14 *variation squares freedom
15 *Total 6.5162500 23
16 *A 5.0625000 2
17 *B 0.3037500 1
18 *A*B 0.0175000 2
19 *C 0.9204170 1
20 *A*C 0.0608330 2
21 *B*C 0.0004160 1
22 *A*B*C 0.0058340 2
23 *D 0.0037500 1
24 *A*D 0.0625000 2
25 *B*D 0.0037500 1
26 *A*B*D 0.0075000 2
27 *C*D 0.0504160 1
28 *A*C*D 0.0108340 2
29 *B*C*D 0.0004180 1
30 *A*B*C*D 0.0058320 2
31 *

```

If we assume that in the above setup the A classifier merely accounts for replications in the experiment (i.e. we have 3 replicates in each class), the components containing the A classifier can be pooled together to form the actual residual sum of squares. This is done by the /TABLE-POOL sucro which must be activated just below the 'Total' line:

```

14 1 SURVO 84C EDITOR Tue May 12 08:55:10 1992 D:\P2\STAT2\ 100 100 0
11 *
12 */TABLE-ANOVA EXP
13 *Source of Sum of Degrees of
14 *variation squares freedom
15 *Total 6.5162500 23
16 */TABLE-POOL A
17 *B 0.3037500 1
18 *C 0.9204170 1
19 *B*C 0.0004160 1
20 *D 0.0037500 1
21 *B*D 0.0037500 1
22 *C*D 0.0504160 1
23 *B*C*D 0.0004180 1
24 *A* 5.2333330 16
25 *

```

The pooled A components appear here on line 24 with a label 'A*'. Of course, similar reductions can be accomplished repeatedly. The values and *P* values of the common *F* statistics are computed by editorial computing and touch mode when necessary. For standard layouts, more refined sucros can be easily created.

For more general applications, the optional module ANOVA (made by Markku Korhonen, 1989) is available. By the ANOVA operation, a wide range of variance and covariance models can be analyzed. Multivariate analysis of

(co)variance and analysis of generalized repeated measurements are included. Tests associated with these models involve e.g. multiple comparisons of means and mean vectors. Tests for equality of group variances are available, too.

7.7 Generalized linear models

GENREG <data>,L

estimates a generalized linear model (GLM) defined by **ERROR** and **LINK** specifications from a Survo data where the dependent variate is activated by **Y** and the covariates by **X**'s. **IND**, **CASES** and **SELECT** specifications are used for selection of observations.

The theory of GLM's is described in '*Generalized Linear Models*' by P.McCullagh and J.A.Nelder (Chapman and Hall 1983).

Variables for the model and for results are selected by masks as follows:

| | |
|----------|--|
| Y | dependent variate (only one Y allowed) |
| X | covariate (also the constant term must be specified by X) |
| R | for output of scaled Pearson residuals (optional) |
| P | for predicted values of Y (optional) |
| N | number of trials in ERROR=Binomial (Y is number of successes) |

Possible error structures in **GENREG** are

| | |
|-----------------------|----------------------------|
| ERROR=Normal | (LINK=Identity) |
| ERROR=Poisson | (LINK=Log) |
| ERROR=Binomial | (LINK=Logit) |
| ERROR=Gamma | (LINK=Reciprocal) |

The default (canonical) links are in parentheses.

The link function is given by any of the following alternatives:

| | |
|-----------------------------------|--------------------------------------|
| LINK=Identity | (default in ERROR=Normal) |
| LINK=Log | (default in ERROR=Poisson) |
| LINK=Logit | (default in ERROR=Binomial) |
| LINK=Reciprocal | (default in ERROR=Gamma) |
| LINK=Probit | (N variable must be given) |
| LINK=Complementary_log-log | (N variable must be given) |
| LINK=Square_root | |
| LINK=Exponent, alpha | (x^{α} power transformation) |

The extent of results given by **GENREG** is controlled by the **RESULTS** specification. If **RESULTS=0**, only minimal information (i.e. deviance, degrees of freedom, the error structure, and the link) is printed. Otherwise also the estimates of the parameters and their standard errors are given.

In any case, the covariance estimates of parameters are saved as a matrix file **PCOV.M** on the current data disk.

The **TABFIT** operation generates, as a by-product, the design matrix extended by the dependent variate and the external weight function according to the current model. This matrix is saved (even in saturated cases) as a matrix file **XTAB.M**.

The matrix file **XTAB.M** as such (or more conveniently, its transform to a data file by **FILE SAVE MAT** operation) can be used as an input data set for **GENREG**. Data for various models including factors and their interactions can be organized easily by this procedure.

The absolute maximum dimensions for the data are $M=90$ for the regressors and about $N=8100$ for the observations, but the true maximum dimensions depend on the total memory available.

If possible, the active part of the data is kept in the central memory up to 64 kilobytes (about 8100 data values). In larger data sets, the rest of the data values are saved in a temporary file **SURVO.TMP** and loaded from it when needed. The default path for **SURVO.TMP** is given by the 'tempdisk' line in **SURVO.APU**, but it can be replaced by another by a **TEMP** specification (e.g. **TEMP=C:** or **TEMP=D:\TMP**).

The number of observations in the central memory can be further limited by **N0=<#_of_observations>**. In that case, more space in the central memory is available for auxiliary vectors needed in computations. Even value **N0=0** is accepted.

If **N0=0**, the need of work space in the CPU is about $8*(4*N+M*(M+6))$ bytes. In a typical MS-DOS environment, about 185 KB is left for this work space. Thus a **GENREG** run with $M=50$, $N=5000$ should be possible.

As an example, we consider the earlier data on British electors and try to estimate by the **GENREG** operation the same log-linear model that was already computed by **TABFIT** in 7.6.

We start by generating the design matrix corresponding to the three-way input table used in **TABFIT**. This is done by trying to use **TABFIT** with a saturated model:


```

18 1 SURVO 84C EDITOR Tue May 12 12:11:20 1992 D:\P2\STAT2\ 200 100 0
65 *
66 *MODEL MS
67 *V*S*C
68 *TABFIT ELECTD,MS,CUR+1
69 *Model MS is saturated (XTAB.M saved!)
70 *
71 *FILE SAVE MAT XTAB.M TO EDATA
72 *FILE STATUS EDATA
73 * Copied from matrix file XTAB.M
74 *FIELDS: (active)
75 * 1 SA_ 8 CASE
76 * 2 NA_ 4 Weight
77 * 3 NA_ 4 ELECTD
78 * 4 NA_ 4 1
79 * 5 NA_ 4 C2
80 * 6 NA_ 4 C3
81 * 7 NA_ 4 V2
82 * 8 NA_ 4 C2V2
83 * 9 NA_ 4 C3V2
84 * 10 NA_ 4 S2
85 * 11 NA_ 4 C2S2
86 * 12 NA_ 4 C3S2
87 * 13 NA_ 4 V2S2
88 * 14 NA_ 4 C2V2S2
89 * 15 NA_ 4 C3V2S2
90 *END
91 *SURVO 84C data file EDATA: record=100 bytes, M1=22 L=64 M=15 N=12
92 *

```

The design matrix with the dependent variate (frequency as variable ELECTD) and with all possible covariates has been created as a matrix file XTAB.M (by TABFIT on line 68) and transformed to a Survo data file EDATA (by FILE SAVE MAT on line 71).

The structure of the data file is displayed by the FILE STATUS command (on lines 72-91). The contents of EDATA can be studied, for example, by FILE SHOW giving the following display:

```

16 1 SURVO 84C EDITOR Wed Jan 25 13:19:30 1992 D:\P2\DATA\ 100 100 0
File EDATA N=12
1 CASE Weight ELECTD 1 C2 CASE C3 CIV1S1 V2
1C1V1S1 1.000 82.000 1.000 0.000 0.000 0.000
2C2V1S1 1.000 79.000 1.000 1.000 0.000 0.000
3C3V1S1 1.000 118.000 1.000 0.000 1.000 0.000
4C1V2S1 1.000 30.000 1.000 0.000 0.000 1.000
5C2V2S1 1.000 53.000 1.000 1.000 0.000 1.000
6C3V2S1 1.000 252.000 1.000 0.000 1.000 1.000
7C1V1S2 1.000 96.000 1.000 0.000 0.000 0.000
8C2V1S2 1.000 101.000 1.000 1.000 0.000 0.000
9C3V1S2 1.000 155.000 1.000 0.000 1.000 0.000
10C1V2S2 1.000 30.000 1.000 0.000 0.000 1.000
11C2V2S2 1.000 34.000 1.000 1.000 0.000 1.000
12C3V2S2 1.000 227.000 1.000 0.000 1.000 1.000
13
14
15
16
17
18
19
20
21
To stop, press EXIT! (F1=HELP)

```

Variables belonging to the model $V*S+V*C$ can be selected from EDATA by activating MASK and on return we have the list of them in the edit field (line 93 in the next display). When GENREG is activated with ERROR=Poisson

and LINK=Log (the latter would be taken also by default), we obtain the same results as in the previous TABFIT example:

```

19 1 SURVO 84C EDITOR Tue May 12 12:48:25 1992 D:\P2\STAT2\ 200 100 0
92 * .....
93 *MASK=--YXXXXXXX--X--
94 *GENREG EDATA,CUR+1 / ERROR=Poisson LINK=Log
95 *Data EDATA: Deviance=2.756985 df=4
96 *Yvariate=ELECTD ERROR=Poisson LINK=Log
97 *Parameter Estimate s.e.
98 *1 4.365689 0.087278
99 *C2 0.011173 0.105705
100 *C3 0.427688 0.096338
101 *V2 -0.896565 0.160224
102 *C2V2 0.360390 0.198329
103 *C3V2 1.649668 0.167438
104 *S2 0.232419 0.080157
105 *V2S2 -0.373227 0.113343
106 *

```

7.8 Multivariate analysis



All linear multivariate operations are more or less connected to the matrix interpreter of Survo. For an experienced user who is familiar with matrix algebra, the MAT operations provided by the matrix interpreter offer the most straightforward means for calculations in multivariate analysis and linear models.

In addition, many of the standard methods will be available as integrated operations. Then the user has no absolute need to master MAT operations. However, some basic knowledge about how Survo works on matrices would be valuable in any case.

Correspondence analysis?

Cluster analysis?

Distance measures?

Multidim. scaling?

Knowledge discovery?

Hunting quanta?

MULTI?

Thus, we shall start this section by giving a short description of Survo matrix files and their conversion from and to Survo data files. The MAT operations will be reported later in a more detailed form.

Another introductory topic is the LINCO operation which is the general tool for computing linear combinations of variables.

This section will be concluded by a general account of MATRUN operations and separate descriptions of various multivariate operations.

7.8.1 Matrix files

As mentioned earlier (7.1.8), many of the statistical operations save their results in a matrix form. These matrices can be used later as input in other operations.

The following data file DECA on the 48 best athletes of the world in decathlon (1973) will be used in several of our examples of multivariate analysis. The structure of the data set is introduced by a FILE STATUS operation as follows:

```

10 1 SURVO 84C EDITOR Tue Mar 31 09:25:05 1987 D:\P2\STAT3\ 120 80 0
1 *
2 *FILE STATUS DECA,3 / 3 first activation columns to be shown
3 *
4 * 48 best athletes in decathlon (1973)
5 * {lower_limit,upper_limit}
6 *FIELDS: (active)
7 * 1 SA-- 8 Name Name of athlete
8 * 2 NA-I 2 Points Total score (####) {7000,9000}
9 * 3 NA-I 2 100m 100 meters run (####) {500,1200}
10 * 4 NA-I 2 L_jump Long jump (####) {500,1200}
11 * 5 NA-I 2 Shot_put (####) {500,1200}
12 * 6 NA-I 2 Hi_jump High jump (####) {500,1200}
13 * 7 NA-I 2 400m 100 meters run (####) {500,1200}
14 * 8 NA-I 2 Hurdles 110 meters hurdles (####) {500,1200}
15 * 9 NA-I 2 Discus (####) {500,1200}
16 * 10 NA-I 2 Pole_vlt Pole vault (####) {500,1200}
17 * 11 NA-I 2 Javelin (####) {500,1200}
18 * 12 NA-I 2 1500m 1500 meters run (####) {400,1200}
19 * 13 NA-R 2 Height in centimeters (###) {160,210}
20 * 14 NA-R 2 Weight in kilograms (###) {50,120}
21 *END
22 *SURVO 84C data file DECA: record=128 bytes, M1=30 L=64 M=14 N=48
23 *CORR DECA / MASK=--AAAAAAAAAAAA-

```

To compute the correlation matrix CORR.M (as well as the matrix of means and standard deviations MSN.M) for selected statistical variables, the CORR operation with a MASK specification on line 23 is activated. Since no line number or label is given for the results in CORR, the results will not appear directly in the edit field. However, all the results in the text form will be saved in the current output file of Survo and the resultant matrices in an accurate mode in the matrix files CORR.M and MSN.M.

Any matrix file created by statistical operations can be loaded to the edit field (entirely or partially) by the MAT LOAD command. Also the format of matrix elements can be given in this command.

In the next display, CORR.M is shown in this way:

```

25 1 SURVO 84C EDITOR Tue Mar 31 09:26:28 1987 D:\P2\STAT3\ 120 80 0
22 *SURVO 84C data file DECA: record=128 bytes, M1=30 L=64 M=15 N=48
23 *CORR DECA / MASK=--AAAAAAAAAAAA-
24 *MAT LOAD CORR.M,##.##,25_
25 *MATRIX CORR.M
26 *R(DECA)
27 */// 100m L_jum Shot_ Hi_ju 400m Hurd1 Discu Pole_ Javel 1500m Hei
28 *100m 1.00 0.17 -0.03 -0.41 0.46 0.32 0.01 0.05 -0.22 -0.29 -0.
29 *L_jump 0.17 1.00 -0.03 -0.00 0.13 0.30 0.02 0.06 0.15 -0.21 -0.
30 *Shot_put -0.03 -0.03 1.00 0.16 -0.30 0.09 0.73 -0.20 0.02 -0.45 0.
31 *Hi_jump -0.41 -0.00 0.16 1.00 -0.34 -0.04 0.22 -0.12 0.15 -0.15 0.
32 *400m 0.46 0.13 -0.30 -0.34 1.00 0.18 -0.34 0.01 -0.10 0.30 -0.
33 *Hurdles 0.32 0.30 0.09 -0.04 0.18 1.00 0.05 -0.07 -0.15 -0.22 0.
34 *Discus 0.01 0.02 0.73 0.22 -0.34 0.05 1.00 -0.18 0.14 -0.57 0.
35 *Pole_vlt 0.05 0.06 -0.20 -0.12 0.01 -0.07 -0.18 1.00 -0.13 0.01 -0.
36 *Javelin -0.22 0.15 0.02 0.15 -0.10 -0.15 0.14 -0.13 1.00 -0.07 -0.
37 *1500m -0.29 -0.21 -0.45 -0.15 0.30 -0.22 -0.57 0.01 -0.07 1.00 -0.
38 *Height -0.11 -0.05 0.62 0.13 -0.16 0.23 0.59 -0.35 -0.04 -0.24 1.
39 *Weight -0.08 -0.05 0.71 0.16 -0.32 0.13 0.64 -0.31 -0.07 -0.40 0.
40 *

```

If the matrix in the selected format is too wide, some of the last columns will be lost in the display. To avoid this, the field can be redimensioned or the operation LOADM used instead. LOADM splits the matrix in parts according to the visible line length of the edit field.

Matrices which are loaded or written in the edit field can be saved in the matrix files (by a `MAT SAVE` operation; see `MAT?`), but, generally, transportation of matrices through the visible form in the edit field should be avoided due to rounding errors.

It is easy to process any matrix in a matrix file directly by `MAT` operations. For example, the following series of `MAT` operations computes the principal components of `CORR.M` and loads 4 first of them with eigenvalues to the edit field.

```

1 1 SURVO 84C EDITOR Sun Jun 07 08:02:58 1992 D:\P2\STAT3\ 120 80 0
40 *
41 *MAT SPECTRAL DECOMPOSITION OF CORR.M TO F,L
42 *MAT L2=L / *L2~L(R(DECA)) 12*1
43 *MAT TRANSFORM L2 BY sqrt(X#)
44 *MAT L2=DV(L2) / *L2~DV(T(L2_by_sqrt(X#))) D12*12
45 *MAT F=F*L2 / *F~S(R(DECA))*DV(T(L2_by_sqrt(X#))) 12*12
46 *MAT LOAD F(*,1:4),END+2
47 *MAT L=L' / *L~L(R(DECA))' 1*12
48 *MAT LOAD L(1,1:4),END+2
49 *
50 *MATRIX F
51 *S(R(DECA))*DV(T(L2_by_sqrt(X#)))
52 */// ev1 ev2 ev3 ev4
53 *100m 0.12816 0.83642 0.00815 0.15782
54 *L_jump 0.00992 0.40515 0.64153 -0.30405
55 *Shot_put -0.84273 0.07682 -0.06674 0.11749
56 *Hi_jump -0.33564 -0.48502 0.32019 -0.14217
57 *400m 0.47815 0.53732 -0.23010 -0.33683
58 *Hurdles -0.15273 0.63906 0.10293 -0.24942
59 *Discus -0.84701 0.06759 0.12843 0.13626
60 *Pole_vlt 0.35474 0.03040 0.29487 0.68117
61 *Javelin -0.08661 -0.31513 0.52890 -0.45301
62 *1500m 0.58976 -0.33353 -0.46695 -0.31092
63 *Height -0.80838 0.10857 -0.33820 -0.19945
64 *Weight -0.88318 0.07353 -0.25084 -0.01554
65 *
66 *MATRIX L
67 *L(R(DECA))'
68 */// 1 2 3 4
69 *eigenval 3.723388 2.035237 1.360654 1.151512
70 *

```

The results appear here on lines 50-69 and comments written by the `MAT` operations are displayed in gray shading. The analysis proceeds through the following steps. `MAT SPECTRAL DECOMPOSITION OF CORR.M TO F,L` finds the spectral decomposition $CORR.M = FLF'$ where L is the diagonal matrix of eigenvalues and F the orthogonal matrix of eigenvectors. To save space, matrix L is saved as a column vector. `MAT L2=L` (on line 42) makes a copy of L and `MAT TRANSFORM` (line 43) takes square roots of all elements of $L2$. Then `MAT L2=DV(L2)` converts vector $L2$ to a diagonal matrix and the columns of the principal component matrix F are rescaled by the matrix multiplication `MAT F=F*L2`. Finally, the results are loaded to the edit field after the last used line.

By using the `DIR` command of Survo we find the following matrix files on the current data disk/path:

```

22 1 SURVO 84C EDITOR Sun Jun 07 08:04:05 1992 D:\P2\STAT3\ 120 80 0
71 *
72 *DIR D:\P2\STAT3\*.M??_
73 *
74 * Volume in drive D has no label
75 * Directory of D:\P2\STAT3
76 *
77 *CORR      M          1072 07.06.92   8.02
78 *MSN       M           664 07.06.92   8.02
79 *F         MAT        1600 07.06.92   8.02
80 *L         MAT         456 07.06.92   8.03
81 *L2        MAT         544 07.06.92   8.02
82 *          5 file(s)          4336 bytes
83 *                               3825664 bytes free
84 *

```

We see that the matrices generated by the MAT operations have the default extension .MAT in their names while those given by statistical operations as intermediate results have the extension .M. Thus in MAT operations, the complete name of the correlation matrix CORR.M produced by the CORR operation must be applied. In all respects the .MAT and .M files are identical in structure and may be employed in various matrix operations.

A data file or a part of it can also be converted to a matrix file provided that the number of elements in the resulting matrix is not more than 8192. The MAT SAVE DATA operation does the job for us.

In MAT SAVE DATA, the MASK, VARS, IND, CASES and SELECT specifications are available for selecting variables and observations. In the following example, a part of the FINLAND data has been saved by MAT SAVE DATA in the matrix file DAT1.MAT and the matrix file has then been loaded to the edit field by the MAT LOAD operation.

```

25 1 SURVO 84C EDITOR Sun Jun 07 08:36:58 1992 D:\P2\STAT3\ 100 100 0
1 *
2 *VARS=Commune,Agri,Industry,Service IND=Popul,100000,500000
3 *MAT SAVE DATA FINLAND TO DAT1
4 *
5 *MAT LOAD DAT1,#####,6_
6 *MATRIX DAT1
7 *
8 *///          Agri Industry  Service
9 *Espoo         0           2         7
10 *Helsinki     0           2         7
11 *Tampere      0           4         5
12 *Turku        0           4         5
13 *Vantaa       0           3         6
14 *

```

When operating with matrix files, Survo keeps track of the column and row labels as well. Thus when a data file is converted into a matrix file, the names of variables appear as column labels. Similarly, row labels are selected as the values of the first active variable, provided that it is a string variable. If the first active variable is not a string variable but a numeric variable, this variable appears as the first column in the matrix and numeric indices 1,2,3, ... are used as row labels. By using various characters (like 'A', 'B', 'X' etc.) for activation of variables, the order of columns may be freely selected because MAT SAVE DATA processes the variables in the alphabetic order of their activation bytes.

An inverse operation is possible, too. A matrix file is converted into a Survo data file by a FILE SAVE MAT operation. We have continued the preceding example by transposing DAT1 to DAT2=DAT1' and then saved DAT2 to a new data file FTRANSP by FILE SAVE MAT. Finally, in order to display the result, FTRANSP has been loaded into the edit field by FILE LOAD.

```

21 1 SURVO 84C EDITOR Sun Jun 07 08:56:25 1992 D:\P2\STAT3\ 100 100 0
15 *
16 *MAT DAT2=DAT1' / *DAT2~DAT1' 3*5
17 *FILE SAVE MAT DAT2 TO FTRANSP
18 *FILE LOAD FTRANSP,19_
19 *DATA FTRANSP*,A,B,C
20 C CASE Espoo Helsinki Tampere Turku Vantaa
21 A Agri 0.000 0.000 0.000 0.000 0.000
22 * Industry 2.000 2.000 4.000 4.000 3.000
23 B Service 7.000 7.000 5.000 5.000 6.000
24 *

```

More information and examples about matrix manipulation in Survo will be given later in Chapter 10.

7.8.2 Linear combinations of variables

In many applications of multivariate methods, one of the main goals is to create new variables as linear combinations of others. In some statistical modules those linear combinations can be computed directly and saved as values of selected variables. For example, in the LINREG operation, the residuals and the predicted values of the regression model are saved in such a way.

In general, however, it is profitable to have the matrix of coefficients for linear combinations available, too. For example, the LINREG operation saves the regression coefficients to a matrix file REG.M.

A special LINCO operation is then available for computing linear combinations afterwards for any data set by using a selected matrix of coefficients.

LINCO <data>, <matrix_of_coefficients>

computes and saves linear combinations of variables in <data> to the same data (file). <matrix_of_coefficients> is a standard matrix file with the following properties:

The names of variables to be used in linear combinations are the row labels (8 first characters will be matched) of the matrix. An additive constant must be indicated by row label 'constant' or '-'.

The names of the linear combinations are the column labels of the matrix. New variables (of 4 bytes) are created when necessary.

In the next display, two linear combinations U,V of variables X,Y,Z

$$U=4X+3Y+2Z$$

$$V=10+2X+2Y-7Z$$

are computed for all 5 observations of data XYZ. At first, the matrix of coefficients COEFF has been written on lines 11-16 and then COEFF has been saved to a matrix file COEFF.MAT by the MAT SAVE COEFF operation on

line 18. Finally, LINC0 XYZ,COEFF computes the desired linear combinations (results shaded).

```

16 1 SURVO 84C EDITOR Wed Apr 01 16:40:20 1987 D:\STAT\ 120 80 0
1 *
2 *DATA XYZ,A,B,N,M
3 M AA AA AA 12345 12345
4 N X Y Z U V
5 A 12 41 23 217 -45
6 * 8 23 11 123 -5
7 * -5 31 9 91 -1
8 * 5 29 31 169 -139
9 B 16 17 20 155 -64
10 *
11 *MATRIX COEFF
12 */// U V
13 *constant 0 10
14 *X 4 2
15 *Y 3 2
16 *Z 2 -7
17 *
18 *MAT SAVE COEFF
19 *LINC0 XYZ,COEFF
20 *

```

To have a possibility of changing names of variables in a particular application of LINC0, some of the column and row labels of the coefficient matrix can be given as %1,%2,%3, ... Then in the LINC0 call of the form

LINC0 <data>, <matrix> (C1,C2,C3),

labels %1,%2 and %3 are replaced by C1,C2 and C3, respectively.

The next exhibit serves as an application of this general form of LINC0. Assume that we have computed a regression model by LINREG as follows:

```

14 1 SURVO 84C EDITOR Sun Jun 07 09:53:22 1992 D:\P2\STAT3\ 100 100 0
1 *
2 *VARS=Points(Y),100m(X),L_jump(X),Shot_put(X),Hi_jump(X),400m(X)
3 *RESULTS=50
4 *LINREG DECA,5
5 *Linear regression analysis: Data DECA, Regressand Points N=48
6 *Variable Regr.coeff. Std.dev. t beta
7 *100m 0.542568 0.306360 1.771 0.199
8 *L_jump 1.361467 0.302477 4.501 0.427
9 *Shot_put 1.171460 0.258941 4.524 0.448
10 *Hi_jump 0.922328 0.261650 3.525 0.370
11 *400m 1.327236 0.363205 3.654 0.409
12 *constant 3559.491 516.3134 6.894
13 *Variance of regressand Points=26131.99956 df=47
14 *Residual variance=10607.97908 df=42
15 *R=0.7983 R^2=0.6372
16 *

```

The LINREG operation has produced a matrix file REG.M of regression coefficients as a by-product. To see the structure of this matrix, it has been loaded to the edit field by MAT LOAD REG.M (on line 17 of the next display).

```

23 1 SURVO 84C EDITOR Sun Jun 07 09:57:02 1992 D:\P2\STAT3\ 100 100 0
16 *
17 *MAT LOAD REG.M,CUR+1
18 *MATRIX REG.M
19 *regr(DECA)
20 */// %1
21 *Constant 3559.491
22 *100m 0.543
23 *L_jump 1.361
24 *Shot_put 1.171
25 *Hi_jump 0.922
26 *400m 1.327
27 *
28 *LINCO DECA,REG.M(Pred)
29 *

```

Since the only column of REG.M is labelled by %1, it is possible to select any variable for saving predicted values of the regression model when LINCO is employed. Thus LINCO DECA,REG.M(Pred) will compute the linear combinations indicated by REG.M and store them as values of 'Pred'. If 'Pred' is an old variable in DECA, its values are overwritten. Otherwise, a new single precision (4 byte) variable will be created in DECA.

*Lags and leads in coefficient files

By introducing a special #lag column in the coefficient matrix, lagged values of variables may be employed in LINCO. The next example shows how a simple 5 term moving average is defined as a coefficient matrix.

```

31 1 SURVO 84C EDITOR Wed Apr 01 17:39:58 1987 D:\STAT\ 120 80 0
1 *
2 * w=1/5
3 *MATRIX MOV5
4 */// %2 #lag
5 *%1 w -2
6 *%1 w -1
7 *%1 w 0
8 *%1 w 1
9 *%1 w 2
10 *
11 *MAT SAVE MOV5
12 *LINCO HELSINKI,MOV5(Temp,Ave5)
13 *

```

Matrix MOV5 (saved by MAT SAVE MOV5 on line 11) is a 5*2 matrix where column %2 gives the weights (here all equal to $w=1/5$) and column #lag the lag indices (positive values are for leads). Label %1 in each row allows the possibility of computing this type of moving averages for any variable in any data. Thus, in our example, moving averages of 'Temp' in the HELSINKI data will be computed as values of a (new) variable 'Ave5'.

7.8.3 MATRUN operations

Series of matrix operations can be performed automatically in several ways. The `sucro` technique and serial execution of commands (see 2.7) provide such possibilities. A third alternative is to use the Survo matrix interpreter itself as a tool for making chains of matrix operations and programs for tasks related to multivariate analysis and linear models. Such programs consist of consecutive `MAT` operations and they are usually saved in text files (with extension `.MTX`) in the subdirectory `.\M` of Survo.

To run a matrix program of the subdirectory `.\M`, a command of the form `MATRUN <program>, <list_of_parameters>` has to be invoked in the Survo editor.

The edit file `INDEX` of subdirectory `.\M` lists the `MATRUN` programs currently available. By loading this edit field (by `LOAD .\M\INDEX`), more information is obtained about each `MATRUN` program. In fact, by activating `MATRUN <program>?`, the matrix program in question temporarily displays instructions on the screen.

Some of the multivariate methods in Survo (like principal components and factor analysis) are based at least partially on `MATRUN` programs. Instructions for those `MATRUN` programs will be given in connection with the method in question. Making of `MATRUN` programs will be described to some extent in the section of the Survo matrix interpreter (See also `MATRUN?`).

As an example of a `MATRUN` program, we consider an optimal projection of a point set in an n -dimensional space to a p -dimensional subspace. This is a variant of the principal component analysis. If $p=2$, it corresponds to the biplot technique of representing multivariate data in two dimensions.

The `MATRUN` program `COMPRESS` is available for the task and we acquire information about it by activating `MATRUN COMPRESS?` as follows:

```

17  1 SURVO 84C EDITOR Sat Apr 04 10:19:38 1987      D:\P2\STAT3\ 120  80  0
1  *
2  *MATRUN COMPRESS?
MATRUN COMPRESS,A,AP,p
Let A be an m*n (m>=n) matrix representing m points in an n dimensional
space. MATRUN COMPRESS generates the best approximation of the
point set in p dimensions by centering, orthogonal rotation and projection.
It gives the result as an m*p matrix AP.
Method: Let AC be the centered form of A (sums of columns=0) and
AC=U*D*V' the singular value decomposition of AC. Then AP=U*DP
where DP is the submatrix of p first columns in the diagonal
matrix D of singular values.
      parameter                default
input:  matrix to be approximated  %1=A
output: approximation              %2=AP
input:  dimension of approximation %3=2
16  *
Press any key!
18  *

```

To demonstrate the performance of `MATRUN COMPRESS`, we consider a two-dimensional set of 4 points forming a square in a three dimensional space. The

points of this square are originally on the plane $Z=4$ in the XYZ space (represented as matrix `SQUARE` below). We make the problem more difficult by an arbitrary three-dimensional rotation `T`. Thus we shall apply `MATRUN COMPRESS` to $A=SQUARE*T$ in order to see whether it reveals the true constellation of points.

In the next display, matrix `A` is generated by `MAT` operations:

```

14 1 SURVO 84C EDITOR Sat Apr 04 11:29:20 1987 D:\P2\STAT3\ 120 80 0
1 *
2 *MATRIX SQUARE
3 */// X Y Z
4 *P1 1 1 4
5 *P2 1 2 4
6 *P3 2 1 4
7 *P4 2 2 4
8 *
9 *MATRIX T ///
10 *1 3 4
11 *5 2 7
12 *6 8 0
13 *
14 *MAT SAVE SQUARE / Saving the original matrix
15 *MAT SAVE T / Saving an arbitrary 3*3 matrix T
16 *MAT GRAM-SCHMIDT DECOMPOSITION OF T TO T!,U / Orthonormalizing T
17 *MAT A=SQUARE*T / Rotating SQUARE
18 *MAT LOAD A,20
19 * Hidden square in three-dimensional space
20 *MATRIX A
21 *SQUARE*T
22 */// 1 2 3
23 *P1 3.81000 1.81597 -0.43143
24 *P2 4.44500 1.10758 -0.12327
25 *P3 3.93700 2.30518 0.43143
26 *P4 4.57200 1.59680 0.73960
27 *

```

Activation of `MATRUN COMPRESS,A,A2,2` gives `A2` as a two-dimensional approximation of `A`:

```

18 1 SURVO 84C EDITOR Sun Jun 07 14:00:32 1992 D:\P2\STAT3\ 150 100 0
27 *
28 *MATRUN COMPRESS,A,A2,2
29 *MATRIX A2
30 *Compressed_A
31 */// svd1 svd2
32 *P1 0.45943 0.53752
33 *P2 0.53752 -0.45943
34 *P3 -0.53752 0.45943
35 *P4 -0.45943 -0.53752
36 *Loss of precision in percentages=0
37 *.....
38 *ROTATION=OBLIMIN
39 *ROTATE A2,2,CUR+1
40 *Rotated factor matrix AFACT.M=A2*inv(TFACT.M)'
41 * svd1 svd2 Sumsqr
42 *P1 0.000 0.707 0.500
43 *P2 0.707 -0.000 0.500
44 *P3 -0.707 -0.000 0.500
45 *P4 0.000 -0.707 0.500
46 *Sumsqr 1.000 1.000 2.000
47 *

```

Results of `COMPRESS` are displayed on lines 29-36. The message on line 36 indicates that the point set is actually at most two-dimensional. By making a rotation according to the direct Oblimin method of factor analysis, it is shown that the 4 points presented by `A2` form a square. Although the Oblimin meth-

od is intended for 'oblique' rotations, we get an orthogonal rotation matrix `TFACT.M` in this case. One can see that by computing `TFACT.M*TFACT.M'` which proves to be `I`.

7.8.4 Principal components

Two `MATRUN` operations are available for making an analysis of principal components from standardized variables (with unit variances).

`MATRUN PCOMP , <data_matrix> , f , <pcomp_matrix> , <pcoeff>`
uses `<data_matrix>` saved in a matrix file (e.g. by `MAT SAVE DATA`) and computes `f` first principal components by singular value decomposition of `<data_matrix>`. The results are saved as two `m*f` matrices (`m` is number of variables):

`<pcomp_matrix>` principal component loadings (correlations)
`<pcoeff>` principal component score coefficients

and displayed below the activated line.

`MATRUN PCOMP` without any parameters is equivalent to
`MATRUN PCOMP , X , f , PCOMP.M , PCOEFF.M`
where `f=m/3`.

Another `MATRUN` operation for principal components is
`MATRUN PCOMPR , <corr> , <msn> , f , <pcomp_matrix> , <pcoeff>` .

It uses a correlation matrix `<corr>` and a matrix of means and standard deviations `<msn>` (like `CORR.M` and `MSN.M` given by the `CORR` operation) as the basis for computations. In this case, the principal components are computed from the spectral decomposition of `<corr>`. The results are same as in `MATRUN PCOMP`.

The default setup `MATRUN PCOMPR` without parameters corresponds to
`MATRUN PCOMPR , CORR.M , MSN.M , f , PCOMP.M , PCOEFF.M`
where `f=m/3`.

The `<pcoeff>` matrix can be used for computing of principal component scores for any set of observations by the `LINCO` operation.

We have already computed principal components for 12 variables in the `DECA` data from the correlation matrix `CORR.M` by a series of `MAT` operations. The same analysis can now be made in either of the two ways as presented below.

We save the data as a matrix `X` and use `MATRUN PCOMP`:

```

13 1 SURVO 84C EDITOR Sat Apr 04 14:07:36 1987 D:\STAT\ 120 80 0
1 *
2 *MAT SAVE DATA DECA TO X / MASK=--AAAAAAAAAAAA
3 *MATRUN PCOMP_
4 *
5 *MATRIX !L
6 *Variances_of_principal_components
7 */// PCOMP1 PCOMP2 PCOMP3 PCOMP4
8 *Variance 3.723388 2.035237 1.360654 1.151512
9 *
10 *MATRIX PCOMP.M
11 *Principal_components
12 */// PCOMP1 PCOMP2 PCOMP3 PCOMP4
13 *100m -0.12816 0.83642 -0.00815 0.15782
14 *L_jump -0.00992 0.40515 -0.64153 -0.30405
15 *Shot_put 0.84273 0.07682 0.06674 0.11749
16 *Hi_jump 0.33564 -0.48502 -0.32019 -0.14217
17 *400m -0.47815 0.53732 0.23010 -0.33683
18 *Hurdles 0.15273 0.63906 -0.10293 -0.24942
19 *Discus 0.84701 0.06759 -0.12843 0.13626
20 *Pole_vlt -0.35474 0.03040 -0.29487 0.68117
21 *Javelin 0.08661 -0.31513 -0.52890 -0.45301
22 *1500m -0.58976 -0.33353 0.46695 -0.31092
23 *Height 0.80838 0.10857 0.33820 -0.19945
24 *Weight 0.88318 0.07353 0.25084 -0.01554
25 *
26 *Use PCOMP.M for factor rotation etc.
27 *and PCOEFF.M for scores by LINCO <data>,PCOEFF.M(P1,P2,...)
28 *

```

or compute the correlation matrix CORR.M (and MSN.M) and then use MATRUN PCOMPR:

```

14 1 SURVO 84C EDITOR Sat Apr 04 14:11:22 1987 D:\STAT\ 120 80 0
1 *
2 *CORR DECA / MASK=--AAAAAAAAAAAA-
3 *MATRUN PCOMPR_
4 *
5 *MATRIX !L
6 *Variances_of_principal_components
7 */// PCOMP1 PCOMP2 PCOMP3 PCOMP4
8 *Variance 3.723388 2.035237 1.360654 1.151512
9 *
10 *MATRIX PCOMP.M
11 *Principal_components
12 */// PCOMP1 PCOMP2 PCOMP3 PCOMP4
13 *100m 0.12816 0.83642 0.00815 0.15782
14 *L_jump 0.00992 0.40515 0.64153 -0.30405
15 * ...

```

In both cases, LINCO DECA, PCOEFF.M(P1, P2, P3) would compute values of three first principal components for (selected) observations of DECA as three (new) variables P1,P2,P3.

7.8.5 Factor analysis



Factor analysis is performed by employing Survo operations CORR, FACTA, ROTATE, LINCO and MATRUN operations PFACT, FCOEFF and FTCEFF. If desired, a special sucro /FACTOR connects the main steps of factor analysis automatically.

Factor analysis
RELIAB?

In following, we first give a summary of a sequence of operations. Thereafter the most important operations FACTA and ROTATE are described in detail.

The typical steps of factor analysis in Survo are:

1. Compute the correlations of selected variables and observations by the CORR operation. This gives the correlation matrix CORR.M and the matrix MSN.M of means and standard deviations.

2. If one likes to save these results more permanently, they should be copied by MAT R=CORR.M and MAT M=MSN.M, for example.

3. Initial solution:

Select the number of factors (say k) and activate

FACTA CORR.M,k,L.

This gives the maximum likelihood solution as the factor matrix FACT.M with k factors. The result is written from line L onwards. Other estimation criteria can be set by the METHOD specification (see FACTA?).

Another alternative:

MATRUN PFACT,R,k

gives the principal axes solution as factor matrix PFACT.M. The difference in estimated and computed communalities is displayed and the computed communalities are placed on the diagonal of R.

MATRUN PFACT,R,k can be reactivated several times until the process has converged. The communalities and efficiencies of the factors can be computed afterwards by MATRUN SUM2,PFACT.M.

4. To rotate the maximum likelihood solution FACT.M (or similarly the principal axes solution PFACT.M), activate

ROTATE FACT.M,k

This gives the Varimax solution AFACT.M and the corresponding rotation matrix TFACT.M with k factors. Other methods (graphical rotation etc.) can be selected by the ROTATION specification (See ROTATE?).

5. The factor scores are computed in two steps. At first, the factor score loadings are computed by

MATRUN FCOEFF,AFACT.M,M (M=means and std.devs)

This gives the matrix of coefficients FCOEFF.M.

If an oblique rotation has been used, the loadings for the factor scores are computed by MATRUN FTCEFF,AFACT.M,TFACT.M,M.

6. Finally, factor scores are obtained by

```
LINCO <data>,FCOEFF.M(F1,F2,...)
```

where <data> is the original (or corresponding) Survo data (file) and F1, F2, ... are (possibly new) variables for factor scores. The factor scores can be analyzed like any other data in Survo.

7. As a crude form of confirmatory factor analysis, the (rotated) factor matrix can be compared to a given theoretical factor matrix or two factor matrices obtained on different data sets can be compared to each other by **MATRUN TRANS A1 , A2** or **MATRUN SYMTRANS A1 , A2**. The former one is used for oblique factor structures and the latter for orthogonal ones.

The next exhibit presents the steps 1-4 of factor analysis on our DECA data as a result of the /FACTOR sucro:

```
22 1 SURVO 84C EDITOR Sun Jun 07 16:39:53 1992 D:\P2\STAT3\ 100 100 0
1 *
2 *MASK=--AAAAAAAAAAAA-
3 */FACTOR DECA,3
4 *CORR DECA / Correlation matrix saved as CORR.M
5 *FACTA CORR.M,3,END+2 / Factor matrix saved as FACT.M
6 *ROTATE FACT.M,3,END+2 / Rotated factor matrix saved as AFACT.M
7 *
8 *Factor analysis: Maximum Likelihood (ML) solution
9 *Factor matrix
10 *
11 *      F1      F2      F3      h^2
11 *100m   0.997  0.022  0.001  0.995
12 *L_jump  0.174 -0.021 -0.088  0.038
13 *Shot_put -0.043  0.689 -0.405  0.641
14 *Hi_jump -0.414  0.105 -0.276  0.259
15 *400m   0.461 -0.173  0.505  0.497
16 *Hurdles 0.312  0.251  0.107  0.172
17 *Discus  0.000  0.668 -0.519  0.715
18 *Pole_vlt 0.063 -0.344 -0.096  0.131
19 *Javelin -0.220 -0.051 -0.183  0.085
20 *1500m  -0.285 -0.363  0.647  0.631
21 *Height  -0.131  0.967  0.119  0.967
22 *Weight  -0.102  0.886 -0.156  0.820
23 *
24 *Rotated factor matrix AFACT.M=FACT.M*TFACT.M
25 *
26 *      F1      F2      F3      Sumsqr
26 *100m   0.806 -0.109 -0.577  0.995
27 *L_jump  0.087 -0.049 -0.168  0.038
28 *Shot_put -0.225  0.662 -0.391  0.641
29 *Hi_jump -0.489  0.141  0.003  0.259
30 *400m   0.656 -0.199  0.164  0.497
31 *Hurdles 0.332  0.215 -0.126  0.172
32 *Discus  -0.259  0.627 -0.505  0.715
33 *Pole_vlt -0.029 -0.354 -0.070  0.131
34 *Javelin -0.289 -0.033 -0.014  0.085
35 *1500m  0.125 -0.280  0.733  0.631
36 *Height  0.030  0.982  0.048  0.967
37 *Weight  -0.113  0.880 -0.182  0.820
38 *Sumsqr  1.668  2.894  1.389  5.952
39 *
40 *Rotation matrix saved as TFACT.M
41 *Factors are orthogonal (RFACT.M=I).
42 *
```

The selection of variables is indicated by a MASK specification on line 2 and the call for the /FACTOR sucro is written on line 3. This is all information in the edit field before activation of /FACTOR.

The second parameter in the /FACTOR call (3) is the number of factors to be extracted. If it is not given, the sucro finds a suitable value by computing

the eigenvalues of the correlation matrix.

By activating the `/FACTOR DECA, 3` command (on line 3), the correlations are computed by `CORR` (on line 4), the initial solution according to the maximum likelihood method with 3 factors is found by `FACTA` (on line 5) and finally a Varimax rotation is performed by `ROTATE` (on line 6). The results appear from line 8 onwards.

FACTA operation

`FACTA <correlation_matrix>, <number_of_factors>, L`
 computes an initial factor solution according to the maximum likelihood (ML) principle. The algorithm presented by K.G.Jöreskog is employed. (See e.g. Chapter 7 in 'Statistical Methods for Digital Computers', Volume III, edited by Enslein, Ralston and Wilf, Wiley 1977.)

Other estimation principles can be selected by

`METHOD=ULS` Unweighted Least Squares, corresponding to the Minres method of Harman or the `MATRUN PFACT` routine,
`METHOD=GLS` Generalized Least Squares.

Thus, the default is

`METHOD=ML` Maximum Likelihood.

Appropriate statistics for testing the number of factors are obtained by giving the specification `N=<number of observations>`. The results will be displayed from the (optional) line `L` onwards. The factor matrix is saved in the matrix file `FACT.M`.

ROTATE operation

`ROTATE <factor_matrix>, <number_of_factors>, L`
 with an optional `ROTATION` specification makes an orthogonal or oblique factor rotation. The default method is the orthogonal Varimax rotation. The following results are obtained as matrix files:

`AFACT.M` Rotated factor matrix **A**
`TFACT.M` Rotation matrix **T**
`RFACT.M` Factor correlation matrix **R** (=I in orthogonal rotations)

If `RESULTS<=70`, only **A** will be written to the edit field (from line `L`). Otherwise, also **T** and **R** are printed to the edit field.

In case of an orthogonal rotation, $\mathbf{A}=\mathbf{FT}$ ($\mathbf{F}=\langle\text{factor matrix}\rangle$) and in an oblique rotation $\mathbf{A}=\mathbf{F}(\mathbf{T}')^{-1}$.

Optionally, an initial rotation matrix can be given by `T=<matrix_file>`. Thus results of analytic rotations can be used as starting points for a more subjective graphical solution.

The main approach in `ROTATE` is graphical rotation in the graphics mode (specified by `MODE=EGA` or `MODE=VGA`; default is given by system parameter `videomode` in `SURVO.APU`).

Graphical rotation

By entering the specification

```
ROTATION=GRAPHICAL
```

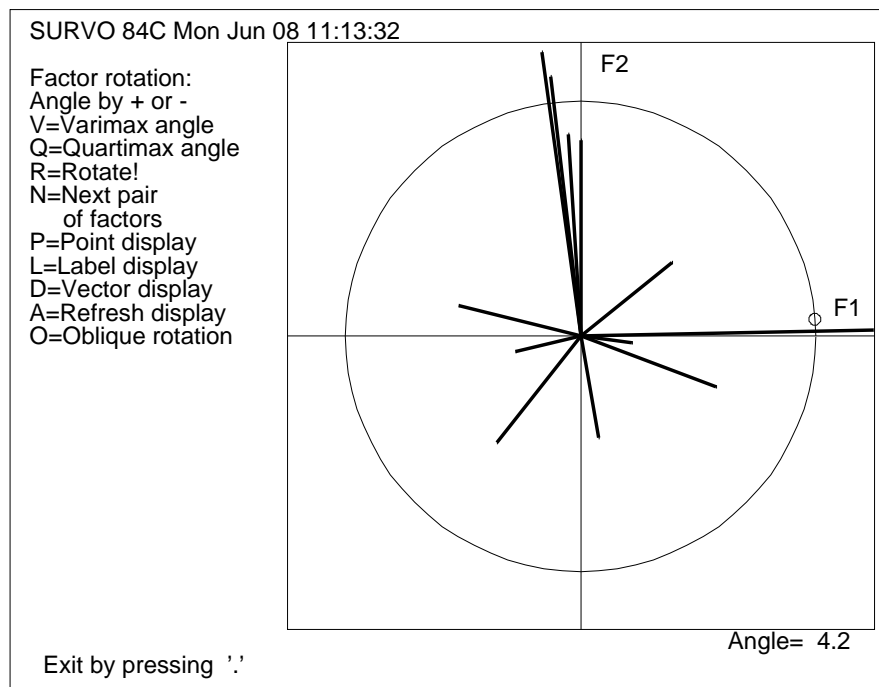
a stepwise graphical rotation is performed on the graphic screen. Both orthogonal and oblique rotations are made as a sequence of two-dimensional rotations.

At first, an orthogonal procedure is suggested and Varimax and Quartimax criteria can be used as an aid in selection of proper rotation angles. When necessary, the user can proceed to the oblique case and indicate the new positions of the selected axes stepwise. The cursor is moved in the display by the **+** and **-** keys. Rotation takes place by **R**. More instructions are give while staying in ROTATE.

Graphical rotation for the 3-factor ML solution of the DECA data (done in the previous example) could be started by the ROTATE command:

```
16 1 SURVO 84C EDITOR Sat Jun 13 09:22:52 1992 D:\P2\STAT3\ 150 100 0
43 *
44 *ROTATE FACT.M,3 / ROTATION=GRAPHICAL
45 *MODE=VGA
46 *
```

This gives the following display:



In the display, the subspace spanned by the two first factors F1 and F2 is presented. The variables appear as vectors. To indicate a new position of the X axis, a cursor (small circle) can be moved on a circle having the radius 0.8 by

the **[+]** and **[-]** keys. In this case, we have pressed the key **[V]** giving the optimal rotation angle (4.2 degrees) according to the Varimax method. By pressing **[R]**, the current angle of rotation is adopted and the factor matrix as well as the display is updated.

Instead of vectors, the variables can also be indicated in the graph by points (by **[P]**) or by the names of the variables (by **[L]**).

The next pair of axes is selected by **[N]**. The order of pairs in case of 3 factors is F1,F2 F1,F3 F2,F3 F1,F2 etc.

For a stepwise graphical oblique rotation, the **[O]** key is pressed. Then the display is replaced by a more general one where both the position of the X axis and Y axis can be selected freely.

In all cases, the rotation is finished by pressing the **[.]** key. Then the current status of the factor matrix is saved as **AFACT.M** and the rotation matrix as **TFACT.M**.

Analytical criteria in rotation

The ROTATE operation supports following analytic methods selected by the ROTATION specification.



ROTATE?

ROTATION=VARIMAX

performs Varimax rotation (default), See Harman: Modern Factor Analysis, 2nd ed. pp.304-313.

ROTATION=COS

gives the oblique Cosine rotation of Ahmavaara and Markkanen by Mustonen's determinant criterion and algorithm.

Let r be the number of factors. The r new factor axes will coincide with the r most orthogonal original variables in the factor space. The solution is found by maximizing the 'volume' of the simplex spanned by r variables.

ROTATION=OBLIMIN, <Delta>, <Max_iter>

computes the direct Oblimin rotation (oblique) by Jennrich and Sampson (Harman: Modern Factor Analysis, 2nd ed., pp.334-341).

Parameters **<Delta>** and **<Max_iter>** are optional. **<Delta>** determines how oblique the factors will be. **<Delta>=0** (default) gives the most oblique factors. On negative values of **<Delta>**, factors become more orthogonal. **<Max_iter>** (default 30) gives the maximum number of iteration rounds.

Thurstone's box problem

The following experiment is available as a Survo tutorial /BOXSTONE. The exhibits below are merely listings produced by this surcro in the edit field.

Presentation of the problem and creation of the data file:

```

1 1 SURVO 84C EDITOR Sat Jun 13 16:48:49 1992 D:\P2\STAT3\ 200 80 0
1 *
2 * Thurstone's box problem
3 *
4 *The famous box problem will be studied here as follows:
5 *We generate a sample of 100 boxes where instead of the basic
6 *dimensions, say X,Y,Z, various derived variables like X+Y, XY etc.
7 *are analyzed.
8 *To make the situation more realistic, each variable will also have
9 *an additive error term which is normally distributed.
10 *The random numbers generated for our experiment are time dependent.
11 *Thus by repeating this tutorial several times, you may study the
12 *sampling variation.
13 *
14 *We start by creating a file BOX for 25 variables and define the
15 *latent' variables X,Y,Z:
16 *FILE CREATE BOX,100,25
17 * Data for Thurstone's box problem
18 *
19 *FIELDS:
20 *1 N 4 X
21 *2 N 4 Y
22 *3 N 4 Z
23 *END

```

Generating the data:

```

1 1 SURVO 84C EDITOR Sat Jun 13 16:51:26 1992 D:\P2\STAT3\ 200 80 0
24 *
25 *Then we initialize the data file by 100 missing observations:
26 *FILE INIT BOX,100
27 *and compute the latent variables X,Y,Z by the following VAR operation:
28 *VAR X,Y,Z TO BOX
29 *X=10*(1+rnd(0))
30 *Y=10*(1+rnd(0))
31 *Z=10*(1+rnd(0))
32 *Hence X,Y,Z are independent and each of them is uniformly distributed
33 *on the interval (10,20).
34 *.....
35 *We assume now that X,Y and Z cannot be measured directly, but there are
36 *certain derived variables available:
37 *X1=X+2*probit(rnd(0)) / error is N(0,4)
38 *Y1=Y+2*probit(rnd(0))
39 *Z1=Z+2*probit(rnd(0))
40 *X2=X*X+50*probit(rnd(0))
41 *Y2=Y*Y+50*probit(rnd(0))
42 *Z2=Z*Z+50*probit(rnd(0))
43 *SXY=X+Y+2*probit(rnd(0))
44 *SXZ=X+Z+2*probit(rnd(0))
45 *SYZ=Y+Z+2*probit(rnd(0))
46 *PXY=X*Y+30*probit(rnd(0))
47 *PXZ=X*Z+30*probit(rnd(0))
48 *PYZ=Y*Z+30*probit(rnd(0))
49 *VAR X1,Y1,Z1,X2,Y2,Z2,SXY,SXZ,SYZ,PXY,PXZ,PYZ TO BOX
50 *

```

```

1 1 SURVO 84C EDITOR Sat Jun 13 16:51:26 1992 D:\P2\STAT3\ 200 80 0
50 *.....
51 *We compute still 6 more variables:
52 *S2XY=sqrt(X*X+Y*Y)+2*probit(rnd(0))
53 *S2XZ=sqrt(X*X+Z*Z)+2*probit(rnd(0))
54 *S2YZ=sqrt(Y*Y+Z*Z)+2*probit(rnd(0))
55 *SXYZ=X+Y+Z+5*probit(rnd(0))
56 *S2XYZ=sqrt(X*X+Y*Y+Z*Z)+3*probit(rnd(0))
57 *PXYZ=X*Y*Z+1000*probit(rnd(0))
58 *VAR S2XY,S2XZ,S2YZ,SXYZ,S2XYZ,PXYZ TO BOX
59 *.....
60 *The complete data set has now been generated and we may start the
61 *analysis.

```

Selection of variables and the solution by the /FACTOR sucro:

The eigenvalues of the correlation matrix (listed on line 76) have been obtained by the command

MAT SPECTRAL DECOMPOSITION OF CORR.M TO S,D.

This command has been removed later in the /FACTOR sucro.

```

1 1 SURVO 84C EDITOR Sat Jun 13 17:00:30 1992 D:\P2\STAT3\ 200 80 0
62 *We shall omit the latent variables X,Y,Z and select variables as
63 *follows:
64 *FILE ACTIVATE BOX
65 *(Variables are selected according to MASK=---AAAAAAAAAAAAAAAAAAAA )
66 *The factor analysis is performed automatically by the sucro /FACTOR.
67 *It selects the number of factors (which should be 3 in this case)
68 *by inspecting the eigenvalues of the correlation matrix.
69 *By default, the /FACTOR sucro finds the initial solution by the maximum
70 *likelihood method (FACTA operation) and the final rotated solution
71 *by the Varimax method (ROTATE operation).
72 */FACTOR BOX / This sucro writes and activates the commands:
73 *CORR BOX / Correlation matrix saved as CORR.M
74 *Eigenvalues of the correlation matrix CORR.M:
75 * ev1 ev2 ev3 ev4 ev5 ev6 ev7 ev8 ev9 ev10 ev11
76 * 7.58 2.94 2.61 0.71 0.59 0.51 0.45 0.43 0.37 0.30 0.25
77 *FACTA CORR.M,3,END+2 / Factor matrix saved as FACT.M
78 *ROTATE FACT.M,3,END+2 / Rotated factor matrix saved as AFACT.M

```

Initial solution as a result of /FACTOR:

```

1 1 SURVO 84C EDITOR Sat Jun 13 17:06:00 1992 D:\P2\STAT3\ 200 80 0
79 *Here are the results given by the /FACTOR sucro:
80 *Factor analysis: Maximum Likelihood (ML) solution
81 *Factor matrix
82 *      F1      F2      F3      h^2
83 *X1      0.343  0.314  0.611  0.590
84 *Y1      0.541 -0.600 -0.140  0.673
85 *Z1      0.562  0.453 -0.353  0.646
86 *X2      0.303  0.212  0.696  0.622
87 *Y2      0.606 -0.647 -0.082  0.792
88 *Z2      0.591  0.475 -0.448  0.776
89 *SXY      0.672 -0.336  0.487  0.801
90 *SXZ      0.677  0.499  0.179  0.740
91 *SYZ      0.757 -0.110 -0.507  0.843
92 *PXY      0.729 -0.353  0.375  0.796
93 *PXZ      0.676  0.562  0.190  0.809
94 *PYZ      0.743 -0.074 -0.432  0.745
95 *S2XY      0.647 -0.309  0.388  0.664
96 *S2XZ      0.619  0.529  0.177  0.694
97 *S2YZ      0.750 -0.126 -0.316  0.679
98 *SXYZ      0.592  0.176  0.135  0.400
99 *S2XYZ      0.691  0.038  0.127  0.495
100 *PXYZ      0.704 -0.052  0.031  0.499
101 *

```

Rotated factor matrix as a result of /FACTOR:

```

1 1 SURVO 84C EDITOR Sat Jun 13 17:07:53 1992 D:\P2\STAT3\ 200 80 0
102 *Rotated factor matrix AFACT.M=FACT.M*TFACT.M
103 *      F1      F2      F3 Sumsqr
104 *X1      0.027  0.013  0.767  0.590
105 *Y1      0.797  0.128 -0.146  0.673
106 *Z1      0.002  0.791  0.142  0.646
107 *X2      0.083 -0.113  0.776  0.622
108 *Y2      0.878  0.111 -0.092  0.792
109 *Z2     -0.002  0.876  0.087  0.776
110 *SXY      0.724 -0.028  0.525  0.801
111 *SXZ      0.077  0.573  0.637  0.740
112 *SYZ      0.544  0.725 -0.142  0.843
113 *PXY      0.767  0.066  0.451  0.796
114 *PXZ      0.030  0.596  0.672  0.809
115 *PYZ      0.513  0.690 -0.072  0.745
116 *S2XY     0.681  0.028  0.448  0.664
117 *S2XZ     0.016  0.552  0.624  0.694
118 *S2YZ     0.564  0.601  0.002  0.679
119 *SXYZ     0.263  0.384  0.428  0.400
120 *S2XYZ     0.431  0.383  0.404  0.495
121 *PXYZ     0.500  0.403  0.294  0.499
122 *Sumsqr   4.379  4.249  3.633 12.261
123 *We should have 3 almost equally strong factors which clearly
124 *describe the true structure.
125 *

```

Checking the result by computing the factor scores:

```

1 1 SURVO 84C EDITOR Sat Jun 13 17:09:46 1992 D:\P2\STAT3\ 200 80 0
126 *To confirm this result, we compute the factor scores and their
127 *correlations with the latent variables X,Y,Z as follows:
128 *At first the coefficients for the scores are computed by
129 *MATRUN FCOEFF,AFACT.M,MSN.M
130 *
131 *The factor score coefficients are saved in FCOEFF.M
132 *Factor scores are computed by LINCO <data>,FCOEFF.M(F1,F2,...)
133 *
134 *LINCO BOX,FCOEFF.M(F1,F2,F3)
135 *
136 *CORR BOX,END+1 / VARS=X,Y,Z,F1,F2,F3
137 *Means, std.devs and correlations of BOX N=100
138 *Variable Mean Std.dev.
139 *X      14.97812  2.746385
140 *Y      15.12966  2.933734
141 *Z      15.17811  2.957628
142 *F1      0.000000  0.966335
143 *F2      0.000000  0.963751
144 *F3      0.000000  0.955565
145 *Correlations:
146 *      X      Y      Z      F1      F2      F3
147 * X      1.0000 -0.0256  0.0225  0.1991 -0.0339  0.9379
148 * Y     -0.0256  1.0000 -0.0425  0.9447  0.0837 -0.1753
149 * Z      0.0225 -0.0425  1.0000 -0.0745  0.9561  0.1226
150 * F1     0.1991  0.9447 -0.0745  1.0000  0.0243  0.0150
151 * F2    -0.0339  0.0837  0.9561  0.0243  1.0000  0.0209
152 * F3     0.9379 -0.1753  0.1226  0.0150  0.0209  1.0000
153 *
154 *There should now be high correlations (pairwise in some order)
155 *between X,Y,Z and F1,F2,F3 variables
156 *which indicates that factor analysis has revealed the true structure
157 *of the data.
158 *

```

7.8.6 Canonical correlations and variables

`CANON <data>,L`

computes canonical correlations and variables for two sets of variables. The first set (X variables) should be denoted by 'X' masks and the second set (Y variables) by 'Y' masks. `IND`, `CASES` and `SELECT` specifications are allowed for selection of observations.

`CANON` gives the results from line L (optional) onwards. Also Bartlett's χ^2 statistics for testing nullity of canonical correlations are printed with their *P* values.

The following matrix files are produced by `CANON`:

| | |
|-----------------------|--|
| <code>LCAN.M</code> | vector of canonical correlations |
| <code>XCAN.M</code> | correlations of canonical variables with X variables |
| <code>YCAN.M</code> | correlations of canonical variables with Y variables |
| <code>XCOEFF.M</code> | coefficients for the first set of canonical variables |
| <code>YCOEFF.M</code> | coefficients for the second set of canonical variables |

Values of canonical variables are computed afterwards by using `XCOEFF.M` and `YCOEFF.M` in the `LINCO` operation. For example,

`LINCO <data>,XCOEFF.M(C1,C2)`

forms the two first canonical variables from the first set as (new) variables `C1,C2`.

The calculations needed are based on orthogonalized data matrices (not on correlations). The maximum number of data values in each data matrix is 8192.

Optionally, confounding variables, whose effects on X and Y variables are to be removed by linear regression, may be included. Confounding variables are to be activated by 'Z'. All the results obtained from `CANON` are then related to the transformed X and Y data sets.

`CANON` creates several work matrices (names initiated by `&`). These are automatically deleted, unless `RESULTS=100` is given. All the work matrices are on the current data disk.

The `MATRUN` operations `CANON2` and `CANON3` work on the same basis as `CANON`. See, for example, `MATRUN CANON2?`.

As an application, we consider the `DECA` data again. We are interested in the relations between the first-day events and second-day events in decathlon. In this case, `CANON` gives following results:

```

13 1 SURVO 84C EDITOR Sun Apr 05 08:41:22 1987 D:\P2\STAT3\ 120 80 0
1 *
2 *MASK=A-XXXXXXXXXX---
3 *CANON DECA,5_
4 *
5 *Canonical analysis on DECA:
6 *Correlation CHI^2 P df (LCAN.M)
7 * 1 0.7894 62.738 0.99996 25
8 * 2 0.5284 22.231 0.86413 16
9 * 3 0.3306 8.6421 0.52905 9
10 * 4 0.2823 3.8400 0.57191 4
11 * 5 0.0972 0.3941 0.46987 1
12 *Coefficients (LINCO) for canonical variables saved in XCOEFF.M,YCOEFF.M
13 *
14 *Correlations of canonical variables with X variables XCAN.M
15 * CAN1 CAN2 CAN3 CAN4 CAN5
16 *100m -0.218 0.729 0.124 0.633 -0.069
17 *L_jump -0.124 0.566 -0.731 -0.301 -0.200
18 *Shot_put -0.892 -0.400 -0.130 0.121 -0.112
19 *Hi_jump -0.250 -0.168 -0.140 -0.424 0.843
20 *400m 0.424 0.174 -0.453 0.762 0.067
21 *
22 *Correlations of canonical variables with Y variables YCAN.M
23 * CAN1 CAN2 CAN3 CAN4 CAN5
24 *Hurdles -0.226 0.555 -0.627 0.496 -0.045
25 *Discus -0.928 -0.336 -0.090 -0.075 -0.108
26 *Pole_vlt 0.191 0.299 0.193 -0.232 -0.885
27 *Javelin -0.000 -0.139 -0.540 -0.808 0.192
28 *1500m 0.806 -0.497 -0.166 0.251 -0.111
29 *

```

Although only the first pair of canonical variables seems to be significant, we have computed values for all five canonical variables in the X set by the LINCO operation. New variables are C1,C2,C3,C4 and C5. Their correlations with the X variables are then computed to verify the validity of XCAN.M.

```

26 1 SURVO 84C EDITOR Sun Apr 05 10:11:14 1987 D:\P2\STAT3\ 120 80 0
29 *
30 *LINCO DECA,XCOEFF.M(C1,C2,C3,C4,C5)
31 *MASK=---XXXXX-----AAAAA
32 *CORR DECA
33 *MAT LOAD CORR.M,##.###,34_
34 *MATRIX CORR.M
35 *R(DECA)
36 *///
37 *C1 1.000 0.000 -0.000 0.000 -0.000 -0.218 -0.124 -0.892 -0.250
38 *C2 0.000 1.000 0.000 0.000 -0.000 0.729 0.566 -0.400 -0.168
39 *C3 -0.000 0.000 1.000 -0.000 0.000 0.124 -0.731 -0.130 -0.140
40 *C4 0.000 0.000 -0.000 1.000 -0.000 0.633 -0.301 0.121 -0.424
41 *C5 -0.000 -0.000 0.000 -0.000 1.000 -0.069 -0.200 -0.112 0.843
42 *100m -0.218 0.729 0.124 0.633 -0.069 1.000 0.172 -0.028 -0.412
43 *L_jump -0.124 0.566 -0.731 -0.301 -0.200 0.172 1.000 -0.034 -0.003
44 *Shot_put -0.892 -0.400 -0.130 0.121 -0.112 -0.028 -0.034 1.000 0.163
45 *Hi_jump -0.250 -0.168 -0.140 -0.424 0.843 -0.412 -0.003 0.163 1.000
46 *400m 0.424 0.174 -0.453 0.762 0.067 0.456 0.133 -0.304 -0.339
47 *

```

7.8.7 Multiple discriminant analysis



Discriminant analysis

`MATRUN DISCR, <data_matrix>, g`

performs multiple discriminant analysis on data matrix with `g` distinct groups. In `<data_matrix>`, the `g` first columns are 0,1 variables, indicating the group of the observation.

Thus, before using `MATRUN DISCR`, a `<data_matrix>` consisting of test variables and dummy variables indicating classification has to be created. Usually the grouping variables are generated by conditional `VAR` operations, and the observations then are saved in a matrix file by `MAT SAVE DATA`. The maximum size of the entire data matrix is 8192 elements.

The following matrix files are produced by `MATRUN DISCR`:

| | |
|-----------------------|---------------------------------------|
| <code>DCAN.M</code> | Canonical correlations |
| <code>DISCR.M</code> | Loadings of discriminant variables |
| <code>DMEAN.M</code> | Means of groups in discriminant space |
| <code>DSCORE.M</code> | Discriminant scores |

In the following tiny application, we enter the data matrix directly in the edit field. It consists of 11 observations of 3 test variables `X1,X2,X3` and it is divided into 3 groups `A,B,C`:

```

17 1 SURVO 84C EDITOR Sun Apr 05 13:43:35 1987 D:\P2\STAT3\ 100 100 0
1 *
2 *MATRIX Z
3 */// A B C X1 X2 X3
4 *A1 1 0 0 2 2 3
5 *A2 1 0 0 1 2 3
6 *A3 1 0 0 2 1 1
7 *A4 1 0 0 1 1 2
8 *B1 0 1 0 5 5 5
9 *B2 0 1 0 3 2 2
10 *B3 0 1 0 4 3 4
11 *C1 0 0 1 7 0 7
12 *C2 0 0 1 8 1 5
13 *C3 0 0 1 4 2 3
14 *C4 0 0 1 4 0 7
15 *
16 *MAT SAVE Z
17 *MATRUN DISCR,Z,3
18 *

```

Activation of `MATRUN DISCR, Z, 3` on line 17 gives the results

```

17 1 SURVO 84C EDITOR Sun Apr 05 13:46:12 1987 D:\P2\STAT3\ 100 100 0
18 *MATRUN DISCR,Z,3_
19 *
20 *MATRIX DCAN.M
21 *Canonical_correlations
22 */// Discr1 Discr2
23 *sing.val 0.873300 0.728827
24 *
25 *MATRIX DISCR.M
26 *Loadings_of_discriminant_variables
27 */// Discr1 Discr2
28 *X1 -0.67245 0.42748
29 *X2 0.36653 0.92883
30 *X3 -0.26989 0.07356
31 *
32 *MATRIX DMEAN.C
33 *Means_of_groups_in_discriminant_space
34 */// Discr1 Discr2
35 *A 0.29316 -0.27046
36 *B 0.12841 0.47533
37 *C -0.38947 -0.08604
38 *
39 *MATRIX DSCORE.C
40 *Discriminant_scores
41 */// Discr1 Discr2
42 *A1 0.21812 -0.05539
43 *A2 0.31121 -0.11457
44 *A3 0.22226 -0.28551
45 *A4 0.27249 -0.33300
46 *B1 0.09788 0.76578
47 *B2 0.16790 -0.00789
48 *B3 0.07066 0.28141
49 *C1 -0.58198 -0.12628
50 *C2 -0.50775 0.11629
51 *C3 0.03194 0.06297
52 *C4 -0.30271 -0.30381
53 *

```

The observations need not to be ordered in groups because the grouping variables determine classification. Obviously also weights (probabilities) summing up to 1 for each observation are allowed instead of a stringent 0,1 classification.

A more general tool for the discriminant analysis is the DISCR operation by Markku Korhonen (documented in SURVO 84C Contributions No.5, 1992). The DISCR operation works in two phases. Firstly, the classification functions and tests associated with them are computed. Secondly, cases from the original data or from other data are classified according to these functions.

7.8.8 Cluster analysis



CLUSTER <data_file>,L

performs cluster analysis on the selected variables and observations of the Survo <data_file>.

Cluster analysis

The clustering criterion is Wilks' lambda and the stepwise procedure for efficient computation of lambda values is based on the algorithm presented by Pekka Korhonen in his doctoral dissertation "A stepwise procedure for multivariate clustering", Computing Centre, University of Helsinki, Research Reports N:o 7, Helsinki 1979. In the CLUSTER module, the dual procedure of

Korhonen's stepwise method is applied.

For general information on cluster analysis, see e.g. M.R.Anderberg: "Cluster Analysis for Applications", Academic Press, New York and London, 1973.

The active observations of <data_file> are defined by `IND`, `CASES` and `SELECT` specifications. The variables used in the analysis are all active variables in <data_file>, except those activated by 'G' or 'I'.

The stepwise clustering procedure is always based on some initial grouping of observations. The user has to give the number (`g`) of clusters by the `GROUPS=g` specification. `GROUPS=2` is the default.

The initial grouping is given by a variable activated by 'I' and the values of this variable must be integers 1,2, ...,`g`. If the initial grouping of observations is not given (no mask 'I' exists), a random initial grouping based on uniform distribution over 1,2, ...,`g` is applied automatically.

The initial grouping (defined by the 'I' mask variable) can also be incomplete (with missing values or values outside the permitted ones 1,2, ...,`g`). In this case, it is assumed that the user has indicated at least one observation in each group. Then, the initial grouping will be selected on this basis by using the "nearest neighbour" principle in the standardized data matrix \mathbf{X} (with the property $\mathbf{X}'\mathbf{X}=\mathbf{I}$).

The main result of `CLUSTER` is the optimal clustering based on the Wilks' lambda criterion and it is saved in the first variable of <data_file> activated by 'G'. If more 'G' variables exist, the `CLUSTER` module will save as many of the best solutions found, provided that a specification `TRIALS=n` where $n > 1$ is given. The possibility for several trials is important in more complicated cases where different initial groupings may lead to different solutions.

There are no limits for the size of the data file. The highest number of variables and groups depends on the available memory space. Theoretical maximum for the number of variables is 90. However, it is seldom reasonable to take more than 10-20 variables into one cluster analysis.

To speed up the iterative process where the data values are scanned several times, `CLUSTER` saves the active part of the data set in a special file `SURVO.CLU` on the path of the temporary files (defined by the line `tempdisk` in `SURVO.APU`). This file (path) can be replaced by another (on a ramdisk, for example) by giving a specification `TEMPFILE=<filename>`.

In randomizations for initial groupings, the seed number of the random number generator is selected according to current time. To use a fixed generator (in order to have the possibility to repeat an experiment), a specification of the form `SEED=<integer>` can be given.

The next simulation experiment illustrates how `CLUSTER` should be used in practice. It shows how important is to let `CLUSTER` to make several attempts by giving the `TRIALS` specification.

In this experiment, a file `N2` of 100 observations is created by `FILE CREATE` (on lines 6-10) and two samples from a bivariate normal distribution with different means are generated by a `VAR` operation (on lines 12-17).

Three more variables (G1,G2,G3) are created by another VAR on lines 19-20 to store three different groupings. These variables are initialised by 0's.

The cluster analysis is performed by the CLUSTER operation on lines 26-27 and giving the results on lines 28-34. The variables are selected and their tasks in the analysis are declared by the MASK=AAGGG specification (on line 26).

```

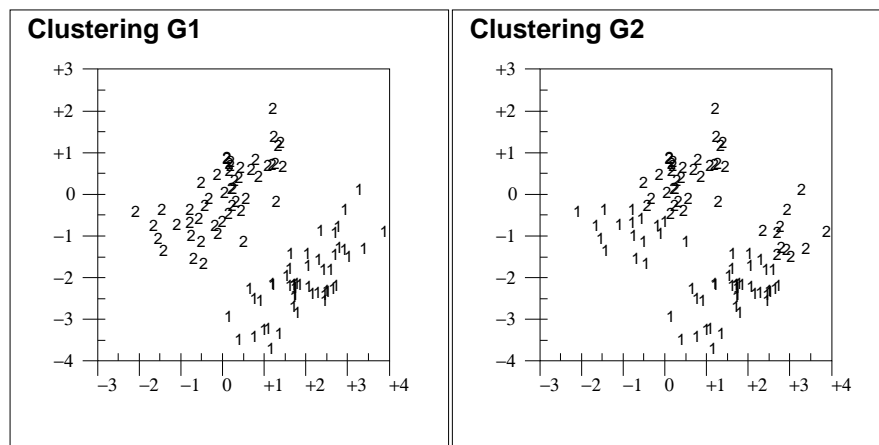
13 1 SURVO 84C EDITOR Tue Mar 03 13:47:37 1989 D:\COMP\ 100 100 0
1 *SAVE CLUSTER
2 * Two samples from a bivariate normal distribution
3 * with different means
4 * but the same covariance matrix are generated:
5 *
6 *FILE CREATE N2,32,10,64,7,100
7 *FIELDS:
8 *1 N 4 X
9 *2 N 4 Y
10 *END
11 *
12 *VAR X,Y TO N2
13 *X=if(ORDER<51)then(X1)else(X2)
14 *Y=if(ORDER<51)then(Y1)else(Y2)
15 *X1=Z1 Y1=r*Z1+s*Z2 r=0.8 s=sqrt(1-r*r)
16 *X2=Z1+2 Y2=r*Z1+s*Z2-2
17 *Z1=probit(rnd(2)) Z2=probit(rnd(2))
18 *
19 *VAR G1:1,G2:1,G3:1 TO N2
20 * G1=0 G2=0 G3=0
21 *
22 * The CLUSTER operation with 10 trials, 2 groups,
23 * and random number generator 2
24 * gives two different solutions:
25 *
26 *MASK=AAGGG TRIALS=10 GROUPS=2 SEED=2
27 *CLUSTER N2,28
28 *Stepwise cluster analysis by Wilk's Lambda criterion
29 *Data N2 N=100
30 *Variables: X, Y
31 *Best clusterings found in 10 trials are saved as follows:
32 * Lambda freq Grouping var
33 * 0.04496 6 G1
34 * 0.14945 4 G2
35 *
36 *
37 *The result can be checked by plotting the graph:
38 *GPLOT N2,X,Y
39 *HEADER=Clustering_G1
40 *POINT=G1 (G2 gives the inferior clustering)
41 *

```

CLUSTER starts by selecting a random partition of observations into 2 groups (GROUPS=2 on line 26). Thereafter the procedure tries to move observations from a group to another and if the criterion value is improved, the observation is really moved. This practice will be continued until no single move improves the criterion value.

In this case, 10 different random starts have been taken (TRIALS=10) and two different solutions (G1,G2) are obtained as seen on lines 33-34.

The first solution G1 having the better criterion value is, of course, the true one. The characteristics of the solutions are clearly revealed by the plots generated by the PLOT scheme on lines 38-40:



7.9 Time series analysis

The Survo functions specially intended for time series are the SER operations for moving averages, differences, cumulative sums, etc., XCORR for auto- and crosscorrelations and FORECAST for automatic forecasting of time series. In addition, there are several other functions which may be helpful in problems related to time series analysis.

Such supporting operations (described elsewhere) are, for example:

1. VAR operation with lags and leads,
2. CORR operation for auto- and cross correlations (after VAR),
3. LINCO operation with a #lag column for moving averages etc.,
4. LINREG and ESTIMATE operations,
5. SMOOTH for semiparametric smoothing,
6. PLOT operation for graphical presentation of time series.

7.9.1 SER operations

SER operations are like VAR operations, but they offer transformations needed in management of time series.

The general form of a SER operation is

SER <new_series>=<operation>(<old_series>,P1,P2,...,Pn) TO <data>
 where <old_series> and <new_series> are variables in <data>.

If <data> is a data file, <new_series> is created automatically when necessary. In data tables, new variables must be defined in advance.

If a sequence of SER operations is activated, the extension TO <data> is not required when the same (current) data file is used.

Moving averages

SER Y=MA(X, <weight_description>) TO <data>

computes moving averages of X to Y according to <weight_description>.

<weight_description> can be any of the three alternatives:

1. List of weights

SER Temp2=MA(Temp, 1, 2, 3, 0, 0)

i.e. $\text{Temp2}[t] = \text{Temp}[t-2] + 2 * \text{Temp}[t-1] + 3 * \text{Temp}[t]$

or

SER Temp3=MA(Temp, 1, 2, 3, *)

being equivalent to

SER Temp3=MA(Temp, 1, 2, 3, 2, 1) (symmetric weights)

Number of weights must be uneven.

2. Name of a weight pattern

SER TempS=MA(Temp, SPENCER)

The weights for SPENCER must be given in the same edit field as a specification

SPENCER=-3, -6, -5, 3, 21, 46, 67, 74, *

3. Weights for polynomial trends

SER Temp4=MA(Temp, P3:21)

fits a cubic (P3) to sets of 21 points. Generally Pp:m implies fitting of a polynomial of degree p with a span of m consecutive points. m must be an odd integer <=89.

In all cases above, the sum of weights is automatically scaled to 1.

SER Y=MAE(X, <weight_description>) TO <data>

works as MA, but provides trend values for the first and last m values of the series as well. MAE can be used in the case of polynomial weights only.

Moving averages in periodical series

SER Y=MA(X, <weight_description>) TO <data> / PERIOD=s

computes periodical moving averages thus using data values

..., X[t-3s], X[t-2s], X[t-s], X[t], X[t+s], X[t+2s], X[t+3s], ...

instead of

..., X[t-3], X[t-2], X[t-1], X[t], X[t+1], X[t+2], X[t+3], ...

The period s is given as an extra specification PERIOD=s.

Another way to enter the period s (say for s=3) is to use the operation

SER Y=MA3(X, <weight_description>) TO <data> .

SER Y=MAE(X, <weight_description>) TO <data> / PERIOD=s

provides smoothed values for the first and last observations as well in case of polynomial weights. Alternative notation is Y=MAE3(X, ... for s=3.

Differences

SER <new_series>=D(<old_series>,L) TO <data>

computes differences $D(t) = X(t) - X(t-L)$ of $X(t) = \text{<old_series>}$. If lag $L=1$, it may be omitted.

Cumulative sums

SER <new_series>=CUM(<old_series>,L) TO <data>

computes the cumulative sum(s)

$Y(t) = X(t)$ for $t = 1, 2, \dots, L$

$Y(t) = Y(t-L) + X(t)$ for $t = L+1, L+2, \dots$

where $Y(t) = \text{<new_series>}$, $X(t) = \text{<old_series>}$.

General linear combinations

SER Y=MS(X,W1,W2,...,Wm) TO <data>

computes linear combinations of lagged values of X as

$$Y[t] = W1 * X[t-m+1] + W2 * X[t-m+2] + \dots + Wm * X[t]$$

without rescaling weights W1, ..., Wm.

By giving one of the weights with a terminating 'T' (say WkT),

$$Y[t] = W1 * X[t-k+1] + W2 * X[t-k+2] + \dots + Wk * X[t] + \dots + Wm * X[t-k+m]$$

is computed. For example,

SER Y=MS(X,1,-1,1T,-1) TO <data>

computes $Y[t] = X[t-2] - X[t-1] + X[t] - X[t+1]$.

In the next display, SER operations have been applied for a variable X in a small data table TEST.

```

28 1 SURVO 84C EDITOR Mon Jun 15 09:19:17 1992      D:\P2\STAT3\ 100 100 0
1 *
2 *SER X1=MA(X,1,1,*) TO TEST
3 *SER X2=MA(X,W3) TO TEST
4 *      W3=1,2,3,*
5 *SER X3=MAE(X,P2:5) TO TEST
6 *SER X4=MS(X,1,1,1T,1) TO TEST
7 *SER X5=CUM(X,2) TO TEST
8 *SER X6=D(X,3) TO TEST
9 *SER X7=MAE2(X,P1:3) TO TEST / with period 2
10 *
11 *DATA TEST,A,B,N,M
12 M 12 1.1 1.1 1.1 11.1 11.1 11.1 11.1
13 N X X1 X2 X3 X4 X5 X6 X7
14 A 2 - - 1.6 - 2.0 - 2.7
15 * 3 3.3 - 3.8 - 3.0 - 4.2
16 * 5 4.7 4.3 5.0 16.0 7.0 - 3.7
17 * 6 5.0 4.6 5.6 18.0 9.0 4.0 3.7
18 * 4 4.0 3.8 4.2 17.0 11.0 1.0 3.3
19 * 2 2.3 2.8 1.9 13.0 11.0 -3.0 3.7
20 * 1 2.0 2.4 1.4 10.0 12.0 -5.0 3.3
21 * 3 3.0 - 2.5 11.0 14.0 -1.0 2.2
22 B 5 - - 5.2 - 17.0 3.0 3.8
23 *

```

7.9.2 Auto- and crosscorrelations

XCORR <data>, <var1>, <var2>, L

computes auto- and crosscorrelations of variables <var1> and <var2> in <data>. If <var2>=<var1>, only autocorrelations of <var1> are computed. The results are represented as a table from line L onwards.

Observations can be limited by IND, CASES and SELECT specifications. The total span of active observations is always from the first non-missing observation to the next missing observation. Thus the observations to be processed are consecutive, unless otherwise stated by IND, CASES and SELECT.

The maximum lag is given as an extra specification MAXLAG. Default is MAXLAG=n/3 where n is the number of observations. Maximum value is MAXLAG=n/2.

The computations are based on a FFT algorithm presented in "Numerical Recipes" by Press, Flannery, Teukolsky and Vetterling (1987).

In the next example, a variable X is generated according to a simple autoregressive process. Thereafter 6 first autocorrelations of X are computed by XCORR with expected results.

```

21 1 SURVO 84C EDITOR Mon Jun 15 09:47:20 1992 D:\P2\STAT3\ 100 100 0
25 *
26 *FILE CREATE SERX,4,1
27 *FIELDS:
28 *1 N 4 X
29 *END
30 *
31 *FILE INIT SERX,200
32 *
33 *VAR X=if(ORDER=1)then(U)else(a*X[-1]+U) TO SERX
34 *a=0.8 U=probit(rnd(1))
35 *
36 *XCORR SERX,X,X,CUR+1 / MAXLAG=6
37 *Autocorrelations of X in data SERX:
38 * Lag X
39 * 0 1.0000
40 * 1 0.8093
41 * 2 0.6241
42 * 3 0.4674
43 * 4 0.3547
44 * 5 0.2872
45 * 6 0.2287
46 *

```

7.9.3 FORECAST operation

FORECAST <data> , <series> , <predictor> , L

makes an automatic forecast of a time series by using a variant of the Holt-Winters' seasonal forecast procedure. (See e.g. B.Abraham and J. Ledolter (1983): *Statistical Methods for Forecasting*, Wiley.)

The period s of the series is given by PERIOD= s . If the PERIOD specification is omitted, FORECAST tries to judge s from the data by trying out alternatives $s=1,2,3, \dots, n/3$ where n is the number of observations in the estimation period. For each s , the Holt-Winters' procedure is applied tentatively and the MSE's are compared. Furthermore a simple weighting function is employed to prefer the basic period and not its multiplicities.

The range of observations used for estimation of the level, slope, and seasonal components can be limited by an IND specification of the form IND=ORDER, <first_obs> , <last_obs> .

The predicted values of <series> are saved in <predictor> for the estimation period plus one complete period (s observations) ahead. Number of forecast values may be changed by the AHEAD=<#_of_values> specification. The forecast values are not saved when character '-' is given in the place of <predictor>.

L (optional) is the first edit line for the results.

The type of the model is selected by the specification C=<type>. Default is C=1 (additive seasonal). Another alternative is C=0 (multiplicative seasonal). In case $s=1$ both models correspond to Holt's double exponential smoothing procedure.

Also values $0 < C < 1$ may be selected, but then the time required for the estimation is much longer. In this general case, parameter C defines the connections between the different components of the time series by the operation

$$X_C Y = F^{-1}(F(X,C) + F(Y,C), C)$$

where

$$F(X,C) = (X^C - 1) / C + C \text{ for } 0 < C \leq 1,$$

$$F(X,0) = \log(X).$$

This means that for $C=1$, $X_C Y = X + Y$ and for $C=0$, $X_C Y = XY$. By this generalization, a smooth transition from an additive model to a multiplicative one may be considered.

The three smoothing coefficients are selected by minimizing the mean square error of the one-step-ahead forecast errors. The initial values of the level, slope and seasonal parameters are obtained first by backforecasting on the data.

However, fixed values for the smoothing coefficients can also given by a specification of the form PAR= a_1, a_2, a_3 where a_1 =level coefficient, a_2 =seasonal coefficient and a_3 =slope coefficient. Each of them must be in the interval (0,1). For stable components, the smoothing coefficients should be close to zero.

After estimation of the smoothing coefficients, the outliers in data may be rejected on the basis of the forecast errors. Only one (the worst) outlier is rejected at a time by replacing the data value by the current forecast and the whole estimation process is repeated.

The $\text{OUTLIERS}=n, k$ specification gives n as the maximum number of such rejections and k as the threshold for an outlier. An observation is rejected (smoothed), if its forecast error exceeds k times the standard error. Default is $\text{OUTLIERS}=3, 2.5$. By $\text{OUTLIERS}=0$, no outliers are considered.

In addition to the forecast, the components of the series may also be saved in the original data (file) by entering certain masks. The components and their mask symbols are:

| | |
|---------------|---|
| Trend (level) | T |
| Slope | B |
| Seasonal | S |

As an example, we consider an artificial time series of 24 observations with $s=4$. At first we have a completely regular pattern and get a nice forecast as follows:

```

17 1 SURVO 84C EDITOR Mon Jun 15 14:04:06 1992 D:\P2\STAT3\ 100 100 0
1 *
2 *DATA X: 1 2 3 4 2 3 4 5 3 4 5 6 4 5 6 7 5 6 7 8 6 7 8 9 END
3 *
4 *FORECAST X,X,-,5
5 *Holt-Winters' Additive Seasonal Forecast: Data X, Variable X
6 *Period=4 obs. (judged from data) Estimation on observations 1-24
7 *MSE=8.34648e-013 a(level)=0.427 a(seasonal)=1.000 a(slope)=0.413
8 *Autocorrelations of residuals: r1=+0.28 r2=-0.25 r3=-0.35 r4=+0.08
9 *Obs.# Forecast
10 * 25 7.0000
11 * 26 8.0000
12 * 27 9.0000
13 * 28 10.0000
14 *

```

To make the situation more complicated, we alter 4 observations (nos. 3,10,19 and 20) thus substantially violating the regular pattern. To have the possibility of saving predicted values (variable 'Pred'), we copy the data set to a data file ZDATA. Note also that 4 missing observations have been inserted at the end of the data set to have space for the actual forecast.


```

25 1 SURVO 84C EDITOR Mon Jun 15 14:05:36 1992 D:\P2\STAT3\ 100 100 0
16 *
17 *DATA Z: 1 2 5 4 2 3 4 5 3 2 5 6 4 5 6 7 5 6 5 10 6 7 8 9 - - - -
18 * END
19 *FILE COPY Z,ZDATA
20 *
21 *FORECAST ZDATA,Z,Pred,22 / IND=ORDER,1,24
22 *Holt-Winters' Additive Seasonal Forecast: Data ZDATA, Variable Z
23 *Period=4 obs. (judged from data) Estimation on observations 1-24
24 *Outliers:19,20,3 (+more to be found)
25 *MSE=0.156429 a(level)=-0.000 a(seasonal)=0.000 a(slope)=0.978
26 *Autocorrelations of residuals: r1=+0.04 r2=+0.08 r3=+0.02 r4=-0.21
27 *Obs.# Forecast
28 * 25 6.8239
29 * 26 7.6693
30 * 27 9.0842
31 * 28 10.128
32 *Predicted values saved as variable Pred (obs. 1-28)
33 *

```

Before giving the final results (on lines 22-32) the FORECAST operation temporarily tells about its decisions on the screen as follows:

```

period=4 (judged from data)
outlier #19 value=5 deviation=2.735774 treshold=2.5
outlier #20 value=10 deviation=2.52977 treshold=2.5
outlier #3 value=5 deviation=3.037758 treshold=2.5

```

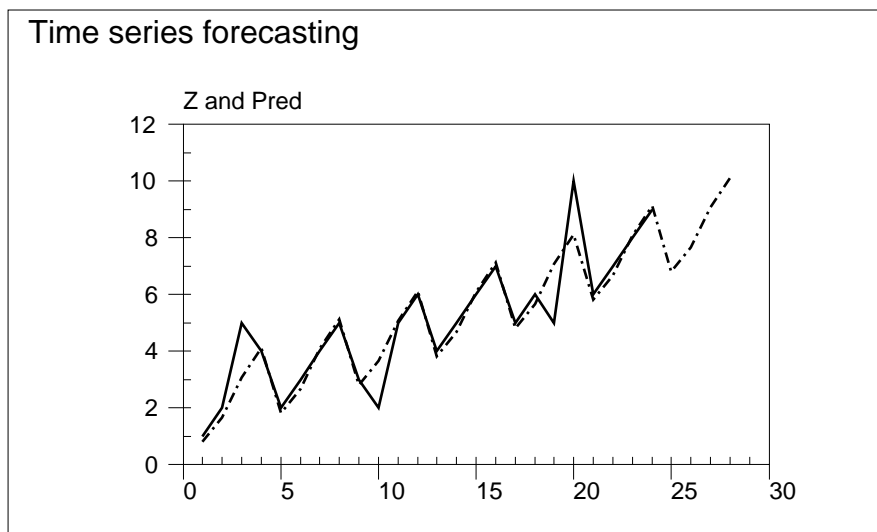
Since by default FORECAST is allowed to reveal at most 3 outliers, the fourth one (#10) remains undetected in this case. However, the forecast seems to be satisfactory.

To illustrate the results, a graph is created by the following PLOT scheme:

```

23 1 SURVO 84C EDITOR Mon Jun 15 14:19:28 1992 D:\P2\STAT3\ 100 100 0
41 *
42 *HEADER=[Swiss(12)],Time_series_forecasting YLABEL=[Swiss(9)],Z_and_Pred
43 *PLOT ZDATA,TIME,Z,Pred
44 *XSCALE=0(5)30 YSCALE=0(2)12 TICK=1,1 PEN=[Swiss(9)]
45 *ZLINE=[line_width(1)],1
46 *PredLINE=[line_type(3)],1
47 *DEVICE=PS SIZE=1164,700 YDIV=100,450,150
48 *

```



8. Graphics

Plotting of graphs is an application area well suitable for the editorial approach of Survo. When making more sophisticated pictures it is never possible to reach the final solution immediately. Usually several trials are necessary in order to find a balance between various constituents of the graph. In such a stepwise refinement, much is gained from an instant control of the plotting process. It should be easy to add and remove various attributes related to the picture. Similarly, it should be possible to employ earlier plots as models for new applications.

There are not so many basic types of graphs available in Survo. However, in each type there are plenty of options. These basic forms of graphs can overlay each other thus giving a great variety of alternatives. One should remember that seemingly complicated things are nothing more than combinations of simple ones.

The control of Survo graphics is to a great extent device-independent. The graphs are described by Survo's own graphical language, i.e. by PLOT schemes with optional specifications.

However, each device offers its particular specialities which should be observed. For example, dynamic features typical for screen graphics cannot be presented on paper. The device-dependent options are given in device drivers (.DEV files in .\SYS). The device itself is indicated by a DEVICE specification. The default device is given by plot_mode in SURVO.APU. Survo device drivers are standard text files and open for modifications and extensions.

The current graphical devices supported by Survo are

| | | |
|----------------------|--------------|----------------|
| | | default driver |
| (Color) PostScript | DEVICE=PS | PS.DEV |
| Screen | DEVICE=G | CRT16.DEV |
| Canon laser printers | DEVICE=Canon | CANON2.DEV |
| HP plotters | DEVICE=H | HP.DEV |

The main alternatives are the two first-mentioned ones. [There are clear signs that, in the nearest future, also screen graphics will be done through the PostScript interface of Survo.](#) Although no real display PostScript is available on PC's, there are good emulators like the Ghostscript program for pre-viewing Survo PostScript plots and reports on the screen. By ready-made macros /GS-PLOT and /GS-PRINT, one can redirect Survo color PostScript graphics and documents to the screen.

Most of the graphs are created by plotting schemes where one or more PLOT (GPLOT) operations are to be activated. The plots can be seen immediately on the graphic screen (GPLOT operation) or directed to another device or to a

file (PLOT operation). When making documents containing both text and graphics, the graphs are usually saved in PostScript files (by PLOT with DEVICE=PS, <file_name>) and then taken into the document by referring to the file name.

In this section, we shall describe the PLOT operation for PostScript printers. The essentials of the GPLOT operation (for the screen) are mostly the same. The specialities related GPLOT are told in 8.10 . [Descriptions for Canon printers \(DEVICE=Canon\) and HP plotters are not given here. The user can find information and examples on Canon plotting in the 1987 version of the user's guide. The essentials on HP plotting can be found by an inquiry HP? .](#)

Most of the specifications used to support PLOT operations are independent of the particular plotting device. However, since various devices offer different possibilities in graphics, special device-dependent supplements may be inserted to most specifications in brackets []. Their location is in front of the right-hand side text.

For example, when plotting a time series, a typical extra specification is LINE=1 telling that consecutive points in the graph will be joined by straight line segments. To indicate the line type, color, width etc., for example, following augmented forms of LINE may be supplied:

```
LINE=[line_width(1)],1
```

```
LINE=[RED][line_width(10)][line_type(2)],1 .
```

The order of these additional characterizations in brackets is usually immaterial. The possible keywords and macros available are given in Survo device drivers.

The following table lists the basic forms of Survo plots and their availability on different devices:

| | PS | Screen | Canon | HP |
|---|----|--------|-------|----|
| Frames, texts, bar and pie charts | + | + | + | + |
| Scatter diagrams, time series line graphs, probability plots | + | + | + | + |
| Curves and families of curves | + | + | + | + |
| Histograms (HISTO operation) | + | + | + | + |
| Contour plots for functions of two variables | + | + | - | - |
| Plots of multivariate data, Chernoff's faces, Andrews' function plots, etc. | + | + | - | - |

The best way to learn Survo graphics is by doing. In addition to general descriptions, we shall give representative examples of plotting schemes of various types. Since the number of different enhancements and their combinations is enormous, it is not possible to cover all possibilities. Therefore, our recommendation is that the user should make experiments by starting from ready-made schemes.

8.1 General specifications

We begin by describing PLOT specifications available in all forms of Survo graphics.

DEVICE=<plotting_device>

gives the instrument to be used as the plotting device. Default is given by the system parameter `plot_mode` in `SURVO.APU`. Various alternatives are

| | |
|--|----------------------------|
| DEVICE =PS | (Color) PostScript |
| DEVICE =G or DEVICE =CRT | Screen (default in GPLLOT) |
| DEVICE =Canon | Canon laser printers |
| DEVICE =H | HP plotters |

The **DEVICE** specification can be extended by a file name (for example, **DEVICE**=PS,GRAPH1.PS). In that case, the graph will not be appear directly but it is saved in the file using the graphical language of the selected device.

Files thus created can be included in Survo documents produced by the **PRINT** operation by a - `picture` control lines in the **PRINT** list. This applies to PostScript and Canon files.

HOME=<x_coord> , <y_coord>

specifies the location of the lower left corner of the graph. Coordinates are given in plotting units. The default value may be obtained by activating **PLOT ?** (or **GPLLOT ?**).

The plotting unit is 0.1 mm (dmm) on all other devices except the screen. On the screen it is one pixel.

SIZE=<width> , <height>

specifies the size of the graph (outer frame) in plotting units. The default value may be obtained by **PLOT ?** (or **GPLLOT ?**).

The whole plotting area, defined by **HOME** and **SIZE**, is divided in 3x3 rectangular regions. The middle region is reserved for the actual graph and the other surrounding regions for labels and scales. The division of the plotting area into these regions is controlled by specifications **XDIV** and **YDIV**.

XDIV=<left_margin> , <plot_width> , <right_margin>

specifies the division of the picture width. It is sufficient to use values proportional to actual parameters. For example, the specifications **XDIV**=1,12,2 and **XDIV**=100,1200,200 are equivalent. Default values may be inquired by **PLOT ?** (or **GPLOT ?**).

YDIV=<bottom_margin> , <plot_height> , <top_margin>

specifies the division of the picture height. It is sufficient to use values proportional to actual parameters. [Default values may be inquired by **PLOT ?** \(or **GPLOT ?**\).](#)

An inquiry for the default settings of the PostScript interface of Survo gives the following information:

```

7 1 SURVO 84C EDITOR Fri Jun 26 13:12:31 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *PLOT ?L / DEVICE=PS,NUL
3 *SIZE=1500,1500 HOME=250,100
4 *XDIV=300,1000,200 YDIV=300,1000,200
5 *x_pixel/y_pixel=1
6 *

```

Above, **NUL** in the **DEVICE** specification defines a dummy file for the non-existent graphical output of this inquiry. Attempt of using **DEVICE=PS** only when the device (PS printer) is not on, leads to an error message.

FRAME=<0,1,2 or 3>

controls plotting of frames around the graph.

FRAME=2 is default and means that all frames are plotted.

FRAME=1 erases the outer frame.

FRAME=0 erases all frames and coordinate axes notations.

FRAME=3 erases the inner frame and coordinate axes notations.

FRAMES=<list_of_frame_specifications>

gives a list of extra frames (boxes) to be drawn in the current graph. Each member of this list is a word (name of the frame) and any such a frame must be given in the form

<name_of_the_frame>=<xhome> , <yhome> , <width> , <height> , <fill_index>

where (xhome,yhome) are the coordinates of the left-bottom corner of the frame and <width> and <height> give the size of of the frame. All the values are in plotting units (not in coordinates of the graph). <fill_index> is optional and gives the shade of gray or color used for painting the entire area inside the current frame. If <fill_index> is omitted, the area is not painted.

The **fill indices** are integers 0,1,2, ... giving various colors (**GPLOT**) or shades of gray from white to black (**PLOT**). The different **GPLOT** colors 0,1,...,15

are defined in the corresponding device driver (CRT16.DEV). The PLOT shadings vary from 0 (white) to 9 (black), but this range can be extended by entering a SHADEMAX specification. For example, SHADEMAX=24 sets 24 grades of gray from 0 (white) to 23 (black).

In color PostScript, a negative fill index, say $-i$, refers to a CMYK (Cyan-Magenta-Yellow-black) color combination defined by a [FILL- i] specification in the current PLOT scheme. For example, [FILL-1]=0.5,0.5,0,0 defines fill index -1 as a light blue color. By this technique, there is practically no upper limit for possible colors in one PostScript graph of Survo.

HEADER=<a_string_without_spaces>

gives a title to be plotted in the upper left corner of the picture area. Default is HEADER=<type>_DIAGRAM_OF_<data>, etc. depending on the type of the graph.

TEXTS=<list_of_text_specifications>

gives a list of extra texts to be written in the current graph. Each member of this list is a word (name of the text), and any such a text must be given in the form

<name_of_the_text>=<string>,x,y

where (x,y) is the starting point of the text in plotting units and <string> is a string without spaces and ', 's to be written. Possible spaces should be replaced by '_'. Similarly ', 's must be replaced by ';'s.

Example: TEXTS=T1,T2 T1=SURVO_84,500,50 T2=9_May_1987,650,50

Any of the text names, say TL, listed in the TEXTS specification, can also refer to a set of several lines. In this case, TL is presented in the form

TL=#LINES:<first_line>,<last_line>,x,y,<line_spacing>

where (x,y) is the starting point of the first line. The following lines will be plotted below the first one by using the given <line_spacing>.

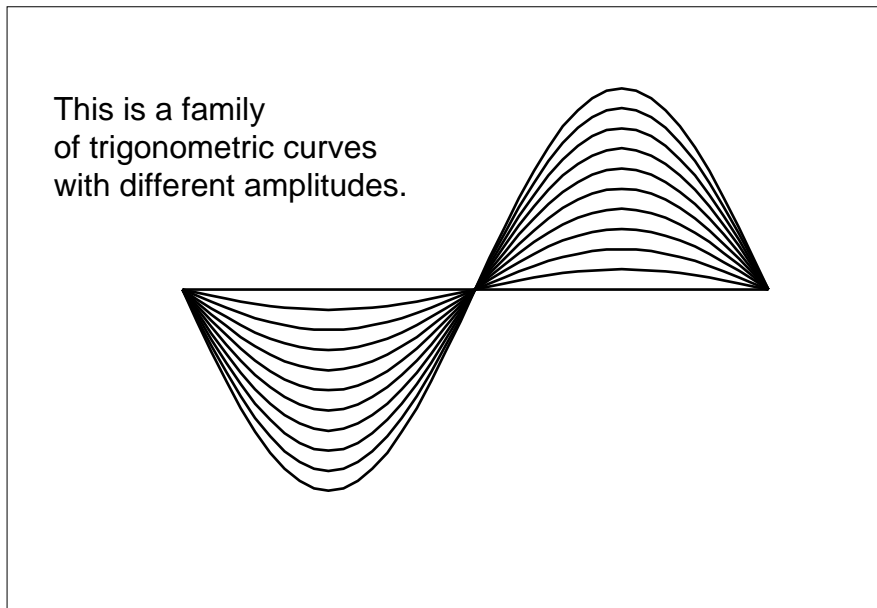
For example, the PLOT scheme

```

19 1 SURVO 84C EDITOR Sat Jul 04 10:13:12 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *PLOT Y(X)=A*sin(X) / A=0,10,1 X=[line_width(1)],-pi,pi,pi/20
3 * pi=3.14159 XSCALE=-pi,0,pi
4 * FRAME=3 SIZE=1164,800 HEADER=
5 * DEVICE=PS
6 *TEXTS=T1 T1=[Swiss(12)],#LINES:A,B,50,650,50
7 *
8 A This is a family
9 * of trigonometric curves
10 B with different amplitudes.
11 *

```

produces a graph



FILLS=<list_of_fill_specifications>

gives a list of extra fill definitions for the current graph. Each member of this list is a word (name of the fill) and any such a definition must be given in the form

<name_of_the_fill>=<x>, <y>, <fill_index>

where <x>, <y> are the coordinates (in plotting units) of the point inside the area to be filled and <fill_index> gives the fill type. **FILLS** is not valid in PostScript and HP plotting because those devices do not provide a 'floodfill' property. However, the **LINES** specification (described later) offers similar means in PostScript plotting.

PEN=<list_of_device-dependent_control_words>

determines the default setting of various graphical attributes of texts to be plotted in the graph like those given by **HEADER** and **TEXTS** specifications. The settings of **PEN** are temporarily overridden by corresponding device-dependent statements given in any of the other specification.

For example,

```
PEN=[Times(24)][rot(0)]
```

tells that texts in this graph should be written in 24 Point Times font in standard horizontal direction. ([rot(0)] means that the rotation angle is 0). Then texts T1 and T2 given by

```
TEXTS=T1,T2
T1=[Times(10)][rot(60)],10,Point_Text_rotated_60_degrees,100,50
T2=Text_in_24_Point_Times,300,200
```

will appear in the graph as follows:



If the `PEN` specification were omitted, both texts would be printed according to the style specified for `T1`.

`LINETYPE=<device-dependent_parameters>`

corresponds to `PEN` but gives default settings for plotting of lines and curves. For example, `LINETYPE=[line_width(0.48)][line_color(0.5)]` sets the line width to 0.48 Points and the line color to 0.5 on the (0,1) grayness scale of PostScript.

8.2 Frames, texts and lines

The simplest form of the `PLOT` operation is `PLOT /FRAME`. It employs general specifications described above as such. Due to `FRAMES` and `TEXTS` specifications, it is suitable for creating various layouts consisting of rectangular regions with explanatory text.

Another alternative for such applications is provided by ‘box’ graphics in the `PRINT` operation.

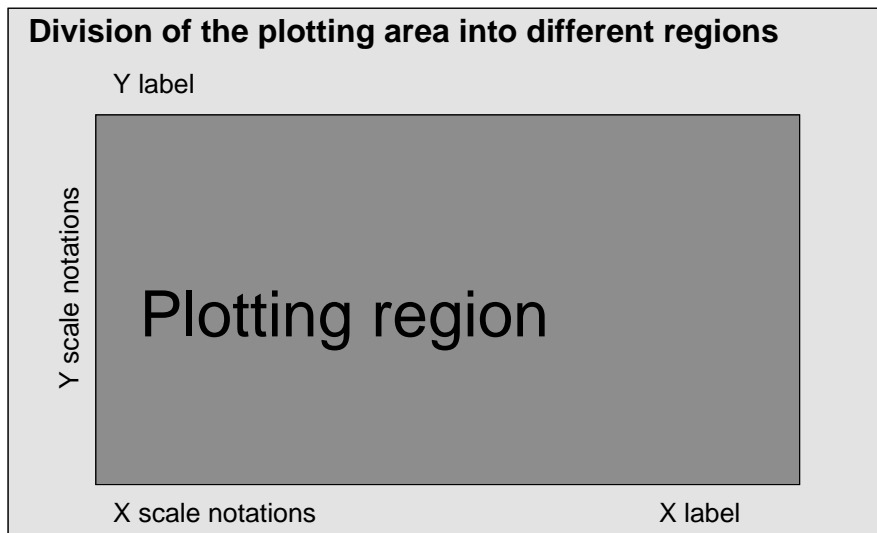
The following `PLOT /FRAME` scheme illustrates various components of a typical Survo graph.


```

12 1 SURVO 84C EDITOR Sat Jun 27 16:57:27 1992 D:\P2\PLOT\ 100 100 0
8 *PLOT /FRAME / DEVICE=PS
9 *HEADER=[Swiss(12)],Division_of_the_plotting_area_into_different_regions
10 *
11 *SIZE=W,H W=1164 H=700 Width and Height of the graph
12 *XDIV=X1,X2,X3 X1=0.1*W X2=0.8*W X3=W-X1-X2 X division
13 *YDIV=Y1,Y2,Y3 Y1=0.1*H Y2=0.7*H Y3=H-Y1-Y2 Y division
14 *PEN=[Swiss(10)][rot(0)] sets the default font. [rot(0)]=texts horizontal
15 *
16 *TEXTS=T1,T2,T3,T4,T5
17 *T1=[Swiss(24)],Plotting_region,1.5*X1,Y1+0.4*Y2
18 *T2=X_scale_notations,1.2*X1,0.3*Y1
19 *T3=[rot(90)],Y_scale_notations,0.8*X1,0.4*Y2 in vertical direction
20 *T4=X_label,X1+0.8*X2,0.3*Y1 T5=Y_label,1.2*X1,H-0.8*Y3
21 *
22 *FRAMES=F1,F2
23 *F1=0,0,W,H,1 Entire graph painted in light gray
24 *F2=X1,Y1,X2,Y2,4 Plotting region painted in darker gray
25 *

```

Activation of PLOT /FRAME on line 8 gives the following graph on a Post-Script printer:



We have defined various basic measures in the graph mostly in the terms of the graph width (W) and height (H). Then if, for some reasons, the dimensions are to be changed, the derived measures (like the placing of frames and texts) will be automatically adjusted according to the new situation.

For drawing of figures consisting of arbitrary line segments, the PLOT /FRAME offers also a LINES specification. It is not available in other PLOT operations.

LINES=<list_of_line_specifications>

gives a list of extra sequences of line segments to be drawn in the current graph. Each member of this list is a word (name of the sequence) and any such a sequence must be given in the form

<name_of_the_sequence>= x_{home} , y_{home} , x_1 , y_1 , x_2 , y_2 , ..., x_n , y_n

Then a polyline starting from point (x_{home}, y_{home}) and going through points $(x_{home}+x_1, x_{home}+y_1)$
 $(x_{home}+x_1+x_2, y_{home}+y_1+y_2)$
 ...
 $(x_{home}+x_1+x_2+\dots+x_n, y_{home}+y_1+y_2+\dots+y_n)$
 will be drawn. All the values are in plotting units (not in user coordinates).

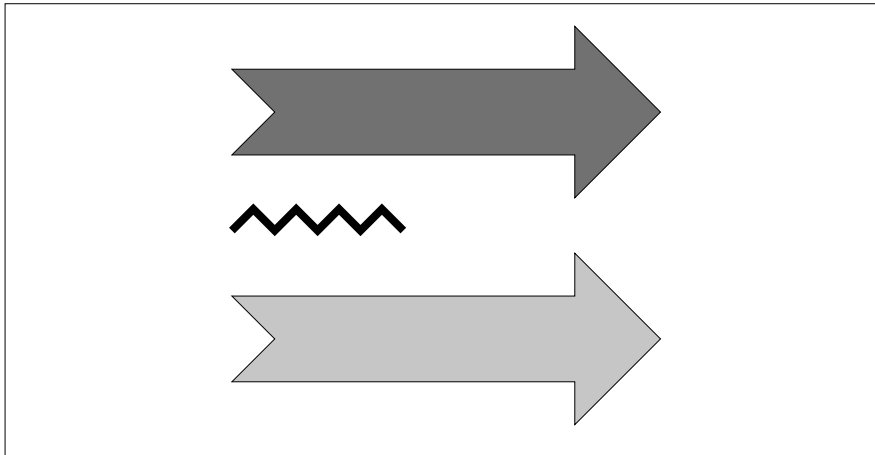
If the sequence of line segments forms a closed curve, the inside area can be painted by means of the FILL specification. The FILL specification cannot be used in PostScript plotting. In this case, one can define the fill index x for painting by a device-specific control word [FILL(x)] where x is 0,1,2, ... for various shades of gray and -1,-2, ... for various color PostScript colors defined by [FILL- x] specifications in the current PLOT scheme. To omit painting in PostScript graphs, use [NO_FILL].

Three polylines are plotted in the next PLOT scheme:

```

12 1 SURVO 84C EDITOR Sat Jun 27 17:42:56 1992 D:\P2\PLOT\ 100 100 0
41 *
42 *PLOT /FRAME / DEVICE=PS
43 *FRAME=3 SIZE=1164,600 XDIV=0,1,0 YDIV=0,1,0
44 * A=300 B=100 H=200 N=8 h=H/2
45 *LINES=ARROW1,ARROW2,WORM
46 *ARROW1=[FILL(5)],A,4*B,H,H,-H,H,N*H,0,0,H,2*H,-2*H,-2*H,0,H,-N*H,0
47 *ARROW2=[FILL(2)],A,B,H,H,-H,H,N*H,0,0,H,2*H,-2*H,-2*H,0,H,-N*H,0
48 *WORM=[NO_FILL][line_width(3)],A,3*B,h,h,h,-h,h,h,h,-h,h,h,h,-h,h,h,h,-h
49 *

```



8.3 Bar and pie charts

PLOT <data>

plots a bar/pie chart of a Survo data set. The first active field of the data set will be assumed to be a string field, and it is used as a label for each bar/pie.

If the label field is not the first one, it can be indicated by a 'L' mask. All other active fields will be plotted.

If the label field is not a string field, no labels are used and all active fields are plotted.

TYPE specifies the type of the chart in the PLOT operation. Default is TYPE=HBAR (horizontal bar chart). Other types are:

| | |
|-------------|--|
| TYPE=VBAR | vertical bar chart |
| TYPE=%HBAR | horizontal bar chart in percentages |
| TYPE=%VBAR | vertical bar chart in percentages |
| TYPE=MHBAR | multiple horizontal bar chart |
| TYPE=MVBAR | multiple vertical bar chart |
| TYPE=%MHBAR | multiple horizontal bar chart in percentages |
| TYPE=%MVBAR | multiple vertical bar chart in percentages |
| TYPE=%AHBAR | as %HBAR but with areas equal to absolute size |
| TYPE=%AVBAR | as %VBAR but with areas equal to absolute size |
| TYPE=PIE | multiple pie chart, total area proportional to column sum |
| TYPE=%PIE | multiple pie chart, total area constant |



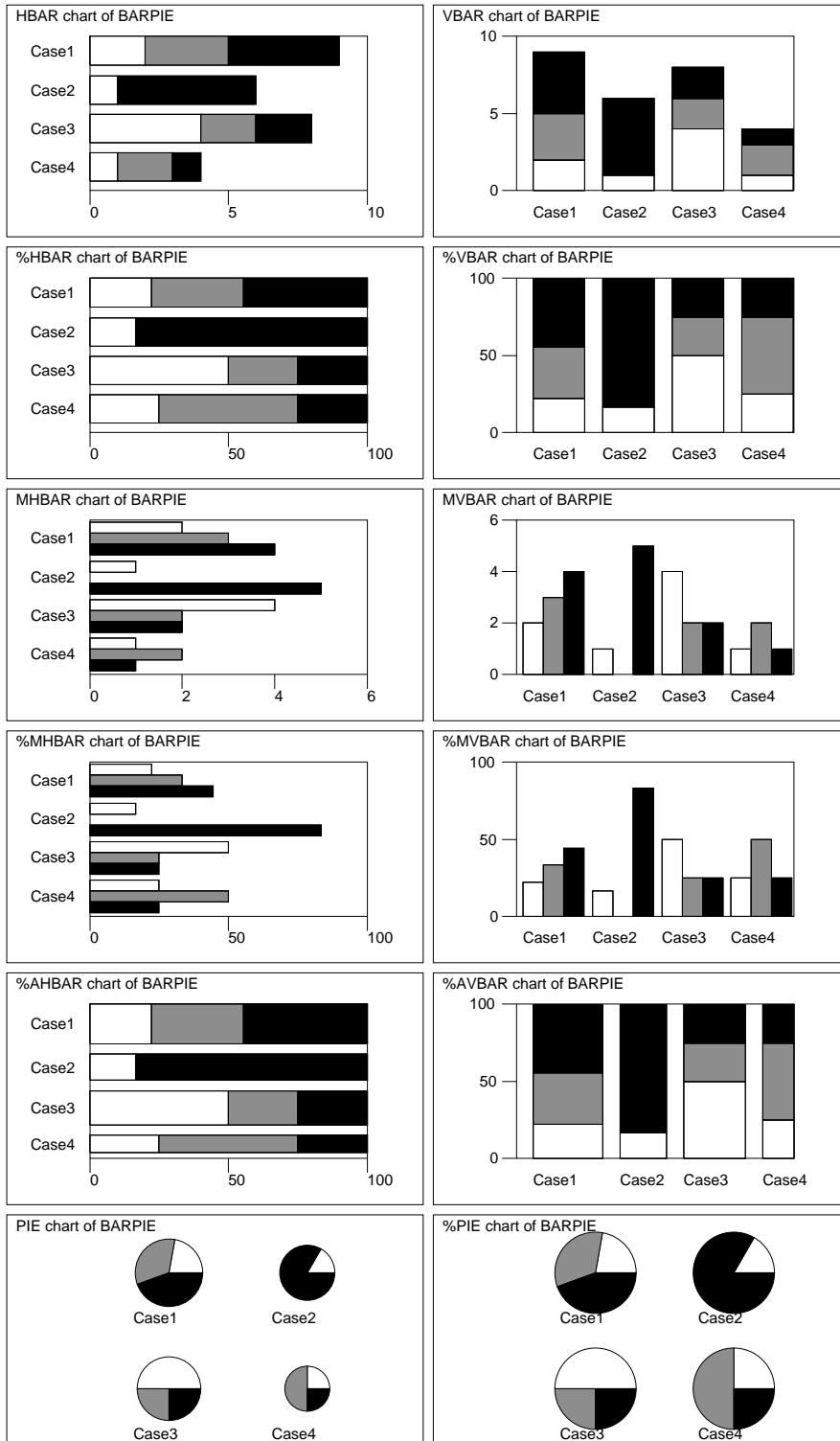
The different forms of these graphs are illustrated by the next cartoon created by the following PLOT schemes:

```

12 1 SURVO 84C EDITOR Sun Jun 28 11:16:28 1992 D:\P2\PLOT\ 100 100 0
1 *
2 **GLOBAL* SIZE=575,320 LEGEND=- PEN=[Swiss(6)]
3 *DATA BARPIE
4 *Name X Y Z
5 *Case1 2 3 4
6 *Case2 1 0 5
7 *Case3 4 2 2
8 *Case4 1 2 1
9 *
10 *
11 *PLOT BARPIE
12 *DEVICE=PS,HBAR1.PS
13 *
14 *PLOT BARPIE
15 *DEVICE=PS,VBAR1.PS TYPE=VBAR
16 *
17 *PLOT BARPIE
18 *DEVICE=PS,%HBAR1.PS TYPE=%HBAR
19 *
20 * etc.

```

Forms of bar and pie charts by changing TYPE:



The cartoon has been printed by the PRINT operation:

```

16 1 SURVO 84C EDITOR Sun Jun 28 11:24:23 1992 D:\P2\PLOT\ 100 100 0
47 *
48 *PRINT CUR+1,END
49 % 300 (= 300 dmm of empty space)
50 - [Times(12)]
51 *Forms of bar and pie charts by changing TYPE:
52 % 334
53 - picture HBAR1.PS,*,*
54 - picture VBAR1.PS,**589,*
55 % 334
56 - picture %HBAR1.PS,*,*
57 - picture %VBAR1.PS,**589,*
58 % 334

```

Extra specifications for bar/pie charts

SHADING=<list of fill indices>

specifies a fill index (shading, color) for each section of bars/pies. The values 0,1,2, ... refer to tones from 'white' to 'black' on paper. Default maximum 9 can be extended by **SHADEMAX** specification when the plotting device gives a possibility for more shades of gray.

Negative values $i=-1,-2, \dots$ refer to PostScript colors defined by [**FILL**- i]=Cyan,Magenta,Yellow,Black specifications. For example [**FILL**-2]=0,0,1,0 defines fill index -2 as 'yellow'.

In pie charts, any of the sectors may be pulled out by putting **P** after the fill index. The distance of the new location of the sector will be 20 per cent of the radius. To alter this distance, use **P3** instead of **P** for 30 per cent distance, for example.

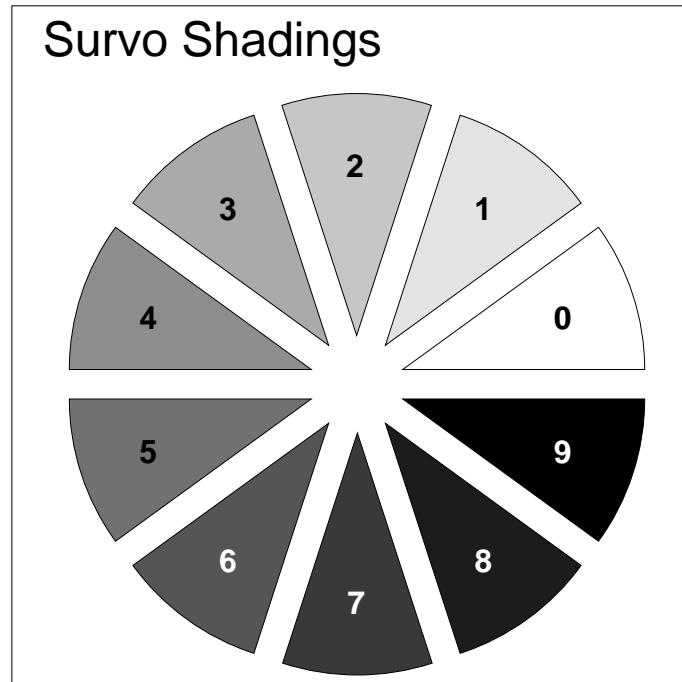
If **SHADING** is not given, an ascending set of shading values is used from 0 to 9 (to **SHADEMAX**) with regular intervals.

Below, the PostScript shadings are displayed by activating the following plotting scheme for a pie chart. The grades of gray may be slightly dependent on the resolution of the PostScript printer in use.

```

13 1 SURVO 84C EDITOR Sun Jun 28 11:34:09 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *DATA SECTORS,A,A,N
3 N Name 0 1 2 3 4 5 6 7 8 9
4 A - 1 1 1 1 1 1 1 1 1 1
5 *
6 *PLOT SECTORS / TYPE=PIE DEVICE=PS
7 *SHADING=0P,1P,2P,3P,4P,5P,6P,7P,8P,9P
8 *HEADER=[Swiss(18)],Survo_Shadings
9 *LEGEND=- LABELS=7
10 *SIZE=900,900 XDIV=30,800,70 YDIV=0,800,100
11 *PEN=[Swiss(12)][BOLD][autom_color(0.4)]
12 *

```



In this example, also specifications `LEGEND` and `LABELS` were employed. They will be explained in the sequel.

`[autom_color (0.4)]` is a device-dependent option telling the printer to use white text on a painted area when the value on the gray scale of PostScript (ranging from 1=white to 0=black) is at most 0.4 .

Each bar/pie chart will include a list of shadings defined by the `LEGEND` specification.

`LEGEND=<a_string_without_spaces>`

gives a title for this list. The list will be omitted by using `LEGEND=-` (as we did in the preceding example). The list of shadings is presented as one row below the graph.

The default setting of the legend text and boxes is overridden by `LEGEND` of the form

`LEGEND=<x_coord> , <y_coord> , <#_of_columns>`

where the coordinates of the starting position of the legend (relatively to `HOME` of the graph) and the number of columns to be used are given. The locations of the boxes and texts (names of variables) describing the shadings are given more accurately by specifications

`LEGEND_BOX=<x_box> , <y_box> , <box_width> , <box_height>`

`LEGEND_TEXT=<x_text> , <y_text>`

where `<x_box>` , `<y_box>` describe the spacing of boxes and `<box_width>` , `<box_height>` (optional) give their dimensions. `<x_text>` , `<y_text>` give the starting position of the text (name of the variable) with respect to the low bot-

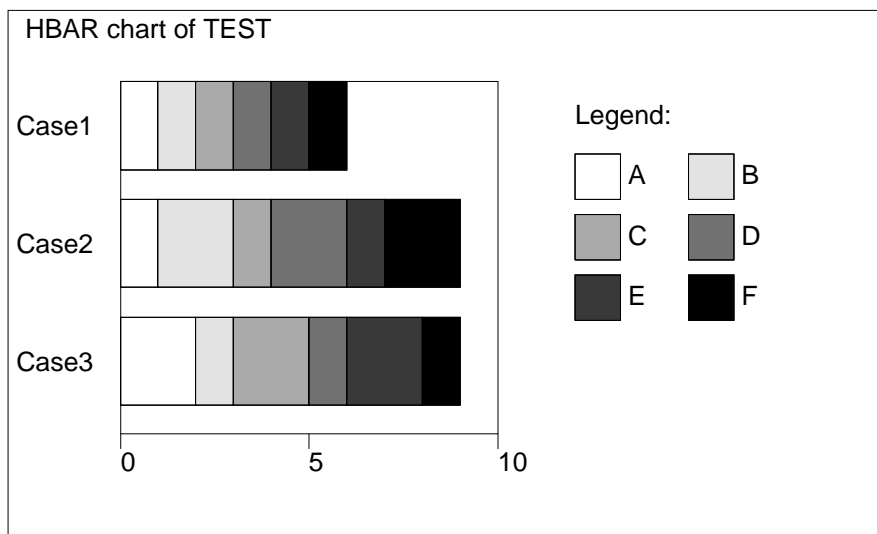
tom corner of the current box.

The extended form of **LEGEND** could be applied as follows:

```

10 1 SURVO 84C EDITOR Sat Jul 18 09:49:32 1992 D:\P2\PLOT\ 100 100 0
74 *
75 *DATA TEST
76 *Name A B C D E F
77 *Case1 1 1 1 1 1 1
78 *Case2 1 2 1 2 1 2
79 *Case3 2 1 2 1 2 1
80 *
81 *PLOT TEST
82 *DEVICE=PS SIZE=1164,700 XDIV=150,500,514
83 *LEGEND=750,450,2 LEGEND_BOX=150,-80,60,60 LEGEND_TEXT=70,20
84 *TEXTS=T1 T1=Legend:,750,550
85 *

```



SCALE=<min.value> , <2nd value> , . . . , <max.value>

defines the scale and the scale labels on the X axis (HBAR, %HBAR, etc.) or on the Y axis (VBAR, %VBAR, etc.).

A scale like **SCALE=0,5,10,15,20,25** may be written in an abbreviated form **SCALE=0(5)25**. Observe that negative values are allowed in **MHBAR** and **MVBAR** charts only.

If **SCALE** is not given, scaling and labelling is automatic and based on the numeric structure of the data and on the size of the graph.

In pie charts, the maximum size of the pies can be defined by giving a specification **MAX**=<value> where <value> typically exceeds the sum of data values in the largest observation.

GRID=<X or Y>

draws straight lines parallel to the X axis (**GRID=X**) in types **VBAR** or to the Y axis (**GRID=Y**) in types **HBAR** through points indicated by **SCALE**.

GRID=<step>

works as **GRID** above but uses the step length <step> as a gap between the grid lines. If the step parameter is negative, only small ticks are made (outside the inner frame).

TICK=<step>

draws small ticks on the scale axis (inside the inner frame) by using the step length <step>.

VALUES=<image> , <distance>

specifies values of the observations to be plotted in the bar/pie in question as a label.

<image> is either of the form ###.# or ##.##. In the latter case, the value of the current section will be plotted in percentages.

<distance> indicates the place of the label. <distance> is given as # of character widths from the lower end of bars. However, if <distance> is negative, labels will be positioned near the higher end of bars. In pie charts, <distance> must be positive and it is given in 10 per cent units of the radius and from the center.

To place the labels outside of the bars (in bar charts), insert an extra word (like 1) to the **VALUES** specification: **VALUES=<image> , <distance> , 1**. The sign of <distance> again indicates to which side of the bars the labels will be placed.

LABELS=<distance>

works as **VALUES** but plots labels of the observations.

GAP=<ratio of the gap between the bars and the bar width>

(in bar charts) has the default value 2/3. For example, **GAP=0** indicates that no gap is left between the bars.

GAP=<gap> , <gap_start> , <gap_end>

(in bar charts) gives the proportional values for gaps between the bars (gap), before the first bar (gap_start), and after the last bar (gap_end). The default values are $gap=2/3*bar_width$, $gap_start=gap$, $gap_end=0$.

MINVALUE=<minvalue>

(in **HBAR** and **VBAR** graphs) converts lengths of bars below <minvalue> to 0. Default is **MINVALUE=-1e30**, i.e. no conversion. **MINVALUE** can be used for clipping bars at the minimum **SCALE** value. For example, to plot data 200,250,230 as a bar chart in scale **SCALE=200 (10) 250**, **MINVALUE=200** could be used.

PLAN=R1,R2,...

(in pie charts) specifies the arrangement of charts in the plotting area. R1 is # of charts on the first row, R2 on the second row, etc.

If PLAN is not given, an automatic solution is generated on the basis of the shape of the plotting area and the number of pies.

ANGLE=<angle_in_degrees>

(in pie charts) gives the start angle for the first sector. Default is ANGLE=0.

8.4 Scatter diagrams and plots of time series

The rules for making scatter diagrams and line graphs (plots of time series) are to a great extent similar and the same Survo program module takes care of both. The characteristic separation is whether consecutive points are distinct or joined to each other by line segments. This is controlled by a LINE specification. Furthermore, in time series, the variable on the X axis is often replaced by an artificial TIME variable.

The form of the PLOT operation is in these cases

PLOT <data>,<X_variable>,<Y_variable>

and it makes an X-Y plot of selected observations of the Survo <data> for variables <X_variable> and <Y_variable>. Specifications IND, CASES and SELECT are used for selection of observations. Each observation is denoted by a small dot or circle in the graph, unless otherwise stated by a POINT specification.

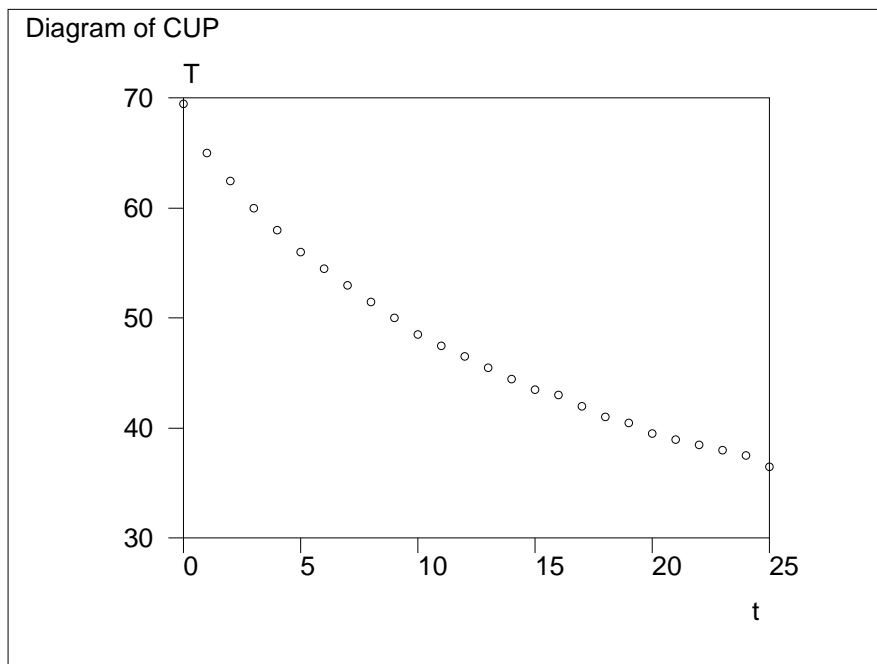
As a simple example, we make an X-Y plot of a data set given as a two variable data list in the edit field as follows:

```

14 1 SURVO 84C EDITOR Fri Jul 03 14:43:18 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *This data set is from an experiment made in a Finnish secondary school
3 *(Pohjoispuiston lukio, Hyvinkää) by Rolf Dahlström and his pupils.
4 *
5 *Temperature T (in C) of a coffee cup as a function of time t (in min)
6 *
7 *DATA CUP:(t,T) 0,69.5 1,65.0 2,62.5 3,60.0 4,58.0 5,56.0 6,54.5 7,
8 *53.0 8,51.5 9,50.0 10,48.5 11,47.5 12,46.5 13,45.5 14,44.5 15,43.5 16,
9 *43.0 17,42.0 18,41.0 19,40.5 20,39.5 21,39.0 22,38.5 23,38.0 24,37.5 25,
10 *36.5 END
11 *
12 *GPLLOT CUP,t,T
13 *

```

The GPLLOT command gives the following graph on the screen:



8.4.1 Common specifications

In addition to general specifications (like `HOME`, `SIZE`, `XDIV`, `YDIV`, `FRAME`, `FRAMES`, `HEADER`, `TEXTS` and `FILLS`), the following extra specifications are available:

The scales for the X and the Y axis are defined by the specifications `XSCALE` and `YSCALE`, respectively. They have the same structure. If the scales are identical, it is sufficient to use a specification word `SCALE` for this common scale.

`XSCALE=<min.value> , <2nd value> , . . . , <max.value>`

determines a linear scale and the scale notations on the X axis. If `XSCALE` (or `SCALE`) is not given, scaling is automatic according to the current data.

In **automatic scaling**, Survo has to scan the active part of the current data set in order to find minimum and maximum values of pertained variables. On the basis of this information, nice rounded values are selected for the ranges and for the scale notations. Thereafter the data is scanned again and plotted one observation at a time.

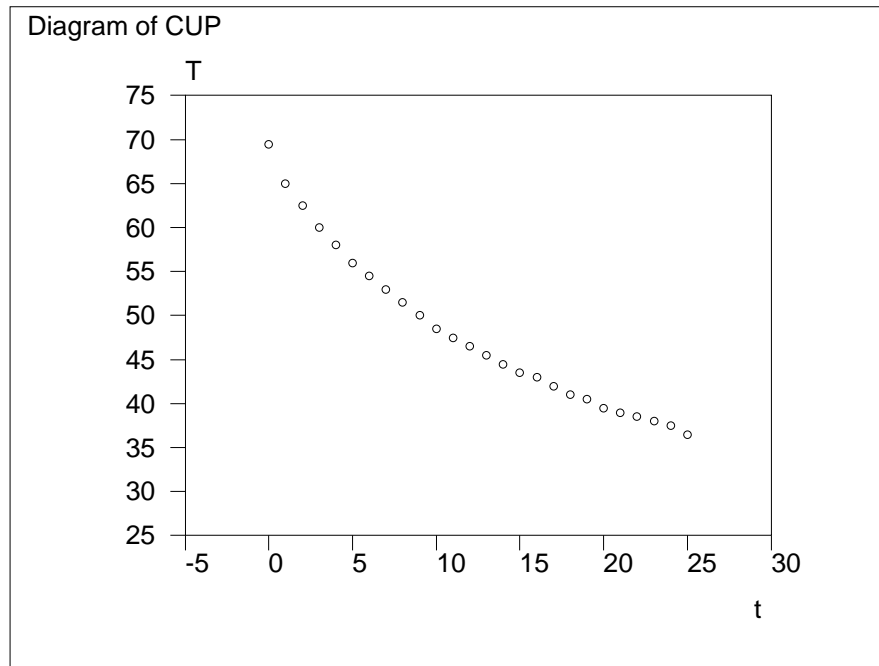
The search for minima and maxima can be avoided in data files where a notation `{min_value,max_value}` appears in the extended field names. In that case, these limits form the basis for automatic scaling.

In the preceding example, automatic scaling was not perfect since the most extreme observations fell on the borders of the plotting area. By adding speci-

fications

`XSCALE=-5(5)30 YSCALE=25(5)75`

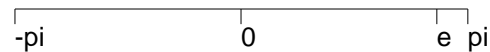
a better result is achieved:



Any point given in XSCALE may be labelled by an arbitrary text given after a colon ':'. For example,

`XSCALE=-3.1416:-pi,0,2.7183:e,3.1416:pi`

determines a scale from $-\pi$ to π with scale notations

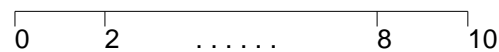


Symbolic notations (without parentheses) are permitted as scale values. Thus the previous scale could be described as

`XSCALE=-pi,0,e,pi pi=3.1416 e=2.7183.`

Each point of notation is indicated by a small tick on the axis. Such ticks may be erased individually by using ':' as above but putting '?' in front of the text.

As an example, the following graph of a plain X axis



has been produced by the PLOT scheme

```

15 1 SURVO 84C EDITOR Sun Jun 28 16:38:33 1992 D:\P2\PLOT\ 100 100 0
29 *
30 *DATA DUMMY:(X,Y) END An empty data set
31 *
32 *PLOT DUMMY,X,Y / DEVICE=PS,XAXIS.PS
33 *HEADER= No header
34 *XSCALE=0,2,4:?.-.-.-.-.-,8,10 Dots in place of 4
35 *YSCALE=0:?,1:? No notations on Y scale (0,1)
36 *FRAME=1 XLABEL= YLABEL= No outer frame and no labels
37 *YDIV=1,0,1 No room for plotting on Y axis
38 *SIZE=900,100
39 *

```

Abbreviations of the form <init value>(<step>)<final value> may also be used. For example, XSCALE=0(1)5,6(2)12,15 generates the scale 0,1,2,3,4,5,6,8,10,12,15 and XSCALE=a(d)a+n*d a=10 d=2 n=4 gives 10,12,14,16,18.

XSCALE=<scale type>,<min.value>,...,<max.value>

determines a nonlinear scale. The rules are the same as those of the previous form of XSCALE.

<scale type> is given in the form *f(x) (*f(y) for YSCALE), where f is any monotonic function of x (y).

In addition to standard functions (sqrt,log,exp etc.) and their combinations, the inverse cumulative distribution function of the standard normal distribution is also available by the name 'probit'.

Examples:

XSCALE=*log(x),1,2,5,10,20,50,100

YSCALE=*probit(y),0.001,0.01,0.1,0.5,0.9,0.99,0.999

The X scale notations can also be located above the graph by giving a XSCALE2 specification, and similarly, the Y scale notations may appear in the right margin, too, by using YSCALE2. Also abbreviated notations like XSCALE2=XSCALE are permitted.

XLABEL=<text_without_spaces> and YLABEL=<text_without_spaces>

give titles for the X and the Y axis. The default is the name of the corresponding variable.

GRID=<X or Y or XY>

draws straight lines parallel to the X axis (GRID=X) or the Y axis (GRID=Y) or both (GRID=XY) through the points indicated by XSCALE and YSCALE, respectively.

GRID=<x_step>,<y_step>

works as GRID above but uses steps <x_step> and <y_step> in drawing. If a step parameter is negative, only a small tick is drawn (outside the inner frame). Mixed notations as GRID=0.1,Y are also possible.

TICK=<x_step> , <y_step>

draws small ticks on the X and Y axes (inside the inner frame) by using step lengths <x_step> and <y_step>.

POINT=<character or name of a variable>

specifies the label to be plotted for each observation in the graph. In case of <name of variable>, the values of that variable will be plotted so that the center of the first character indicates the position of the observation. If POINT is omitted, a small dot or circle is plotted for each observation.

Another form of the POINT specification is

POINT=<marker_type> , <marker_size>

or

POINT=<marker_type> , <max_marker_size> , <marker_var.> , <max_value>

permitting symbols of various sizes depending on the value of the marker variable. The possible values of <marker_type> are 0,1,2, ... and their interpretation depends on the plotting device.

Survo marker types on different devices are:

| | <u>PostScript</u> | <u>Screen</u> | <u>HP</u> | <u>Canon</u> |
|---------|-------------------|---------------|-----------|--------------|
| 0 | filled circle | same | same | same |
| 1 | plus | same | same | same |
| 2 | asterisk | same | same | same |
| 3 | circle | same | same | same |
| 4 | cross | same | same | same |
| 5 | square | same | same | same |
| 6 | filled square | same | same | same |
| 7 | triangle | same | same | same |
| 8 | filled triangle | same | same | same |
| 9 | diamond | same | same | - |
| 10 | filled diamond | same | same | - |
| 11 | - | pixel | - | - |
| default | circle | circle | circle | circle |
| size | 5 | 2 | 1 | 10 |



The size of the symbol in the general case is

$\text{<max_marker_size> * <value_of_marker_variable> / <max_value>}$.

If <max_value> is 0, the values of <marker_var.> will appear as <marker_type> values. Thus different observations may have different markers.

The next exhibit presents possible markers on PostScript plots of Survo in sizes 2,4,6,8,10:

| | | | | | |
|----|---|---|---|---|---|
| 0 | · | • | ● | ● | ● |
| 1 | · | + | + | + | + |
| 2 | · | + | * | * | * |
| 3 | · | ○ | ○ | ○ | ○ |
| 4 | · | × | × | × | × |
| 5 | · | □ | □ | □ | □ |
| 6 | · | ■ | ■ | ■ | ■ |
| 7 | · | △ | △ | △ | △ |
| 8 | · | ▲ | ▲ | ▲ | ▲ |
| 9 | · | ◇ | ◇ | ◇ | ◇ |
| 10 | · | ◆ | ◆ | ◆ | ◆ |

Each of the 'rows' could have been generated by a PLOT scheme of the type

```

11 1 SURVO 84C EDITOR Sun Jun 28 18:21:09 1992      D:\P2\PLOT\ 100 100 0
41 *
42 *SIZE=600,50 XSCALE=0,6 YSCALE=0,2 XLABEL= YLABEL=
43 *DATA K:(X,Y) 1,1 2,1 3,1 4,1 5,1 END
44 *PLOT K,X,Y / POINT=0,10,X,5 HEADER=_0 FRAME=0 DEVICE=PS
45 *

```

LINE=<line type> , <thickness> , <line label>

gives the line type used for connection of consecutive points. <line type> can have values 0,1,2,3,4,5,6,7.

<thickness> (optional) is an integer 1,2,3,...,13. Default=1. In most cases, it is reasonable to use the default value 1 and set the line thickness by a device-specific code word [line_width(x)]. In PostScript plotting, x is given in Points.

<line label> (optional) is a string without spaces to be plotted at the end of the time series (in the right margin). Default=no label.

Possible <line type> values and their interpretation when two consecutive points (x_1, y_1) and (x_2, y_2) are connected are:

- 0 move from (x_1, y_1) to (x_2, y_2) i.e. no visible line (default)
- 1 line from (x_1, y_1) to (x_2, y_2)
- 2 line from (x_1, y_1) to (x_2, y_1) and move from (x_2, y_1) to (x_2, y_2)
- 3 line from (x_1, y_1) to (x_2, y_1) and line from (x_2, y_1) to (x_2, y_2)
- 4 line from (x_1, y_1) to (x_1, y_2) and move from (x_1, y_2) to (x_2, y_2)
- 5 line from (x_1, y_1) to (x_1, y_2) and line from (x_1, y_2) to (x_2, y_2)
- 6 move from (x_1, y_1) to (x_2, y_2) and additional points
- 7 line from (x_1, y_1) to (x_2, y_2) and additional points

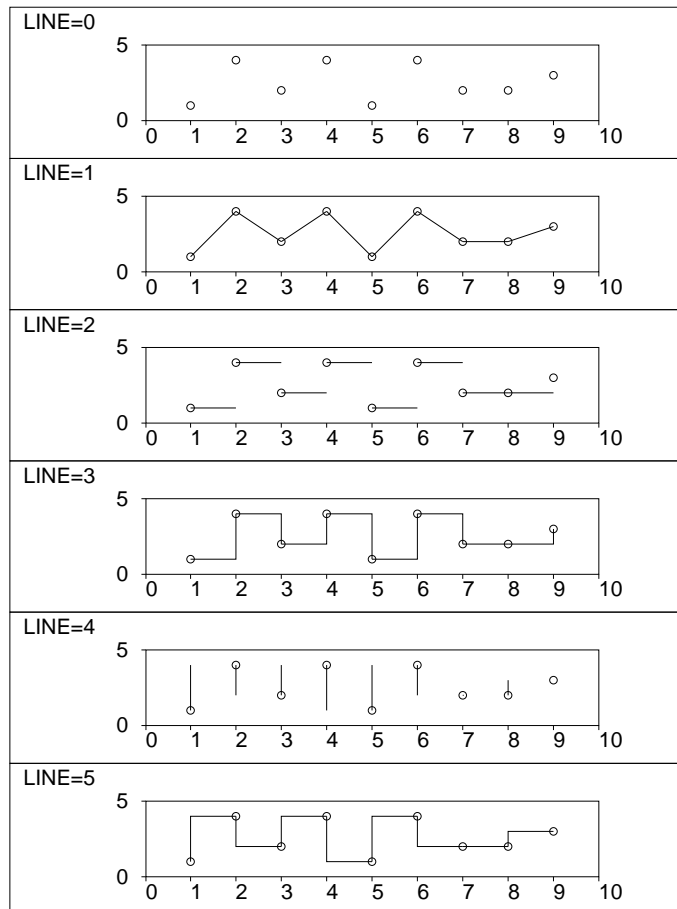
Alternatives 2,3,4 and 5 produce various step functions. The next PLOT

schemes illustrate the nature of line types 0,1,2,3,4,5:

```

11 1 SURVO 84C EDITOR Sun Jun 28 19:53:20 1992 D:\P2\PLOT\ 100 100 0
1 *
2 * *GLOBAL*
3 *SIZE=900,200 XSCALE=0(1)10 YSCALE=0,5 XLABEL= YLABEL=
4 *POINT=3 PEN=[Swiss(8)] YDIV=1,2,1 TICKLENGTH=3
5 *DATA K:(X,Y) 1,1 2,4 3,2 4,4 5,1 6,4 7,2 8,2 9,3 END
6 *
7 *PLOT K,X,Y / LINE=0 HEADER=LINE=0 DEVICE=PS,L0.PS
8 *PLOT K,X,Y / LINE=1 HEADER=LINE=1 DEVICE=PS,L1.PS
9 *PLOT K,X,Y / LINE=2 HEADER=LINE=2 DEVICE=PS,L2.PS
10 *PLOT K,X,Y / LINE=3 HEADER=LINE=3 DEVICE=PS,L3.PS
11 *PLOT K,X,Y / LINE=4 HEADER=LINE=4 DEVICE=PS,L4.PS
12 *PLOT K,X,Y / LINE=5 HEADER=LINE=5 DEVICE=PS,L5.PS
13 *

```



Alternatives 6 and 7 permit additional points and line segments to be connected to the graph.

If <line type> is either 6 or 7, each point (x,y) in the diagram will be connected to additional points given by LINE2, LINE3, ..., LINE6 specifications. LINE2 has the form LINE2=<X2var>, <Y2var> where <X2var> and <Y2var> are two variables (or constants) defining the coordinates of an additional point. Other LINE_n specifications have the same structure.

The end point given by LINE2 can be labelled by a POINT2 specification

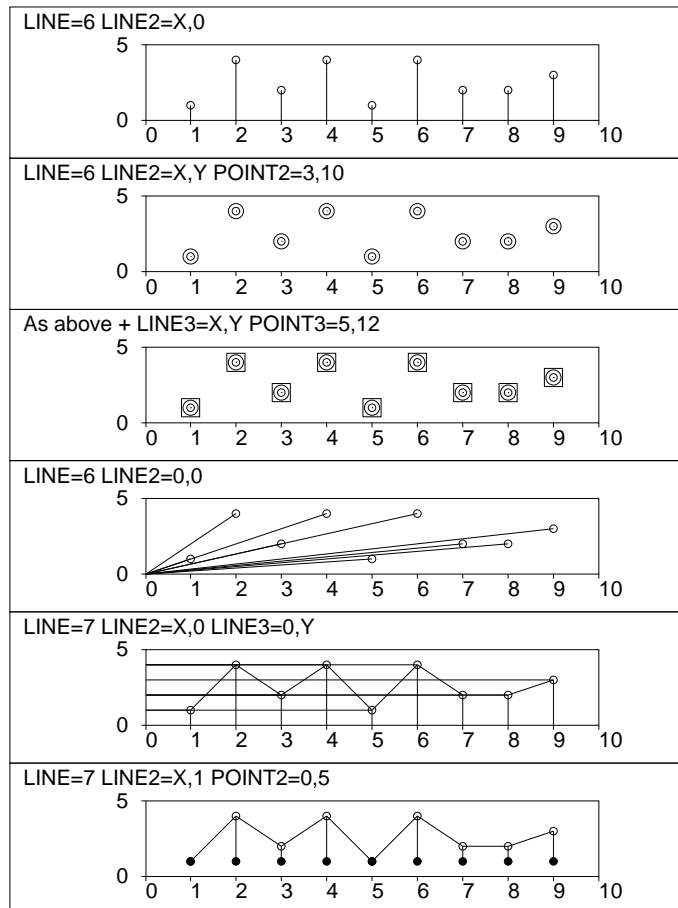
of the form POINT2=<marker_type>, <marker_size> (see POINT). Points defined by LINE3, LINE4, ... specifications may be labelled similarly by POINT3, POINT4, ...

These options offer a rich variety of graph types already with two variables as seen from the next example. The data set K and the global settings are the same as in the preceding example.

```

11 1 SURVO 84C EDITOR Mon Jun 29 09:41:52 1992 D:\P2\PLOT\ 100 100 0
30 *
31 *PLOT K,X,Y / LINE=6 LINE2=X,0 HEADER=LINE=6_LINE2=X;0 DEVICE=PS,L61.PS
32 *
33 *PLOT K,X,Y / LINE=6 LINE2=X,Y POINT2=3,10
34 * HEADER=LINE=6_LINE2=X;Y_POINT2=3;10 DEVICE=PS,L62.PS
35 *
36 *PLOT K,X,Y / LINE=6 LINE2=X,Y POINT2=3,10 LINE3=X,Y POINT3=5,12
37 * HEADER=As_above+_LINE3=X;Y_POINT3=5;12 DEVICE=PS,L63.PS
38 *
39 *PLOT K,X,Y / LINE=6 LINE2=0,0
40 * HEADER=LINE=6_LINE2=0;0 DEVICE=PS,L64.PS
41 *
42 *PLOT K,X,Y / LINE=7 LINE2=X,0 LINE3=0,Y
43 * HEADER=LINE=7_LINE2=X;0_LINE3=0;Y DEVICE=PS,L71.PS
44 *
45 *PLOT K,X,Y / LINE=7 LINE2=X,1 POINT2=0,5
46 * HEADER=LINE=7_LINE2=X;1_POINT2=0;5 DEVICE=PS,L72.PS
47 *

```



8.4.2 Contour ellipses and trend lines

Curves and families of them can be plotted afterwards on an existing scatter diagram or a line graph by using the graphics overlay technique of Survo. However, certain curves, namely contour curves and lines parallel to a linear trend line, are available readily as optional attributes in scatter plots, etc. These curves will be included by giving CONTOUR, BINORM and TREND specifications.

CONTOUR=eps1,eps2, . . . BINORM=E(X),E(Y),S(X),S(Y),R(X,Y) determines contour ellipses to be plotted on levels eps1,eps2, ... on the basis of the two-dimensional normal distribution defined by a BINORM specification. The parameters eps1,eps2, ... refer to the probabilities of an observation to be found inside the ellipse. In BINORM, parameters are the expected values E(X),E(Y), the standard deviations S(X),S(Y) and the correlation coefficient R(X,Y). If BINORM is missing, parameters are selected according to the plotted data. Particularly, eps1=0 causes the principal axes to be plotted.

TREND=C1,C2, . . .

determines lines parallel to a linear trend to be plotted. If the trend is $Y=aX+b$ and the residual variance is s^2 , then the lines $Y=aX+b-Cs$ and $Y=aX+b+Cs$ will be plotted for $C=C1,C2, \dots$ For example, TREND=0 implies the trend itself to be drawn. The trend is estimated by the OLS method from the plotted data. However, if a BINORM specification is given, the trend will be computed according to the corresponding bivariate normal distribution.

As an example, we generate a sample of 1000 observations from a bivariate normal distribution:

```

18 1 SURVO 84C EDITOR Fri Jul 03 13:42:35 1992 D:\P2\PLOT\ 100 100 0
65 *
66 *FILE CREATE BINORM,8,2
67 *Sample of 1000 observations from a bivariate normal distribution
68 *with E(X)=E(Y)=0 S(X)=S(Y)=1 R(X,Y)=0.7
69 *FIELDS:
70 *1 N 4 X
71 *2 N 4 Y
72 *END
73 *
74 *FILE INIT BINORM,1000
75 *
76 *VAR X,Y TO BINORM
77 *R=0.7
78 *U=probit(rnd(1)) V=probit(rnd(1))
79 *X=U Y=R*U+sqrt(1-R*R)*V
80 *

```

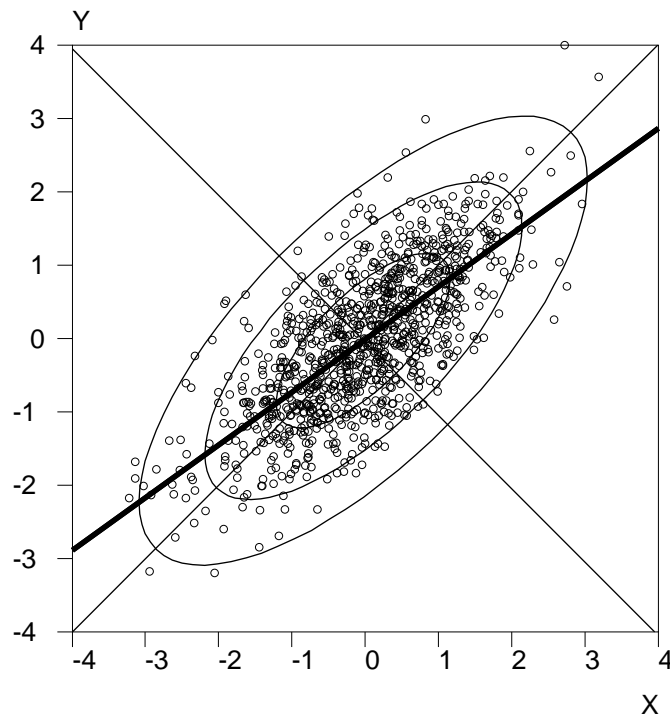
To check the sample, we compute basic statistics by CORR and plot the correlation diagram with contour ellipses, principal axes and the trend (heavier line):

```

16 1 SURVO 84C EDITOR Fri Jul 03 13:46:17 1992 D:\P2\PLOT\ 100 100 0
80 *
81 *CORR BINORM,CUR+1
82 *Means, std.devs and correlations of BINORM N=1000
83 *Variable Mean Std.dev.
84 *X -0.023622 1.008356
85 *Y -0.023204 1.010290
86 *Correlations:
87 * X Y
88 * X 1.0000 0.7177
89 * Y 0.7177 1.0000
90 *
91 *PLOT BINORM,X,Y
92 *SCALE=-4(1)4
93 *CONTOUR=[line_width(0.5)],0,0.5,0.9,0.99
94 *TREND=[line_width(2)],0
95 *SIZE=1164,1164 DEVICE=PS
96 *

```

Diagram of BINORM



The trend and the contour plots are done without a BINORM specification, i.e. they are based on the sample statistics. In this case, the agreement is good. For example, there are about 12 observations outside the 0.99 ellipse; 10 would be the theoretical mean number.

8.4.3 Time series

When plotting a time series, the PLOT operation is usually written with a `LINE=1` specification in the form

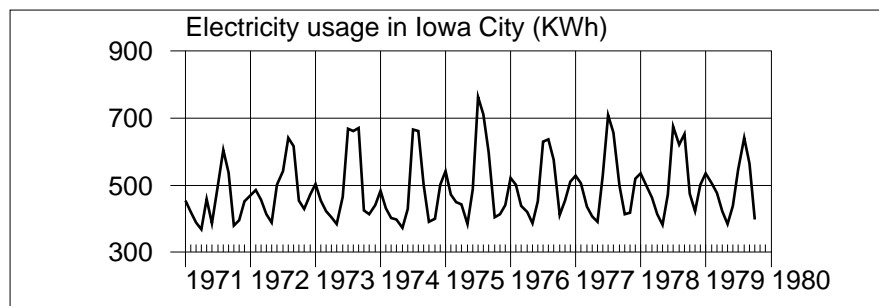
```
PLOT <data>, TIME(<time_variable>), <Y_variable>.
```

`TIME` is an artificial variable with values 1,2, ..., N where N is the number of observations. The `XSCALE` specification should be given according to these values, and the scale points are denoted by using the corresponding values of the `<time_variable>`.

For example, if a monthly time series 'El.usage' from data IOWA is plotted by using the variable 'Year' as a `<time_variable>`, the PLOT scheme

```
30 1 SURVO 84C EDITOR Mon Jun 29 12:00:46 1992 D:\P2\PLOT\ 150 100 0
1 *
2 *(Source: Abraham and Ledolter: Statistical Methods for Forecasting)
3 *Monthly average residential electricity usage in Iowa City
4 *(in kilowatt-hours), January 1971 to October 1979
5 *
6 *DATA IOWA,A,B,C
7 C Year Month El.usage
8 *
9 A 1971 1 454
10 * 1971 2 421
11 * 1971 3 389
12 * 1971 4 368
13 * 1971 5 460
14 * 1971 6 386
15 * 1971 7 501
16 * 1971 8 606
17 * 1971 9 539
18 * 1971 10 381
19 * 1971 11 396
20 * 1971 12 452
21 * 1972 1 470
22 * 1972 2 485
23 * 1972 3 456
-----
113 * 1979 9 566
114 * 1979 10 399
115 * 1979 11 -
116 * 1979 12 -
117 B 1980 1 -
118 *
119 *PLOT IOWA, TIME(Year), El.usage-
120 *HEADER= YLABEL=Electricity_usage_in_Iowa_City_(KWh) XLABEL=
121 *XSCALE=1(12)109 GRID=XY TICK=1 LINE=[line_width(1)],1 SIZE=1164,400
122 *DEVICE=PS
123 *
```

produces a picture where the years are labelled and the months are denoted by ticks:



The following specifications are helpful in time series plotting:

`LAG=<X_step>, <Y_step>`

indicates the amount of dislocation of the graph in the current coordinate system (given by `XSCALE` and `YSCALE`). `<Y_step>` is optional and default is `LAG=0, 0`.

The main purpose of `LAG` is to enable plotting of lagged variables or positioning of the observation points in the middle of time intervals (`LAG=0.5`).

`FILL=<density>, <init.val.>, <fin.val.>, <base val. or var.>, <line_type>`

draws line segments parallel to `Y` axis from the points on the time series curve to another time series `<base variable>` or to a fixed `<base value>`.

`<base val. or var.>` is optional and the default is 0. The area to be filled by these line segments is limited by the optional parameters `<initial value>`, `<final value>`. The default is the whole plotting range.

`<density>` (an integer 1,2,3...) gives the gap between the consecutive fill lines as a multiple of the plotting unit.

`<line_type>` (optional) gives the form of the line for the `<base variable>`. Possible values of `<line_type>` are 1,2,3,4,5 (default 1), and their interpretation is the same as that of the first parameter (line type) of the `LINE` specification.

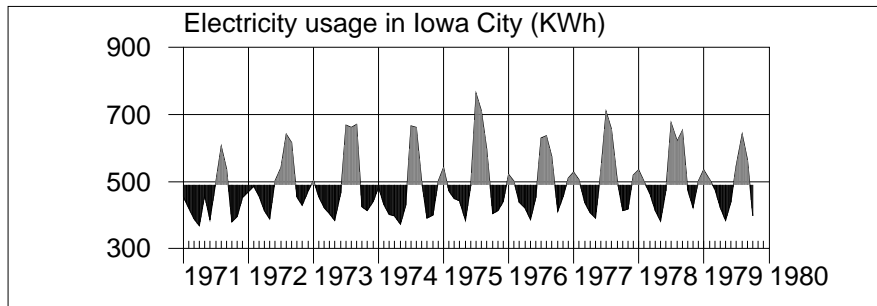
`FILL-=<density>`

supports the `FILL` specification by giving the possibility of filling the negative parts between the two series by using another density (color). If `FILL-` is not given, the same fill is applied to both positive and negative parts.

The previous `PLOT` scheme for `IOWA` could be extended by specifications
`FILL=[line_color(0.5)],1,1,106,490 / Mean of the series is 490.`
`FILL-=[line_color(0)],1.`

`[line_color(x)]` is here a device-specific control word giving the fill color in terms of PostScript gray scale from 0 (black) to 1 (white). The graph will be

changed into the form:



8.4.4 Multiple time series

Several Y variables are plotted simultaneously against one X variable (usually TIME) by giving the PLOT operation in the form
 PLOT <data>, <X_variable>, <Yvar1>, <Yvar2>, ...

Consecutive points of each <Yvar> are connected by a line specified by LINE. Each <Yvar> can have a line type of its own by giving specifications of the form

<Yvar>LINE=<line type>, <thickness>, <line label> .

The consecutive points of each <Yvar> are labelled by the POINT specification. Each <Yvar> can have a label of its own by giving specifications of the form

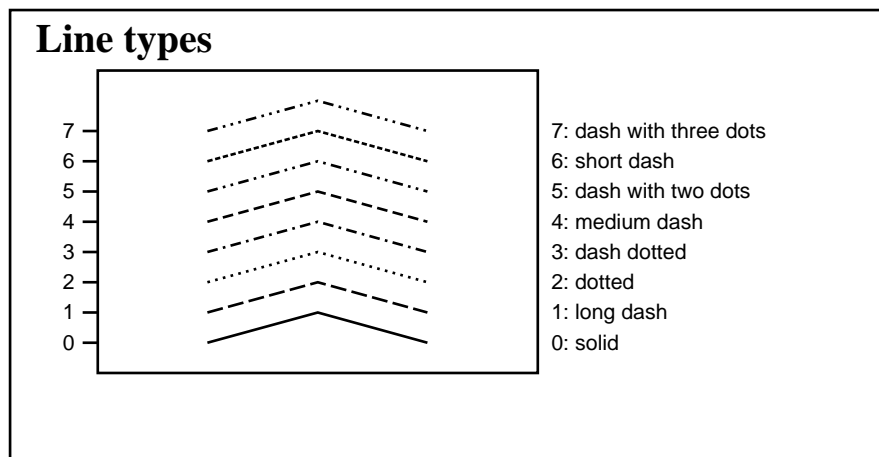
<Yvar>POINT=<list of parameters according to POINT> .

The next PLOT scheme presents Survo PostScript line types as 'multiple time series'. Here typical a device-dependent code word [line_type(x)] is used for $x=0,1,2,\dots,7$.

```

36 1 SURVO 84C EDITOR Mon Jun 29 19:16:10 1992 D:\P2\PLOT\ 240 100 0
1 *
2 *HEADER=[Times(15)],Line_types SIZE=1164,600
3 *DATA TEST
4 *X Y0 Y1 Y2 Y3 Y4 Y5 Y6 Y7
5 *1 0 1 2 3 4 5 6 7
6 *2 1 2 3 4 5 6 7 8
7 *3 0 1 2 3 4 5 6 7
8 *
9 *PLOT TEST,X,Y0,Y1,Y2,Y3,Y4,Y5,Y6,Y7 / DEVICE=PS
10 *
11 *LINETYPE=[line_width(1)] PEN=[Swiss(8)]
12 *Y0LINE=[line_type(0)],1,1,0:_solid
13 *Y1LINE=[line_type(1)],1,1,1:_long_dash
14 *Y2LINE=[line_type(2)],1,1,2:_dotted
15 *Y3LINE=[line_type(3)],1,1,3:_dash_dotted
16 *Y4LINE=[line_type(4)],1,1,4:_medium_dash
17 *Y5LINE=[line_type(5)],1,1,5:_dash_with_two_dots
18 *Y6LINE=[line_type(6)],1,1,6:_short_dash
19 *Y7LINE=[line_type(7)],1,1,7:_dash_with_three_dots
20 *
21 *XSCALE=0:?,4:? YSCALE=-1:?,0(1)7,9:?
22 *XDIV=1,5,4 YLABEL= XLABEL=
23 *

```



8.4.5 Normal probability plots

PLOT <data>, <X_variable>, PROBIT

plots <X_variable> in <data> on a normal probability paper. The data must be sorted in ascending order of <X_variable>.

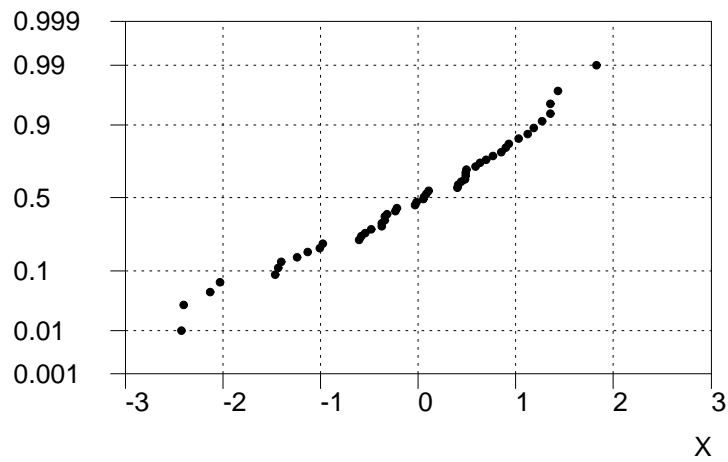
The extra specifications of this form of PLOT are the same as those of PLOT for scatter diagrams. The default YSCALE is in this case

YSCALE=*probit(y), 0.001, 0.01, 0.1, 0.5, 0.9, 0.99, 0.999.

In the following example, the normality of a sum of 12 independent uniform variables is studied by plotting a sample of 50 observations on a normal probability paper.

```
22 1 SURVO 84C EDITOR Mon Jun 29 19:29:35 1992 D:\P2\PLOT\ 240 100 0
27 *
28 *FILE CREATE SUM12,4,1,64,7,50
29 *FIELDS:
30 *1 N 4 X
31 *END
32 *
33 *VAR X=sum12-6 TO SUM12
34 * sum12=for(I=1)to(12)sum(rnd(1))
35 *FILE SORT SUM12 BY X TO SUMSORT
36 *PLOT SUMSORT,X,PROBIT
37 *XSCALE=-3(1)3 SIZE=1164,700 GRID=[line_type(2)],XY
38 *POINT=[line_type(0)],0,5
39 *DEVICE=PS,SUM12.PS
```

Diagram of SUMSORT



8.5 Curves and families of curves

Analytic curves are plotted by using the same general specifications as in scatter plots. The equations of the curves are given directly in the PLOT operation, but various notations appearing in the edit field can also be referred to.

Also families of curves depending on one or more parameters can be plotted. For each parameter, a set of equally spaced values is given and all combinations of parametrized curves are then obtained. Values of parameters can also be taken from a given Survo data set. This feature offers interesting possibilities for creating new graphical presentations of empirical data.

8.5.1 Simple curves

Simple curves are plotted by a command of the form

PLOT Y(X)=f(X) .

Examples:

PLOT Y(X)=2*X+3

PLOT Y(x)=1/sqrt(2*pi)*exp(-x*x/2) / pi=3.14159265

PLOT Y(time)=10*sin(time)

The plotting area is controlled by the XSCALE and YSCALE specifications.

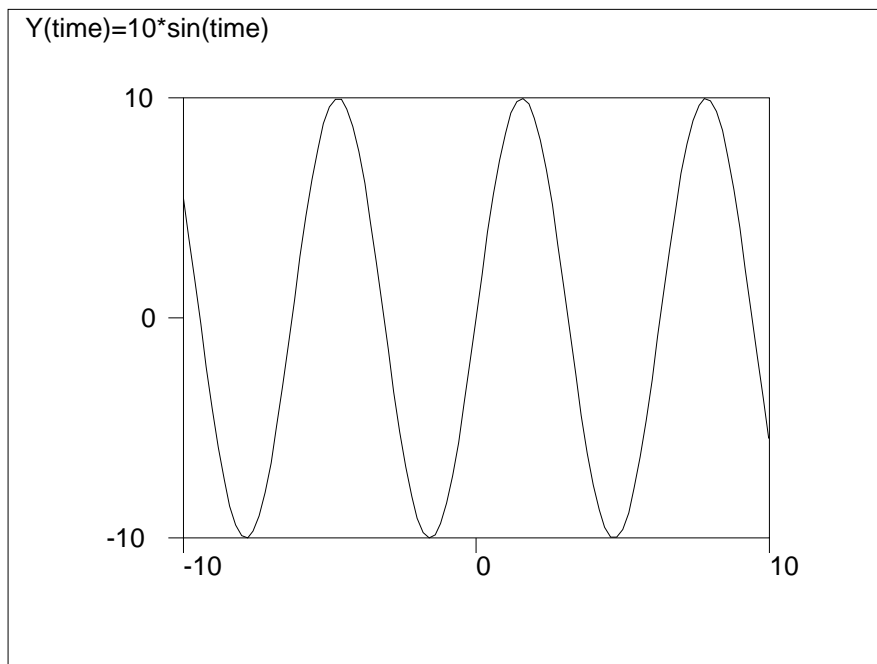
Their default settings are XSCALE=-10,0,10 YSCALE=-10,0,10.

The last example written for screen graphics (GPLOT) as

```

27 1 SURVO 84C EDITOR Wed Jul 01 20:19:04 1992 D:\P2\PLOT\ 100 100 0
 1 *
 2 *GPLOT Y(time)=10*sin(time)-
 3 *
```

gives the following result:



Any word can be used as an argument for the function to be plotted. Above, 'X', 'x' and 'time' were used as arguments. The range and the step length (plotting accuracy) for the argument are given as
`<argument>=<init_value>,<end_value>,<step>` .

If `<step>` is missing, $(\text{end_value} - \text{init_value}) / 100$ is used as a step length. If no range (i.e. no `<argument>` specification) is given, the range of XSCALE is selected.

8.5.2 Parametric representation

In the preceding examples, Y was presented as a function of another variable. For more complicated situations, a parametric representation of a curve $X(t)=f(t)$, $Y(t)=g(t)$ is available. Then, the PLOT operation is written in the form

`PLOT X(t)=f(t), Y(t)=g(t)`

or

`PLOT X(t)=f(t),
Y(t)=g(t)`

where 't' can be replaced by any word. The range and the step length of the parameter are indicated in the same way as those of the argument above.

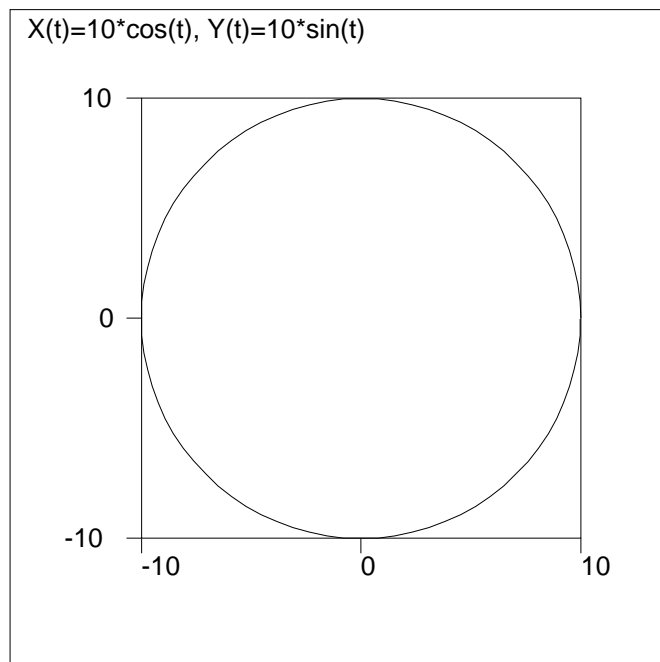
For example,

```

35 1 SURVO 84C EDITOR Wed Jul 01 21:49:58 1992 D:\P2\PLOT\ 100 100 0
6 *
7 *G PLOT X(t)=10*cos(t),Y(t)=10*sin(t)
8 *t=0,2*pi,pi/40 pi=3.14156
9 *MODE=VGA SIZE=400,400
10 *
11 *PLOT X(t)=10*cos(t),Y(t)=10*sin(t)
12 *t=0,2*pi,pi/40 pi=3.14156
13 *SIZE=873,873 DEVICE=PS
14 *

```

produces a circle of radius 10. The upper G PLOT scheme is for the graphic screen. The lower PLOT scheme is for PostScript printers and the following graph is a product of this scheme:



In fact, we plotted a regular polygon of 80 vertices that well approximates a circle.

In the G PLOT graph, the shape of the circle would be correct since in VGA mode (indicated by `MODE=VGA`) the aspect ratio is sufficiently close to 1. If the EGA mode (usually the default mode or obtained by `MODE=EGA`) were used, the aspect ratio is about 0.75 and the picture would be distorted. In EGA mode, the correct form would be reached by setting `SIZE=465,349` with `XDIV=1,10,1` and `YDIV=1,10,1`, for example. On other devices, the aspect ratio is always 1 and no problems occur.

The expressions appearing in equations must be written according to the rules of editorial arithmetics. Constants and expressions may also be entered sepa-

rately in the same subfield (or in the *GLOBAL* subfield). Also conditional statements (if-then-else) and temporary functions are permitted but not library functions.

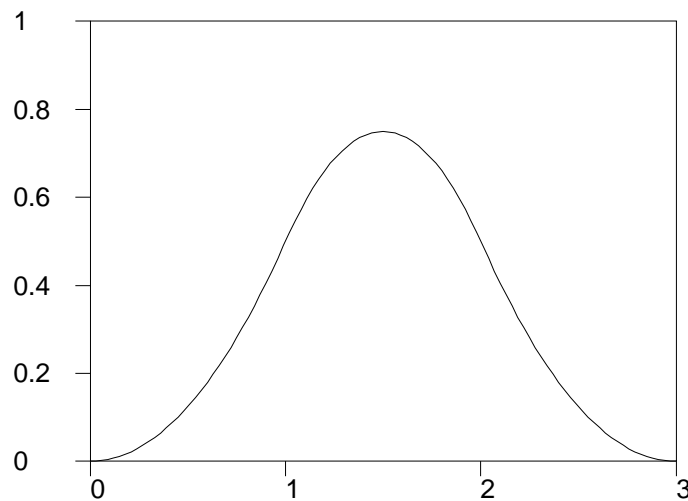
The density function of the sum of 3 independent variables from the uniform distribution on (0,1) is plotted by the GPLOT scheme

```

38 1 SURVO 84C EDITOR Sat Jul 04 11:23:59 1992      D:\P2\PLOT\ 100 100 0
24 *
25 *HEADER=Density_of_the_sum_of_3_independent_variables_from_Uniform(0,1)
26 *
27 *XSCALE=0(1)3 YSCALE=0(0.2)1
28 *
29 *GPLOT y(x)=if(x<1)then(x*x/2)else(y2)
30 *      y2=if(x<2)then(x*x/2-3*(x-1)*(x-1)/2)else((x-3)*(x-3)/2)
31 *

```

Density of the sum of 3 independent variables from Uniform(0,1)



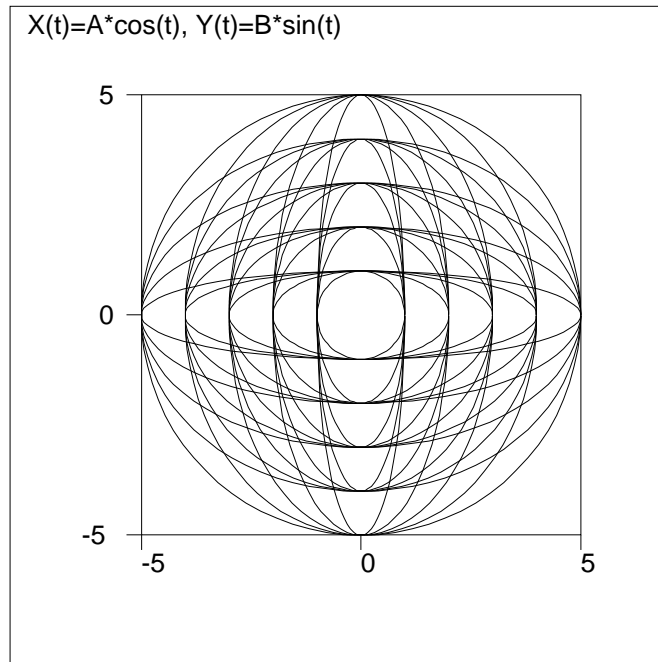
8.5.3 Families of curves

Families of curves depending up to 10 simultaneous parameters are plotted by entering selected values of each parameter as a specification
<name_of_parameter>=<init_value> , <end_value> , <step>.

For example,

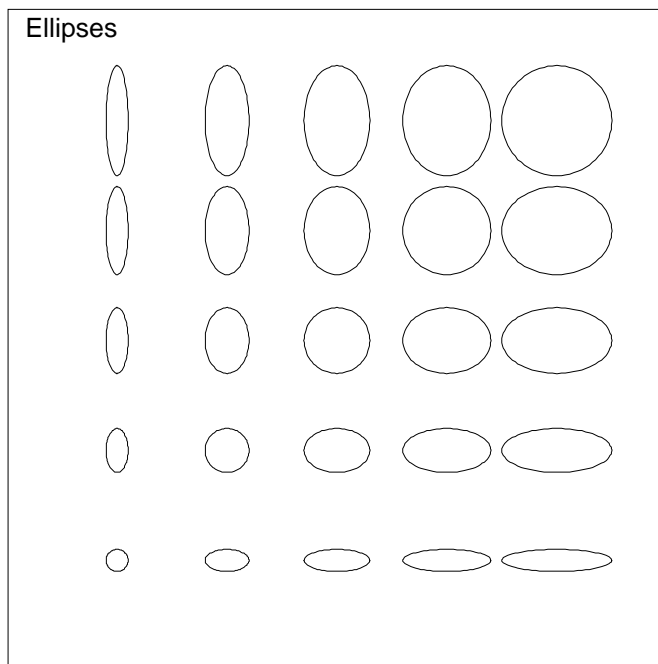
```
33 1 SURVO 84C EDITOR Thu Jul 02 11:53:19 1992 D:\P2\PLOT\ 100 100 0
20 *
21 *PLOT X(t)=A*cos(t),Y(t)=B*sin(t)
22 *A=1,5,1 B=1,5,1 t=0,2*pi,pi/40 pi=3.14159265
23 *SCALE=-5,0,5 SIZE=873,873 DEVICE=PS
24 *
```

plots a family consisting of $5*5=25$ different ellipses:



In this graph, the ellipses have the same center and thus they are strongly overlapping each other. To make a cartoon of distinct ellipses, the varying parameters A,B are also used as coordinates X0,Y0 of the centers as follows:

```
40 1 SURVO 84C EDITOR Sun Jul 05 11:17:51 1992 D:\P2\PLOT\ 100 100 0
24 *
25 *HEADER=Ellipses
26 *N=5 X0=2*A*N-N Y0=2*B*N-N
27 *GPLOT X(t)=X0+A*cos(t),Y(t)=Y0+B*sin(t)
28 *A=1,N,1 B=1,N,1 t=0,2*pi,pi/40 pi=3.14159265
29 *SCALE=0,2*N*N MODE=VGA SIZE=479,479
30 *FRAME=3 XDIV=1,10,1 YDIV=1,10,1
31 *
```



8.5.4* Data values as varying parameters

Parameters can also take their values from variables of a Survo data set, say TEST. Such parameters, say A and B, are given in the form

A=DATA:TEST,Result1

B=DATA:TEST,Result2[-2] (Result2 is lagged by 2.)

Then a curve for each active observation in TEST is plotted using values of variable Result1 in data TEST as parameter A and similarly values of variable Result2 (lagged by 2) in data TEST as parameter B. Only one data set can be used in the PLOT scheme at a time.

All other varying parameters like C=0,10,1 can be used in combination with data-dependent parameters. Then the data values appear just once in each combination of other varying parameters.

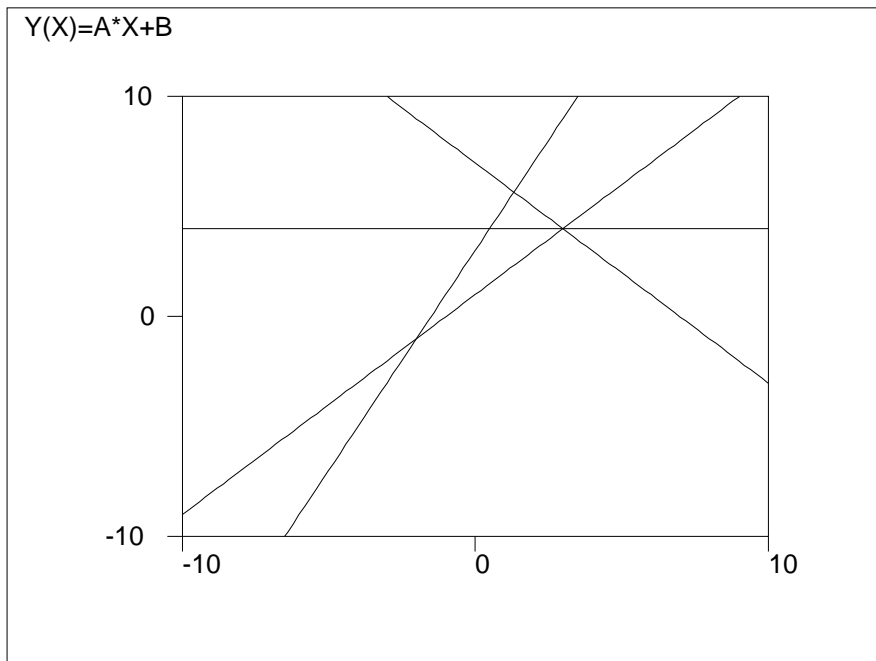
The following examples tell about various possibilities of using data-dependent parameters.

The first example shows how to plot a family of straight lines $Y=A \cdot X+B$ where the parameters A,B are variables a,b in a data set LINES.

```

17 1 SURVO 84C EDITOR Sat Jul 04 15:28:09 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *DATA LINES
3 * a b
4 * 1 1
5 * 2 3
6 * 0 4
7 * -1 7
8 *
9 *GPLOT Y(X)=A*X+B
10 *A=DATA:LINES,a
11 *B=DATA:LINES,b
12 *

```



Polynomial interpolation

In the second example, interpolation of third degree is demonstrated. At first, a data set K of two variables X,Y where $Y=10*\sin(X)+15$ is created and a plot of it is saved in a PostScript file POL3.PS .

```

11 1 SURVO 84C EDITOR Mon Jul 06 09:41:52 1992      D:\P2\PLOT\ 150 100 0
1 *
2 *HEADER=Third_degree_interpolation *GLOBAL*
3 *XSCALE=0(1)12 YSCALE=0(5)30 GRID=XY SIZE=1164,800
4 *.....
5 *DATA K,A,B,N,M
6 M 11 11.1
7 N X Y
8 A 0 15.0
9 * 1 23.4
10 * 2 24.1
11 * 3 16.4
12 * 4 7.4
13 * 5 5.4
14 * 6 12.2
15 * 7 21.6
16 * 8 24.9
17 * 9 19.1
18 * 10 9.6
19 B 11 5.0
20 *
21 *VAR Y=10*sin(X)+15 TO K
22 *.....
23 *PLOT K,X,Y / DEVICE=PS,POL3.PS POINT=3,10
24 *.....

```

The data will now be interpolated as follows. Let (X_1, Y_1) , (X_2, Y_2) , (X_3, Y_3) , (X_4, Y_4) be 4 consecutive points. In the interval (X_2, X_3) , we use the polynomial of third degree that meets those four points exactly.

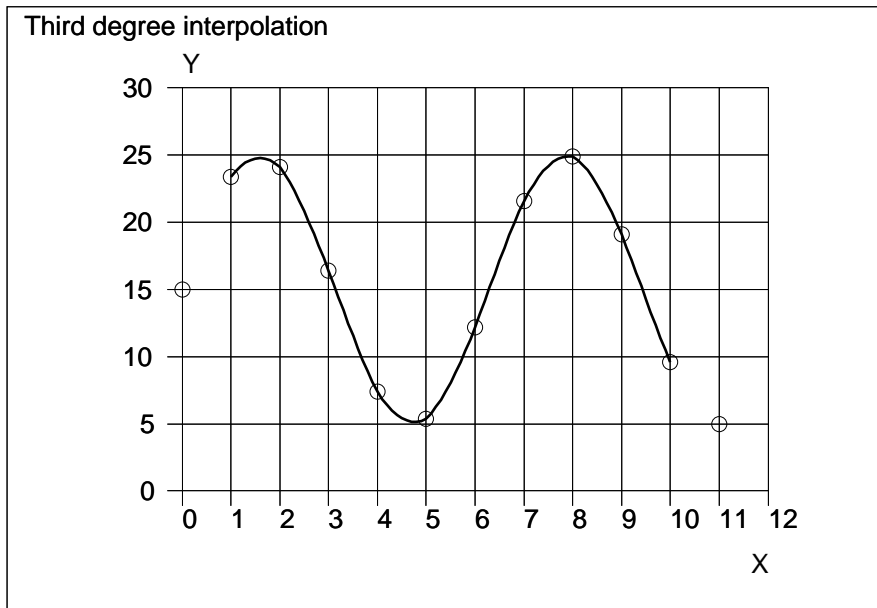
The polynomials are computed and plotted by a PLOT scheme using data-dependent parameters:

```

14 1 SURVO 84C EDITOR Mon Jul 06 09:45:44 1992      D:\P2\PLOT\ 150 100 0
23 *.....
24 *PLOT X(T)=X,Y(T)=F1+F2+F3+F4
25 *T=[line_width(1)],0,1,0.1 IND=X,1,9 DEVICE=PS,POL3F.PS
26 *X1=DATA:K,X[-1] X2=DATA:K,X X3=DATA:K,X[+1] X4=DATA:K,X[+2]
27 *Y1=DATA:K,Y[-1] Y2=DATA:K,Y Y3=DATA:K,Y[+1] Y4=DATA:K,Y[+2]
28 *X=X2+T*(X3-X2)
29 *F1=Y1*(X-X2)*(X-X3)*(X-X4)/(X1-X2)/(X1-X3)/(X1-X4)
30 *F2=Y2*(X-X3)*(X-X4)*(X-X1)/(X2-X3)/(X2-X4)/(X2-X1)
31 *F3=Y3*(X-X4)*(X-X1)*(X-X2)/(X3-X4)/(X3-X1)/(X3-X2)
32 *F4=Y4*(X-X1)*(X-X2)*(X-X3)/(X4-X1)/(X4-X2)/(X4-X3)
33 *.....
34 *PRINT CUR+1,E
35 % 850
36 - picture POL3.PS
37 - picture POL3F.PS
38 E

```

The PRINT operation combines the graphs and produces a picture:



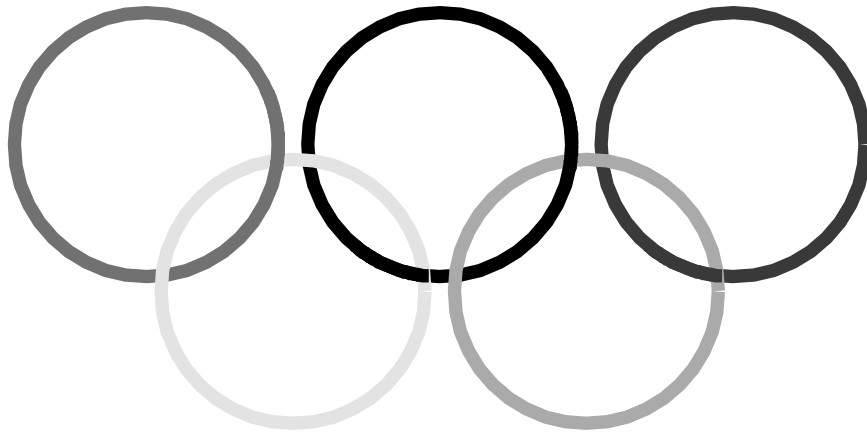
Olympic rings

The next scheme creates the "Olympic Rings". This is a PostScript version with various shades of gray replacing true colors. A color version is presented for the screen in the inquiry system (PLOT? -> Curves -> Families of curves).

```

45 1 SURVO 84C EDITOR Sun Jul 05 15:04:07 1992 D:\P2\PLOT\ 150 100 0
67 *
68 *Example 2: Olympic rings (See how the correct overlapping is achieved)
69 *DATA OLYMPIC
70 * A B C D E Comments
71 * -2 0 5 1 0 Blue A: X coord of the center
72 * 0 0 9 1 0 Black B: Y coord of the center
73 * 2 0 7 1 0 Red C: Color
74 * -1 -1 1 1 0 Yellow D: Length of the arc
75 * 1 -1 3 1 0 Green E: Start angle of the arc
76 * -2 0 5 0.1 -0.2 Blue over yellow
77 * 0 0 9 0.1 4 Black over yellow
78 * 0 0 9 0.1 -0.2 Black over green
79 * 2 0 7 0.1 4 Red over green
80 *
81 *PLOT X(T)=A+R*COS(D*T+E),Y(T)=B+R*SIN(D*T+E)
82 *DEVICE=PS
83 *R=0.90 Radius of the circles
84 *T=[line_width(5)],0,2*pi,pi/20 pi=3.141592653589793
85 *A=DATA:OLYMPIC,A B=DATA:OLYMPIC,B D=DATA:OLYMPIC,D E=DATA:OLYMPIC,E
86 *XSCALE=-3,3 YSCALE=-2,1 SIZE=1164,582
87 *COLOR_CHANGE=C C=DATA:OLYMPIC,C FRAME=0 XDIV=0,1,0 YDIV=0,1,0 HEADER=
88 *

```

In the scheme, a `COLOR_CHANGE` specification selects colors (shades of gray) according to parameter `C` (variable `C` in `OLYMPIC`) for each curve separately. The values of `COLOR_CHANGE` correspond to the values of `SHADING` in bar/pie charts.

To make a color version in PostScript, the only addition is to give suitable `[FILL(x)]` specifications. For example, 'yellow' should be defined by `[FILL(-1)]=0,0,1,0`.

Buffon's needle problem

`SURVO /BUFFON` is available in Survo for simulation of the famous Buffon's needle problem. An essential part of this `sucro` is a dynamic graph on the screen describing toss by toss the needles thrown randomly over a set of equidistant, parallel lines. This graph is generated by using data-dependent parameters when the needles (line segments) are plotted.

Another `sucro` of similar type is `/CONFMEAN`. It generates samples from a normal distribution $N(0,1)$, computes the confidence interval of the mean for each sample and plots these intervals on the screen. To test its functions, one has to activate `/CONFMEAN` without parameters.

The following exhibits are simply snapshots taken from various stages of `/BUFFON`. One can start the `sucro` by the command `/BUFFON` without parameters. This command will then be overwritten by a short description:

```

1 1 SURVO 84C EDITOR Sun Jul 19 09:57:49 1992 D:\P2\PLOT2\ 120 100 0
1 *
2 *Sucro /BUFFON simulates Buffon's needle experiment.
3 *
4 *It creates a file of results and plots them with appropriate
5 *statistics.
6 *
7 *Usage:
8 */BUFFON <name_of_file>,s,N,P
9 *where
10 *   s = needle length,
11 *   N = number of tosses,
12 *   P = confidence level, e.g. 0.95
13 *
14 *_

```

By activating /BUFFON with parameters `TOSSES,1,300,0.95`, the sucro starts by creating a data file `TOSSES` with 10 fields and 300 (missing) observations.

```

34 1 SURVO 84C EDITOR Sun Jul 19 10:01:46 1992 D:\P2\PLOT2\ 120 100 0
13 *
14 */BUFFON TOSSES,1,300,0.95
15 *Creating a Survo data file for results:
16 *FILE CREATE TOSSES,45,10,64,7,300_
17 *FIELDS:
18 * 1 N 8 N
19 * 2 N 4 X
20 * 3 N 4 Y
21 * 4 N 4 A
22 * 5 N 4 X1
23 * 6 N 4 Y1
24 * 7 N 4 X2
25 * 8 N 4 Y2
26 * 9 N 1 I
27 *10 N 8 F
28 *END
29 *

```

The actual simulation takes place by a VAR operation which computes observations describing location of the needle. Also an indicator variable (I) telling whether the needle intersects lines or not and its cumulative sum (F) are generated.

```

38 1 SURVO 84C EDITOR Sun Jul 19 10:06:40 1992 D:\P2\PLOT2\ 120 100 0
29 *
30 *Generating data:
31 *N=ORDER
32 *n=10 X=-0.5+(n+1)*rnd(0) X,Y are coordinates of
33 *   Y=-0.5+(n+1)*rnd(0) the center of the needle.
34 *A=pi*rnd(0) pi=3.141592653589793 Direction of the needle
35 *R=0.5 Needle length/2
36 *X1=X+R*cos(A) Y1=Y+R*sin(A) End point 1
37 *X2=X+R*cos(A+pi) Y2=Y+R*sin(A+pi) End point 2
38 *I=if(int(Y1)=int(Y2))then(0)else(1) Intersection indicator
39 *F=if(ORDER=1)then(I)else(F[-1]+I) Cumulative frequency
40 *VAR N,X,Y,A,X1,Y1,X2,Y2,I,F TO TOSSES_
41 *.....

```

All statistics needed for the dynamic graphical presentation are now ready. The presentation is started by making a background picture consisting merely of explanatory text. This picture is saved in file `BUFFON.SPX`:

```

13 1 SURVO 84C EDITOR Sun Jul 19 10:15:15 1992 D:\P2\PLOT2\ 120 100 0
41 *.....
42 *Making the background graph:
43 *HEADER=Buffon's_needle_problem_(SURVO_84C_sucro_/BUFFON_by_S.Mustonen
44 *GPLOT /FRAME / OUTFILE=BUFFON WAIT=0
45 *SIZE=479,479 FRAME=0
46 *TEXTS=T1 MODE=VGA PALETTE=NUL
47 *T1=#LINES:a,b,380,430,15
48 *
49 aA needle of length s is tossed
50 *randomly on the XY plane. The
51 *probability that the needle
52 *intersects any of the parallel
53 *lines y=k,
54 *k=...-2,-1,0,1,2,...
55 *is 2s/pi when s<=1 and
56 *1-2s/pi*(arcsin(1/s)/s
57 * +sqrt(1-1/s^2)-1)
58 *when s>1. Above pi=3.14159...
59 *
60 *These results make possible to
61 *attain rough estimates for pi.
62 *When the needle intersects lines,
63 *it is colored red; otherwise it
64 *is green.
65 *Below, N is number of tosses,
66 *P is proportion of intersections,
67 *"pi" is current estimate and
68 *l.limit, u.limit are confidence
69 *limits.
70 *
71 *Needle length s=1
72 *Confidence level=0.95
73 * N
74 * P
75 * "pi"
76 * l.limit
77 * u.limit
78 *

```

The final graph shows each toss and displays after each 100th toss relevant statistics (relative frequency of intersections, "estimate of π " and its confidence interval). This dynamic graph is obtained by activating the following PLOT scheme with data-dependent parameters from TOSSES.

```

41 1 SURVO 84C EDITOR Sun Jul 19 10:21:39 1992 D:\P2\PLOT2\ 120 100 0
79 *.....
80 *Making the final graph:
81 *HEADER=Buffon's_needle_problem
82 *GPLOT X(t)=X1+(X2-X1)*t,Y(t)=Y1+(Y2-Y1)*t
83 *t=0,1,1 R=0.5 INFILE=BUFFON OUTFILE=BUFFON WAIT=60
84 *X1=DATA:TOSSES,X1 Y1=DATA:TOSSES,Y1
85 *X2=DATA:TOSSES,X2 Y2=DATA:TOSSES,Y2
86 *MESSAGES=M1,M2,M3,M4,M5
87 *M1=N,100,450,70 N=DATA:TOSSES,N
88 *M2=P,100,450,55 P=F/N F=DATA:TOSSES,F
89 *M3=fpi(P),100,450,40 V=if(R<0.5)then(1)else(-1)
90 *fpi(P):=if(R<0.5)then(4*R/P)else(fpi2(P))
91 *fpi2(P):=4*R/(1-P)*(arcsin(1/2/R)/2/R+sqrt(1-1/4/R/R)-1)
92 *PT=1.9599639845401 PT=N.G(0,1,1-(1-0.95)/2)
93 *T=PT*sqrt(P*(1-P)/N)
94 *M4=fpi(P+V*T),100,450,25 M5=fpi(P-V*T),100,450,10 (confidence limits)
95 *SCALE=-1(1)11 SIZE=479,479 FRAME=0 XDIV=0,8,2 YDIV=1,8,1
96 *GRID=Y
97 *I=DATA:TOSSES,I
98 *COLOR_CHANGE=if(I=0)then(3)else(2)
99 *PALETTE=NUL MODE=VGA
100 *

```

The graph is a family of line segments from points (X1,Y1) to (X2,Y2). Param-

eters $X1, Y1, X2, Y2$ are selected as corresponding variables in data `TOSSES` (lines 84-85). Thus the number of line segments will equal to the number of observations (300) in the `TOSSES` file. The `SCALE` and `GRID` specifications (lines 95-96) take care of drawing the equidistant, parallel lines.

Variable `I` determines the color of the needle (2=red, 3=green) through the `COLOR_CHANGE` specification (lines 97-98). It is red when the needle intersects lines and green otherwise.

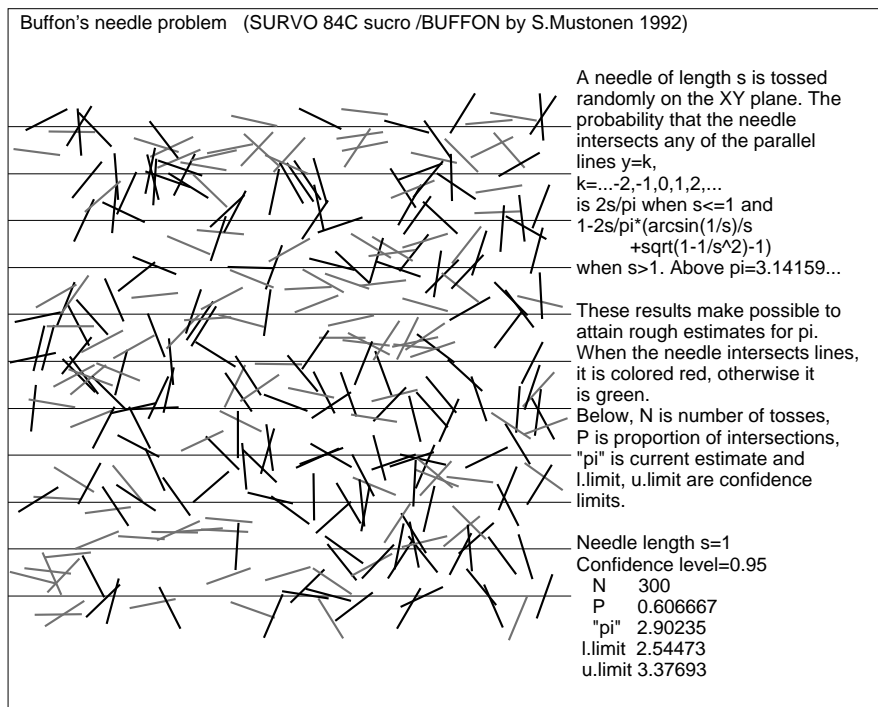
An essential enhancement is provided by a `MESSAGES` specification and its adjoints (lines 86-94). `MESSAGES` itself gives a list of the names of the messages to be displayed regularly on the graphic screen. Each of messages has the structure

`<message>=<function_of_parameters>, n, x, y`

telling that the value of the given function of parameters should be displayed from the position (x, y) onwards for curves nos. $n, 2n, 3n, 4n, \dots$

In this way, statistics related to the current status of the experiment will be presented at regular intervals.

The final graph of this realization of `/BUFFON` is



8.5.5 Integral functions

PLOT INTEGRAL Y(X)=f(X)

plots an integral function $I(X)$ of $f(X)$. $I(X)$ is scaled by $I(X_0)=0$ where X_0 is the initial value of X . The integral is computed numerically by the trapezoid rule by using the step of X as the step length. Thus, the integral is approximated by the same polygon that approximates $f(X)$ in the graph.

For example,

PLOT INTEGRAL Y(X)=2*X / XSCALE=0(1)3

plots the function $f(x)=x^2$ on the interval (0,3).

8.5.6 Specifications in curve plotting

Specifications like XSCALE, YSCALE, XLABEL, YLABEL, GRID, TICK, etc. work as in scatter diagrams. Scale transformations are applied in the same way (see XSCALE).

In curve plotting, certain additional specifications are also available.

INTEGRAL=<positive constant>

normalizes the function to be plotted so that the integral of the function on the range of plotting is equal to the constant given by INTEGRAL. The standard trapezoid rule is used in numeric integration, and the step length is the same as in plotting.

The INTEGRAL specification is useful e.g. in plotting density functions of probability distributions since the function in question may then be entered without any normalizing coefficient. Such coefficients are cumbersome in many distributions. For example, the following plotting scheme depicts the density function of normal distribution $N(0,4)$:

PLOT Y(X)=EXP(-0.5*(X/sigma)^2)

INTEGRAL=1 XSCALE=-8(2)8 YSCALE=0(0.1)0.5 sigma=2

Due to INTEGRAL=1, the integral of $Y(X)$ on the interval (-8,8) will be computed before plotting and the values of $Y(X)$ are then divided by the value of this integral when the curve is drawn.

By entering the plotting scheme in the form

PLOT INTEGRAL Y(X)=EXP(-0.5*(X/sigma)^2)

INTEGRAL=1 XSCALE=-8(2)8 YSCALE=0(0.1)0.5 sigma=2

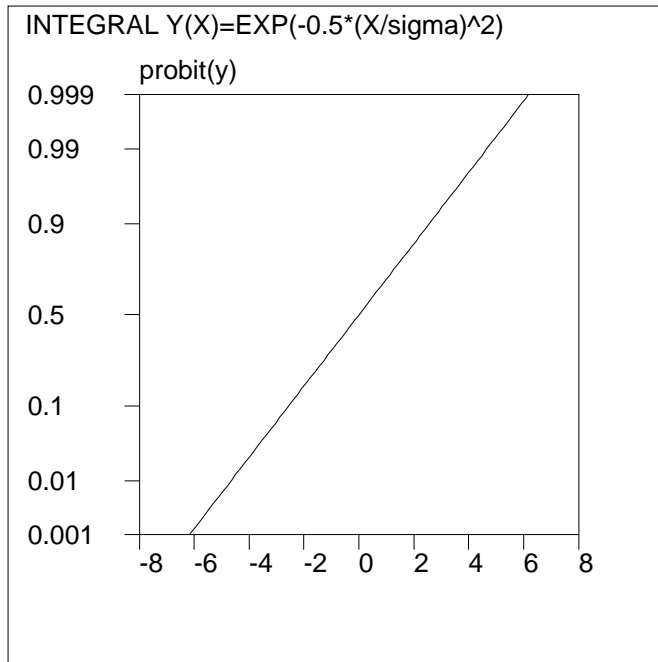
the distribution function of $N(0,4)$ would be plotted. By changing the YSCALE specification into the form

YSCALE=*probit(y),0.001,0.01,0.1,0.5,0.9,0.99,0.999

the distribution function would be plotted on a normal probability paper. Thus, it should appear a straight line.

The last example is presented as a Gplot scheme:

```
42 1 SURVO 84C EDITOR Sun Jul 05 18:49:58 1992 D:\P2\PLOT\ 100 100 0
60 *
61 *Gplot INTEGRAL Y(X)=EXP(-0.5*(X/sigma)^2)_
62 *INTEGRAL=1 XSCALE=-8(2)8 sigma=2
63 *YSCALE=*probit(y),0.001,0.01,0.1,0.5,0.9,0.99,0.999
64 *MODE=VGA SIZE=479,479
65 *
```

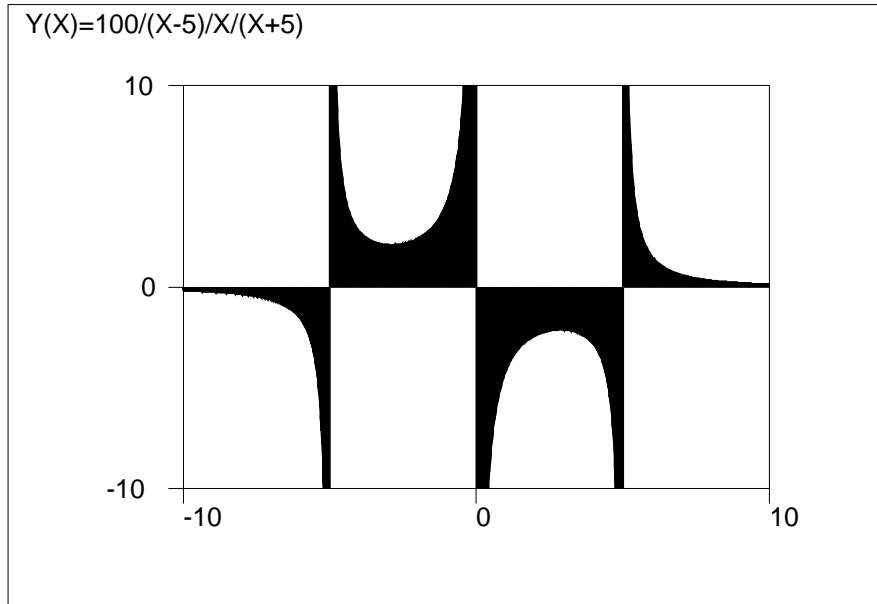


FILL=<density value> , <initial value> , <final value>

draws line segments parallel to Y axis from the points on the curve to the X axis in the interval defined by the optional parameters <initial value> , <final value>. The default is the whole plotting range. <density value> (an integer 1,2,3...) gives the gap between consecutive fill lines as a multiple of the plotting step.

In the next example, FILL helps to see the points of discontinuities of the function:

```
28 1 SURVO 84C EDITOR Sun Jul 05 19:07:06 1992 D:\P2\PLOT\ 100 100 0
73 *
74 *PLOT Y(X)=100/(X-5)/X/(X+5)_
75 *X=[line_width(1)],-10,10,0.02
76 *DEVICE=PS SIZE=1164,800
77 *FILL=1,-10,10
78 *
```



YFILL=<density value> , <initial value> , <final value>
works as **FILL** but draws horizontal fill lines.

OFILL=<density value> , <initial value> , <final value>
draws fill lines from the points on the curve to the origin (0,0).

IFILL=<density value> , <initial value> , <final value>
draws fill lines from the points on the curve to the initial point of the current curve.

COLOR_CHANGE=<expression> , <max>

(for families of curves in screen and PostScript graphics) selects the line color for each curve individually according to the <expression> depending on any of the varying parameters.

<max> is optional. If it is given, color indices are selected modulo <max>. Then the possible values are 0,1,2, ... ,<max>-1 .

In PostScript graphics, values of color indices refer by default to various shades of gray as in the **SHADING** specification of bar/pie charts. However, if a specification of form

[**FILL**(-x)] = <cyan> , <magenta> , <yellow> , <black>

is given for a (positive) color index **x**, the corresponding shade of gray is replaced by this color.

8.6 Contour plots

`PLOT Z(X,Y)=<function of X,Y> / TYPE=CONTOUR`

(in screen and PostScript graphics) draws a contour plot of a function of two variables as a raster image.

The function is written according to the rules of curve plotting. X and Y may be replaced by any words. Scaling is indicated by the `SCALE`, `XSCALE`, `YSCALE`, X and Y specifications, for example, as follows:

`SCALE=0(0.1)1 X=0,1,0.01 Y=0,1,0.01`.

The last parameter (0.01) in X and Y specifications gives the step length. The ranges of `XSCALE` and X (and `YSCALE` and Y) must coincide. The function is evaluated in the middle of each cell defined by the X and Y specifications and these function values are mapped to various colors or shades of gray.

In color mapping, the function values are assumed to be within the interval (0,1). If they are not, the values are treated modulo 1. This feature is very useful when clear contour curves should be obtained. Before color mapping, the function values $f(x,y)$ can be linearly transformed to $a*f(x,y)+b$ by giving a specification `ZSCALING=a,b`.

On the screen, various palettes are provided by the `PALETTE` specification. For different shades of gray, two special .PAL files, `VGAGRAY.PAL` and `EGAGRAY.PAL`, are available. The former gives 16 shades of gray on `MODE=VGA` and the latter 16 slightly colored gray shades on `MODE=EGA`. (In EGA, only 4 genuine gray tones can be produced.) On PostScript devices, the true amount of gray shades greatly depends on the resolution.

The `SCREEN=NEG` specification reverses the order of the palette colors or gray shades. Default is `SCREEN=POS`.

On PostScript devices, the `SCREEN` specification has an extended form `SCREEN=<POS or NEG>, <raster_width_(samples/inch)>, <raster_angle>`.

On the small laser printers (300 pixels/inch), the default setting is `SCREEN=POS,60,0`.

A simple paraboloid is plotted on the screen by the `PLOT` scheme:

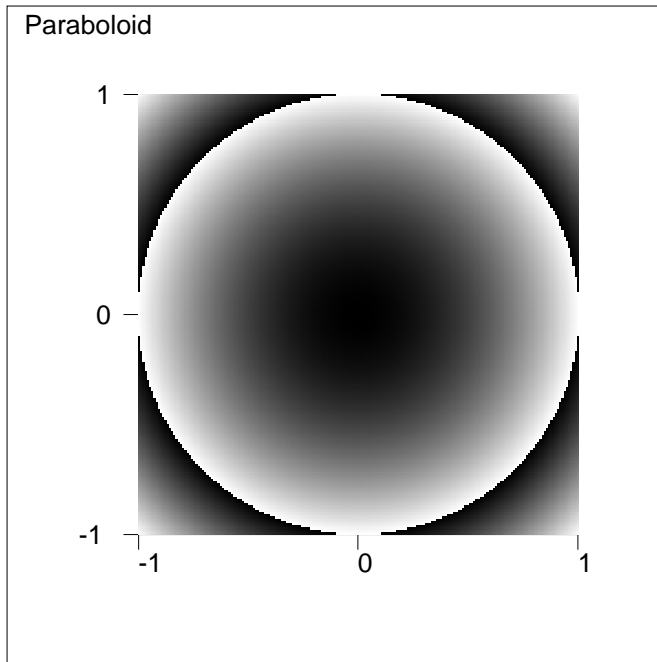
```

21 1 SURVO 84C EDITOR Mon Jul 06 15:00:31 1992 D:\P2\PLOT\ 100 100 0
1 *
2 *HEADER=Paraboloid
3 *GPLOT Z(X,Y)=X*X+Y*Y / TYPE=CONTOUR
4 *SCALE=-1,0,1 X=-1,1,0.01 Y=-1,1,0.01
5 *ZSCALING=1,0
6 *MODE=VGA SIZE=479,479
7 *PALETTE=PAL1,VGAGRAY,EGAGRAY
8 *

```

The `PALETTE` specification selects `PAL1` as the primary set of colors and the graph is displayed row by row in these colors. When the contour plot is ready, one can test other alternatives (`VGAGRAY`, `EGAGRAY`) simply by pressing 'B' or 'C'. The primary palette (`PAL1`) is obtained again by 'A'. The next exhibit corresponds to the alternative `VGAGRAY` but it is obtained by a modi-

fied PostScript PLOT scheme (by using DEVICE=PS and SIZE=873,873).



The graph on the front cover of this book is a modification of the previous one and made by the PLOT scheme

```

42 1 SURVO 84C EDITOR Tue Aug 11 15:46:39 1992 D:\P2\PLOT2\ 100 100 0
50 *
51 *PLOT Z(X,Y)=if(R<1)then(A)else(2047/2048)_
52 *R=X*X+Y*Y A=R*(1+0.2*sin(10*X)+0.2*sin(15*Y))
53 *X=-1,1,0.002 ZSCALING=8,0
54 *Y=-1,1,0.002
55 *TYPE=CONTOUR DEVICE=PS
56 *SIZE=1500,1500 XDIV=0,1,0 YDIV=0,1,0 FRAME=0 HEADER=
57 *

```

*Influence curves of the correlation coefficient

As a more advanced example, the influence function of the correlation coefficient is considered. Assume that we have a sample of n observations on two variables. Let the correlation coefficient computed from this sample be r . If now one more observation x,y is obtained, it usually changes the correlation coefficient a little; let the new value be $r(x,y)$. We shall study the function $z(x,y)=r(x,y)-r$ i.e. the change of the value of the correlation coefficient due to a new observation x,y . It turns out that $z(x,y)$ can be written in terms of $n, x, y, r, \bar{x}, \bar{y}, s_x,$ and s_y where \bar{x}, \bar{y} are means and s_x, s_y standard deviations of the original sample. The details are presented in the next example where the original sample has been selected from the DECA data file.

At first, the essential statistics are computed from DECA and copied to line 12 by using notations above:

```
47 1 SURVO 84C EDITOR Mon Jul 06 17:20:40 1992 D:\P2\PLOT2\ 100 100 0
1 *
2 *CORR DECA,3 / VARS=Height,Weight
3 *Means, std.devs and correlations of DECA N=48
4 *Variable Mean Std.dev.
5 *Height 186.9583 5.090493
6 *Weight 85.56250 6.847600
7 *Correlations:
8 * Height Weight
9 * Height 1.0000 0.8522
10 * Weight 0.8522 1.0000
11 *.....
12 *r=0.85 mx=186.96 my=85.56 sx=5.09 sy=6.85 n=48_
```

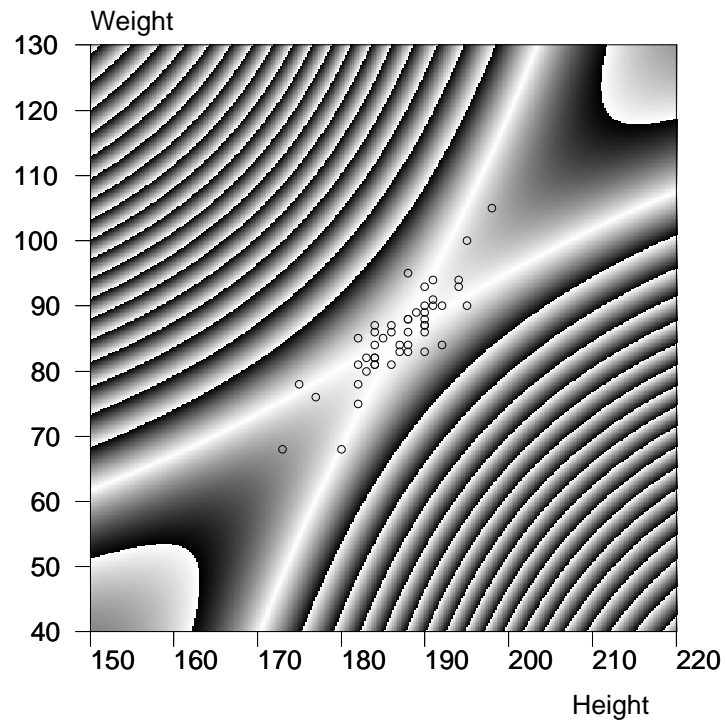
The contour plot as well as a standard scatter diagram are made by the following PLOT schemes:

```
31 1 SURVO 84C EDITOR Mon Jul 06 17:24:21 1992 D:\P2\PLOT2\ 100 100 0
11 *.....
12 *r=0.85 mx=186.96 my=85.56 sx=5.09 sy=6.85 n=48
13 *HEADER=Influence curves for the correlation coefficient
14 *PLOT z(x,y)=abs(r*(1-w)+u*v)/w_
15 * u=sqrt(n/(n*n-1))*(x-mx)/sx
16 * v=sqrt(n/(n*n-1))*(y-my)/sy
17 * w=sqrt((1+u*u)*(1+v*v))
18 *TYPE=CONTOUR ZSCALING=20,0 (1/0.05=20)
19 * SCREEN=NEG
20 *XSCALE=150(10)220 YSCALE=40(10)130 SIZE=1164,1164
21 *x=150,220,0.2 y=40,130,0.2
22 *DEVICE=PS,INF.PS
23 *.....
24 *PLOT DECA,Height,Weight
25 *XSCALE=150(10)220 YSCALE=40(10)130 SIZE=1164,1164
26 *HEADER=
27 *DEVICE=PS,DECA.PS
28 *.....
```

In this application, the original function values are multiplied by 20 ($ZSCALING=20,0$) which gives a complete cycle of shadings when the function value changes by $1/20 = 0.05$. Thus, the final graph will depict contours of r with increments of 0.05.

By printing the graphs on each other, one can see that, for example, a new observation $x=190, y=40$ would decrease the original r from 0.85 by 6×0.05 to 0.55.

Influence curves for the correlation coefficient



8.7 Plots of multivariate data

Survo offers several means for graphic presentation of statistical data sets of more than two variables. Besides various enhancements related to scatter diagrams or to plotting of curves with data-dependent parameters, the following techniques are provided:

| | |
|---------------|---|
| TYPE=MATRIX | Data matrix plotted as a raster image |
| TYPE=DRAFTS | Draftsman's display (Matrix scatter plot) |
| TYPE=CHERNOFF | Chernoff's faces |
| TYPE=ANDREWS | Andrews' function plot |
| TYPE=PROFILE | Profile symbol plot |
| TYPE=STARS | Star symbol plot |

These special types are available on screen and PostScript graphics only.

8.7.1 Enhancements in standard plots

In scatter diagrams, interplay between two variables is studied. Each observation is represented as a point in a two-dimensional graph. By extending the concept of 'point', additional information from other related variables can be included.

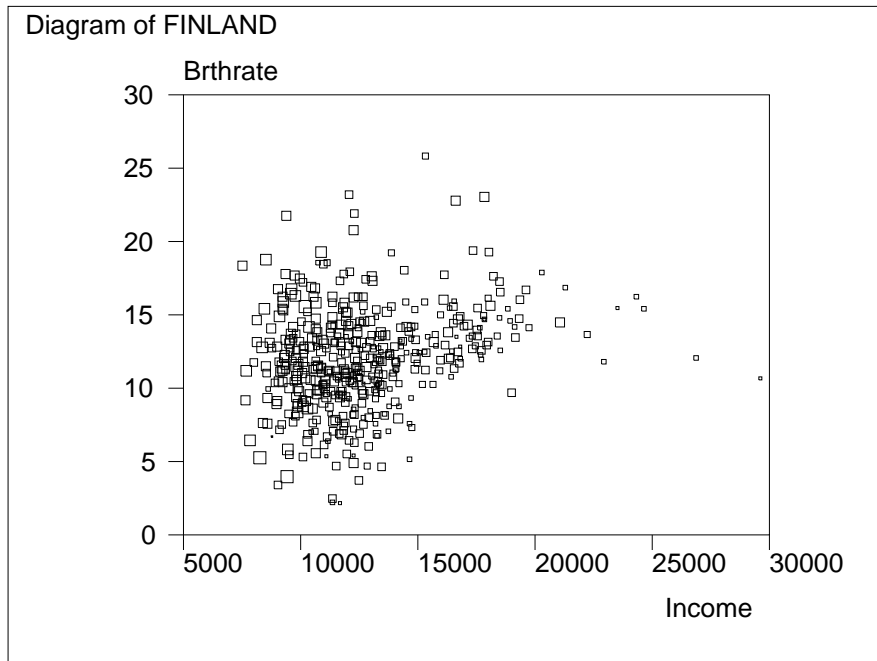
The simplest way of doing so in Survo graphics is to use the general form of the POINT specification (see 8.4). It permits a third variable to control the size of the point. The specification LINE=6 with different extensions gives still more possibilities.

In the next scatter diagram of variables 'Income' and 'Brthrate' of the FINLAND data, the point size is proportional to $\text{Taxdiff} = \text{Tax} - 12$.

```

30 1 SURVO 84C EDITOR Tue Jul 07 12:19:07 1992 D:\P2\PLOT2\ 100 100 0
1 *
2 *VAR Taxdiff=Tax-12 TO FINLAND
3 *
4 *GPLOT FINLAND, Income, Brthrate
5 *POINT=5,10,Taxdiff,8
6 *

```



*Rectangle plots

A more general approach is provided by curve plotting with data-dependent parameters. As an example, we shall make a graph of four variables. Two of them determine the coordinates and two remaining the width and height of a rectangle for each observation. The observation lies in the middle of the rectangle.

The following PLOT scheme is partly data-dependent (lines 2-11) and partly general (lines 13-21). By editing the data-dependent part, modifications for any data set and variables are easily obtained.

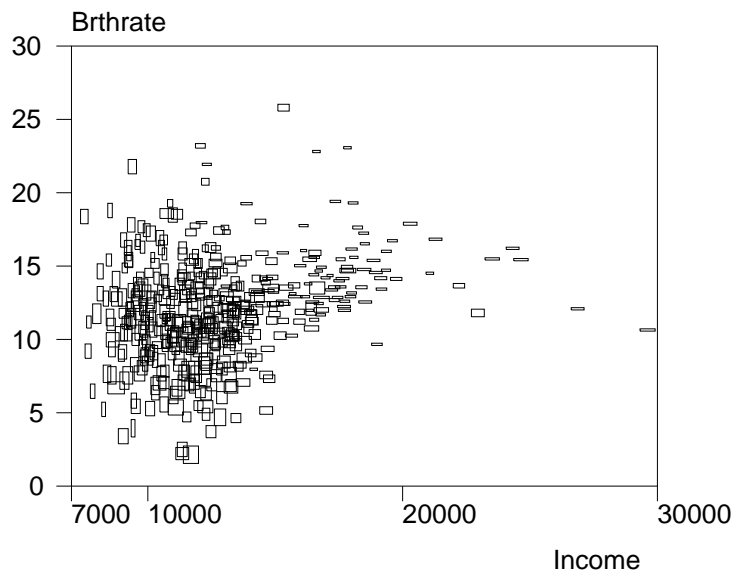
Also the general part can be modified; instead of rectangles formed by four line segments, other piecewise defined data-dependent curves could represent observations.

```

22 1 SURVO 84C EDITOR Tue Jul 07 13:06:05 1992      D:\P2\LOT2\ 100 100 0
1 *
2 *DATA-DEPENDENT PART:
3 *  HEADER=FINLAND: _Rectangle_width_=20-Tax____Rectangle_height_=Agri+1
4 *  X-Y variables:
5 *      VX=DATA:FINLAND,Income VY=DATA:FINLAND,Brthrate
6 *      XSCALE=7000,10000,20000,30000 YSCALE=0(5)30
7 *      XLABEL=Income                YLABEL=Brthrate
8 *  Rectangle variables:
9 *      Vx=DATA:FINLAND,Tax   Vy=DATA:FINLAND,Agri
10 *  Transformations:
11 *      vx=100*(20-Vx)   vy=(Vy+1)/6
12 *
13 *GENERAL PART:
14 *  xx=VX-vx/2 yy=VY-vy/2
15 *  T=0,4,1
16 *  X1=if(T<=1)then(xx+T*vx)else(X2)  Y1=if(T<=1)then(yy)else(Y2)
17 *  X2=if(T<=2)then(xx+vx)else(X3)    Y2=if(T<=2)then(yy+(T-1)*vy)else(Y3)
18 *  X3=if(T<=3)then(xx+vx-(T-2)*vx)else(X4)  Y3=if(T<=3)then(yy+vy)else(Y4)
19 *  X4=xx                                Y4=yy+vy-(T-3)*vy
20 *
21 *G PLOT X(T)=X1,Y(T)=Y1
22 *

```

FINLAND: Rectangle width = 20-Tax Rectangle height = Agri+1



8.7.2 Data matrix as a raster image

PLOT <data> / TYPE=MATRIX

makes a matrix plot of the active part of <data> as a raster image.

In the graph, the active variables appear as columns and active observations as rows. For each data value, a box with a color depending on the value of the current variable in the current observation will be drawn.

By default (NORM=C), the values are scaled by columns (i.e. separately for each variable). Other alternatives are indicated by NORM=R (scaling by rows) and NORM=T (uniform scaling over entire data). After scaling, the highest values are set to 1 and lowest to 0. These values are mapped to various colors or shades of gray as in contour plots (see 8.6).

The color for missing values can be given by MISSING=x where $0 \leq x \leq 1$. Default is MISSING=0.

In matrix plots, the columns are labelled by names of active variables and rows by names of cases as in HBAR plots. Setting of labels is adjusted by specifications

ROWLABELS=1, <number_of_label_columns>, <max.length_of_label>

COLUMNLABELS=1, <number_of_label_rows>

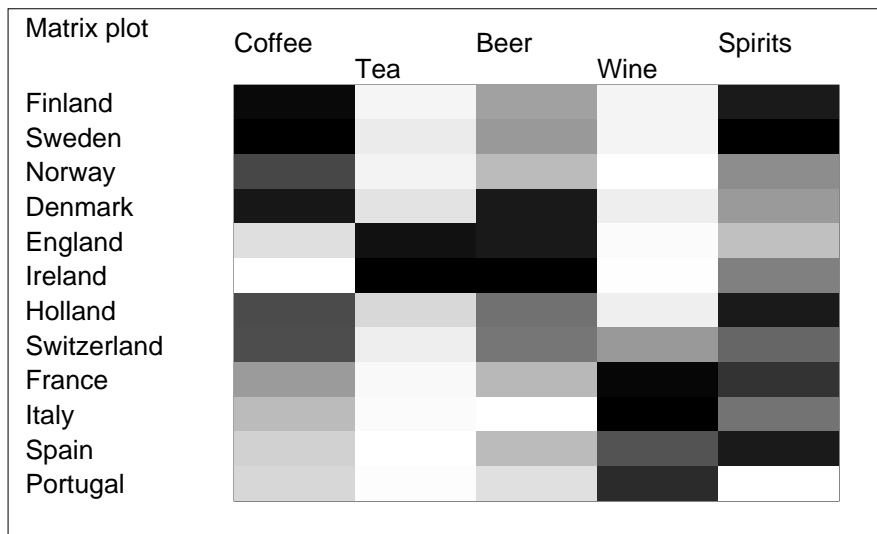
In both specifications, the second parameter is an integer (default 1). Its purpose is to give more space for otherwise too tight setting of labels. The labels are omitted by giving ROWLABELS=0, COLUMNLABELS=0.

The data table COUNTRIES is described as a matrix plot as follows:

```

15 1 SURVO 84C EDITOR Thu Jul 09 12:38:33 1992 D:\P2\PLOT2\ 100 100 0
16 *
17 *Consumption of various beverages in 12 European countries
18 *DATA COUNTRIES
19 * Country Coffee Tea Beer Wine Spirits
20 * Finland 12.5 0.15 54.7 7.6 2.7
21 * Sweden 12.9 0.30 58.3 7.9 2.9
22 * Norway 9.4 0.19 43.5 3.1 1.8
23 * Denmark 11.8 0.41 113.9 10.4 1.7
24 * England 1.8 3.49 113.7 5.1 1.4
25 * Ireland 0.2 3.73 124.5 3.8 1.9
26 * Holland 9.2 0.58 75.5 9.7 2.7
27 * Switzerland 9.1 0.25 73.5 44.9 2.1
28 * France 5.2 0.10 44.5 104.3 2.5
29 * Italy 3.6 0.06 13.6 106.6 2.0
30 * Spain 2.5 0.03 43.6 73.2 2.7
31 * Portugal 2.2 0.03 27.5 89.3 0.9
32 *
33 *PLOT COUNTRIES / TYPE=MATRIX SCREEN=NEG DEVICE=PS
34 *SIZE=1164,700 XDIV=300,800,64 YDIV=50,550,100
35 *COLUMNLABELS=1,2

```



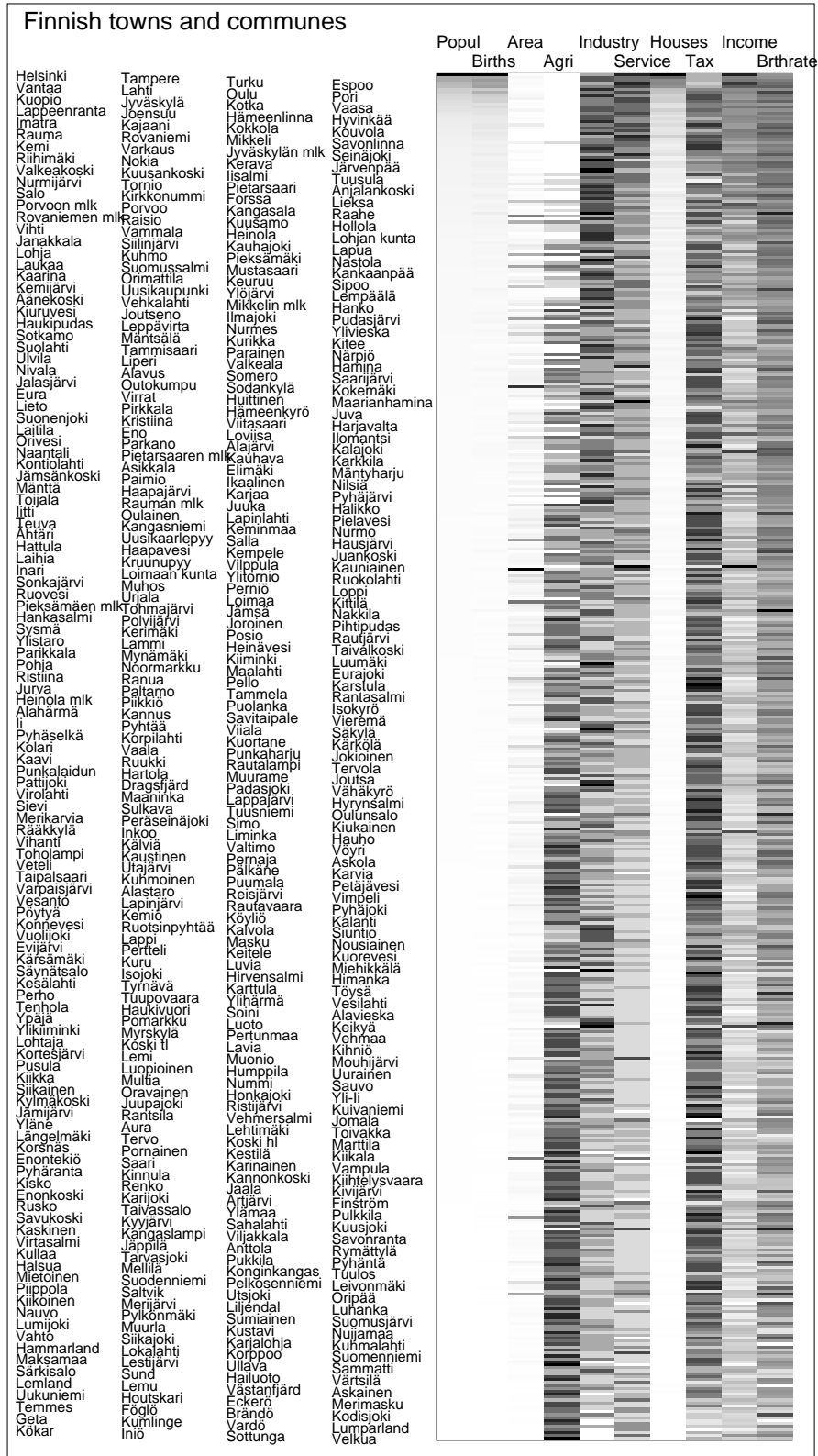
Even fairly large data sets can be visualized by this technique. By sorting the data with respect to some basic variable, one can see how other variables are related to this one. Outliers will appear as striking lines in this compressed data 'spectrum'.

In the next example, the Finnish towns and communes are sorted by their population and a matrix plot of 10 variables is made for the sorted data set.

```

14 1 SURVO 84C EDITOR Thu Jul 09 12:48:45 1992 D:\P2\PLOT2\ 100 100 0
40 *
41 *MASK=A-AAAAAAAAAA--
42 *FILE SORT FINLAND BY -Popul TO FINLAND2
43 *HEADER=Finnish_towns_and_communes
44 *PLOT FINLAND2_ / TYPE=MATRIX SCREEN=NEG DEVICE=PS
45 *SIZE=1164,2100 XDIV=620,514,30 YDIV=30,1970,100
46 *ROWLABELS=[Swiss(6)],1,4,10 COLUMNLABELS=[Swiss(7)],1,2
47 *

```

8.7.3 Draftsman's display

PLOT <data> / TYPE=DRAFTS

plots the draftsman's display, i.e. all pairwise scatter plots of m active variables as an $m \times m$ array of graphs. The output can be limited to the upper triangle of the array by TYPE=DRAFTS,UPPER or to the lower triangle by TYPE=DRAFTS,LOWER.

All normal specifications related to the size and the location of the graph are available. Also the POINT specification can be used as in standard scatter plots (see POINT?).

Scaling of variables is determined automatically according to the minimum and maximum values of the current data. Discrete variables with few possible values (normally producing uninformative plots because many points overlap) can be jittered (scattered) by entering a specification JITTER= k . Then variables with at most k distinct values are randomized uniformly around the true values within a total interval (jitter step) of $(\max - \min)/(h-1)$ where h is the number of distinct values.

The ranges of the variables and their jitter steps are saved in a text file by a specification OUTSCALES=<name_of_file>. The automatic settings of ranges and jitter steps may be changed by editing this file and reproducing the plots with a specification INSCALES=<name_of_file>.

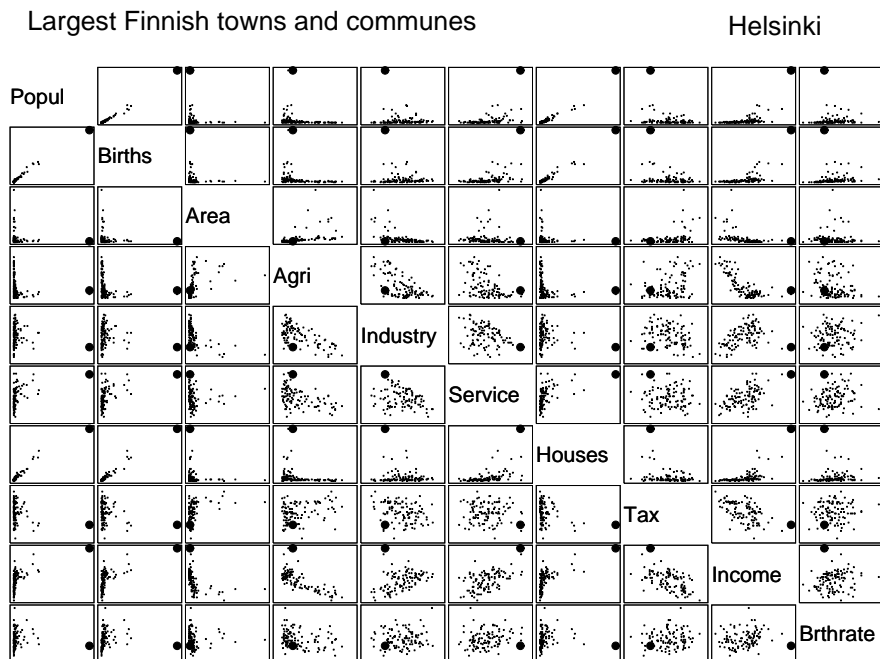
OUTSCALES and INSCALES are also helpful when partial data sets with different point symbols are to be overlaid as seen from the following example.

The largest Finnish towns and communes (population at least 10000 inhabitants) are plotted for 10 variables. The position of one observation (Helsinki) is studied by replotting this case with a larger dot as a marker.

```

14 1 SURVO 84C EDITOR Thu Jul 09 16:39:12 1992 D:\P2\PLOT2\ 100 100 0
1 *
2 *Plotting the whole data (specifying scalings and jittering):
3 *GPLOT FINLAND / TYPE=DRAFTS OUTSCALE=SCALES.TXT JITTER=30
4 *IND=Popul,10000,500000
5 *XDIV=0,1,0 YDIV=0,10,1 HEADER=Largest_Finnish_towns_and_communes
6 *MASK=--AAAAAAAAA MODE=VGA OUTFILE=A
7 *.....
8 *Plotting one particular case (large dots on the previous graph):
9 *GPLOT FINLAND / TYPE=DRAFTS INSCALE=SCALES.TXT
10 *IND=Popul,10000,500000
11 *XDIV=0,1,0 YDIV=0,10,1 HEADER=
12 *MASK=--AAAAAAAAA MODE=VGA INFILE=A POINT=[RED],0,3 TEXTS=Commune
13 *CASES=Commune:Helsinki Helsinki can be replaced by any
14 *Commune=Helsinki,500,450 other case.
15 *

```



8.7.4 Chernoff's faces

PLOT <data> / TYPE=FACES

plots multidimensional data according to a technique presented by H.Chernoff in "Using faces to represent points in k -dimensional space graphically", JASA,68,361-368 (1973).

The mapping from data to various features of the face is defined by a VARIABLES list written after the PLOT line. To create this list, it is best to activate PLOT (with TYPE=FACES) without any list. Then a model of the list with appropriate comments will be displayed below the current PLOT line. A typical list is following:

```

1 1 SURVO 84C EDITOR Thu Jul 09 17:12:33 1992 D:\P2\PLOT2\ 100 100 0
1 *_
2 *VARIABLES: xmin      xmax      Features      fmin fmax
3 *<X1>      <min X1> <max X1> Radius_to_corner_of_face_OP 0.6 1.0
4 *<X2>      <min X2> <max X2> Angle_of_OP_to_horizontal 0.0 0.6
5 *<X3>      <min X3> <max X3> Vertical_size_of_face_OU 0.6 1.0
6 *<X4>      <min X4> <max X4> Eccentricity_of_upper_face 0.5 1.5
7 *<X5>      <min X5> <max X5> Eccentricity_of_lower_face 0.5 1.5
8 *<X6>      <min X6> <max X6> Length_of_nose 0.1 0.5
9 *<X7>      <min X7> <max X7> Vertical_position_of_mouth 0.2 0.8
10 *<X8>     <min X8> <max X8> Curvature_of_mouth_1/R -4.0 4.0
11 *<X9>     <min X9> <max X9> Width_of_mouth 0.2 1.0
12 *<X10>    <min X10> <max X10> Vertical_position_of_eyes 0.0 0.4
13 *<X11>    <min X11> <max X11> Separation_of_eyes 0.3 0.8
14 *<X12>    <min X12> <max X12> Slant_of_eyes -0.5 0.5
15 *<X13>    <min X13> <max X13> Eccentricity_of_eyes 0.3 1.0
16 *<X14>    <min X14> <max X14> Size_of_eyes 0.1 0.2
17 *<X15>    <min X15> <max X15> Position_of_pupils -0.1 0.1
18 *<X16>    <min X16> <max X16> Vertical_position_of_eyebrows 0.2 0.4
19 *<X17>    <min X17> <max X17> Slant_of_eyebrows -0.5 0.5
20 *<X18>    <min X18> <max X18> Size_of_eyebrows 0.1 0.5
21 *END of plotting specifications
22 *

```

The variables selected will be mapped to various features linearly so that $x_{min} \rightarrow f_{min}$ and $x_{max} \rightarrow f_{max}$. This scheme may be freely edited by altering x_{min} and y_{min} values (eventually also f_{min} and f_{max}) to achieve a desired result. For example, the influence of certain variable could be reversed by changing its x_{min} and x_{max} values. To keep certain features constant, use character ‘-’ instead of a variable. Then $(f_{min}+f_{max})/2$ will be used as the value of the feature for all observations.

The default list obtained after activating PLOT with TYPE=FACES but without a list has ‘-’ as each variable, ‘*’ for x_{min} and ‘**’ for x_{max} . If now ‘-’ is replaced by a name of a variable and PLOT is activated, the faces are plotted by using the true minimum value in place of ‘*’ and maximum value in place of ‘**’. (The list is automatically updated with true values terminated by ‘*’ or ‘**’.) The user may then add various features gradually and see their effects immediately. The mapping becomes fixed by erasing ‘*’s.

The PLOT operation for Chernoff’s faces is supported by the extra specifications HEADER, HOME, SIZE, XDIV, YDIV as in curve plotting schemes.

Furthermore, two special specifications LABEL and FSIZE are available. LABEL=<label variable> selects the name of the observation to be printed under the face. If LABEL is omitted, # of observation will be used as default.

FSIZE= c gives the side length of the square to be used as an area for each face. c is given in plotting units and has the default value which is one fifth of the current width of the plot. The specifications SIZE, XDIV, YDIV and FSIZE together determine the layout for the faces.

In screen plotting (GPLOT), when more faces are to be plotted than the selected layout allows, a pause will be created after each page is completed. The next page will be obtained by pressing any key.

On the screen, various parts of the faces can be drawn with different colors possibly depending on values of selected variables. The face and the eyes can also be painted. The colors are selected by adding a COLORS list (before the END line in the VARIABLES list).

A model of such a list is obtained by activating GLOT with TYPE= FACES without a VARIABLES list.

As an example, we study a small data set on skull and teeth measurements of certain human races (3 cases), apes (6 cases) and fossils (6 cases). The 8 variables presented in the data set FOSSILS are transformations of original measurements by means of multiple discriminant analysis. This data was already used by Andrews for illustrating his function plot technique.

Our idea is trying to devise a mapping from these variables to various traits of Chernoff's faces in such a way that the human beings and apes have typical characteristics of their own. We do not necessarily assert that the faces of fossils would then be truelike, but anybody can see that "Proconsul Africanus" is really an odd fellow in this society.

Here is the data set

```

1 1 SURVO 84C EDITOR Thu Jul 09 17:53:58 1992 D:\P2\PLOT2\ 100 100 0
23 *
24 *DATA FOSSILS
25 * Species D1 D2 D3 D4 D5 D6 D7 D8 Label
26 * Westafr -8.09 0.49 0.18 0.75 -0.06 -0.04 0.04 0.03 H1
27 * British -9.37 -0.68 -0.44 -0.37 0.37 0.02 -0.01 0.05 H2
28 * Austral -8.87 1.44 0.36 -0.34 -0.29 -0.02 -0.01 -0.05 H3
29 * Gorilla1 6.28 2.89 0.43 -0.03 0.10 -0.14 0.07 0.08 A1
30 * Gorilla2 4.28 1.52 0.71 -0.06 0.25 0.15 -0.07 -0.10 A2
31 * Orangl 5.11 1.61 -0.72 0.04 -0.17 0.13 0.03 0.05 A3
32 * Orang2 3.60 0.28 -1.05 0.01 -0.03 -0.11 -0.11 -0.08 A4
33 * Chimpan1 3.46 -3.37 0.33 -0.32 -0.19 -0.04 0.09 0.09 A5
34 * Chimpan2 3.05 -4.21 0.17 0.28 0.04 0.02 -0.06 -0.06 A6
35 * Pith.Pek -6.73 3.63 1.14 2.11 -1.90 0.24 1.23 -0.55 F1
36 * Pith.P2 -5.90 3.95 0.89 1.58 -1.56 1.10 1.53 0.58 F2
37 * Par.Robu -7.56 6.34 1.66 0.10 -2.23 -1.01 0.68 -0.23 F3
38 * Par.Cras -7.79 4.33 1.42 0.01 -1.80 -0.25 0.04 -0.87 F4
39 * Megantro -8.23 5.03 1.13 -0.02 -1.41 -0.13 -0.28 -0.13 F5
40 * Proc.Afr 1.86 -4.28 -2.14 -1.73 2.06 1.80 2.61 2.48 F6
41 *

```

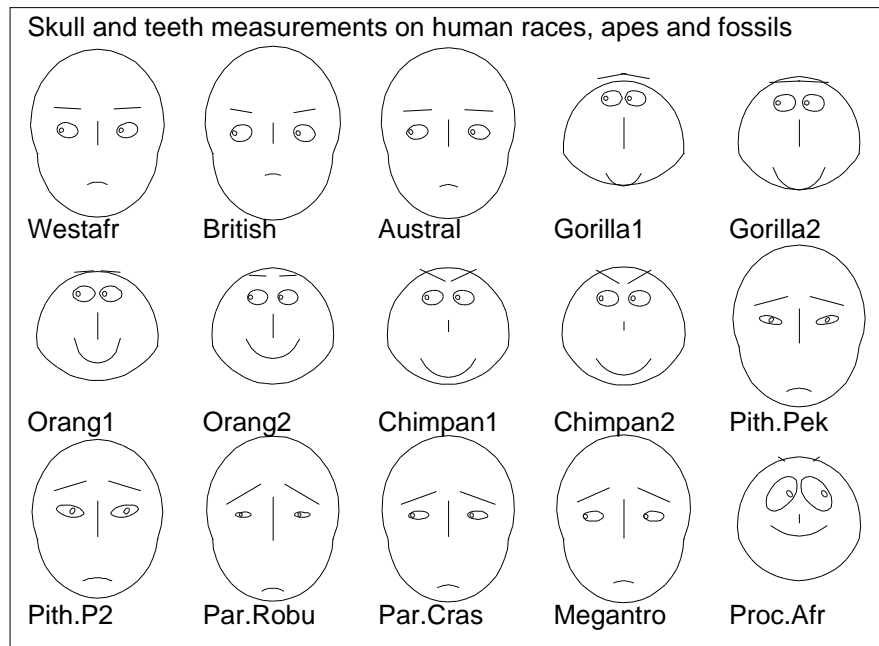
and this is the PLOT scheme

```

13 1 SURVO 84C EDITOR Thu Jul 09 17:55:19 1992 D:\P2\PLOT2\ 100 100 0
41 *
42 *HEADER=Skull_and_teeth_measurements_on_human_races;_apes_and_fossils
43 *PLOT FOSSILS / TYPE=FACES LABEL=Species DEVICE=PS,FACES.PS
44 * SIZE=1164,850 YDIV=0,800,50
45 *VARIABLES: xmin xmax Features fmin fmax
46 * - * ** Radius_to_corner_of_face_OP 0.6 1.0
47 * - * ** Angle_of_OP_to_horizontal 0.0 0.6
48 *D1 6.28** -9.37* Vertical_size_of_face_OU 0.6 1.0
49 * - * ** Eccentricity_of_upper_face 0.5 1.5
50 * - * ** Eccentricity_of_lower_face 0.5 1.5
51 *D2 -4.28* 6.34** Length_of_nose 0.1 0.5
52 *D3 -2.14* 1.66** Vertical_position_of_mouth 0.2 0.8
53 *D1 -9.37* 6.28** Curvature_of_mouth_l/R -4.0 4.0
54 *D1 -9.37* 6.28** Width_of_mouth 0.2 1.0
55 *D1 -9.37* 6.28** Vertical_position_of_eyes 0.0 0.4
56 *D1 6.28** -9.37* Separation_of_eyes 0.3 0.8
57 *D4 -1.73* 2.11** Slant_of_eyes -0.5 0.5
58 *D5 -2.23* 2.06** Eccentricity_of_eyes 0.3 1.0
59 *D6 -1.01* 1.8** Size_of_eyes 0.1 0.2
60 *D7 -0.28* 2.61** Position_of_pupils -0.1 0.1
61 *D8 -0.87* 2.48** Vertical_position_of_eyebrows 0.2 0.4
62 *D2 -4.28* 6.34** Slant_of_eyebrows -0.5 0.5
63 *D3 -2.14* 1.66** Size_of_eyebrows 0.1 0.5
64 *END of plotting specifications
65 *

```

producing the cartoon



8.7.5 Andrews' function plots

PLOT <data> / TYPE=ANDREWS

plots each observation of a multivariate data set as a curve

$$Y(t) = X_1/\sqrt{2} + X_2 * \sin(t) + X_3 * \cos(t) + X_4 * \sin(2t) + X_5 * \cos(2t) + \dots \quad (-\pi < t < \pi)$$

where X_1, X_2, \dots are (scaled) variables selected from <data>. The variables as well as their scalings are determined by a list following the PLOT operation:

```
VARIABLES:  A    B    Term
<name of X1>  A1  B1  1/sqrt(2)
<name of X2>  A2  B2  sin(t)
<name of X3>  A3  B3  cos(t)
etc.
```

END

Each variable X_i will be scaled by using it in the form $(X_i - A_i)/B_i$. If A_i is replaced by a '*', the mean of the variable in the current data will be used. Similarly a '*' as B_i implies the standard deviation to be selected. A fixed scaling with constant A_i and B_i values A, B can be given by a specification FSCALING= A, B and it overrides the values in the VARIABLES list. For example, FSCALING=0, 1 means that all the variables are used without rescaling.

If the PLOT operation (with TYPE=ANDREWS) is activated without a VARIABLES list, a model of such a list will be written in the edit field below

the PLOT line on request.

The PLOT operation for Andrews' curves is supported by the typical extra specifications used in curve plotting (like HEADER, HOME, SIZE, FRAME, XDIV, YDIV, GRID, TICK, XSCALE, YSCALE, XLABEL, YLABEL). However, neither scale transformations nor fills are permitted.

LABEL=<label variable> , <step> , <shift>

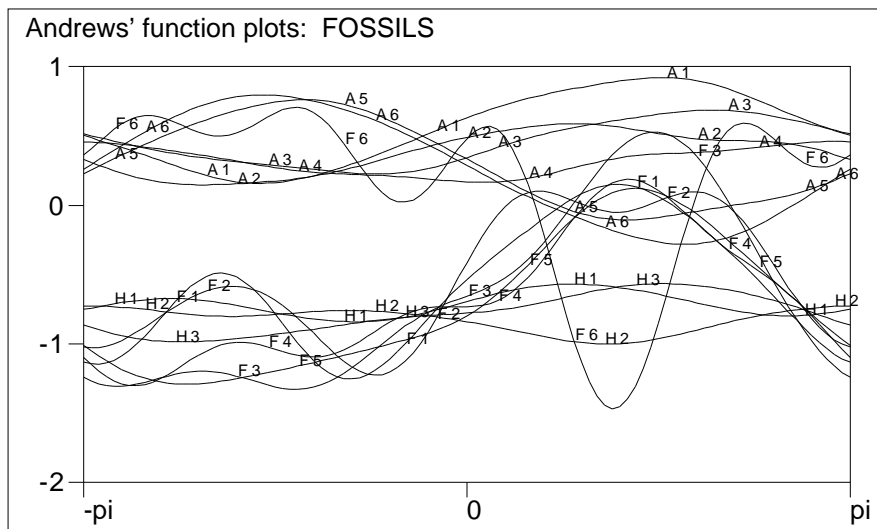
is a specification determining a label variable (name) for identifying each curve by a label. This label is printed on the curve in regular intervals specified in # of plotting steps by <step>. To avoid overlapping of labels, <shift> gives the distance between the labels of consecutive curves in # of plotting steps. Parameters <step> and <shift> are optional and their default values are 30 and 4, respectively.

Data FOSSILS is represented as function plots as follows. The last column 'Label' in the data table is used as a LABEL variable.

```

13 1 SURVO 84C EDITOR Fri Jul 10 10:18:25 1992      D:\P2\PLOT2\ 100 100 0
68 *
69 *PLOT FOSSILS / TYPE=ANDREWS FSCALING=0,1 LABEL=[Swiss(6)],Label
70 * YSCALE=[char_width(5)],-2(1)1 XDIV=100,1014,50
71 * DEVICE=PS SIZE=1164,700 YDIV=75,550,75
72 *
73 *VARIABLES: A      B      Term
74 *D1      0      1      1/sqrt(2)
75 *D2      0      1      sin(t)
76 *D3      0      1      cos(t)
77 *D4      0      1      sin(2*t)
78 *D5      0      1      cos(2*t)
79 *D6      0      1      sin(3*t)
80 *D7      0      1      cos(3*t)
81 *D8      0      1      sin(4*t)
82 *END of plotting specifications
83 *

```



PLOT <data> / TYPE=ANDREWS,POLAR,c

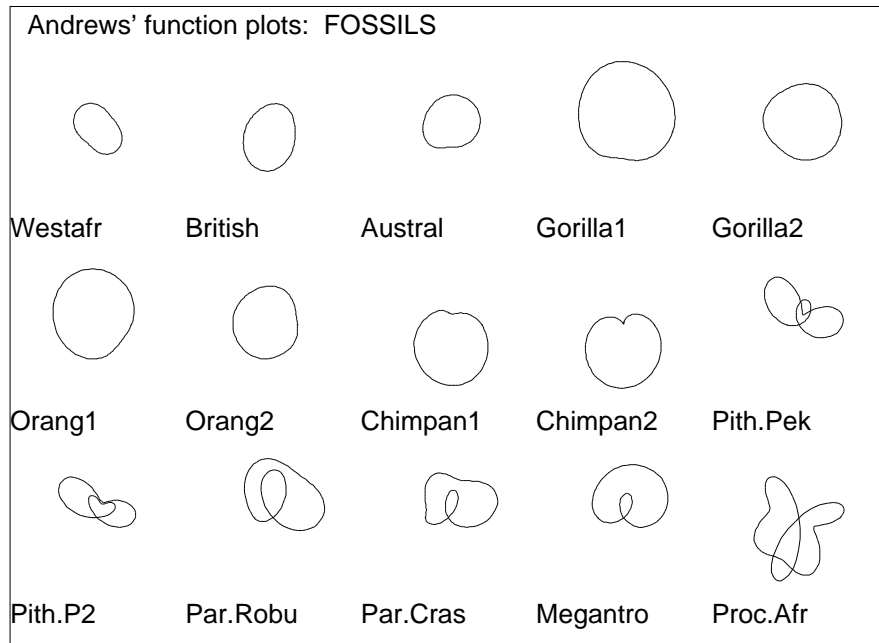
draws the Andrews' curves in polar coordinates and gives the results in a layout similar to Chernoff's faces. Also FSIZE and LABEL specifications are used as in Chernoff's faces. Parameter c is an optional additive constant in $Y(t)$. Default is $c=0$.

The next PLOT scheme gives Andrews' curves in polar coordinates for the FOSSILS data.

```

13 1 SURVO 84C EDITOR Fri Jul 10 13:25:49 1992 D:\P2\PLOT2\ 120 100 0
91 *
92 *PLOT FOSSILS_ / TYPE=ANDREWS,POLAR,0.2 LABEL=Species DEVICE=PS
93 *FRAME=3 FSCALING=0,1.5 SIZE=1164,850 YDIV=0,800,50
94 *
95 *VARIABLES: A B Term
96 *D1 0 1 1/sqrt(2)
97 *D2 0 1 sin(t)
98 *D3 0 1 cos(t)
99 *D4 0 1 sin(2*t)
100 *D5 0 1 cos(2*t)
101 *D6 0 1 sin(3*t)
102 *D7 0 1 cos(3*t)
103 *D8 0 1 sin(4*t)
104 *END of plotting specifications
105 *

```



8.7.6 Profile and star symbol plots

PLOT <data> / TYPE=PROFILES

plots multidimensional data as profile symbol plots. Each observation of m active variables is represented as a broken line connecting points $(1, y_1)$,

$(2, y_2), \dots, (m, y_m)$ where y coordinates are the scaled values $y=x/\max(\text{abs}(x))$. Also the base line and the vertical side lines are plotted for each profile.

The order of activate variables in the profiles is determined by the alphabetic order of their mask symbols and secondarily by their order in the data set.

The general layout of the profile symbol plots in the plotting region is the same as that of Chernoff's faces. The LABEL and FSIZE specifications are available as well.

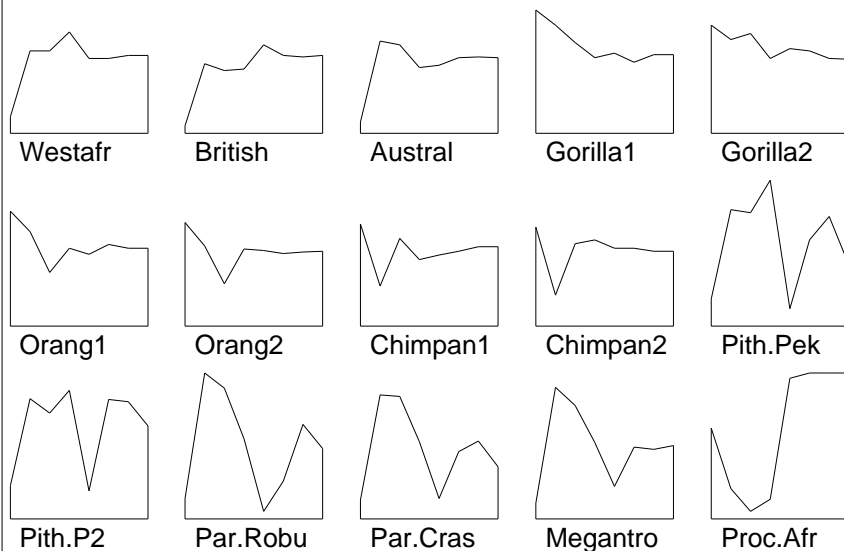
Data FOSSILS is described as profiles by the PLOT scheme:

```

13 1 SURVO 84C EDITOR Fri Jul 10 15:29:05 1992 D:\P2\PLOT2\ 100 100 0
1 *
2 *DATA FOSSILS
3 * Species D1 D2 D3 D4 D5 D6 D7 D8 Label
4 * Westafr -8.09 0.49 0.18 0.75 -0.06 -0.04 0.04 0.03 H1
5 * British -9.37 -0.68 -0.44 -0.37 0.37 0.02 -0.01 0.05 H2
6 * Austral -8.87 1.44 0.36 -0.34 -0.29 -0.02 -0.01 -0.05 H3
7 * Gorilla1 6.28 2.89 0.43 -0.03 0.10 -0.14 0.07 0.08 A1
8 * Gorilla2 4.28 1.52 0.71 -0.06 0.25 0.15 -0.07 -0.10 A2
9 * Orang1 5.11 1.61 -0.72 0.04 -0.17 0.13 0.03 0.05 A3
10 * Orang2 3.60 0.28 -1.05 0.01 -0.03 -0.11 -0.11 -0.08 A4
11 * Chimpan1 3.46 -3.37 0.33 -0.32 -0.19 -0.04 0.09 0.09 A5
12 * Chimpan2 3.05 -4.21 0.17 0.28 0.04 0.02 -0.06 -0.06 A6
13 * Pith.Pek -6.73 3.63 1.14 2.11 -1.90 0.24 1.23 -0.55 F1
14 * Pith.P2 -5.90 3.95 0.89 1.58 -1.56 1.10 1.53 0.58 F2
15 * Par.Robu -7.56 6.34 1.66 0.10 -2.23 -1.01 0.68 -0.23 F3
16 * Par.Cras -7.79 4.33 1.42 0.01 -1.80 -0.25 0.04 -0.87 F4
17 * Megantro -8.23 5.03 1.13 -0.02 -1.41 -0.13 -0.28 -0.13 F5
18 * Proc.Afr 1.86 -4.28 -2.14 -1.73 2.06 1.80 2.61 2.48 F6
19 *
20 *PLOT FOSSILS / TYPE=PROFILES LABEL=Species MASK=-AAAAAAA-
21 *FRAME=3 DEVICE=PS SIZE=1164,850 YDIV=0,800,50
22 *

```

Profile symbol plot: FOSSILS



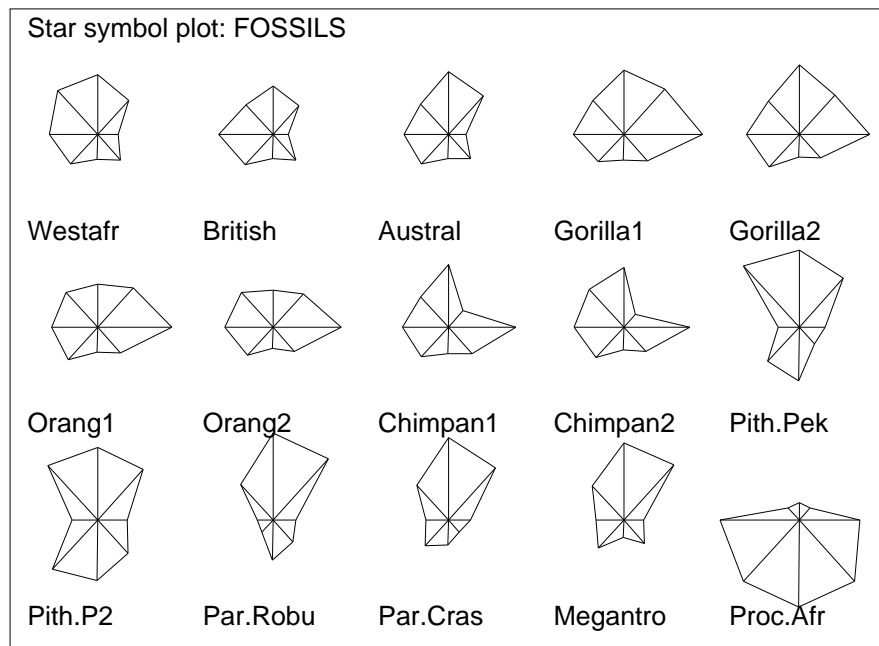
PLOT <data> / TYPE=STARS

plots multidimensional data as star symbol plots. Each observation of m active variables is represented as a set of rays whose directions are equally spaced around the circle and the length of the ray corresponding to the variable x is $(1-c)*(x-xmin)/(xmax-xmin)+c$.

c is a constant with the default value 0.2. Another value for c can be selected by entering a **TYPE** specification in the form **TYPE=STARS, c** . The order of activate variables in the stars is determined by the alphabetic order of their mask symbols and secondarily by their order in the data set.

The general layout of the star symbol plots in the plotting region is the same as that of Chernoff's faces. The **LABEL** and **FSIZE** specifications are available as well.

The previous **PLOT** scheme with **TYPE=STARS** gives



8.8* Device-dependent features

The graphics functions of Survo are to a great extent independent of the device (printer, plotter or screen) used for the final output. With a few exceptions, the rules of plotting given in preceding sections are the same irrespective of the device. Since each alternative has enhancements of its own, supplementary device-dependent information may be added to most specifications.

The device-dependent additions are given in brackets as seen from previous examples. The general structure of a specification is
`<word>=[...][...]... [...], <specification_list> .`

The parameters given in brackets must be defined in the current device driver. This driver is normally selected by

| | | |
|----------------------------------|-----------------------------|---------------------------|
| <code>ps_dev=PS.DEV</code> | for PostScript | <code>DEVICE=PS</code> |
| <code>crt_dev=CRT16.DEV</code> | for screen graphics (GPLOT) | <code>DEVICE=CRT</code> |
| <code>plot_dev=CANON2.DEV</code> | for Canon | <code>DEVICE=Canon</code> |
| <code>plot2_dev=HP.DEV</code> | for HP plotters | <code>DEVICE=HP</code> |

in the `SURVO.APU` text file that defines all Survo system parameters. The default driver can temporarily be altered by giving an `INCLUDE` specification. The default device is set (in `SURVO.APU`) by `plot_mode=<device>` where `<device>` is `PS`, `CRT`, `Canon` or `HP`.

Usually there is no need to make changes in drivers but an advanced user can modify existing ones or write new alternatives. The device drivers are standard text files (not edit fields) and easily accessed by `SHOW`, `LOADP` and `SAVEP` commands.

Since most plots contain text, the device drivers take the responsibility of text printing, too. The primary task of the device driver is to present the control characters and control code sequences of the device in a more legible form and to build higher level control parameters from these words.

For example, in the `PS.DEV` driver, some of the graphic capabilities of the PostScript printer are defined as follows:

```

21 1 SURVO 84C EDITOR Thu Jul 16 18:37:52 1992 D:\P2\LOT4\ 100 100 0
1 *
2 *Selected lines from PS.DEV loaded into the edit field:
3 *
4 *SHOW C:\E\SYS\PS.DEV_
5 * define [S] [2/0]
6 * define [LF] [0/10]
7 * define [line_width(x)] x[S]setlinewidth[S]
8 * define [line_color(x)] x[S]setgray[S]
9 * define [line_type(x)] x[S]plintype[S]
10 * define [PEN] [Swiss(10)]
11 * define [LINE] [line_width(0.24)][line_color(0)][line_type(0)]
12 * ...
13 * /plintype { dup ltype length ge { pop 0 } if[LF]
14 * ltype exch get exec } def[LF]
15 *

```

This sample includes 7 ‘define’ lines naming various line and text attributes. On lines 5 and 6 two characters [2/0] (= hex(20)) and [0/10] (= hex(1a)) are renamed as [S] (space) and [LF] (line feed), respectively. These characters are then used in definitions of other code words as delimiters.

The control word [LINE] gives the default setting for line plotting and [PEN] similarly for text plotting.

The definitions (as such on lines 7-9) often refer to various PostScript functions which are either ready-made (like **setlinewidth** and **setgray**) or specially defined in PS.DEV (like **plintype** on lines 13-14). In fact, the PostScript driver has two main sections. The first one gives various definitions for control words intended for Survo users. The second one consists of new PostScript functions needed in control word definitions.

The user can add more definitions by entering new ‘define’ lines in the driver file. For example, a code word [THICK_SOLID_LINE] could be defined by

```
define [THICK_SOLID_LINE] [line_width(4)][line_type(0)]
```

When adding new features offered by a particular device, the user should consult the technical manual of that device to see the possible control sequences. In implementing Survo graphics for new printers and plotters, certain C functions for graphic primitives must also be rewritten, compiled, and linked with the graphic program modules of Survo.

In text printing (by PRINT), however, no extra programming is needed; it is sufficient to write a new driver as a text file.

In screen graphics (GPLOT), Survo uses the Microsoft C graphics library for the screen control. This library provides drivers for the common screen graphics modes like VGA, EGA, and CGA. A separate Survo driver (text file) is also needed. The default Survo driver for screen graphics is given by the ‘crt_dev’ parameter in SURVO.APU.

Only graphic primitives (like ‘draw a line’ or ‘fill a rectangle’) are taken from the graphics library. The higher control functions of graphics are maintained from Survo itself.

8.9 Plotting on a PostScript printer

Currently, the main output device of Survo is a PostScript printer. For hardcopies of the screen, other alternatives are available.

The most important control words for PostScript plotting are the following:

[`line_width(x)`] sets the line width `x` in Points (1 Point is 1/72 inches or 0.352777... mm).

[`line_type(x)`] sets the line type with alternatives given already in 8.4.4. The possible line types are defined in `PS.DEV` as a vector `ltype`.

[`line_color(x)`] sets the shade of gray `x` on the interval from 0 (black) to 1 (white). It corresponds to PostScript code

```
x setgray
```

as seen in the previous partial listing of `PS.DEV`.

[`color(c,m,y,k)`] sets the line color according to the CMYK color system and it corresponds to PostScript code

```
c m y k setcmykcolor .
```

For example, [`color(0,0,0,0.2)`] is equivalent to [`line_color(0.8)`].

The scaling and rotation of a PostScript graph are selected by optional `SCALING=x,y` and `ROTATION=<angle_in_degrees>` specifications. Observe, however, that pictures are scaled and rotated more easily at the printing stage when the PostScript files are pasted to the report by the `PRINT` operation.

The default line attributes are selected by the control word [`LINE`], defined in the device driver. This setting may be temporarily replaced by a `LINETYPE` specification of the form

```
LINETYPE=[line_width(1)][line_type(0)][line_color(0)] .
```

Similarly, the default text font used in the graph is selected by the control word [`PEN`] in the device driver. It may be overridden in any plotting scheme by a `PEN` specification like

```
PEN=[Times(12)][BOLD] .
```

The device-dependent additions in any specification override default settings as well as the `LINETYPE` and `PEN` specifications.

The default font in Survo PostScript plots is 10 Point Helvetica [`Swiss(10)`]. The control words used for text fonts and styling are explained in the section "Printing of reports".

Picture files

Any PostScript picture generated by the PLOT operation is saved in a selected file by entering a `DEVICE=PS,<name_of_file>` specification. Then, all the characters normally sent to the printer are saved in a PostScript file. A picture saved in a file can then be included in a report produced by the PRINT operation by using a control line of the form

```
- picture <name_of_file>,<x>,<y>
```

in the print list. The lower left corner of the picture (HOME) will be repositioned to `<x>,<y>` (unit is 0.1 mm) on the current page. However, if a '*' is entered instead of a numeric value, the location of the picture is automatically adjusted so that '*' in place of `<x>` refers to the left marginal position of the text and in place of `<y>` it refers to the current line. More details of the control line `- picture` will be given in the section "Printing of reports". For example, the size of the graph can be altered by scaling coefficients as seen already in the next example.

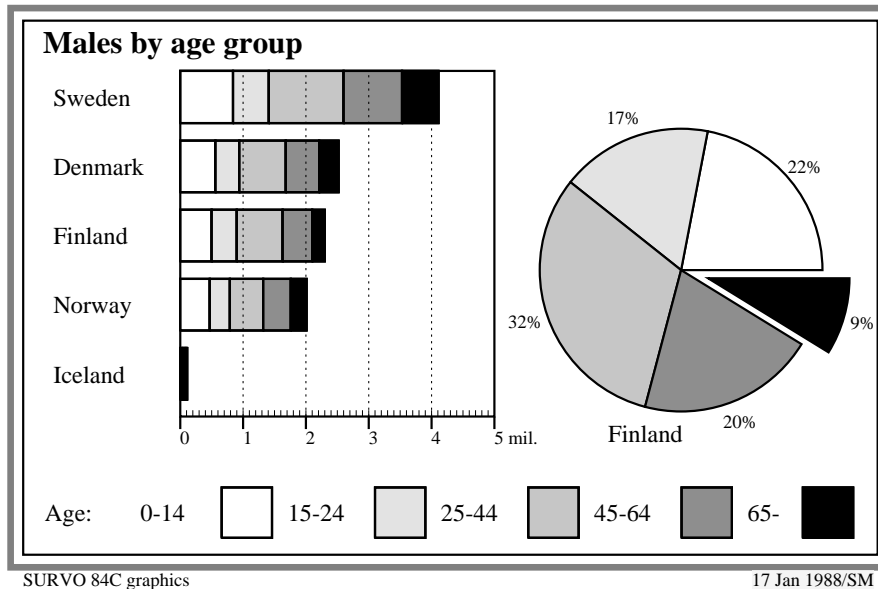
The PRINT operation with picture files allows even partial or complete overlays of graphs. In the next example, a bar chart and a pie chart are plotted and saved in PLOT files `BAR2.PS` and `PIE2.PS`.

```
13 1 SURVO 84C EDITOR Fri Jul 17 14:33:30 1992 D:\P2\PLOT4\ 100 100 0
1 *
2 * *GLOBAL* a=1400-2*d b=850 c=500 d=20 dimensions
3 *.....
4 *DATA FINLAND,S,S,M
5 *DATA COUNTRIES,A,B,M
6 M COUNTRY _0-14 15-24 25-44 45-64 65-
7 A Sweden 841 571 1188 930 585
8 * Denmark 564 385 731 537 308
9 S Finland 506 399 727 468 202
10 * Norway 474 316 534 442 251
11 B Iceland 32 22 29 20 10
12 *
13 *HOME=280,150 SIZE=a,b XDIV=250,c,a-c-250 YDIV=220,b-300,80
14 *HEADER=[MAIN][TIMES(14)],Males_by_age_group LEGEND=Age:
15 *GRID=[line_type(2)][line_width(0.48)],X TICK=[line_width(0.24)],100
16 *XSCALE=[INDEX],0,1000:1,2000:2,3000:3,4000:4,5000:5_mil.
17 *PLOT COUNTRIES / DEVICE=PS,BAR2.PS PEN=[TIMES(11)]
18 *LINETYPE=[line_type(0)][line_width(1)][line_color(0)]
19 *TEXTS=A,B A=[INDEX],SURVO_84C_graphics,0,-50 SHADING=0,1,2,4,9
20 * B=[INDEX],17_Jan_1988/SM,a-200,-50
21 *FRAMES=I I=[line_width(3)][line_color(0.5)],-d,-d,a+2*d,b+2*d,0
22 *.....
23 *PLOT FINLAND / DEVICE=PS,PIE2.PS PEN=[TIMES(11)]
24 *TYPE=%PIE HOME=280+c+260,150 SIZE=c,b
25 *XDIV=0,1,0 FRAME=0 LEGEND=- HEADER=
26 *LINETYPE=[line_type(0)][line_width(1)][line_color(0)]
27 *SHADING=0,1,2,4,9P VALUES=[INDEX],##%,11.5
28 *
```

These files are then included in this text as one picture by a PRINT operation so that the original graphs overlap:

```
12 1 SURVO 84C EDITOR Fri Jul 17 14:35:36 1992 D:\P2\PLOT4\ 100 100 0
28 *.....
29 *PRINT 30,32
30 % 750 (75 mm space for the picture)
31 - picture BAR2.PS,*+20,*0.831,0.831
32 - picture PIE2.PS,*+660,*0.831,0.831
33 *
```

This example is taken from a previous report on Survo PostScript printing (Mustonen 1988). We alter the picture size a little by rescaling the graphs from the original size (width 140 mm) to current size (116.4 mm) by scaling coefficients $1164/1400=0.831$ and the output of the PRINT operation will be:



Encapsulated PostScript files

The PostScript files generated by the PLOT operations are designed for efficient use in Survo environment. They do not conform the requirements of the encapsulated PostScript standard. However, an encapsulated PostScript (EPS) file can be created as a by-product in a PLOT operation or a Survo PostScript file (created by PLOT) can be converted to an EPS file.

EPS <PS_file>, <EPS_file>

converts a Survo PS file <PS_file> created by a PLOT operation with **DEVICE=PS**, <PS_file> to an EPS file <EPS_file>. For example,

EPS BAR2.PS, BAR2.EPS

makes that conversion for the first graph in the preceding example.

Another alternative is to add in the PLOT scheme a specification of the form **EPSFILE=<EPS_file>**. In that case, the PLOT scheme must also contain **DEVICE=PS, <PS_file>** since the conversion is made from <PS_file> to <EPS_file>.

Encapsulated PostScript files (also made by other graphics programs) are taken into Survo documents by using in PRINT operation control lines of the form

- **epsfile <EPS_file>, <x>, <y>**, etc.



EPS?

8.10 Plotting on the screen

The alternatives for screen graphics in Survo are the VGA, EGA, and CGA modes. The Survo graphs are plotted on the graphic screen either by using GPLOT instead of PLOT or by giving the specification DEVICE=CRT. The graphics mode is selected by the specification MODE=<VGA or EGA or CGA>. The default mode is given in SURVO.APU by the word 'videomode'.

When a GPLOT operation has been activated, the screen will be cleared and the plot will be displayed. The graphic display stays on the screen as long as **ENTER** is pressed. Then the graph disappears and the normal display of the edit field is restored.

The graph generated by the GPLOT operation is saved to a selected file by giving the specification

OUTFILE=<name_of_file>.

The default extension of the filename is SPX. A plot in a SPX file can be displayed later by the command

GPLOT FILE <name_of_file>.

In EGA mode, two pictures can be displayed alternately by

GPLOT FILE <file1>, <file2>.

To change the picture, after GPLOT FILE has been activated, press keys **1** and **2**.

A ready-made picture saved by the OUTFILE specification can be used as a background for another by plotting the latter with an

INFILE=<SPX file of the first picture>

specification. This overlay of pictures can be saved again by a proper OUTFILE specification and used as a background for new plots.

The screen graphics in Survo is controlled by the device driver given as 'crt_dev' in the system file SURVO.APU. The following statements are valid for the CRT16.DEV driver.

The most important control words, to be used in specifications, are:

[line_width(x)] sets the line width in pixels. Possible values of x are integers 1,2,3,...

[`line_type(x)`] sets the line type with following alternatives for `x`:

- 1 solid
- 2 long dash
- 3 dotted
- 4 dash dotted
- 5 medium dashed
- 6 dash with two dots
- 7 short dash


[`line_color(x)`] sets the line color with following alternatives for `x`:

- 0 black
- 1 white
- 2 red
- 3 green
- 4 blue
- 5 yellow
- 6 cyan
- 7 magenta
- 8-15 various additional colors


[`character_color(x)`] sets the text color similarly.

For both line and character colors also the control words [BLACK], [WHITE], [RED], [GREEN], [BLUE], [YELLOW], [CYAN] and [MAGENTA] are available.
Example: `HEADER=[RED],Figure_1`

[`background(x)`] sets the background color. Also notations [/BLACK], [/WHITE], etc. are possible for specifying the background color.

The basic color combination on the graphic screen is selected by the `COLORS` specification. The default is `COLORS=[BLACK] [/WHITE]`. 

Color palette

The colors used in screen graphics are selected by a `PALETTE` specification. 
`PALETTE=<palette_A>, <palette_B>, ...`

gives a list of alternative palette files located in the `SYS` subdirectory of Survo. The graph is shown at first according to colors of `<palette_A>`. By pressing `[B]`, `[C]`, etc., the colors in the graph are immediately changed according to the corresponding palette.

Palette files are generated and edited in an interactive mode by the command

`GLOT /PALETTE <old_file>, <new_file> .`

The default palette is defined by the system parameter `crt_palette` in the `SURVO.APU` file. When Survo is run under Windows, `PALETTE=NUL` (or `crt_palette=NUL`) should be selected.

8.11 Graphics examples

This section tells about how the PLOT operation is used for various applications. In each of the following exhibits, the graphs are coupled with extracts from the edit field completely describing the PLOT operations activated and the necessary attributes.

Many examples are related to statistical graphics. Besides demonstrating various enhancements of Survo graphics, we also try to show how pictures can be used as an efficient tool in various research and teaching situations. Of course, an immense number of variations could be derived from the basic graphs or from combinations of standard alternatives. Therefore, our list of examples cannot be complete. The reader can easily generate more examples by modifying existing ones and save solutions for subsequent application.

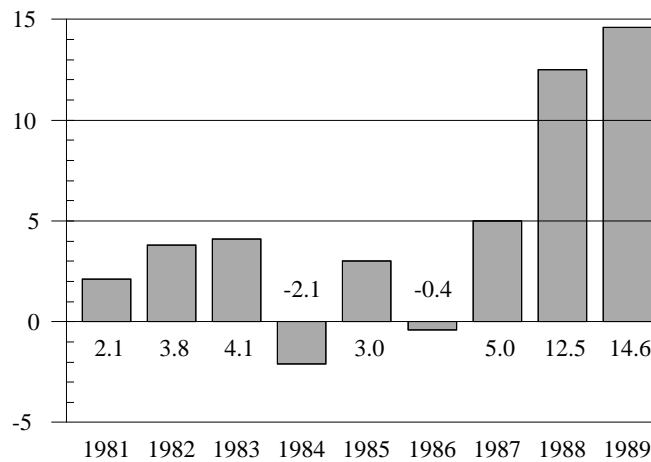
Also some purely mathematical curves and families of curves are presented. Among them are some more complicated ‘artistic’ creations displaying the power of our graphical language.

```

13 1 SURVO 84C EDITOR Thu Jul 23 12:56:46 1992      D:\P2\PLOT4\ 150 80 0
14 *
15 *PLOT FINLAND / TYPE=VBAR
16 *SIZE=1164,800 GRID=5 TICK=1 SHADING=3 LEGEND=-
17 *VALUES=[move(-20,0)],###.#,-2,1 (moved 2 mm to the left)
18 *PEN=[Times(9)] [move(0,0)] (default setting: no move)
19 *TEXT=A,B A=SURVO_84C_graphics,10,20
20 *      B=23.7.1992/SM,950,20
21 *DEVICE=PS,CAPITAL.PS
22 *.....

```

Private gross fixed capital formation (changes in %)



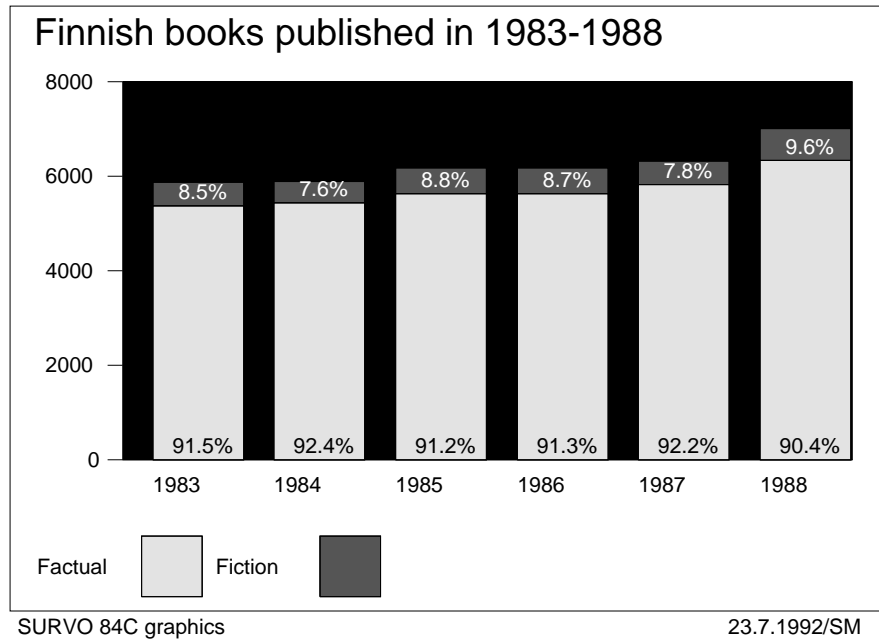
SURVO 84C graphics

23.7.1992/SM

```

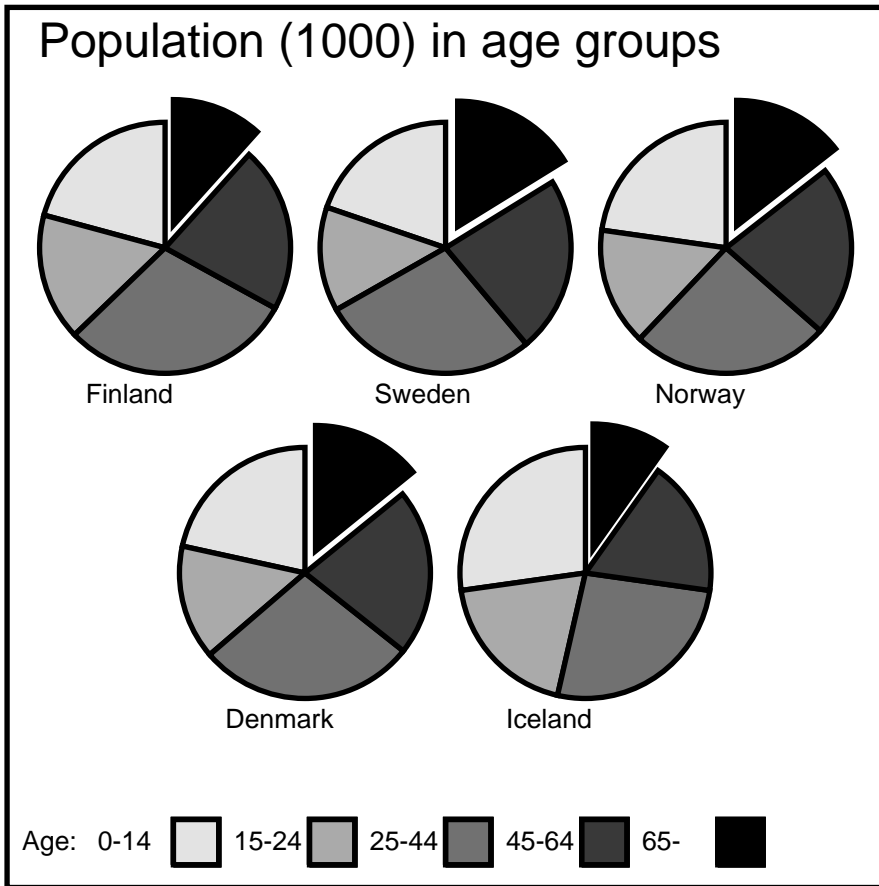
11 1 SURVO 84C EDITOR Thu Jul 23 13:08:56 1992 D:\P2\PLOT4\ 150 80 0
22 *
23 *HEADER=[Swiss(14)],Finnish_books_published_in_1983-1988
24 *
25 *DATA BOOKS,X,Y,O
26 O Year Factual Fiction
27 X 1983 5384 497
28 * 1984 5447 445
29 * 1985 5634 547
30 * 1986 5644 539
31 * 1987 5825 495
32 Y 1988 6345 671
33 *
34 *TYPE=VBAR SHADING=1,6 PEN=[Swiss(8)][autom_color(0)]
35 *VALUES=[autom_color(0.4)],###.##,0.5
36 *SIZE=1164,800 XDIV=150,964,50 YDIV=200,500,100
37 *PLOT BOOKS
38 *TEXTS=A,B A=SURVO_84C_graphics,10,-30
39 * B=23.7.1992/SM,950,-30
40 *FRAMES=F1 F1=150,200,964,500,9
41 *DEVICE=PS,BOOKS.PS
42 *

```



```

12 1 SURVO 84C EDITOR Mon Jul 20 11:04:25 1992 D:\P2\PLOT4\ 150 80 0
43 *
44 *HEADER=[Swiss(18)],Population_(1000)_in_age_groups
45 *DATA NORDIC,A,B,M
46 M Country 0-14 15-24 25-44 45-64 65-
47 A Finland 990 780 1420 1015 555
48 * Sweden 1641 1117 2316 1883 1345
49 * Norway 925 617 1041 895 588
50 * Denmark 1102 753 1433 1099 724
51 B Iceland 61 43 59 39 22
52 *
53 *SIZE=1164,1164 XDIV=0,1114,50 YDIV=200,854,100
54 *SHADING=1,3,5,7,9P DEVICE=PS,POPUL.PS
55 *PLOT NORDIC_ / TYPE=%PIE ANGLE=90 (start angle of first sector)
56 *LINETYPE=[line_type(0)][line_width(2)]
57 *TEXT=L,A,B A=[Swiss(7)],SURVO_84C_graphics,0,-30
58 * B=[Swiss(7)],18.7.1992/SM,950,-30
59 * L=Age: ,20,50
60 *LEGEND=220,30,5 LEGEND_BOX=180,0,60,60 LEGEND_TEXT=-100,20
61 *
    
```



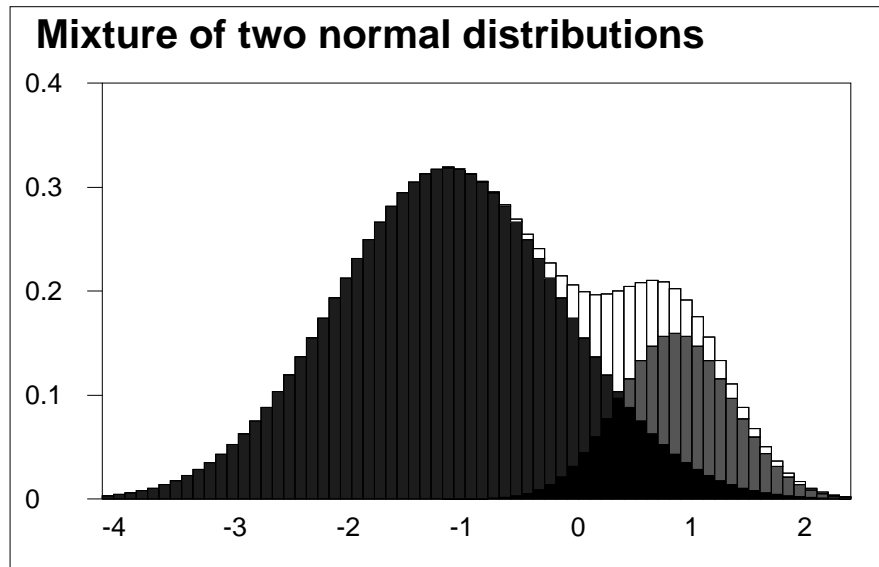
SURVO 84C graphics

18.7.1992/SM

```

13 1 SURVO 84C EDITOR Mon Jul 20 11:14:55 1992      D:\P2\LOT4\ 100 80 0
1 *
2 *VAR f,g,min1,dev1,dev2,min2 TO MIXNORM
3 *   f=0.8*N.f(-1,1,x)
4 *   g=0.2*N.f(1,0.25,x)      This VAR operation
5 *   min1=min(f,g)            generated the data values.
6 *   dev1=max(f-g,0)         'x' column had originally
7 *   dev2=max(g-f,0)         values -4.0(0.1)2.5
8 *   min2=min(f,g)
9 *
10 *PLOT MIXNORM
11 *HEADER=[Swiss(15)] [BOLD],Mixture_of_two_normal_distributions
12 *LEGEND=- SHADING=9,6,3,0 DEVICE=PS,MIXNORM.PS
13 *TYPE=VBAR GAP=0 YSCALE=0(0.1)0.4
14 *SIZE=1164,750 XDIV=124,990,50 YDIV=100,550,100
15 *
16 *DATA MIXNORM,A,B,N,M
17 N  x    f      g      min1    dev1    dev2    min2
18 M 12.1 ----- ----- 1.1234  1.1234  1.1234  1.1234
19 A -4_  0.0035  0.0000  0.0000  0.0035  0.0000  0.0000
20 *    -  0.0048  0.0000  0.0000  0.0048  0.0000  0.0000
21 *    -  0.0063  0.0000  0.0000  0.0063  0.0000  0.0000
22 *    -  0.0083  0.0000  0.0000  0.0083  0.0000  0.0000
23 *    -  0.0109  0.0000  0.0000  0.0109  0.0000  0.0000
24 *    -  0.0140  0.0000  0.0000  0.0140  0.0000  0.0000
25 *    -  0.0179  0.0000  0.0000  0.0179  0.0000  0.0000
26 *    -  0.0227  0.0000  0.0000  0.0227  0.0000  0.0000
27 *    -  0.0284  0.0000  0.0000  0.0284  0.0000  0.0000
28 *    -  0.0352  0.0000  0.0000  0.0352  0.0000  0.0000
29 *   -3_  0.0432  0.0000  0.0000  0.0432  0.0000  0.0000

```



```

15 1 SURVO 84C EDITOR Mon Jul 20 11:17:20 1992 D:\P2\LOT4\ 120 80 0
1 *
2 *DATA SURVO:(X,Y)
3 * 2,3 3,2 4,2 5,3 5,4 2,5 2,6 3,7 4,7 5,6
4 * 6,- 6,7 6,3 7,2 8,2 9,3 9,7
5 * 10,- 10,2 10,7 12,7 13,6 13,5 12,4 10,4 12,- 12,4 13,2
6 * 14,- 14,7 15,2 16,2 17,7
7 * 18,- 18,3 18,6 19,7 20,7 21,6 21,3 20,2 19,2 18,3 END
8 *
9 *PLOT SURVO,X,Y
10 *YSCALE=0(5)10 XSCALE=0(5)25 XLABEL= YLABEL=
11 *SIZE=1164,550 DEVICE=PS,SURV01.PS *GLOBAL*
12 *.....
13 *PLOT SURVO,X,Y
14 *LINE=[line_width(4)],1 DEVICE=PS,SURV02.PS
15 *

```

Diagram of SURVO

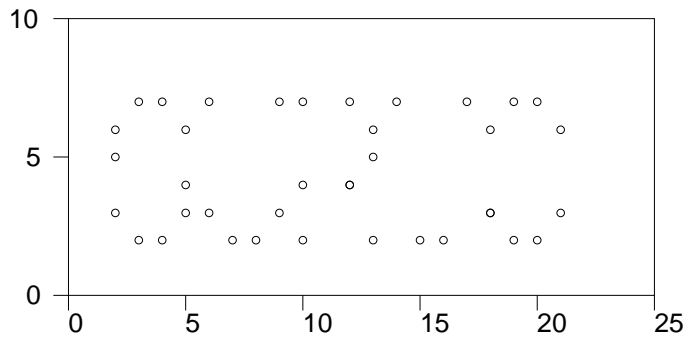
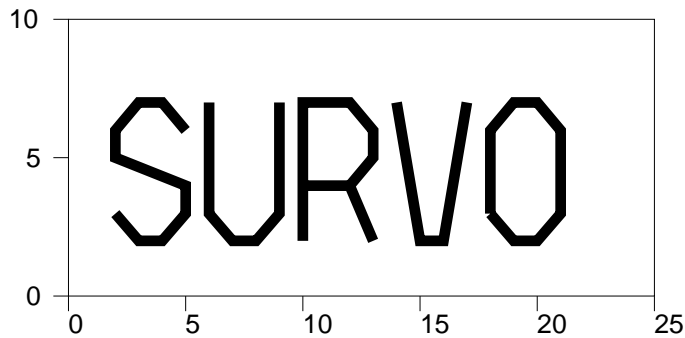


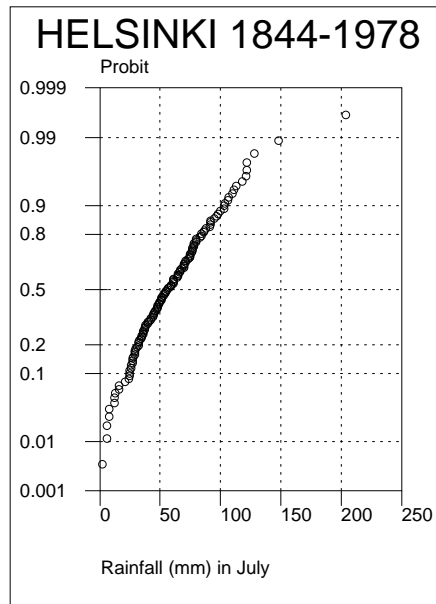
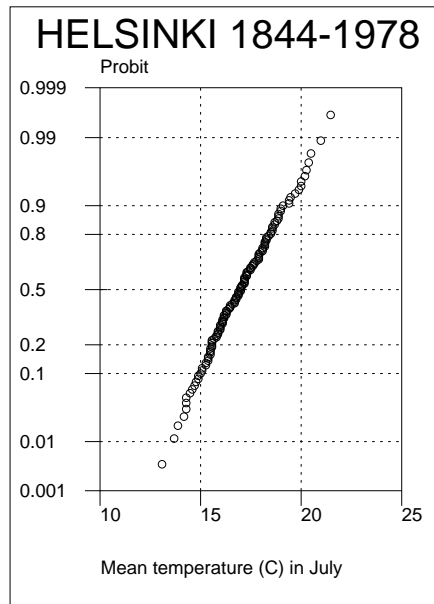
Diagram of SURVO



```

25 1 SURVO 84C EDITOR Mon Jul 20 12:20:37 1992      D:\P2\LOT4\ 120 80 0
1 *
2 * *GLOBAL*
3 * SIZE=570,800 XDIV=120,400,50 GRID=[line_type(2)],XY TICK=0,1
4 * YSCALE=*probit(y),0.001,0.01,0.1,0.2,0.5,0.8,0.9,0.99,0.999
5 * HEADER=[Swiss(15)],HELSINKI_1844-1978 PEN=[Swiss(7)]
6 * YLABEL=Probit XLABEL=
7 * DEVICE=PS
8 *
9 *FILE SORT HELSINKI BY Temp TO HELSORT
10 *PLOT HELSORT,Temp,PROBIT / DEVICE=PS,HEL1.PS
11 *TEXTS=T1 T1=Mean_temperature_(C)_in_July,120,50
12 *
13 *FILE SORT HELSINKI BY Rain TO HELSORT
14 *PLOT HELSORT,Rain,PROBIT / DEVICE=PS,HEL2.PS
15 *TEXTS=T1 T1=Rainfall_(mm)_in_July,120,50
16 *XSCALE=0(50)250
17 *

```

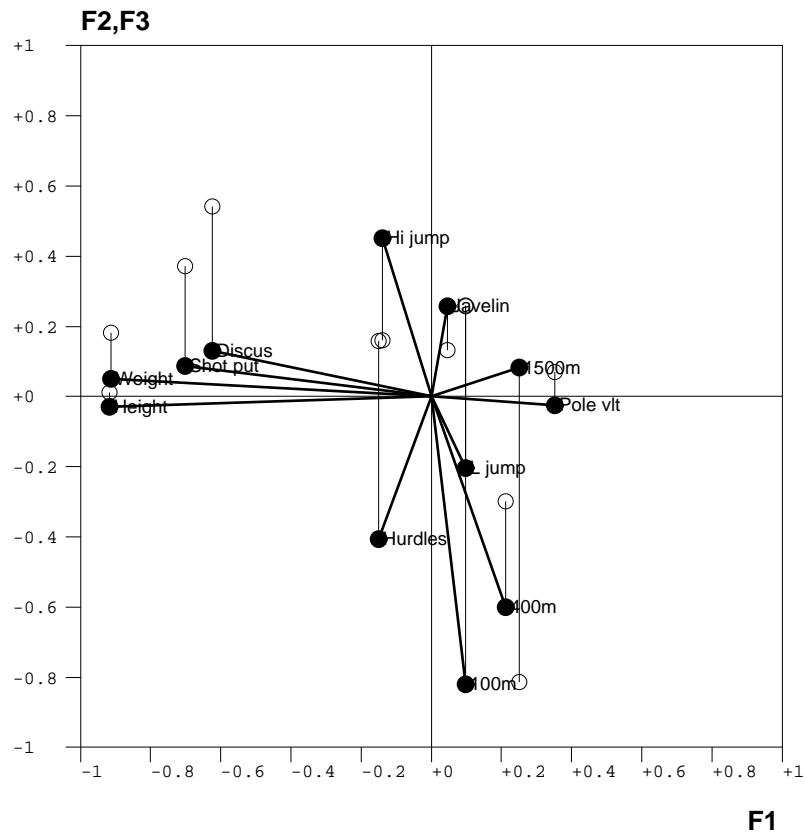



```

19 1 SURVO 84C EDITOR Mon Jul 20 12:27:32 1992      D:\P2\PLOT4\ 120 80 0
1 *
2 *Rotated factor matrix labelled as a data set FACTORS:
3 *DATA FACTORS
4 *Variable      F1      F2      F3
5 * 100m         0.098 -0.819  0.260
6 * 1_jump      0.099 -0.204  0.258
7 * Shot_put   -0.701  0.088  0.372
8 * Hi_jump    -0.139  0.452  0.160
9 * 400m       0.213 -0.600 -0.299
10 * Hurdles   -0.149 -0.406  0.159
11 * Discus    -0.625  0.131  0.541
12 * Pole_vlt  0.353 -0.024  0.070
13 * Javelin   0.047  0.257  0.132
14 * 1500m     0.250  0.084 -0.813
15 * Height   -0.918 -0.029  0.011
16 * Weight   -0.913  0.050  0.182
17 *
18 *HEADER=Rotated_factor_matrix  PEN=[Swiss(10)]
19 *PLOT FACTORS,F1,F2 / XLABEL=[Swiss(10)],F1 YLABEL=[Swiss(10)],F2,F3
20 *SCALE=[Courier(7)],-1(+0.2)1 GRID=1,1 POINT=[Swiss(7)],Variable
21 *LINE=6 LINE2=[line_width(1)],0,0
22 *LINE3=F1,F2 POINT3=0,10 LINE4=[line_width(0.24)],F1,F3 POINT4=3,10
23 *DEVICE=PS,FACTORS.PS SIZE=1160,1160 XDIV=1,8,1 YDIV=1,8,1

```

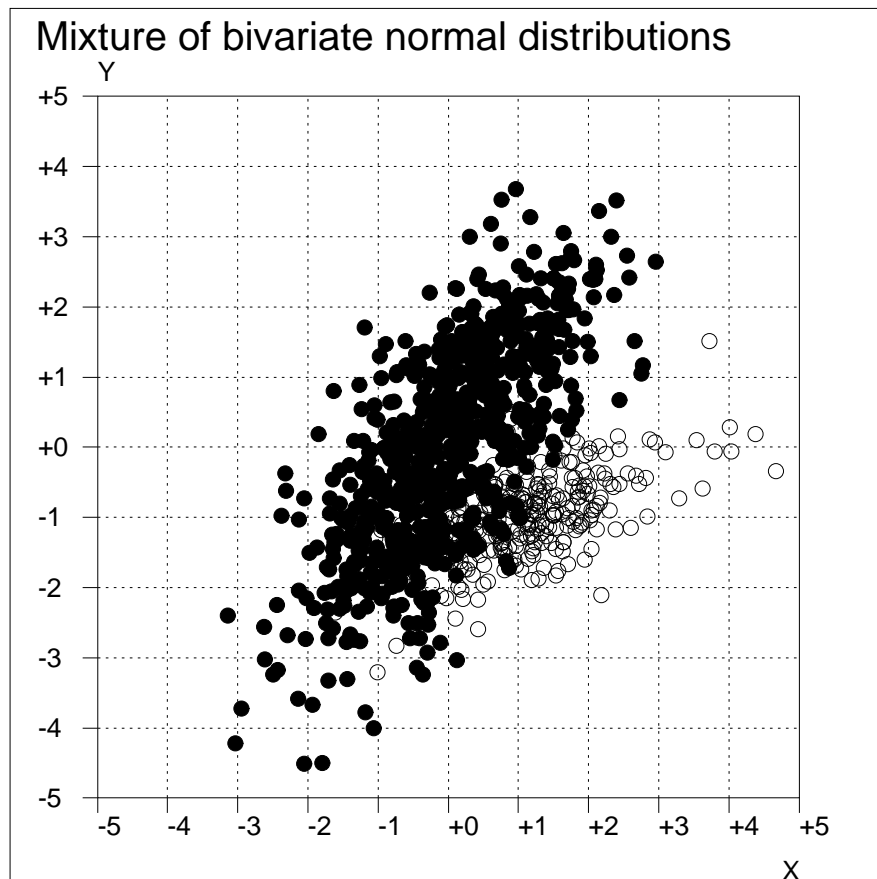
Rotated factor matrix



```

17 1 SURVO 84C EDITOR Mon Jul 20 12:30:44 1992 D:\P2\LOT4\ 120 80 0
28 *
29 *FILE CREATE MIXTURE,9,3,64,7,1000
30 *FIELDS:
31 *1 N 1 Group
32 *2 N 4 X
33 *3 N 4 Y
34 *END
35 *
36 *VAR Group,X,Y TO MIXTURE
37 * Group=3*int(0.3+rnd(1)) possible values 0 and 3 (0 with prob. 0.7)
38 * U=probit(rnd(1)) V=probit(rnd(1))
39 * X=if(Group=0)then(U)else(U+1)
40 * Y=if(Group=0)then(U+V)else(0.3*U+0.5*V-1)
41 *
42 *HEADER=[Swiss(15)],Mixture_of_bivariate_normal_distributions
43 *PLOT MIXTURE,X,Y
44 *POINT=[line_type(0)],0,10,Group,0
45 *SCALE=-5(+1)5 GRID=[line_type(2)],XY
46 *SIZE=1160,1160 XDIV=1,8,1 YDIV=1,8,1 DEVICE=PS,MIXTURE.PS

```



```

35 1 SURVO 84C EDITOR Mon Jul 20 12:43:25 1992 D:\P2\PLOT4\ 120 80 0
1 *
2 *Years 1980 and 1981 are left out in order to show the behaviour
3 *of PLOT when missing values are encountered.
4 *DATA ALCO
5 * Year Spirits Wines Label
6 * 1967 5492 1801 1967
7 * 1968 5464 1950 -
8 * 1969 5823 1956 1969
9 * 1970 6464 2179 1970
10 * 1971 7764 2337 -
11 * 1972 8084 2363 1972
12 * 1973 9433 2930 1973
13 * 1974 11384 3836 1974
14 * 1975 11131 3361 1975
15 * 1976 11880 3146 -
16 * 1977 12210 3044 1977
17 * 1978 11390 3210 -
18 * 1979 11051 3164 1979
19 * 1980 - - -
20 * 1981 - - -
21 * 1982 13599 3783 1982
22 * 1983 13713 3624 1983
23 * 1984 13977 3350 1984
24 * 1985 13635 3177 1985
25 * 1986 14414 3458 1986
26 * 1987 14689 3532 1987
27 * 1988 14259 3717 1988
28 * 1989 14342 3893 1989
29 * 1990 13925 4100 1990
30 * 1991 12355 4620 1991
31 *
32 *HEADER=Consumption_of_spirits_and_wines_in_Finland
33 *XSCALE=[Swiss(7)],1(2)25 YLABEL=1000_1_100%_(log)
34 *YSCALE=*log(y),1000(1000)4000,6000(2000)10000,12000,15000
35 *PLOT ALCO,TIME(Year),Spirits,Wines
36 * SpiritsLINE=[line_width(2)][line_type(0)],1,1,Spirits
37 * WinesLINE=[line_width(2)][line_type(1)],1,1,Wines
38 *SIZE=1164,800 DEVICE=PS,ALKO1.PS
39 *GRID=[line_type(2)][line_width(0.24)],XY TICK=1,1000

```

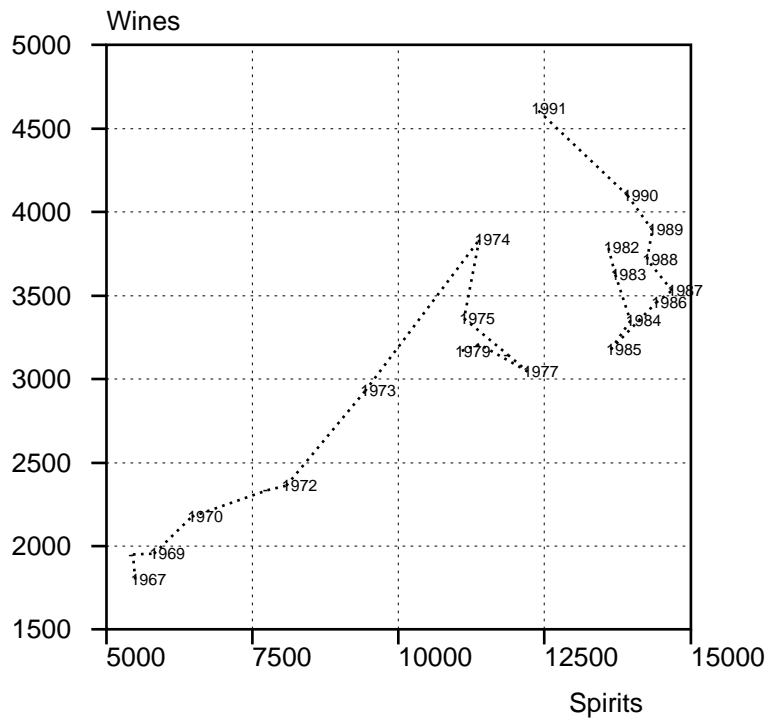


```

24 1 SURVO 84C EDITOR Mon Jul 20 12:45:43 1992 D:\P2\PLOT4\ 120 80 0
42 *
43 *HEADER=Consumption_of_spirits_and_wines_in_Finland
44 *PLOT ALCO,Spirits,Wines
45 *LINE=[line_width(1)][line_type(2)],1 POINT=[Swiss(6)],Label
46 *GRID=[line_width(0.24)][line_type(2)],XY
47 *SIZE=1160,1160 LINETYPE=[line_width(1)] DEVICE=PS,ALKO2.PS
48 *XSCALE=5000(2500)15000 YSCALE=1500(500)5000
49 *

```

Consumption of spirits and wines in Finland

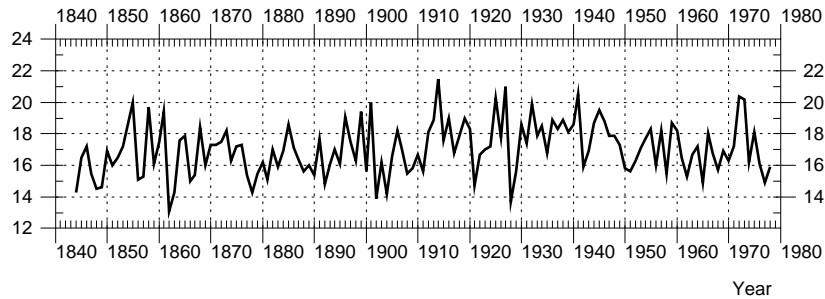


```

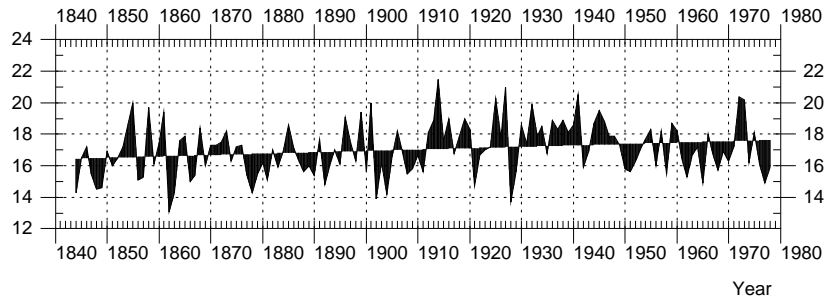
30 1 SURVO 84C EDITOR Mon Jul 20 12:54:54 1992      D:\P2\PLOT4\ 120 80 0
1 *
2 *PLOT HELSINKI,TIME(Year),Temp
3 *HEADER=[Times(11)],Mean temperature in July, Helsinki_1844-1978 YLABEL=
4 *LINE=[line_width(1)],1 PEN=[Swiss(7)] *GLOBAL*
5 *SIZE=1160,450 XDIV=100,960,100 YDIV=100,250,100
6 *XSCALE=-3:1840,7(10)127,137:1980 XSCALE2=XSCALE TICK=1,1
7 *YSCALE=12(2)24 YSCALE2=14(2)22 GRID=[line_type(2)],XY TICK2=1,1
8 *
9 *LINREG HELSINKI / VARS=Temp(Y),Year(X),Trend(P)
10 *PLOT HELSINKI,TIME(Year),Temp
11 *FILL=1,1,135,Trend LINE=[line_width(0.24)],1
12 *
13 *PLOT HELSINKI,TIME(Year),Temp_ / FILL=-[line_color(0.5)],1
14 *FILL=[line_color(0)],1,1,135,Trend LINE=[line_width(0.24)],1
15 *

```

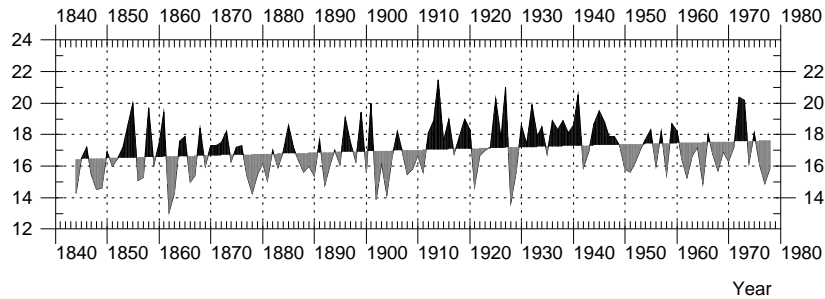
Mean temperature in July, Helsinki 1844-1978



Mean temperature in July, Helsinki 1844-1978



Mean temperature in July, Helsinki 1844-1978

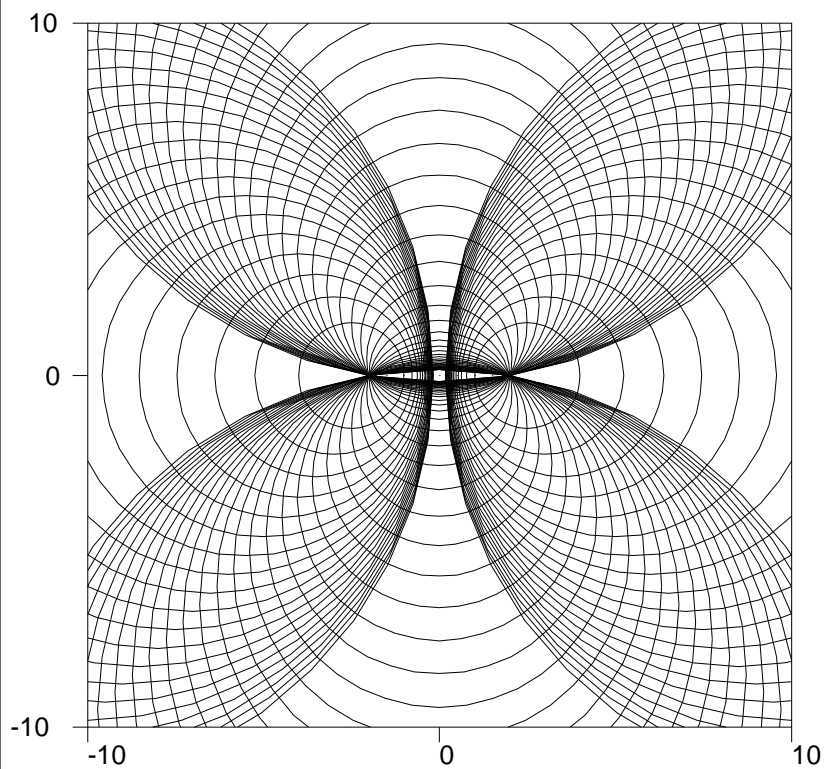


```

26 1 SURVO 84C EDITOR Tue Jul 21 09:31:07 1992 D:\P2\PLOT4\ 100 100 0
1 *
2 *HEADER=Steiner_circles
3 *XDIV=1,8,1 YDIV=1,8,1 SCALE=-10,0,10
4 *T=0,2*pi,pi/20 pi=3.141592653589793
5 *R=-12,12,0.5 ind=0,1,1 C=2
6 *A=R*R+(1-2*ind)*C*C B=if(A<0)then(0)else(sqrt(A))
7 *PLOT X(T)=B*cos(T)+ind*R,
8 *      Y(T)=B*sin(T)+(1-ind)*R
9 *SIZE=1164,1164 DEVICE=PS,STEINER.PS
10 *

```

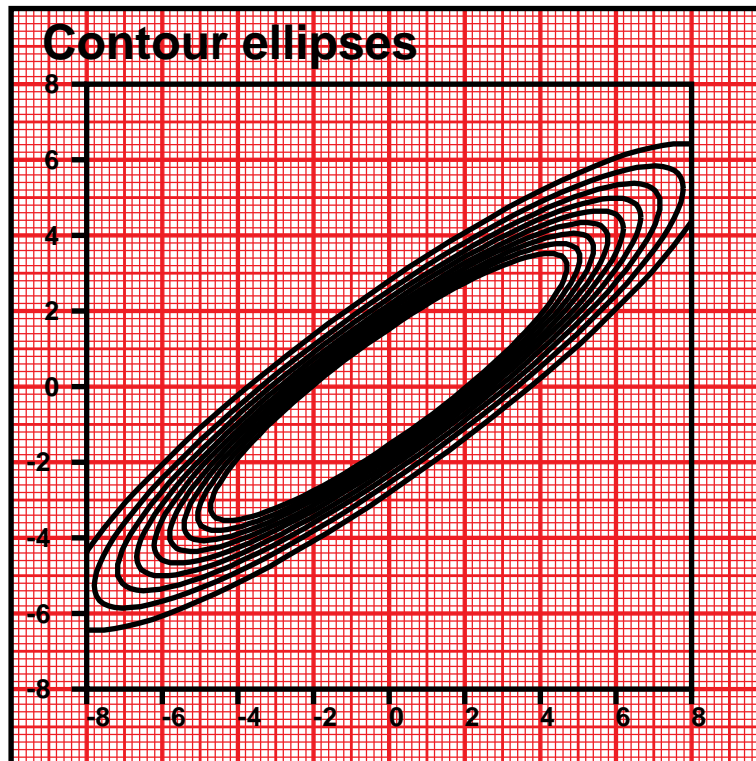
Steiner circles



```

12 1 SURVO 84C EDITOR Tue Jul 21 09:34:32 1992 D:\P2\PLOT4\ 100 100 0
11 *
12 *XDIV=0,1,0 YDIV=0,1,0 SIZE=1000,1000 HEADER= FRAME=0 SCALE=0,1000
13 *t10=0,1000,100 t5=50,950,100 t=0,1000,10 u=0,1000,1000
14 *PLOT Y(u)=t10 / LINETYPE=[line_width(1.2)] DEVICE=PS,MMX10.PS
15 *PLOT Y(u)=t5 / LINETYPE=[line_width(0.72)] DEVICE=PS,MMX5.PS
16 *PLOT Y(u)=t / LINETYPE=[line_width(0.24)] DEVICE=PS,MMX1.PS
17 *PLOT X(u)=t10,Y(u)=u / LINETYPE=[line_width(1.2)] DEVICE=PS,MMY10.PS
18 *PLOT X(u)=t5,Y(u)=u / LINETYPE=[line_width(0.72)] DEVICE=PS,MMY5.PS
19 *PLOT X(u)=t,Y(u)=u / LINETYPE=[line_width(0.24)] DEVICE=PS,MMY1.PS
20 *
21 *PLOT X(t)=s1*sqrt(-2*log(1-eps))*cos(t),
22 * Y(t)=s2*sqrt(-2*log(1-eps))*sin(t+atn(rho/sqrt(1-rho*rho)))
23 *HEADER=[Swissb(18)],Contour_ellipses
24 *s1=4 s2=3 rho=0.9 t=0,2*pi pi=3.14159265 eps=0.5,0.9,0.05
25 *XDIV=1,8,1 YDIV=1,8,1 SIZE=1000,1000
26 *SCALE=[Swissb(10)],-8(2)8
27 *LINETYPE=[line_width(2)] DEVICE=PS,CONTOURS.PS
28 *TEXTS=T T=[BOLD],Plotting_a_graph_paper_and_contour_ellipses,10,-50
29 *
30 *PRINT 31,38
31 % 1050
32 - picture MMX10.PS,*,+82,*
33 - picture MMX5.PS,*,+82,*
34 - picture MMX1.PS,*,+82,*
35 - picture MMY10.PS,*,+82,*
36 - picture MMY5.PS,*,+82,*
37 - picture MMY1.PS,*,+82,*
38 - picture CONTOURS.PS,*,+82,*

```



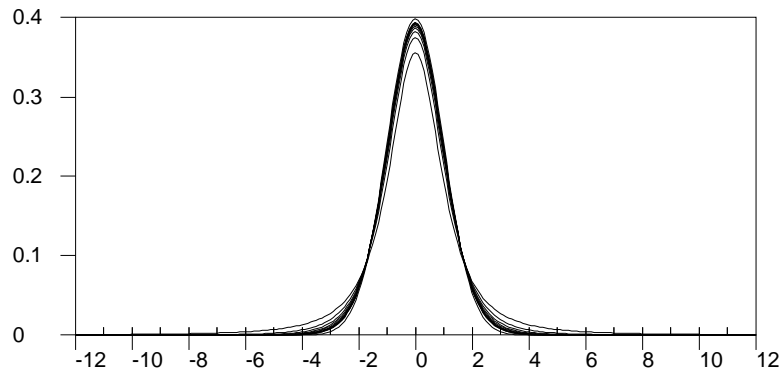
Plotting a graph paper and contour ellipses

```

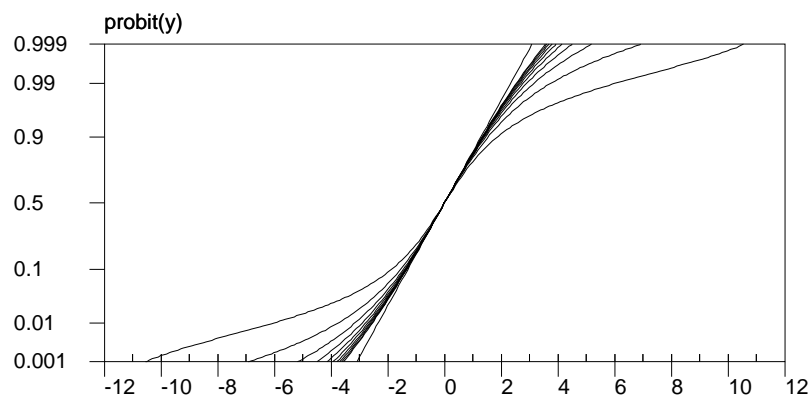
40 1 SURVO 84C EDITOR Tue Jul 21 09:37:12 1992      D:\P2\LOT4\ 100 100 0
1 *
2 *HEADER=Normal distribution as a limit of t distributions (df=2,4,...,20)
3 *SIZE=1164,600 XDIV=170,900,94 YDIV=80,420,100 PEN=[Swiss(8)]
4 *INTEGRAL=1 XSCALE=-12(2)12 TICK=1                *GLOBAL*
5 *YSCALE=0(0.1)0.4 X=-12,12,0.1 DEVICE=PS
6 *
7 *PLOT Y(X)=exp(-0.5*X*X)                          / DEVICE=PS,TDIST1.PS
8 *
9 *PLOT Y(X)=(1+X*X/N)^(-(N+1)/2) / N=2,20,2
10 *X=-12,12,0.1 HEADER= FRAME=0                   / DEVICE=PS,TDIST2.PS
11 *
12 *YSCALE=*probit(y),0.001,0.01,0.1,0.5,0.9,0.99,0.999
13 *PLOT INTEGRAL Y(X)=exp(-0.5*X*X)                / DEVICE=PS,TDIST3.PS
14 *
15 *YSCALE=*probit(y),0.001,0.01,0.1,0.5,0.9,0.99,0.999
16 *PLOT INTEGRAL Y(X)=(1+X*X/N)^(-(N+1)/2) / N=2,20,2
17 *X=-12,12,0.1 HEADER= FRAME=0                   DEVICE=PS,TDIST4.PS
18 *

```

Normal distribution as a limit of t distributions (df=2,4,...,20)



Normal distribution as a limit of t distributions (df=2,4,...,20)

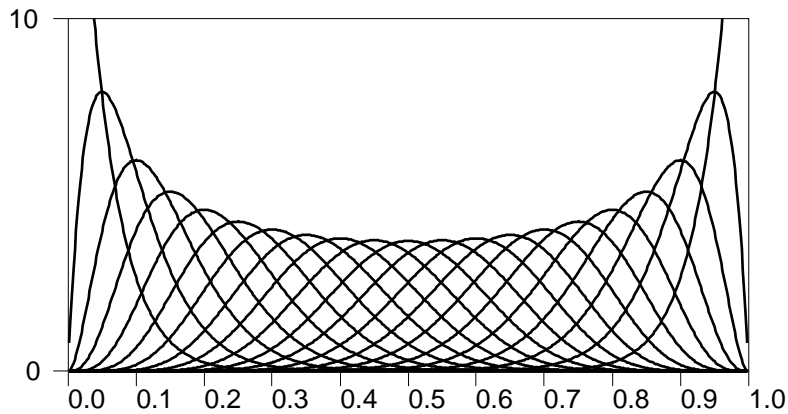



```

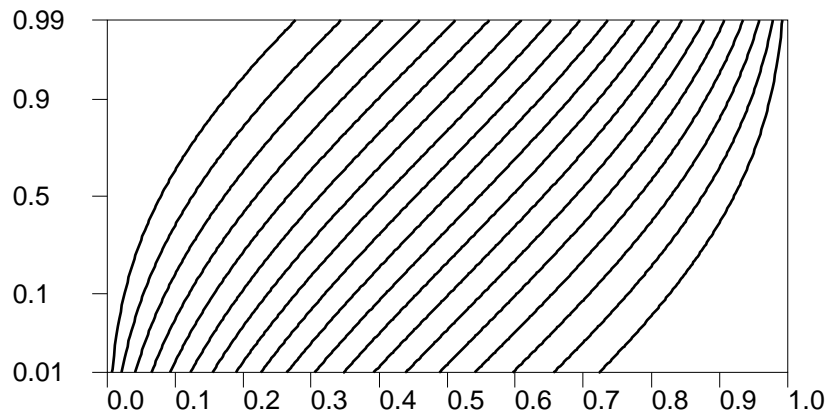
39 1 SURVO 84C EDITOR Tue Jul 21 09:40:36 1992      D:\P2\PLOT4\ 100 100 0
1 *
2 *HEADER=Order_statistics: Sample_from Uniform(0;1)_N=20
3 *PLOT Y(X)=X^(m-1)*(1-X)^(n-m) / INTEGRAL=1
4 *X=[line_width(1)],0.002,0.998,0.002          *GLOBAL*
5 *n=21 m=1,21,1  SIZE=1164,700 XDIV=164,900,100 DEVICE=PS,BETA1.PS
6 *XSCALE=0:0.0,0.1(0.1)0.9,1:1.0
7 *YSCALE=0,10  XLABEL=  YLABEL=
8 *.....
9 *HEADER=Distributions_on normal probability paper
10 *PLOT INTEGRAL Y(X)=X^(m-1)*(1-X)^(n-m)
11 *n=21 m=2,20,1  DEVICE=PS,BETA2.PS
12 *YSCALE=*probit(y),0.01,0.1,0.5,0.9,0.99
13 *

```

Order statistics: Sample from Uniform(0,1) N=20



Distributions on normal probability paper

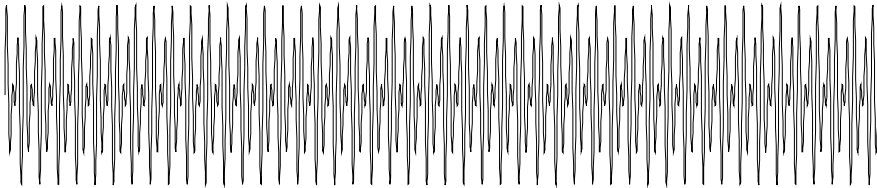


```

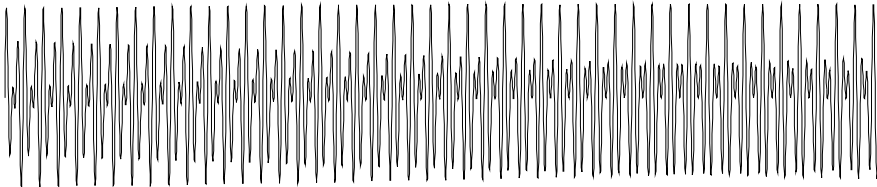
12 1 SURVO 84C EDITOR Tue Jul 21 12:17:59 1992 D:\P2\PLOT5\ 150 80 0
1 *
2 * Musical intervals in different temperaments
3 *
4 *YSCALE=-1,1 XSCALE=0,30 X=0,30,0.01
5 *SIZE=1164,350 YDIV=50,250,50 XDIV=0,1,0 FRAME=0
6 *.....
7 *HEADER=Pure Fifth
8 *PLOT Y(X)=(SIN(20*X)+SIN(30*X))/2 / DEVICE=PS,FIFTH1.PS
9 *.....
10 *HEADER=Fifth in Equal Temperament 20*2^(7/12)=29.9661415375
11 *PLOT Y(X)=(SIN(20*X)+SIN(29.96614*X))/2 / DEVICE=PS,FIFTH2.PS
12 *.....
13 *HEADER='Wolf' Fifth in Mean Tone Temperament 20*128*5^(-11/4)=30.624743
14 *PLOT Y(X)=(SIN(20*X)+SIN(30.62474*X))/2 / DEVICE=PS,FIFTH3.PS
15 *.....
16 *PRINT 17,22
17 % 400
18 - picture FIFTH1.PS
19 % 370
20 - picture FIFTH2.PS
21 % 370
22 - picture FIFTH3.PS
23 *

```

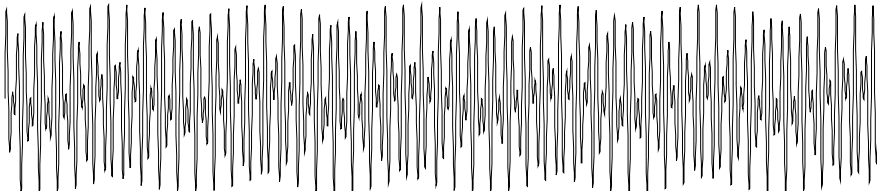
Pure Fifth



Fifth in Equal Temperament



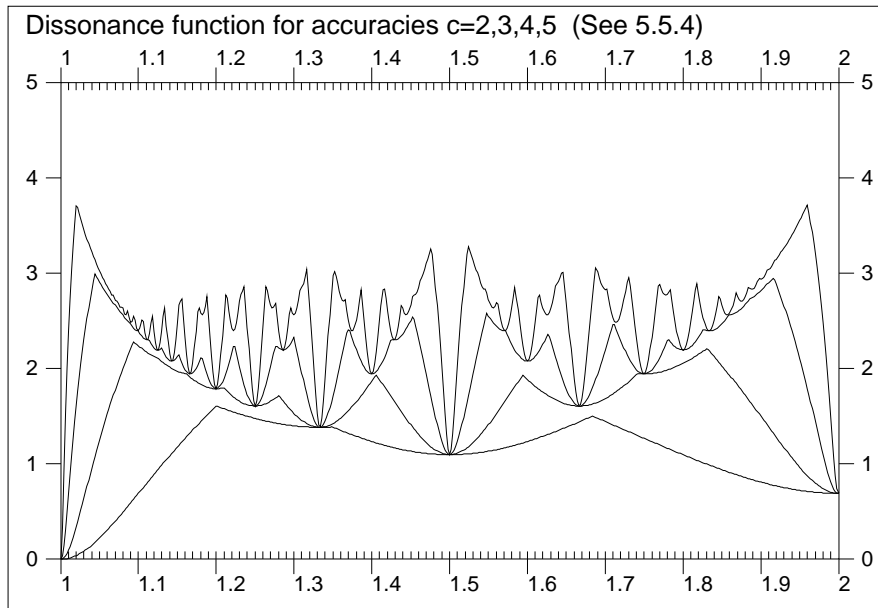
'Wolf' Fifth in Mean Tone Temperament



```

36 1 SURVO 84C EDITOR Sat Jul 25 12:13:10 1992      D:\P2\PLOT5\ 100 100 0
1 *
2 *FILE CREATE DISS,20,5,64,8,501
3 *FIELDS:
4 *1 N 4 X
5 *2 N 4 DISS2
6 *3 N 4 DISS3
7 *4 N 4 DISS4
8 *5 N 4 DISS5
9 *END
10 *
11 *VAR X,DISS2,DISS3,DISS4,DISS5 TO DISS
12 *X=1+(ORDER-1)/500
13 *DISS2=Diss(2,X) DISS3=Diss(3,X) DISS4=Diss(4,X) DISS5=Diss(5,X)
14 *.....
15 *HEADER=[Swiss(10)],Dissonance_function_for_accuracies_c=2,3,4,5_(See_5.
16 *SIZE=1164,800 XDIV=70,1030,64 YDIV=70,630,100
17 *          XSCALE=1(0.1)2 YSCALE=0(1)5 TICK=0.01,0 TICK2=TICK
18 *          XSCALE2=XSCALE YSCALE2=YSCALE XLABEL= YLABEL=
19 *LINE=1
20 *PLOT DISS,X,DISS2,DISS3,DISS4,DISS5
21 *DEVICE=PS,DISS.PS PEN=[Swiss(8)]
22 *

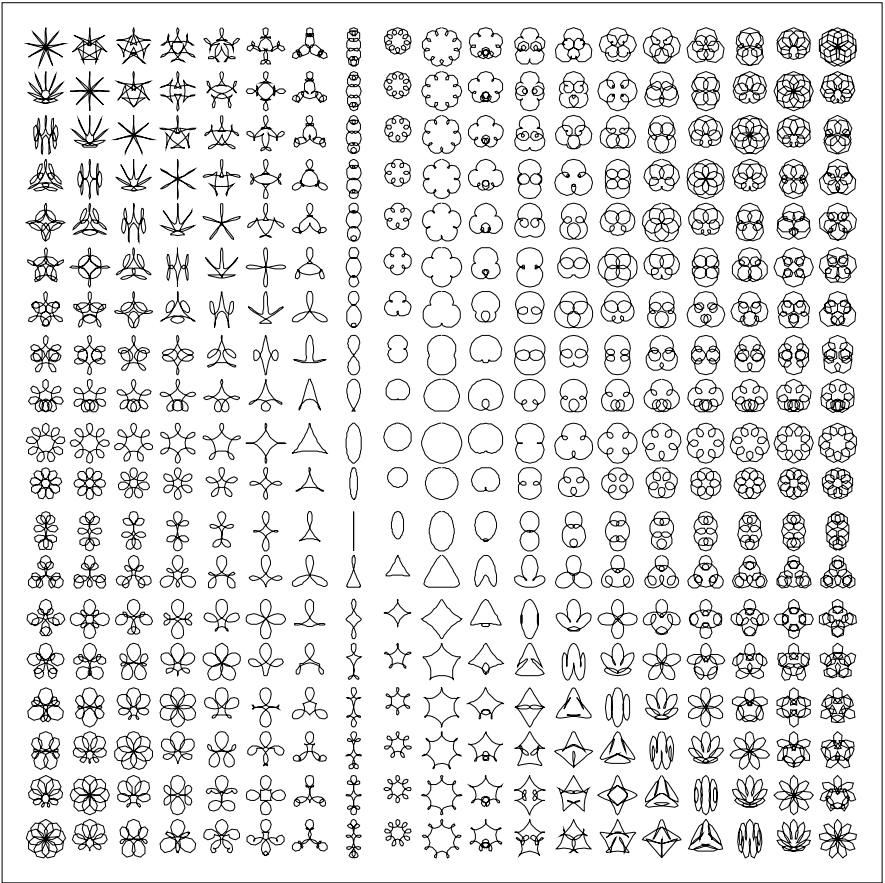
```



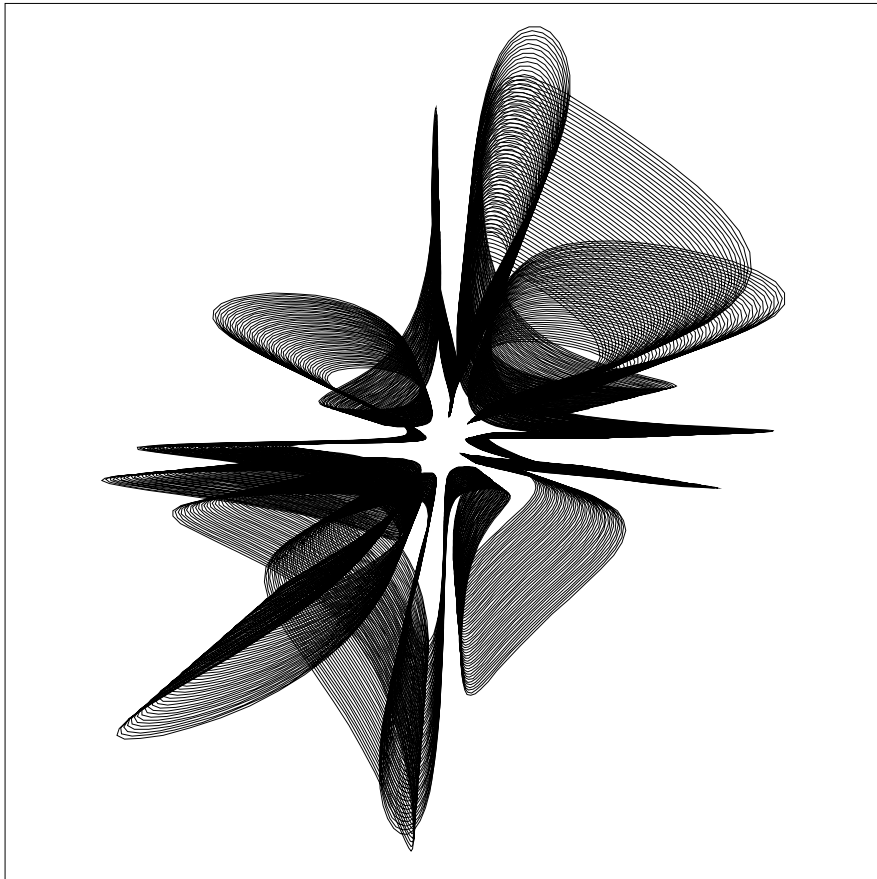
Herman Helmholtz (1821-1894) in his classical treatise "On the Sensations of Tone" has given a corresponding graphical presentation. It is based on his practical experiments on violin and on his acoustic theory (see *ibid* pp. 192-3 and appendix XV).

Our approach in 5.5.4 is quite different but gives very similar results. The Survo library for editorial computing includes functions $\text{diss}(c,x)$ and $\text{diss.f}(c,x)$. The first one gives the dissonance value of interval x for accuracy of the ear $c > 0$. The latter one gives the optimal ratio $n:m$ in the form $m+n/1000$.

```
56 1 SURVO 84C EDITOR Tue Jul 21 12:22:07 1992 D:\P2\LOT5\ 150 80 0
1 *
2 * "Origin of species"
3 *
4 *XDIV=0,1,0 YDIV=0,1,0 SIZE=1164,1164 HEADER= FRAME=3
5 *A=-8,10,1 B=-8,10,1 T=0,2*pi,pi/30 pi=3.14159265
6 *XSCALE=-9,11 YSCALE=-9,11 DEVICE=PS,SPECIES.PS
7 *
8 *PLOT X(T)=A+0.225*SIN(T)+0.139*SIN(A*T)+0.086*SIN(B*T),-
9 * Y(T)=B+0.225*COS(T)+0.139*COS(A*T)+0.086*COS(B*T)
10 *
```



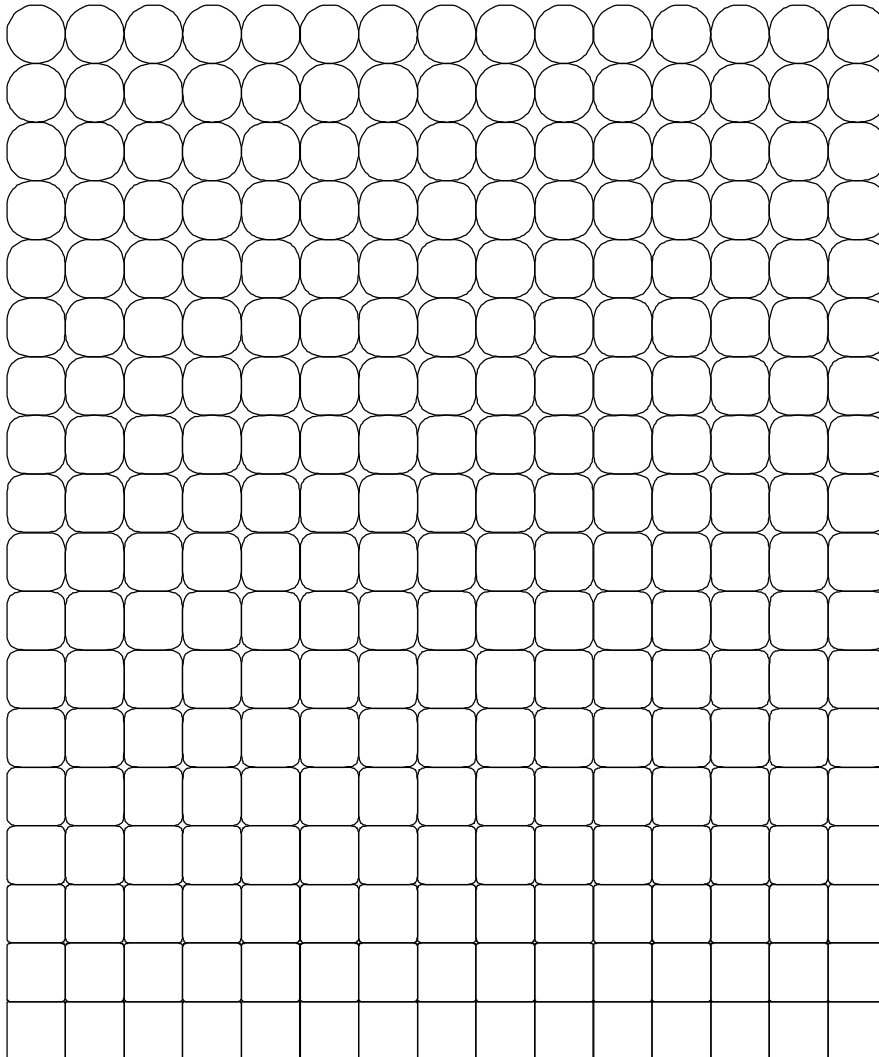
```
45 1 SURVO 84C EDITOR Tue Jul 21 12:54:02 1992 D:\P2\LOT5\ 150 80 0
33 *
34 *HEADER= FRAME=3 SIZE=1164,1164 DEVICE=PS, SPIRAL.PS
35 *XDIV=0,1,0 YDIV=0,1,0 SCALE=-23,23 T=300,600,0.01
36 *PLOT X(T)=0.03*T*(exp(-1+sin(17*T)))*cos(T),_
37 * Y(T)=0.09*T*(exp(-1.9+cos(11*T)))*sin(T+0.5)
38 *
```



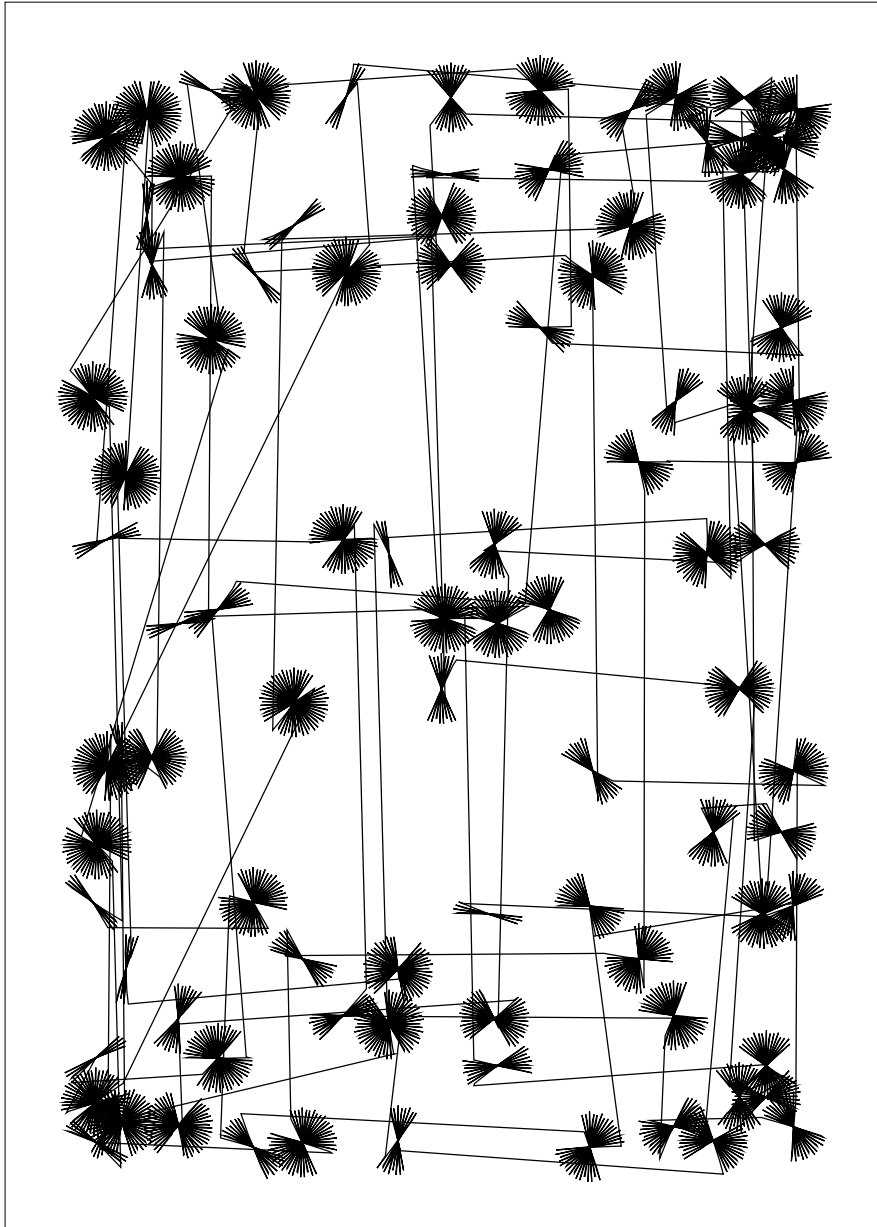
```

32 1 SURVO 84C EDITOR Tue Jul 21 13:29:09 1992 D:\P2\PLOT5\ 150 80 0
39 *
40 * "From circle to square"
41 *
42 * PLOT X(T)=M+1-P+R*CT*ABS(CT)^A,
43 * Y(T)=Q+R*ST*ABS(ST)^A
44 * CT=COS(T) ST=SIN(T) DEVICE=PS,CIRCSQR.PS HEADER=
45 * T=[line_width(0.48)],0.001,2*pi+0.001,pi/30 pi=3.14159
46 * FRAME=0 R=0.5 A=-1+(P+M*(Q-1))/(M*N)
47 * M=15 N=18 P=1,M,1 Q=1,N,1
48 * XSCALE=0.5,M+0.5 YSCALE=0.5,N+0.5
49 * XDIV=0,1,0 YDIV=0,1,0 SIZE=1164,1164*N/M
50 *

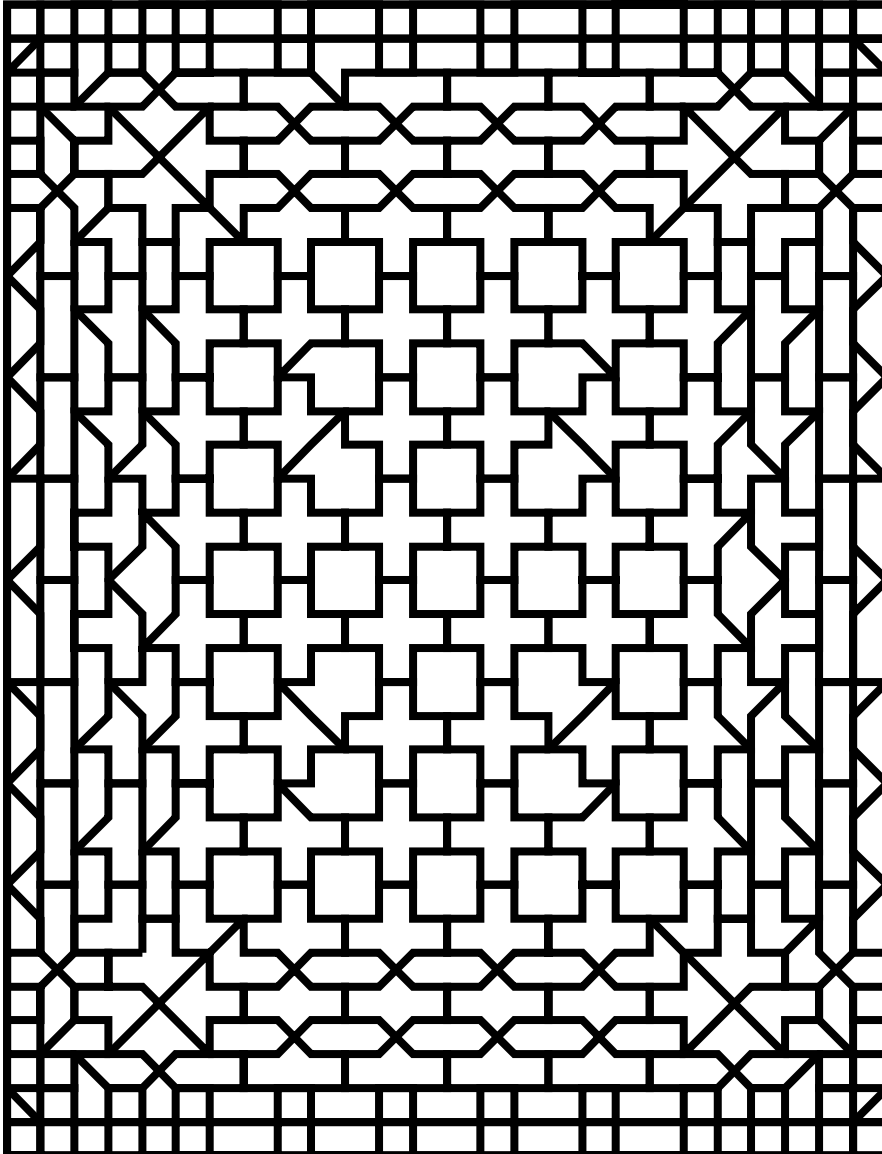
```



```
49 1 SURVO 84C EDITOR Tue Jul 21 13:49:17 1992 D:\P2\LOT5\ 150 80 0
51 *
52 *XSCALE=-10,10 YSCALE=-14,14 FRAME=3
53 *SIZE=1164,1630 XDIV=0,1,0 YDIV=0,1,0
54 *a=53 t=[line_width(0.48)],0,50*pi,pi/150 pi=3.14159
55 *HEADER= DEVICE=PS,THING.PS
56 *PLOT X(t)=8*sin(int(0.3*t))+0.8*sin(a*t)*cos(t),_
57 * Y(t)=12*sin(int(0.35*t))+0.8*sin(a*t)*sin(t)
58 *
```



```
31 1 SURVO 84C EDITOR Tue Jul 21 14:39:02 1992 D:\P2\PLOT5\ 150 80 0
59 *
60 *PLOT X(T)=int(M*cos(N*T)+0.5),-
61 * Y(T)=int(N*sin(M*T)+0.5)
62 *M=13 N=17 T=[line_width(3)],0,2*pi,pi/2048 pi=3.14159
63 *HEADER= FRAME=0 XSCALE=-M,M YSCALE=-N,N
64 *XDIV=0,1,0 YDIV=0,1,0 SIZE=1164,1164*N/M DEVICE=PS,CARPET.PS
65 *
```

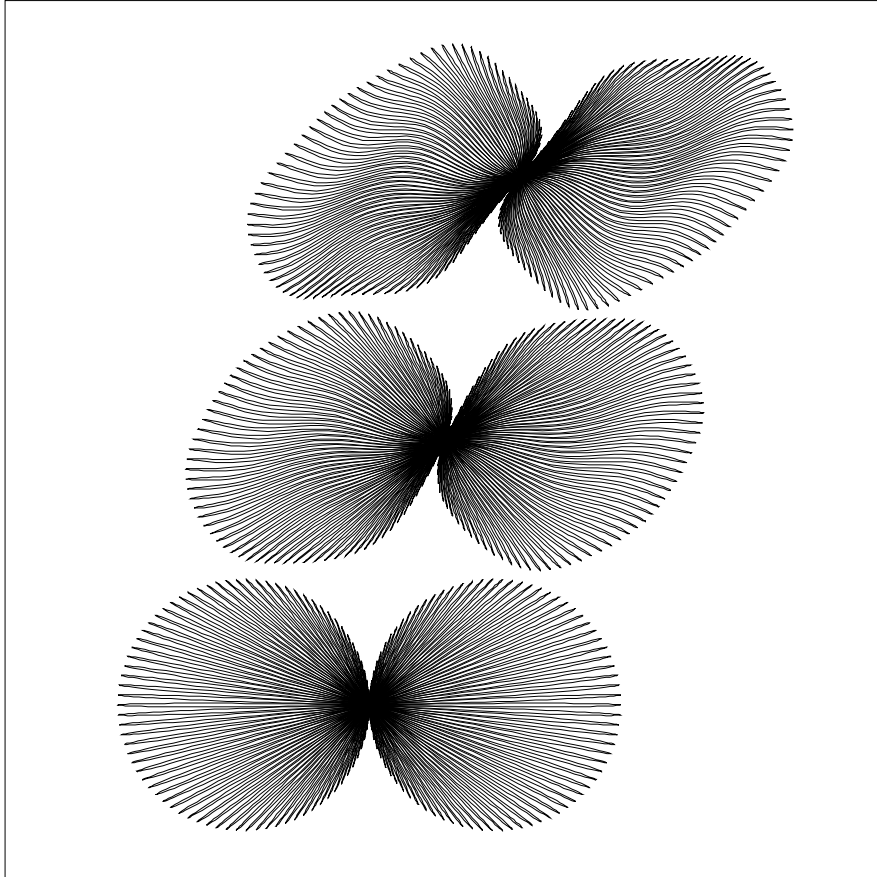


Butterfly 2009/SM

```

38 1 SURVO 84C EDITOR Tue Jul 21 15:04:29 1992      D:\P2\PLOT5\ 150 80 0
65 *
66 *HEADER= FRAME=3 SIZE=1164,1164  XDIV=0,1,0  YDIV=0,1,0
67 *T=0,2*pi,pi/4000  pi=3.14159265  SCALE=-3.5,3.5  DEVICE=PS,THING2.PS
68 *R=cos(78*T)+cos(80*T)  A=R*cos(T)  B=R*sin(T)
69 *p=0,2,1  s=0.25*p  u=0.03*p
70 *PLOT X(T)=0.6*(p-1)+A+s*B+u*sin(5*B),-
71 *      Y(T)=2.1*(p-1)+B+u*sin(5*A)
72 *

```



9. Printing of reports

Survo offers effective means for publishing results as complete documents. This text (produced entirely as a Survo application) is a typical example of its strength in desktop publishing. Thus, extensive reports with tables and pictures automatically included can be printed while working with Survo.

This section is devoted to the PostScript interface of Survo. At the moment, this interface provides the best possibilities for desktop publishing applications. Many features to be described are available also on other printers supported by Survo.

It is *not* necessary to be familiar with the PostScript language. All the important features of PostScript have been converted to the form of Survo printing and graphics. In some advanced applications, however, it may be useful to become somewhat acquainted with PostScript since one can insert pure PostScript code on control lines (on lines with a ‘-’ in the control column) of PRINT lists. Examples of this will be given later on.

For the most part, the PostScript interface of Survo is defined in the device driver `PS.DEV` which like the other Survo drivers is a standard ASCII file. Its current contents can be studied by the command `SHOW .\SYS\PS.DEV`, for example.

The first half of `PS.DEV` defines the keywords (in brackets) to be used in PRINT on control lines and in PLOT as specifications. These keywords are principally the same as those on other printers. The user can redefine these keywords during the printing process (this is one feature it has in common with PostScript) and define more. If one wishes to change `PS.DEV` permanently, it is best to make a copy of the standard version of `PS.DEV` under another name and make changes in that copy. The new file is selected as a default driver by changing the line `print_dev` in the `SURVO.APU` file correspondingly.

The default driver can also be temporarily overridden by another with the extension `.DEV` in its file name by giving a control line

```
- include <driver.DEV>
```

as the first line in the PRINT list.

The second half of `PS.DEV` includes several PostScript definitions and routines needed for carrying out the tasks required by the Survo control words. There is no reason to ‘understand’ these items in normal use. However, this transparent interface greatly simplifies more demanding applications.

The entire character stream to be sent to the printer can also be directed to a file. In the PRINT operation, this happens by giving the command in the form `PRINT 11,END TO A:REPORT.PS`

and in PLOT operations by a `DEVICE` specification

DEVICE=PS, PICTURE1.PS .

DEVICE=PS (without a file name) implies that the graph will be plotted directly on the printer.

The PostScript files generated by Survo should have the extension .PS in their names. They are standard ASCII files. Thus, one can edit them with any text editor. In Survo, operations like LOADP and SAVEP are useful for this purpose.

The PostScript files are to a great extent device-independent. Thus documents defined by them can be printed on any PostScript printer whether this is connected to the current Survo installation or not. PostScript files are also accepted as such on certain high-quality typesetters.

9.1 Text printing

The general structure of the PRINT operation is simply PRINT L1,L2 where L1 is the first and L2 the last edit line to be printed. Text and graphics from other sources are included by lines - chapter, - text, - picture, and - epsfile in the PRINT list. This will be described later in this section.

The control words and codes of Survo are written in brackets on control lines or their influence is carried within a single line by means of shadow characters (color codes).

A simple example of text output is:

```

12 1 SURVO 84C EDITOR Wed Jan 13 19:28:56 1988      D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 4,END-
3 *
4 - [Swiss(10)][line_spacing(12)][left_margin(12)]
5 *These lines are printed using the 10 point Helvetica font (Swiss)
6 *and 12 point line spacing. The left margin is 1 inch or 12 picas.
7 *The relations between the units are
8 * 1 inch = 12 picas = 72 Points.
9 *

```

When PRINT on line 2 is activated, we get the printout:

```

These lines are printed using the 10 point Helvetica font (Swiss)
and 12 point line spacing. The left margin is 1 inch or 12 picas.
The relations between the units are
 1 inch = 12 picas = 72 Points.

```

The following fonts should always be available on PostScript devices:

| | |
|--------------------------------|----------------------------|
| | In Survo (Point size 10.5) |
| Times-Roman | [TIMES(10.5)] |
| Times-Bold | [TIMESB(10.5)] |
| <i>Times-Italic</i> | [TIMESI(10.5)] |
| <i>Times-BoldItalic</i> | [TIMESBI(10.5)] |

| | |
|---|-------------------|
| Helvetica | [SWISS(10.5)] |
| Helvetica-Bold | [SWISSB(10.5)] |
| <i>Helvetica-Oblique</i> | [SWISSO(10.5)] |
| <i>Helvetica-BoldOblique</i> | [SWISSBO(10.5)] |
| Courier | [COURIER(10.5)] |
| Courier-Bold | [COURIERB(10.5)] |
| <i>Courier-Oblique</i> | [COURIERO(10.5)] |
| <i>Courier-BoldOblique</i> | [COURIERBO(10.5)] |
| Symbol | [SYMBOL] |
| αβχδεφγηηθικλμνοπθρστυωξψζ √∃∞←↑→↓°∅∩∪ etc. | |

Furthermore, depending on the printer, other PostScript fonts can be used. PSFONTS.DV2 is an auxiliary driver for additional fonts (See PSFONTS?).

To activate them, a control line

```
- include PSFONTS.DV2
```

must be appear in the PRINT list.

9.2 Control lines and words

The control words given within brackets on control lines of the PRINT list are defined by the `define` lines of the printer driver. In case of PostScript, the default file is `PS.DEV` in the `.\SYS` subdirectory of Survo. The user can also change these definitions and create new control words by inserting control lines starting with `- define`.

For example, the line

```
- define [BIGSPACE] [line_spacing(24)]
```

specifies a 24 point line spacing [BIGSPACE]. The definition line as such is not enough to activate [BIGSPACE]. This must be done (maybe in connection with other controls) as follows:

```
- [SWISS(14)][BIGSPACE].
```

In bracketed control words, the PRINT and PLOT operations are not case sensitive. For example, [TRIM(60)] is the same as [trim(60)]. This does not hold true with characters from the latter half of the ASCII code like the accented letters å, ä, ö, etc.

If more control words are defined in the PRINT list, it is permitted to employ any earlier control word in those definitions. In most cases, the predefined control words (given in `PS.DEV`) are sufficient. The most important words are the following:

```
[line_spacing(x)]
```

selects a line spacing of `x` Points (`x/72` inches). Default is 12.

```
[page_length(x)]
```

selects a page length of `x` Points. Default is 780 Points.

[left_margin(x)]

sets the left margin to x *picas* ($x/12$ inches). Default is 18.

[margin(x)]

sets the left margin to x *dmm* (10 dmm = 1 mm).

[COPIES(x)]

sets the number of copies of each page to x . Default is 1.

[LANDSCAPE]

rotates the A4 page 90 degrees and the next lines will be printed from the top of the rotated page using the current left margin. The page length is not changed automatically, but it can be reselected separately by

[page_length(528)], for example.

[PORTRAIT]

restores the original orientation of the page and the next lines will be printed from the top line onwards.

[STORE(x)]

saves the current printing position as a point with a label x . This x can be any word. Example: [STORE(start)].

[JUMP(x)]

restores the printing position saved by [STORE(x)].

[POINT(x,y)]

takes a new printing position with coordinates x,y given in dmm units. The origin is the bottom left corner of the page. The left margin is not changed, but it can be redefined when necessary with [margin(x)] where x is given in dmm, too.

[COUNT_OFF]

breaks counting of lines and thus prevents an automatic page change.

[COUNT_ON]


continues line counting from the situation before the last [COUNT_OFF].

Control words [COUNT_ON] and [COUNT_OFF] are useful in printing text in several columns, for example.

9.3 Control and shadow characters

The printer is guided also by various characters put in the control column of the edit field. One can also employ shadow characters for various enhancements within printable lines. Then there is no need to insert additional control codes and the text on the screen is not cluttered up with extra characters.

Control characters

Each edit line starts with a control character (in column 0) and it is normally an asterisk (*). For editing purposes, one can reach the control column by keeping down the left arrow key  long enough.

In PRINT, various control characters have the following functions:

- The line is not printed as such, but it contains control words in brackets or PostScript code.
- / initiates a new page. A page change occurs automatically when the current page becomes full. See [page_length()].
- (indicates a conditional initiation of a new page. When a ‘ (’ in the control column has been encountered, the printing is temporarily deferred and the first line below the current line with a ‘) ’ in the control column will be searched for. Then the PRINT program studies whether all the lines between these parentheses can be printed on the current page. If not, a new page will be started before printing.
-) see (above.

T indicates a tab line. See 9.9.

U suppresses temporarily the effect of a ‘T’ (tab) line and trimming.

V extends the line to full width set by - [trim()].

C centers the current line within the current line width selected by the control word [trim()].

R moves the current line to the right edge according to the line width.

r works as R, but only for odd-numbered pages.

< initiates the first column in a printout of several columns.

= initiates an intermediate column in a printout of several columns. At the end of the previous column, a proper value for the left margin must be set by [left_margin()].

> initiates the last column.

& prints a given number of empty lines using the current line spacing. For example, & 20 prints 20 empty lines.

% works as & above but gives the empty space (in vertical direction) in dmm. For example, to make space for a picture having a height of 8.5 cm, a control line % 850 should be entered.

Shadow characters

Control words and characters are useful when entire lines and chapters are to be regulated. We also need, however, means for adjustments within lines. To avoid cryptic control sequences in the text, the Survo Editor provides extra shadow lines for characterwise control information.

A shadow line can be defined for each edit line by first moving the cursor to that line and pressing keys **F2:PREFIX** and **S**. Then the line below the current line will be temporarily erased and replaced by an empty line having the label *Shadow:* instead of the line number. The shadow line thus displayed can then be filled by any characters and edited in a normal way. When the keys **F2** and **S** are pressed anew, the normal display is resumed and the current contents of the shadow line registered. Usually non-blank characters of the shadow line create various display effects (like different color combinations, blinking, etc.).

Typing of shadow characters is made easier by a special keyboard sucro /S (activated by the key combination **F2 M S**). According to instructions of that sucro, one can type any number of different shadow characters in different positions directly without extra key strokes.

For the common shadow characters 1,2,3,4,5,6,7, the **F5:FORMAT** key provides the simplest method. This key, when pressed repeatedly, selects shadows 1,2,3,4,5,6,7,space,1,2,3,... in turn. When text is typed, the current shadow will be automatically included. When some of the shadows 1-7 is selected and the space bar is pressed, the current character in the edit field will also receive that shadow character. Thus, with the space bar, one can easily paint words in the edit field with a selected shadow effect (color).

The selected shadow character always appears as the last character of the header line of the edit field. The default character (space) appears as '0'. The **ENTER** key restores the default shadow.

The shadow lines constitute an essential part of the edit field. Their number in one edit field is limited by default to 20, but this number can be changed by the REDIM operation. For example REDIM 120,80,50 redimensions the current edit field to have 120 lines, 80 columns (+control column) and up to 50 shadow lines. If the allowed number of shadow lines is exceeded, an error message is displayed and the user is prompted to use REDIM with suitable parameters. When the edit field is saved by the SAVE operation, the shadow lines are saved with it. Similarly, when loading an edit field by LOAD, the shadow lines are loaded as well.

Shadow lines are used to achieve desired color combinations and display effects, but they also have special tasks in certain operations. In the text to be printed, the shadow characters indicate various enhancements like underlining, italics, etc.

Below, a mathematical expression is getting suitable shadow characters for printing. We have pressed keys **F2** and **S** when the cursor was on line 4 and now editing the shadow line thus created:

```

21 1 SURVO 84C EDITOR Sat Jun 06 19:13:57 1987 D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 3,END
3 *Printing a mathematical expression:
4 * (x1+x2+...+xn)2
Shadow: 662662666666263_
6 *

```

When the keys **F2** and **S** are pressed again, the shadow characters below line 4 will be registered as a shadow line of the edit line 4, and the normal display is taken up:

```

13 1 SURVO 84C EDITOR Sat Jun 06 19:13:57 1987 D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 3,END
3 *Printing a mathematical expression:
4 * (x1+x2+...+xn)2 (various color combinations on this line)
5 *This line was temporarily replaced by a shadow line.
6 *

```

If PRINT on line 2 is activated, we get the following printout:

```

Printing a mathematical expression:
(x1+x2+...+xn)2
This line was temporarily replaced by a shadow line.

```

In this example, we used shadow characters '6', '2' and '3' for italicizing, subscripts and superscripts, respectively. In fact, digits 1,2,...,7 as shadow characters are reserved for typical enhancements in text printing according to the following table:

| Shadow character | Effect on the screen | Effect in printing |
|------------------|----------------------|---|
| space | black (on white) | normal |
| 1 | red (white) | bold |
| 2 | gray (white) | subscript form of the current font |
| 3 | blue (white) | superscript form of the current font |
| 4 | white (blue) | <u>underlining</u> |
| 5 | blinking yellow | text on a gray background |
| 6 | green (white) | <i>oblique or italicized form of the current font</i> |
| 7 | blue (light blue) | white text on black background |
| S | gray (red) | Symbol font in the current font size |

The effects of the shadow characters in printing are defined by the shadow lines in printer drivers (or by - shadow lines in the print list). The general structure of a control line is then

```
shadow <character>: <prefix_code> <postfix_code>
```

where the effect of any <character> is defined by two control codes. The first of them, <prefix_code>, is to be sent before the actual character is printed and

the second one, <postfix_code> afterwards.

For example, if character 'i' has shadow '2', the PS.DEV driver sends 'i' according to a shadow definition

```
shadow 2: [SUB] [SUBN]
```

as a code sequence

```
[SUB]i[SUBN]
```

The control words [SUB] and [SUBN] are defined as PostScript functions **sub** and **subn**. The first of them reduces the size of the current text font to about 70 per cent and moves the cursor slightly downwards. Then 'i' is printed as a subscript. The second PostScript function **subn** (corresponding to [SUBN]) simply restores the original font size and line position.

In many ways, shadow characters are helpful in indicating changes of the font type within a line.

In PostScript, for example, Times-Roman, Times-Bold, and Times-Italic are technically not related to each other; they are separate font types. In Survo, however, it is reasonable to keep them derivatives of a single font 'Times'. Thus, when using shadow characters 1=Bold and 6=Italic, the PS.DEV driver is able to select a modification of a 'right' font.

The next PRINT scheme

```
13 1 SURVO 84C EDITOR Fri Jan 15 16:39:46 1988 D:\P2\PRI\ 120 80 0
13 *
14 *PRINT 15,END
15 - [TIMES(10)]
16 * Here is text as bold, underlined and in italics.
17 - [SWISS(10)]
18 * Here is text as bold, underlined and in italics.
shadow:          1111 4444444444 6666666666
20 - / Shadows of line 18 are displayed here for illustration, only.
21 - / Control lines starting with / surrounded by blanks have no effect.
22 *
```

gives the result:

```
Here is text as bold, underlined and in italics.
Here is text as bold, underlined and in italics.
```

The font type may be changed within a line by defining suitable new shadow codes for desired fonts. For example, in the following exhibit a word in Courier will be inserted into text in Times by defining I as the shadow of 12 point Courier and M as the shadow of 12 point Times:

```

13 1 SURVO 84C EDITOR Fri Jan 15 17:07:50 1988 D:\P2\PRI\ 120 80 0
23 *
24 *PRINT 25,END
25 - shadow I: [Courier(12)]
26 - shadow M: [Times(12)]
27 *
28 * New control words are described by define lines.
Shadow:
30 *

```

We get the following printout:

```
New control words are described by define lines.
```

9.4 Multipage documents

The PRINT list can also include control lines referring to chapters in other edit fields, text files, and picture files. Thus, material from other sources may be linked to make a uniform multipage document by a single PRINT activation.

Selected lines from another edit field are included by naming these lines in the corresponding edit file as a chapter by a DEF definition of the form

```
DEF <name_of_chapter> , <first_line> , <last_line> .
```

This chapter is then referred to in a PRINT list by a control line of the form

```
- chapter <name_of_chapter> IN <name_of_edit_file> .
```

The PRINT lists for large documents are mainly lists of chapters defined in such a way. See examples in 3.1-2.

Also parts of text files may be included. A control line of the form

```
- text <name_of_text_file> , L1 , L2
```

specifies lines L1 to L2 from an ASCII file to be included in the set of lines to be printed. If L1 and L2 are omitted the entire file will be printed. If L2 is omitted, lines from L1 up to the end of file are printed.

Usually edit files are more suitable for Survo documents because various enhancements within the text (by means of control lines and shadow characters) are then available. Various text attributes for text files have to be selected by control codes before entering the - text control line.

Page changes

The PRINT operation starts a new page automatically when the current page length (set by [page_length(x)]) is reached. The user may, however, control page changes by inserting certain characters in the control column. '/' always starts a new page. '(' starts and ')' terminates a table or a chapter that should be printed on the same page as a whole.

When PRINT is working, all printable lines are displayed in a temporary window below the PRINT line. To monitor the page changes, the user can press the `[+]` key. The process is then interrupted each time when a page ends. To continue, any key can be pressed. By redirecting the output to a file or to NUL, page changes can be checked without waste of paper.

Page headers and numbers

A control line in the PRINT list of the form

- `header_lines <first>, <last>`

specifies the header lines to be printed on each page. `<first>` and `<last>` refer to lines in the current edit field. The place and format for the line number can be specified on any of these lines in the form `###`. The page numbers to be printed will be 1,2,3, ... unless otherwise stated by a control line of the form

- `[page_number(51)]`

(i.e. 51 is the first page number).

Odd and even pages can be given different headers by

- `header_lines <odd_first>, <odd_last>, <even_first>, <even_last> .`

Roman numerals (i, ii, iii, iv, etc.) are printed as page numbers by entering masks of the form `@@@` on the header lines.

9.5 Pictures

PostScript files made by the PLOT operation and saved on disk by the specification `DEVICE=PS, <picture_file>` are included by a control line

- `picture <picture_file>, x, y, kx, ky, z`

where `x,y` are the coordinates (in dmm) of the bottom left corner of the graph, `kx,ky` are scaling coefficients and `z` an angle of rotation in degrees. If `z` is positive, the picture is will first be scaled and then rotated. If `z` is negative, the picture will first be rotated and then scaled by angle `-z`.

Furthermore, `x` and `y` can be replaced by relative coordinates of the form `*+C` or `*-C` where `C` is a constant. The reference level `*` of `y` is the base level of the current line and the reference level `*` of `x` is the current start position of the line (determined by the left margin). This feature simplifies positioning of (possibly overlapping) pictures with respect to the current text.

Normally Survo plots are transparent. However, in the preceding example we partly overlapped the graphic designs by using a **FRAMES** specification (on line 10). Here the only box **F** defined coincides exactly with the current **SIZE** specification. The last parameter (0) in the **F** specification paints the entire area white, thus erasing all the previous graphics and text inside the box.

Encapsulated PostScript files (EPS)

Encapsulated PostScript files made in other systems are included in the Survo PostScript printing by inserting a control line of the form

```
- epsfile <EPS_file>,x,y
```

in the **PRINT** list. The options are the same as for the `- picture` lines described above. In the graphs made by Survo itself, the `- picture` option should always be preferred to.

Extra space for pictures

It is the user's responsibility to reserve enough space for pictures. A control line of the form

```
% n
```

(% in the control column) moves the current printing position on the page `n` `dmm` downwards.

9.6 Justifying the right edge of the text

The right edge will be correctly aligned by adjusting the gaps between the words evenly. The desired line length is set by `[trim(x)]`, where `x` is given in picas (1/12 inches).

The **PRINT** operation never alters the setting of words between the lines. It is the user's responsibility to edit the lines to a suitable form by any of **TRIM** commands.

The simplest way of adjusting text according to a selected line length and a selected font type and size is to use the `/TRIMP` survo command having the form

```
/TRIMP L1,L2,C,F(S)
```

where `L1-L2` are the lines to be adjusted, `C` is the desired line length in picas, `F` is the font type and `S` is the font size in Points. For more information, see 3.3.

The line width can be changed within the text by giving a new `[trim()]`. As a special case, `[trim(0)]` discards the fixed line length and subsequent lines will be printed without any right edge adjustment. In fact, this is the default setting when **PRINT** is activated.

When using a specific line length there is a certain tolerance (defined in `PS.DEV`) for line adjustment. If the line is too short so that the gaps between

the words would be very wide, no adjustment will take place. Similarly, if the line is too long, it will be printed with minimal gaps (typically 70 per cent of a normal blank) and the result will be seen as an overlong line. The adjustment can also be suspended temporarily for selected lines by putting a 'U' in the control column. On the other hand, a short line can be extended to a full line length by a 'V' in the control column.

The tolerance parameters in automatic typesetting of a line are controlled by the code word [END_TOL(u,v)] where u refers to the tolerance in normal lines and v to the tolerance in lines ending to a full stop, colon, etc. [END_TOL(1,0.3)] is the default setting.

9.7 Footnotes

Footnotes can be written and placed at the bottom of the current page automatically as follows:

Indicate in the text the place that refers to the footnote by the character '*' with 'F' as the shadow character. In the final printing, this character will be replaced by the index of the footnote.

This line must be followed by an empty line and the lines containing the footnote text. The footnote text is terminated by an empty line.

For example, the PRINT scheme

```

12 1 SURVO 84C EDITOR Sat Dec 10 18:04:19 1988 D:\P2\PRI\ 150 80 0
1 *
2 *PRINT 3,END
3 *This is a line having references* to two footnotes.*
Shadow:                               F           F
4 *
5 * This is the first footnote text.
6 *
7 * This is the second footnote text
8 *taking two lines.
9 *
10 *The text continues...
11 *
12 *This is the last line on the page.
13 *

```

produces the following 'page':

This is a line having references¹ to two footnotes.²

The text continues...

This is the last line on the page.

¹This is the first footnote text.

²This is the second footnote text
taking two lines.

The control of the page changes is automatic. PRINT takes into account the lengths of the footnotes.

The reference numbers for footnotes are 1,2,3,... unless otherwise stated by [`footnote_number(n)`] where n will be the number of the next footnote. By using [`footnote_number(1)`] on header lines, counting will be reset for each page.

The typeface etc. for footnotes is controlled by the control words [`NOTE_START`] and [`NOTE_END`]. See the current device driver for default settings.

The bar above the first footnote is drawn by the PostScript procedure `footnote_bar` defined in `PS.DEV`. The user can redefine this procedure in the PRINT list.

9.8 Automatic index

An index of keywords and phrases occurring in the text can be created as a by-product of PRINT. The automatic index file is initiated by the control line - `index <name_of_text_file>`

The previous contents of the text file will be lost.

The keywords must be indicated in the text either by putting a '*' as a control character to any place within a word or by indicating the start of a (longer) keyphrase by the shadow '[' and the end of it by the shadow ']'.

The keyboard sucro S simplifies typing of such shadow characters. The resulting text file containing the keywords and their page numbers can be edited later. For example, it is converted into a Survo data file by `FILE SAVE`, sorted by `FILE SORT` and edited by `FILE SHOW`.

9.9 Tables

Printing of tables in variable pitch fonts also calls for special attention. Although the numerals are of equal pitch, the blank (space) symbol is usually narrower and this may twist columns which ought to be aligned.

We assume that any table to be printed should be correctly aligned already in the edit field or file before printing. To preserve the alignment, we must specify a tab line having a 'T' in the control column in the PRINT list just before the table.

The next example illustrates the use of a T line.

```

12 1 SURVO 84C EDITOR Tue Mar 01 12:38:52 1988      D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 3,END_
3 *
4 - [SWISS(9)][line_spacing(12)]
5 *Countries of Northern and Western Europe
6 *
7 T T *5          (          ) T   (          ) (          )
8 * Country      Population Urban Density Growth
9 * Finland      4907  66.9    15  0.30
10 * Denmark     5137  85.9    119 0.14
11 * Iceland     243   89.6     2  0.91
12 * Ireland     3484  60.9    50 0.99
13 * Norway      4150  57.2    13 0.38
14 * Sweden      8295  89.2    18 0.08
15 * Great Britain 55757 91.7   228 0.04
16 * Austria     7463  56.1    89 0.03
17 * Belgium    10076 73.6   330 0.42
18 * France     54265 80.4    99 0.34
19 * Germany, (FRG) 60013 86.1   241 -0.20
20 * Luxembourg  358   81.8   138 0.01
21 * Netherlands 14446 76.3   354 0.40
22 * Switzerland 6284  60.4   152 0.10
23 T

```

In this setup, the tab line 7 controls the setting of columns. The positions indicated by 'T's will be vertically aligned in the final printout as well and the text in the table will be positioned freely to the right of these columns until the next 'T' or a '(' is encountered on the tab line. In this table, columns 'Country' and 'Urban' will be aligned in this way.

The other columns enclosed by parentheses will be aligned according to the rightmost position indicated by a ')'. In these columns, the text (typically numbers) will be freely positioned to the left.

By default, each position in the table occupies 60 per cent of the font size. This means that, for example, when using the fixed pitch Courier font, the tab line has no effect. On variable pitch fonts, 60 per cent of the font size may be too much (when numbers and lower case letters are concerned). The user may then specify another character width by giving a notation of the type *n in any free place on the tab line. n is the desired character width in points. In our example, *5 on line 7 implies 5 Points per a position instead of the default value of $0.6 \times 9 = 5.4$ Points.

An empty T line without any other 'T' cancels the effect of a previous tab line. One can remove the effect of the tab line for any line temporarily by inserting the control character 'U'.

PRINT on line 2 gives the following table:

Countries of Northern and Western Europe

| Country | Population | Urban | Density | Growth |
|----------------|------------|-------|---------|--------|
| Finland | 4907 | 66.9 | 15 | 0.30 |
| Denmark | 5137 | 85.9 | 119 | 0.14 |
| Iceland | 243 | 89.6 | 2 | 0.91 |
| Ireland | 3484 | 60.9 | 50 | 0.99 |
| Norway | 4150 | 57.2 | 13 | 0.38 |
| Sweden | 8295 | 89.2 | 18 | 0.08 |
| Great Britain | 55757 | 91.7 | 228 | 0.04 |
| Austria | 7463 | 56.1 | 89 | 0.03 |
| Belgium | 10076 | 73.6 | 330 | 0.42 |
| France | 54265 | 80.4 | 99 | 0.34 |
| Germany, (FRG) | 60013 | 86.1 | 241 | -0.20 |
| Luxemburg | 358 | 81.8 | 138 | 0.01 |
| Netherlands | 14446 | 76.3 | 354 | 0.40 |
| Switzerland | 6284 | 60.4 | 152 | 0.10 |

The tables can also be surrounded by rectangular boxes which are defined by marking the opposite corner points. The previous table takes the form

Countries of Northern and Western Europe

| Country | Population | Urban | Density | Growth |
|----------------|------------|-------|---------|--------|
| Finland | 4907 | 66.9 | 15 | 0.30 |
| Denmark | 5137 | 85.9 | 119 | 0.14 |
| Iceland | 243 | 89.6 | 2 | 0.91 |
| Ireland | 3484 | 60.9 | 50 | 0.99 |
| Norway | 4150 | 57.2 | 13 | 0.38 |
| Sweden | 8295 | 89.2 | 18 | 0.08 |
| Great Britain | 55757 | 91.7 | 228 | 0.04 |
| Austria | 7463 | 56.1 | 89 | 0.03 |
| Belgium | 10076 | 73.6 | 330 | 0.42 |
| France | 54265 | 80.4 | 99 | 0.34 |
| Germany, (FRG) | 60013 | 86.1 | 241 | -0.20 |
| Luxemburg | 358 | 81.8 | 138 | 0.01 |
| Netherlands | 14446 | 76.3 | 354 | 0.40 |
| Switzerland | 6284 | 60.4 | 152 | 0.10 |

by using parentheses as shadow characters as follows:

```

13 1 SURVO 84C EDITOR Tue Mar 01 18:36:57 1988      D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 3,END
3 *
4 - [SWISS(9)][line_spacing(12)]
5 *Countries of Northern and Western Europe
6 *
7 T T *5          (          ) T (          ) (          ) T
8 * Country      Population Urban Density Growth
Shadow:(
9 * Finland      4907 66.9      15 0.30
10 * Denmark     5137 85.9      119 0.14
11 * Iceland     243 89.6       2 0.91
12 * Ireland     3484 60.9      50 0.99
13 * Norway      4150 57.2      13 0.38
14 * Sweden      8295 89.2      18 0.08
15 * Great Britain 55757 91.7    228 0.04
16 * Austria     7463 56.1      89 0.03
17 * Belgium    10076 73.6    330 0.42
18 * France     54265 80.4     99 0.34
19 * Germany, (FRG) 60013 86.1    241 -0.20
20 * Luxemburg   358 81.8     138 0.01
21 * Netherlands 14446 76.3    354 0.40
22 * Switzerland 6284 60.4     152 0.10
Shadow:
23 T
24 * Both shadow lines have been made here visible just for illustration.
25 *

```

The boxes here have been formed from two rectangles. The left upper corner of them both (in the beginning of line 8) is denoted by the shadow character ‘(’. The right bottom corner of the first box is denoted at the end of the same line by the shadow character ‘)’. The corresponding end point for the larger rectangle is given as a shadow character ‘)’ on line 22. An extra T at the end of line 7 guarantees that the lines will meet on the right side.

The PS.DEV driver specifies two pairs of parentheses, namely () and { }, as shadow characters for definition of two simultaneous boxes. In fact, more boxes can be drawn with these shadows whenever some of the boxes have a common starting point. Furthermore, the user can define more pairs of characters for this task according to the model given by the - shadow lines of the above-mentioned characters.

The next blank form has been created by using two pairs of box shadows

Blank form

| | |
|-------|-------|
| Box 1 | Box 2 |
| Box 3 | |
| Box 4 | |

as follows:

```

12 1 SURVO 84C EDITOR Sat Jan 16 17:23:48 1988 D:\P2\PRI\120 80 0
1 *
2 *PRINT 3,18
3 *
4 - [SWISS(14)][line_spacing(12)]
5 *Blank form
6 - [SWISS(8)]
7 T
8 * Box 1 T Box 2 T
Shadow: ( {
9 *
10 *
Shadow: )
11 * Box 3
12 *
13 *
Shadow: )
14 * Box 4
Shadow: (
15 *
16 *
Shadow: )
17 *
18 *

```

9.10 Box graphics

Another and in most cases simpler way to draw vertical and horizontal lines as well as boxes, is to use the graphic characters of the extended ASCII character set. To accomplish this, an auxiliary driver `GCHAR.DV2` must be included by giving a control line

```
- include GCHAR.DV2
```

in the PRINT list.

Typing of box characters in the edit field is easy by using a special keyboard macro `/BOX`. More information about `BOX` is obtained by activating `/BOX ?`.

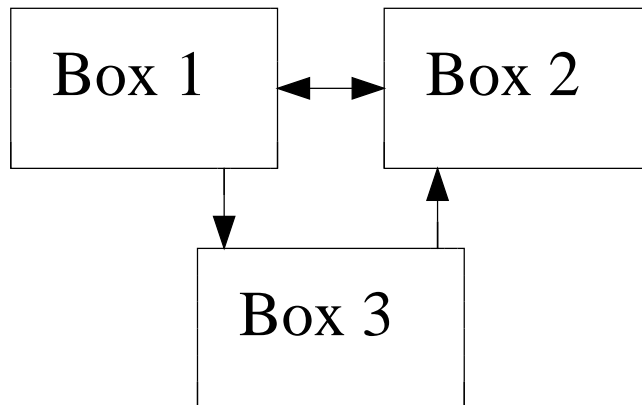
As an example, we reproduce a 'blank form' in a more elaborated shape by using graphic characters.


```

14 1 SURVO 84C EDITOR Wed Aug 19 16:11:19 1992 D:\P2\PRI\ 100 100 0
25 *
26 *PRINT CUR+1,E
27 - include GCHAR.DV2
28 - [Times(24)][line_spacing(10)]
29 *
30 T T *5 T T T
31 *
32 *
33 *
34 *
35 *
36 *
37 *
38 *
39 *
40 T *5 T T
41 *
42 *
43 *
44 *
45 *
46 *
47 *
48 T
49 E

```

gives the output:



9.11* Mathematical expressions

The current PostScript interface of Survo is not intended for large-scale, advanced typesetting of mathematical formulas. However, for scientific reports containing mathematical formulas to a lesser extent, an auxiliary driver, `MATH.DV2`, is available.

We are not going to describe all possibilities provided by `MATH.DV2`. Only a sample listing of a `PRINT` scheme involving mathematical expressions with its output is given here.

In the listing below, the shadow lines playing an important role are made visible. More information is obtained by listing the `MATH.DV2` file (in directory `.\SYS`).

```

12 1 SURVO 84C EDITOR Sun Dec 11 15:11:05 1988      D:\P2\PRI\ 100 100 0
1 *
2 *PRINT 3,END_
3 - include MATH.DV2
4 - define [MAIN] [Times(11)]
5 - define [INDEX] [Times(7.7)]
6 - [line_spacing(11)][MAIN]
7 - shadow m: [MAIN]
8 - shadow i: [INDEX]
9 *
10 *x1 + x2 + x3 + ... + xn
Shadow:62G G62G G62G      G68
11 *
12 *
13 - [INTEGRAL(1,1)]
14 * exp(-x2 / 2) dx -¥ ¥ =
Shadow: SS63gSgSS 66 iP SPQSQmBb b-
15 - [ROOT(1,1)]
16 * 2p
Shadow: ST
17 *
18 *
19 - [SIGMA(1.5,1.5)]
20 * n2 n=1 N = N(N+1)(2N+1) / 6
Shadow: 63 iPb6 PQg6QmBb 6S6SSSS6SSSgSgS -
21 - [PRODUCT(1.5,1.5)]
22 * n n=1 N = N!
Shadow: 6 iPbG6 PQ 6QmBb 6
23 *
24 *
25 - shadow 0: [RATIO(0.3,0.5,-0.5)]
26 * a + b c + d + d - e f - g
Shadow:0p6gSg6pq6gSg6 rq S 0p6gSg6pq6gSg6 rq
27 *
28 *
29 - [RATIO(0,-0.2,0.3)]
30 * lim n fi ¥
Shadow:iq qp6gSgSpm -
31 - shadow (: [LPAR(0,1,1.5)]
32 - shadow ): [RPAR(0,1,1.5)]
33 - shadow 0: [RATIO(0.3,0.6,-0.4)]
34 * 1 + x n n = e x
Shadow:(gSGS 0p6pq6qr)0pb9pg 6g9
35 *

```

$$x_1 + x_2 + x_3 + \dots + x_n$$

$$\int_{-\infty}^{\infty} \exp(-x^2/2) dx = \sqrt{2\pi}$$

$$\sum_{n=1}^N n^2 = N(N+1)(2N+1)/6 \quad \prod_{n=1}^N n = N!$$

$$\frac{a+b}{c+d} + \frac{d-e}{f-g}$$

$$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x$$

9.12* PostScript code in the PRINT list

In the PRINT operation, pure PostScript code may be written on the control lines of the PRINT list. For example, the scheme

```

12 1 SURVO 84C EDITOR Thu Jan 21 19:33:05 1988 D:\P2\PRI\ 120 80 0
1 *
2 *PRINT 3,END-
3 *Rotating square:
4 % 100
5 - [SWISSB(22)]
6 - /edge 25 def[LF]
7 - /square { currentpoint newpath moveto edge neg 0 rlineto[LF]
8 -           0 edge rlineto edge 0 rlineto closepath[LF]
9 -           gsave 1 setgray fill grestore stroke } def[LF]
10 - /S { edge -0.8 mul edge 0.2 mul moveto (S) show } def[LF]
11 - gsave currentpoint /y exch def /x exch def x edge add y translate[LF]
12 - 0 5 90 { gsave /angle exch def edge 8 div angle mul 0 translate[LF]
13 -         angle neg rotate 0 0 moveto square S grestore } for[LF]
14 - grestore x y moveto[LF]
15 *

```

produces the following:

Rotating square:



Although our intention here is not to teach PostScript programming, we try to explain briefly the essence of this example.

All the PostScript lines (in a PRINT list) must have a '-' in the control column and they must be terminated by a line feed [LF] or a space [S]. These line end codes are essential because PRINT does not automatically produce any line end codes on control lines.

PostScript code written in the PRINT list will be treated in the environment created by the driver PS.DEV and by the earlier lines in the same PRINT list. Thus various bracketed control words of Survo have their natural effects in PostScript code.

In our example, line 5 defines the 22 point Helvetica as the current font type. The actual PostScript lines 6-14 work as follows:

Line 6 gives the value 25 Points for the side length of the square. One Point or 1/72 inches is the default unit in PostScript plotting and Survo obeys this convention, too.

On lines 7-9, a procedure `square` is defined. It creates a path for a square, paints it white, and finally draws the edges.

Procedure `S` on line 10 writes the capital S inside the square. Line 11 saves the current graphics state and the current point (x,y). The origin is then moved

by the edge of the square to the right from the current point (on the left margin).

The program loop on lines 12-13 rotates the coordinate system by angles of 0,5,10,...,90 degrees and at the same time moves the bottom right corner of the square $\text{angle}/8$ Points to the right. During each round of the loop the square and 'S' in it will be plotted.

Line 14 restores the original graphics state and takes (x, y) again as the current point. Thus in PRINT we are back in the same situation on the page as we were before making these special graphics.

More information is found about PostScript e.g. in the following books:

PostScript Language Reference Manual, Adobe Systems Inc.,

PostScript Language Tutorial and Cookbook, Adobe Systems Inc.

These documents have been published in 1985 by Addison-Wesley.

We emphasize that mastering PostScript is not essential for a Survo user. All important features of PostScript have well-defined counterparts as regular control words in Survo printing and plotting. Only in very demanding applications is direct application of PostScript necessary.

10. Matrix interpreter

Many of the Survo operations give their results in a matrix form and save them in matrix files. Especially in statistical modules working in the areas of linear models and multivariate analysis, it is necessary to have a common standard for matrix management and operations.

In some cases, manipulation of matrices obtained in standard analysis is required. In teaching situations, it is also usually clearer to perform matrix computations step by step. Similarly, when testing new computational methods based on matrix algebra, it is valuable to have immediate access to matrix computations.

The Survo matrix interpreter is a tool for these multifarious computational needs. Although some of its properties have already been treated in the section on multivariate statistical operations, we shall here give a general account.

The matrix interpreter is conducted by Survo operations having **MAT** as the first command word. There are about 50 different **MAT** operations. All of them work on matrix files having the default extension **.MAT** in their file names.

Each matrix file always contains one matrix of real elements in double precision. In addition to the matrix elements, the labels of matrix rows and columns are saved as strings of 8 characters. Default row and column labels are 1,2,3, ..., but when e.g. a part of a Survo data set is saved in a matrix file, the names of observations and variables are used as labels.

The use of row and column labels greatly simplifies reading matrices and results of **MAT** operations because the labels are automatically moved and adjusted according to the nature of the operation. For example, when a matrix is transposed or inverted, its row and column labels are interchanged. This kind of a 'label algebra' guarantees resulting matrices with feasible labels.

Also the internal name of the matrix will be transformed by **MAT** operations to a form corresponding to the current contents of the matrix file. For example, if two matrices with internal names **A+B** and **C+D** are multiplied, the result will have **(A+B)*(C+D)** as its internal name.

The maximum size of a matrix is 8192 elements. Then, for example, square matrices up to dimensions 90*90 can be processed, provided that enough memory space is available for Survo. Space is allocated dynamically for all matrices during the run. Even larger matrices can be maintained in a partitioned form as supermatrices having their (super)elements in matrix files.



The **MAT** operations do not print their numerical results in the edit field. A separate **MAT LOAD** operation is used for this purpose. However, a **MAT** operation usually gives comments on the operation line after the operation has

been completed.

For example, assume that we have two 20*20 matrices (matrix files) A and B with internal names $X*X'$ and $Y*Y'$ and we want to compute C as a sum of matrices A and B. This is carried out by activating

```
MAT C=A+B .
```

After activation, the operation line will be changed into a form

```
MAT C=A+B / *C~X*X'+Y*Y' S20*20
```

where the comment tells that the result is a symmetric 20*20 matrix in the matrix file C with an internal name $X*X'+Y*Y'$.

In the preceding example, the actual matrix files were A.MAT, B.MAT and C.MAT on the default data path given on the header line of the edit field. To override this assumption, complete pathnames must be supplied. For example,

```
MAT D:C=INV(C:\E\D\CORR.M)
```

inverts matrix CORR.M on the path C:\E\D and saves the inverted matrix to matrix file C.MAT on disk D:

To speed up matrix operations, it is profitable to use a hard disk or a ram-disk for matrix files.

10.1 Matrices in the edit field

Although the most commonly employed matrices are taken from data files or are results from various Survo operations, it is also useful to enter matrices from the edit field.

When writing a matrix in the edit field, two alternative representations are available. In the first representation, the matrix is given without labels and comments. In the second one, both row and column labels as well as a free verbal description of the matrix can be supplied.

In the next example, matrix A is given without labels and matrix CONS with them.

```
19 1 SURVO 84C EDITOR Tue Jun 09 15:37:22 1987 C:\P\MAT\ 120 80 0
1 *
2 *MATRIX A ///
3 * 12.7 32.5 10.0 9.5
4 * 9.8 -7.0 11.9 28.1
5 * 28.3 -11.8 5.7 3.0
6 *
7 *MATRIX CONS
8 * Consumption per inhabitant
9 *CONS
10 */// Coffee Tea Beer Wine
11 *Finland 12.5 0.15 54.7 7.4
12 *Sweden 12.9 0.30 58.3 7.9
13 *Denmark 11.8 0.41 113.9 10.4
14 *
15 *MAT SAVE A
16 *MAT SAVE CONS AS B
17 *
```

Both matrices have 3 rows and 4 columns. The first one is A, given on lines

2-5. On line 2, we have the keyword **MATRIX**, (file) name of the matrix (**A**), and finally **///**. These three slashes tell that the elements of the matrix will be found on the next consecutive edit lines without any row or column labels. The elements are in this case on lines 3-5. An empty line (6) terminates the matrix.

Matrix **CONS** is defined on lines 7-13. Now, the **MATRIX** line gives the name of the matrix only. Since **///** is missing, the elements of the matrix will not start from the next line but from the first line having **///** in the beginning (as line 10 here). The lines between the **MATRIX** line and the **///** line contain free text e.g. about the origin of the matrix. The last of these comment lines (9) gives the internal name of the matrix. If the comment lines are missing (i.e. **///** is on the line immediately after the **MATRIX** line), the internal name will be same as the **MATRIX** name (**CONS**).

The **///** line gives the column labels and the elements are listed on the next consecutive lines. Each of those lines is preceded by a row label. The row and column labels must be contiguous strings (without spaces), and their maximum length is 8 characters. The matrix is terminated by an empty line (14).

The **MAT** operations do not directly employ matrices given in the edit field. They must be first saved in matrix files by a **MAT SAVE** operation. In our example, the **MAT SAVE** operations on lines 15 and 16 save matrix **A** to the matrix file **A.MAT** and matrix **CONS** to the matrix file **B.MAT** on the current Survo data disk (path).

Due to the edit line 9, the internal name of the matrix in the matrix file **B** will be **CONS**.

The stored matrices (like any matrices obtained as results of **MAT** operations) are loaded to the edit field by a **MAT LOAD** operation as follows:

```

21 1 SURVO 84C EDITOR Tue Jun 09 16:22:40 1987 C:\P\MAT\ 120 80 0
17 *
18 *MAT LOAD A,19
19 *MATRIX A
20 *///          1          2          3          4
21 * 1          12.7000  32.5000  10.0000  9.5000
22 * 2           9.8000  -7.0000  11.9000  28.1000
23 * 3          28.3000 -11.8000  5.7000  3.0000
24 *
25 *MAT LOAD B,###.##,26
26 *MATRIX B
27 * Consumption per inhabitant
28 *CONS
29 *///      Coffee      Tea      Beer      Wine
30 *Finland  12.50      0.15  54.70      7.40
31 *Sweden   12.90      0.30  58.30      7.90
32 *Denmark  11.80      0.41 113.90     10.40
33 *

```

MAT LOAD A,19 on line 18 has loaded matrix **A** and written it in the edit field from line 19 onwards. Although the original matrix was given without labels, rows and columns are now equipped with integer labels.

The numerical accuracy and format of elements is selected automatically on the basis of the numerical structure of the matrix in question and according to

the general accuracy parameter of the system file SURVO.APU. The output format can also be selected by an image parameter of the form ###.## as seen in the latter MAT LOAD operation.

Because an unavoidable contradiction exists between the numerical accuracy requirements and the readability of matrices, Survo maintains matrices in full (double) precision in matrix files but permits the user to load any matrix easily to the edit field for visual inspection.

As an indication of how the matrix interpreter maintains names and labels of matrices, we consider the following simple chain of MAT operations:

```

14 1 SURVO 84C EDITOR Tue Jun 09 16:48:16 1987 C:\P\MAT\ 120 80 0
33 *
34 *MAT S=B' / *S~CONS' 4*3
35 *MAT S=S*B / *S~CONS'*CONS S4*4
36 *MAT LOAD S,38
37 *
38 *MATRIX S
39 *CONS'*CONS
40 */// Coffee Tea Beer Wine
41 *Coffee 461.90 10.58 2779.84 317.13
42 *Tea 10.58 0.28 72.39 7.74
43 *Beer 2779.84 72.39 19364.19 2049.91
44 *Wine 317.13 7.74 2049.91 225.33
45 *

```

The matrix CONS in file B has been transposed and saved as matrix S by the MAT S=B' operation on line 34. Then the product of matrices S (CONS') and B (CONS) is computed and saved back to S by MAT S=S*B on line 35. Finally, S is loaded by MAT LOAD on line 36.

See how the matrix interpreter has identified the results of each operation and how the labels and names are upheld. All the text produced by MAT operations is displayed above with a gray shading.

The three MAT operations above could have been executed by one activation (by **F2:PREFIX** and **ESC**) as a sequence of Survo operations. In chains of MAT operations, this procedure is exceptionally profitable because the matrix interpreter is able to perform adjacent MAT operations on its own without the control of the Survo Editor. This speeds up matrix programs. A still more effective procedure is to save a MAT chain to a text file and execute it by MATRUN.

Since matrix products of the form $A'A$ or $A'A'$ are common in many applications, special MAT operations are available for them. The S matrix in the example could have been computed directly by **MAT S=MTM(B)**.

Matrices with symbolic elements

When scalars are encountered, the MAT operations are able to use constants and expressions given according to the style of editorial computing.

The elements of matrices entered in the edit field can be given as expressions of symbolic constants written in the edit field. In the next display, a 4*4

correlation matrix R is entered with symbolic expressions as its elements:

```

14 1 SURVO 84C EDITOR Tue Jun 09 17:13:50 1987 C:\P\MAT\ 120 80 0
1 *
2 * r=sqrt(1/n) n=4
3 *
4 *MATRIX R ///
5 * 1 r r^2 r^3
6 * r 1 r r^2
7 * r^2 r 1 r
8 * r^3 r^2 r 1
9 *
10 *MAT SAVE R
11 *
12 *MAT LOAD R,13
13 *MATRIX R
14 *///
15 * 1 1.000000 0.500000 0.250000 0.125000
16 * 2 0.500000 1.000000 0.500000 0.250000
17 * 3 0.250000 0.500000 1.000000 0.500000
18 * 4 0.125000 0.250000 0.500000 1.000000
19 *

```

R is saved in a numerical form only. If r is altered, MAT SAVE R must be activated again.

10.2 Moving data to matrix files

It is usually easy to transform any tabular representation to the form of a Survo matrix in the edit field by using standard tools for text and table management. On the other hand, many of the tabular results of Survo operations are readily saved as matrix files (with default extension `.M`). Thus, it is simple to give information in a proper form to the matrix interpreter.

The only case deserving special attention here is how data files and parts of them are converted to matrix files. Of course, any part of a data file can be loaded to the edit field by a FILE LOAD operation, then formatted to the MATRIX representation and finally saved by MAT SAVE as described above. In some cases, this technique is not adequate, however, since numerical precision may be lost in double conversion from numeric to alpha and then back to numeric.

An accurate method is to use the MAT SAVE DATA operation. It moves a selected part of any Survo data file directly to a matrix file. The general syntax is

```
MAT SAVE DATA <data_file> TO <matrix_file>
```

and the standard VARS, MASK, IND, CASES and SELECT specifications may be used for selection of variables and observations.

The active variables will appear as columns of the matrix and the observations as rows. The column labels will be the names of the variables. The row labels are, by default, indices 1,2,3, ... but by selecting a string variable as the first active variable, the values of this variable will be row labels. If other string variables exist among active variables, they are treated numerically and appear as columns like all numerical variables.

In the next example, 'Height', 'Weight', and the first day events of the Decathlon data DECA for competitors having 'Height' over 190 cm are copied to a matrix file X. When the MASK specification is used, the variables will appear as columns in the alphabetic order of their mask characters. By keeping the string variable 'Name' as the first variable (mask 'A'), names of competitors will appear as row labels. The result is verified by loading X to the edit field by MAT LOAD.

```

13 1 SURVO 84C EDITOR Tue Jun 09 19:11:09 1987 C:\P\MAT\ 120 80 0
1 *
2 *MASK=A-CCCC-----BB IND=Height,191,210
3 *MAT SAVE DATA DECA TO X
4 *
5 *MAT LOAD X,6
6 *MATRIX X
7 *///
8 *Height Weight 100m L_jump Shot_put Hi_jump 400m
9 *Hedmark 195 90 853 853 814 769 833
10 *Le_Roy 191 90 879 951 799 779 838
11 *Zeilbaue 192 84 826 931 793 865 875
12 *Zigert 198 105 879 840 924 857 788
13 *Kiseljev 191 91 879 869 792 891 784
14 *Linkmann 195 100 905 861 763 788 792
15 *Thiemig 194 94 905 774 751 804 889
16 *Tselnoko 191 94 853 832 759 725 842
17 *Jachmien 192 90 756 806 774 900 762
18 *Dzhurov 194 93 747 798 835 689 792
18 *

```

Conversely, it is possible to transform any matrix file to a Survo data file by a FILE SAVE MAT <matrix_file> TO <new_data_file> .

Thus, it is simple to move between matrix and data file representations.

10.3 MAT operations

Various MAT operations are now described concisely. More extensive examples are given later.

Basic operations

MAT REM <text>

no operation, for comments etc.; useful in matrix chains.

MAT DIM A

writes the dimensions of matrix A as a comment on the same line in the form / * rowA=20 colA=8 .

If A is diagonal, also the rank of A (# of nonzero diagonal elements) is given in the form rankA=5.

MAT C=ZER(10,3)

null matrix 10*3

MAT C=IDN(10,10)

identity matrix 10*10

MAT C=IDN(10,10,0.3)

0.3*IDN(10,10)

```

MAT C=CON(10,3)
      10*3 matrix with all elements =1
MAT C=CON(10,3,0.3)
      10*3 matrix with all elements =0.3
MAT C=TRI(10,10,0.3)
      upper triangular matrix with non-zero elements =0.3
MAT C=A
      copies A to C.
MAT C=A'
      transpose of A
MAT C=A+B
MAT C=A-B
MAT C=A*B
MAT C=MMT(A)
      C=A*A'
MAT C=MTM(A)
      C=A'*A
MAT C=MMT2(A,B)
      C=A*B'
MAT C=MTM2(A,B)
      C=A'*B
MAT C=INV(A)
      inverse of A
MAT C=INV(A,det)
      inverse and determinant of A
      Determinant is given in the form / * det=1.0045 .
      See also MAT SOLVE.
MAT C=DINV(A)
      makes a 'generalized inverse' of a diagonal matrix A
      by inverting the non-zero diagonal elements only.
      See Singular value decomposition.
MAT C=DINV(A,eps)
      uses value eps as upper limit for 'zero'.
      Default value for eps is 1e-15.
      (Use MAT DIM C to see the rank.)
MAT TRACE A
      trace of A in the form / * trA=30.7
MAT C=SUM(A)
      row vector of the sums of the columns
MAT C=MAX(A)
      row vector of the column maxima
MAT C=MIN(A)
      row vector of the column minima

```

MAT C=CENTER(A)
 centers the columns of A by subtracting means.
 Means of columns are saved in **MEAN.MAT** as a by-product.

MAT C=NRM(A)
 rescales the columns of A to the length=1.
 Lengths of A columns are saved in **NORM.MAT**.

MAT C=DV(A)
 makes a diagonal matrix of a column vector A.

MAT C=VD(A)
 forms a column vector of the diagonal of A.

MAT C=VEC(A)
 forms a single column vector C of all A columns.

MAT C=VEC(A,k)
 forms a k-column matrix C of the elements of A.

MAT C=PERM(A,P)
 If P is a column vector ($m \times 1$) consisting of numbers
 1,2, ..., m in any order, the rows of A will be
 permuted according to P.
 If P is a row vector, the columns of A are permuted.

MAT C=P(A,k)
 Pivotal operation with a pivot A(k,k)

MAT C=P(A,h:k)
 Pivotal operation with pivots A(h,h), ..., A(k,k).
 If A is a 10×10 matrix, **MAT=C=P(A,1:10)** inverts A.

Element by element transformations

MAT TRANSFORM <matrix file> **BY** <function of X#>
 transforms each element of the matrix according
 to <function of X#>. For example, to take natural
 logarithms of all elements of A, activate
MAT TRANSFORM A BY LOG(X#).

MAT TRANSFORM <X_matrix> **BY** <Y_matrix> **AND** <f(X#,Y#)>
 transforms each element X# of <X_matrix> by
 function f(X#,Y#) where Y# is the
 corresponding element of <Y_matrix>.
 For example, to divide each element of A by the
 corresponding element of B, activate
MAT TRANSFORM A BY B AND X#/Y#.

Besides current elements (X#,Y#), both forms of MAT TRANSFORM permit a reference to current row and column indices by I# and J#, respectively. For example, a 10*10 matrix **R** of the form

$$\begin{matrix} 1 & r & r^2 & \dots & r^9 \\ r & 1 & r & \dots & r^8 \\ r^2 & r & 1 & \dots & r^7 \\ \dots & \dots & \dots & \dots & \dots \\ r^9 & r^8 & r^7 & \dots & 1 \end{matrix}$$

where $r=0.9$, is generated as follows:

```

19 1 SURVO 84C EDITOR Wed Jun 10 17:44:49 1987 C:\P\MAT\ 120 80 0
1 *
2 *                               r=0.9
3 *MAT R=CON(10,10)
4 *MAT TRANSFORM R BY r^abs(I#-J#)
5 *
6 *MAT LOAD R, #.###, 7
7 *MATRIX R
8 *T(R_by_r^abs(I#-J#))
9 *///
10 * 1      1.000 0.900 0.810 0.729 0.656 0.590 0.531 0.478 0.430 0.387
11 * 2      0.900 1.000 0.900 0.810 0.729 0.656 0.590 0.531 0.478 0.430
12 * 3      0.810 0.900 1.000 0.900 0.810 0.729 0.656 0.590 0.531 0.478
13 * 4      0.729 0.810 0.900 1.000 0.900 0.810 0.729 0.656 0.590 0.531
14 * 5      0.656 0.729 0.810 0.900 1.000 0.900 0.810 0.729 0.656 0.590
15 * 6      0.590 0.656 0.729 0.810 0.900 1.000 0.900 0.810 0.729 0.656
16 * 7      0.531 0.590 0.656 0.729 0.810 0.900 1.000 0.900 0.810 0.729
17 * 8      0.478 0.531 0.590 0.656 0.729 0.810 0.900 1.000 0.900 0.810
18 * 9      0.430 0.478 0.531 0.590 0.656 0.729 0.810 0.900 1.000 0.900
19 * 10     0.387 0.430 0.478 0.531 0.590 0.656 0.729 0.810 0.900 1.000
20 *

```

Scalars in MAT operations

Both symbolic and numeric constants and expressions may appear in certain MAT operations. The values of the symbolic constants must be given in the same subfield. The symbolic constants have to be written in lowercase letters.

Examples:

```

18 1 SURVO 84C EDITOR Wed Jun 10 18:00:06 1987 C:\P\MAT\ 120 80 0
1 *
2 *m=10 n=5 x=k/100 coeff=2.5 k=3000
3 *MAT C=CON(m,m+n,x)
4 *MAT DIM C /* rowC=10 colC=15
5 *MAT C=coeff*C / *C~coeff*CON 10*15
6 *MAT C=(sqrt(k))*C / *C~(sqrt(k))*coeff*CON 10*15
7 *
8 *MAT LOAD C(1,1),9 / loading a single element
9 *MATRIX C
10 *(sqrt(k))*coeff*CON
11 *///
12 * 1      4107.919
13 *

```

Submatrices

A submatrix of A consisting of rows from $r1$ to $r2$ and of columns from $c1$ to $c2$ is denoted in **MAT** operations by $A(r1:r2, c1:c2)$. If all the rows are included, the submatrix is denoted by $A(*, c1:c2)$, and if all the columns are included, the submatrix is denoted by $A(r1:r2, *)$. The constants $r1, r2, c1, c2$ can also be expressions of symbolic constants.

```
MAT C=A(a1:a2,b1:b2)
           C is a submatrix of A
MAT C(a1,b1)=A
           A is copied partially over C so that
           C(a1,b1) will be A(1,1).
MAT LOAD A(a1:a2,b1:b2),L
           loads a submatrix of A to the edit field
           from line L onwards.
MAT C(3,5)=3.2345
MAT C(1,2)=k
           giving single elements
MAT C(0,2)="sum"
           Label of column 2 will be sum.
MAT C(5,0)="Factor5"
           Label of column 5 will be Factor5.
```

Matrix names and labels

```
MAT NAME A AS Factor_matrix
           changes the internal name of A to "Factor_matrix".
MAT C!=A+B
           Due to !, the internal name of C will be "C"
           regardless of the names of A and B.
MAT RLABELS FROM A TO B
           Row labels are copied from A to B.
MAT CLABELS FROM A TO B
           Column labels are copied from A to B.
MAT CLABELS "COMP" TO B
           Columns of B are labelled by COMP1, COMP2, ...
```

Deleting matrix files

MAT KILL A,B,C

deletes matrix files A.MAT, B.MAT, C.MAT on the current Survo data path.

MAT KILL !*

deletes all matrix files with names starting by '!'.

Matrix decompositions

MAT C=CHOL(A)

forms the Cholesky decomposition $A=CC'$ of a positive definite matrix A, where C is a lower triangular matrix.

MAT GRAM-SCHMIDT DECOMPOSITION OF A TO S,U

finds the decomposition $A=S*U$ where the columns of S are orthonormal and U is upper triangular. The columns of A must be linearly independent.

If the columns of A are linearly dependent, the process is interrupted when the first dependency is found and an error message is displayed. Furthermore, the coefficients indicating linear dependency are saved in U as a column vector. The user may check the dependency of columns by computing $A*U$ which should be 0.

The accuracy in testing linear dependence may be controlled by an extra parameter, i.e. by writing

MAT GRAM-SCHMIDT DECOMPOSITION OF A TO S,U,eps .

The default value for eps is $1e-15$.

MAT SPECTRAL DECOMPOSITION OF A TO S,L

forms the spectral decomposition $A=S*L*S'$ of a symmetric matrix A where S is the orthogonal matrix of eigenvectors (as columns) and L is the diagonal matrix of eigenvalues (saved as a column vector).

MAT SINGULARVALUE DECOMPOSITION OF A TO U,Q,V

forms the singular value decomposition of A ($m*n, m \geq n$) $A=U*Q*V'$ where U is $m*n$ and $U'U=I$, V is $n*n$ and $V'V=I$ and Q ($n*n$) is diagonal and consists of singular values. To save space, Q will be saved as a column vector. This decomposition has several applications.

For example, $X=V*DINV(diag(Q))*U'$ is a generalized inverse of A with properties $AXA=A, XAX=X, (AX)'=AX, (XA)'=XA$.

Solving linear equations

`MAT SOLVE X FROM A*X=B`

where A is $m \times n$, $m \geq n$, $r(A) = n$ and B is $m \times k$,

yields the solution X of linear equations (when $m=n$) or the least squares solution X (when $m > n$).

The algorithm is automatically selected according to the nature of A :

If A is diagonal, solution is trivial,

else if A is triangular, straight back-substitution is used,

else if A is symmetric, 'choldet1' and 'cholsol1' is used,

else (when $m \geq n$) 'Ortholin1' is used.

If the columns of A are linearly dependent, an error message will be displayed. The tolerance limit for 'non-zero' entities is $1e-15$. In this case, the singular value decomposition may be used.

Reference: Wilkinson-Reinsch: *Handbook for Automatic Computation*,
Vol.II, Linear Algebra.

10.4 Examples

From the basic set of MAT operations, various matrix programs (matrix chains) can be generated by entering them as an operation sequence. Such a program is activated from the first operation by **F2:PREFIX**, **ESC** (see 2.7). This also speeds up the process since the matrix interpreter is able to perform MAT operations without giving the control to the editor between them.

It is easy to save a sequence of MAT operations to a text file and apply it in a MATRUN operation as described in 10.5.

10.4.1 Matrix inversion

The matrix chain below inverts the matrix A and checks the result by multiplication.

```

30 1 SURVO 84C EDITOR Mon Jun 15 19:01:32 1987 C:\P\MAT\ 120 80 0
1 *
2 *MATRIX A ///
3 * 2 8 9 5
4 * 0 4 5 6
5 * 3 2 5 0
6 * 1 1 7 2
7 *
8 *MAT SAVE A
9 *MAT B=INV(A,det)
10 *MAT LOAD B,END+2
11 *MAT C=A*B
12 *MAT LOAD C,##.##### ,END+2_
13 *

```

If the cursor is moved to line 8 and the keys **F2:PREFIX** and **ESC** are pressed, the MAT operations on lines 8-12 will be automatically executed and thereafter the following display is obtained.

```

1 1 SURVO 84C EDITOR Mon Jun 15 19:02:40 1987 C:\P\MAT\ 120 80 0
7 *
8 *MAT SAVE A
9 *MAT B=INV(A,det) / *B~INV(A) det=-247 4*4
10 *MAT LOAD B,END+2
11 *MAT C=A*B / *C~A*INV(A) 4*4
12 *MAT LOAD C,##.##### ,END+2
13 *
14 *MATRIX B
15 *INV(A)
16 *///
17 * 1 -0.29960 0.36032 0.64372 -0.33198
18 * 2 0.26721 -0.18623 -0.14170 -0.10931
19 * 3 0.07287 -0.14170 -0.12955 0.24291
20 * 4 -0.23887 0.40891 0.20243 -0.12955
21 *
22 *MATRIX C
23 *A*INV(A)
24 *///
25 * 1 1.000000000 -0.000000000 0.000000000 0.000000000
26 * 2 0.000000000 1.000000000 0.000000000 0.000000000
27 * 3 0.000000000 -0.000000000 1.000000000 0.000000000
28 * 4 -0.000000000 -0.000000000 0.000000000 1.000000000
29 *

```

Note that due to parameter 'det' in the MAT INV operation, the value of the

determinant appears on the same line. By using END+2 in the MAT LOAD operations the results will be properly written on the free lines of the edit field. END as a line label refers always to the last non-empty line; thus END+2 leaves one empty line between consecutive matrices.

Use of END-type line labels in MAT chains for the results enables dynamic solutions without a need to change line labels when the dimensions of the matrices are changed. When using such matrix chains several times, the user must take care of keeping enough empty lines at the end of the edit field. When necessary, the older results should be erased by SCRATCH or CLEAR, for example.

10.4.2 Correlation matrix

The next matrix program computes a correlation matrix R from a given data matrix X. Normally, standard Survo operations like CORR are used for this purpose. If the original data set is in a data file, it can be moved to a matrix file X as follows:

```

30 1 SURVO 84C EDITOR Wed Jun 17 08:48:50 1987 C:\P\MAT\ 120 80 0
1 *
2 *MASK=A-A-A---A---AA
3 *MAT SAVE DATA DECA TO X
4 *
5 *MAT LOAD X(1:10,*),#####6_ / only 10 first rows loaded
6 *MATRIX X
7 *///      100m Shot_put   Discus   Height   Weight
8 *Skowrone  853      725     772     184     81
9 *Hedmark   853      814     855     195     90
10 *Le_Roy   879      799     819     191     90
11 *Zeilbaue 826      793     729     192     84
12 *Zigert   879      924     866     198     105
13 *Bennett  905      647     651     173     68
14 *Blinjaje 879      785     897     190     90
15 *Katus    853      772     748     184     81
16 *Berendse 804      795     801     189     89
17 *Gorbacho 853      815     762     186     87
18 *

```

The MAT SAVE DATA DECA TO X operation (on line 3) saves all observations and selected variables (see MASK on line 2) from DECA to matrix file X. As a check, the 10 first rows of X have been loaded to the edit field by a MAT LOAD operation (on line 5). The entire X matrix has 48 rows and 5 columns in this case.

The MAT chain for computing a correlation matrix R from X is presented as follows. The comments given by the chain are displayed here in gray shading.

```

1 1 SURVO 84C EDITOR Wed Jun 17 08:58:48 1987 C:\P\MAT\ 120 80 0
18 *
19 *   Computing correlations R from data matrix X
20 *
21 *MAT X=CENTER(X) / *X-CENTER(X) 48*5
22 *MAT X!=NRM(X) / *X-NRM(CENTER(X)) 48*5
23 *MAT R=MTM(X) / *R-X'*X S5*5
24 *MAT LOAD R,END+2
25 *
26 *MATRIX R
27 *X'*X
28 *///      100m Shot_put   Discus   Height   Weight
29 *100m      1.00000 -0.02795  0.01434 -0.11006 -0.08200
30 *Shot_put  -0.02795  1.00000  0.72733  0.61724  0.70826
31 *Discus    0.01434  0.72733  1.00000  0.58793  0.63541
32 *Height    -0.11006  0.61724  0.58793  1.00000  0.85217
33 *Weight    -0.08200  0.70826  0.63541  0.85217  1.00000
34 *

```

In this solution, the variables (columns) are first standardized by MAT operations CENTER and NRM. The correlation matrix R is then obtained from the normed data matrix X simply as the matrix product $R=X' * X$.

As by-products of the CENTER and NRM operations, the row vectors MEAN (column means) and NORM (column lengths) are obtained. By using these vectors, it is easy to expand the previous MAT chain to give the means and standard deviations of variables as well. This done in the next MAT chain.

```

17 1 SURVO 84C EDITOR Wed Jun 17 09:24:48 1987 C:\P\MAT\ 120 80 0
18 *
19 *   Means, standard deviations and correlations
20 *
21 *MAT X=CENTER(X)
22 *MAT X!=NRM(X)
23 *MAT R=MTM(X)
24 *MAT NAME MEAN AS Means
25 *MAT LOAD MEAN,END+2
26 *MAT DIM X
27 *MAT REM c=1/sqrt(rowX-1) normalizing constant
28 *MAT NORM=c*NORM
29 *MAT NAME NORM AS Standard_deviations
30 *MAT LOAD NORM,END+2
31 *MAT NAME R AS Correlations
32 *MAT LOAD R,END+2_
33 *

```

If the MAT chain is activated from line 21 (by **F2** and **ESC**), the following results are obtained:

```

1 1 SURVO 84C EDITOR Wed Jun 17 09:29:55 1987 C:\P\MAT\ 120 80 0
18 *
19 *      Means, standard deviations and correlations
20 *
21 *MAT X=CENTER(X) / *X-CENTER(X) 48*5
22 *MAT X!=NRM(X) / *X-NRM(CENTER(X)) 48*5
23 *MAT R=MTM(X) / *R-X'*X S5*5
24 *MAT NAME MEAN AS Means
25 *MAT LOAD MEAN,END+2
26 *MAT DIM X /* rowX=48 colX=5
27 *MAT REM c=1/sqrt(rowX-1) normalizing constant
28 *MAT NORM=c*NORM / *NORM~c*NORM(CENTER(X)) 1*5
29 *MAT NAME NORM AS Standard_deviations
30 *MAT LOAD NORM,END+2
31 *MAT NAME R AS Correlations
32 *MAT LOAD R,END+2
33 *
34 *MATRIX MEAN
35 *Means
36 *///      100m Shot_put Discus Height Weight
37 *Mean      828.1875 740.7708 747.4583 186.9583 85.5625
38 *
39 *MATRIX NORM
40 *Standard_deviations
41 *///      100m Shot_put Discus Height Weight
42 *Norm      59.30256 61.82757 62.28212 5.09049 6.84760
43 *
44 *MATRIX R
45 *Correlations
46 *///      100m Shot_put Discus Height Weight
47 *100m      1.00000 -0.02795 0.01434 -0.11006 -0.08200
48 *Shot_put -0.02795 1.00000 0.72733 0.61724 0.70826
49 *Discus    0.01434 0.72733 1.00000 0.58793 0.63541
50 *Height   -0.11006 0.61724 0.58793 1.00000 0.85217
51 *Weight   -0.08200 0.70826 0.63541 0.85217 1.00000
52 *

```

The Survo module CORR computes the same results starting directly from the data file (list, table) and produces matrix files CORR.M (for correlations) and MSN.M (jointly for means and standard deviations).

10.4.3 Partial correlations

The next matrix chain computes partial correlations of the first 3 variables (100m, Shot_put, Discus) with the remaining variables (Height, Weight) held constant. The starting point is the correlation matrix of these 5 variables obtained in the previous example.


```

1 1 SURVO 84C EDITOR Wed Jun 17 09:59:06 1987 C:\P\MAT\ 120 80 0
1 *
2 *   Partial correlations
3 *
4 *MAT R11!=R(1:3,1:3)
5 *MAT R12!=R(1:3,4:5)
6 *MAT R22!=R(4:5,4:5)
7 *MAT R22=INV(R22) / *R22~INV(R22) S2*2
8 *MAT S=R12*R22 / *S~R12*INV(R22) 3*2
9 *MAT R12=R12' / *R12~R12' 2*3
10 *MAT S=S*R12 / *S~R12*INV(R22)*R12' 3*3
11 *MAT S=R11-S / *S~R11-R12*INV(R22)*R12' S3*3
12 *MAT D!=VD(S) / *D~VD(R11-R12*INV(R22)*R12') 3*1
13 *MAT TRANSFORM D BY 1/sqrt(X#)
14 *MAT D!=DV(D) / *D~DV(T(D_by_1/sqrt(X#))) D3*3
15 *MAT S=D*S / *S~D*(R11-R12*INV(R22)*R12') 3*3
16 *MAT S=S*D / *S~D*(R11-R12*INV(R22)*R12')*D S3*3
17 *MAT KILL R11,R12,R22,D
18 *MAT LOAD S,END+2
19 *
20 *MATRIX S
21 *D*(R11-R12*INV(R22)*R12')*D
22 */// 100m Shot_put Discus
23 *100m 1.000000 0.045841 0.096124
24 *Shot_put 0.045841 1.000000 0.508144
25 *Discus 0.096124 0.508144 1.000000
26 *

```

The first task is to remove submatrices R11, R12 and R22 from the original correlation matrix R (lines 4-6). Then the partial covariance matrix $R11-R12*INV(R22)*R12'$ is computed (lines 7-11) and normalized to a correlation matrix (lines 12-16). Finally, work matrices are deleted (MAT KILL on line 17) and the resulting matrix of partial correlations loaded.

It is easy to extend this scheme to a more general case where, say, k last variables are to be held constant. The first lines of our MAT chain should then be replaced by

```

30 1 SURVO 84C EDITOR Wed Jun 17 10:30:03 1987 C:\P\MAT\ 120 80 0
1 *
2 *   Partial correlations
3 *
4 *Number of variables held constant k=2
5 *                               s=rowR-k
6 *MAT DIM R
7 *MAT R11!=R(1:s,1:s)
8 *MAT R12!=R(1:s,s+1:rowR)
9 *MAT R22!=R(s+1:rowR,s+1:rowR)
10 *MAT R22=INV(R22)
11 *MAT S=R12*R22

```

The MAT DIM R operation gives the dimensions of R as a comment on the same line in the form

```
/* rowR=5 colR=5
```

and then the correct matrices R11, R12 and R22 are copied from R according to the values of k and s given on lines 4,5.

If the variables to be held constant are not the last ones in R, the rows and columns of it can be easily permuted. Assume e.g. that we want to change the places of three first and two last variables. This is done by defining a permutation vector P and using MAT operations as follows:

```

1 1 SURVO 84C EDITOR Wed Jun 17 10:55:31 1987 C:\P\MAT\ 120 80 0
1 *
2 *MATRIX P ///
3 *3 4 5 1 2
4 *
5 *MAT SAVE P
6 *MAT S=PERM(R,P) / permuting columns
7 *MAT P=P' / *P~P' 5*1
8 *MAT S=PERM(S,P) / permuting rows
9 *MAT LOAD S,END+2
10 *
11 *MATRIX S
12 *PERM(PERM(X'*X))
13 */// Height Weight 100m Shot_put Discus
14 *Height 1.00000 0.85217 -0.11006 0.61724 0.58793
15 *Weight 0.85217 1.00000 -0.08200 0.70826 0.63541
16 *100m -0.11006 -0.08200 1.00000 -0.02795 0.01434
17 *Shot_put 0.61724 0.70826 -0.02795 1.00000 0.72733
18 *Discus 0.58793 0.63541 0.01434 0.72733 1.00000
19 *

```

10.4.4 Linear regression analysis by orthogonalization

We consider a linear regression model

$$y = b_1x_1 + b_2x_2 + \dots + b_mx_m + \varepsilon$$

where y is the regressand, x_1, x_2, \dots, x_m are regressors, ε is the error term, and b_1, b_2, \dots, b_m are unknown regression coefficients.

The regression coefficients will be estimated from a sample of n observations

$$\begin{array}{cccccc}
 y_1 & x_{11} & x_{12} & \dots & x_{1m} \\
 y_2 & x_{21} & x_{22} & \dots & x_{2m} \\
 \dots & \dots & \dots & \dots & \dots \\
 y_n & x_{n1} & x_{n2} & \dots & x_{nm}
 \end{array}$$

or

$$\mathbf{Y} = \mathbf{X}\mathbf{b}$$

where \mathbf{Y} is a column vector ($n \times 1$) and \mathbf{X} an $n \times m$ matrix. We denote the column vector ($m \times 1$) of regression coefficients by \mathbf{b} . Then the model can be represented in a matrix form

$$\mathbf{Y} = \mathbf{X}\mathbf{b} + \varepsilon$$

The least squares solution is obtained by minimizing

$$\mathbf{S} = (\mathbf{Y} - \mathbf{X}\mathbf{b})'(\mathbf{Y} - \mathbf{X}\mathbf{b})$$

with respect to \mathbf{b} . To get this solution, we orthogonalize \mathbf{X} by columns by a decomposition of the form

$$\mathbf{X} = \mathbf{U}\mathbf{R}$$

where \mathbf{U} is $n \times m$ with orthonormal columns, i.e.

$$\mathbf{U}'\mathbf{U} = \mathbf{I} \quad (m \times m)$$

and \mathbf{R} a $m \times m$ matrix. If the Gram-Schmidt decomposition is used for this purpose, \mathbf{R} will be an upper triangular matrix. Now $\mathbf{X}\mathbf{b}$ can be written in the form

$$\mathbf{Xb} = \mathbf{URb} = \mathbf{Uc},$$

where

$$\mathbf{c} = \mathbf{Rb}.$$

Then

$$\begin{aligned} \mathbf{S} &= (\mathbf{Y} - \mathbf{Xb})'(\mathbf{Y} - \mathbf{Xb}) = (\mathbf{Y} - \mathbf{Uc})'(y - \mathbf{Uc}) = (\mathbf{Y}' - \mathbf{c}'\mathbf{U}')(\mathbf{Y} - \mathbf{Uc}) \\ &= \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{Uc} - \mathbf{c}'\mathbf{U}'\mathbf{Y} + \mathbf{c}'\mathbf{U}'\mathbf{Uc}. \end{aligned}$$

Since $\mathbf{U}'\mathbf{U} = \mathbf{I}$ and $\mathbf{Y}'\mathbf{Uc} = (\mathbf{Y}'\mathbf{Uc})' = \mathbf{c}'\mathbf{U}'\mathbf{Y}$ (as a scalar product), we get

$$\begin{aligned} \mathbf{S} &= \mathbf{Y}'\mathbf{Y} - 2\mathbf{Y}'\mathbf{Uc} + \mathbf{c}'\mathbf{c} = \mathbf{c}'\mathbf{c} - 2\mathbf{Y}'\mathbf{Uc} + \mathbf{Y}'\mathbf{UU}'\mathbf{Y} - \mathbf{Y}'\mathbf{UU}'\mathbf{Y} + \mathbf{Y}'\mathbf{Y} \\ &= (\mathbf{c} - \mathbf{U}'\mathbf{Y})'(\mathbf{c} - \mathbf{U}'\mathbf{Y}) - \mathbf{Y}'\mathbf{UU}'\mathbf{Y} + \mathbf{Y}'\mathbf{Y}. \end{aligned}$$

Thus \mathbf{S} is minimized with respect to \mathbf{c} when

$$\mathbf{c} = \mathbf{U}'\mathbf{Y}$$

and the minimum value is

$$\mathbf{S} = \mathbf{Y}'\mathbf{Y} - \mathbf{Y}'\mathbf{UU}'\mathbf{Y} = \mathbf{Y}'(\mathbf{I} - \mathbf{UU}')\mathbf{Y}.$$

The estimate of \mathbf{b} is then obtained from

$$\mathbf{Rb} = \mathbf{c} = \mathbf{U}'\mathbf{Y}.$$

Since $\mathbf{X}'\mathbf{X} = (\mathbf{UR})'(\mathbf{UR}) = \mathbf{R}'\mathbf{U}'\mathbf{UR} = \mathbf{R}'\mathbf{R}$, the equation $\mathbf{Rb} = \mathbf{U}'\mathbf{Y}$ when multiplied from the left by \mathbf{R}' gives

$$\mathbf{R}'\mathbf{Rb} = \mathbf{R}'\mathbf{U}'\mathbf{Y}$$

or

$$\mathbf{X}'\mathbf{Xb} = \mathbf{X}'\mathbf{Y}$$

which is the regular representation for normal equations. However, orthogonalization of \mathbf{X} to form $\mathbf{X} = \mathbf{UR}$ and solving \mathbf{b} from $\mathbf{Rb} = \mathbf{U}'\mathbf{Y}$ is numerically a more stable procedure.

The regression coefficients are calculated by the following matrix chain:

```

1 1 SURVO 84C EDITOR Wed Jun 17 15:01:05 1987 C:\P\MAT\ 120 80 0
1 *
2 *MASK=AAAAAAA
3 *MAT SAVE DATA DECA TO Z
4 *
5 *   Linear regression analysis by orthogonalization
6 *   The data matrix Z must have the regressor Y as
7 *   its first column and regressors as the remaining columns.
8 *   Before computing regression coefficients,
9 *   this MAT chain removes Y from Z and replaces it by
10 *   the constant term 1.
11 *
12 *MAT DIM Z /* rowZ=48 colZ=6
13 *MAT X=CON(rowZ,colZ)
14 *MAT X(0,1)="Constant"
15 *MAT C=Z(*,2:colZ)
16 *MAT X(1,2)=C
17 *MAT NAME X AS X
18 *MAT Y!=Z(*,1)
19 *MAT GRAM-SCHMIDT DECOMPOSITION OF X TO U,R
20 *MAT C=U' /*C-GS(X)' 6*48
21 *MAT C=C*Y /*C-GS(X)'*Y 6*1
22 *MAT SOLVE B FROM R*B=C
23 *MAT NAME B AS Regression_coefficients
24 *MAT LOAD B,END+2
25 *
26 *MATRIX B
27 *Regression_coefficients
28 *///      Points
29 *Constant 3559.491
30 *100m      0.543
31 *L_jump   1.361
32 *Shot_put  1.171
33 *Hi_jump   0.922
34 *400m     1.327
35 *

```

Certain parts of the data file DECA have been copied to a matrix file by MAT SAVE DATA on line 3. In the MAT chain, the work matrices X and Y are created on lines 12-18. The regression coefficients are then computed by the MAT operations on lines 19-22. See also how automatic labelling technique of the matrix interpreter produces sensible designations for the output.

10.4.5 Principal components

In this and in next two examples, the singular value decomposition has a prominent role. Our solutions are based mainly on the suggestions in *Computational Methods of Data Analysis* (Wiley 1977) by J.M.Chambers.

The pertinent results of principal component analysis are obtained simply by making the singular value decomposition

$$X = \text{PSCORE} * Q * \text{PCOMP}'$$

for the data matrix X of standardized variables. Then PCOMP will be formed by the principal component loadings and PSCORE by principal component scores. Both are normalized afterwards by multiplying columns by corresponding singular values in the diagonal matrix Q. Squares of the singular values are the eigenvalues of the traditional analysis based on covariances or correlations.

The following matrix chain performs the analysis

```

29 1 SURVO 84C EDITOR Wed Jun 17 16:00:56 1987 C:\P\MAT\ 120 80 0
1 *
2 *MASK=A-AAAAAAAAA
3 *MAT SAVE DATA DECA TO X
4 *
5 *   Number of principal components f=4
6 *
7 *MAT DIM X
8 *MAT X=CENTER(X)
9 *MAT X!=NRM(X) / if this is omitted, analysis is based on covariances
10 *MAT SINGULAR_VALUE DECOMPOSITION OF X TO PSCORE,Q,PCOMP
11 *MAT Q=DV(Q)
12 *MAT PCOMP=PCOMP*Q
13 *MAT PSCORE=PSCORE*Q
14 *MAT PSCORE=PSCORE(*,1:f)
15 *MAT PSCORE!=(sqrt(rowX-1))*PSCORE
16 *MAT Q=VD(Q)
17 *MAT Q=Q'
18 *MAT NAME Q AS Singular_values
19 *MAT LOAD Q(1,1:f),END+2
20 *MAT NAME PCOMP AS Principal_component_loadings
21 *MAT LOAD PCOMP(*,1:f),END+2
22 *MAT NAME PSCORE AS Principal_component_scores
23 *MAT LOAD PSCORE(1:5,*),END+2_
24 *

```

and when activated, it yields the results:

```

1 1 SURVO 84C EDITOR Wed Jun 17 16:03:42 1987 C:\P\MAT\ 120 80 0
24 *
25 *MATRIX Q
26 *Singular_values
27 *///      1      2      3      4
28 *diag    1.613089 1.416973 1.098463 1.032982
29 *
30 *MATRIX PCOMP
31 *Principal_component_loadings
32 *///      1      2      3      4
33 *100m    -0.22937  0.81856 -0.16769  0.03552
34 *L_jump  0.01962  0.45105  0.71348 -0.21559
35 *Shot_put 0.78217  0.20376 -0.26706  0.13231
36 *Hi_jump  0.49058 -0.38403  0.29577 -0.06899
37 *400m    -0.63734  0.41117  0.08840  0.32727
38 *Hurdles  0.02610  0.64951  0.21171  0.07459
39 *Discus   0.84639  0.22277 -0.15054  0.04996
40 *Pole_vlt -0.26796  0.01509 -0.08554 -0.88911
41 *Javelin  0.25012 -0.24124  0.65422  0.15808
42 *1500m   -0.66255 -0.49634 -0.00452  0.25764
43 *
44 *MATRIX PSCORE
45 *Principal_component_scores
46 *///      1      2      3      4
47 *Skowrone 0.17650  1.17044  1.97901 -1.25294
48 *Hedmark  2.30347  1.50614  1.70021  0.79766
49 *Le_Roy   1.09613  2.69606  0.76093 -2.25493
50 *Zeilbaue 0.15535  1.07416  1.60437 -0.02347
51 *Zigert   2.80861  1.69121 -1.65466 -0.33083
52 * ....

```

10.4.6 Canonical correlations

In canonical analysis, the following procedure based on the singular value decomposition is superb when compared to the common practice of using correlation matrix as a starting point.

Our example is here formulated as a work scheme with instructions and enables various applications by slight modifications. Of course, Survo pro-

vides still more general approaches for canonical correlations (See CANON?).

```

12 1 SURVO 84C EDITOR Wed Jun 17 16:40:29 1987 C:.\D\ 100 100 0
1 *SAVE CANON2_
2 *
3 *
4 *Canonical correlations 2.7.1984/SM
5 *Move the cursor to line 7 and press F2 and ESC!
6 *
7 *GOTO 1,13
8 *Selecting data:
9 *# of variables in the first set >= # of variables in the second set
10 *
11 *MASK=A-XXXXX----- / mask for the first set of variables
12 * also 'Name' included to appear as row label
13 *MAT SAVE DATA DECA TO X / saves the first data set to X.MAT
14 *GOTO 1,19
15 *.....
16 *
17 *MASK=A-----YYYYY--- / mask for the second set of variables
18 *
19 *MAT SAVE DATA DECA TO Y / saves the second data set to Y.MAT
20 *GOTO 24,26
21 *.....
22 *
23 *

```

```

1 1 SURVO 84C EDITOR Wed Jun 17 16:44:03 1987 C:\P\MAT\ 100 100 0
24 * Canonical analysis of X,Y
25 *
26 *MAT X!=CENTER(X) / *X-CENTER(X) 48*5
27 *MAT Y!=CENTER(Y) / *Y-CENTER(Y) 48*5
28 *MAT GRAM-SCHMIDT DECOMPOSITION OF X TO QX,RX
29 *MAT GRAM-SCHMIDT DECOMPOSITION OF Y TO QY,RY
30 *MAT W=QX' / *W-GS(X)' 5*48
31 *MAT W=W*QY / *W-GS(X)*GS(Y) 5*5
32 *MAT SINGULARVALUE DECOMPOSITION OF W TO U,LAMBDA,V
33 *MAT SOLVE ALPHA FROM RX*ALPHA=U
34 *MAT SOLVE BETA FROM RY*BETA=V
35 *MAT ALPHA!=NRM(ALPHA) / *ALPHA-NRM((ALPHA_from_RX*ALPHA=U)) 5*5
36 *MAT BETA!=NRM(BETA) / *BETA-NRM((BETA_from_RY*BETA=V)) 5*5
37 *MAT LAMBDA!=LAMBDA' / *LAMBDA-Dsvd(GS(X)*GS(Y))' 1*5
38 *MAT NAME LAMBDA AS Canonical_correlations
39 *MAT LOAD LAMBDA,END+2
40 *MAT NAME ALPHA AS Loadings_for_the_first_set_of_variables
41 *MAT LOAD ALPHA,END+2
42 *MAT NAME BETA AS Loadings_for_the_second_set_of_variables
43 *MAT LOAD BETA,END+2
44 *_

```

```

1 1 SURVO 84C EDITOR Wed Jun 17 16:48:03 1987 C:\P\MAT\ 100 100 0
44 *
45 *MATRIX LAMBDA
46 *Canonical_correlations
47 *///      1      2      3      4      5
48 *sing.val 0.789434 0.528435 0.330558 0.282276 0.097224
49 *
50 *MATRIX ALPHA
51 *Loadings_for_the_first_set_of_variables
52 *///      1      2      3      4      5
53 *100m    -0.46312  0.71262  0.38229  0.30248  0.25744
54 *L_jump  -0.14033  0.42967 -0.62006 -0.45930 -0.29369
55 *Shot_put -0.74706 -0.39202 -0.23961  0.29950 -0.17059
56 *Hi_jump  -0.18051  0.10652 -0.07843 -0.07235  0.85925
57 *400m     0.41849 -0.37754 -0.63703  0.77628  0.28296
58 *
59 *MATRIX BETA
60 *Loadings_for_the_second_set_of_variables
61 *///      1      2      3      4      5
62 *Hurdles  -0.12002  0.39125 -0.75447  0.49786 -0.17282
63 *Discus   -0.89191 -0.68553 -0.21954  0.19432 -0.45813
64 *Pole_vlt 0.08373  0.15611  0.00235 -0.25740 -0.81418
65 *Javelin  0.14436  0.00044 -0.50617 -0.73429  0.06907
66 *1500m    0.40278 -0.59380 -0.35548  0.33007 -0.30427
67 *

```

10.4.7 Multiple discriminant analysis

This form of analysis can be considered a special case of the canonical analysis. The first set of variables consist of dummy variables (with values 0 and 1) indicating the correct group of the observation.

In the next example, the discriminant analysis has been performed for a small data set consisting of three groups (A,B,C) and three variables (X1,X2,X3).

The matrix chain assumes that the number of groups is not greater than the number of variables plus 1. In the opposite case, the number of the discriminant variables will be the same as the number of variables. Then matrix W must be transposed before the singular value decomposition and the results (DISCR and V) are interchanged.

```

16 6 SURVO 84C EDITOR Wed Jun 17 17:34:26 1987 C:\P\MAT\ 120 80 0
1 *
2 *MATRIX Z
3 *///  A  B  C  X1  X2  X3
4 *A1   1  0  0  2  2  3
5 *A2   1  0  0  1  2  3
6 *A3   1  0  0  2  1  1
7 *A4   1  0  0  1  1  2
8 *B1   0  1  0  5  5  5
9 *B2   0  1  0  3  2  2
10 *B3  0  1  0  4  3  4
11 *C1  0  0  1  7  0  7
12 *C2  0  0  1  8  1  5
13 *C3  0  0  1  4  2  3
14 *C4  0  0  1  4  0  7
15 *
16 *MAT SAVE Z
17 *MAT Y!=Z(*,1:3)
18 *MAT X!=Z(*,4:6)_
19 *
20 *.....
21 *

```

```

1 1 SURVO 84C EDITOR Wed Jun 17 17:35:00 1987 C:\P\MAT\ 120 80 0
21 *
22 *Number of discriminant variables discr=colY-1
23 *
24 *MAT DIM X /* rowX=11 colX=3
25 *MAT DIM Y /* rowY=11 colY=3
26 *MAT X!=CENTER(X) / *X-CENTER(X) 11*3
27 *MAT X!=NRM(X) / *X-NRM(X) 11*3
28 *MAT GRAM-SCHMIDT DECOMPOSITION OF X TO QX,RX
29 *MAT GRAM-SCHMIDT DECOMPOSITION OF Y TO QY,RY
30 *MAT W=QX' / *W-GS(X)' 3*11
31 *MAT W=W*QY / *W-GS(X)'*GS(Y) 3*3
32 *MAT SINGULAR_VALUE DECOMPOSITION OF W TO DISCR,LAMBDA,V
33 *MAT SOLVE ALPHA FROM RX*ALPHA=DISCR
34 *MAT SOLVE BETA FROM RY*BETA=V
35 *MAT L=DV(LAMBDA) / *L-DV(Dsvd(GS(X)'*GS(Y))) D3*3
36 *MAT BETA!=BETA*L / *BETA~(BETA_from_RY*BETA=V)*DV(Dsvd(GS(X)'*GS(Y)))
37 *MAT LAMBDA=LAMBDA' / *LAMBDA~Dsvd(GS(X)'*GS(Y))' 1*3
38 *MAT LAMBDA=LAMBDA(1,1:discr)
39 *MAT NAME LAMBDA AS Canonical_correlations
40 *MAT LOAD LAMBDA,END+2
41 *MAT ALPHA=ALPHA(*,1:discr)
42 *MAT NAME ALPHA AS Discriminant_loadings
43 *MAT LOAD ALPHA,END+2
44 *MAT BETA=BETA(*,1:discr)
45 *MAT NAME BETA AS Means_of_groups_in_discriminant_space
46 *MAT LOAD BETA,END+2
47 *MAT D=X*ALPHA / *D~X*Discriminant_loadings 11*2
48 *MAT NAME D AS Discriminant_scores
49 *MAT LOAD D,END+2
50 *_

```

```

1 1 SURVO 84C EDITOR Wed Jun 17 17:35:32 1987 C:\P\MAT\ 120 80 0
50 *_
51 *MATRIX LAMBDA
52 *Canonical_correlations
53 */// 1 2
54 *sing.val 0.873300 0.728827
55 *
56 *MATRIX ALPHA
57 *Discriminant_loadings
58 */// 1 2
59 *X1 -0.67245 0.42748
60 *X2 0.36653 0.92883
61 *X3 -0.26989 0.07356
62 *
63 *MATRIX BETA
64 *Means_of_groups_in_discriminant_space
65 */// 1 2
66 *A 0.25602 -0.19712
67 *B 0.11214 0.34643
68 *C -0.34013 -0.06271
69 *
70 *MATRIX D
71 *Discriminant_scores
72 */// 1 2
73 *A1 0.21812 -0.05539
74 *A2 0.31121 -0.11457
75 *A3 0.22226 -0.28551
76 *A4 0.27249 -0.33300
77 *B1 0.09788 0.76578
78 *B2 0.16790 -0.00789
79 *B3 0.07066 0.28141
80 *C1 -0.58198 -0.12628
81 *C2 -0.50775 0.11629
82 *C3 0.03194 0.06297
83 *C4 -0.30271 -0.30381
84 *

```


10.5 MATRUN operations

Survo matrix programs written in the edit field and consisting of several consecutive **MAT** operations can also be saved as text files and run later as **MATRUN** operations. When converting a **MAT** chain to a **MATRUN** operation, it is possible to specify a list of parameters including matrices and scalars with selected default values. This makes the **MATRUN** operations more generally applicable.

As an example, we show how the **MAT** chain computing partial correlations is altered into a more general **MATRUN** operation **PARTCORR**. In the next display, the **SAVEP** operation saves edit lines from 3 to 'E' (27) as a text file **PARTCORR.MTX** to the subdirectory **\M** of the current Survo system path.

```

27 1 SURVO 84C EDITOR Thu Jun 18 09:30:09 1987 C:\P\MAT\ 150 80 0
1 *
2 *SAVEP 3,E,\M\PARTCORR.MTX
3 */ MATRUN PARTCORR,R,k,RP
4 */ computes partial correlations RP from the correlation matrix R.
5 */ The last k variables in R will be held constant.
6 */ parameter default
7 */ input: correlation matrix %1=R
8 */ input: k =variables to be held constant %2=1
9 */ output: partial correlation matrix %3=RP
10 *MAT DIM %1
11 *MAT REM s=row%1-%2
12 *MAT &R11!=%1(1:s,1:s)
13 *MAT &R12!=%1(1:s,s+1:row%1)
14 *MAT &R22!=%1(s+1:row%1,s+1:row%1)
15 *MAT &R22=INV(&R22)
16 *MAT &S=&R12*&R22
17 *MAT &R12=&R12'
18 *MAT &S=&S*&R12
19 *MAT &S=&R11-&S
20 *MAT &D!=VD(&S)
21 *MAT TRANSFORM &D BY 1/sqrt(X#)
22 *MAT &D!=DV(&D)
23 *MAT %3=&D*&S
24 *MAT %3=%3*&D
25 *MAT KILL &*
26 *MAT NAME %3 AS Partial_correlations
27 EMAT LOAD %3,END+2
28 *

```

The sequence of **MAT** operations is almost identical with that given in our earlier example of a **MAT** chain for partial correlations. However, notations have been generalized in following ways. All the matrices and scalars now acting as **MATRUN** parameters are replaced by notations **%1**, **%2** etc. in the order they will appear in the parameter list. All the work matrices generated during the run are denoted by names having a **&** as the first character. This simplifies deleting of them and lessens a possible confusion with existing matrix files on the current data path.

In this **MATRUN** operation, we shall have three parameters with default values **%1=R**, **%2=1** and **%3=RP**. Thus **R.MAT** will be the default correlation matrix file to be used as an input matrix. The default number of last variables

to be held constant is $k=1$ and the default name for the resulting partial correlation matrix file is `RP.MAT`.

In front of the `MAT` operations there are comment lines (from 3 to 9) starting by `'/'`. These lines provide information for the user. Thus if `MATRUN PARTCORR?` is activated (see the next display), these comment lines will be shown temporarily.

```

17 1 SURVO 84C EDITOR Thu Jun 18 18:52:16 1987 C:\P\MAT\ 120 80 0
1 *
2 *MATRUN PARTCORR?
MATRUN PARTCORR,R,k,RP
computes partial correlations RP from the correlation matrix R.
The last k variables in R will be held constant.
parameter                default
input: correlation matrix  %1=R
input: k =variables to be held constant %2=1
output: partial correlation matrix %3=RP
10 *
Press any key!
12 *

```

We can now compute the same partial correlations as in the previous example by this `MATRUN PARTCORR` operation. To do everything as simply as possible, we employ first `CORR` for correlations:

```

25 1 SURVO 84C EDITOR Thu Jun 18 18:59:58 1987 C:\P\MAT\ 150 80 0
29 *
30 *MASK=--A-A---A---AA
31 *CORR DECA
32 *
33 *MATRUN PARTCORR CORR.M,2
34 *
35 *MATRIX RP
36 *Partial_correlations
37 *///          100m Shot_put  Discus
38 *100m        1.000000 0.045841 0.096124
39 *Shot_put    0.045841 1.000000 0.508144
40 *Discus      0.096124 0.508144 1.000000
41 *

```

Since the `CORR` operation saves the correlation matrix as a matrix file `CORR.M`, we have used this file as the input matrix for `MATRUN PARTCORR`.

All rules for making program files for the `MATRUN` mode are not given here. Those who are interested in creating more demanding applications on this basis are advised to study existing `MATRUN` files. They all are located in the `.\M` subdirectory and have the extension `.MTX` in their file names. Some of the `MAT` chains given here as examples have their counterparts in `MATRUN` operations. The `INDEX` field of the `.\M` directory lists current `MATRUN` operations:

```

1 1 SURVO 84C EDITOR Sun Jul 26 14:17:14 1992      D:\P2\MAT\ 100 100 0
1 *SAVE INDEX
2 *
3 *MATRUN programs in SURVO 84C
4 *Activate lines below to see instructions for these programs
5 *
6 *MATRUN PCOMP?      / Principal components from a data matrix
7 *MATRUN PCOMPR?    / Principal components from correlations
8 *MATRUN PFACT?     / Factor analysis: principal axes solution
9 *MATRUN SUM2?      / Sums of squares by rows and columns
10 *MATRUN FCOEFF?    / Factor score coefficients
11 *MATRUN FTCOEFF?   / Factor score coefficients (after oblique rotation)
12 *MATRUN TRANS?    / Transformation analysis (naive model)
13 *MATRUN SYMTRANS? / Symmetric transformation analysis
14 *MATRUN CANON2?   / Canonical correlations and variates
15 *MATRUN CANON3?   / + elimination of confounding variables
16 *MATRUN REGR?     / Linear regression by sing.val.decomposition
17 *MATRUN APPROX?   / Matrix approximation by lower rank
18 *MATRUN COMPRESS? / Optimal projection of a point set
19 *MATRUN DISCR?    / Multiple discriminant analysis
20 *MATRUN PARTCORR? / Partial correlations
21 *MATRUN CHI2?     / Chi-square statistics from a matrix of frequencies
22 *

```

10.6 Partitioned matrices



Although the current implementation of Survo matrix interpreter does not allow larger than 90*90 square matrices, this threshold can be overstepped by giving matrices in a partitioned form. Then there are, in principle, no limits for the dimensions of matrices to be processed.

A partitioned matrix is defined in the edit field as a supermatrix having standard matrices (matrix files) as its elements. The matrix interpreter has certain special operations for these supermatrices. Of course, all other MAT operations are available for the management of the superelements.

A supermatrix is described in the edit field in the form:

```

MATRIX A%           / % indicates a supermatrix
A11 A12 A13
A21 A22 A23

```

In this example, A% is a 2*3 supermatrix. Its true dimensions depend on the dimensions of its superelements. E.g. if each of the elements is a 10*5 matrix, the true dimensions of A% are 20 and 15.

When A% is used as an operand in supermatrix operations, each of its superelements A11, A12, ... must have been saved as a matrix file and the dimensions of the superelements must be mutually compatible.

If A% is used as a result matrix, it is not necessary that its superelements A11, A12, ... exist as matrix files. However, a description of A% must be given in the edit field so that the matrix interpreter can save the resulting superelements correctly.

The following MAT operations are available for supermatrices:

```

MAT C%=A%
MAT C%=A%' / MAT A%=A%' is not allowed
MAT C%=A%+B%
MAT C%=A%-B%
MAT C%=A%*B% / C% cannot be same as A% or B%
MAT C%=INV(A%) / diagonal superelements non-singular

```

As an application, we shall create a 300*300 matrix and invert it. Our matrix R% is like a correlation matrix with the same value $r=0.8$ for all correlations. R% is represented as a 6*6 supermatrix with 50*50 matrices as super-elements. In this case, all superelements are either of type A or B where A is a 50*50 correlation matrix (with $r=0.8$) and B is a 50*50 matrix with all elements equal to $r=0.8$.

```

1 1 SURVO 84C EDITOR Sun Jul 26 14:52:28 1992 D:\P2\MAT\ 100 100 0
1 *
2 * Inversion of a 300*300 matrix
3 *MATRIX R%
4 *A B B B B B
5 *B A B B B B
6 *B B A B B B
7 *B B B A B B
8 *B B B B A B
9 *B B B B B A
10 *
11 *MATRIX S%
12 *S11 S12 S13 S14 S15 S16
13 *S21 S22 S23 S24 S25 S26
14 *S31 S32 S33 S34 S35 S36
15 *S41 S42 S43 S44 S45 S46
16 *S51 S52 S53 S54 S55 S56
17 *S61 S62 S63 S64 S65 S66
18 *
19 * n=50 r=0.8
20 *MAT B=CON(n,n,r)
21 *MAT A=IDN(n,n,1-r)
22 *MAT A=A+B / *A-IDN+CON S50*50
23 *TIME Sun Jul 26 14:43:27 1992
24 *MAT S%=INV(R%)
25 *TIME Sun Jul 26 14:52:28 1992
26 *

```

The supermatrix R% is defined on lines 3-9 and its superelements being either A or B are computed on lines 20-22. On lines 11-17, another supermatrix S% is described. The MAT INV operation on line 24 has inverted R% and saved its superelements to matrix files S11, S12, ...

We have run the MAT chain on the lines 20-25 on the IBM PS/2 Model 70 and the TIME commands around the MAT INV operation tell that inversion of this 300*300 matrix took about 9 minutes. It should be noticed that this time also includes reading and writing of supermatrix elements on the disk. There is a lot of such traffic because, at one time, only three matrices will be held in the central memory. The method of inversion is simply a pivotal operation performed once for each diagonal superelement.

As a check, some parts of the inverted matrix have been loaded to the edit field. Due to the simple structure of R%, it is easy to compute its typical elements directly. Below, x is the common diagonal and y is the common off-diagonal element.

```

20 1 SURVO 84C EDITOR Sun Jul 26 15:22:31 1992          D:\P2\MAT\ 100 100 0
26 *
27 *MAT LOAD S11(1:3,1:3),##.#####28
28 *MATRIX S11
29 *INV(IDN+CON)
30 *///
31 * 1          1          2          3
32 * 1          4.983347210657008 -0.016652789342992 -0.016652789342992
33 * 2          -0.016652789342994  4.983347210657007 -0.016652789342994
34 * 3          -0.016652789342992 -0.016652789342992  4.983347210657008
35 *MAT LOAD S66(1:3,1:3),##.#####36
36 *MATRIX S66
37 *INV(IDN+CON)
38 *///
39 * 1          1          2          3
40 * 1          4.983347210657754 -0.016652789342245 -0.016652789342245
41 * 2          -0.016652789342245  4.983347210657754 -0.016652789342245
42 * 3          -0.016652789342245 -0.016652789342245  4.983347210657754
43 *.....
44 *
45 *n=50 r=0.8 N=6*n
46 *x=(1+(N-2)*r)/q q=1+(N-2)*r-(N-1)*r*r
47 *y=-r/q
48 *x=4.9833472106578
49 *y=-0.01665278934221

```

10.7 Operations on polynomials

Closely related to MAT operations are the POL operations working on polynomials with real or complex coefficients. A polynomial of degree n with complex coefficients

$$P(z)=A_0+iB_0+(A_1+iB_1)z+(A_2+iB_2)z^2+\dots+(A_n+iB_n)z^n$$

can be represented as a matrix

MATRIX P ///

A_0 B_0

A_1 B_1

A_2 B_2

.. ..

A_n B_n

and saved in a matrix file P.MAT by MAT SAVE P. If the coefficients are real, the second column may be omitted. Any matrix saved as a MAT file can be used as a polynomial in the POL operations. Eventual excessive columns 3,4, ... are not used.

The following POL operations are working on polynomials saved in matrix files:

POL P=P1+P2

POL P=P1-P2

POL P=P1*P2

POL Q=P1/P2
residual not obtained

POL Q(R)=P1/P2
R will be the residual, i.e. $P1=Q*P2+R$

POL D=DER(P)
 $D(x)=P'(x)$ (derivative of $P(x)$)

POL R=ROOTS(P)
Roots of algebraic equation $P(x)=0$

POL P=PRODUCT(R)
 $P(x)=(x-r1)(x-r2)...$ (inverse operation for ROOTS)

POL V=P(X)
V=values of polynomial P on components of vector X

The resulting polynomials will be saved in corresponding matrix files and their coefficients can be loaded to the edit field by MAT LOAD, for example.

The next display shows how POL operations are used together with MAT operations. We shall create two polynomials

$$P_1(x)=(x-2)(x-3)=6-5x+x^2$$

$$P_2(x)=(x^2+1)(x-2)=-2+x-2x^2+x^3$$

and compute

$$P(x)=P_1(x)P_2(x).$$

Finally, we solve the equation $P(x)=0$.

```

1 1 SURVO 84C EDITOR Fri Jun 19 12:38:30 1987 C:\P\MAT\ 100 100 0
1 *
2 *MATRIX P1 /// (x-2)(x-3)=[6,-5,1]
3 *6 -5 1
4 *
5 *MATRIX P2 /// (x^2+1)(x-2)=[-2,1,-2,1]
6 *-2 1 -2 1
7 *
8 *MAT SAVE P1
9 *MAT SAVE P2
10 *MAT P1=P1' / *P1~P1' 3*1
11 *MAT P2=P2' / *P2~P2' 4*1
12 *POL P=P1*P2
13 *POL R=ROOTS(P)
14 *MAT LOAD R,END+2
15 *
16 *MATRIX R
17 *Roots_of_P=0
18 */// real imag
19 * 1 2.00000 0.00000
20 * 2 3.00000 0.00000
21 * 3 0.00000 1.00000
22 * 4 2.00000 -0.00000
23 * 5 0.00000 -1.00000
24 *

```

10.8 Linear programming

SIMPLEX S, M1, M2, M3, L

solves a linear programming problem presented by the matrix file **S** with **M1+M2+M3** constraints and lists the results from line **L** (optional) onwards.

The ordinary simplex algorithm is used. The maximum size of the **S** matrix is 8192 elements. Thus, problems with 40 variables and 200 constraints, for example, can be solved.

The solution vector and the values of the **M1+M2** slack variables will be given as results. These vectors are also saved in matrix files **SIMPLEX.M** and **SLACK.M**, respectively.

The algorithm is a conversion of the Fortran routine given in "*Numerical Recipes*" by Press, Flannery, Teukolsky and Vetterling.

The problem to be solved is:

Maximize

$$Z=A(0,1)*X_1+A(0,2)*X_2+ \dots +A(0,N)*X_N$$

subject to the primary constraints

$$X_1 \geq 0, X_2 \geq 0, \dots, X_N \geq 0$$

and simultaneously subject to $M=M_1+M_2+M_3$ additional constraints,

M_1 of them of the form

$$A(I,1)*X_1+A(I,2)*X_2+ \dots +A(I,N)*X_N \leq B(I), B(I) \geq 0, I=1, \dots, M_1$$

M_2 of them of the form

$$A(I,1)*X_1+A(I,2)*X_2+ \dots +A(I,N)*X_N \geq B(I) \geq 0, I=M_1+1, \dots, M_1+M_2$$

and M_3 of them of the form

$$A(I,1)*X_1+A(I,2)*X_2+ \dots +A(I,N)*X_N = B(I) \geq 0, I=M_1+M_2+1, \dots, M$$

The matrix of coefficients **S** with **M+1** rows and **N+1** columns has the form

$$\begin{array}{cccccc} 0 & A(0,1) & A(0,2) & \dots & A(0,N) & \\ B(1) & -A(1,1) & -A(1,2) & \dots & -A(1,N) & \\ B(2) & -A(2,1) & -A(2,2) & \dots & -A(2,N) & \\ \dots & \dots & \dots & \dots & \dots & \\ B(M) & -A(M,1) & -A(M,2) & \dots & -A(M,N) & \end{array}$$

and it must be saved in a **MAT** file before **SIMPLEX** is activated.

Example 1:

Maximize $Z=X_1+X_2+3*X_3-0.5*X_4$ with all the X 's non-negative and

$$\text{also with } X_1+2*X_3 \leq 740$$

$$2*X_2-7*X_4 \leq 0$$

$$X_2-X_3+2*X_4 \geq 0.5$$

$$X_1+X_2+X_3+X_4 = 9$$

In this case, have $M_1=2$, $M_2=1$, $M_3=1$. This problem is described and solved by:

```
22 1 SURVO 84C EDITOR Tue Jul 28 10:14:12 1992 D:\P2\MAT\ 150 100 0
52 *
53 *MATRIX S
54 */// C X1 X2 X3 X4
55 *Z 0 1 1 3 -0.5
56 *Z1 740 -1 0 -2 0
57 *Z2 0 0 -2 0 7
58 *Z3 0.5 0 -1 1 -2
59 *Z4 9 -1 -1 -1 -1
60 *
61 *MAT SAVE S / Saving the matrix
62 *SIMPLEX S,2,1,1,CUR+2
63 *
64 *Simplex solution for Linear Programming problem S:
65 *Number of variables 4
66 *Number of '<' constraints 2
67 *Number of '>' constraints 1
68 *Number of '=' constraints 1
69 *Max.value=17.025 of Z obtained for:
70 * X1 0.0000
71 * X2 3.3250
72 * X3 4.7250
73 * X4 0.9500
74 *Values of slack variables:
75 * Z1 730.55
76 * Z2 0.0000
77 * Z3 0.0000
78 *
```


Example 2:

Solving a two-person zero-sum game:



| | | | | |
|-------|-------|-------|-------|-------|
| | B_1 | B_2 | B_3 | B_4 |
| A_1 | 3 | 6 | 1 | 4 |
| A_2 | 5 | 2 | 4 | 2 |
| A_3 | 1 | 4 | 3 | 5 |

```

22 1 SURVO 84C EDITOR Tue Jul 28 10:19:50 1992      D:\P2\MAT\ 150 100 0
80 *
81 *MATRIX A
82 *///      C      A1      A2      A3      V
83 *C        0      0      0      0      1
84 *B1       0      3      5      1     -1
85 *B2       0      6      2      4     -1
86 *B3       0      1      4      3     -1
87 *B4       0      4      2      5     -1
88 *V        1     -1     -1     -1      0
89 *
90 *MAT SAVE A
91 *SIMPLEX A,4,0,1,END+2 / gives the optimal mixed strategy for player A
92 *MAT B=A' / *B-A' 5*6
93 *MAT DIM B /* rowB=5 colB=6
94 *MAT B(1,colB)=-1
95 *SIMPLEX B,0,3,1,END+2 / gives the optimal mixed strategy for player B
96 *
97 *Simplex solution for Linear Programming problem A:
98 *Number of variables 5
99 *Number of '<' constraints 4
100 *Number of '>' constraints 0
101 *Number of '=' constraints 1
102 *Max.value=3.2500 of C obtained for:
103 * A1      0.1250
104 * A2      0.5000
105 * A3      0.3750
106 * V      3.2500
107 *Values of slack variables:
108 * B1      0.0000
109 * B2      0.0000
110 * B3      0.0000
111 * B4      0.1250
112 *
113 *Simplex solution for Linear Programming problem B:
114 *Number of variables 4
115 *Number of '<' constraints 0
116 *Number of '>' constraints 3
117 *Number of '=' constraints 1
118 *Max.value=-3.2500 of C obtained for:
119 * B1      0.0833
120 * B2      0.4167
121 * B3      0.5000
122 * B4      0.0000
123 * V      3.2500
124 *Values of slack variables:
125 * A1      0.0000
126 * A2      0.0000
127 * A3      0.0000
128 *

```

11. Control operations



DESKTOP?

Miscellaneous auxiliary Survo commands and operations are introduced in this section. Some of them are related to the control of the edit field and the screen. Also operations for code conversions and time control will be presented. Finally, possibilities of calling other programs and giving commands of the operating system during a Survo session will be discussed.



COLOR? CHECK?
COLX? STRDIST?
LOWLINE? SURVOPOINT?
Demo

11.1 Changing dimensions of the edit field

REDIM <# of lines> , <# of columns>

redimensions the edit field, provided that the new dimensions are not too small for the present contents of the field. *Max. # of columns is 252 and the maximum size of of the field is 65536 characters* (the control column and the shadow lines included). *Min. # of columns is 40.*



REDIM <# of lines> , <# of columns> , <# of shadow lines>

redimensions the field and changes the maximum number of shadow lines (lines with special display attributes). *Default is 20.*



INIT <# of lines> , <# of columns> , <# of shadow lines>

initializes the Survo system parameters, **clears** the edit field, redimensions the field, and (optionally) changes the maximum number of shadow lines. *Default is 20.*



INIT is faster than REDIM, but it does not preserve the current contents of the edit field.

11.2 Moving the cursor in the edit field

GOTO L1 , L2 , C

changes the display of the current edit field and the position of the cursor by showing L1 as the first visible line and placing the cursor to column C on line L2. GOTO is especially useful in automatic sequences of Survo operations when jumping from one work scheme to another.

The default value of L2 is L1 and that of C is the current column. As lines L1 and L2, both numeric and symbolic labels (referring to a character in the control column) can be employed. Furthermore, references like A-1, END+2 and CUR-5 are allowed. 'END' is the last nonempty line in the edit field and 'CUR' the current line (indicated by the cursor).

GOTO L1,L2,C1,C2

moves the cursor by showing the edit field with L1 as the first visible line, C1 as the first visible column and the cursor on line L2 and on column C2.

11.3 Time control

WAIT <time in seconds>

gives a pause. **WAIT** is useful in automatic sequences of operations.

TIME

gives the date and the time in the form

TIME Fri Jun 19 16:52:53 1987

on the current (**TIME**) line. **TIME** is useful in time measurement for sequences of Survo operations activated by **F2:PREFIX** **ESC**.

In sucros, a more accurate time recording statement {save time Wi} is available.

An alarm at a specific time of the day during a Survo session is set by the system parameter **aLarm** in the **SURVO.APU** file. This is done most easily by a sucro command **/ALARM hh:mm:ss**.

11.4 Selecting the data disk

DISK <path designation>

selects the disk or the path for Survo data and results.

Examples: **DISK B:** **DISK C:.\DATA**

Another way is to press key **F4:DISK** as many times as needed. Then the standard alternatives **A:**, **B:**, **C:.\D**, **D:**, etc. are obtained. The current choice is always displayed on the header line of the edit field.

A special list of alternatives for the **DISK** key can be defined by setting the lines 'last_disk' and 'paths' in the **SURVO.APU** file as follows:

last_disk=Z:

paths=C:\MYDATA\,C:\E\D\,D:\DAT2\,A:,B:

'Z' as 'last_disk' implies that the selection given by 'paths' should be used when **DISK** is pressed.

The default disk (and path) for Survo data is given by the 'edisk' parameter in the system file **SURVO.APU**.

11.5* Code conversions

For code conversions in Survo, binary files (usually of only 256 characters) are used.

For example, in sorting (operations `SORT`, `FILE SORT`) a specific file, `SORTCODE.BIN` is used to give the correct (alphabetic) order of ASCII characters.

Similarly in the `PRINT` operation, a 256 byte binary file can be specified (by a `- codes <conversion file>` control line) to imply a proper code conversion from the screen characters to the printer. Normally, the Survo device driver supplies a suitable `- codes` line automatically.

In a normal use, it is not necessary to alter these binary files. In special cases, the user can, however, maintain code files (and any other files as well) by the `CODES LOAD` and `CODES SAVE` operations.

`CODES LOAD <binary_file>`

loads the 256 first characters of `<binary_file>` to the edit field and lists them on 256 consecutive lines below the current line. (Thus before activating `CODES LOAD`, enlarge the edit field, if necessary, by `INIT 300,80`, for example.)

The list of the characters has the following form:

```
CODES LOAD .\SYS\SORTCODE.BIN
  0      0      .  .
  1      1      .  .
  2      2      .  .
  3      3      .  .
  ..     ..     .  .
 97     65     a  A
 98     66     b  B
 99     67     c  C
  ..     ..     .  .
```

where the second column gives the decimal code of the corresponding character in the file. For example, in `SORTCODE.BIN` which is used in alphabetic sorting, 97 is mapped to 65 (i.e. 'a' is like 'A'). Columns 3 and 4 echo printable characters in columns 1 and 2, respectively and are used as comments only.

`CODES SAVE <binary_file>`

is the converse operation of `CODES LOAD` and saves a list of codes of the previous type to a file.

Binary files of more than 256 bytes are edited by CODES LOAD/SAVE operations of the form

```
CODES <LOAD or SAVE>,n
```

where n is the record number 1,2,3, ... for records of size 256. For example,

```
CODES LOAD PITCH.BIN,1
```

is same as

```
CODES LOAD PITCH.BIN .
```

```
CONVERT L1,L2,<conversion_file>
```

converts all characters on lines L1-L2 in the current edit field according to <conversion_file>. In addition to conversion between different code systems, CONVERT can be used for various special tasks. For example, text on selected lines can be capitalized by creating first a conversion file that maps all lower case letters to the upper case and applying CONVERT with this file. In fact, file SORTCODE.BIN used in sorting is also suitable for that purpose.

11.6 Changing the inquiry system

In Survo, more than one inquiry system may be maintained and used parallelly, provided that different paths are selected for them. The user can create own solutions for information retrieval in various applications. Also parallel versions of the inquiry system in different languages may be maintained and consulted.

The path for the default inquiry system is given in the SURVO.APU file as `qpath=<default path>` (`qpath=C:\E\Q\EDQ`, for example).

The three last characters (EDQ in the example) refer to a common forepart of all file names of the inquiry system. The files are standard edit files.

The root file giving the keywords and their references to inquiry files has the name EDQ.EDT (or corresponding).

It is easy to convert edit fields to a form needed in inquiries. The rules are learnt simply by examining files in the default inquiry system.

```
QPATH <path designation>
```

changes the inquiry system by selecting another path for subsequent inquiries.

11.7 Screen colors

```
COLOR ALL
```

displays all possible display modes (color combinations) on the current (IBM compatible) screen. This is a help function for selecting colors for different shadow values.

```
COLOR #n
```

displays the color combination for value n. Values 0,1,2,...,255 are permitted.

COLOR <integer>

changes the current palette randomly sampling it from color combinations from 0 up to the selected integer. Default is 128, i.e. all the non-blinking alternatives.

The colors are selected most easily by using the `/SURVO-SETUP` macro which is a general tool for setting of Survo system parameters.

11.8 Calling other programs from Survo



CHILD <pathname of a program>

calls another program and runs it. If this other program terminates normally, the control returns to the current Survo session.

In this way, it is possible to use other systems and programs while staying in the Survo session. The other programs are loaded to the remaining free memory. If there are not enough memory, an error message is given and the normal Survo session continues. Activating **CHILD** without parameters calls another copy of Survo itself.

Intermediate jobs with Survo itself or with other programs can be done without losing the current situation. For example, more data can be imported to text files by using a telecommunication program.

The users having the Microsoft C compiler and the tools for Survo C programming can write more Survo modules in the edit field. They can also compile, link, and test programs immediately while working with Survo. The tools for making Survo modules in C are presented in Mustonen (1989).

Also the command processor `COMMAND.COM` of MS-DOS can be invoked by `CHILD COMMAND` (provided that it is found by the operating system). The work with the command processor must be terminated by the `EXIT` command (not a Survo command) and the Survo session will continue normally.

11.9 MS-DOS commands



MS-DOS commands are given during the Survo session by typing them on edit lines with a preceding prompt character '>>' and activating them as any Survo operation. For example,

```
>>COPY D:\DATA A: COPY D:\DATA A:
```

copies all files in the subdirectory `\\DATADATA` of the `DD:` disk to the disk `AA:`. Thus when working with Survo, it is easy to maintain files by using standard MS-DOS commands.

DESKTOP?

The `CCHILDCHILD` command and MS-DOS commands give a possibility of staying in Survo 'forever' and using other facilities of the computer as well. The only limitation is that the Survo Editor needs about 170 KB while staying resident.

11.10 DIR command

The **MS-DOS** commands, when activated from Survo, list their messages on the screen temporarily and give no results in the edit field. By using the redirection facilities of the operating system it is, of course, possible to save those results in text files and load this information in the edit field by **LOADP** or **SHOW**. Thus, for example,

```
>DIR D:\DATA>F:LIST
```

copies all the directory entries of **D:\DATA** to file **F:LIST** and this list can then be loaded in the edit field by

```
LOADP F:LIST .
```

To simplify this procedure, **DIR** is also available as a Survo command and then it lists the directory entries at once in the edit field below the current line. Thus

```
DIR D:\DATA
```

is a straightforward solution in the example above.

A still more refined tool is the **/DIR** *suco* made by Mikko Karpoja.

```
/DIR <path>
```

lists all files of **<path>** to the edit field, sorts them according to the type of the file and writes an appropriate viewing command (like **FILE SHOW**) in front of each file name.

12. Sucros

One of the strongest features of Survo is the possibility of creating new functions while using the system. The **tutorial mode** provides the simplest alternative for the user's own commands and operations. Any session with Survo can be recorded in this working mode. Even very complicated routines can be made available for other users on the basis of a model session worked out in the tutorial mode.

Originally, the tutorial mode was planned for teaching purposes and tutorials were made for telling how Survo is applied in various situations. During these first stages, it soon became apparent that the same principle and working method is also suitable in applications where several Survo operations had to be connected into one macro operation. Combined operations of this kind are more general than macros in the normal sense. Therefore we have introduced the term **sucro** for these super macros or Survo macros.

The range of sucros is wide. The smallest ones are like ordinary macros, but the largest sucros are specialized operations performing Survo operations conditionally and writing complete reports with text, tables and graphical illustrations automatically from the data at hand. By combining the power of the various Survo activities with conditional statements of the sucro language, extensive expert applications are created.

In all categories of problems, the sucros lessen the need for making actual computer programs. They are also more compact than programs. A typical sucro file for a sequence of statistical operations requires only a few kilobytes while a corresponding C program might take about 100 kilobytes.

To a great extent, the sucros are made simply by turning on the tutorial mode and starting to use Survo to solve one typical problem. Then all the keystrokes will be saved in a file selected by the user. On exit from the tutorial mode, the file will be closed and it is readily available for an automated repetition.

When making a new sucro in the tutorial mode, one can type text and activate both Survo operations and other sucros. The possibility of using other sucros as building blocks makes this technique very powerful.

Any sucro file can be edited later in Survo by the **TUTLOAD** and **TUTSAVE** operations. The **TUTLOAD** operation loads the compressed sucro file into the edit field, interprets it, and forms a readable list which could be called a sucro program. The user can edit the program, insert conditional statements and material from other sucros. After editing, the program is compressed and saved back in the file by the **TUTSAVE** operation.

Even in large applications, the files can be kept limited in size since the sucros can be chained.

Before describing how sucros are made, we give a short account on their usage.

12.1 Using ready-made sucros

When working with the Survo Editor, sucros are called normally by activating the name of the sucro with a '/' in front of the name and a list of parameters after the name. Thus a typical sucro call is

```
/SUCRO1 DATA3 ,VAR5 .
```

Since sucros are not regular Survo operations, their functions are not usually described in the inquiry system. However, a common practice is that a sucro gives information on itself by activating it without parameters or with a '?' as the only parameter. Thus

```
/SUCRO1 ?
```

should tell about the functions of sucro SUCRO1. This information will be written in the edit field and it may overwrite text below the current line.

Although sucros are technically alike, different types of them can be seen from the user's point of view.

Sucros as Survo operations

There are clear signs that more and more Survo operations will be available in an enhanced form as sucros. The sucro technique permits making of various combinations of standard Survo operations with a more user-friendly interface. Typical examples are the sucro /COMPARE for testing small samples (see 7.3) and the sucro /FACTOR for 'automatic' factor analysis (see 7.8.5).

The general style of working in sucros of this type is the same as in standard Survo operations. One minor difference is that in sucros emulating Survo operations, usually no line for the results is given. It is tacitly assumed that the sucro has a permission to erase all text below the command line and to write its intermediate information as well as final results from this line onwards.

Teaching programs

In sucros made for teaching of Survo functions or for teaching of various Survo applications, it is important to control the pace of the process. Since such teaching programs present true Survo activities, it should always be possible for the user to interrupt the run and continue on his/her own from the setup generated by the sucro so far.

Thus any sucro (made for teaching) can be stopped by pressing the key. A temporary interrupt is achieved by the space bar. Another press of the space bar makes the sucro to continue. To slow down the speed, the key is pressed (a couple of times). The original speed is resumed by pressing equally many times. Many teaching programs have their own internal speed control and there should be no urgent need for using the and keys.

Key sucros

Sucros having one-letter names are called key sucros because instead of normal activation (by /X for example) a sucro X can also be called without visible traces by the key combination `PREFIX M X`.

Typical macros of Survo have been made as key sucros. For example, sucro X changes the position of the current word by the next one on the same line. In such a task, it is important to invoke the action without typing any command.

A list of key sucros and other sucro tools is obtained by activating the sucro /SUCROS .

Keyboard sucros

The functions of selected keys can be redefined by the sucro technique. It is possible to alter the tasks of all regular typewriter keys.

When a keyboard sucro is activated, it normally gives a message (on the bottom line) about the changes of the keyboard functions. The keyboard sucro is terminated usually by the `#` key and then normal keyboard functions are resumed.

Standard keyboard sucros have been created, for example, for typing shadow characters (/S), for typing box graphics characters (/BOX), and for writing sucro code (/SUCRO).

A list of keyboard sucros and other sucro tools is obtained by activating the sucro /SUCROS .

Sucro paths

When a sucro is called without a pathname, it will be looked for in this order from directories

1. current data path
2. sucropath, given by `sucropath=<path>` in `SURVO.APU`
(e.g. `sucropath=.\SUCROS\`)
3. `.\S\` (various tools)
4. `.\TUT\` (teaching programs in English)

If the sucro does not exist in these directories, an error message '*Sucro <name> not found!*' will be displayed.

The second alternative (by sucropath) is intended for the user's own sucros.

12.2 Making a sucro: an example

We shall start by describing a typical small sucro. The same example is presented among the basic tutorials of the Survo system under the header "*How to make sucros*".

Assume that we have Survo in use and we wish to define a sucro that displays the current time by using the `TIME` command of Survo.

The definition is initiated by pressing the keys `PREFIX` and `T`. Then the following prompt will be displayed on the bottom line of the screen:

```
Sucro functions: S=Start definition, R=RUN ? _
```

If the `S` key is pressed, this prompt is replaced by another:

```
Define a sucro: Name of file ? _
```

We enter a name, say `TIME`, and press the `ENTER` key. Then without any further notice, the system opens a sucro file `TIME.TUT` on the current data path, the prompt disappears and the normal setup in the current edit field is resumed. From this on, all our actions (keystrokes) will be saved in the file `TIME.TUT`.

As a visible indication of staying in the tutorial mode, the level of the current sucro (usually 1) will be displayed in the second position of the header line of the screen.

Assume now that the cursor has been on line 3 and we press the `ENTER` key (thus coming to the start of line 4) and then we type the word `TIME` and press `SPACE`. The display is following:

```
1 6 1 SURVO 84C EDITOR Sun Oct 16 16:36:29 1988 D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *TIME _
5 *
6 *
7 *
```

We activate the `TIME` command by the `ESC` key and get

```
1 6 1 SURVO 84C EDITOR Sun Oct 16 16:36:35 1988 D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *TIME Sun Oct 16 16:36:35 1988
5 *
6 *
7 *
```

We move the cursor to the first position on line 4 by the `HOME` key and delete the first 16 characters by pressing the `DELETE` key 16 times:

```

1 6 1 SURVO 84C EDITOR Sun Oct 16 16:37:10 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:35 1988
5 *
6 *
7 *

```

Finally, we move the cursor 8 steps to the right (by the right arrow key) and erase the year (1988) with the `ctrl-END` key thus obtaining a plain time designation:

```

1 6 1 SURVO 84C EDITOR Sun Oct 16 16:37:22 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:35_
5 *
6 *
7 *

```

This is all we wish to have. Hence, we terminate the definition of the TIME sucro by pressing the keys `PREFIX` and `T`. Then the following prompt will appear on the bottom line:

Sucro functions:

E=End of definition, C=Control codes, R=Repeat ? _

By pressing the `E` key, the TIME.TUT file will be closed, the prompt will be deleted and the normal editorial working mode is resumed. Also '1' in the second position of the top line is cleared.

We can test the TIME sucro immediately by activating a /TIME command:

```

6 1 SURVO 84C EDITOR Sun Oct 16 16:38:49 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:29
5 */TIME_
6 *
7 *

```

This will repeat all our previous actions and gives a result in the following form:

```

6 1 SURVO 84C EDITOR Sun Oct 16 16:38:52 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:29
5 */TIME
6 *16:38:49_
7 *

```

Thus a sucro can be activated like any Survo command by using its name preceded by a slash '/'.

To edit a sucro, it is loaded in the edit field by a TUTLOAD command:

```

6 1 SURVO 84C EDITOR Sun Oct 16 16:40:38 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:49
5 */TIME
6 *16:38:29
7 *
8 *TUTLOAD TIME
9 *{R}
10 *TIME {act}{home}{del16}{r8}{erase}{end}
11 *

```

The TUTLOAD TIME command on line 8 gives a list of the sucro program on lines 9 and 10. Interpretation of the code is obvious in this case. The normal text typed during the definition (here 'TIME ') appears as such. The 'invisible' control keys have their counterparts as code words written in braces.

Here {R} on line 9 refers to the **ENTER** (RETURN) key, and codes {act}, {home}, and {erase} on line 10 correspond to the keys **ESC**, **HOME**, and **ctrl-END**, respectively. Above, {del16} means that **DELETE** is to be pressed 16 times and {r8} that the right arrow key is to be pressed 8 times. The sucro listing is terminated by {end}.

Thus the whole story with the 'cursor choreography' is unambiguously described in the listing. Now it is also possible to alter the program and make various enhancements.

For example, another sucro, say HOUR, could be derived from the preceding listing simply by editing it so that also the minutes and seconds are erased from the original output. This happens by altering {r8} above to {r2}, editing TUTLOAD TIME on line 8 into the form TUTSAVE HOUR and activating it:

```

6 1 SURVO 84C EDITOR Sun Oct 16 16:41:14 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:49
5 */TIME
6 *16:38:29
7 *
8 *TUTSAVE HOUR
9 *{R}
10 *TIME {act}{home}{del16}{r2}{erase}{end}
11 *

```

If now /HOUR is typed on line 11 and activated, it gives the current hour on line 12 as follows:

```

6 1 SURVO 84C EDITOR Sun Oct 16 16:42:44 1988      D:\P2\SUC\ 120 80 0
1 *
2 *
3 *
4 *16:36:49
5 */TIME
6 *16:38:29
7 *
8 *TUTSAVE HOUR
9 *{R}
10 *TIME {act}{home}{del16}{r2}{erase}{end}
11 */HOUR
12 *16_
13 *

```

As a more general enhancement, we shall make this sucro to write sentences like *'Now it is morning.'*, *'Now it is afternoon.'*, etc. on the basis of the current hour. For example, if the current hour is 16, the sucro should write *'Now it is afternoon.'*

This means that the sucro must be able to make decisions which depend on results obtained during the session or on the user's actions. For these decisions, the sucro gets information from the edit field or from the user (as answers to questions) and saves that information in special **sucro memory** which earlier was called a tutstack. The locations of the sucro memory are called W1, W2, ... For example, a number or a word touched by the cursor will be read to the memory location W1 by the command `{save word W1}`.

The HOUR sucro will now be extended by adding some conditional statements depending on `W1=hour`. The final form of our sucro with the name TIME2 could have the following listing:

```

14 1 SURVO 84C EDITOR Wed Jul 29 11:20:18 1992      D:\P2\SUC\ 120 80 0
8  *TUTSAVE TIME2_
9  *{R}
10 /
11 / The current hour, say 16, is displayed:
12 *TIME {act}{home}{del16}{r2}{erase}
13 /
14 / Move the cursor to the start of the line:
15 *{line start}
16 / The cursor is touching the hour (16)
17 /
18 / Put the hour to the sucro memory (W1=16):
19 *{save word W1}
20 /
21 / Write the beginning of the output sentence:
22 *Now it is
23 /
24 / Branch according to the hour:
25 - if W1 > 18 then goto evening
26 - if W1 > 12 then goto afternoon
27 - if W1 = 12 then goto noon
28 - if W1 > 5 then goto morning
29 /
30 / If none of above if statements is valid, we have 1 <= W1 <= 5 .
31 / We continue by typing ' night' and go to the end of the sucro:
32 * night{goto end}
33 /
34 / If W1 > 18, we have come to the label 'evening'.
35 / Text ' evening' will be typed and we go to the end of the sucro:
36 + evening: evening{goto end}
37 /
38 / Other alternatives are treated similarly:
39 + afternoon: afternoon{goto end}
40 + noon: noon{goto end}
41 + morning: morning
42 /
43 / All alternatives lead to label 'end' and character '.' is typed
44 / at the end of the sentence.
45 + end: .{end}
46 *

```

In the listing, the control column is used for indicating various special statements and lines. The lines having a '/' as the control character are for remarks and comments only. Lines of conditional statements have a '-' and lines starting with a label (to be referred by gotos) have a '+' in the control column.

The standard code lines have the default control character ‘*’.

All this information is saved by the TUTSAVE command in a condensed form. When loaded back into the edit field by TUTLOAD, the same readable listing with comments will be obtained.

The size of the TIME2.TUT file is 948 bytes. However, if the comments are deleted, the size would be only 307 bytes. Although the comment lines appear within the code in the listing, the TUTSAVE operation separates them from the code and saves them at the end of the code in the sucro file. Then the use of comments does not slow down the performance.

As a comparison, a corresponding Survo operation has been programmed in C. It has the following listing:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include "survo.h"
#include "survoext.h"

main(argc,argv)
int argc; char *argv[];
{
    long ltime;
    int hour;
    char sentence[50];
    int edit_line;

    s_init(argv[1]);
    time(&ltime);
    hour=atoi(ctime(&ltime)+11);
    strcpy(sentence,"Now it is ");
    if (hour>18) strcat(sentence,"evening.");
    else if (hour>12) strcat(sentence,"afternoon.");
    else if (hour==12) strcat(sentence,"noon.");
    else if (hour>5) strcat(sentence,"morning.");
    else strcat(sentence,"night.");
    edit_line=rl+r;
    edwrite(space,edit_line,1);
    edwrite(sentence,edit_line,1);
}
```

The list may be shorter, but the program file needs over 14 kilobytes since various run-time library routines (taking care of input/output and Survo interface) must be linked to the program. For details of programming Survo modules in C, see Mustonen (1989).

12.3 Code words of the sucro language

Although a sucro were originally created in the tutorial mode, refinements are best to carry out by writing sucro program code in the edit field. In this code, standard text to be written by the sucro in the edit field appears as such. The control keys (arrow keys and other keys moving the cursor or changing the status of the edit field in other ways) must have visible code words in the program listing. These codes are given in braces.

All sucro code words and statements are presented in Appendix 3. The inquiry system gives information by activating SUCRO? . One can also learn

various codes by creating a test sucro in the tutorial mode and by using various special keys and key combinations in it. After definition, it is easy to activate TUTLOAD and see what appears in the listing. After such an experiment, we could have in the edit field:

```

14 1 SURVO 84C EDITOR Wed Jul 29 11:44:02 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTLOAD TEST1_
3 *{R}
4 *Pressing the right arrow key 7 times:{r7}{R}
5 *HOME key pressed once:{home}{R}
6 *TAB key three times:{tab}{tab}{tab}{R}
7 *{end}
8 *

```

It is easy to find the code words of the special keys in the listing. The text in the listing can be freely divided between the edit lines. For example, the following listing defines the same sucro:

```

14 1 SURVO 84C EDITOR Wed Jul 29 11:48:51 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTSAVE TEST1_
3 *{R}
4 *Pressing the right arrow key
5 * 7 times:{r7}{R}
6 *HOME key pres
7 *sed once:{home}{R}
8 *TAB key three times:{tab}{tab}{tab}{R}
9 *{end}
10 *

```

The TUTLOAD command, however, has a practice of starting a new line after each line feed {R} character. Then the listing reflects the true action of the sucro. Lines extending over the visible length of the edit line are broken according to the situation. One can control the line changes in the listing also by using comment lines with a '/' in the control column.

Writing of more complicated structures (like prompts for the user, conditional actions, etc.) is supported by the sucro /SUCRO. Information on its usage is obtained by activating /SUCRO ? .

The following table summarizes the most important keys and their sucro codes:

| Key | Code word | Key | Code word |
|----------|------------|-----------|------------|
| ENTER | {R} | END | {line end} |
| ESC | {act} | F1:HELP | {help} |
| HOME | {home} | F2:PREFIX | {pre} |
| PgDn | {pgdn} | F3:TOUCH | {touch} |
| PgUp | {pgup} | F4:DISK | {disk} |
| right | {r} | F5:FORM | {form} |
| left | {l} | F6:MERGE | {merge} |
| up | {u} | F7:REF | {ref} |
| down | {d} | F8:EXIT | {exit} |
| ctrl-END | {erase} | alt-F2 | {words} |
| INSERT | {ins} | alt-F3 | {copy} |
| DELETE | {del} | alt-F4 | {block} |
| INS_LINE | {ins line} | alt-F5 | {search} |
| DEL_LINE | {del line} | alt-F6 | {file act} |
| TAB | {tab} | alt-F7 | {code} |
| { | {(} | } | {)} |

The codes {r}, {l}, {u}, {d}, {del}, {form} that often occur many times in succession can alternatively be given with a count. Thus {r3} is the same as {r}{r}{r} and {del2} is the same as {del}{del}.

Other important codes

The only compulsory code word in a sucro listing is {end} which terminates the listing and to which all conditional branches of the sucro must be directed.

{init} (key combination {pre} b) initializes the display parameters of the edit field. For example, the normal overtyping mode is selected instead of the insert mode, a reference point set by the F7:REF key is deleted, a shadow line made visible by the keys {pre} S is made invisible, the left edge of the edit field reached by the {R} key is the first position, and an unfinished move by either the BLOCK or the WORDS key is interrupted. By using {init} in the beginning of a sucro, one can make sure that it will start without any harmful side-effects set by earlier actions. In certain technical cases, where unusual settings of the edit field have importance, {init} should not be given.

{line start} (key combination {pre} B) moves the cursor always to the first position of the current edit line in such a way that the first column of the field will also be the first visible column. {line start} is safer than {home} since the function of the latter key depends on the current position of the cursor and the edit field on the screen.

{tempo -1} increases the speed of the sucro by one step, usually to the maximum speed. If tempo codes are not used, sucros take a speed suitable for teaching programs; the user has enough time to read the text written by the sucro. In sucros working as tools in various applications, a greater speed achieved by {tempo -1} should be maintained. It does not prevent the user from regulating the pace of the process by the + and - keys. Before terminating, the original speed should be restored by {tempo +1}.

As an example of using these codes, a sucro /HDATE is made for writing the current place and date in the form

Helsinki, July 29, 1992

```

14 1 SURVO 84C EDITOR Wed Jul 29 16:09:26 1992      D:\P2\SUC\ 150 100 0
1  *
2  *TUTSAVE HDATE
3  / Move to the start of the command line and erase it:
4  *{init}{line start}{erase}{erase}
5  /
6  / Activate sucro /DATE which gives the date in the form
7  / January 15, 1992
8  */DATE{act}
9  /
10 / Increase speed, move to the start of the line, insert mode on:
11 *{tempo -1}{line start}{ins}
12 /
13 / Write in insert mode `      Helsinki, ':
14 *      Helsinki, {}
15 /
16 / Back to overtyping mode and to the start of the line:
17 *{ins}{line start}
18 /
19 / Restore the original speed and stop:
20 *{tempo +1}{end}
21 *

```

In this sucro, the main task is performed by a ready-made sucro /DATE activated on line 8 in the listing above. /DATE works in a maximum speed but reselects the normal speed at the end. The /HDATE takes the maximum speed only after that on line 11 in order to write the necessary additions as quickly as possible.

12.4 Sucro memory

The general means of Survo in information management are available for sucros as well. Thus sucros are able to save and load data by standard Survo operations. In this way, edit fields, data files and other files maintained by Survo serve as ‘memories’ in sucro applications. For immediate working, however, a special sucro memory is needed. Earlier, this memory was called a *tutstack* because one could fill it only like a stack. Currently, this restriction has been abolished and information can now be saved in any place in the sucro memory.

Technically, the sucro memory is simply a string in the central memory of the computer and it consists of partial strings each of which terminates to a ‘@’. The strings in the sucro memory represent both numbers and textual data. The items (locations) in the memory are referred to by notations W1, W2, ... It is possible to replace these names by more legible labels.

The sucros are collecting information to the sucro memory from the edit field and from prompts they are presenting to the user. Information in the sucro memory can then be written to the edit field and it can be used as a basis for various decisions made by conditional sucro statements. Within the sucro

memory, information may be copied from one location to another. One can also make simple calculations with the numbers stored in the sucro memory. For heavier computations, numerical and statistical functions of Survo are immediately available.

Initialization by parameters of the sucro

When a sucro is activated in the edit field by a command with parameters, the parameters are saved in the first locations of the sucro memory. Assume that we have activated a sucro /MINOR by the command
/MINOR 19.1,12.

The parameters 19.1 and 12 are then saved in locations W1 and W2. Assume now that the task of the sucro is simply to announce which of the parameters is smaller by writing in this case: *'Of the numbers 19.1 and 12, the smaller is 12.'*

```

9  1 SURVO 84C EDITOR Wed Jul 29 18:37:56 1992      D:\P2\SUC\ 100 100 0
1  *
2  *TUTSAVE MINOR
3  *{R}{erase}{erase}
4  - if W1 < W2 then goto A1
5  - if W2 < W1 then goto A2
6  *Numbers {print W1} and {print W2} are equal.{goto E}
7  + A1: {W3=W1}{goto B}
8  + A2: {W3=W2}
9  + B: Of the numbers {print W1} and {print W2}, the smaller is
10 * {print W3}.
11 + E: {end}
12 *
13 */MINOR 19.1,12
14 *Of the numbers 19.1 and 12, the smaller is 12.
15 *
16 *TUTSTACK
17 *19.1@12@12@
18 *

```

The sucro code on lines 3-11 is saved by the TUTSAVE command on line 2 as a sucro /MINOR. The sucro is tested on line 13 and it gives the result on line 14. In the program, the comparisons on lines 4 and 5 imply that the smaller of the numbers W1, W2 is copied by the statement {W3=W1} or {W3=W2} to W3. Finally, the code on lines 9-10 writes the required sentence. In this sentence, output statements {print W1} etc. are used for printing contents of the sucro memory. In case of equal numbers, another kind of statement is written by the code on line 6.

If a sucro is activated without parameters, the sucro memory will be initialized by a single word W1=(empty) . On the basis of this information, the sucro is able to study whether the expected parameters are missing. For example, in the beginning of the /MINOR sucro, we could have a check

```

- if W1 '<>' (empty) goto A
*The numbers to be compared are missing!{goto E}
+ A: ...

```

Here, the relation '<>' means that the items have to be compared as strings, not as numbers. Thus, if the first parameter is not (empty), the sucro will continue from label A. Otherwise a message of missing parameters is written and the sucro is terminated by a jump to the label E of the {end} statement.

If a sucro is used as a starting program for the Survo session, W1 will get an exceptional value W1=(start) . Then the sucro is able to recognize when it is used as a starting program.

Deleting and saving of the contents of the sucro memory

All items of the sucro memory are deleted by the statement {del stack}. Items from Wi onwards are deleted by {del stack Wi}. In principle, the number of locations of the memory is not restricted but the total length of information is 255 characters at most. If this limit is exceeded, an error message is displayed. Since the sucro memory should be used for temporary savings only, the limitation is not serious in practice.

Sometimes it is necessary to have a spare copy of the contents of the sucro memory. This is done by the statement {save stack}. The {load stack} statement concatenates the information saved by {save stack} to the present contents of the sucro memory.

The next example illustrates management of the sucro memory by these statements.

```

16 1 SURVO 84C EDITOR Thu Jul 30 10:09:55 1992      D:\P2\SUC\ 100 100 0
 9 *
10 *PUTSAVE MEMORY1
11 *{del stack}
12 / Memory contents deleted.
13 /
14 *{W1=7}{W2=25}{save stack}
15 / Current contents 7@25@ saved.
16 /
17 *{W1=W1+W2}
18 / W1=7+25=32 computed. Memory: 32@25@
19 /
20 *{del stack W2}
21 / Memory deleted from W2 onwards. Memory: 32@
22 /
23 *{load stack}
24 / Current contents concatenated by the saved contents. Memory: 32@7@25@
25 /
26 *{end}
27 *

```

It is also possible to save the entire contents of the sucro memory in a file by the statement {save stack ABC} where ABC stands for the file name. The file will always be located on the path of temporary Survo files defined by temp-disk in SURVO.APU . The information thus saved is loaded back by {load stack ABC}. The former contents of the sucro memory are destroyed in this case.

The name of the file can also be given by an item in the memory; for example, {save stack W3} .

Simple calculations

The functions of sucros are supported in arithmetics by the editorial computing and the touch mode. As one can see in the previous example, also certain immediate calculations may be performed in the sucro memory. This topic will be considered later more thoroughly. However, already now it is worthwhile to notice that the four basic arithmetical operations

$\{W_i=W_j+W_k\}$ $\{W_i=W_j-W_k\}$ $\{W_i=W_j*W_k\}$ $\{W_i=W_j/W_k\}$

are available. Also constants can appear in these computations. For example, $\{W_5=W_5+1\}$ increases the current value of W_5 by 1.

Information (both numbers and strings) is copied by statements like $\{W_i=W_j\}$ and concatenated by $\{W_i=W_j\&W_k\}$. For example, after the statements

$\{W_1=SURVO\}$ $\{W_1=W_1\& 84C\}$,

we shall have the string "SURVO 84C" in W_1 .

Names of the memory locations

The names of the sucro memory locations W_1, W_2, \dots are renamed by more legible labels starting by the capital W on comment lines of the form

```
/ def Wname1=W1 Wname2=W2 Wname3=W3
/ def Wname4=W4 Wname5=W5 .
```

This improves the readability of the sucro program and simplifies editing.

Often one has to change places of operands in the sucro memory. This happens, for example, when new parameters are added to the sucro operation. In those modifications, usually only the `def` lines need editing and no changes in the code itself are essential.

The same memory location can be named several times with different labels in the sucro listing.

The next sucro calculates the least common denominator of two integers and gives the result as a sentence of the form: *'The least common denominator of 144 and 54 is 432.'*

The numerical result is calculated by the means of editorial computing, then isolated from the context by overtyping '=' by a space, and finally saved in the sucro memory by $\{save\ word\ Wlcd\}$.

```

12 1 SURVO 84C EDITOR Thu Jul 30 11:27:35 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTSAVE LCD
3 / /LCD M,N computes the Least Common Denominator of positive integers
4 / M and N as an expression M*N/gcd(M,N) where gcd(M,N) is the
5 / Greatest Common Divisor (library function of Survo).
6 /
7 / Full speed and standard initialization of the edit field:
8 *{tempo -1}{init}
9 /
10 / Empty line for the result is inserted. Cursor to the first position:
11 *{ins line}{line start}
12 /
13 / def WM=W1 WN=W2 Wlcd=W3 (Locations renamed)
14 /
15 / Computing M*N/gcd(M,N):
16 *{print WM}*{print WN}/gcd({print WM},{print WN})={act}
17 /
18 / We have in the beginning of the line 144*54/gcd(144,54)=432
19 / - (cursor)
20 / One step to the left and '=' is overtyped by a space.
21 / (Empty code {} indicates that the space belongs to the line.)
22 *{1} {}
23 /
24 / The current contents of the line is: 144*54/gcd(144,54) 432
25 / - (cursor)
26 / The integer indicated by the cursor (432) is saved in Wlcd:
27 *{save word Wlcd}
28 /
29 / Move to the line start, erase it and write the result:
30 *{line start}{erase}The least common denominator of
31 * {print WM} and {print WN} is {print Wlcd}.
32 /
33 / The original speed is restored before termination:
34 *{tempo +1}{end}
35 *

```

The listing is slightly overcommented. As an example of a normal style in sucro programming, a generalized sucro /GCDLCD is presented. It gives both the Greatest Common Divisor and the Least Common Denominator.

```

41 1 SURVO 84C EDITOR Thu Jul 30 11:34:13 1992      D:\P2\SUC\ 100 100 0
36 *
37 *TUTSAVE GCDLCD
38 / /GCDLCD M,N computes the Greatest Common Divisor and the Least Common
39 / Denominator of two positive integers M and N.
40 / def WM=W1 WN=W2 Wgcd=W3 Wlcd=W4
41 /
42 *{tempo -1}{init}{ins line}
43 /
44 / Computing GCD:
45 *{line start}gcd({print WM},{print WN})={act}{1} {save word Wgcd}
46 /
47 / Computing LCD:
48 *{line start}{erase}{print WM}*{print WN}/{print Wgcd}={act}{1} {}
49 *{save word Wlcd}
50 /
51 / Result is written on two lines. Line 1:
52 *{line start}{erase}The greatest common divisor of {print WM} and
53 * {print WN} is {print Wgcd}
54 /
55 / Line 2:
56 *{ins line}{line start}and the least common denominator is {print Wlcd}.
57 *{tempo +1}{end}
58 *
59 */GCDLCD 144,54
60 *The greatest common divisor of 144 and 54 is 18
61 *and the least common denominator is 432..
62 *

```

On lines 59-61, an application of the sucro can be seen.

12.5 Probe statements

In many applications, sucros will study the edit field containing information written by the user or results given by Survo operations. They have to find various information and save it in the sucro memory for subsequent processing. In contrast to a normal user, the sucros are blind; instead of "seeing" different objects, they have to search for them by probing and touching various places in the field.

For these tasks, Survo provides various key combinations having counterparts as sucro **probe** statements. Besides various search codes, the probe statements include input and output statements between the sucro memory and the edit field as well as statements for moving the cursor in the edit field.

Search and input statements

`{find <character>}` (key combination `{pre} C <character>`) finds the next occurrence of a given character on the current line. The cursor will be moved to touch the character. If the character is not found, the cursor remains in its present position.

`{find Wi}` does the same thing with the first character of the string saved in `Wi`.

`{find string Wi}` finds the next occurrence of the string saved in `Wi` on the current line. `Wi` can be replaced by a literal string.

Examples: `{W4= TO }{find string W4}` = Find " TO "
`{find string ABC}` = Find "ABC"

`{next word}` (keys `{pre} W`) moves the cursor to the first character of the next 'word' on the current line. A 'word' here means a contiguous string containing no spaces. If the cursor is already touching the last word on the line or the next word is not in the visible part of the edit line, the cursor stays in its original position.

`{next col}` works as `{next word}` but finds also words which are not in the visible part of the edit line. In those cases, also the visible part of the edit field will be moved correspondingly.

`{save word Wi}` saves the word touched by the cursor in `Wi`. If the cursor touches a space character, this will be saved in `Wi`.

`{save char Wi}` saves the character touched by the cursor in `Wi`.

`{save line Wi}` saves the part of the line which is to the right from the cursor, in `Wi`. Trailing blanks are not saved.

Each of these save statements have also an older form saving information as the first free item of the sucro memory. For example, `{stack word}` saves the current word to the end of the sucro memory.

Output statements

In sucro programs, all text not within braces is copied to the edit field as such as the user had written it. Thus, the insert mode and the selected display mode have their normal effects. Within such literal text, sucros are able to write information gathered in the sucro memory. Writing can be conditional depending i.e. on what kind of results have been achieved.

`{print Wi}` writes the contents of `Wi` from the position indicated by the cursor onwards. The cursor moves correspondingly. Eventual shadow characters remain as such and earlier text will be overwritten even if the insert mode is on. `{print Wi}` works as Survo operations when they are writing results in the edit field.

`{write Wi}` works as `{print Wi}` but observes the current display and insert mode. It emulates writing of the user. Also the automatic wrap-around function is in use, i.e. when the visible line becomes full, the current word will be written to the next line.

Moving the cursor

In addition to the `REF` key (code `{ref}`) and arrow keys `{r}`, `{l}`, `{u}`, `{d}`, sucros have also other means for moving the cursor and the visible part of the edit field.

`{save cursor Wi,Wj}` saves the current location of the cursor. The line number will be saved in `Wi` and the column number in `Wj`.

`{save corner Wi,Wj}` saves the number of the first visible edit line in `Wi` and the number of the first visible column in `Wj`.

`{jump Wi,Wj,Wk}` moves the cursor to line `Wj` and column `Wk`. The line `Wi` is displayed as the first visible one. Each of `Wi,Wj,Wk` can be replaced by a constant integer.

`{jump Wi,Wj,Wh,Wk}` works as jump above but displays `Wh` as the first visible column.

For example, the current status of the cursor and the position of the visible part of the edit field could be saved by statements

```
{save cursor Wrow,Wcol}{save corner Wrow1,Wcol1} .
```

Later, this status is reached again by

```
{jump Wrow1,Wrow,Wcol1,Wcol} .
```

Of course, the same thing is easier to carry out by the `{ref}` code. Also statements for maintaining up to 8 simultaneous reference point are available. `{ref set i}` defines the current cursor/screen position as an additional reference point `i` (`i=1,2, ..., 8`), `{ref jump i}` moves the cursor to the reference point `i`, and `{ref del i}` removes the reference point `i`.

More probe statements are presented in Appendix 3.

Example 1: Incomplete year designations

Assume that we have in the edit field a list dates in the form day.month.year but in some of them the two first digits (19) of the year have been left out. Our task is to make a sucro /YEARS which inserts the missing digits. The sucro should also make remarks about false dates having not enough '.'s.

```

7 1 SURVO 84C EDITOR Thu Jul 30 16:25:53 1992 D:\P2\SUC\ 100 100 0
1 *
2 *Dates:
3 */YEARS_
4 *27.12.1989
5 *1.3.87
6 *2511.1990
7 *1.5.91
8 *12.7.67
9 *

```

Since the dates are of different length, we must search for the year designation after the second '.' on each line. We assume also that the list ends to an empty line. The task is carried out by the following sucro:

```

15 1 SURVO 84C EDITOR Thu Jul 30 16:29:09 1992 D:\P2\SUC\ 100 100 0
9 *
10 *OUTSAVE YEARS_
11 *{tempo -1}{init}
12 / Take a new line:
13 + A: {R}
14 / Save the first word on the line:
15 *{save word W1}
16 / If the word is empty, the end of the list is encountered. Stop:
17 - if W1 '=' {sp} then goto End
18 / Find the first '.' and save it:
19 *{find .}{save char W1}
20 / If no '.', the date is erroneous:
21 - if W1 '<>' . then goto Error
22 / One step to the right. Find and save next '.':
23 *{r}{find .}{save char W1}
24 / If no '.', the date is erroneous:
25 - if W1 '<>' . then goto Error
26 / Overtyping '.' in front of year by a space:
27 * {}
28 / Save the year. Retype '.':
29 *{save word W1}{1}.
30 / If the year is over 99, no change. Goto next case:
31 - if W1 > 99 then goto A
32 / Since the year is less than 100, add 1900 and type the new value:
33 *{W1=W1+1900}{print W1}{goto A}
34 / In erroneous cases, a message is written at the end of the line:
35 + Error: {line end} Erroneous date{goto A}
36 + End: {tempo +1}{end}
37 *

```

When the sucro is activated on the line 3, it makes the following results:

```

1 1 SURVO 84C EDITOR Thu Jul 30 16:31:34 1992 D:\P2\SUC\ 100 100 0
1 *
2 *Dates:
3 */YEARS
4 *27.12.1989
5 *1.3.1987
6 *2511.1990 Erroneous date
7 *1.5.1991
8 *12.7.1967
9 *_

```

Example 2: Number of lines in the edit field

Sucro /LINENUM finds the number of lines in the current edit field by moving the cursor downwards as long it really moves. This is detected by studying changes in the current line number.

```

9 1 SURVO 84C EDITOR Thu Jul 30 16:46:01 1992 D:\P2\SUC\ 100 100 0
39 *
40 *TUTSAVE LINENUM
41 / def Wline=W1 Wcolumn=W2 Wline2=W3 Wcolumn2=W4
42 *{tempo -1}{init}
43 / Set a reference point. Current line is Wline:
44 *{ref}{save cursor Wline,Wcolumn}
45 / Move to the next line:
46 + A: {R}
47 / Save the new line number Wline2:
48 *{save cursor Wline2,Wcolumn2}
49 / If the line numbers are equal, the end of the field is reached:
50 - if Wline = Wline2 then goto L
51 / Otherwise update Wline and try again:
52 *{Wline=Wline2}{goto A}
53 / Return to the starting point and neglect it as a reference point:
54 + L: {ref}{ref}
55 / Move to the end of the command line and write # of lines:
56 *{line end} = {print Wline}
57 *{tempo +1}{end}
58 *
59 */LINENUM =100
60 *

```

The application on line 59 shows how the sucro writes the result. /LINENUM is not needed in practice since the sucro language has a statement {save dim Wi,Wj} which stores the number of lines and columns of the current edit field.

Example 3: Reversing the order of letters in a word

Sucro /WORDREV reverses a word given on the next line as follows:

```

35 1 SURVO 84C EDITOR Thu Jul 30 17:00:20 1992 D:\P2\SUC\ 100 100 0
61 *
62 */WORDREV
63 *SUCROS in reversed form is SORCUS..
64 *

```

The listing of the program is:

```

16 1 SURVO 84C EDITOR Thu Jul 30 17:01:40 1992 D:\P2\SUC\ 100 100 0
65 *
66 *TUTSAVE WORDREV
67 / def Wrev=W1 Wchar=W2
68 *{tempo -1}{init}{line start}
69 / Move to the beginning of the word on the next line:
70 *{R}
71 / Put an empty string to Wrev. The reversed word will be set there:
72 *{Wrev=}
73 / Save the character indicated by the cursor:
74 + A: {save char Wchar}
75 / If the character is a space, the word has been scanned:
76 - if Wchar '=' {sp} then goto L
77 / Copy the new character to front of the earlier reversed string:
78 *{Wrev=Wchar&Wrev}
79 / Move one step to the right and go to study the next character:
80 *{r}{goto A}
81 / Eventual characters after the original word are erased:
82 + L: {erase}{erase}
83 / Write the result:
84 * in reversed form is {print Wrev}..{tempo +1}{end}
85 *

```

12.6 Activation of sucros

A sucro is normally activated by a command having a '/' in front of the sucro name. The system tries to find a sucro file with the command name primarily in the current data directory of Survo and secondarily in various sucro paths as explained in 12.1 . Also a complete pathname can be given. For example, the command

```
/D:\SUCROS\JOB1
```

activates a sucro JOB1 saved on the path D:\SUCROS .

The contents of the sucro memory is deleted and replaced by the parameters of the command. If no parameters are given, (empty) is put in W1. However, the present contents of the sucro memory remains intact by using @ as the only parameter in the sucro call.

A sucro can also be started without a written command by pressing the keys **P**REFIX and **T** (as for a definition of a new sucro). Then on the bottom line of the screen, the following prompt is displayed:

```
Sucro functions: S=Start definition, R=RUN ? _
```

To run a sucro, **R** is pressed and the prompt is replaced by:

```
Run a sucro: Name of file ? _
```

The user enters a (path)name, the prompt disappears and the sucro starts its work.

While a sucro is running, it can call other sucros by typing sucro commands and by activating them, but it can do that also without visible commands. In fact, there are two main alternatives: the sucros can be chained or they can be nested. In the former case, done by {load <sucro>}, one sucro is followed by another and there is no return to the first one (unless the second calls the first again). In the latter case, done by {call <sucro>}, the second sucro is used as a subroutine and the calling sucro continues when the second one has finished its work.

In both cases, the contents of the sucro memory and other system parameters are not altered. The new sucro continues from the situation reached by the calling sucro.

The next examples will clarify the idea how sucros are working together.

```

4 1 SURVO 84C EDITOR Fri Jul 31 10:26:31 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTSAVE AA
3 * aa{load BB}{end}
4 *
5 *TUTSAVE BB
6 * bb{load AA}{end}
7 *
8 */AA
9 *

```

In this example, sucro AA writes ‘aa’ and then terminates by loading the BB sucro which writes ‘bb’ and terminates by loading AA again, etc. By activating /AA on line 8, we get the following output:

```

36 1 SURVO 84C EDITOR Fri Jul 31 10:37:20 1992      D:\P2\SUC\ 100 100 0
7 *
8 */AA aa bb aa bb aa bb aa bb aa bb aa bb aa bb aa bb aa bb aa bb
9 *aa bb aa bb aa bb aa bb aa bb aa bb_
10 *

```

Because the sequence here will never terminate automatically, we interrupted the run by pressing the key.

Chaining of sucros by this method is based on the {load <sucro>} statement in the sucro program. It makes it possible to split large applications into sucros of a moderate size. When developing larger applications, a good practice is to make individual parts as separate sucros. The parts are then glued together by **load** statements.

The **load** statement can also be used conditionally in the **if** and **switch** statements of a sucro program. There are no limits for the number of **load** statements in sucros.

In the next example, three sucros CC, DD, and EE are written and saved by TUTSAVE commands.

```

4 1 SURVO 84C EDITOR Fri Jul 31 10:43:11 1992      D:\P2\SUC\ 100 100 0
12 *
13 *TUTSAVE CC
14 *{R}
15 *CC starts ...{call DD}{R}
16 *CC ends ...{end}
17 *
18 *TUTSAVE DD
19 *{R}
20 *DD starts ...{call EE}{R}
21 *DD ends ...{end}
22 *
23 *TUTSAVE EE
24 *{R}
25 *EE works ...{end}
26 *
27 */CC_
28 *

```

Here the sucros are calling each other in a nested form. The CC sucro is a ‘main program’ calling DD by a {call DD} statement. The DD sucro in turn calls EE as its subroutine. Thus, when the /CC command on line 24 is activated, the following results will be obtained:

```

12 1 SURVO 84C EDITOR Fri Jul 31 10:47:00 1992      D:\P2\SUC\ 100 100 0
26 *
27 */CC
28 *CC starts ...
29 *DD starts ...
30 *EE works ...
31 *DD ends ...
32 *CC ends ...
33 *

```

During the run, the level of the sucro in action (1, 2, ...) can be seen on the top

line of the screen as the second character.

The maximum number of levels in nested sucros is 5. If this limit is exceeded, an error message *'Too many nested sucros!'* will be displayed and working in the tutorial mode is interrupted. In practice, there is no need for many levels in nested sucros when the application is planned properly.

12.7 Conditional statements

Conditional operations in the sucro language are done by **goto**, **if**, and **switch** statements. These statements refer to labelled points in the program. Any string consisting of letters and digits can be used as a label. In the listing, the labels appear always in the beginning of an edit line with a '+' in the control column. The label is terminated by characters ':' and a space. For example, the code sequence

```
21 *{R}Text div{goto ABC}
22 *This part is not written.
23 + ABC: ided into two parts
```

would write *'Text divided into two parts'*. One space immediately after + ABC: belongs to label notation and the code (here text 'ided ...') continues after that space.

If statement

This statement has either the form

- if <condition> then goto <label>

or

- if <condition> then load <sucro>.

In the second form, when the condition is satisfied, the present sucro will be interrupted and another sucro will continue the work. In both forms, when the condition is not true, the current sucro will continue from the code after the **if** statement.

The possible conditions are simple equalities or inequalities between memory locations W1,W2, ... and constants. The conditions are either numeric or literal (based on string comparisons). In the latter case, the relation symbol (for example, =) must be given in inverted commas ('=').

For example, assume that W1=1.0. Then the condition W1 '=' 1.0 is true, but the condition W1 '=' 1 is not. As numerical comparisons, both W1 = 1.0 and W1 = 1 are true.

The possible relational symbols are =, <, <=, >, >=, and <>. Assuming that W1=1.0, W2=1, W3=ABC and W4=BCD, the following conditions get the truth values:

```

W1 < W2          false
W1 = W2          true
W1 '=' W2        false
W3 '<' W4         true
W3 '<>' W4        true
W1 '<=' W4        true

```

Constants in conditions are written as such. The space character is notated by {sp} and an empty string by {}. For example, if the sucro call needs three parameters, incompleteness of the parameter list could be tested by - if W3 '=' {} then goto Error .

Example 1:

The next sucro merely demonstrates various comparisons:

```

44 1 SURVO 84C EDITOR Fri Jul 31 13:47:32 1992 D:\P2\SUC\ 100 100 0
35 *
36 *TUTSAVE SORT1
37 / def Wnumber1=W1 Wnumber2=W2 Wnumber3=W3 Wtmp=W4
38 *{tempo -1}{init}{line start}
39 - if Wnumber3 '<>' {} then goto A
40 *{ins line}/SORT1 number1,number2,number3{line start}
41 *{ins line}prints the numbers in increasing order on the next line.
42 *{goto End}
43 + A: {R}
44 - if Wnumber1 <= Wnumber2 then goto B
45 *{Wtmp=Wnumber1}{Wnumber1=Wnumber2}{Wnumber2=Wtmp}
46 + B: {}
47 - if Wnumber2 <= Wnumber3 then goto C
48 *{Wtmp=Wnumber2}{Wnumber2=Wnumber3}{Wnumber3=Wtmp}
49 + C: {}
50 - if Wnumber1 <= Wnumber2 then goto D
51 *{Wtmp=Wnumber1}{Wnumber1=Wnumber2}{Wnumber2=Wtmp}
52 + D: {erase}{erase}Numbers in increasing order: {print Wnumber1} {}
53 *{print Wnumber2} {print Wnumber3}
54 + End: {tempo +1}{end}
55 *
56 */SORT1 14.3 -5.5 12.8
57 *Numbers in increasing order: -5.5 12.8 14.3_
68 *

```

It is not reasonable to write any general sorting programs using the sucro language only. When sorting is needed in sucros, it is best to let the sucro call Survo operations SORT, FILE SORT, etc.

Example 2:

The next sucro computes the number of lines below the current line down to the first empty line in the edit field. The sucro has to detect also the case where no empty lines occur.

```

15 1 SURVO 84C EDITOR Fri Jul 31 14:07:44 1992      D:\P2\SUC\ 100 100 0
61 *
62 *PUTSAVE LINES1
63 / def Wsize=W1      # of line in the edit field
64 / def Wlines=W2     # of lines before the first empty line
65 / def Wline=W3 Wcol=W4
66 / def Wtext=W5
67 *(tempo -1){init}
68 / Move to the start of the command line and set the reference point:
69 *{line start}{ref}
70 / Save the total # of lines in Wsize and set Wlines to 0:
71 *{save dim Wsize,Wcol}{Wlines=0}
72 / Move to the start of the next line and save its contents:
73 + A: {R}{save line Wtext}
74 / If the line is empty, go to L:
75 - if Wtext '=' {} then goto L
76 / The line is not empty. Increase # of lines:
77 *{Wlines=Wlines+1}
78 / Save the cursor position. If not the last line, return to the start:
79 *{save cursor Wline,Wcol}
80 - if Wline < Wsize then goto A
81 / An empty line or the end of field found. Return to the command line:
82 + L: {ref}{ref}
83 / Write the result at the end of the command line:
84 *{line end}  Number of lines is {print Wlines}.
85 *(tempo +1){end}
86 *

```

If `sucro /LINES1` is activated above this listing, on line 61, the result would appear on the same line in the form

```
61 */LINES1  Number of lines is 24.
```

Switch statement

This statement permits branching of the program in several alternative ways on the basis of the contents of one of the `sucro` memory locations.

The function and syntax of **switch** is clarified by the next example. We write a `sucro /WEEKDAY` that tells the current weekday below the command line as a sentence of the form "*It is Friday today.*" This `sucro` is essentially supported by the `Survo TIME` command which gives a date/time designation using abbreviations `Sun`, `Mon`, `Tue`, etc. for the weekdays.

```

16 1 SURVO 84C EDITOR Fri Jul 31 14:26:46 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTSAVE WEEKDAY
3 *{tempo -1}{init}
4 / Move to the start of the next line and erase it:
5 *{R}{line start}{erase}{erase}
6 / Activate the TIME command. Move the cursor one step to the right:
7 *TIME{act}{r}
8 / We have the date in the form: *TIME Fri Jul 31 14:20:05 1992
9 /
10 / Save the weekday designation in W1:
11 *{save word W1}
12 / Erase the line and write the beginning of the sentence:
13 *{line start}{erase}It is
14 /
15 / Select the correct alternative according to the contents of W1:
16 - switch W1
17 -   case Sun: goto Sun
18 -   case Mon: goto Mon
19 -   case Tue: goto Tue
20 -   case Wed: goto Wed
21 -   case Thu: goto Thu
22 -   case Fri: goto Fri
23 -   case Sat: goto Sat
24 -   default: continue
25 /
26 / switch statement leads to one of the alternatives:
27 + Sun:  Sunday{goto L}
28 + Mon:  Monday{goto L}
29 + Tue:  Tuesday{goto L}
30 + Wed:  Wednesday{goto L}
31 + Thu:  Thursday{goto L}
32 + Fri:  Friday{goto L}
33 + Sat:  Saturday
34 / Complete the sentence:
35 + L:  today.{tempo +1}{end}
36 *

```

Possible forms of the **case** lines in the **switch** statement are

```

case <value>: goto <label>
case <value>: load <sucro>
case <value>: continue

```

In the last form, the sucro will continue from the point just after the last **case** line. After the last **case** line, also a **default** line of one of the forms

```

default: goto <label>
default: load <sucro>
default: continue

```

may appear. This alternative is applied to those cases to which none of the **case** lines apply.

12.8 User interaction

When a sucro is active, the user has different means for controlling the work. The pace of the process may be adjusted by +, -, and the space bar. A sucro can also be interrupted by . On the other hand, it can present questions for the user or prompt the user to press a key. In certain cases, a sucro allows the user to work quite independently for a while. Such possibility is important in data management applications, for example.

Prompts

Sucros present their questions by a prompt statement of the general form

- prompt <Question ?> {}
- default <default answer>
- answer <memory location for the answer>
- length <max. length of the answer>
- wait <max. waiting time in 0.1 sec.>

The possible question on the **prompt** line will appear on the screen in the place indicated by the cursor. A space whose length in bytes is given on the **length** line is reserved for the answer. In that space, a default answer given by the **default** line will be displayed. The user is expected to edit the default answer either by overtyping or by inserting and deleting characters. The maximum time for answering is given on the **wait** line in 0.1 seconds. The user confirms the answer by pressing **ENTER**. If this key is not pressed within the waiting time, the sucro automatically accepts the current answer. The answer is saved in the memory location given on the **answer** line.

It is also possible break the prompt statement by the **ESC** key. Then the text (**break**) will be saved as an answer. Whenever appropriate, this case must be tested separately in the sucro program.

Although both the prompt text and the answer appear on the screen, they are not automatically written in the edit field. If such a situation is preferred to, the sucro can write the prompt text before the **prompt** statement. Similarly, it is easy to write the answer by a **print** statement after **prompt**.

As an example, we present a sucro /AGE which wants to know your age and makes various comments about the answers.

```

12 1 SURVO 84C EDITOR Fri Jul 31 17:42:41 1992      D:\P2\SUC\ 100 100 0
38 *
39 *TUTSAVE AGE
40 *{tempo -1}{init}{line start}{R}
41 *{erase}{erase}SCRATCH {act}{line start}{erase}
42 *{W1=}
43 + Question:
44 - prompt Your age in years, please ? {}
45 -   default W1
46 -   answer W2
47 -   length 3
48 -   wait 600
49 *{R}{erase}
50 - if W2 '=' {} then goto No_answer
51 - if W2 '=' 0 then goto Too_young
52 - if W2 = 0 then goto Odd_answer
53 - if W2 < 3 then goto Too_young
54 - if W2 > 99 then goto Too_old
55 *You gave your age as {print W2} years.{goto End}
56 + No_answer:
57 *You didn't answer!{R}{goto Question}
58 + Odd_answer:
59 *Your answer is not an acceptable integer!{R}{goto Question}
60 + Too_young:
61 - if W1 '=' W2 then goto OK
62 *You cannot be so young!
63 *{W1=W2}{R}{goto Question}
64 + Too_old:
65 - if W1 '=' W2 then goto OK
66 *You are hardly so old!
67 *{W1=W2}{R}{goto Question}
68 + OK: OK, Let's believe you!
69 + End: {tempo +1}{end}
70 *

```

If the answer is not given or it seems exceptional, the sucro repeats the question. Working with this sucro may lead to a following 'conversation':

```

23 1 SURVO 84C EDITOR Fri Jul 31 17:46:20 1992      D:\P2\SUC\ 100 100 0
70 *
71 */AGE
72 *Your age in years, please ?      *
73 *You didn't answer!
74 *Your age in years, please ? 0.0*
75 *Your answer is not an acceptable integer!
76 *Your age in years, please ? 1  *
77 *You cannot be so young!
78 *Your age in years, please ? 2  *
79 *You cannot be so young!
80 *Your age in years, please ? 103*
81 *You are hardly so old!
82 *Your age in years, please ? 103*
83 *OK, Let's believe you!_
84 *

```

On key statement

This statement is intended for controlling single key strokes. The syntax is

- on key
- key <char1>: <goto label | load sucro | continue>
- key <char2>: <goto label | load sucro | continue>
- etc.
- wait <max. waiting time>

The sucro waits until the user has pressed a key and selects the corresponding alternative from the **key** lines. 'continue' on a **key** line means continuing

along the code after the **on key** statement. If the user does not press any key within the waiting time or if a key not listed on the **key** lines is pressed, the sucro will continue according to the first alternative.

Special keys are notated on **key** lines as

ENTER, RIGHT, LEFT, UP, DOWN, HOME, HELP, ESC, INSERT, INS_LINE,
DELETE, DEL_LINE, ERASE, NEXT, PREV, DISK, BACKSP, REF, MERGE, COPY,
TAB, HELP, SRCH, ACTIV, MOVE, END, WORDS.

The space character is SPACE and ':' is COLON.

In the following excerpt, the sucro wants to know whether the user likes to continue the work:

```

42 1 SURVO 84C EDITOR Fri Jul 31 18:01:21 1992 D:\P2\SUC\ 100 100 0
86 *
87 *Do you like to continue (Y/N) ?
88 - on key
89 - key N: goto End
90 - key Y: continue
91 - key y: continue
92 - wait 300
93 *{line start}{erase}So you are continuing!_
94 *

```

Because it is obvious that some users press 'y' instead 'Y', it is reasonable to keep also 'y' as an affirmative answer. All other alternatives, like 'N', lead to a negative conclusion.

Disabling the keyboard

In some applications, it is necessary to prohibit user interventions (like pressing of keys + - . and space) in critical situations. For example, in teaching programs, it is important that the user does not break activities when the sucro is making preparations for forthcoming exercises.

The keyboard is disabled temporarily by the {break off} statement and it becomes active again by {break on}. This pair of statements has a vital role also in keyboard sucros as will be described later.

Careless usage of {break off} may lead to difficulties. For example, the next sucro will be stuck in a loop which is stopped only by pressing **ctrl-C** :

```

*TUTSAVE TRAP
+ A: {break off}{goto A}{end}

```

Independent user actions during a sucro

Often sucros are made to help users who do not know very much about Survo but who, for example, should be advised to do things related to data management. The idea is that such a user may start a work with Survo by giving a simple start command (made as a sucro). This leads the user to the actual work process (in FILE SHOW, for example). Normally, sucros do not allow the user to do arbitrary things; they control the work even during operations

like `FILE SHOW`.

To give the user freedom for independent working in applications described above, the control of the sucro can be temporarily suspended by giving a statement `{interaction on}` before starting the saving process. Then the sucro will be inactive until the user terminates the work (in `FILE SHOW`, for example, by pressing the `F8` key). Thereafter the statement `{interaction off}` returns the sucro in normal operation in all situations. The code for this kind of applications could be

```
FILE SHOW {print Wdata}
{interaction on}{act}{interaction off}.
```

The **interaction on/off** statements are valid for Survo operations which are run as child processes of the Survo Editor and are interactive in their work with the user. Such operations are, for example, `FILE EDIT`, `FILE SHOW`, `FILE ACTIVATE` and the inquiry system.

12.9 Arithmetics

The sucro language is not, like Basic or C, a general programming language. Its main role is to produce user-friendly solutions by using Survo functions and objects as tools. In many situations, however, it is convenient to perform simple calculations immediately in the sucro memory without a need to apply true computational means of Survo. Also moving of information from one location to another in the sucro memory is essential. These tasks are carried out by statements:

| | |
|-----------------------------|--|
| <code>{Wi=Wj}</code> | Contents of <code>Wj</code> is copied to <code>Wi</code> . |
| <code>{Wi=Wj+Wk}</code> | Addition |
| <code>{Wi=Wj-Wk}</code> | Subtraction |
| <code>{Wi=-Wj}</code> | Changing the sign |
| <code>{Wi=Wj*Wk}</code> | Multiplication |
| <code>{Wi=Wj/Wk}</code> | Division |
| <code>{Wi=Wj%Wk}</code> | Remainder in integer division |
| <code>{Wi=Wj&Wk}</code> | Concatenation of strings |

Also constants are accepted as operands. For example, by `{W3=W3-2}`, the value in `W3` is diminished by 2. In string constants, the characters

`+ - * / % & { } @`

are not allowed. For example, `{W1= SURVO 84C}` and `{W2= &W2}` are valid statements. In the latter one, 5 spaces are inserted in front of the current contents of `W2`.

Calculations with more than two operands have to be divided into sequences of basic statements. For example, the sum of three locations `W1`, `W2`, `W3` is obtained in two stages by `{W4=W1+W2}{W4=W4+W3}`.

The results of calculations are converted to strings by using as short representation for the number as possible. For example, leading spaces and trailing

zeros in the decimal part are cancelled.

Computations with integers are done using integers of type long. If decimal numbers appear as operands or numbers are divided, floating point arithmetics is applied with operands of type double. The latter operations are much slower since the Survo Editor does not make floating point calculations itself. These tasks are handled by a separate program. In this way, about 10 KB of valuable space in the central memory is saved.

Example 1: Prime numbers

Sucro /PRIMES lists prime numbers 2,3,5,7,11,... in the edit field until it is interrupted by pressing . The program studies consecutive odd numbers N by dividing them by smaller odd numbers. If the remainder in integer division is zero, the number N is composite. It is enough to try only divisors whose squares are N at most.

```

48 1 SURVO 84C EDITOR Sat Aug 01 09:45:20 1992 D:\P2\SUC\ 100 100 0
1 *
2 *PUTSAVE PRIMES
3 / def Wnumber=W1 Wdivisor=W2 Wremainder=W3 Wsquare=W4
4 *{tempo 1}{R}
5 *SCRATCH {act}{home}2 3{Wnumber=1}
6 + A: {Wnumber=Wnumber+2}{Wdivisor=1}
7 + B: {Wdivisor=Wdivisor+2}{Wremainder=Wnumber%Wdivisor}
8 - if Wremainder = 0 then goto A
9 *{Wsquare=Wdivisor*Wdivisor}
10 - if Wsquare < Wnumber then goto B
11 * {write Wnumber}{goto A}{end}
12 *
13 */PRIMES
14 *2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
15 *101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191
16 *193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283
17 *293 307 311 313 317 331 337 347 349 353 359 367_
18 *

```

If a corresponding program is written in Basic, for example, and it is run under a Basic interpreter, it would be about 5 times faster than this sucro program. The sucro is slower because it has no numeric variables. All information in the sucro memory is saved as strings. In arithmetic statements, each operand must first be converted to a numeric form. After computations, the result is to be transformed back to a string.

This slowness has no great significance in typical sucro applications where the proportion of immediate arithmetic operations in the sucro memory is always negligible when compared to calculations done by Survo operations. On the other hand, the environment provided by the edit field and the sucro language gives interesting possibilities for educational purposes.

Example 2: Multiplication tables

Loop structures are created in sucro language by taking some of the memory locations as counters. The counter is initialized before the loop, incremented within the loop, and tested at the end of the loop. It is easy to generate multiple loops, too.

Sucro /MULT creates a multiplication table for numbers 1,2, ..., K in base

K by using a double loop with a counter WM for the first factor and WN for the second one. To get values in the number system K , numeric conversions of Survo are used. For example, 249 is converted into a hexadecimal number by

249(10:16)=_ (giving F9).

When the base number K is greater than 10, extra symbols are needed. Our selection in conversion routines is A=10, B=11, ..., F=15, etc. For example, hexadecimal F9 is $16 * F + 9 = 249$. All multiplications are performed in the decimal system and the results are converted below the current line to base K .

```

13 1 SURVO 84C EDITOR Sat Aug 01 13:58:15 1992      D:\P2\SUC\ 100 100 0
21 *
22 *MULTSAVE MULT
23 / def Wbase=W1 WM=W2 WN=W3 WMN=W4 Wsquare=W5
24 *{tempo -1}{init}{line start}{R}
25 *SCRATCH {act}{home}Multiplication table for integers in base
26 * {print Wbase}:{R}
27 *{Wsquare=Wbase*Wbase}{WM=0}
28 + M: {WM=WM+1}
29 - if WM > Wbase then goto End
30 *{WN=0}
31 + N: {WN=WN+1}
32 - if WN <= Wbase then goto MN
33 *{R}
34 *{goto M}
35 + MN: {WMN=WM*WN}
36 - if WMN >= Wsquare then goto A1
37 * {}
38 + A1: {}
39 - if WMN >= Wsquare then goto A2
40 * {}
41 + A2: {ref}{R}
42 *{print WMN}{10:{print Wbase}}={act}{1} {save word WMN}{home}{erase}
43 *{ref}{ref}{write WMN}{goto N}
44 + End: {tempo +1}{end}
45 *

```

The hexadecimal table generated by /MULT is:

```

1 1 SURVO 84C EDITOR Sat Aug 01 14:00:04 1992      D:\P2\SUC\ 100 100 0
45 *
46 *MULT 16
47 *Multiplication table for integers in base 16:
48 * 1 2 3 4 5 6 7 8 9 A B C D E F 10
49 * 2 4 6 8 A C E 10 12 14 16 18 1A 1C 1E 20
50 * 3 6 9 C F 12 15 18 1B 1E 21 24 27 2A 2D 30
51 * 4 8 C 10 14 18 1C 20 24 28 2C 30 34 38 3C 40
52 * 5 A F 14 19 1E 23 28 2D 32 37 3C 41 46 4B 50
53 * 6 C 12 18 1E 24 2A 30 36 3C 42 48 4E 54 5A 60
54 * 7 E 15 1C 23 2A 31 38 3F 46 4D 54 5B 62 69 70
55 * 8 10 18 20 28 30 38 40 48 50 58 60 68 70 78 80
56 * 9 12 1B 24 2D 36 3F 48 51 5A 63 6C 75 7E 87 90
57 * A 14 1E 28 32 3C 46 50 5A 64 6E 78 82 8C 96 A0
58 * B 16 21 2C 37 42 4D 58 63 6E 79 84 8F 9A A5 B0
59 * C 18 24 30 3C 48 54 60 6C 78 84 90 9C A8 B4 C0
60 * D 1A 27 34 41 4E 5B 68 75 82 8F 9C A9 B6 C3 D0
61 * E 1C 2A 38 46 54 62 70 7E 8C 9A A8 B6 C4 D2 E0
62 * F 1E 2D 3C 4B 5A 69 78 87 96 A5 B4 C3 D2 E1 F0
63 * 10 20 30 40 50 60 70 80 90 A0 B0 C0 D0 E0 F0 100
64 *_

```

12.10 Sucro files

The sucros are saved by `TUTSAVE` commands in files with the default extension `.TUT`. The default path is the current data path of Survo. This path is overridden by writing the name of the sucro with a complete path designation. The program code is saved in a compressed form and it is not readable like a text file. The `TUTLOAD` command is the only way for decrypting the code into a legible form.

Typically, the size of one individual sucro file is from about 20 to 2000 bytes. Numerous small files spend much more space on the disk since the operating system reserves room for files in larger quantities.

To save space, it is profitable to put several sucros in the same file. This helps in keeping things related to the same topic together as one family of sucros. Also copying of files is much faster.

A family of sucros is created by selecting a suitable file name, say `ABC`, and saving the first sucro program, say `PART1`, by the command `TUTSAVE ABC-PART1`.

If the file `ABC.TUT` does not exist, it is automatically created and the code of `PART1` is saved in it. The next sucros, say `PART2` and `LASTPART`, belonging to the same family are saved by commands `TUTSAVE ABC-PART2`, `TUTSAVE ABC-LASTPART`. After these actions, we have a file `ABC.TUT` containing sucros `ABC-PART1`, `ABC-PART2`, and `ABC-LASTPART`. They are called by using complete names. For example, the second one is called by `/ABC-PART2`. The same sucro is listed in the edit field by the command `TUTLOAD ABC-PART2` and it is saved back by the corresponding `TUTSAVE` command.

In sucro families, both the file name and the name of the sucro itself can have 8 characters at most. There are no limits in practice for the number of sucros saved in the same file.

In order to clarify the management of sucro families, we create a sucro file `ABC` as follows (the sucros are purely formal):

```

1 1 SURVO 84C EDITOR Sat Aug 01 16:23:15 1992 D:\P2\SUC\ 100 100 0
1 *
2 *TUTSAVE ABC-PART1
3 *{R}This is the first member PART1.{end}
4 *
5 *TUTSAVE ABC-PART2
6 *{R}This is the second member PART2.{end}
7 *
8 *TUTSAVE ABC-LASTPART
9 *{R}This is the last member LASTPART.{end}
10 *
11 *TUTLOAD ABC
12 *{R}
13 *SURVO 84C SUCROS@ 3@{R}
14 *PART1 81@{R}
15 *PART2 116@{R}
16 *LASTPART 152@{R}
17 *{end}
18 *
19 */ABC
20 *SURVO 84C SUCROS@ 3@
21 *PART1 81@
22 *PART2 116@
23 *LASTPART 152@
24 *_

```

The members of the sucro family ABC are saved by TUTSAVE commands on lines 2, 5, and 8. The family is represented on the disk by a single file ABC.TUT . If it is listed by the FILELOAD command as we have done on line 11, we get a list of sucros belonging to ABC.TUT with their addresses in the file. For example, ABC-PART2 starts from the position 116.

Formally, ABC is a sucro itself. If it is activated (as done on line 19) we get also a corresponding list of sucros belonging to the file.

Warning! Never edit or save sucros with a plain family name only. If, for example, the TUTLOAD ABC command on line 11 is changed to a TUTSAVE ABC command and activated, all current members will be deleted. ABC.TUT will contain that list of names and addresses only.

A sucro, say PART2, belonging to a family of sucros, say ABC, is deleted by the command

```
TUTDEL ABC-PART2 .
```

After this, file ABC.TUT will contain sucros ABC-PART1 and ABC-LASTPART:

```

1 1 SURVO 84C EDITOR Sat Aug 01 16:41:53 1992 D:\P2\SUC\ 100 100 0
24 *
25 *TUTDEL ABC-PART2
26 *
27 */ABC
28 *SURVO 84C SUCROS@ 2@
29 *PART1 63@
30 *LASTPART 98@
31 *_

```


12.11 Types of sucros

Sucros can be classified according to their usage and structure into different categories. The prominent features of each type are reflected in the programming technique. The following types of sucros will be considered:

1. Key sucros
2. Keyboard sucros
3. Expert applications
4. Sucros as Survo operations
5. Teaching programs
6. Starting Survo by a sucro

Key sucros

If the name of a sucro has only one character, for example **X**, it can be activated, instead of the command `/X` written in the edit field, also by the key combination `[PREFIX] [M] [X]`. The latter alternative has the advantage that no traces are left in the edit field.

Due to this simple activation method, one-character sucros are called key sucros. Possible names are all characters which are accepted in file names, for example, 0,1,2,3, ... ,9,A,B,C, ...

Certain auxiliary functions related to text editing are made as key sucros since writing of visible commands within the text would be annoying in such tasks. For example, following key sucros belong to the standard repertoire:

- X** Interchanging two consecutive words
- C** Converting lowercase letters in the current word into upper case
- L** Converting uppercase letters in the current word into lower case
- H** Hyphenating the last/first word on the line

Key sucros are usually tools which should work quickly without any harmful side-effects in the edit field. The user can obtain more information about them by activating the sucro in question with '?' as a parameter. For example, `/X ?` tells about the **X** sucro on the command line. If the sucro needs more lines for the instructions, it usually inserts empty lines below the command line.

The current **X** sucro program is:

```

17 1 SURVO 84C EDITOR Sun Aug 02 10:32:15 1992      D:\P2\SUC\ 100 100 0
1 *
2 *TUTLOAD C:\E\S\X
3 / Key sucro X bt SM (revised 1 Aug 1992)
4 /
5 *{tempo -1}{init}
6 /
7 / When /X ? is activated, tell about usage:
8 - if W1 '<>' ? then goto A
9 *{erase}{erase} PREFIX M X exchanges the current and the next word.
10 *{goto End}
11 /
12 / Find the start of the first word:
13 + A: {save char W1}
14 - if W1 '<>' {sp} then goto C
15 + B: {1}{save cursor W1,W2}
16 - if W2 = 1 then goto End
17 *{save char W1}
18 - if W1 '=' {sp} then goto B
19 + C: {save char W1}
20 - if W1 '=' {sp} then goto D
21 *{save cursor W1,W2}
22 - if W2 = 1 then goto E
23 *{1}{goto C}
24 + D: {r}
25 /
26 / Save the first word W1 and set a reference point:
27 + E: {save word W1}{ref}
28 /
29 / Save the second word W2 and return to the reference point:
30 *{next word}{save word W2}{ref}
31 /
32 / Write the words in opposite order:
33 *{print W2} {print W1}
34 /
35 / Delete unnecessary spaces between the words:
36 + F: {save char W1} {}
37 - if W1 '<>' {sp} then goto F
38 /
39 / Return to the reference point and delete it:
40 *{ref}{ref}
41 + End: {tempo +1}{end}
42 *

```

It is assumed that the words to be interchanged are visible and on the same line. X is used by placing the cursor to touch the space between the words or any character of the first word and by pressing the keys **PREFIX** **M** **X**.

The X sucro has a secondary function, too. If it is activated after the last word on the line, the line will be extended by a copy of this word.

Even sucros with multicharacter names can be activated as key sucros although it is more complicated. For example, a sucro TEST could be activated by pressing the keys

PREFIX ! T E S T **ENTER**

This corresponds to activation of the command /TEST. If the contents of the sucro memory should be preserved, the sucro is activated by keys

PREFIX ; T E S T **ENTER**

This amounts to the sucro program statement {call TEST}.

Keyboard sucros

It is possible to change functions of selected keys on the keyboard temporarily by the sucro technique. Various tasks from simple macros to sophisticated series of actions can be defined. Sucros having such properties are called keyboard sucros.

In keyboard sucros, a `{get key}` statement plays an essential role. Other important statements are `{message}`, `{break off}`, and `{break on}`.

When `{get key W1}` is executed in a sucro program, the system waits until the user has pressed a key and saves the code of that key in the memory location `W1`. If the key is a normal typewriter key, it is not echoed, i.e. no character is displayed on the screen or written in the edit field. In case of special keys, like arrow keys, function keys, `[ENTER]`, `[ESC]`, etc. the function of that key is performed normally and the text `SK` (=Special Key) is saved in `W1`. Thus the normal typewriter keys can get a special treatment. The sucro is able to test which key has been pressed and continue accordingly.

As a simple example, we present a sucro `/S1` which alters the function of 'S' only. Whenever the key `[S]` is pressed, the sucro writes the text 'SURVO 84C '. Otherwise the keyboard works normally.

In order to restore the original functions of the keyboard, one of the keys must be selected for termination of the sucro. Our suggestion is that '#' should be used as a termination key.

```

11 1 SURVO 84C EDITOR Sun Aug 02 12:15:56 1992      D:\P2\SUC\ 100 100 0
44 *
45 *TUTSAVE S1
46 *{tempo -1}
47 + A: {get key W1}
48 - if W1 '=' S then goto S
49 - if W1 '=' SK then goto A
50 - if W1 '=' # then goto End
51 *{write W1}{goto A}
52 + S: SURVO 84C {goto A}
53 + End: {tempo +1}{end}
54 *

```

Sucro `/S1` stays constantly in a loop starting from label `A` (line 47). If the user presses `[S]`, the sucro jumps to label `S`, writes the string 'SURVO 84C ' and returns to the start of the loop. If the user presses a special key (`SK`), its function is carried out and the loop is repeated. If `[#]` is pressed, the sucro terminates. All typewriter keys lead to line 51. They are echoed by the `{write W1}` statement and the loop is repeated.

A proper keyboard sucro must keep the user aware about the fact that the Survo Editor is not in a normal status but certain keyboard functions are exceptional. This is done by displaying an announcement on the bottom line of the screen. A `{message}` statement is available for this purpose. Its syntax is `{message}<text to be displayed>@`.

The text is terminated in the program by character '@'. An improved version

of /S1 with {message} statements could be:

```

4  1 SURVO 84C EDITOR Sun Aug 02 12:35:23 1992      D:\P2\SUC\ 100 100 0
44 *
45 *PUTSAVE S1
46 *{tempo -1}
47 *{message}          S = SURVO 84C          To stop, press #@
48 + A: {get key W1}
49 - if W1 '=' S then goto S
50 - if W1 '=' SK then goto A
51 - if W1 '=' # then goto End
52 *{write W1}{goto A}
53 + S: SURVO 84C {goto A}
54 + End: {message}@{tempo +1}{end}
55 *
56 */S1_
57 *
58 *
59 *
60 *
61 *
62 *
63 *
64 *
65 *
66 *
      S = SURVO 84C          To stop, press #

```

One can see how activation of /S1 affects the bottom line of the screen. Before terminating, keyboard sucros should remove their messages. It is done simply by giving an empty message by the statement {message}@ (see line 54 above). If the execution of the sucro leads to an error condition or if the user interrupts it by the key, the current message will be erased automatically.

As an example of a genuine keyboard sucro, we present /S which is intended for typing of shadow characters. It belongs to the set of standard sucros. Because it has only one character in its name, it can also be activated simply as a key sucro.

```

17 1 SURVO 84C EDITOR Sun Aug 02 13:33:54 1992      D:\P2\SUC\ 100 100 0
68 *
69 *PUTLOAD C:\E\S\S_
70 *{tempo -1}{break off}
71 - if W1 '<>' ? then goto 1
72 *{R}
73 *{erase}{erase}/S is a keyboard sucro for typing shadow characters.{R}
74 *{erase}{erase}The '-' key turns on/off the shadow writing mode.{R}
75 *{erase}{erase}This keyboard sucro is terminated by the '#' key.{R}
76 *{erase}{erase}{goto End}
77 + 1: {message}          Enter shadow mode by -. Stop: #@
78 + A: {del stack}{get key}
79 - if W1 '=' SK then goto A
80 - if W1 '=' - then goto 2
81 - if W1 '=' # then goto End
82 *{write W1}{goto A}
83 + 2: {message}          SHADOW typing. Resume edit mode by -. Stop: #@
84 + B: {del stack}{get key}
85 - if W1 '=' SK then goto B
86 - if W1 '=' - then goto 1
87 - if W1 '=' # then goto End
88 *{d}{u}{pre}S{d}{write W1}{u}{pre}S{goto B}
89 + End: {message}@{break on}{tempo +1}{end}
90 *

```

Sucro /S helps in typing of shadow characters. In the normal procedure (without /S), writing a shadow character in the place indicated by the cursor needs

7 keystrokes:

| | |
|--------------------|----------------------------|
| PREFIX | |
| S | Open the shadow line |
| <arrow down> | Move to the shadow line |
| <shadow character> | Write the shadow character |
| PREFIX | |
| S | Close the shadow line |
| <arrow up> | Return to the current line |

It is too tedious in applications where shadows are regularly in use. This happens, for example, in text editing when various fonts are marked by different colors. By using the /S sucro, these 7 keystrokes are replaced by just one.

When /S is activated, it starts in normal writing mode. By pressing the sucro selects the shadow mode. In this mode, each keystroke means typing a corresponding shadow character in the cursor position. Sucro /S executes here the 7 keystroke sequence automatically.

Hence, /S has two different states defined by loops starting from labels 1 (line 77) and 2 (line 83). The key moves the system from state 1 to state 2 and vice versa.

Already in the beginning, a {break off} statement is given. This ensures that also characters normally used for the speed control (+, -, etc.) are available as shadows.

Other keyboard sucros for a general use are, for example, /BOX for typing of graphical characters and /SUCRO for writing statements of the sucro language. All these sucros are in the subdirectory .\S .

Expert applications

This is one of the most fruitful areas for the sucro technique. Those who are familiar with Survo and specialists in certain application areas can record their professional tasks as sucros.

Such tasks are, for example, regular (daily, weekly, monthly) work routines that include updating of source data, statistical summaries, analyses and computations, presenting results in textual, tabular and graphical form, and finally making a report of results ready for publication. Each stage in those tasks may also require conditional actions. For example, if the phenomenon under consideration is going to a positive direction, it should be commented in different phrases and illustrations as in a negative case.

The main idea is to make the repetition of the same application easier. Then there is no more a need to know about the technical matters related to the job. If an expert application is properly planned as a sucro, it may be employed even by people who are not familiar e.g. with all details of statistical methods in question but who are able to enter new data values and interpret the final results.

Experienced Survo users who make sucros tell that they have been able to program their expert applications in a time which is negligible compared to that required when doing the same thing with standard programming languages. In many problems, the first version of the solution is made simply by turning the tutorial mode on and letting the system save the user's actions.

Often, making of a sucro application starts from the observation that certain series of actions tend to appear repeatedly. The user feels that it would be nice to have the task in a more automatic form. On the basis of such experiences, advanced users build their own applications as sucros.

Extensive expert solutions are not always created immediately. The problem is not in programming itself. Rather the difficulty is how to present the task at hand in a suitable form. Good ideas deserve experimenting and testing before they are developed to solutions that work. Thus, a sucro programmer usually creates various prototypes of the task and saves them as work schemes in edit files. Only after such experiments, it is worthwhile to think the possibility of an automatic solution as a sucro.

Expert applications made as sucros can appear in various forms. Some of them are like extensions of standard operations and commands. The largest applications are often like independent information systems. Even the existence of Survo can be hidden behind their own interface presented by menus, for example.

Sucros as Survo operations

Sucros of this type are closely related to the expert applications. In this category, the formal usage of the sucro should resemble that of a normal Survo operation or command. Thus, we are calling the sucro by a command with parameters.

The desired properties and typical examples were mentioned already in 12.1.

Teaching programs

The sucro technique was originally invented for teaching the editorial interface of Survo. By tutorials made as sucros, one could demonstrate how Survo is used in different applications. The current version includes several teaching programs and demonstrations initiated by sucros `/TUTORS` and `/!TEST`. The largest set of tutorials (`/OPETUS`) consisting of over 100 sucros is currently available as a Finnish version only.

Many tutorials have been made on various subjects which are not only related to usage of Survo but to various teaching situations in mathematics, probability, statistics, etc.

Most parts of teaching programs are originally created by working in the tutorial mode and by letting the system to save the keystrokes. These prototypes are then enhanced by inserting prompts and conditional statements. The teaching programs usually consist of many sucros linked together through menus. They may include both advisory text, examples and exercises to be

done by Survo operations.

In teaching, the sucro technique is especially suitable for "telling of stories". Instead of watching the story like a TV program, the user may take part, for example, by interrupting the tutorial and by trying to continue on his/her own from the current point of the presentation. Such tutorials are, in fact, hypertext applications.

When making teaching programs as sucros it is important to pay attention to the speed control and to the timing of the activities. The basic writing speed of a sucro is rather slow and corresponds to the reading speed of an average user. As described before, the user is able to control the pace by certain keys. An important alternative in teaching is the `F1:HELP` key. If it is pressed, the automatic execution of the tutorial will be interrupted and the user may then continue stepwise according to prompts displayed on the screen. By pressing `HELP` again, the tutorial restores the automatic execution mode and the basic speed again.

There are also specific statements for the speed control and timing in the sucro language. The statement `{tempo n}`, where `n` is an integer, selects the speed. The default value for `n` is 2 and greater values set a **slower** speed. `{tempo 1}` corresponds to the maximum speed. Also `{tempo 0}` is possible but then the speed cannot anymore be slowed down by the user.

In most cases, relative changes done by `{tempo -1}` (faster) and `{tempo +1}` (slower) are recommended. In sucro tools intended for a fast execution, `{tempo -1}` is selected in the beginning and at the end the standard speed is restored by `{tempo +1}`. This is important especially in sucros which are used as subroutines since it guarantees a uniform time control in the entire application.

In teaching programs, the standard speed is generally the default choice. Sometimes it is, however, better to fill the screen rapidly with explanatory text and let the user study it until he/she presses a key. This is done by the following statements in the sucro program:

```
*{tempo-1}
Text on the screen
*{message}          Continue by pressing ENTER!@
- on key
-   key _: continue
-   wait 600
*{message}@{tempo +1}
```

When a tutorial uses the basic or a slower speed, temporary halts can be set in any place by `{wait n}` statements where `n` is the waiting time in 0.1 seconds. For example, phrases terminated by a comma, could be halted by `{wait 5}` and

complete sentences by `{wait 10}` or even by `{wait 20}`. When certain results are to be studied, still longer waits should be used. A wait is interrupted when the user presses a key.

Starting Survo by a sucro

The default entry is the `START` edit field in the system directory of Survo. When Survo is started, the contents of this field are displayed on the screen.

This entry can be replaced by another by using a start sucro defined by a line

```
start_sucro=<pathname of a sucro>
```

in the system file `SURVO.APU`.

The start sucro can also be specified when calling the program `S.EXE` (the editor) in the system directory of Survo. The name of a start sucro is given as the first parameter of the call. Normally Survo is entered by the command `SURVO` which invokes a batch file `SURVO.BAT` of the form

```
REM ... Loading Survo ...
CD \E
S %1
CD \
```

The parameter (described by `%1` above) is copied as such to the `S` command. Thus, by the command

```
SURVO <sucro>
```

the system will be initiated and the sucro activated.

If no parameters appear in the `SURVO` (or `S`) command, Survo activates the sucro given by line `start_sucro` in `SURVO.APU`. If this line is missing, the `START` field is loaded immediately.

The task of a start sucro is to lead the user straight to an environment which is appropriate for the beginning of the actual work. It makes the use of Survo easier for unexperienced users. They are guided, for example, to a menu which gives entries to various teaching programs or to various applications. In fact, in the original setup of Survo, we have typically

```
sucro_path=SURVO-START
```

in `SURVO.APU`. This start sucro opens the "main menu" of Survo.

In general, there are no limits for the nature of a start sucro. Usually it simply selects a data directory, opens a suitable edit field, moves the cursor to a specific point in that field, and stops. It can also present questions to the user and make selections according to answers.

A start sucro is essential when interactive application systems are created. In those cases, the start sucro opens the first menu. In such applications, the users have no need to know about the existence of Survo and about its normal usage.

The main menu of Survo is maintained by the following SURVO-START sucro:

```

27 1 SURVO 84C EDITOR Mon Sep 21 10:33:23 1992      D:\P2\SUC\ 100 100 0
1 *
2 *LOAD C:\E\S\SURVO-START
3 / Start sucro for Survo                               S.Mustonen, 1992
4 / The menus used by /SURVO-START are in the edit field .\S\SURVOM.EDT .
5 / The English version is SURVOME.EDT and the Finnish SURVOMS.EDT .
6 /
7 *{tempo -1}{init}
8 + S: {line start}{erase}{erase}
9 /
10 *{save systempath W2}LOAD {print W2}S\SURVOM,1,21,50{act}
11 + VD: {message}                                     Press ENTER after your selection!@
12 - prompt
13 -   default 1
14 -   answer W2
15 -   length 1
16 -   wait 3000
17 -   switch W2
18 -   case 0: goto V0
19 -   case 1: goto V1
20 -   case 2: goto V2
21 -   case 3: goto V3
22 -   case 4: goto V4
23 -   case 5: goto V5
24 -   case X: goto VX
25 -   case x: goto VX
26 -   case 0: goto V0
27 -   case o: goto V0
28 -   default: goto VD
29 + V0: {R}
30 *{d}{message}@{W1=(menustart)}{tempo +1}{load OPETUS}
31 + V1: {R}
32 *{d}{message}@{tempo +1}{load RETURN}
33 + V2: {jump 1,1,1,1}SCRATCH {act}{R}
34 *Entering the inquiry system:{R}
35 *{interaction on}HELP?{act}{interaction off}{goto S}
36 + V3: {R}
37 *{d}{message}@{W1=(menustart)}{tempo +1}{load !TEST}
38 + V4: {R}
39 *{d}{message}@{W1=(menustart)}{tempo +1}{load SURVO-SETUP}
40 + V5: {R}
41 *{d}{message}@{W1=(menustart)}{tempo +1}{load SURVO-WORDS}
42 + VX: {R}
43 *{d}{message}@{form5}Exit from Survo (Y/N)? N{1}{form3}
44 - on key
45 -   key _: continue
46 -   key Y: goto VX2
47 -   key y: goto VX2
48 -   wait 1200
49 *{goto S}
50 + VX2: {exit}Y{R}
51 *{end}
52 *

```

The menus are in the edit field SURVOM.EDT . The sucro works in coordination with this field. Combinations of sucros and ready-made edit fields are typical in application systems.

The display when SURVO-START is working is following (in the English version):

```

50 1 SURVO 84C EDITOR Mon Sep 21 10:35:07 1992      D:\P2\SUC\ 100 100 0
1 *
2 *
3 *
4 *
5 *
6 *
7 *
8 *
9 *
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *

```

| |
|--|
| Main menu of the Survo system |
| 0. Teaching programs (in Finnish) 1. Standard entry (START field) 2. Inquiry system (HELP) 3. Testing Survo functions (a guided tour) 4. Survo setup functions 5. Survo keywords as a hypertext |
| X. Exit |
| Select one of the alternatives 1,2,... 1 |

```

Press ENTER after your selection!

```

12.12 Error control in sucros

Errors in applications using various Survo operations may have harmful consequences in sucros. For example, a sucro may try to operate on a non-existent data file suggested by the user. To overcome such failures, an automatic error recovery scheme has been created.

Whenever an error occurs in a Survo operation, it gives an error message of its own. The message is displayed until the user presses a key.

Such messages are not given when the operation is activated by a sucro. Instead of an interrupt, the current contents of the sucro memory will be replaced by

```
ERR@<error_no.>@<name_of_operation>@<error_message>@
```

A special error handler is called to display the error message on the bottom line and the current job is terminated.

We illustrate this practice by the following sucro /CORREL which computes correlations from a data set given by the user as a parameter. The actual task is carried out by the standard CORR operation.

In the next display, the /CORREL program listing and the situation after an attempt to apply it to a non-existent data set is presented:

```

1 19 1 SURVO 84C EDITOR Wed Aug 05 14:52:36 1992      D:\P2\SUC\ 100 100 0
2 *SAVE EX11
3 *TUTSAVE CORREL
4 *{tempo -1}{init}{line start}{R}SCRATCH {act}{line start}
5 *CORR {print W1},CUR+1{act}
6 *{tempo +1}{end}
7 *
8 */CORREL NODATA
9 *CORR NODATA,CUR+1_
10 *
11 *
12 *
13 *
14 *
15 *
16 *
17 *
18 *
19 *
20 *
21 *
22 *
23 *
      Cannot open Survo data file D:\P2\SUC\NODATA.SVO!  Press ENTER!

```

As the CORR operation is unable to find the data set NODATA, it does not give it own error message but saves information about the error condition to the sucro memory in the form

```

W1=...          (indicates an error)
W2=1           (error no.)
W3=FILE OPEN  (function where the error occurs)
W4=Cannot open Survo data file D:\P2\SUC\NODATA.SVO!
                Press ENTER!

```

and terminates. The Survo Editor notices immediately that an error has occurred (from W1 in the sucro memory). It removes the error condition by setting W1=ERR and calls the current error handler sucro. The message on the bottom line of the display is given by this error handler. This sucro is still running (see '1' in the beginning of the top line) and it is terminated as soon as the user presses **ENTER**. Then also the error message is erased.

The default error handler is SURVOERR.TUT in the directory .\S:

```

24 1 SURVO 84C EDITOR Wed Aug 05 15:12:23 1992      D:\P2\SUC\ 100 100 0
9 *
10 *TUTLOAD C:\E\S\SURVOERR
11 / SURVOERR: default Survo error handler for sucros      3.5.1991/SM
12 *{tempo -1}{disp on}{W5=          }{W4=W5&W4}{W5= Press ENTER!}
13 *{W4=W4&W5}{message W4}
14 - on key
15 -   key _: continue
16 -   wait 1200
17 *{message}@{tempo +1}{end}
18 *

```

Sucro /SURVOERR simply modifies the error message W4 by inserting some blanks and displays that string by using the {message} statement. Then it stays in an **on key** statement and terminates when the user presses a key.

The default error handler can be replaced by any other sucro, in this case by a sucro /CORREERR, for example. It will be active in error conditions after the statement {error handler CORREERR}.

In our previous example, the default error handler is replaced by CORREERR in this way:

```

50 1 SURVO 84C EDITOR Wed Aug 05 15:30:29 1992      D:\P2\SUC\ 100 100 0
19 *
20 *TUTSAVE CORREL2
21 *{tempo -1}{init}{line start}{R}SCRATCH {act}{line start}
22 *{error handler CORREERR}CORR {print W1},CUR+1{act}
23 *{tempo +1}{end}
24 *
25 *TUTSAVE CORREERR
26 *{tempo -1}{R}Error in CORR operation:{R}{print W4}
27 *{tempo +1}{end}
28 *
29 */CORREL2 NODATA
30 *CORR NODATA,CUR+1
31 *Error in CORR operation:
32 *Cannot open Survo data file D:\P2\SUC\NODATA.SVO!_
33 *

```

In the new version of the sucro /CORREL2, the error handler is reselected by {error handler CORREERR} before activating the CORR operation. The program code of /CORREERR is on lines 26-27. When now /CORREL2 is activated with a non-existent data set, the new error handler gives its own announcement on lines 31-32.

Instead of stopping the work, the error handler could after the message give the user a chance to try again. There is no limit for how many times the error handler is reselected during a sucro application. Thus each stage of the work can have an error handling procedure of its own.

In the current Survo version (4.04), only certain Survo operations are employing the new error recovery scheme. Such operations are FILE SHOW, CORR, LINREG, FACTA, for example. Whether a Survo operation has this feature or not can be seen by testing it under a sucro and with the default error handler.

Appendix 1

Keys and their functions

Function keys

HELP (F1)

activates the inquiry system of Survo. **HELP** usually gives information of the last operation attempted. By pressing **HELP** another time while staying in the inquiry system, information on the inquiry system itself is displayed.

PREFIX (F2)

is a prefix key which alters the functions of other keys temporarily. A list of those special functions is given later in this section.

TOUCH (F3)

enters the touch mode. More information is obtained by **HELP** while staying in the touch mode.

DISK (F4)

selects the Survo data disk by giving the alternatives, say, **A:**, **B:**, **C:** . \D\ in turn on the header line of the edit field.



FORMAT (F5)

selects various display attributes, colors etc. for characters to be typed. The current alternative is given as the last character on the header line.

MERGE (F6)

splits the current line into two lines from the position of the cursor (provided that the next line is empty). If the current line to the right from the cursor is empty, the next line will be connected to the current line.

REF (F7)

stores the current position of the cursor. If **REF** is pressed another time while the cursor is indicating another point, the display corresponding to the reference point will be restored.

If **REF** is pressed another time when the cursor is in the reference point, the current reference point is released and another may be selected.

EXIT (F8)

Exit from Survo

INSERT (F9)

toggles the insert mode or inserts a space.

(See also **PREFIX INSERT** and **PREFIX DELETE**.)

DELETE (F10)

deletes a character.

WORDS (alt-F2)

initiates the definition of a sequence of words (sentence) either to be moved to another place or to be deleted.

Prompts for alternative actions will appear on the bottom line. The sequence of words is defined by indicating the first letter of the first word by **WORDS** and the last letter of the last word by **WORDS** again.

Thereafter the sequence is deleted by **ctrl-END** or moved to another place indicated by the cursor by **WORDS**. In the latter case the cursor should usually point to a space between two words. To preserve the sequence in its original position, enter the insert mode before pressing **WORDS**. In any case, the sequence is saved in a special file and it can be copied to another place later by pressing **WORDS** in that place 4 times.

The text chapters are also automatically adjusted if a model for a suitable **TRIM** command is given by a line of the form

```
autotrim=TRIM3 72
```

in the **SURVO.APU** file.

COPY (alt-F3)

copies a line.

Prompt '*Line to be copied?*' appears on the bottom line.

Default answer is the current line.

BLOCK (alt-F4)

initiates a sequence of key strokes for defining a rectangular block in the current edit field. Prompts for alternative actions will appear on the bottom line.

A block is defined by indicating its two opposite corners (by **BLOCK**) and it can then be copied to various places (by **BLOCK**) and/or it can be erased (by **ctrl-END**). To interrupt these actions after **BLOCK** has pressed, press **DELETE**.

SEARCH (alt-F5)

initiates a search in the edit field. The user is prompted to enter the characters of the search string one after another and the first occurrence of the string is shown immediately in each stage. The process is interrupted until the given string is not found anymore or by pressing **ENTER**. (See also **FIND?**)

FILE ACT (alt-F6)

displays the structure of the current Survo data file (see **FILE ACTIVATE**).

CODE (alt-F7)

types characters not readily available on the keyboard. At first, prompt 'CODE?' is displayed and the decimal value of the character has to be entered. **CODE** holds this character until it is changed by pressing **PREFIX** and then **CODE**. If no code value is given, **CODE** will be the current character indicated by the cursor.

Another way to select special characters is provided by the key combination **PREFIX P**.

LINE INS (alt-F9)

inserts a new empty edit line.

LINE DEL (alt-F10)

deletes the current edit line.

Other special keys**ESC**

activates the command on the current line.

ctrl-END

erases the line from the current position to the right.

right arrow

moves the cursor one step to the right.

left arrow

moves the cursor one step to the left.

up arrow

moves the cursor one step upwards.

down arrow

moves the cursor one step downwards.

HOME

moves cursor primarily to the first column, then to the first line and finally to the first position in the edit field.

PgDn

turns the next page in the edit field.

PgUp

turns the previous page in the edit field.

TAB

moves the cursor to the next tab position. The tab positions are defined by the T characters on the first edit line having a 'T' in its control column.

If no T line is found, columns 11,21,31,41,... are the default tab positions. After TAB is pressed, numbers to be typed will be correctly aligned to form straight columns.

PREFIX key codes

After the key F2:PREFIX has been pressed, a small rectangle is displayed in the left upper corner of the screen and the next key will have an alternate function:

PREFIX PREFIX

moves the cursor to the end of the visible line.

PREFIX ESC

activates a sequence of Survo operations. Operations on consecutive lines will be performed until '.' is pressed or an empty line or an invalid operation is met.

PREFIX down arrow

moves the cursor to the last line of the current page or if it is already there, below the last non-empty line of the edit field.

PREFIX ENTER

defines the current column as the return position. Thus hereafter when ENTER is pressed the cursor is moved to this column on the next line. This feature simplifies typing of columns.

PREFIX T

is a prefix for various functions in the tutorial mode:

PREFIX T S opens a survo file for recording of the session,

PREFIX T E terminates recording by closing the survo file,

PREFIX T R starts a survo.

PREFIX S

displays the shadow line of the current line below the current line. The shadow line can then be edited as a normal edit line.

To restore the original display after editing, press **PREFIX S** again.

The keyboard sucro /S (activated by **PREFIX M S**) is useful when many shadow characters are to be typed at the same time.

Note that the standard display effects controlled by the **FORM** key correspond to values 1-7 on the shadow line. The maximum number of shadow lines is 20 for each edit field. This value can be changed by commands **REDIM** and **INIT**.

PREFIX P

picks the character indicated by the cursor to be used as the extra character which can be typed by the **CODE** key.

This is a useful procedure for typing and copying special characters which already appear in the edit field.

PREFIX INSERT

initiates the automatic insert mode. Thereafter **INSERT** selects/deselects the insert mode.

PREFIX DELETE

cancels the automatic insert mode. Thereafter **INSERT** always inserts one space in the current cursor position.

PREFIX LINE INS (F2 alt-F9)

inserts the last line lost by **LINE DEL (alt-F10)**.

PREFIX J

initiates a search for an incomplete word just before the cursor and completes it by the first matching word found in the current edit field.

Sucro codes

The following key combinations are useful when recording sucros in the tutorial mode. It is hardly reasonable to use them in standard text editing.

PREFIX W: {next word}

seeks the next word on the current line.

PREFIX C <character>: {find <character>}

seeks the next occurrence of <character> on the current line.

PREFIX R: {stack cursor}

saves the current line and column of the cursor as two last items in the sucro memory.

PREFIX =: {print W1}
prints the first item of the sucro memory in the edit field.

PREFIX @ i: (i=1,2, ..., 9) {print Wi}
prints the ith word of the sucro memory in the edit field as it were a result of a Survo operation (shadow line unaffected etc.).

PREFIX # i: (i=1,2, ..., 9) {write Wi}
writes the ith item of the sucro memory in the edit field as it were typed by the user (insert mode and shadow mode observed).

PREFIX x: {stack char}
appends the current character indicated by the cursor to the sucro memory.

PREFIX w: {stack word}
appends the current 'word' indicated by the cursor to the sucro memory.

PREFIX X: {save line}
puts the current edit line to the right from the cursor in the sucro memory. Trailing spaces are omitted.

PREFIX 0: {del stack}
clears the sucro memory.

PREFIX B: {line start}
moves the cursor to the beginning of the current edit line.

PREFIX E: {line end}
moves the cursor to the end of the current edit line.

PREFIX I: {pre}I
cancels the insert mode.

PREFIX D: {pre}D
deletes the current reference point (set by REF).

PREFIX y: {get key}
appends the next key pressed by the user to the sucro memory. In case of a special key, word SK is appended and the function of the key executed.

PREFIX M <char>:
activates a key sucro with one-letter name <char>.TUT .

PREFIX !<name> ENTER:
activates sucro <name> without echoing the call.

PREFIX d: {save datapath}
puts the current Survo data path in the sucro memory.

PREFIX o: {save default datapath}
puts the default Survo data path in the sucro memory.

PREFIX h: {save tempdisk}
puts the path of temporary Survo files in the sucro memory.

PREFIX g: {save systemdisk}
puts the Survo system disk designation (C:) in the sucro memory.

PREFIX ^: {save systempath}
puts the Survo system path (C:\E\, for example) in the sucro memory.

PREFIX l <any_key>:
puts the key label of <any_key> (name of key) in the sucro memory.

- PREFIX A:** {pre}A
enables (when the insert mode is on) automatic insertion of space for new lines.
- PREFIX a:** {pre}a
cancels automatic insertion of lines in the insert mode.
- PREFIX b:** {init}
initializes the display parameters of the edit field.
- PREFIX u <char>:**
selects the shadow character for the prompt line (default l=red).
- PREFIX L o:** {disp off}
disables the screen. Text is written normally in the edit field but not shown on the screen.
- PREFIX L O:** {disp on}
restores the normal screen display.
- PREFIX /:** {save time}
appends the time (in ms) elapsed from the start of the current Survo session to the sucro memory.

Key sucros

- PREFIX M X**
exchanges two words.
- PREFIX M C**
converts from lowercase to uppercase.
- PREFIX M L**
converts from uppercase to lowercase
- PREFIX M H**
hyphenates the first/last word on the line.
- PREFIX M S**
is a keyboard sucro for typing of shadow characters.

Appendix 2

Survo system parameters

This appendix is intended for users who like to alter the working environment according to their own requirements.

The Survo system is controlled by several parameters which are defined in the `SURVO.APU` text file in the Survo system directory.

The default settings of the system parameters should correspond to the demands specified by the user when Survo is delivered. In some cases, however, certain parameter values must be changed afterwards. This is done by editing the `SURVO.APU` file.

The simplest way for making changes is to activate the `/SURVO-SETUP` sucro which can also be called from the main menu of the Survo system. `/SURVO-SETUP` gives also information about system parameters and about the current status of the Survo installation.

Each time when Survo is taken into use, it reads the system parameters from `SURVO.APU`. A typical listing of the current contents of `SURVO.APU` is as follows:

```

21 1 SURVO 84C EDITOR Sun Aug 09 10:38:23 1992      D:\P2\APP\ 100 100 0
1 *
2 *LOADP C:\E\SURVO.APU
3 */ survo.apu 18.12.1985/SM (13.7.1992)
4 *ep4=100
5 *llength=256 / don't change
6 *lname=32 / don't change
7 *ed1=101
8 *ed2=100
9 *ed3=20 / # of special lines
10 *er3=23
11 *ec3=72
12 *speclist=4000
13 *specmax=200
14 *edisk=C:\E\D\
15 *last_disk=F:
16 *tempdisk=F:
17 *qpath=C:\E\Q\EDQ
18 *sucropath=C:\SUCROS\
19 *eout=F:RESULTS
20 *printer=PRN
21 *print_dev=PS.DEV
22 *plot_mode=PostScript
23 *ps_dev=PS.DEV
24 *crt_dev=CRT16.DEV
25 *plot_dev=CANON2.DEV
26 *plot2_dev=HP.DEV
27 *scale_check=2
28 *accuracy=7
29 *results=70
30 *insert_type=1
31 *disp_wait=2
32 *videomode=EGA
33 *pxl_ratio=0.75
34 *space_break=1
35 *child_wait=0
36 *autosave=0
37 *autotrim=TRIM3 72
38 *M1=SCRATCH /
39 *M2=LOAD INDEX
40 *M3=TRIM3 60,P
41 *M5=/AUTOLOAD /
42 *M6=DISK C:\E\D
43 *M7=3.141592653589793
44 *M?=/SOFTKEYS lists current softkeys on the next 10 lines.
45 *IBM=1
46 *shadows=112,116,120,113,23,254,114,49,62,118
47 *start_sucro=SURVO-START
48 *

```

The SURVO.APU file itself has no line numbers. The line labels above are just line numbers of the current edit field. In this description, they serve as reference numbers.

The lines of SURVO.APU are either mere comments (starting by an isolated '/' as line 3) or of the form

```
<system_parameter>=<value> / <comment>
```

There is no fixed order of the parameters. Certain obscure names of parameters are inherited from the earlier Survo versions.

Max. number of columns on a line

Parameter ep4 (=100 on line 4) has several related tasks. When the user activates Survo operations, the system has to interpret edit lines consisting of several words separated by spaces and commas. Also when reading tables written in the edit field, the system must 'see' the number of columns etc. In many cases, ep4 is the maximum number of such words or numbers on one

edit line. In certain statistical analyses it will be the maximum number of variables.

Max. lengths of lines and names

Parameter `llength` (=256 on line 5) corresponds to the maximum line length in edit fields. It cannot be changed by the user; the entire system must be recompiled if this parameter is changed. Parameter `lname` (=32 on line 6) has also a fixed value in the same sense. It is the maximum allowed length of a path-name in Survo functions.

Size of the edit field

The size of the edit field is controlled by parameters `ed1`, `ed2` and `ed3` (`ed1=101`, `ed2=100`, `ed3=20` on lines 7-9). `ed1` is the field width (control character included), `ed2` is the number of edit lines and `ed3` is the maximum number of shadow lines. These values seldom have practical meaning since each edit field loaded from disk resets their values (as the `START` field when Survo is entered). The `REDIM` or `INIT` command is used to select new values for `ed1`, `ed2` and `ed3` for the current edit field.

Window size

The size of the visible part of the edit field (the window size) is determined by parameters `er3` and `ec3` (`er3=23`, `ec3=72` on lines 10-11). `er3` is # of lines and `ec3` # of columns.

Specifications

Parameters `speclist` and `specmax` (`speclist=2000`, `specmax=100` on lines 12-13) limit the amount of extra specifications. When a Survo operation is activated, all specification text (consisting of specifications in the current subfield and in the potential `*GLOBAL*` subfield) are collected to a list having a length of `speclist` characters at most and consisting of `specmax` items at most.

Paths for data and temporary files

Parameter `edisk` (`=C:\E\D\` on line 14) is the default data disk path). `last_disk` (`=F:` on line 15) is the last available disk drive in the current installation.

Parameter `paths` defines a sequence of predetermined Survo data paths (see `PATHS?`). It is valid only when `lastdisk=Z:`.

`tempdisk` (`=F:` on line 16) defines the path for temporary files created during a Survo session. If `tempdisk` is not given, the Survo system path is used for temporary files.

Inquiry system

Parameter `qpath` (`=C:\E\Q\EDQ` on line 17) selects the path of the default inquiry system (here `C:\E\Q`) and the common forepart (here `EDQ`) for the

files which belong to this inquiry system. The same Survo installation may keep several inquiry systems one of which is active at a time. `QPATH` is a Survo command for reselecting the inquiry system during the session.

Path for sucros

Parameter `sucropath` (=C:\SUCROS\ on line 18) tells the path where the user's own sucros are saved.

Path for measurement units

Parameter `measures` gives the pathname for the edit file containing the alphabetic list of measures used in numerical conversions.

Default is `.\SYS\MEASURES.EDT` (see `CONV?`).

Output file

The name of the output file/device is given by parameter `eout` (=F:RESULTS on line 19). It is changed during a Survo session by the `OUTPUT` command.

Printer

Parameter `printer` (=PRN on line 20) gives the printer port for the default printer.

Parameter `print_dev` (=PS.DEV on line 21) specifies the default device driver in the `PRINT` operation.

Parameter `plot_mode` (=PostScript on line 22) tells the default device in `PLOT` operations. The `DEVICE` specification always dominates this setting.

Parameter `printdef` gives the maximum amount of bytes for definitions of control words in the `PRINT` operation. Default is 6000.

Drivers for plotting

The device drivers for `PLOT` operations are selected by parameters

| | |
|------------------------|---|
| <code>ps_dev</code> | for PostScript (<code>ps_dev=PS.DEV</code> on line 23) |
| <code>crt_dev</code> | for the screen (<code>crt_dev=CRT16.DEV</code> on line 24) |
| <code>plot_dev</code> | for Canon laser printers (<code>plot_dev=CANON2.DEV</code> on line 25) |
| <code>plot2_dev</code> | for HP plotters (<code>plot2_dev=HP.DEV</code> on line 26) |

Control parameters in statistical operations

Parameters `scale_check`, `accuracy` and `results` control the performance of statistical operations. Use keywords `SCALES?`, `ACCURACY?` and `RESULTS?` to acquire more information.

INSERT mode

Parameter `insert_type` (=1 on line 30) determines the function of the `INSERT` key in text editing. Value 1 implies that `INSERT` merely selects/deselects the insert mode. Value 0 implies that each `INSERT` stroke adds one space at the current cursor position.

When typing text between existing lines, an automatic line insertion mode

is adopted (i.e. a new line is inserted when the current becomes full) when parameter `insert_lines` is 1. If it is 0, lines are not inserted automatically. This setting can also be adjusted during the Survo session. To toggle the insert lines mode on, press `[PREFIX] [A]`. To put it off, press `[PREFIX] [a]`.

Display time

`disp_wait` (=2 on line 31) gives the time in minutes for keeping the normal display of the edit field. If the user has not touched the keyboard during this time, the display is turned off temporarily until a key is pressed.

Video modes

`videomode=<EGA, VGA or CGA>` (`videomode=EGA` on line 32) gives the default video mode used in GLOT operations. Parameter `pxl_ratio` should be altered accordingly. `pxl_ratio` (=0.75 on line 33) gives the aspect ratio suitable for the current video mode in screen graphics. Good values are `pxl_ratio=0.75` for EGA and `pxl_ratio=1` for VGA.

Halting in operations

`space_break` (=1 on line 34) controls the possibility of halting the temporary output on the screen given by various Survo operations. If `space_break=1`, the user can halt the output by pressing the space bar. If `space_break=0`, halting is not possible.

`child_wait` (=0 on line 35) gives the waiting time (in sec.) between the exit from a child process (Survo operation) and continuation of the parent process (the editor). The normal setting is `child_wait=0`. Other values are useful, for example, in testing new C modules. Then there is time to see error messages which appear only temporarily on the screen before the editor rewrites the display.

Automatic saving of the edit field

By setting `autosave=n`, the current edit field will be saved in the `SURVO.EDT` file in the Survo directory of temporary files (given by `tempdisk`) once in every `n` minutes. Then, if by accident the contents of the edit field is lost, the last spare copy will be found by the (sucro) command `/AUTOLOAD` or it can be scanned by `/AUTOSHOW`. To omit temporary savings, set `autosave=0` (as above on line 36).

Automatic trimming

`autotrim` (=TRIM3 72 on line 37) gives a `TRIM` command which is to be used for automatic formatting when text chapters are moved by the `WORDS` (`alt-F2`) key.

Macros

Simple macros (softkeys) for typing commonly used commands and other texts are defined on lines `M1=`, `M2=`, ..., `M9=`. Each macro is activated by

pressing the **F2:PREFIX** key and then **#** of the macro. In our example, macros **M1-M7** are defined on lines 38-43. Macros of this type are meant for typing of text only. No special characters are allowed. In Survo, key macros offer more general possibilities.

Screen types

Survo selects the mode used in the screen control by the parameter **IBM** (=1 on line 45). There are 4 alternatives:

- IBM=1** IBM color screen (EGA, for example)
- IBM=2** IBM monochrome screen
- IBM=5** IBM color screen (with retrace wait)
- IBM=6** IBM monochrome screen (with retrace wait)

Shadow characters

The **shadows** list on line 46 gives the attribute bytes (in decimal code) for various Survo display modes. The colors corresponding to possible values (0-255) can be displayed by the Survo command **COLOR ALL**. The 10 selected attribute values in the **shadows** list correspond to the shadow characters **<space>**, 1, 2, 3, 4, 5, 6, 7, 8, 9 according to the table below.

The colors (and background colors) in the second column are those corresponding to the **shadows** list on line 46.

| shadow | display | printer |
|--------|---------------------|------------------|
| space | black (white) | normal |
| 1 | red (white) | bold |
| 2 | gray (white) | subscript |
| 3 | blue (white) | superscript |
| 4 | white (blue) | <u>underline</u> |
| 5 | blinking yellow | dotting |
| 6 | green (white) | <i>italics</i> |
| 7 | blue (light blue) | reversed |
| 8 | yellow (light blue) | - |
| 9 | brown (white) | - |

When selecting colors for display modes, their characteristic differences should be observed. Survo uses them in many special functions (inquiries, touch mode etc.) and relies on the descriptions given above. The easiest way to maintain display colors is to activate **/SURVO-SETUP**.

Alarm

Parameter **alarm** gives the time in the form **09:15:30** for an alarm. When the clock of the computer reaches this time point, the user is alarmed by a beep and by a message

ALARM!!! Press # .

The message stays blinking until the user presses the # key. Survo alarms only when it is updating the clock on the top line of the edit field. Thus, an alarm may be delayed when a long Survo function is operating.

Break sucro

Parameter `break_sucro=<pathname of a sucro>` gives the name of a sucro which is called whenever a sucro is interrupted by the `[.]` key. Break sucros are intended for Survo installations used for demonstrations only. In such applications, the break sucro leads the user automatically to the main menu of the application, for example.

Start sucro

Parameter `start_sucro=<pathname of a sucro>` specifies a sucro which initiates the Survo session on the current installation. Example: `start_sucro=SURVO-START` selects the menu-based start for Survo.

Colors in screen graphics

`crt_palette` gives the default color palette in screen graphics. To obtain true colors when Survo is run under Windows, enter `crt_palette=NUL`. If this parameter is missing, the original Survo palette is selected. Other alternatives are `crt_palette=<file>` where `<file>` is a palette file name with default extension `.PAL` and path `.\SYS`.

Example: `crt_palette=VGAGRAY.PAL` (gives shades of gray on the VGA screen). Palette files are created and edited by the command `GLOT /PALETTE <initial_file>, <new_file>`.

Changing the system parameters

The simplest method for making changes in system parameters is to use the `/SURVO-SETUP` sucro. It can be called also from the main menu of Survo. Of course, experienced users make modifications directly by editing file `SURVO.APU` by `LOADP` and `SAVEP` commands, for example.

The system parameters may be reset during the Survo session according to another system file which has same structure as `SURVO.APU`. This takes place by the `SETUP` command (see `SETUP?`).

To provide separate environments for various users on the same Survo installation, each of them may have a system file of the `SURVO.APU` type. For example, the user `SM` could have a file `SM.APU` for this purpose. Furthermore he/she could create a start sucro (say `/SM`) in the Survo directory which among other things activates `SETUP C:SM.APU`. If now Survo is called by `SURVO SM` from the operating system, the start sucro `/SM` will be started immediately and it creates the environment needed. (See also 12.11).

Temporary changes

The command

SYSTEM <parameter>=<value>

changes values of certain system parameters. The changes are valid in the current Survo session only. Accepted parameters are e.g. `speclist`, `specmax`, `scale_check`, `accuracy`, `results`. Example: `SYSTEM accuracy=16`.

Also macros denoted by `M<character>=<text>` can be changed and more such macros defined. Example: `SYSTEM M8=PRINT CUR+1,END`

Appendix 3

Sucro codes and statements

Key codes

| | |
|------------|--|
| {R} | ENTER (also a line feed in program listing) |
| {home} | HOME |
| {pgdn} | PgDn (next page) |
| {pgup} | PgUp (previous page) |
| {erase} | ERASE (ctrl-END) |
| {ins} | INSERT |
| {del} | DELETE {del3} is same as {del}{del}{del} |
| {ins line} | Insert mode (alt-F9) |
| {del line} | Delete line (alt-F10) |
| {act} | ESC (activation of a Survo command) |
| {r} | Right arrow {r3} same as {r}{r}{r} |
| {l} | Left arrow {l3} same as {l}{l}{l} |
| {u} | Up arrow {u3} same as {u}{u}{u} |
| {d} | Down arrow {d3} same as {d}{d}{d} |
| {()} | Characters { and } |
| {sp} | Space character |
| {tab} | TAB |
| {line end} | END |
| {help} | HELP (F1) |
| {pre} | PREFIX (F2) |
| {touch} | TOUCH (F3) |
| {disk} | DISK (F4) |
| {form} | FORM (F5) {form3} same as {form}{form}{form} |
| {merge} | MERGE (F6) |
| {ref} | REF (F7) |
| {exit} | EXIT (F8) |
| {words} | WORDS (alt-F2) |
| {copy} | COPY (alt-F3) |
| {block} | BLOCK (alt-F4) |
| {search} | SEARCH (alt-F5) |
| {file act} | Activation of a Survo data file (alt-F6) |
| {code} | CODE (alt-F7) |

Control codes and statements

| | |
|-------------|---|
| {end} | terminates the program. |
| {init} | performs initialization of certain display parameters. It guarantees, for example, that insert mode is not on, the reference point is cancelled, the first column is set to the point of return when ENTER is pressed, no blocks and shadow lines are being defined, etc. {init} neither affects the position of the cursor nor changes the contents of the edit field. |
| {break off} | prevents the effects of keys . + and - used in the sucro run for speed control and stopping. Used in keyboard sucros. |
| {break on} | cancels the effect of {break off}. |

| | |
|--------------------------|--|
| {message}<string>@ | displays a message <string> on the bottom line. To cancel the message (this must be done always before normal termination of a sucro), use {message}@ . |
| {message Wi} | displays a message contained in Wi . |
| {message shadow <char >} | sets the shadow character (color) used in messages. Default is {message shadow 1} (red). |
| {wait <time>} | sets a wait of <time> in 0.1 seconds before continuing. |
| {tempo <speed>} | sets the basic speed of the sucro. Default is {tempo 2}. {tempo 1} is the fastest possible. {tempo -1} increases the speed by 1 and {tempo +1} slows it down by 1. Normally only these relative forms should be used. |
| {keys i} | (i=0,1,2) indicates whether the labels of the keys 'pressed' by the sucro are displayed. {keys 0} (default) no display {keys 1} key labels are displayed {keys 2} the user is requested to press the key displayed before proceeding. |
| {next word} | finds the next 'word' on the visible part of the current line. |
| {next col} | works as {next word} but finds also words outside the visible part of the current line. |
| {ref set i} | defines the current cursor/screen position as an additional reference point i (i=1,2,...,8). |
| {ref jump i} | returns the cursor to the point defined by {ref set i}. |
| {ref del i} | removes the additional reference point i. |
| {disp off} | disables the screen. Text is written normally in the edit field but not shown on the screen. |
| {disp on} | restores the normal screen display. |

Useful key combinations in sucros

| | |
|---------------|--|
| {line start} | {pre}B moves the cursor to the start of the line. |
| {next word} | {pre}W finds the next 'word' on the current line. |
| {find <char>} | {pre}C<char> finds the next occurrence of a character on the line. |
| {pre}J | continues an incomplete word automatically. |
| {pre}{pre} | moves the cursor to the end of visible part of the line. |
| {pre}{R} | selects the current column where to return by ENTER. |
| {pre}S | displays the shadow line / closes the shadow line. |
| {pre}P | picks the current character to be written by CODE key. |
| {pre}I | cancels the insert mode in typing. |
| {pre}D | cancels the reference point set by {ref} . |

Sucro memory statements

| | |
|--------------------------|--|
| {stack char} | saves the current character (touched by the cursor) to the end of the sucro memory. |
| {save char Wi} | saves the current character in Wi. |
| {stack word} | saves the current word (touched by the cursor) to the end of the sucro memory. |
| {save word Wi} | saves the current word in Wi. |
| {stack cursor} | saves the number of the current edit line and the number of the current column as two last items in the sucro memory. |
| {save cursor Wi,Wj} | saves the number of the current edit line in Wi and the number of the current column in Wj. |
| {stack corner} | saves the number of the first visible edit line and the number of the first visible column as two last items in the sucro memory. |
| {save corner Wi,Wj} | saves the number of the first visible edit line in Wi and the number of the first visible column in Wj. |
| {stack dim} | saves the number of lines and number of columns of the current edit field as two last items in the sucro memory. |
| {save dim Wi,Wj} | saves the number of lines of the current edit field in Wi and the number of columns in Wj. |
| {stack line} | saves the current line from to the right from the cursor to the end of the sucro memory. |
| {save line Wi} | saves the current line from to the right from the cursor in Wi. |
| {stack datapath} | saves the Survo data path (displayed on the header line of the edit field) to the end of the sucro memory. |
| {save datapath Wi} | saves the Survo data path in Wi. |
| {stack default datapath} | saves the default data path given by edisk in SURVO.APU to the end of the sucro memory. |
| {stack tempdisk} | saves the path for temporary files in Survo operations given by tempdisk in SURVO.APU to the end of the sucro memory. |
| {save tempdisk Wi} | saves the path for temporary files in Wi. |
| {stack systemdisk} | saves the Survo system disk designation (typically C:) to the end of the sucro memory. |
| {save systemdisk Wi} | saves the Survo system disk designation in Wi. |
| {save systempath Wi} | saves the Survo system path designation (typically C:\E\) in Wi. |
| {del stack} | clears the entire sucro memory. |
| {del stack Wi} | clears the sucro memory from Wi onwards. |
| {get key} | appends the next key pressed by the user to the sucro memory. In case of a special key, the sucro memory is appended by 'SK' and the function of the key executed. |
| {get key Wi} | puts the next key pressed in Wi. |
| {save time Wi} | puts the current time (in milliseconds) elapsed from the start of the Survo session in Wi. |

| | |
|-------------------------------------|--|
| {save stack} | saves the entire sucro memory temporarily. |
| {load stack} | appends the sucro memory saved by {save stack} to the end of the current sucro memory. Thus, to replace the current sucro memory by the saved one, use {del stack}{load stack} . |
| {save stack <file>} | saves the current sucro memory in file <file> on the path of Survo temporary files (defined by tempdisk in SURVO.APU) |
| {load stack <file>} | replaces the current sucro memory by the one saved earlier by {save stack <file>}. Please, note the difference to {load stack}; no {del stack} is needed. The file name <file> can be given literally as {save stack TEMP1} or through a sucro memory member as {save stack W1} . |
| {save field <file>} | saves the current edit field on the disk. Default path is defined by tempdisk in SURVO.APU. |
| {print Wi} | prints the contents of Wi in the edit field as a string. |
| {write Wi} | writes the contents of Wi in the edit field as a string. The 'write' command types text as if it were written by the user (i.e. insert and display modes, end of line, etc. are observed) while 'print' simply sets down text regardless of the display status. |
| {find Wi} | finds the next occurrence of the first character of Wi on the current line. |
| {find string Wi} | finds the next occurrence of the string saved in Wi on the current line. Wi can be replaced by a literal string. |
| {pre}l<key> | appends the label of the key to the sucro memory. For example, {pre}l{home} appends 'HOME'. |
| {jump Wi,Wj,Wk} | moves the cursor to line Wj and column Wk. The line Wi is displayed as the first visible one. Each of Wi,Wj,Wk can be replaced by an integer constant. |
| {jump Wi,Wj,Wh,Wk} | works as jump above but displays Wh as the first visible column. |
| {save system <system word> Wi} | saves the system parameter defined by <system_word> in the Survo system file SURVO.APU in Wi. Example: {save system accuracy W3} saves the 'accuracy' parameter in W3. |
| {save spec <specification_word> Wi} | saves the value of the specification given in the current edit field by <specification_word> in Wi. If the specification doesn't exist, an empty string {} is saved. Example: {save spec DEVICE W4} saves the current value of the DEVICE specification in W4. |

Labels and conditional statements

- + Part2: <sucro code continued>
with '+' in the control column defines label 'Part2'.
The colon ':' and the space after it belong to the label notation.
- {goto Part2} Go to the place in the sucro code indicated by label 'Part2'.
- {load Cont} Continue with another sucro 'Cont'.
The contents of the sucro memory and other system parameters are preserved.
- if <condition> then goto <label>
(character - in the control column) continues the program from the point labelled by <label> if the condition is true. Otherwise the program continues from the next statement.
- if <condition> then load <another_sucro>
if the condition is true, the program continues by another sucro.

Also an extended form with an 'else' clause is available.

- switch Wi
- case <value1>: <goto label | load sucro | continue>
- case <value2>: <goto label | load sucro | continue>
- etc.
- default: <goto label | load sucro | continue>
branches the sucro according to the value of Wi to one of the alternatives given on the case lines.

User interaction

- prompt <Question ?> { }
- default <default answer>
- answer <memory location for the answer (this line is optional)>
- length <max. length of the answer>
- wait <max. waiting time>
prompts the user to give answer to a question.
- on key
- key <key1>: <goto label | load sucro | continue>
- key <key2>: <goto label | load sucro | continue>
- etc.
- wait <max. waiting time>
waits until the user presses a key and branches according to the key pressed.

- {interaction on} when given before activating an interactive Survo operation (like FILE SHOW) releases the function of that operation from the sucro control. The user has a possibility of working on his/her own premises. On return to the editorial mode, the sucro continues normally.
- {interaction off} resumes the automatic execution mode of the sucro. Then also interactive work modes of Survo (like FILE SHOW) are controlled by the sucro. This is the default setting.

Arithmetics

- {Wi=Wj} Contents of Wj is copied to Wi.
 {Wi=Wj+Wk} Addition
 {Wi=Wj-Wk} Subtraction
 {Wi=-Wj} Changing the sign
 {Wi=Wj*Wk} Multiplication
 {Wi=Wj/Wk} Division
 {Wi=Wj%Wk} Remainder in integer division
 {Wi=Wj&Wk} Concatenation of strings
 Constants can also appear as operands.

Calling other sucros

- {load <sucro>} the current sucro is followed by another. The contents of the sucro memory as well as values of system parameters are preserved.
- {call <sucro>} calls another sucro as a subroutine.

Error control

- {error handler <sucro>} selects the error handler sucro.

Appendix 4

Installation of SURVO 84C

1. Equipment

SURVO 84C runs on the IBM compatible PC's with a memory of 640KB, a hard disk and an EGA or VGA display. The math co-processor is very useful although it is not absolutely necessary.

Any PostScript printer with the Centronics (parallel) interface and the Canon LBP-8 laser printers are supported in making complete reports with graphical illustrations. In standard printing without graphics, several other printers may be employed.

For Survo printouts, each printer requires a special device driver. It is a text (ASCII) file (with extension `.DEV`). The user may create new device drivers on the basis of ready-made ones.

2. Operating system

MS-DOS 3.2+

3. Disk space

The complete version of SURVO 84C needs about 12MB of disk space on the hard disk. Before installation, check that there is enough free space by the `DIR C:` command, for example.

All Survo program and system files will be saved in directory `\E` and in its subdirectories. The directories needed will be automatically created during the installation process.

4. Laser printers

The standard version of SURVO 84C supports in printing and plotting either PostScript printers (the main alternative) or the Canon LBP-8 printer with vector graphics.

PostScript

To have a direct access to a PostScript printer, the Centronics (parallel) interface should be used. Please, observe that the printouts can also be redirected to any text file (extension `.PS` is recommended). Thereafter the actual printout may take place simply by the MS-DOS command `COPY` (in parallel interface) or by any telecommunication routine (in serial interface).

The standard device driver for PostScript printers is `PS.DEV`.

Canon

Either Canon LBP-8 A2 or LBP-8 II or newer (with 1.5MB memory) can be used. The printer has to be connected to the computer by the parallel (Centronics) interface.

On the A2 model, the DIP switches at the rear of the printer have to be in the positions suggested by the printer manual except SW3-1 which must to be ON (ISO mode). Thus in the swith boxes SW3 and SW4 (totalling 16 switches) all except SW3-1 and SW4-5 are OFF.

However, switches SW3-5,6,7 selecting the country graphics set can be in any positions in SURVO 84C applications, since language used will be controlled by the Survo printer driver (CANON.DEV or CANONG.DEV, for example).

Survo works in the ISO mode and not in the Diablo mode, since the latter does not allow vector graphics. Various font cartridges may be attached. Currently the most suitable is the Garland PS N cartridge which is supported by the CANONG.DEV driver and provides characters in proportional pitch.

On the LBP-8 II model two font cartridges can be used. A typical arrangement is the Garland PS N on the right side and either the Dutch 801 or the Swiss 721 cartridge on the left side. The CANON2.DEV driver supports this setup.

5. Other printers

Other printers are supported in text mode only. Currently, following drivers are available:

| | |
|--------------|------------------------------|
| EPSON.DEV | for Epson F80 and compatible |
| PROPR.DEV | for the IBM Proprinter |
| LASERJET.DEV | for the HP Laserjet |
| DESKJET.DEV | for the HP Deskjet |

6. Copying SURVO 84C to the hard disk

Before starting the installation, make sure that you have an up-to-date backup of the hard disk. If something goes wrong in the Survo installation, you can restore the original situation on the hard disk by this backup.

SURVO 84C is copied from the diskettes to the hard disk as follows:

- 1) On the DOS command level, select disk drive A:
- 2) Insert diskette SUR0 into drive A:
- 3) Give command SURVOA .

At first, the installation program SURVOA prompts the user to check that the conditions for the Survo installation are satisfied.

The installation process will then continue according to the prompts appearing on the screen. The diskettes must be inserted in the order suggested. If any errors occur, the whole process has to be restarted.

The diskettes (SUR) used in the installation are: 0,1,2,3,4,5,6,7,Q1,Q2,D.

SURVOA without parameters installs SURVO 84C in the \E directory on the C: disk and in its subdirectories. All directories are automatically created.

Survo can also be installed on other disks or in other directories by giving the SURVOA command in the form SURVOA <path> (for example, SURVOA D:\E).

7. Software key

SURVO 84C is a protected system. It cannot be used without a software key. Two alternative keys models are available.

Model "Software key"

The key is to be plugged into the RS-232 serial connector (COM1 or COM2). No key diskettes are required during SURVO 84C sessions.

If the software key is connected to COM2, the SUR0 installation disk has to be inserted into drive A: and the command

```
A:SURVOKEY C:\E (C:\E is the Survo path)
```

has to be given. This changes the software key port from COM1 to COM2 and vice versa.

Since the far end of the software key has a 25 pin connector, other serial devices which are normally connected directly to the computer can be connected instead to the far end of the key. It has been noticed, however, that in some cases the other devices connected to the computer via the software key may prevent entering Survo. In such a case the other device must be disconnected before trying again.

When using the software key without other devices plugged to its far end no difficulties have been reported.

Model "Microphar key"

The key is plugged into any of the printer (parallel) ports. A printer can be connected to the far end of the key. The system finds the key automatically. No extra settings are needed.

8. Entering SURVO 84C

After copying the diskettes, Survo will be ready for use and it can be entered from the root directory by the command SURVO. The exit from Survo always takes place by the function key F8.

It is the batch file SURVO.BAT which is activated when the SURVO command is given. This file should remain in the root directory of the hard disk permanently if there is no aim to alter the method for calling Survo.

When changing this initialization process, it must be observed that before entering SURVO 84C the directory \E has to be selected (by CD \E) and then the Survo main module S has to be called (by S).

9. Backup of SURVO 84C

After installation, it is good to make a backup of Survo (directory \E and all its subdirectories). This can be done easily by the DOS command
 BACKUP C:\E A: /S, for example.

10. Deleting SURVO 84C system files

The entire SURVO 84C system is deleted from the hard disk as follows. Insert the installation disk SUR0 in drive A: and give the command
 A:SURVODEL C:\E (C:\E is the Survo path).

The user is prompted to give permission for deletion of files in each subdirectory separately. Finally, all empty directories will be removed.

11. Subdirectories

All the SURVO 84C system files are located in subdirectories of the Survo system path .\ (usually C:\E) as follows:

| | |
|-------|--|
| .\ | Editor and most of the modules |
| .\SYS | .SYS, .BIN, .DEV files |
| .\D | Data path for SURVO 84C demonstration data |
| .\TUT | Tutorials and sucros |
| .\Q | Inquiry system files |
| .\F | Library functions (editorial computing) |
| .\M | MATRUN files |
| .\FI | FILE operations |
| .\TAB | TAB operations |
| .\TBL | Tables of percentiles, etc. |
| .\L | LIST operations |
| .\P | PLOT operations |
| .\G | GPLOT operations (screen graphics) |
| .\PS | PLOT operations (PostScript) |
| .\H | PLOT operations (HP plotters) |
| .\S | Sucros |

12. Updating SURVO 84C

Updated versions of SURVO 84C are installed in the same way as the first version. Observe, however, that all system files in and in its subdirectories will be overwritten by the files of the new version. Therefore, if the user has changed the SURVO.APU file, for example, it is good to make a spare copy of it before the new installation. After the installation, the new SURVO.APU file can be edited according to the user's own file.

If there is enough space on the hard disk, several versions of Survo, even with different options, can be maintained simultaneously by installing them on different disks or paths.

13. Installation of SURVOS

The reduced version SURVOS is installed in the same way. The installation diskettes are SURVOS A, B, C, S, Q1, Q2. The default path for the SURVOS system is C:\SURVOS.

14. Installation of the network version

1) Install SURVO 84C in the server normally. It is not copy-protected. Thus no software key is needed. Any disk/path can be selected for SURVO 84C. In the following, we assume that the path is G:\E (as seen from the workstations).

2) Start SURVO 84C on the server and make adjustments to the system files on the diskette "SURVO 84C Network Update" by loading the edit field A:\SYS\UPDATE.EDT and following instructions given in that field.

3) Copy system files by the MS-DOS batch command A: SURVON .

4) Certain network systems (e.g. PC Lan and Lan Manager) do not allow several users to keep a file open at the same time although it is opened in read-only mode. In those environments, certain Survo sub-directories must be set into read-only mode by the ATTRIB command of DOS.

This should be done at least for following directories:

('.' indicates the Survo system directory on the server.)

```
.\Q    Help system
.\S    Sucro tools
.\M    Matrix chains
```

For example, the help system is set into read-only mode by

```
ATTRIB .\Q\*. * +R .
```

This extra task is not needed in Novell Netware, for example.

5) This step is necessary only when there is no hard disk on the workstation. For each workstation using SURVO 84C:

Assume that the path for temporary files (unique for this work station) is F:\WS1 .

a) Create a unique copy of the system file SURVO.APU having the edisk, tempdisk and eout lines as follows:

```
edisk=F:\WS1\SURVO\D\
tempdisk=F:\WS1\SURVO\
eout=F:\WS1\SURVO\RESULTS
```

Do not change other lines in SURVO.APU. The network update diskette contains a ready-made SURVO.APU file of this form. Save the modified SURVO.APU file on the path F:\WS1 .

- b) To use SURVO 84C from this workstation, create a SURVO.BAT file of the form (a model on the network update diskette):

```
G:  
CD G:\E  
MD F:\WS1\SURVO  
MD F:\WS1\SURVO\D  
MD F:\WS1\SURVO\TUT  
COPY G:\E\CBLOCK.EXE F:\WS1\SURVO  
S /S:F:\WS1\SURVO.APU
```

and save it on path F:\WS1 .

Then, SURVO 84C can be started on this workstation by selecting the path F:\WS1 and by giving the command SURVO.

Appendix 5

Survo keywords

Various names of commands, specifications, and general terms related to Survo are listed alphabetically. Types of terms are given in parentheses.

This index is not exhaustive. For example, most of the specification words typical for particular Survo operations are not shown. These keywords can be found in the description of the operation in question.

More information is obtained by using keywords of the inquiry system of Survo. Such keywords appear at the end of lines. For example, the reference at the end of the line

- Sound effects in data browsing (FSHOW?:6)

means that more information on the topic will be seen by activating FSHOW? and by selecting item 6 from the list of alternatives.

The functions which are valid also in the reduced SURVOS version are indicated by bullets (•) in the left margin.

When using the Survo system, this list of keywords is available as a hyper-text from the main menu or by activating command /SURVO-WORDS . Words in the list are then activated by the keys **F2** **F1** and the inquiry system of Survo tells more about the activated word.

- *GLOBAL* (definition) for global specifications in the edit field
- /23 (sucro) selects the 23 line window (default) for the edit field (VGA only).
- /48 (sucro) selects the 48 line window for the edit field (VGA only).
- ACCURACY (specification) gives the precision of results in editorial computing.
Aggregation -> FILE AGGRE (AGGRE?)
- /ALARM (sucro) sets an alarm time.
Analysis of covariance (ANOVA?)
- Analysis of variance
 General (ANOVA? GENREG? TABFIT?)
- one-way (COMPARE?)
- one-way, distribution-free (COMPARE?)
Anderson-Darling-test for normality (COMPARE?:3)
- Andrews' function plots (ANDREWS?)
- Angles (measurement) (CONV?)
ANOVA (operation)
 a wide set of methods related to analysis of variance
 and covariance programmed by Markku Korhonen (1989).
- Area measures (CONV?)
- Arithmetics (ARIT? TOUCH?)
- Arrow keys (KEYS2?)
 Auto- and crosscorrelations (AUTOCOR?)
- BACKGROUND (specification) defines the background color combination (GPLOT).
- Bar charts (BAR?)
- Clipping of bars (MINVALUE?)
- Colors and shades of gray (SHADING?)

- Gaps between bars (GAP?)
- Legends (LEGEND?)
- Outlines of pies (PIEBORDER?)
- Setting of items in pie charts (PLAN?)
- Texts (LABELS?)
- Values (VALUES?)
- Basic statistics (STAT?)
 - Binomial coefficients (MATHF?)
 - Binomial distribution -> Distributions
- BINORM (specification) defines a bivariate normal distribution whose
 - contour ellipses are plotted by a CONTOUR specification.
- BLOCK (key alt-F4) defines and moves rectangular blocks in the edit field.
- /BOX (keyboard sucro) for writing of box graphics in the edit field
- /BUFFON (sucro) simulates Buffon's needle experiment.
- /C (key sucro) converts lowercase letters to uppercase.
- C operations (C+,C*,C-,C/,C%,Cfunction) compute new columns in the edit field.
- Calling other programs within a Survo session (CHILD?)
 - CANON (operation) canonical correlations
 - Canonical correlations (CANON?) -> MATRUN
- CASES (specification) for selecting of observations
- CHANGE (command) exchanges columns or rows in a table.
- Chernoff's faces (CHERNOFF?)
- Chi-square test (TAB? HISTO?)
 - /CHI2 (sucro) Chi-square test for a table of frequencies (by L.Tarkkonen)
- CHILD (command) spawns other programs as child processes.
 - Cholesky decomposition (MATDEC?)
- Classified distributions (HISTO? TAB? STAT?)
 - CLASSIFY (operation) makes new variables according to a given classification.
- CLEAR (command) clears a part of the edit field.
 - CLUSTER (operation) cluster analysis
- CODE (key alt-F7) for typing characters not available in the keyboard
 - Code conversions (CONVERT? CONV? CODES?)
 - CODES LOAD (command) for reading (binary) files in sequences of 256 bytes.
 - CODES SAVE (command) for writing (binary) files in sequences of 256 bytes.
 - COLOR (command) for selecting colors in the text display
- COLORS (specification) defines color/background color (GPLOT)
- Colors
 - Background colors in screen graphics (BACKGROUND?)
 - Color palette in screen graphics (PALETTE?)
 - Colors in screen graphics (COLORS?)
 - Edit field (COLOR?)
 - Filling of areas (FILL? SHADING?)
 - PostScript printers (PS?:K)
 - Screen graphics (CRT?)
 - Variable colors in families of curves (COLORCH?)
- COLOR_CHANGE (specification) for selecting variable colors in curve plotting
- Commands and operations (OPER?)
 - Hiding a command (HIDDEN?)
 - Matrix chains (MATRUN?)
 - Operation sequences PREFIX ESC (PREFIX?)
 - Specifications (SPEC?)
- COMPARE (operation) statistical tests of samples
 - /COMPARE (sucro) for automatic comparison of samples
- Comparison of samples (COMPARE?)
- /CONFMEAN (sucro) studies the confidence interval of the mean by simulation.
- CONTOUR (specification) defines contour ellipses in scatter plots.
- Contour ellipses (CONTOUR?)
 - Contour plots (PLOTCONT?)
 - CONVERT (command) for code conversions in the edit field

- COPY (command) for copying lines in the edit field
- COPY (key alt-F3) for copying of a single line
- CORR (operation) means, standard deviations and correlations
- Correlation diagrams (SCAT? DRAFTS?)
 - Connecting of consecutive points (LINE?)
 - Contour ellipses (CONTOUR?)
 - Dislocation of points (LAG2?)
 - Point types (POINT?)
 - Trends (TREND?)
- Correlation coefficients
 - Auto- and crosscorrelations (AUTOCOR?)
 - Computing of a correlation matrix (CORR?)
 - Kendall's Tau (COMPARE?)
 - Spearman's Rho (COMPARE?)
 - Cosine rotation in factor analysis (ROTATE?)
- COUNT (command) for systematic and cyclic numbering of lines in a table.
- Cross tabulation (TAB? AGGRE? MTAB?)
- Cumulative sums (SER? VAR?)
- Currencies (CURRENCY?)
- Curve plotting (CURVES?)
 - Contour plotting (PLOTCONT?)
 - Integral functions (CURVES? INTEGRAL?)
- D'Agostino's test for normality (COMPARE?)
- DATA (definition) for data lists and data tables in the edit field
- Data files (FILE?)
 - Activation of fields (FACTIV? AF6? MASK? VARS?)
 - Aggregation (AGGRE?)
 - Copying data files to the edit field or to a text file (FLOAD?)
 - Copying data from text files (FSAVE?)
 - Copying of data sets (FCOPY?)
 - Creating a data base (FCOPY? FCREATE?)
 - Data input and editing (FSHOW? FIEDIT?)
 - Data transformation -> Variables (statistical)
 - Definition of new fields (variables) (FUPDATE? VAR?)
 - Deletion of a data file (FILE?:D)
 - Empty records (inserting) (FINIT?)
 - Listing of the structure of a data file (FSTATUS?)
 - Protection of data (FACTIV? AF6?)
 - Reduction of a data file (FILE?:C)
 - Scales of measurement (SCALES?)
 - Search for data (FSHOW? FIEDIT?)
 - Sorting (FSORT?)
 - Transformation of string variables (VARSTR?)
- Data representation (DATA?)
 - /DATE (sucro) writes the current date in the form 'September 19, 1992'.
 - DEF (definition) for text chapters in the edit fields
 - DELETE (command) deletes columns etc. from the edit field.
 - DELETE (key DEL or F10) deletes a character.
 - /DELLIN (sucro) deletes lines with a given character in the control column.
 - DER (command) analytic derivatives of functions
 - Desktop publishing -> Printouts (PRINT? PS?)
 - DEVICE (specification) defines the device for plotting.
 - Device drivers (DEVICE?)
 - Differences (SER?)
 - DIR (command) lists the names of the files in the edit field.
 - /DIR (sucro) lists names of files according to their types (by M.Karpoja)
 - Discriminant analysis (DISCR?) -> MATRUN
 - DISK (command) selects the data disk/path of Survo.
 - DISK (key F4) selects the next data path from predetermined alternatives (PATH?)

- Disk drive selection (DISK? PATH?)
- Distributions
 - Cumulative distribution functions (FUNCSTAT?)
 - Density functions (FUNCSTAT?)
- DOS (command) invokes commands of the operating system.
- Durbin-Watson test statistics (LINREG? REGDIAG?)
- Edit field -> Text editing
- Editorial computing (ARIT?)
- Eigenvalues and vectors (MATDEC?)
- END (key) moves the cursor to the end of the line.
- Entropy (STAT?)
- ERASE (command) erases selected characters from the edit field.
- ESTIMATE (operation) non-linear regression analysis and general ML estimation
- Exchange rates (CURRENCY?)
- EXIT (key F8) for exit from Survo
- F test (COMPARE? ANOVA?)
- FACTA (operation) primary solutions for factor analysis
- Factor analysis (FACTOR?)
 - Automatic factor analysis -> /FACTOR
 - Primary factors (FACTA?) -> MATRUN
 - Rotation (ROTATE?)
 - Factor scores (FACTOR?) -> MATRUN
- /FACTOR (sucro) combines different stages of factor analysis automatically.
- FIELD (specification) for displaying edit field in graphs (GPLOT?:S:5)
- FILE operations for management of Survo data files
 - FILE ACTIVATE Activation of fields (FACTIV?)
 - FILE AGGRE Aggregation of records (AGGRE?)
 - FILE CREATE Creation of files (FCREATE?)
 - FILE INIT Initializing with empty records (FINIT?)
 - FILE COPY Copying and merging of data files (FCOPY?)
 - FILE SAVE Copying text files to data files (FSAVE?)
 - FILE LOAD Copying data files to the edit field or to text files (FLOAD?)
 - FILE SHOW Data saving, browsing, searching and editing (FSHOW?)
 - FILE EDIT Data saving, browsing, searching and editing (FEDIT?)
 - FILE SORT Sorting (FSORT?)
 - FILE STATUS Listing the structure of a data file (FSTATUS?)
 - FILE UPDATE Updating the structure of a data file (FUPDATE?)
 - FILE REDUCE Reducing dimensions of a data file (FILE?)
 - FILE DEL Deleting a data file (FILE?)
- Files
 - -> Data files
 - -> Text files
 - Edit fields -> Text editing
 - Intermediate results in matrix files (MATRES?)
 - Matrix files (MAT?)
 - Output files (OUTPUT?)
 - Picture files (GPLOT?)
 - PostScript files (PS?)
- FILL (specification) for filling areas in plotting
- Fill colors (FILL? SHADING?)
- FIND (command) for searches in the edit field
- Footnotes in publications (FOOTNOTE?)
- Force measures (MEASURES?)
- FORECAST (operation) automatic forecasting of periodic time series
- FORM (command) for formatting of tables in the edit field
- Fractiles (STAT?)
- FRAME (specification) for frames around graphs
- FRAMES (specification) for frames and boxes in graphics
- Frequency distributions (HISTO? TAB? STAT?)

- Function keys (F?)
 - F1: HELP starts an inquiry (HELP?).
 - F2: PREFIX prefix key (PREFIX?) -> PREFIX
 - F3: TOUCH enters the touch mode (TOUCH?).
 - F4: DISK selects the next data disk/path (PATH?).
 - F5: FORM selects the next display mode in the edit field (SHADOWS?).
 - F6: MERGE mergers/splits lines in the edit field (MERGE?).
 - F7: REF maintains a reference point in the edit field (REF?).
 - F8: EXIT exit from Survo
 - F9: INSERT toggles the insert mode (F9?).
 - F10: DELETE deletes the current character (F10?).
 - alt-F1: not in use
 - alt-F2: WORDS moves text in the edit field (WORDS?).
 - alt-F3: COPY LINE copies lines (AF3?).
 - alt-F4: BLOCK copies rectangular blocks (BLOCK?).
 - alt-F5: SEARCH for sequential search for text in the edit field (SEARCH?)
 - alt-F6: FILE ACT for activating fields in a Survo data file (FACTIV?)
 - alt-F7: CODE for typing characters not available in the keyboard
 - alt-F8: not in use
 - alt-F9: LINE INS inserts an empty line (AF9?).
 - alt-F10: LINE DEL deletes the current line (AF10?).
- GAP (specification) sets the gap between bars in graphs.
- Generalized linear models (GENREG? TABFIT?)
 - GENREG (operation) generalized linear models
- GHISTO (operation) histograms and fitting univariate models
- GOTO (command) moves the cursor to a given point.
- GPLOT (operation) -> Plotting of graphs on the screen
 - Gram-Schmidt decomposition (MATDEC?)
 - Graphical rotation in the factor analysis (ROTATE?)
- Greatest common divisor (GCD?)
- GRID (specification) for drawing of grid lines in graphs
- /H (key sucro) splits and merges parts of words at the end of an edit line.
- HEADER (specification) for the header of a graph
- HELP (command) initiates the help system.
- Help system (HELP?)
- HISTO (operation) histograms and fitting univariate distributions
- Histograms (HISTO? STAT?)
 - Fitting of standard distributions (HISTO?)
 - Fitting user-defined distributions (HISTO2?)
 - Plotting of a histogram (HISTO? STAT?)
- /HLIST (sucro) for scanning and searching for keywords of the inquiry system
- HOME (key) returns the cursor step by step to the start of the edit field.
- HOME (specification) for the location of a graph
- IFILL (specification) for filling an area from the starting point of a curve
- IND (specification) for selecting of observations
 - Indexes in publications (INDEX?)
- INFILE (specification) sets a ready-made picture as a background (GPLOT?).
- INIT (command) redimensions the edit field and initializes it.
- Inquiries (HELP?)
 - Changing the inquiry system (QPATH?)
- INSERT (command) inserts more empty columns in the edit field.
- INSERT (key INS or F9) toggles the insert mode.
- INTEGRAL (specification) normalizes a function according to its integral (PLOT?)
- INTERP (command) interpolates columns by polynomial regression.
- Keyboard
 - -> Function keys
 - -> PREFIX
 - Codes (CODE?)
- Keywords in publications (INDEX?)

- Kolmogorov-Smirnov test (COMPARE?)
- Kruskal-Wallis test (COMPARE?)
- Kurtosis (excess) of a distribution (STAT?)
- /L (key sucro) converts uppercase letters to lowercase.
- L-operations (L+,L*,L-,L/L%,Lfunction) compute new rows in tables.
- LABELS (specification) writes names of variables in bar charts.
- /LABELS (sucro) conceals the line numbers of the edit field.
- Lags (LAG?)
- Leads (LAG?)
- LEGEND (specification) for legends of shadings and colors in graphs.
- Length measures (CONV?)
- Library functions (FLIBR?)
- LIMITS (specification) gives the limits of shadings in matrix plots.
- LINCO (operation) linear combinations of variables
- LINE (specification) defines the type and the thickness of a line in graphs.
- Line graphs -> Correlation diagrams
- Linear models -> Regression analysis
- Linear optimization (SIMPLEX?)
- LINES (specification) for plotting of line segments.
- LINREG (operation) linear regression analysis
- LIST (definition) for texts in several edit fields
- LIST (operations) for management of long reports and text lists
 - LIST SHOW Displaying a list, searches
 - LIST REPLACE Replacing a string by another in a list
 - > /HLIST
 - > /LIST
- /LIST (sucro) for browsing and editing of text in lists
- LOAD (command) loads a ready-made edit field from an edit file.
- LOADM (command) writes a matrix file in the edit field.
- LOADP (command) writes a text file in the edit field.
- Log-linear models (TABFIT? GENREG?)
- Macros
 - Simple text macros -> System parameters (SYSTEM?)
 - Sucros (SUCRO?)
- Mann-Whitney test (COMPARE?)
- MASK (specification and command) for selection of variables
- MAT (command) matrix interpreter commands
- Mathematical operations (MATH?)
 - > Solving equations
- Editorial computing (ARIT?)
- Formal derivatives (DER?)
 - Linear optimization (SIMPLEX?)
 - Matrix operations (MAT?)
 - Polynomials (POL?)
- Touch mode (TOUCH?)
- MATRIX (definition) labels a matrix in the edit field.
- Matrix diagrams (PLOTMAT?)
- Matrix operations, matrix interpreter (MAT?)
 - Basic operations (MATO?)
 - Copying matrices to data files (MATFILE?)
 - Decompositions (MATDEC?)
 - Element by element transformations (MATTRANS?)
 - Least squares solutions (MATSOL?)
 - Linear equations (MATSOL?)
 - Matrices as data sets (MATD?)
 - Matrix approximations -> MATRUN
 - Matrix programs and chains (MATRUN?)
 - Normalizing and derived matrices (MATNORM?)
 - Printout of matrices (MATLOAD?)

- Row and column labels (MATLABEL?)
- Saving (MATSAVE?)
- Scalars in matrix operations (MATCONST?)
- Submatrices (SUBMAT?)
- Super matrices (MATSUP?)
- MATRUN (commands) for matrix chains
 - MATRUN APPROX? Approximation of a matrix by another with a lower rank
 - MATRUN CANON2? Canonical correlations
 - MATRUN CANON3? Canonical correlations with nuisance variables
 - MATRUN CHI2? Chi-square test (by L.Tarkkonen)
 - MATRUN COMPRESS? Optimal projection of a set points
 - MATRUN DISCR? Multiple discriminant analysis
 - MATRUN FCOEFF Factor score coefficients
 - MATRUN FTCOEFF? Factor score coefficients (after oblique rotations)
 - MATRUN PARTCORR? Partial correlations
 - MATRUN PCOMP? Principal components from a data matrix
 - MATRUN PCOMPR? Principal components from a correlation matrix
 - MATRUN PFACT? Principal axes solution for factor analysis
 - MATRUN REGR? Regression analysis by singular value decomposition
 - MATRUN SUM2? Sums of squares of matrix rows and columns
 - MATRUN SYMTRANS? Symmetric transformation analysis
 - MATRUN TRANS? Transformation analysis (naive model)
- Maximum likelihood estimation (ESTIMATE?)
- Mean numbers (MEANS?)
 - Arithmetic means (STAT? CORR?)
 - Conditional means (TAB? MTAB?)
 - General power means (STAT?)
 - Geometric means (STAT?)
 - Harmonic means (STAT?)
 - Quadratic means (STAT?)
- Measures and their conversions (CONV?)
- Median (STAT?)
- MERGE (key F6) for merging and splitting of lines in the edit field
- MINVALUE (specification) for clipping of bars in bar charts
- MODE (specification) selects the video mode in screen plotting
- MOVE (command) for moving text in the edit field
- Moving averages (SER?)
- Moving the cursor (arrow keys, GOTO?)
- MTAB (operation) multiway tables of means and other statistics (by M.Korhonen)
- Multiple tasks in Survo (CHILD?)
- Multivariate methods (MULTI?)
- Multiway tables (TAB?)
- Non-linear regression analysis (ESTIMATE?)
- Normal distribution
 - Bivariate normal distribution (BINORM?)
 - Plotting on probability paper (PROBIT?)
 - Test for normality (COMPARE?)
- /NSIMUL (sucro) simulates multivariate normal distribution.
- Number systems (conversions) (CONV?)
- Numerical transformations (CONV? VAR?)
- Oblimin rotation (ROTATE?)
- Observations, records
 - Classification (CLASSIFY? AGGRE? CLUSTER?)
- Data definition (DATA?)
- Interpolation (INTERP?)
- Normalizing (VARR?)
- Order numbers (ranks) (VARR?)
- Saving in a data file (FSHOW? FIEDIT?)
- Selection of observations (IND? CASES? SELECT?)

- Sorting FILE SORT or SORT (FSORT? SORT?)
- Standardizing (VARR?)
- Transformations (VAR? SER? CLASSIFY? TRANSFORM? AGGRE?)
- Truncating and Winsoring (VARR?)
- OFILL (specification) fills an area from origin in curve plotting.
- OIO (command) Finnish spelling checker (by K.Koskenniemi)
- Operating system commands (DOS? DIR?)
- Optimal projection of a point set -> MATRUN
- Order statistics (STAT?)
- OUTFILE (specification) saves a graph in a .SPX file (GPLOT)
- OUTPUT (command) selects the output file.
- PALETTE (specification) for selecting the color palette (GPLOT)
- Parameter estimation (ESTIMATE? HISTO?)
- Partial correlations -> MATRUN
- PATH (command) selects the data disk/path of Survo.
- PgDn (key) shows the previous page of the edit field.
- PgUp (key) shows the next page of the edit field.
- Physical measures (MEASURES?)
- Pie charts -> Bar charts
- PIEBORDER (specification) implies outlines to be plotted in pie charts (GPLOT).
- PLAN (specification) for setting of items in pie charts
- PLOT (operation) plotting of graphs
 - Boxes and straight lines (FRAMES? LINES?)
 - Frames (FRAME? FRAMES?)
 - Graphs in reports and publications (PRINT2? PS?)
 - Grid lines (GRID?)
 - Headers (HEADER?)
 - Location of the graph (HOME?)
 - Proportions of the parts of the graph (XDIV? YDIV? RATIO? SIZE?)
 - Saving of graphs and overlays of them (OUTFILE? INFILE? PRINT2?)
 - Scales (SCALE? XSCALE? YSCALE?)
 - Size of the graph (SIZE?)
 - Texts in graphs (TEXTS? XLABEL? YLABEL? HEADER?)
 - Tick lines on the axes (TICK? TICKLEN?)
 - Types of graphs -> TYPE
- PLOT /FRAME (operation) plotting of frames, boxes and line segments
- PLOT /PALETTE (operation) for editing color palettes in screen graphics
- Plotting of multivariate data (PLOTM?)
- POINT (specification) selects the point type and size in scatter plots.
- POL (command) operations for polynomials
- PostScript plotting and printing (PS?)
- Power measures (MEASURES?)
- PREFIX (prefix key F2) (PRE?)
- Pressure measures (MEASURES?)
- Prime factors of integers (CONV?:8)
- Principal component analysis (FACTOR?) -> MATRUN
- PRINT (operation)
 - Chapters to be printed (DEF?)
 - Code conversions (PRINTCOD?)
 - Control codes, definition (PRINTDEF?)
 - Control codes, using (PRINTINC?)
 - Control column (PRINTCON?)
 - Device drivers (DRIVER?)
 - Footnotes (FOOTNOTE?)
 - Keywords, index (INDEX?)
 - Operating system commands in the PRINT list (PRINT2?:O)
 - Page changes (PRINT2?:J)
 - Page headers and numbers (PRINT2?:D)
 - Pictures (PRINT2?:C EPSFILE?)

- Printing in a text file (ASCIITXT?)
- Redirection to another printer or to a file (PRINTTO?)
- Shadow characters (PRINT2?:K PRINTSHA? SHADOWS?)
- Text files (PRINT2?:B)
 - Text to be printed, definition (PRINT2?:A,B DEF?)
- Printouts
 - Accuracy of results (ACCURACY?)
 - Extent of results (RESULTS?)
 - Output control (OUTCNTRL?)
 - Output device/file (OUTPUT?)
- Probability paper (PROBIT?)
 - Procrustes method -> MATRUN SYMTRANS, MATRUN TRANS
 - Profile symbol plots (PROFILES?)
- QPATH (command) changes the active inquiry system.
- Quantiles (STAT?)
- Quartiles (STAT?)
 - Quartimax rotation (ROTATE?)
- Randomized tests (COMPARE?)
- RATIO (specification) sets the pixel (aspect) ratio.
- Ratios (converting a decimal number to a ratio of integers) (CONV?)
 - Records -> Observations
- REDIM (command) redimensions the edit field without changing the contents.
- REF (key F7) sets, selects and deletes a reference point in the edit field.
- REGDIAG (operation) linear regression analysis and regression diagnostics
- Regression analysis
 - Linear regression (LINREG? REGDIAG? MATSOL?) -> MATRUN
 - Non-linear regression (ESTIMATE?)
 - Polynomial interpolation (INTERP?)
 - Regression diagnostics (REGDIAG?)
 - Semiparametric smoothing of data (SMOOTH?)
- Stepwise linear regression -> /STEPREG
- REPLACE (command) replaces a string by another in the edit field.
- RESULTS (specification) gives the extent of results in statistical operations.
- Roman numerals (CONV?)
 - ROTATE (operation) rotation in factor analysis
- /S (keyboard sucro) for typing of shadow characters
- SAVE (command) saves the current edit field.
- SAVEP (command) saves a part of the edit field in a text file.
- Saving of data (FSHOW? FIEDIT?)
- SCALE (specification) gives a scale in graphs.
- Scales of measurement (SCALES?)
- Scatter plots -> Correlation diagrams
- SCRATCH (command) erases all text below the command line.
- SEARCH (alt-F5) starts a sequential search for a string in the edit field.
- Sector plots -> Bar charts
- SELECT (specification) for general rules in selection of variables
 - SER (commands) for management of time series
- SET (command) copies a given text to several lines.
- SETUP (command) for changing of Survo system parameters
- SHADING (specification) for selecting colors and shadings in bar/pie charts
- Shapiro-Wilk test for normality (COMPARE?)
- SHOW (command) for display of edit and text files.
- SIMPLEX (operation) linear optimization
- Simulation
 - Buffon's needle problem (/BUFFON)
 - Confidence intervals for means (/CONFMEAN)
 - Jittering of observations in scatter plots (DRAFTS?)
- Making random deviates (CHANCE?)
- Multinormal distribution -> /NSIMUL

- Factor analysis (/BOXSTONE)
- Singular value decomposition (MATDEC?)
- SIZE (specification) gives the size of a graph.
- Skewness (STAT?)
- SMOOTH (operation) semiparametric smoothing of data
- /SOFTKEYS (sucro) lists macros defined in SURVO.APU .
- Solving equations
 - Algebraic equations (ROOTS?)
 - Linear equations (MATSOL?)
 - Non-linear equations (ESTIMATE?)
- SORT (command) sorts lines of a table in the edit field.
- Sorting (SORT? FSORT?)
- Sound effects in data browsing (FSHOW?:6)
- Specifications (SPEC?)
- Spectral decomposition of a symmetric matrix (MATDEC?)
- Standard deviations (STAT? CORR? TAB? MTAB?)
- Star symbol plots (STARS?)
- STAT (operation) basic statistics
- Statistical methods (STATIS?)
- Statistical tests (TEST?)
- /STEPREG (sucro) stepwise linear regression analysis (by M.Karpoja)
- /SUCRO (keyboard sucro) helps in writing statements of the sucro language.
- /SUCROS (sucro) presents the sucros available as general tools.
- Sucros (SUCRO?)
- /SURVO-SETUP (sucro) for checking and changing of Survo system parameters
- /SURVO-START (sucro) for a menu-based start of the Survo system
- SYSTEM (command) for temporary changes of system parameters
- SURVO.APU file for system parameters (SYSTEM?)
- System parameters (SYSTEM?) in SURVO.APU file
- T -> TRIM
- *t* test (COMPARE?)
- /T2 (sucro) Hotelling's T^2 test for two samples
- TAB (key) moves the cursor to the next tab position.
- TAB operations (TAB?)
- Computing of multiway tables
 - Log-linear models (TABFIT?)
 - Management of multiway tables
- Tab positions (KEYS2?)
- TABFIT (operation) estimation of log-linear models
- TABLE (definition) labels a multiway table in the edit field.
- Tables in the edit field
 - Computing new columns (SET? COUNT? C+? VAR?)
 - Copying (COPY?)
 - Formatting (FORM?)
 - Interpolation (INTERP?)
 - Matrices (MAT?)
- Moves of rectangular blocks (BLOCK? MOVE?)
- Moving and deleting of columns (CHANGE? INSERT? DELETE?)
- Moving of rows (CHANGE?)
- Multiway tables (TAB?:3)
- Sorting (SORT?)
- Sums etc. of columns (C+?)
- Sums etc. of rows (L+?)
- Transposing (TRANSP?)
- Teaching programs (SUCRO?)
- Tests (TEST?)
- Text editing
- Automatic saving of the edit field (AUTOSAVE?)
- Box and line graphics -> /BOX

- Clearing the edit field (SCRATCH? INIT? CLEAR? DELETE? ERASE?)
- Code conversions (CONVERT?)
- Copying of lines (AF3? COPY?)
- Date -> /DATE
- Display attributes (F5?)
- Exchanging consecutive words -> /X
Finnish spelling checker (OIO?)
- Formatting (TRIM? FORM?)
- Hyphenating at the end of lines -> /H
- Loading data files (FLOAD?)
- Loading edit fields from files (LOAD? SHOW?)
- Loading information to the edit field (LOAD? LOADP? FLOAD? LOADM?)
- Loading text files (LOADP? SHOW?)
- Lowercase to uppercase -> /C
- Merging and splitting of lines (F6?)
- Moving of text (WORDS?)
- Moving rectangular blocks (BLOCK? MOVE? INSERT? CHANGE? UPDATE?)
- Printout on paper or to a file -> Printouts (PRINT?)
- Saving of the edit field (SAVE? SAVEP?)
- Saving text from the edit field (SAVE? SAVEP?)
- Searches (SEARCH? FIND? REPLACE?)
- Selecting the data disk/path (F4? DISK?)
- Setting a reference point (F7?)
- Shadow characters -> /S
- Size of the edit field (REDIM? INIT?)
- Trimming (TRIM?)
- Typing characters not available on the keyboard (CODE?)
- Uppercase to lowercase -> /L
- Word completing (PREFIX J)
- Text files (ASCII)
 - Copying text from the edit field to a text file (SAVEP?)
 - Moving data files to text files (FLOAD?)
 - Moving to data files (FSAVE?)
 - Printout (PRINT2?)
 - Reading (LOADP? SHOW?)
- TEXTS (specification) for additional texts in graphs
- TICK (specification) for tick lines on coordinate axes in graphs
- TICKLEN (specification) sets the length of TICK lines.
- TIME (command) gives the date and time.
 - Time control
 - Date and time (TIME1?)
 - Date in words -> /DATE
 - Wait in operation sequences (WAIT?)
 - Wait in graphics display -> WAIT (specification)
 - Alarm clock -> /ALARM
 - Time measures (CONV?:2:5)
 - Time series analysis (TIMESER?)
 - Time series plotting (SCAT?)
 - Time series transformations (SER?)
- TOUCH (key F3) toggles the touch mode for computing.
- Touch mode for computations (TOUCH? F3?)
- TRANSFORM (operation) for data transformations
Transformation analysis -> MATRUN
- TRANSP (command) transposes a table in the edit field.
- TREND (specification) for linear trends in scatter plots
- TRIM (command) adjusts the line length of texts in the edit field.
- TYPE (specification) defines the type of the graph.
 - TYPE=HBAR Stacked horizontal bar chart
 - TYPE=VBAR Stacked vertical bar chart

- TYPE=%HBAR Stacked horizontal bar chart in percentages
- TYPE=%VBAR Stacked vertical bar chart in percentages
- TYPE=MHBAR Multiple horizontal bar chart
- TYPE=MVBAR Multiple vertical bar chart
- TYPE=%MHBAR Multiple horizontal bar chart in percentages
- TYPE=%MVBAR Multiple vertical bar chart in percentages
- TYPE=%AHBAR As %HBAR but areas equal to absolute size
- TYPE=%AVBAR As %VBAR but areas equal to absolute size
- TYPE=PIE Multiple pie chart, size proportional to column sum
- TYPE=%PIE Multiple pie chart, total area constant
- TYPE=CONTOUR Contour plots of functions of 2 variables as a raster image
- TYPE=MATRIX (Data) matrix as a raster image
- TYPE=ANDREWS Andrews' function plots
- TYPE=FACES Chernoff's faces
- TYPE=DRAFTS Draftsman's display, matrix scatter plots
- TYPE=PROFILES Profile symbol plots
- TYPE=STARS Star symbol plots
- -> Correlation diagrams
- -> Curve plotting
- -> Histograms
- Types of graphs -> TYPE
- Typographic measures (MEASURES?)
- UPDATE (command) updates lines in an edit file.
- VALUES (specification) for values and percentages in bar/pie charts
- VAR (operation) transformation of variables
- Variables (statistical)
 - Linear combinations (LINCO?)
 - Scales of measurement (SCALES? FACTIV?)
 - Selection (SELECTV? MASK? VARS?)
 - Transformation (VAR? CLASSIFY? TRANSFORM? SER?)
 - Varimax rotation (ROTATE?)
- Volume measures (CONV?)
- WAIT (command) sets a wait in operation sequences.
- WAIT (specification) sets a wait in screen graphics.
- Weight measures (MEASURES?)
- Wilcoxon test (COMPARE?)
- WORDS (key alt-F2) for moving text in the edit field
- Work measures (MEASURES?)
- /X (key sucro) exchanges locations of two consecutive words.
- XCORR (operation) auto- and crosscorrelations
- XDIV (specification) defines the division of the picture width.
- XLABEL (specification) labels the X axis in graphs.
- XSCALE (specification) gives the scale for the X axis in graphs.
- YDIV (specification) defines the division of the picture height.
- YFILL (specification) draws horizontal fill lines in curve plotting.
- YLABEL (specification) labels the Y axis in graphs.
- YSCALE (specification) gives the scale for the Y axis in graphs.
- Zero-sum games (SIMPLEX?)

References

- Abraham, B. and Ledolter, J. (1983). *Statistical Methods for Forecasting*, Wiley.
- Adobe Systems Inc. (1985). *PostScript Language Reference Manual*, Addison-Wesley.
- Adobe Systems Inc. (1985). *PostScript Language Tutorial and Cookbook*, Addison-Wesley.
- Anderberg, M.R. (1973). *Cluster Analysis for Applications*, Academic Press.
- Andrews, D.F. (1972). Plots of high-dimensional data, *Biometrics* 28, 125-36.
- Belsley, D.A., Kuh, E. and Welch, R.E. (1980). *Regression Diagnostics*, Wiley.
- Butler, D.E. and Stokes, D. (1974). *Political Change in Britain*, (2nd Edition), Academic Press.
- Chambers, J.M. (1977). *Computational Methods for Data Analysis*, Wiley.
- Chernoff, H. (1973). Using faces to represent points in k-dimensional space graphically, *JASA* 68, 361-368.
- Conover, J. (1971). *Practical Nonparametric Statistics*, 2nd Edition, Wiley.
- Einslein, Ralston and Wilf, editors (1977). *Statistical Methods for Digital computers*, Wiley.
- Harman, H. (1967). *Modern Factor Analysis*, 2nd Edition, University of Chicago Press.
- Helmholz, H. (1877). *On the Sensations of Tone*, (English edition by A.J.Ellis 1885), Reprinted by Dover 1954.
- Korhonen, M. (1989). *Analysis of Variance and covariance*, SURVO 84C Contributions 4, Dept. of Statistics, University of Helsinki.
- Korhonen, M. and Sadeniemi, L. (1992). *Additional Survo operations I*, SURVO 84C Contributions 5, Dept. of Statistics, University of Helsinki.
- Korhonen, P. (1979). *A stepwise procedure for multivariate clustering*, Research Reports No.7, Computing Centre, University of Helsinki.
- Mustonen, O. (1983). *Quatrilles and 6 Bagatelles for solo violin*, Fazer, Helsinki.
- Mustonen, S. (1980). *SURVO 76 Editor, a new tool for interactive statistical computing, text and data management*, Research Report No.19, Dept. of Statistics, University of Helsinki.
- Mustonen, S. (1982). *Statistical computing based on text editing*, Proceedings in Computational Statistics, 353-358, Physica-Verlag, Wien.
- Mustonen, S. (1987). *SURVO 84C User's Guide*, Dept. of Statistics, University of Helsinki.
- Mustonen, S. (1988). *PostScript Printing in SURVO 84C*, SURVO 84C Contributions 1, Dept. of Statistics, University of Helsinki.

- Mustonen, S. (1988). *Sucros in SURVO 84C*, SURVO 84C Contributions 2, Dept. of Statistics, University of Helsinki.
- Mustonen, S. (1989). *Programming SURVO 84 in C*, SURVO 84C Contributions 3, Dept. of Statistics, University of Helsinki.
- McCullagh, P. and Nelder, J.A. (1983). *Generalized Linear Models*, Chapman and Hall.
- Microsoft Corporation (1987). *Microsoft C Run-time Library Reference*.
- Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T. (1987). *Numerical Recipes*, Cambridge, USA.
- Walsh, G.R. (1975). *Methods of Optimization*, Wiley.
- Wells, D. (1987). *The Penguin Dictionary of Curious and Interesting Numbers*, Penguin.
- Wilkinson, J.H, and Reinsch, C. (1971). *Handbook for Automatic Computation, Vol.II, Linear Algebra*, Springer.