CONF-780879

PROCEEDINGS OF THE 6th ANNUAL SPEAKEASY CONFERENCE

PICK-CONGRESS HOTEL
CHICAGO, ILLINOIS
AUGUST 17-18, 1978

Sponsored by
ARGONNE NATIONAL LABORATORY

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

The facilities of Argonne National Laboratory are owned by the United States Government. Under the terms of a contract (W-31-109-Eng-38) between the U. S. Department of Energy, Argonne Universities Association and The University of Chicago, the University employs the staff and operates the Laboratory in accordance with policies and programs formulated, approved and reviewed by the Association.

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately-owned rights. Mention of commercial products, their manufacturers, or their suppliers in this publication does not imply or connote approval or disapproval of the product by Argonne National Laboratory or the U. S. Department of Energy.

CONF - 78079

Distribution Category
Mathematics and Computers
(UC-32)

PROCEEDINGS OF THE 6th ANNUAL SPEAKEASY CONFERENCE

HELD AT THE PICK-CONGRESS HOTEL

CHICAGO, ILLINOIS

AUGUST 17-18, 1978

Sponsored by

ARGONNE NATIONAL LABORATORY

NOTICE

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the United States Department of Energy, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness or usefulness of any information, apparatus, product or process disclosed, or represents that its use would not infringe privately owned rights.

6TH ANNUAL
SPEAKEASY - 1978
ATTENDEE ADDRESS LIST

Anderson, Janet B.
Argonne National Laboratory

Augustine, Charles A.
Carnegie-Mellon University
5111 Science Hall
Pittsburgh, PA 15213

Bailey, Mike
Computer Aided Design and Graphics Lab
School of Mechanical Engineering
Purdue University
W. Lafayette, IN 47907

Baker, James F.
Joseph Schlitz Brewing Company
235 W. Galena
Millaukee, Wisconsin 53212

Ballengee, David W.
General Public Utilities
Madison Avenue
Morristown, New Jersey 07960

Bavinger, Bill A.
Rice University
P. O. Box 1892
Houston, Texas 77001

Bengtson, Susan E.
Clemson University
Clemson, SC 29631

Bharucha-Reid, Albert T.
Wayne State University
Detroit, MI 48202

Bitterli, Charles V.
Applied Physics Lab
Johns Hopkins University
Johns Hopkins Rd
Laurel, Md. 20810

Bosse, James K.
Sikorsky Aircraft
N. Main Street
Stratford, Conn. 06602

Burner, Kenneth
Carnegie-Mellow University
5103 Science Hall
Pittsburgh, PA 15213

Burris, Anthony E.
Foreign Demand And Competion Div.
Economic Research Service
U.S. Department of Agriculture
500 12th Street, S.W.
Washington, D.C. 20250

Capra, Flavio
Banca Commerciale Italiana
Ufficio Studi
Via Borgonuovo 2
Milano, Italy 20100

Child, Margaret E.
University of California San Francisc
Rm U-76,UCSF
San Francisco, CA 94114

Cohen, Stan
Speakeasy Computing Corporation
222 West Adams Street
Chicago, Illinois 60606

Coldrick, David B.
National Research Council
Computation Centre
Bldg. M-60
Montreal Rd
Ottawa, Ontario K1A OR6

Conley, Roy D.
Federal Reserve Bank of San Francisco
P. O. Box 7702
San Francisco, CA 94120

Crema, Alice M.
Signetics
811 E. Arques Bin 3
Sunnyvale, CA 94086

Diament, Alex
Federal Reserve Bank of Philadelphia
100 N. Sixth Street
Philadelphia, PA 19105

Furusawa, T.
Fujitsu Limited
1015 Kamikodanaka
Nakahara-ku
Kawasaki, Japan

Goldhammer, Donald H
University of Chicago
Computation Center
5737 S. University
Chicago, Illinois 60637

Grace, Thom
Argonne National Laboratory

Hall, Janet M.
Clemson University
Computer Center
Clemson, SC 29631

Harris, James S.
Exxon Corporation
P. O. Box 153
Florham Park, NJ 07932

Haycock, Andrew B.
National Research Council
Bldg. M-60
Montreal Rd
Ottawa, Ontario K1A OR6

Herman, Ronald I.
Federal Reserve
Data Processing
20th & Constitution Avenue
Washington, D.C. 20551

Hopper, Marcy
Speakeasy Computing Corporation
222 West Adams Street
Chicago, Illinois 60606

Hull, Harry M.
Computing Center
University of Wisconsin
1210 W. Dayton St.
Madison, Wisconsin 53706

Jones, Alan J.
Unilever Computer Services
Box 110
Watford, Herts
England WDI ISA

Kaplan, Randy
Federal Reserve Bank of Phila.
100 N. Sixth Street
Philadelphia, PA 19105

Kau, Jackie Y.
Federal Reserve Bank of S.F.
400 Sansome St.
San Francisco, CA 94120

Kilgore, James A.
Department of Army
USAMSSA(CSAM-DPD-D)
Pentagon
Washington, D.C. 20310

Lachmuth, Lyle T.
Imperial Oil Ltd
Systems & Computer Services
500 6 Avenue SW
Calgary Alberta
Canada T2P 2S1

Laplanche, Gilbert J.
University of Liege
C.E.C.T.I.
15 Avenue des Tilleuls
Liege Belgium B-4000

Leibson, Norman H.
Security Pacific Bank
611 North Brand
Glendale, CA 91203

Maccabee, Halaine
Northeastern Illinois Planning
Commission
400 West Madison
Chicago, Illinois 60606

Machusick, Richard A.
GPU Service Corporation
Box 1018
Reading, PA 19603

Meyer, Robert
Gulf University Research
Consortium Suite 600
5909 W. Loop South
Bel Air, Texas 77401

Massagli, Robert A.
Federal Reserve Bank of S.F.
400 Sansome St.
San Francisco, CA 94120

Massaquoi, Steve
Argonne National Laboratory

Meacham, Robert C.
Union Carbide Corporation
Nuclear Division
M.S.259,4500N
P. O. Box X
Oak Ridge, TN 37830

O'Donnell, Patrick T.
Pennzoil Company
700 Milam
Houston, Texas 77001

O'Reilly, Daniel F.X.
Argonne National Laboratory

Overton, Charles E.
U. S. Dept. of Agriculture
500 12th S.W.
Washington, D. C. 20250

Paloranta, Duane L.
Control Data Corporation
8100-34th Avenue So.
Minneapolis, MN. 55440

Peterson, Barbara C.
Florida Power Corporation
3201 - 34th Street South
St. Petersburg, Florida 33711

Pieper, Steven
Argonne National Laboratory

Reynolds, John H.
COMSAT
P. O. 115
Clarksburg, Maryland 20734

Richau, Clem
A. S. Hansen, Inc.
EDP
1080 Green Bay Rd.
Lake Bluff, Illinois 60044

Rower, Jack
Economic Research Service
U. S. Department of Agriculture
500 12th Street, S.W.
Washington, D.C. 20250

Rutiezer, Howard
A.S. Hansen, Inc.
EDP
1080 Green Bay Rd.
Lake Bluff, Illinois 60044

St. George, George
Department of Agriculture
Economic Research Service
500 12th SW Room 272
Washington, D. C. 20023

Saxe, David
Educational Testing Service
Rosedale Road
Princeton, New Jersey 08540

Schlichting, Richard
Argonne National Laboratory

Schwartz, Martin W.
Economic Research Service
U. S. Department of Agriculture
500 12th Street, S.W.
Washington, D. C. 20250

Sharp, Geoffrey H.
Lockheed
3251 Hanover
Palo Alto, CA 94304

Skrydlak, James S.
Amdahl Corporation
1250 East Arques Avenue
Sunnyvale, CA 94086

Stack, Dick
Thomas L. Jacobs & Associates
53 W. Jackson
Suite 1339
Chicago, Illinois 60604

Stob, Mike
Speakeasy Computing Corporation
222 West Adams Street
Chicago, Illinois 60606

Stokes, Houston H.
University of Illinois
Box 4348
Chicago, Illinois 60637

Van Hassel, Bill
Educational Testing Service
Rosedale Road
Princeton, New Jersey 08540

Wehrenberg, Monika
Argonne National Laboratory

Weiss, Michael
U.S. Department of Agriculture
500 12th St S.W., Room 202
Washington, D. C. 20250

6th ANNUAL SPEAKEASY CONFERENCE

August 17-18, 1978
Pick-Congress Hotel
Chicago, Illinois 60605

PROGRAM

THURSDAY, AUGUST 17

8:30 a.m.	Registration	
9:30 a.m.	Welcome Overview of Conference General Discussion	*S. Cohen*
10:30 a.m.	*COFFEE*	

GRAPHICS

11:00 a.m.	Graphics under Speakeasy	*J. H. Reynolds*
11:30 a.m.	Speakeasy on a Mini	*M. Bailey*
11:50 a.m.	Colour Graphics	*A. Jones*
12:15 p.m.	*LUNCH*	

TIME SERIES

1:45 p.m.	TDAM/OASIS A User Oriented System at U.S. Department of Agriculture	*M. Schwartz* *G. St. George*
2:30 p.m.	Implementation Difficulties with Box Jenkins Analysis	*R. Conley*
2:50 p.m.	General Discussion about Time Series Analysis	
3:15 p.m.	*COFFEE*	

APPLICATIONS

3:45 p.m.	Weather and Crop Yield Analysis System	*M. Weiss*
4:00 p.m.	Property Investment Analysis System	*D. Stack*
4:15 p.m.	General Discussion about Contributions	

4:30 p.m.	Art and the Computer	*M. Wehrenberg*
6:00 p.m.	BANQUET--Family House Restaurant	
10:30 p.m.	Bus Leaves Family House for Pick-Congress	

FRIDAY, AUGUST 18

DATA BASES UNDER SPEAKEASY

9:00 a.m.	Relational Data Base	*R. Schlichting*
9:15 a.m.	Applications of Relational Data Bases	*B. Bavinger*
9:45 a.m.	TOTAL Interface, Implementation and Applications	*A. Diament*
10:15 a.m.	*COFFEE*	

SURVEY ANALYSIS

11:00 a.m.	Survey Analysis Package from Liege	*G. Laplanche*
11:30 a.m.	F4STAT and Its Future under Speakeasy	*D. Saxe*
11:45 a.m.	General Discussion about Statistics and Speakeasy	
12:15 p.m.	*LUNCH*	

NEW FEATURES IN SPEAKEASY

1:45 p.m.	Partial Differential Equations Compiler/Optimization	*D. O'Reilly* *T. Grace*
3:00 p.m.	*COFFEE*	
3:45 p.m.	Open	
5:00 p.m.	End of Conference	

The conference schedule has been kept relatively open so that new topics can be added. The last session has been left open so that the audience can interact as a whole.

COMSAT Speakeasy Graphics

A Status Report

John H. Reynolds*

I. INTRODUCTION

This note will describe the status of the Speakeasy graphics system developed at COMSAT Laboratories. The objective of this system is to provide an uncomplicated graphics facility. Like many other Speakeasy linkules, graphics linkules can be thought of as generalized operators which map data onto a graphic display in a readily understandable form. The internal workings of this process while quite complicated are and should be invisible to the user. The inherent problem is that a format which is "readily understandable" for one user or one application may be incomplete or inappropriate for another user or application. The solution is to provide a stable, modular means of modifying the form of the graphic output. Stability here means that various sets of "bells and whistles" do not interfere with one another. It requires adherence to a central guiding philosophy and standards. Modularity means that the user can select only those modifications of interest to him. He should in fact be able to proceed without knowing the full capabilities of the system.

The ability to modify the graph has been steadily strengthened. The original version had scaling and labeling options and the ability to change the location and size of the axes. Later versions provided alternate axis formats such as probability scales and a USERAXIS which allows the user to specify the axis axis format in some detail. The flexibility of text output has been steadily improved and now exploits the capabilities of newer graphic devices. Special capabilities of supported devices, such as graphic input, have also been utilized.

This work is by no means complete. Because its development has been driven primarily by the requirements of COMSAT, the present system was designed to plot scientific and engineering data: line graphs on orthogonal axes. This format is of course also appropriate for many other applications but other formats such as histograms are also needed. Additional axis formats such as calendar scales are required for some applications. Compatible three-dimensional graphics should be supported. If the

*Author's address:
Device Physics Department
COMSAT Laboratories
Clarksburg, MD 20734

system is to continue to grow, these and other capabilities should be added.

A basic design principle is that present and future capabilities be compatible with all supported devices. When a requested function, such as graphic input, can not be performed by the device, a reasonable action must be taken. An example of this is shown later when the new character rotation facility is illustrated. A corollary is that a graph specification (and graphics programs) should be portable; that is, once a graph has been generated on one device a "similar" graph should be producible on any other supported device.

The next section contains an overview of the graphics system design. The means by which the system meets the objectives of stability, modularity, and portability are briefly described. Areas requiring improvement or development are pointed out. The last section illustrates some recent enhancements. Many of them were added to exploit the capabilities of the Tektronix 4662 pen plotter but can be adapted to similar new graphics devices. The result of recent work has been the ability to generate graphs of increasingly high quality on a greater variety of devices.

II. SYSTEM OVERVIEW

A. Operational Components

The operational part of the system consists of:

1. graphics linkules which are compatible with the standard Speakeasy processor;

2. sets of device dependent parameters used to initialize the system, control the hardware interface software, and provide appropriate default values for the graph specifications;

3. Help Documents and other user-oriented documentation.

Graphics linkules are classified as device independent or device dependent. The former are used to set up graph specifications and to interact with the Speakeasy processor. The latter interact with a graphic device or return results which depend on the characteristics of the device. Each device dependent linkule is self contained incorporating all necessary device driver software. Since the driver software is unique to a specific device or device family, multiple linkule libraries are required to support diverse graphic devices. As described below, the appropriate library is automatically attached to the processor and used by the graphics system. At present, support is available for Tektronix CRT terminals, the Tektronix 4662 pen plotter, and Anderson-Jacobson terminals.

Figure 1 illustrates the overall flow of a graphics session. The GRAPHICS linkule initializes the system. The initialization process, which is discussed in more detail below, creates communication areas (TERMPARM and PLOTPARM) which remain in Named Storage for the duration of the graphics session. Graph specifications stored in PLOTPARM are modified by device independent linkules such as SETXSCALE. Device dependent linkules, such as GRAPH, ANNOTATE, and CURSOR, interact with the graphic device. For example, the GRAPH linkule which is described in more detail below, uses the graph specifications in PLOTPARM and the device dependent parameters in TERMPARM to generate a graph.

Most graphics linkules are straightforward but two key linkules, GRAPHICS and GRAPH, require a more detailed description. Figure 2 illustrates the initialization procedure performed by the GRAPHICS linkule. One of the parameters passed to GRAPHICS is a word (such as TEK4014) which identifies the graphic device. This must be the name (or alias name) of a member of an attached library. The member contains the appropriate device dependent parameter set. (In the present implementation, the parameters are stored in members of 'SPEAK.PROCLIB'). The parameters are used to initialize TERMPARM. One of them is the file name of the supporting device dependent linkule library. This file is dynamically attached to the processor using the Speakeasy Data Management Facility.

If PLOTPARM already exists in Named Storage, it is retained; otherwise it is allocated and initialized. GRAPHICS itself does not initialize the graphic device; instead it schedules the device initialization linkule, STARTGRA, and returns control to the processor. The processor then calls STARTGRA from the newly attached device dependent linkule library.

Figure 3 shows the processing done by the GRAPH linkule. After verifying the argument list, GRAPH plots the data according to the specifications in PLOTPARM using the device dependent parameters in TERMPARM. GRAPH does not generate axes or labels. Instead it schedules the axis generation linkules and a label generation linkule and returns to the processor. The processor calls the scheduled linkules. An axis can be generated either by special axis linkules or by default linkules. Special axis linkules which create special formats such as a probability scale are specified by keywords stored in PLOTPARM by the SETXAXIS and SETYAXIS linkules. By convention, the corresponding linkule names are @XKEYWOR (for an X axis) and @KEYWOR (for a Y axis). KEYWOR is the axis keyword shortened if necessary to six characters. A keyword is validated simply by determining that the corresponding linkule is in an attached library. This mechanism makes it quite simple to add special axes. The default situation is signaled by a blank keyword parameter. The default axis linkule name is determined by the data scaling mode (linear and logarithmic scaling are currently supported). Only a default labeling linkule is presently available, but alternate linkules could be specified via a similar mechanism.

A logical improvement is to move the curve generation code into another linkule. Then the GRAPH linkule would become device independent, serving primarily as a checker and scheduler. Additional data formats, such as histograms, could then be provided by adding alternate data plotting linkules.

The other graphics linkules are more straightforward and will not be discussed in this overview. They perform elementary functions such as modifying specifications in PLOTPARM or generating text. The graphics Help Documents provide a functional description.

B. Support and Development Components

The graphic software is organized into three levels. The highest is the graphic linkule code. The second is the graphic subroutines called by the graphic linkules. There are two classes: 1) utility subroutines which perform internal functions such as scaling or extracting data from the communication areas; 2) device interface subroutines which interface between the linkules and the device driver software. Device interface subroutines perform elementary functions such as drawing a straight line or generating a line of text. All communication with graphic devices must occur through interface subroutines. (An exception is the startup linkule which may make direct calls to the driver software.) The lowest level is the device driver software supplied by the device manufacturer. Only elementary driver routines are required.

This organization allows expansion in two directions: 1) new or improved linkules can be written; 2) new device interface libraries can be added to support other graphic devices. The system has the potential to grow geometrically as new capabilities are added and new devices are supported.

System documentation and source text for the graphic help documents is in Waterloo Script readable form. Procedures have been developed which generate a formatted Help Document listing and which build a standard Speakeasy Help Document library directly from the source text. Other support facilities include a linkule to create or modify the device parameter sets.

III. RECENT WORK

Recent work has concentrated on providing support for the Tektronix 4662 pen plotter. The plotter has graphic input capabilities. A new version of the CURSOR linkule uses this capability and is useful for digitizing existing graphs or drawings. The plotter also has improved character output, including variable character size and character rotation. This capability is used by both the GRAPH and ANNOTATE linkules. Character rotation is illustrated in Figures 4 through 6. Figure 4 contains a Speakeasy program which calls the new ANNOTATE. The resulting output on a pen plotter is shown in Figure 5. The same program using a Tektronix 4014 CRT terminal which does not support character rotation is shown in Figure 6. It is seen that the text arrangement has been altered so that it remains readable in all orientations. This is an example of portability -- Figure 6 is "similar" to Figure 5.

The plotter also supports variable character width and and height. This feature is exploited by a variable pitch option which generates a version of Engineering Gothic script: the character width and spacing is variable; some character pairs such as "LT" are overlapped. The result is comparable to hand drawn lettering. Superscript and subscript notation is also supported. Superscript characters which are part of the "TN train" character set can be generated. They are signaled by escape characters in the text string; for example, the expression "X to the minus one" can be designated by the Speakeasy statement: ANNOTATE(1, 1,"X%-%1"). The percent signs are escape characters which indicate that the following character is to be translated to a special character. General subscript and superscript notation is provided by the escape sequences: "%>" and "%<". The former (read as half space up) is used to prefix a string of superscripts; the later (read as half space down) prefixes a string of subscripts. Half space down is also used to return from the superscript mode to the normal text mode; similarly, half space up terminates the subscript mode. Mixed subscripts and superscripts are allowed. An example of the new text output on a pen plotter is shown in Figure 7. The program is shown in Figure 8. Subscripts and superscripts can be generated on all supported devices.

Another new feature is the ORIENTGRAPH command. This command allows the user to effectively override the graph location and length specifications; the modifications remain in effect for the duration of the session. ORIENTGRAPH uses the graphic input facility of the graphics device (the crosshairs on Tektronix CRT terminals and the pen position on the Tektronix pen plotter) to indicate the location of the graph boundary. The axis lengths are adjusted and the display coordinates are translated and (if necessary) rotated to align the graph with the indicated points. The coordinate transformation is applied to all subsequent graphic output including text generated by the ANNOTATE command. It is also applied by the CURSOR linkule which returns values relative to the adjusted graph location. This facility is useful when digitizing. The ORIENTGRAPH Help Document contains more detailed information. One of the available options can be used to align the graph with respect to a calibration point (or benchmark) located at an arbitrary position. A sample program illustrating ORIENTGRAPH is shown in Figure 9; sample output is shown in Figures 10 and 11. The top graph was generated according to the specifications while the bottom graph was generated after the specifications were overridden by ORIENTGRAPH. Figure 11 shows that the graph can be rotated to correct for misalignment of the paper.

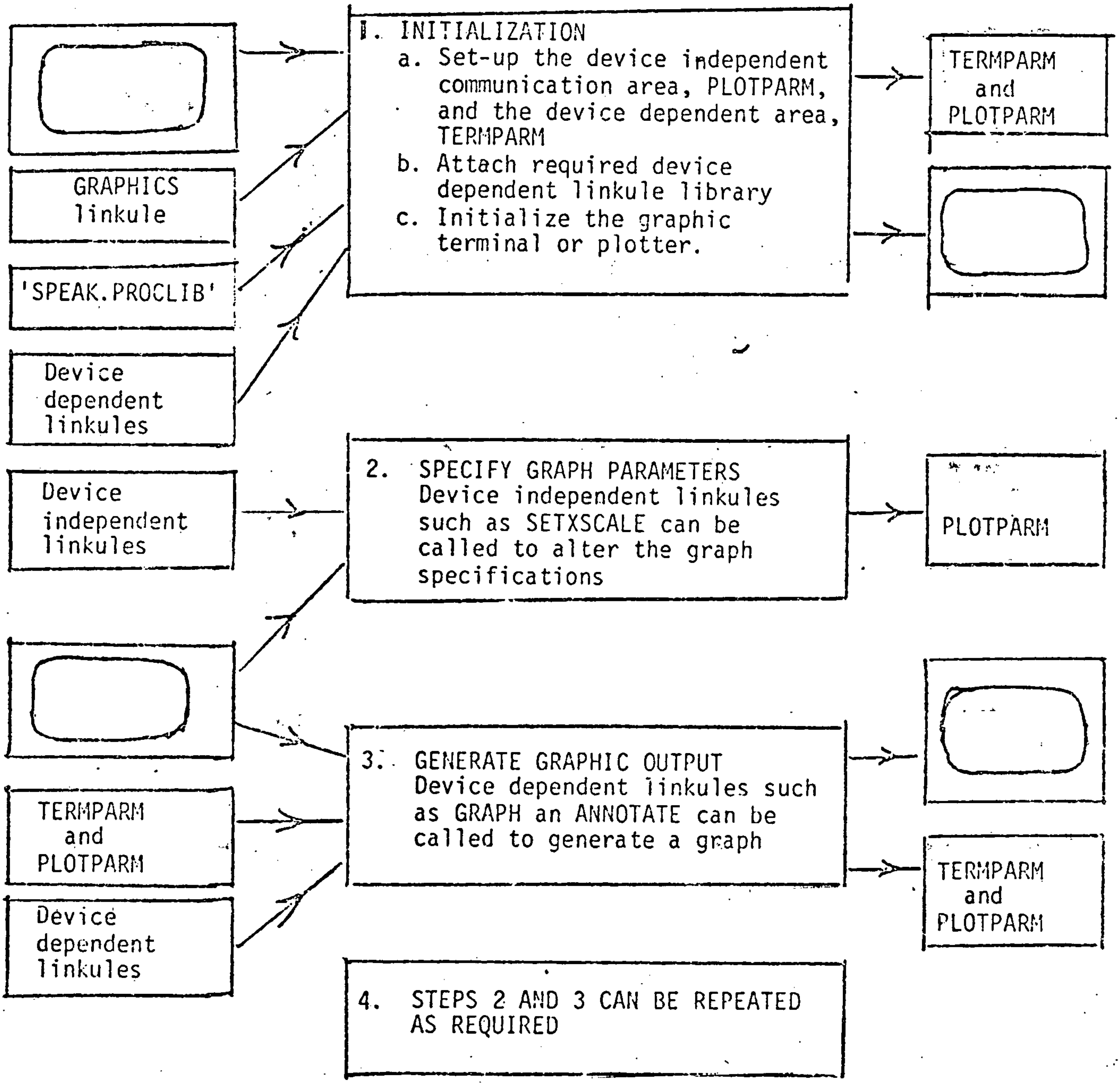
The last example illustrates a new data smoothing linkule as well as a Speakeasy program used to plot smoothed data with gaps in the neighborhood of the original data points. The linkule, SMOOTHCURVE, is an interface to IMSL data smoothing subroutines. It is described in detail in the SMOOTHCURVE Help Document. A notable option is DEGLITCH which invokes an empirical procedure to eliminate NON-random errors in the data. It adjusts data values which differ significantly from neighboring points. GAPCURVE shown in Figure 12 is the Speakeasy program. It uses SMOOTHCURVE and a number of other graphics linkules. Sample output from GAPCURVE is plotted in Figure 13 which also shows the original data points.

Two variations of SMOOTHCURVE, SMOOTH1D and SMOOTH2D, return respectively the first and second derivatives of the smoothed curve.

ACKNOWLEDGMENT

I would like to thank my colleagues at COMSAT Laboratories, particularly Gary Gordon, whose comments and suggestions contributed greatly to the development of the system. I also greatly appreciate Gary's invaluable assistance in the preparation of the Help Documents. Jerry Shifrin installed the COMSAT graphics driver packages and made a lot of balky software perform as specified. Al Berger made improvements to the processor which made many recent developments possible. Steve Pieper made many helpful suggestions and improvements. Finally, I would like to add special thanks to Stan Cohen for his continuing advice, encouragement and support.

GRAPHICS SESSION FLOW



1. INITIALIZATION
   a. Set-up the device independent communication area, PLOTPARM, and the device dependent area, TERMPARM
   b. Attach required device dependent linkule library
   c. Initialize the graphic terminal or plotter.

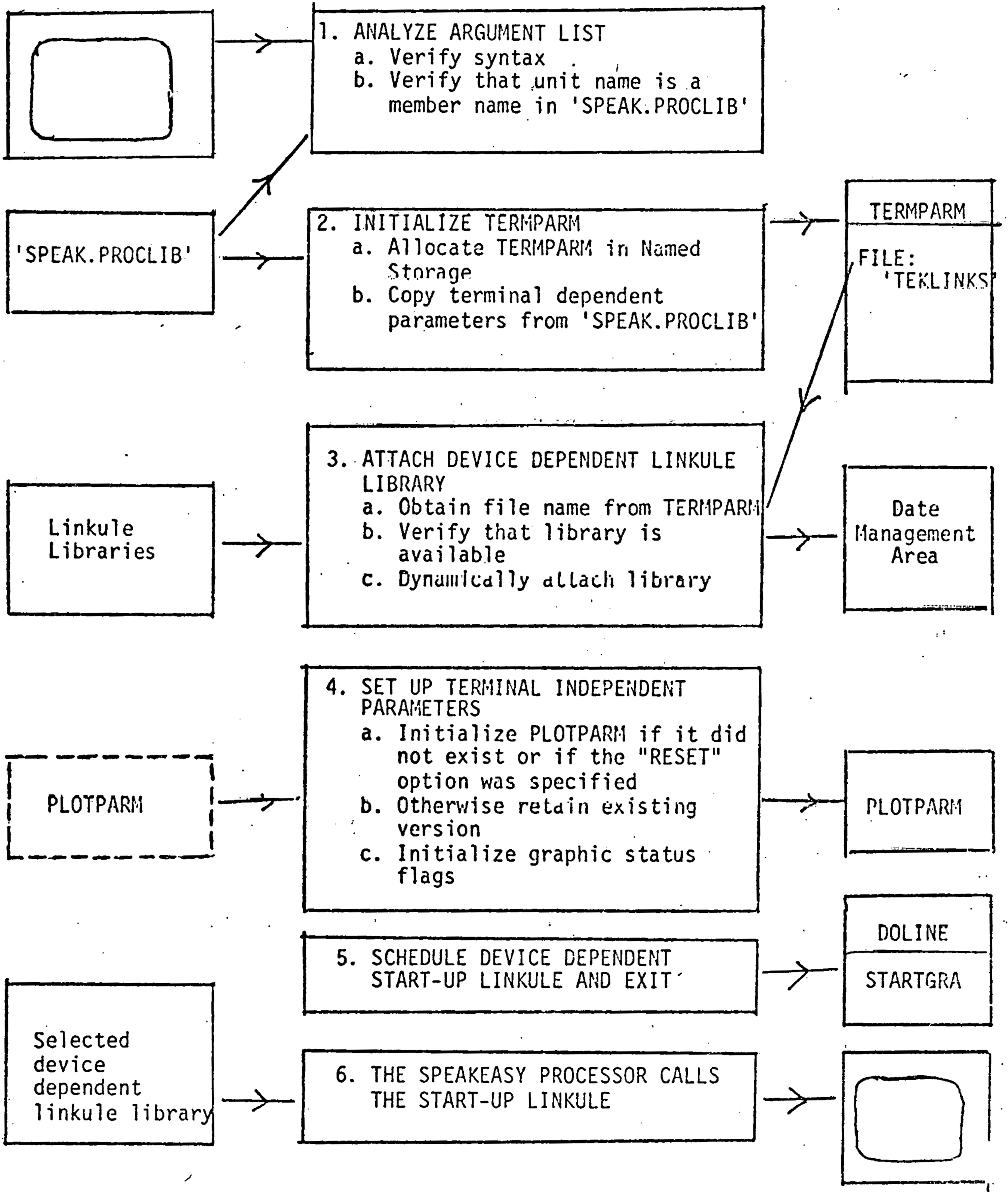
2. SPECIFY GRAPH PARAMETERS
Device independent linkules such as SETXSCALE can be called to alter the graph specifications

3. GENERATE GRAPHIC OUTPUT
Device dependent linkules such as GRAPH an ANNOTATE can be called to generate a graph

4. STEPS 2 AND 3 CAN BE REPEATED AS REQUIRED

Figure 1

THE GRAPHICS LINKULE



1. ANALYZE ARGUMENT LIST
   a. Verify syntax
   b. Verify that unit name is a member name in 'SPEAK.PROCLIB'

2. INITIALIZE TERMPARM
   a. Allocate TERMPARM in Named Storage
   b. Copy terminal dependent parameters from 'SPEAK.PROCLIB'

3. ATTACH DEVICE DEPENDENT LINKULE LIBRARY
   a. Obtain file name from TERMPARM
   b. Verify that library is available
   c. Dynamically attach library

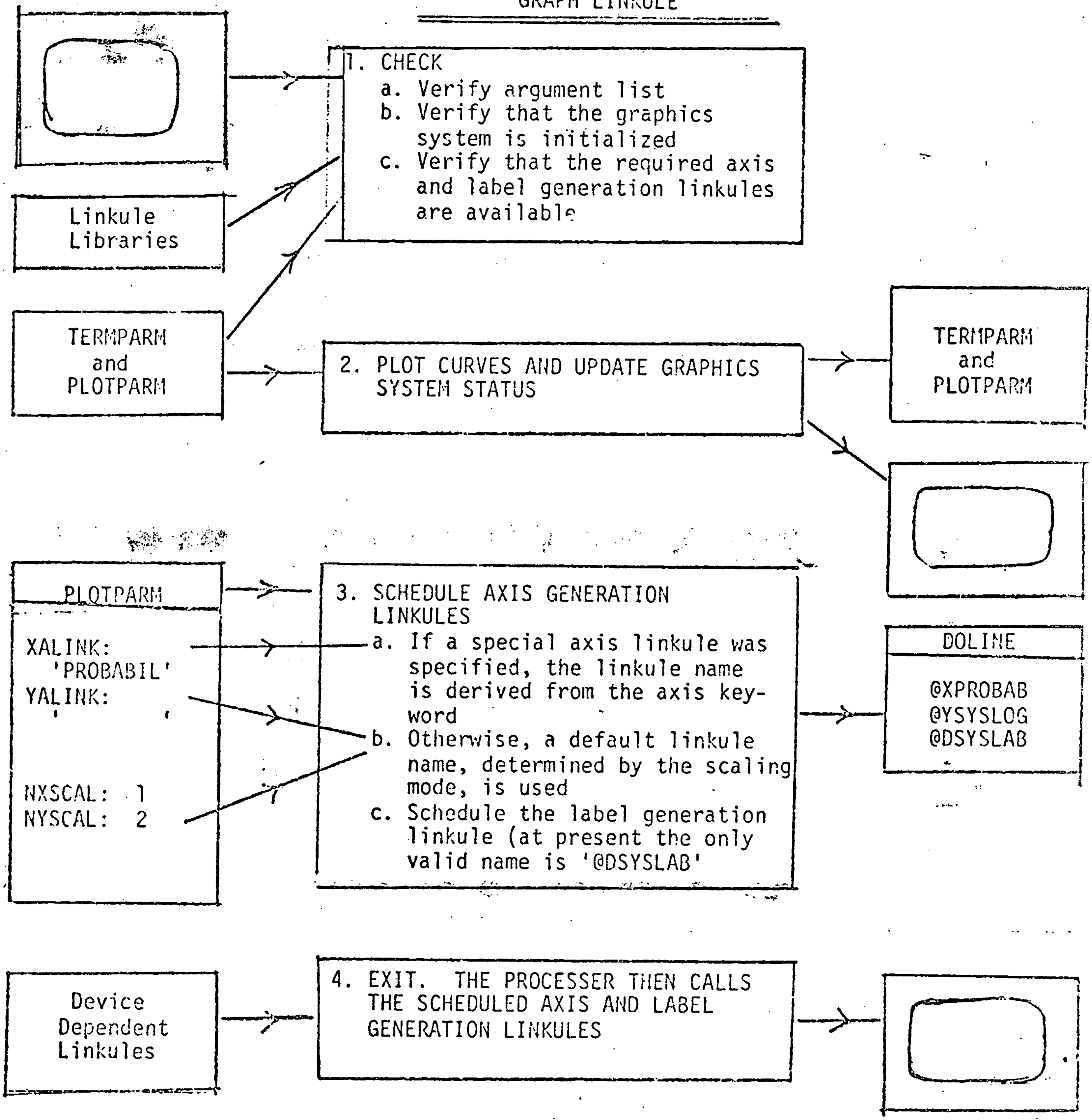
4. SET UP TERMINAL INDEPENDENT PARAMETERS
   a. Initialize PLOTPARM if it did not exist or if the "RESET" option was specified
   b. Otherwise retain existing version
   c. Initialize graphic status flags

5. SCHEDULE DEVICE DEPENDENT START-UP LINKULE AND EXIT

6. THE SPEAKEASY PROCESSOR CALLS THE START-UP LINKULE

Figure 2

GRAPH LINKULE



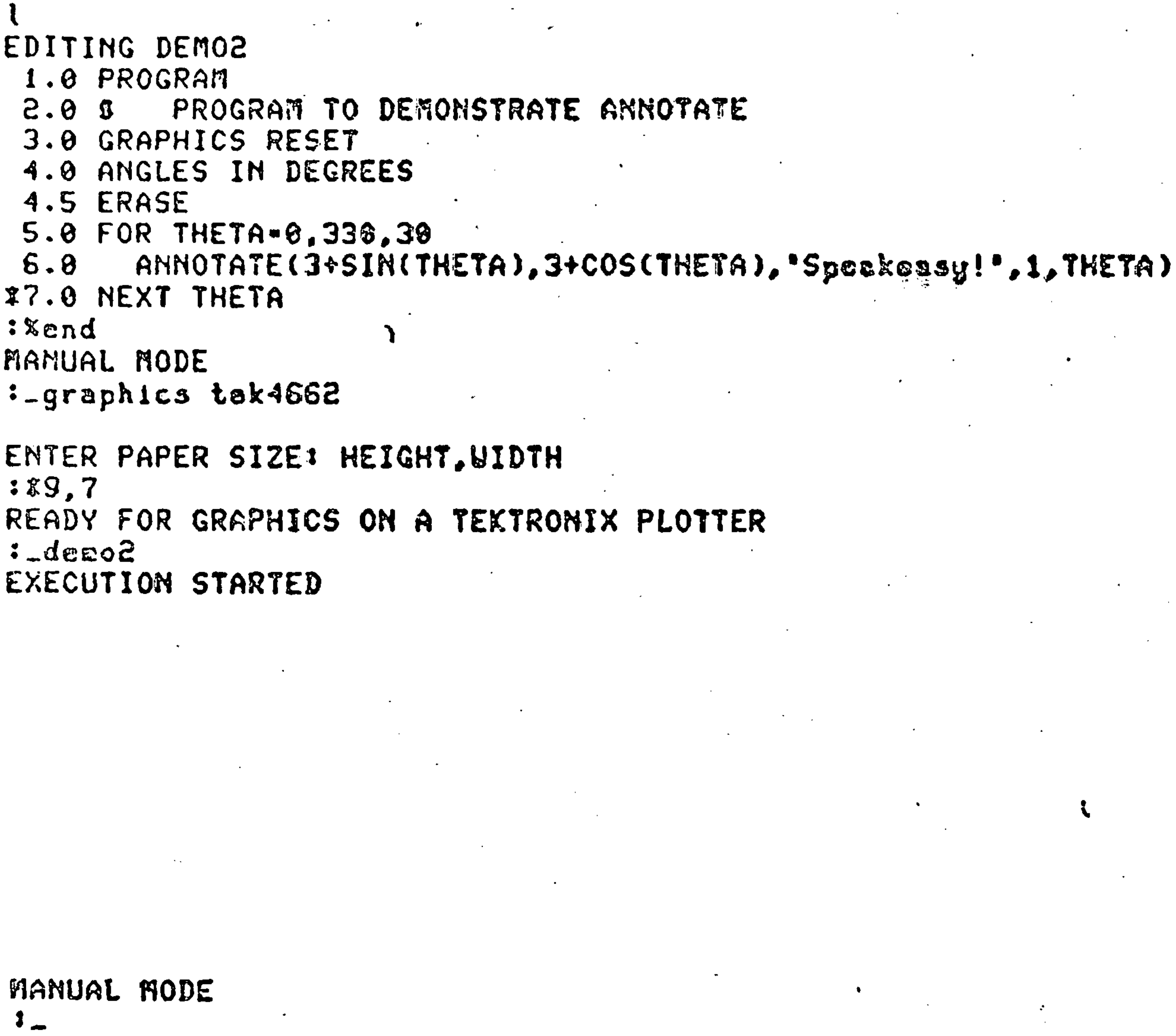
1. CHECK
   a. Verify argument list
   b. Verify that the graphics system is initialized
   c. Verify that the required axis and label generation linkules are available

2. PLOT CURVES AND UPDATE GRAPHICS SYSTEM STATUS

3. SCHEDULE AXIS GENERATION LINKULES
   a. If a special axis linkule was specified, the linkule name is derived from the axis keyword
   b. Otherwise, a default linkule name, determined by the scaling mode, is used
   c. Schedule the label generation linkule (at present the only valid name is '@DSYSLAB'

4. EXIT. THE PROCESSER THEN CALLS THE SCHEDULED AXIS AND LABEL GENERATION LINKULES

Figure 3



```
EDITING DEMO2
 1.0 PROGRAM
 2.0 $   PROGRAM TO DEMONSTRATE ANNOTATE
 3.0 GRAPHICS RESET
 4.0 ANGLES IN DEGREES
 4.5 ERASE
 5.0 FOR THETA=0,330,30
 6.0   ANNOTATE(3+SIN(THETA),3+COS(THETA),'Speakeasy!',1,THETA)
*7.0 NEXT THETA
:%end
MANUAL MODE
:_graphics tek4662

ENTER PAPER SIZE: HEIGHT,WIDTH
:*9,7
READY FOR GRAPHICS ON A TEKTRONIX PLOTTER
:_demo2
EXECUTION STARTED




MANUAL MODE
:_
```

Figure 4

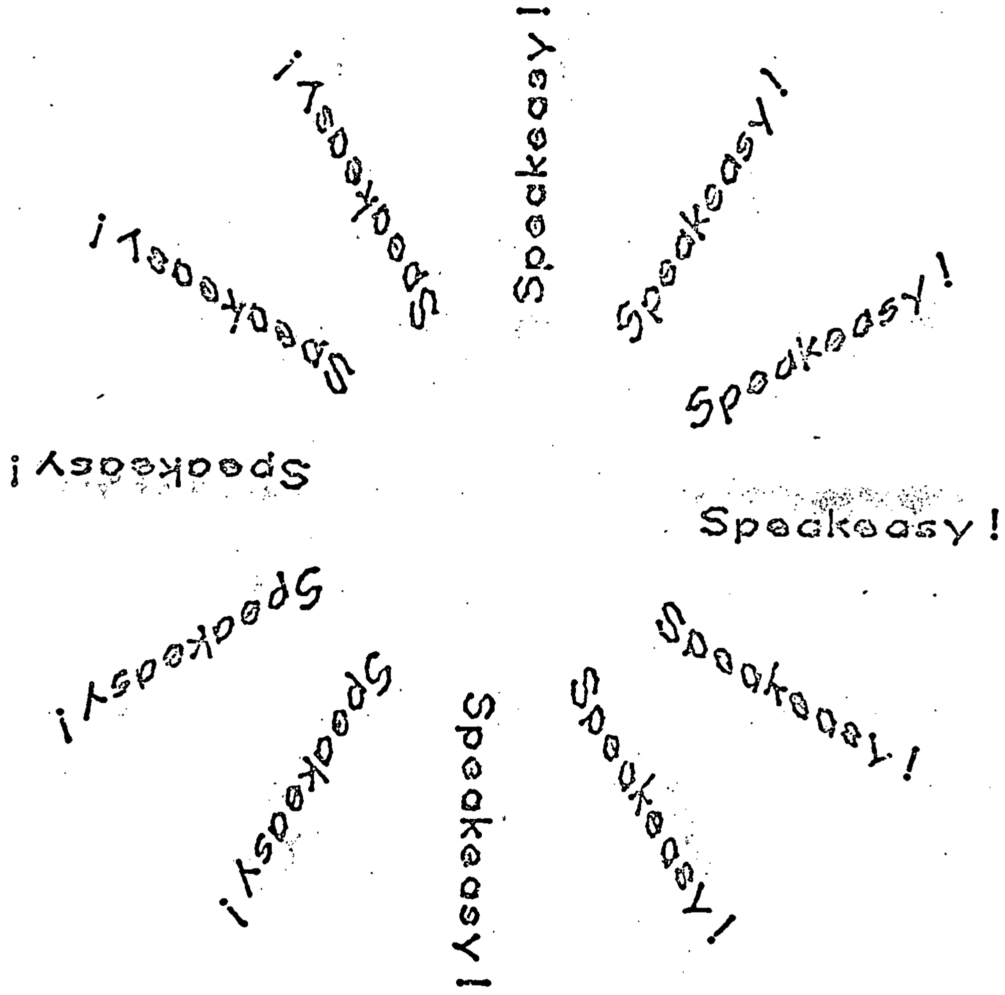

Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!
Speakeasy!

Figure 5

MANUAL MODE

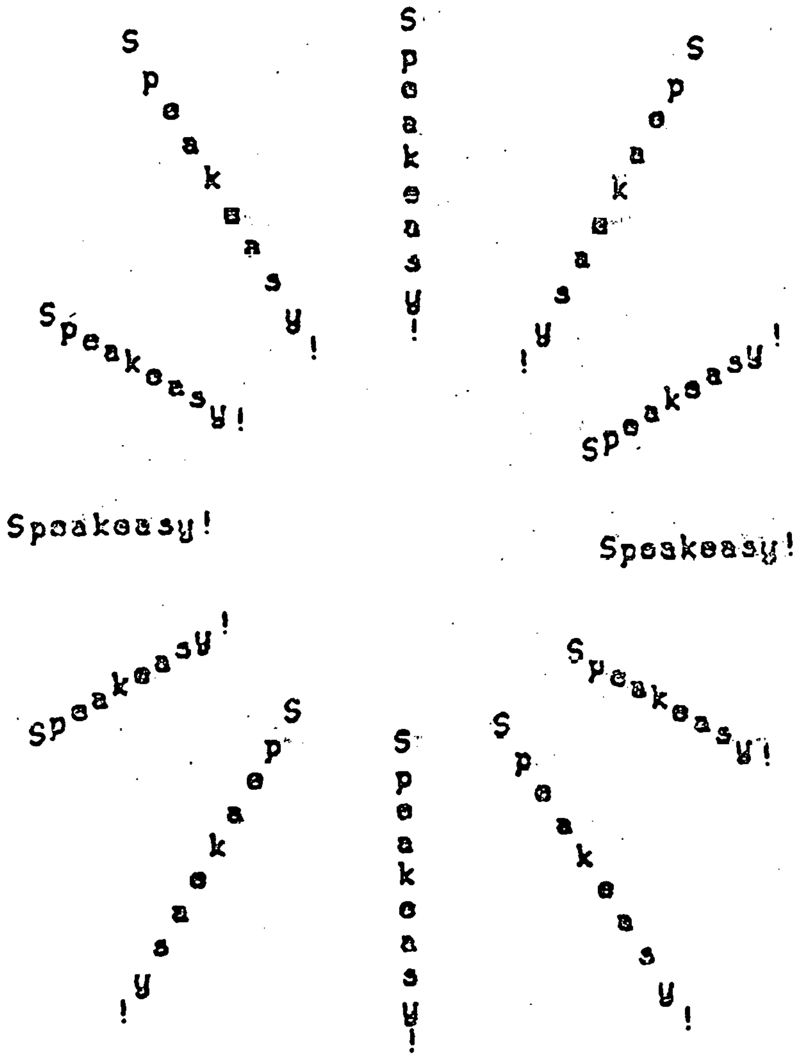


Figure 6

SPECIAL CHARACTER OUTPUT

TN CHARACTERS

"H%2 = X%2 + Y%2"

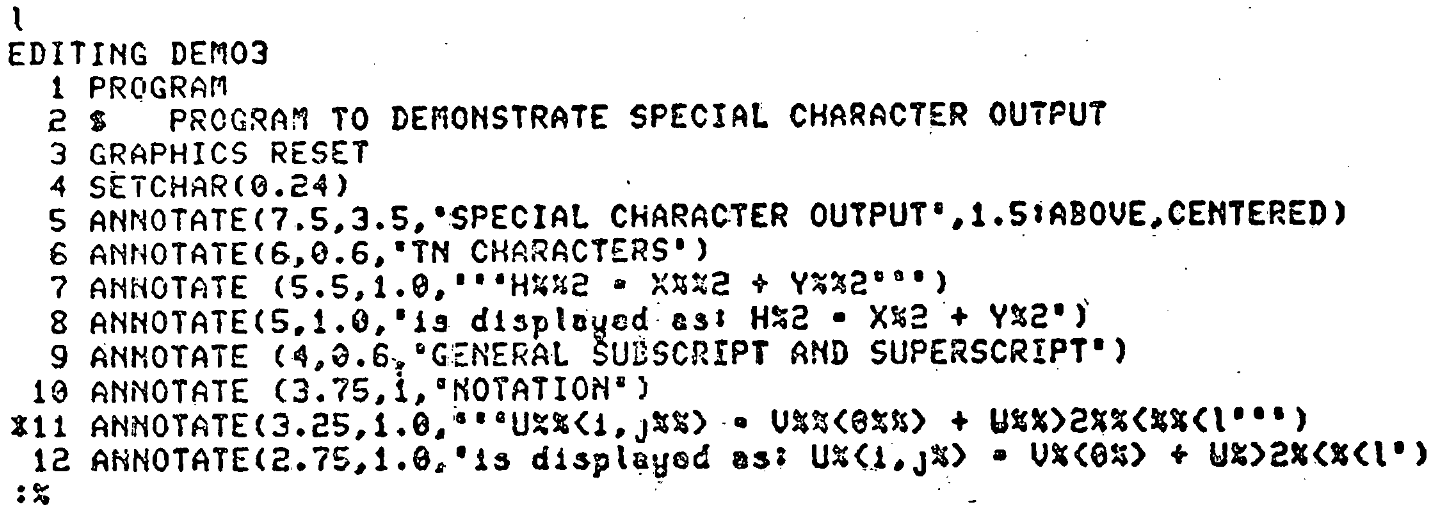
Is displayed as: $H^2 = X^2 + Y^2$

GENERAL SUBSCRIPT AND SUPERSCRIPT NOTATION

"U%<I,J%> = V%<0%> + W%>2%<%<I"

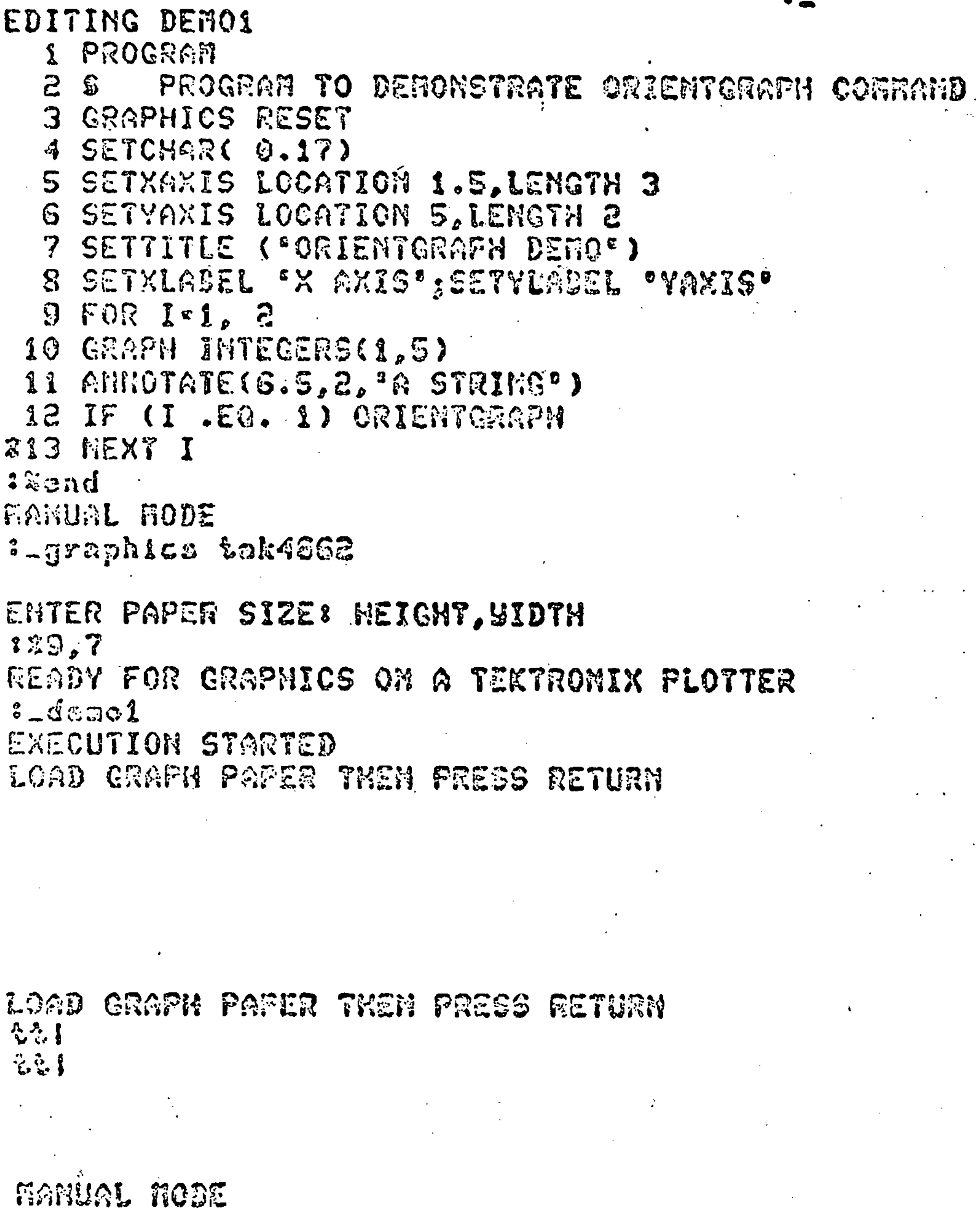
Is displayed as: $U_{I,J} = V_0 + W_I^2$

Figure 7



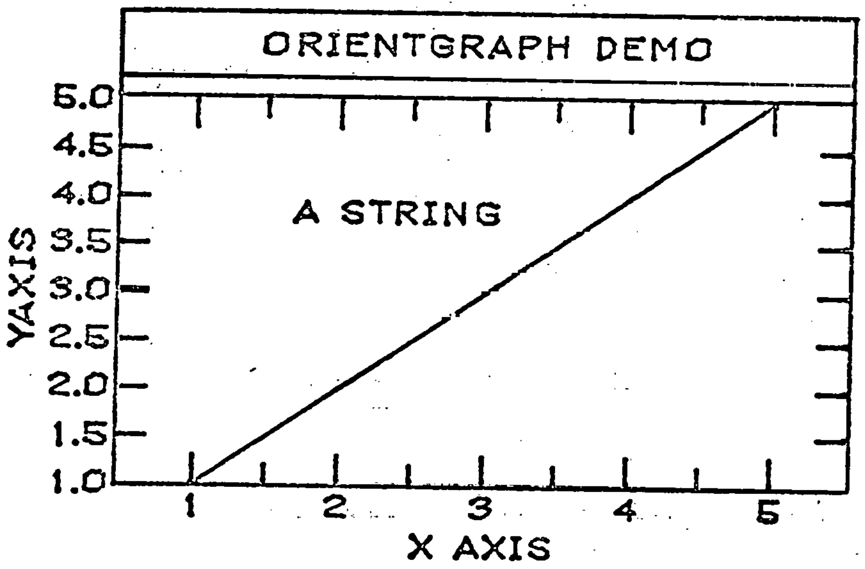
```
l
EDITING DEMO3
  1 PROGRAM
  2 $   PROGRAM TO DEMONSTRATE SPECIAL CHARACTER OUTPUT
  3 GRAPHICS RESET
  4 SETCHAR(0.24)
  5 ANNOTATE(7.5,3.5,'SPECIAL CHARACTER OUTPUT',1.5:ABOVE,CENTERED)
  6 ANNOTATE(6,0.6,'TN CHARACTERS')
  7 ANNOTATE (5.5,1.0,'''H%%2 = X%%2 + Y%%2''')
  8 ANNOTATE(5,1.0,'is displayed as: H%2 = X%2 + Y%2')
  9 ANNOTATE (4,0.6,'GENERAL SUBSCRIPT AND SUPERSCRIPT')
 10 ANNOTATE (3.75,1,'NOTATION')
%11 ANNOTATE(3.25,1.0,'''U%%(i,j%%) = V%%(0%%) + W%%)2%%(%%(l''')
 12 ANNOTATE(2.75,1.0,'is displayed as: U%(i,j%) = V%(0%) + W%)2%(%(l')
:%
```

Figure 8



```
EDITING DEMO1
  1 PROGRAM
  2 $   PROGRAM TO DEMONSTRATE ORIENTGRAPH COMMAND
  3 GRAPHICS RESET
  4 SETCHAR( 0.17)
  5 SETXAXIS LOCATION 1.5,LENGTH 3
  6 SETYAXIS LOCATION 5,LENGTH 2
  7 SETTITLE ('ORIENTGRAPH DEMO')
  8 SETXLABEL 'X AXIS';SETYLABEL 'YAXIS'
  9 FOR I=1, 2
 10 GRAPH INTEGERS(1,5)
 11 ANNOTATE(6.5,2,'A STRING')
 12 IF (I .EQ. 1) ORIENTGRAPH
%13 NEXT I
:%end
MANUAL MODE
:_graphics tek4662

ENTER PAPER SIZE: HEIGHT,WIDTH
:29,7
READY FOR GRAPHICS ON A TEKTRONIX PLOTTER
:_demo1
EXECUTION STARTED
LOAD GRAPH PAPER THEN PRESS RETURN




LOAD GRAPH PAPER THEN PRESS RETURN
 &&!
 &&!


 MANUAL MODE
```

Figure 9


ORIENTGRAPH DEMO
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
YAXIS
A STRING
1
2
3
4
5
X AXIS

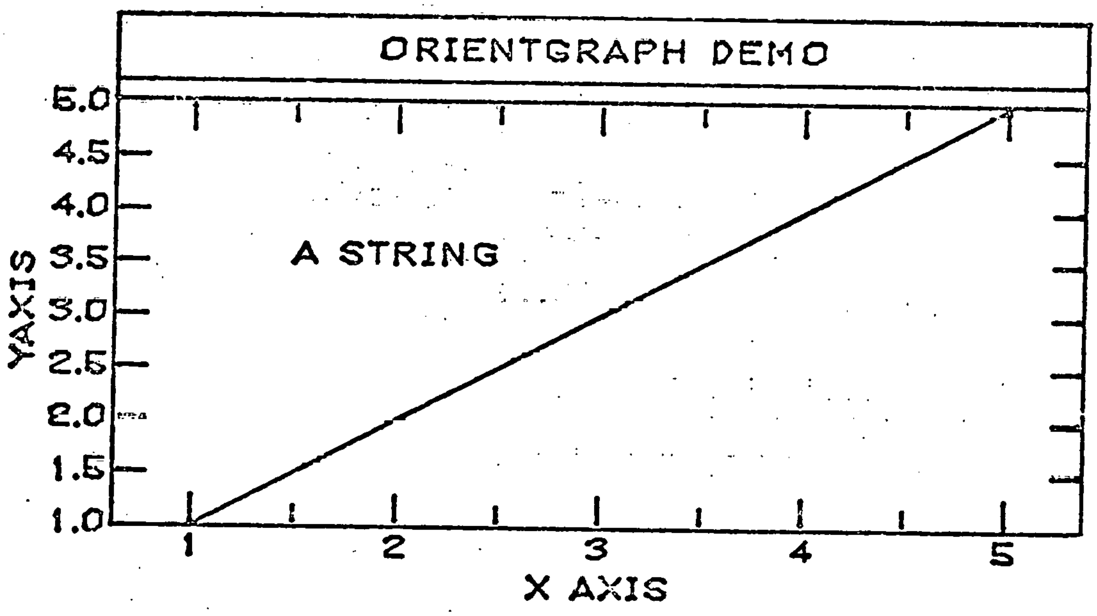
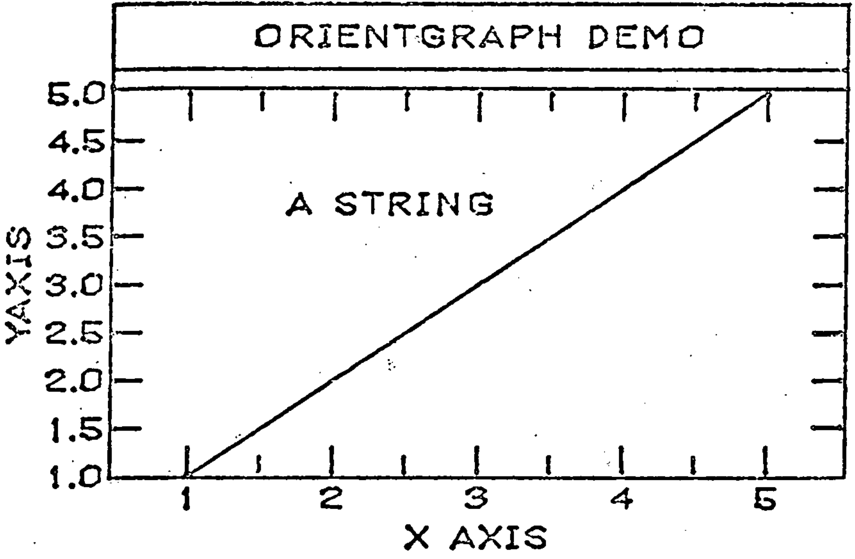

ORIENTGRAPH DEMO
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
YAXIS
A STRING
1
2
3
4
5
X AXIS

Figure 10


ORIENTGRAPH DEMO
A STRING
YAXIS
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
2
3
4
5
X AXIS

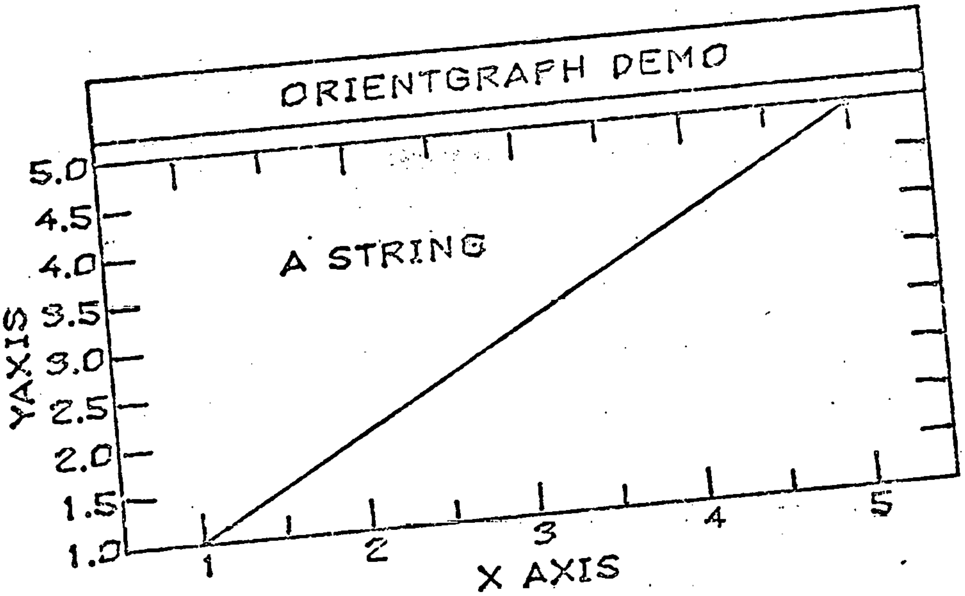
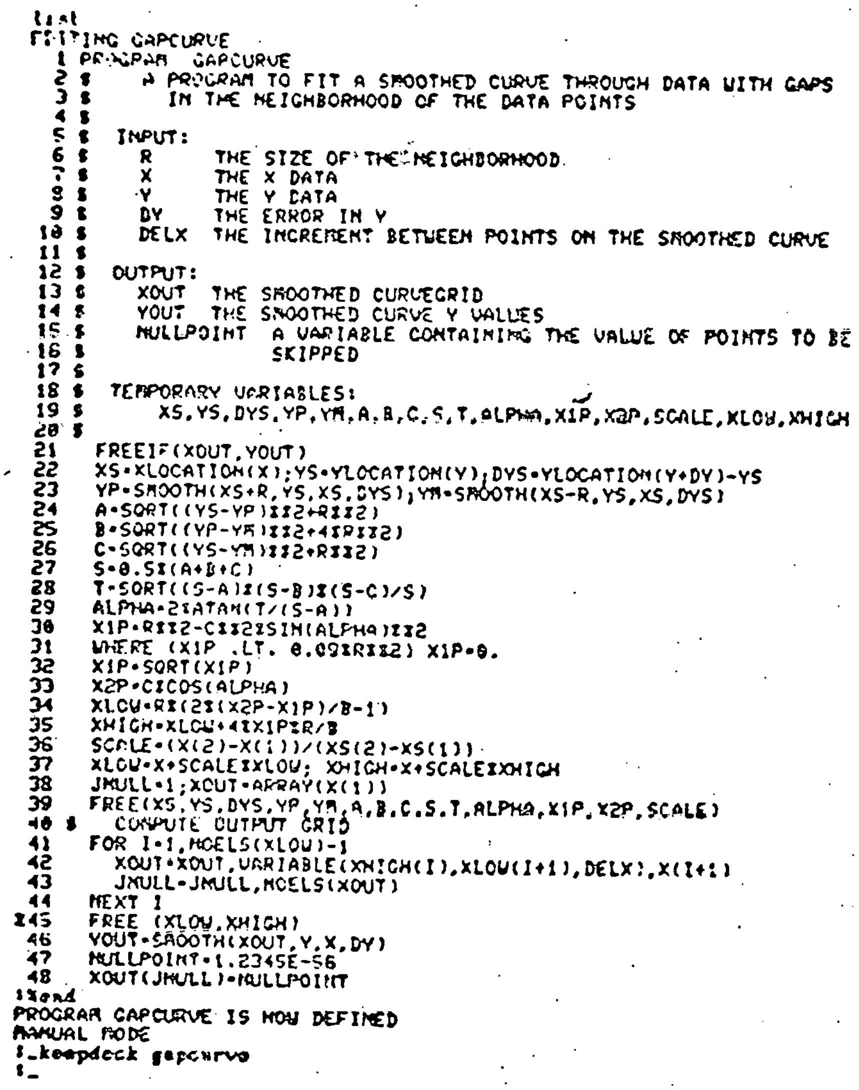

ORIENTGRAPH DEMO
A STRING
YAXIS
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
1
2
3
4
5
X AXIS

Figure 11



```
list
EDITING GAPCURVE
PROGRAM  GAPCURVE
$     A PROGRAM TO FIT A SMOOTHED CURVE THROUGH DATA WITH GAPS
$        IN THE NEIGHBORHOOD OF THE DATA POINTS
$
$   INPUT:
$     R      THE SIZE OF THE NEIGHBORHOOD
$     X      THE X DATA
$     Y      THE Y DATA
$     DY     THE ERROR IN Y
$     DELX   THE INCREMENT BETWEEN POINTS ON THE SMOOTHED CURVE
$
$   OUTPUT:
$     XOUT   THE SMOOTHED CURVEGRID
$     YOUT   THE SMOOTHED CURVE Y VALUES
$     NULLPOINT  A VARIABLE CONTAINING THE VALUE OF POINTS TO BE
$                SKIPPED
$
$   TEMPORARY VARIABLES:
$       XS,YS,DYS,YP,YM,A,B,C,S,T,ALPHA,X1P,X2P,SCALE,XLOW,XHIGH
$
    FREEIF(XOUT,YOUT)
    XS=XLOCATION(X);YS=YLOCATION(Y);DYS=YLOCATION(Y+DY)-YS
    YP=SMOOTH(XS+R,YS,XS,DYS);YM=SMOOTH(XS-R,YS,XS,DYS)
    A=SQRT((YS-YP)**2+R**2)
    B=SQRT((YP-YM)**2+4*R**2)
    C=SQRT((YS-YM)**2+R**2)
    S=0.5*(A+B+C)
    T=SQRT((S-A)*(S-B)*(S-C)/S)
    ALPHA=2*ATAN(T/(S-A))
    X1P=R**2-C**2*SIN(ALPHA)**2
    WHERE (X1P .LT. 0.09*R**2) X1P=0.
    X1P=SQRT(X1P)
    X2P=C*COS(ALPHA)
    XLOW=R*(2*(X2P-X1P)/B-1)
    XHIGH=XLOW+4*X1P*R/B
    SCALE=(X(2)-X(1))/(XS(2)-XS(1))
    XLOW=X+SCALE*XLOW; XHIGH=X+SCALE*XHIGH
    JNULL=1;XOUT=ARRAY(X(1))
    FREE(XS,YS,DYS,YP,YM,A,B,C,S,T,ALPHA,X1P,X2P,SCALE)
$     COMPUTE OUTPUT GRID
    FOR I=1,NCELS(XLOW)-1
      XOUT=XOUT,VARIABLE(XHIGH(I),XLOW(I+1),DELX),X(I+1)
      JNULL=JNULL,NCELS(XOUT)
    NEXT I
    FREE (XLOW,XHIGH)
    YOUT=SMOOTH(XOUT,Y,X,DY)
    NULLPOINT=1.2345E-56
    XOUT(JNULL)=NULLPOINT
$%end
PROGRAM GAPCURVE IS NOW DEFINED
MANUAL MODE
$_keepdeck gapcurve
$_
```

Figure 12

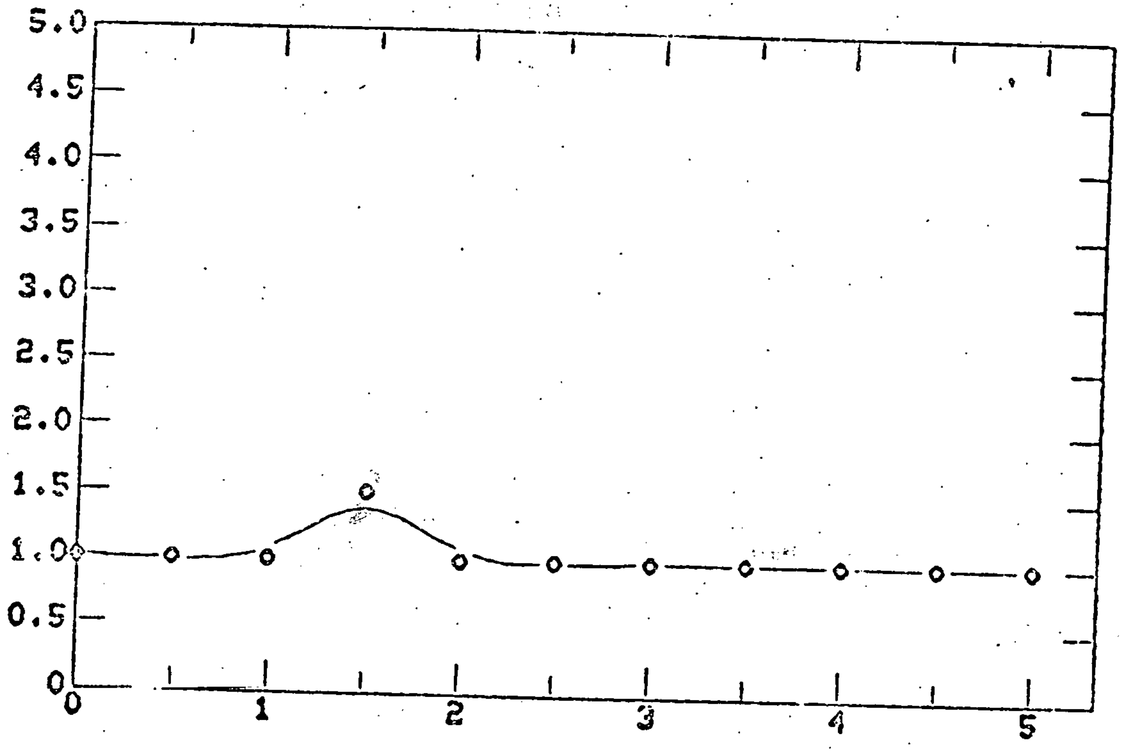


Figure 13

SPEAKEC: MINICOMPUTER SPEAKEASY

Mike Bailey

Dave Anderson[1]

Abstract

Speakec is a minicomputer-based interpretive language modeled after Speakeasy[1]. It is being developed at the Computer Aided Design and Graphics Laboratory in Purdue University's School of Mechanical Engineering. Speakec runs in 18K on a PDP 11/40 under the UNIX operating system.

Introduction

Although there has been an effort to maintain compatibility with Speakeasy, certain concessions have been made to the limitations of a minicomputer. Speakec has only the manual (:_) mode, and the sole variable "kind" is real. However, variable "klasses" include scalars, arrays, vectors, matrices, and a special geometric modeling data structure of our own design.

The attempt to model the Speakeasy environment on a minicomputer has its origins in several goals:

- Our primary objective is research into interactive computer graphics methods and techniques. Speakeasy's powerful ability to manipulate numbers seemed to be an excellent method for achieving very tight control over the definition and modification of a 3D object.

- The on-site existence of a Speakeasy-based system would enhance our educational environment.

- Using Speakec as a central monitor and data manipulator,

[1] Authors' address:
Computer Aided Design and Graphics Lab
School of Mechanical Engineering
Purdue University
W. Lafayette, IN 47907

many graphical ideas and techniques may be integrated intc one Computer Aided Design (CAD) system.

This third objective, using Speakec as a framework for a CAD system, has been of particular interest to us. The use of the "plug-in" linkules creates a unique CAD environment: minicomputer-based with a large variety of functions, but all functions available within one program rather than a loosely scattered group of programs. This lends a coherence that, to our knowledge, is unavailable elsewhere on a minicomputer.

Implementation

Speakec is implemented on a PDP 11/40 under UNIX[2], a PDP-11 operating system developed at Bell Telephone Laboratories. UNIX is a multi-user system for PDP-11's that has been rapidly growing in popularity. The implementaion language for Speakec is C (hence the name SpeakeC), a structured programming language running under UNIX which is similar to PL/I and Pascal.

The UNIX-C combination produces a very powerful environment. It would not be an exaggeration to say that our minicomputer model of Speakeasy would have been impossible without it. This team allows the user access to system routines without nightmarish encounters with assembly language. The system routines are useful in performing such difficult tasks as expanding the program's data area, linkule loading, special disk accesses, and the initialization of other processes on the system from within a program.

The heart of Speakec is its linkule loader. When a linkule is to be used, the loader first searches memory to see if that linkule has already been loaded from a previous use. If not, the loader searches a list of available linkules to determine its disk location. A linked list of allocated memory is then examined to determine if a large enough slot exists for the linkule. If not, other linkules are removed from memory, consolidating slots until either a large enough slot opens up or the load ultimately fails.

As the linkule is loaded, certain machine language words are relocated (changed to reflect the memory locations at which the linkule is being loaded) according to flags imparted to the linkule by a standard option of the C compiler. Thus, linkules may be loaded from the disk to any available memory slot.

The Speakec processor occupies 18K words of PDP-11 memory. Our current linkule libraries consist of about 150 linkules[3].

Applications

Some of the current developments in Speakec are shown in the figures. Figure 1 shows a cross section for a keyed shaft being created by graphically editing a 14-sided polygon. Figure 2 shows a twisted keyed shaft produced by simultaneously rotating and translating the cross section. Figure 3 shows the same object with hidden lines removed. Although this particular hidden line program is not a part of the Speakec system, a linkule was written to produce a data file to be used as input to the hidden line program [4].

Figure 4 shows a goblet constructed with 36 points. Figure 5 shows the same goblet with surface patches fitted through the points. Surface patches are continuous, analytically defined elements of surface area which are also continuous with adjacent patches. Each boundary curve of the patch is formed with a parametric cubic curve. Because of their smoothness, surface patching is a popular tool for N/C machining. While only the patch boundary curves are shown in Figure 5, Figure 6 demonstrates the same patch network with interior cross-hatching applied to each patch.

Figure 7 is a quarter section of a disk being analyzed for radial (from the top toward the center) loading. Figures 8 and 9 show the results of Speakec's plane stress finite element routines. A linkule produced a data file representing the geometry of the quarter disk. An analysis program capable of reading this data file was then triggered and ran in the "background" (in UNIX one job may start another). When completed, the analysis program produced an output data file which was then accessed by other linkules. Figure 8 shows exaggerated deflections and Figure 9 shows the stress distribution represented as vertical displacements. The conversion from output data to meaningful graphical constructs was easily accomplished using standard Speakeasy grammar.

Another application currently being developed is in geometric modeling. Closed volumes are defined and may be translated, rotated, or scaled. Given two bodies, it will be possible to take their union, intersection, or difference to form a third body. In this manner, fairly complex 3D objects may be built very quickly and painlessly. Physical properties such as surface area, volume, center of gravity, and moments of inertia will be computable from the body data structure.

The Future

One of the main drives in our future work is to move Speakec to a minicomputer which is less obsolete than the 11/40. Recently we have had limited access to a PDP 11/70, and preliminary tests show speed increases by more than a factor of 50 times! Also, UNIX on the 11/70 allows a program's instruction and data areas to be separated and each occupy its own 32K block of memory. Under the 11/40, the entire program must occupy a single 32K block. Thus, more program expansion will be possible in the future.

Another advantage that the 70 would allow us is the presence of Fortran IV-Plus. We believe that a suitable interface may be developed to utilize the power of the Fortran IV-Plus compiler to allow Fortran linkules to be written, even though the Speakec processor is written in C.

Another idea on the drawing board is the use of virtual data, that is, allowing certain parts of the user-defined data area to be placed out on disk when not in use. This will allow for larger overall data structures, necessary for more elaborate 3D surface and finite element work.

Acknowledgements

Our profound thanks to Stan Cohen. Without his openness, cooperation, and encouragement, this bird would have never gotten off the ground.

This work was indirectly funded in part by NASA grant NSG-2192, Ames Research Center.

References:

[1] Cohen, Stanley and Pieper, Steven C., The SPEAKEASY-3 Reference Manual/Level MU, Argonne National Laboratory Report ANL-8000, August 1977.

[2] Ritchie, Dennis M. and Thompson, Ken, "The UNIX Time-Sharing System," Communications of the ACM, Volume 17, Number 7, July 1974, pp365-375.

[3] Bailey, Mike and Anderson, Dave, "Speakec User's Guide," Computer Aided Design and Graphics Lab, Purdue University, W. Lafayette, Indiana, August 1978.

[4] MOVIE.BYU is a general purpose computer graphics software system written by Hank Christianson and Mike Stephenson. The system is designed around a program which generates line drawings and continuous tone shaded images.

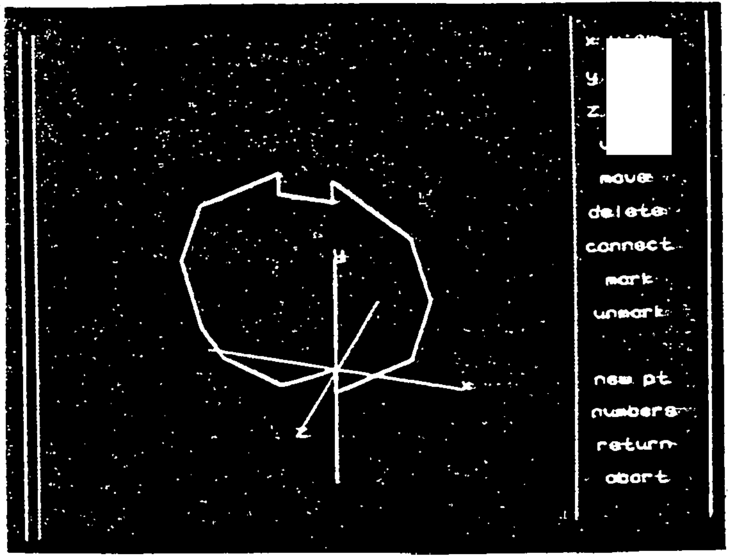


Figure 1:

Graphically creating a keyed cross section from a 14-sided polygon

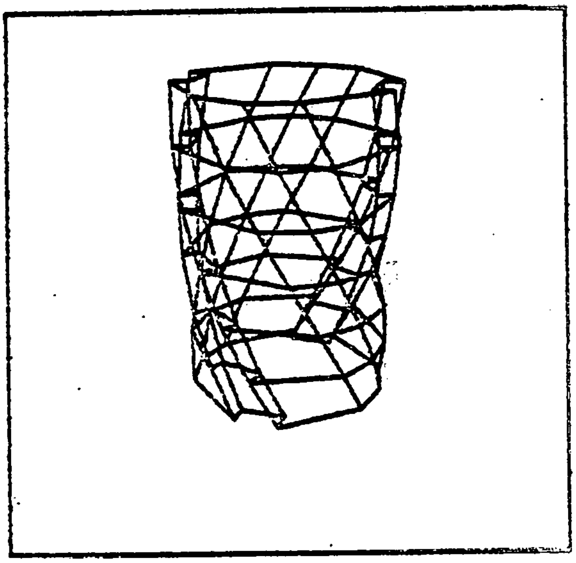


Figure 2:

Transformation of the cross section to produce a twisted shaft

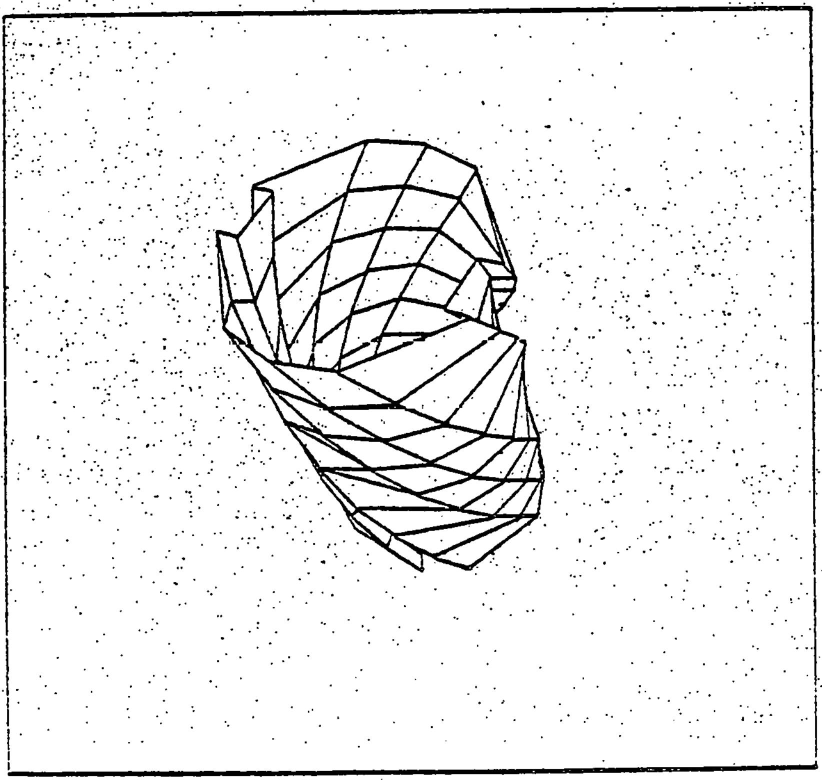


Figure 3: Shaft with hidden lines removed

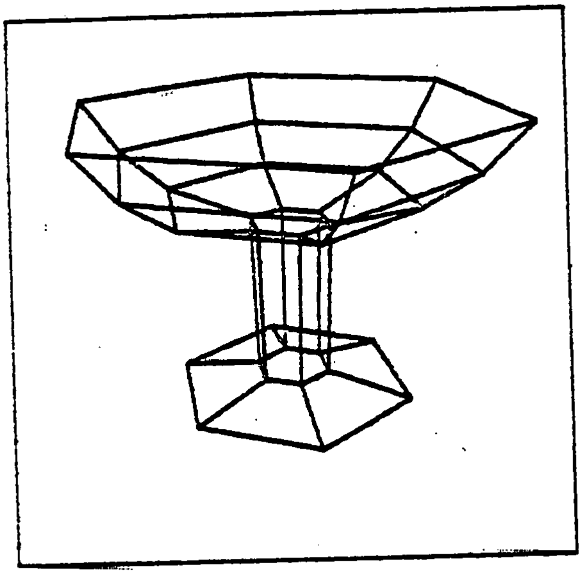


Figure 4:

Goblet with straight line connections

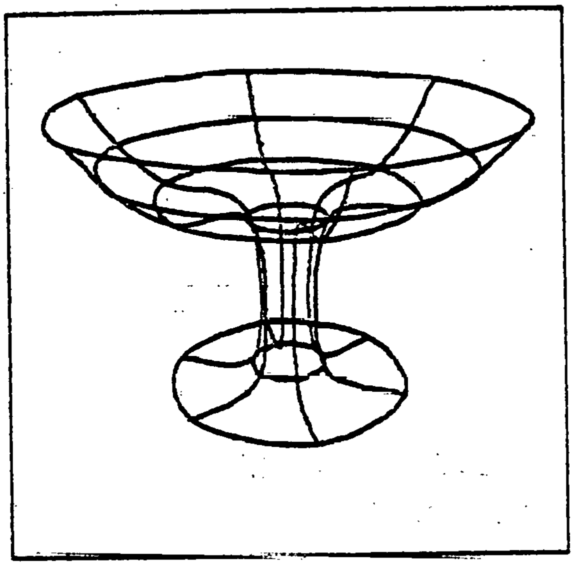


Figure 5:

Goblet with patch boundary curves

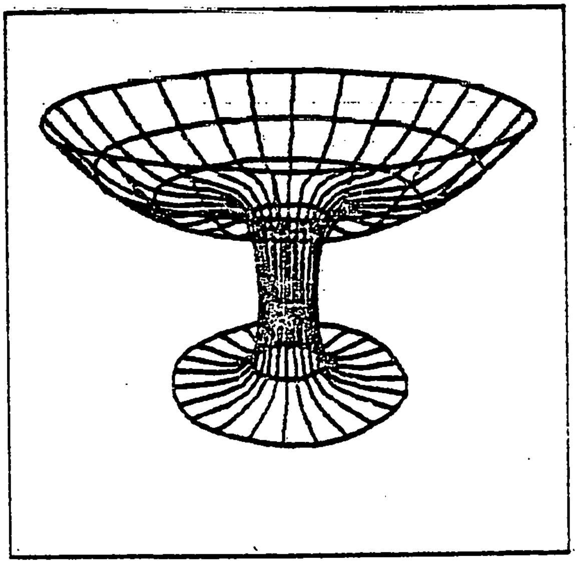


Figure 6: Patched goblet with interior patch traces

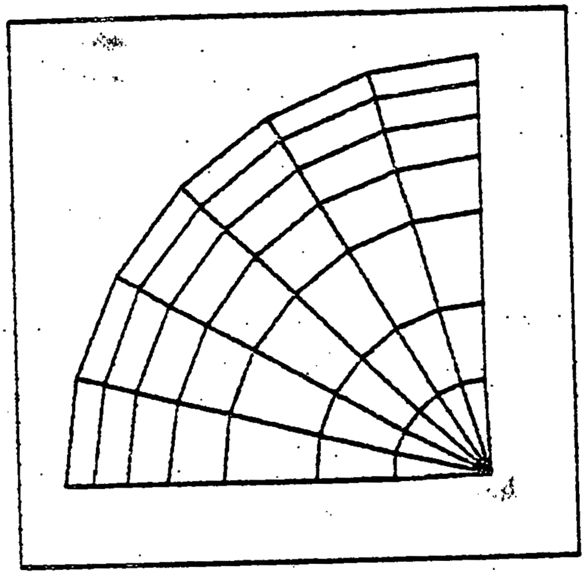


Figure 7:

One quarter of a disk

Figure 8:

Exaggerated deflections from radial loading

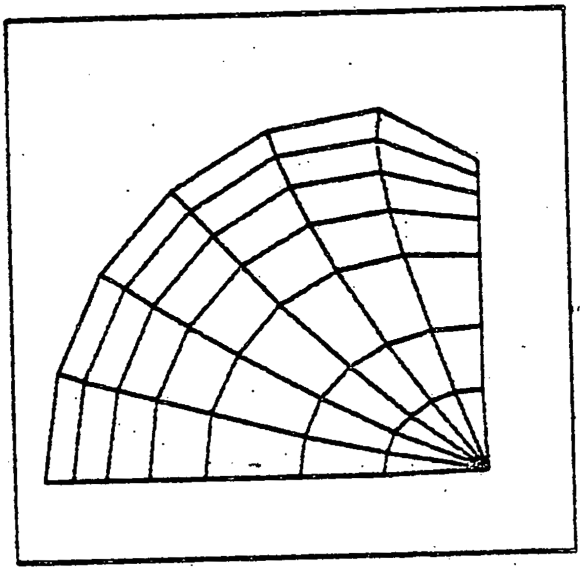
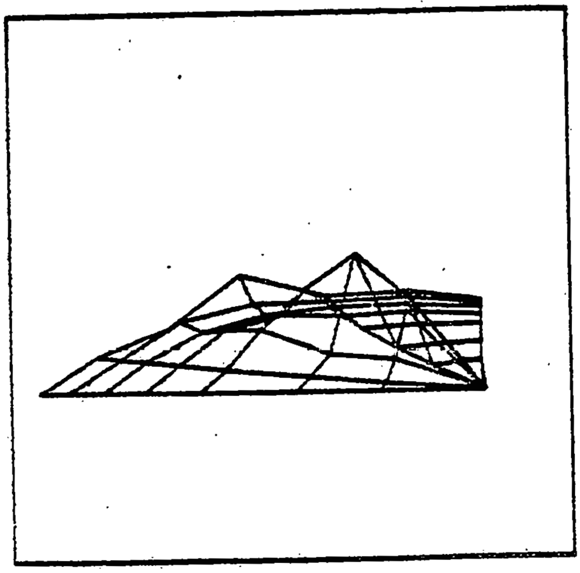


Figure 9:

Stress distribution represented as vertical displacements



COLOUR GRAPHICS within SPEAKEASY

by

A. J. JONES
Software Consultant
UNILEVER COMPUTER SERVICES LIMITED

What do the figures 38, 22, 36 conjure up in your mind ? Perhaps a colour picture of the object you imagine would be appreciated.

Disappointed when it turns out to be a squat box (Slide 1) ? There is no doubt that a coloured diagram is more easily understood than a table of numbers or even a black and white sketch use different shading patterns to provide contrast.

Few people today, with all the visual aid devices available, would make a presentation without the use of colour to emphasise words or detail on their slides. Nevertheless even for exception reports most systems designers do not look beyond printing columns of figures; few people, if any, think of printer plotting to produce simple bargraphs or histograms.

With the increasing availability and use of interactive processing, the problem has been shifted back on the user. By putting him in front of a Vdu, he is persuaded to view his own data in a reactive manner. Unfortunately there is not an abundance of software tools designed for use by the non-dp person. Some do exist, such as RAMIS and Easytrieve, and achieve varying levels of useability; I have given these systems the group name of CUDOS - Complete User Direct Operated Systems. Pre-eminent in this class is Speakeasy.

As you all know Speakeasy allows the user to manipulate his data without the need to know the more intricate details of programming. This has been extended into the area of graphical representation on such devices as storage tubes and plotters. As colour diagrams are easy to appreciate, UCSL decided to investigate the availability of a suitable colour vdu to use as a terminal into Speakeasy via our timesharing system. We could then extend Speakeasy to have the ability to produce histograms and bargraphs with the same ease as current graphics operators.

We were unable to find a colour vdu that had a standard ASCII code interface. Apparently most colour vdu's have been used in the process control field and normally directly coupled to minis. In the end we chose an Intecolor 8051 produced by Intelligent Systems Corporation as being the easiest to program to fit into our tp environment. This involved translating between the internal code of the micro in the vdu and standard ASCII transmission code.

Cont'd...

The vdu has a 48 line by 80 character screen; has six colours as well as black and white, any of which can be used as background or foreground for any character position. It had been extended to include internal software to facilitate the plotting of points, vectors and bars automatically on a 160 by 192 grid (slides 2 - 4).

Our first aim was the production of histograms to meet the need to present financial data. To present a picture that was understandable and would be left undisturbed while further commands were entered, the screen is split into three areas when in graphics mode. At the top is a four line key area, then a histogram area of forty one lines giving a plot area of 160 x 160 grid and a three line scrolled area for command entry.

A listing of the various draft help documents is attached (shown on slides 5 - 10). Also attached are tabulations of the various groups of data used in producing the examples of histograms from the minimum input to the most comprehensive.

Unfortunately, it has proved impossible to get colour prints of the pictures of the displays produced in time to go with this synopsis of my presentation.

```
  help intecolr
COLOUR GRAPHICS
*
 UCSL has programmed an INTECOLOR 8000 series vdu to accept/send ACSII
 transmission characters(eg 7 bit) by applying conversion to/from its
 internal 8 bit code.
*
  When used in graphics mode the screen is split into three areas:-
       (i) A title and key area of 4 lines;
      (ii) A histogram area of 41 lines;
     (iii) A command area of 3 lines which is scrolled.
*
 There are three commands and two objects used in the graphics mode.
*
COLHISTO (D1,...,D6:I:K1,...)   draws a histogram.
ADDHISTO (D:K)   adds columns to the current histogram.
CUMHISTO (D1,...,D6:I:K1,...)  draws a cumulative histogram.
TITLE  is used to title the histogram.
COLOUR="fgbgV1V2V3V4V5V6" is used to set the selection order of colours
*
AJJ UCSL
:_
```

```
  help colhisto

COLHISTO (D1,...,D6:I:K1,...)   draws a histogram.
 D1      is a two dimensional array mxn where m=<6 and mxn=<72;
         each row is treated as a different variable.
 OR
 D1,D2,...,D6 are a set of 1 to 6 one dimensional arrays of equal size
         such that the total number of elements is not more than 72.
These represent the dependent variable(s);automatic scaling takes place;
they are drawn side by side in different colours as set in COLOUR.
 I       is the independent variable and is a 1xn array, where n is the
         number of columns in D1. The values of I are converted to
         character if necessary and used to title the X axis.
 K1      can be a two dimensional character object;
 K1,K2,...,K6 can be a set of 1 to 6 one dimensional charater objects;
       the values of the K(s) are used to title the key of the histogram;
       if not present the names of the dependent variable(s) are used.
AJJ UCSL
:_
```

```
  help addhisto
ADDHISTO (D:K)   adds columns to the current histogram for the dependent
        variable D which is a one dimensional array with the same number
        of columns as the original D1.
        If any value in D exceeds the current height of the histogram, a
        comment is issued and the column for that value drawn to the top
        of the histogram.
        K is used to add to the key;if not present then the name "D" is
        used.
AJJ UCSL
:_
```

```
  help cumhisto
CUMHISTO (D1,...,D6:I:K1,...)  draws a cumulative histogram.
 D1     is a two dimensional array mxn where m=<6 and mxn=<70;
     each row is treated as a different variable.
 D1,...,D6 are a set of 1 to 6 one dimensional arrays of n elements
     each where n =< 71;
 These are the dependent variables;automatic scaling takes place;
 they are drawn one above the other in the order given,each in a
 diffent colour as set in COLOUR.
 I      is the independent variable and is a 1xn array, where n is the
     number of columns in D1. The values of I are converted to
     character if necessary and used to title the X axis.
 K1     can be a two dimensional character object;
 K1,K2,...,K6 can be a set of 1 to 6 one dimensional charater objects;
     the values of the K(s) are used to title the key of the histogram;
     if not present the names of the dependent variable(s) are used.
AJJ UCSL
:_
```

```
  help title
TITLE   if a character object named TITLE exists it is used to title
        the histogram.
AJJ UCSL
:_
```

```
  help colour
COLOUR="fgbgV1V2V3V4V5V6" is used to set the colours used for
      foreground,background and the several dependent variables.
      The default value is White,Black,Red,Yellow,Green,Cyan,
      Magenta,Blue.
      The values used to set the colours are the characters which have
      keys of the appropriate colour i.e. P for black,W for white,T for
      blue,R for green,Q for red,S for yellow,V for cyan and U for
      magenta.
AJJ UCSL
:_
```

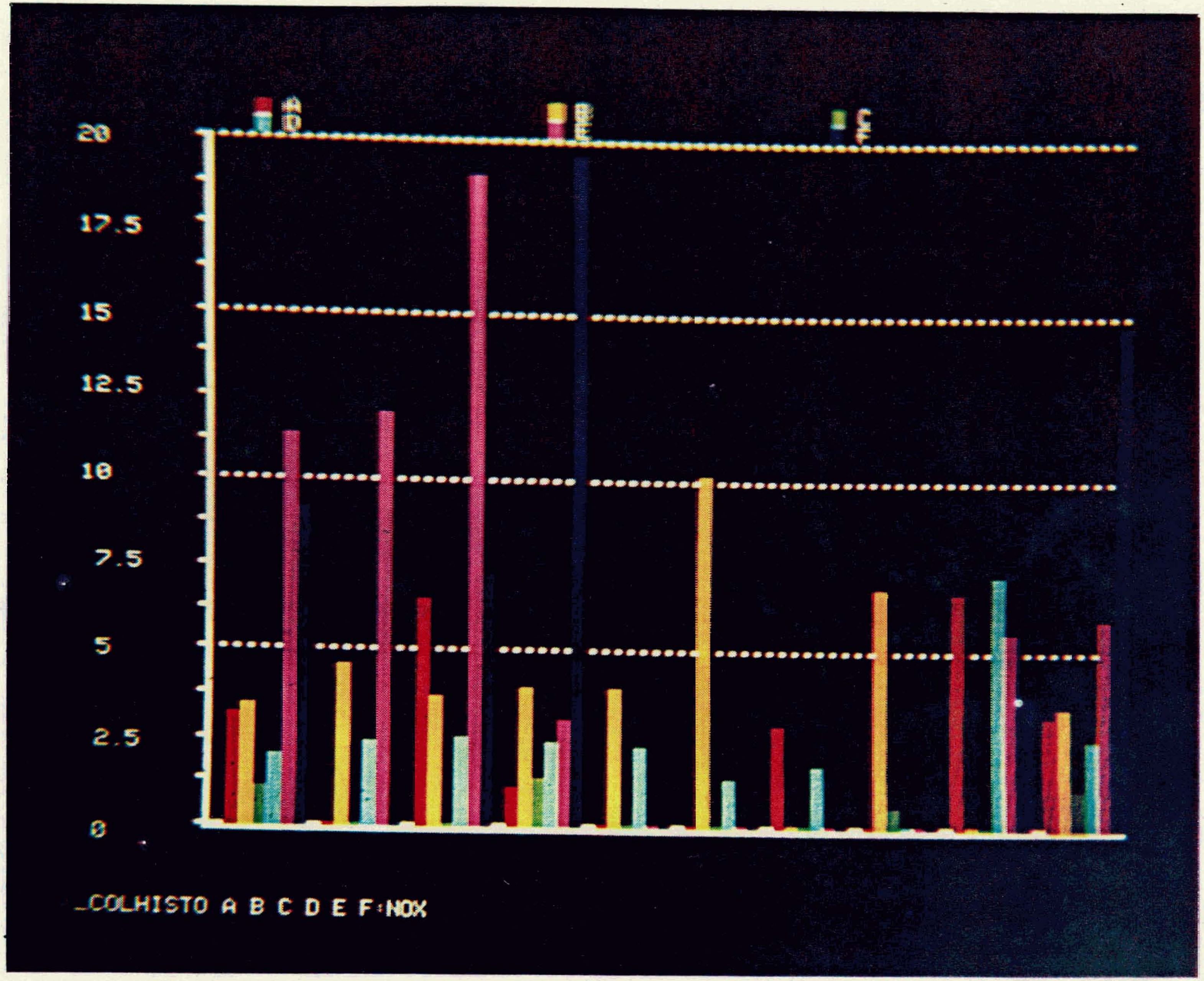
```
              INFORMATION USED IN COLHISTO AND ADDHISTO SLIDES

  X        A         B          C          D         E         F          LANG
*****  *******   ********   ********   *******   ********  ********   **********
USER1   3.1436    3.3292    1.0709     1.9833    11.413     9.1831    PL/I OPT.
USER2   0         4.4552    0          2.2746    12         0         ANS4 COBOL
USER3   6.5146    3.5833    0          2.4397    19         7.1875    FORTRAN G1
USER4   1.0435    3.8141    1.313      2.3412     2.9145   19.448     EASYTRIEVE
USER5   0         3.8627    0          2.255      0         0         RAMIS 180K
USER6   0        10.133     0          1.3462     0         0         RAMIS 210K
USER7   2.8174    0         0          1.6667     0         0
USER8   0         6.8095     .49153    0          0         0
USER9   6.68      0         0          7.2593     5.5882    3.7143
USER0   3.0522    3.3651    1.022      2.3989     5.9604   14.614

:_title
LANGUAGE COMPARISON
:_
```

DATA USED FOR CUMHISTO EXAMPLE

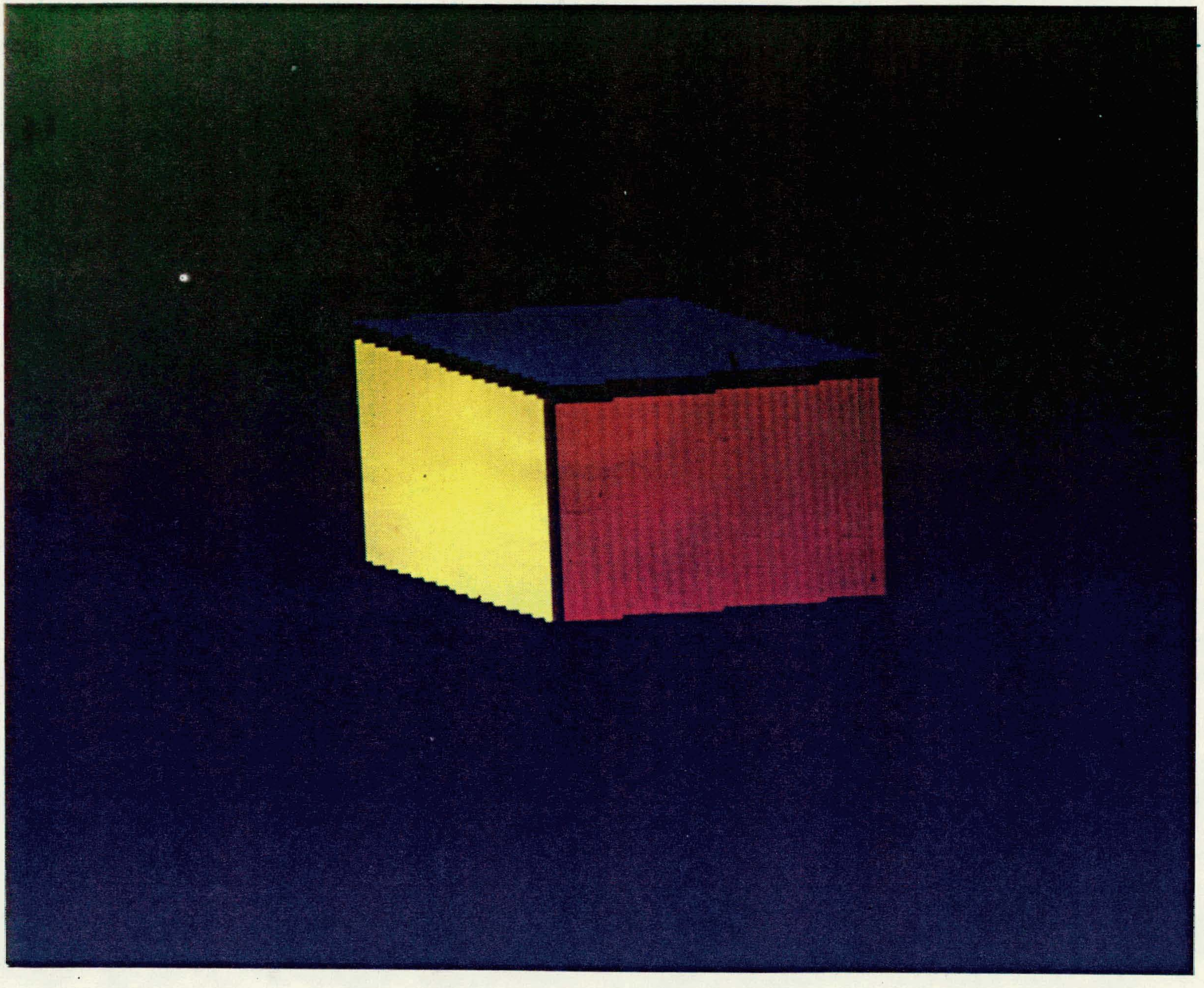
TARGETS	WORST			LIKELY			BEST			ACTUAL
*******	***************			********			**************			******
50	48	47	46	2	2	2	1.5	1.5	1.5	47
54	53	47	46	2	2	2	1.5	1.5	1.5	49
54	49	49	50	2	2	2	1.5	1.5	1.5	51
61.5	56	58	59	2	2	2	1.5	1.5	1.5	60
55	55	54	56	2	2	2	1.5	1.5	1.5	57
55	54	54	55	2	2	2	1.5	1.5	1.5	56
48	46	46	46	2	2	2	1.5	1.5	1.5	47
45	43	43.5	44	2	2	2	1.5	1.5	1.5	45
48	46	45	45	2	2	2	1.5	1.5	1.5	46.5
49	47	48	47.5	2	2	2	1.5	1.5	1.5	49
49	48	47.5	0	2	2	0	1.5	1.5	0	0
48	0	0	0	0	0	0	0	0	0	0

```
:_title
REVENUE PERFORMANCE
:_colour
WPRPSVQT
:_key
TARGET
WORST
LIKELY
BEST
ACTUAL
:_xtitle
 JAN   FEB   MAR   APR   MAY   JUN   JUL   AUG   SEP   OCT   NOV   DEC
:_
```

```
:_edit perform
EDIT COMMAND MODE
:%1
EDITING PERFORM
   1 PROGRAM
   2 GETLIST LEN
   3 GETLIST REVENUE
   4 J=INTEGERS(1,56,5)
   5 T=A1D(60);W=T;L=T;B=T;A=T
   6 T(J)=TARGETS
   7 K=INTEGERS(5,60,5)
   8 A(K)=ACTUAL
   9 II=A1D(24:J,K)
  10 I=INTEGERS(1,60);I=RELCOMP(I,II)
  11 W(I)=A1D(:WORST);L(I)=A1D(:LIKELY);B(I)=A1D(:BEST)
*12 CUMHISTO T W L B A:XTITLE:KEY
:%end
MANUAL MODE
:_
```



```
_COLHISTO A B C D E F:NOX
```



OASIS - AN OUTLOOK AND SITUATION INFORMATION SYSTEM FOR THE U.S. DEPARTMENT OF AGRICULTURE

MARTIN W. SCHWARTZ AND GEORGE E. ST. GEORGE

- The design and implementation of a computer-based information system for research and for outlook and situation processsing within the Economics Statistics, and Cooperatives Service uncovered many critical areas: user orientation, data management, analytical capability, and clarity of output. Many computer-related design criteria were considered, such as free-format vocabulary, extendability, linkage to different data and program storage devices, and error detection-correction capabilities. Data storage and manipulation techniques were emphasized since these form a vital part of the outlook and situation process. The resulting system is one in which the casual or novice user can communicate a problem to the computer in a natural manner with little knowledge about programming or internal workings of the computer.

- Keywords: Information management, data bases, programming languages, computers, research support systems.

A major function of the Economics, Statistics, and Cooperatives Service (ESCS) is to provide economic intelligence on the agricultural sector to public and private decisionmakers. In this article, we describe the development of an effective, efficient process for analyzing and reporting this information—OASIS: Outlook and Situation Information System.

THE NEED FOR A COMPREHENSIVE SYSTEM

> Behold, I set before you this day a blessing and a curse: the blessing, if ye shall hearken unto the commandments . . . which I command you this day; and the curse, if ye shall not hearken unto the commandments . . . but turn aside out of the way which I command you this day.
> Deuteronomy Ch. 11:26-28.

While much of the operational structure needed already existed in ESCS, the information flow was fragmented before OASIS and no uniform format existed. This situation resulted, in part, because of the size and diversity of the staff and the distribution of functions in the food and fiber divisions. Further, conflicts occur between conducting basic research and performing such staff functions as special analyses, outlook and situation work, and providing current economic intelligence. Finally, no common research and information system with a strong data base existed through which researchers, analysts, and policymakers could interact.

Up to 1977, the Economic Research Service (ERS, now part of ESCS) had been establishing the roots of a strong, common research system and data base, but progress moved slowly because support was informal and vested in an ad hoc group of researchers, modelers, and computer specialists drawn together by common interests. Then, in 1977, a multitude of events in the agricultural sector intensified the demand for quick and comprehensive analysis. The existing system could not keep up with the flow of analysis and information required, and ERS Acting Administrator Kenneth Farrell formed the OASIS task force which began work in October 1977.

THE DEVELOPMENT OF OASIS

Formation of the Task Force

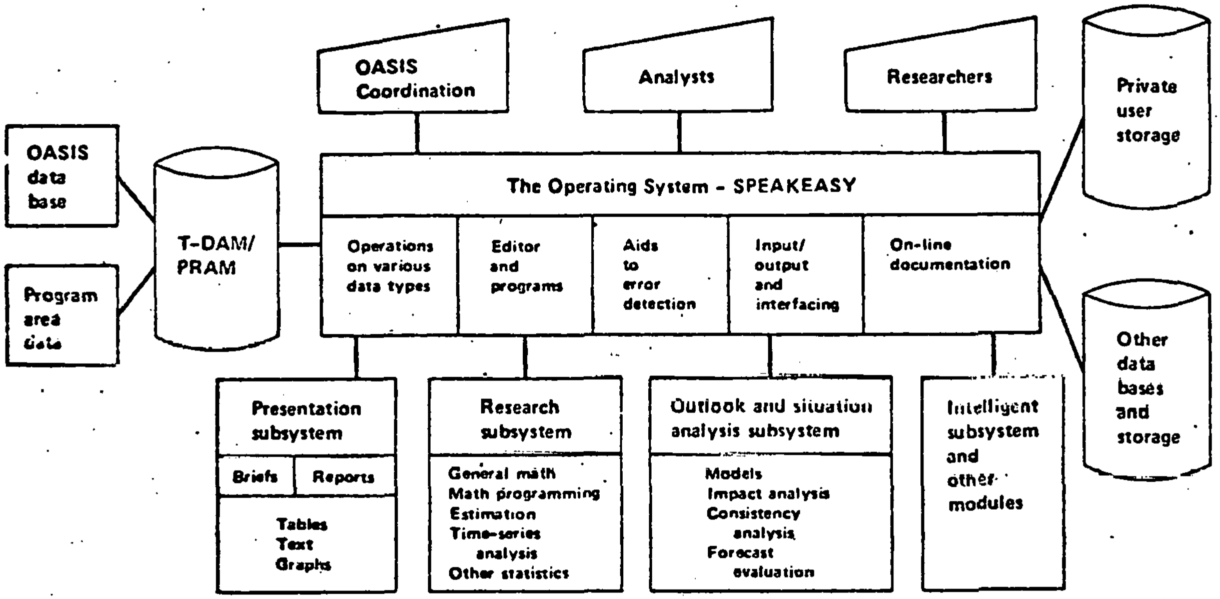
The makeup and operational license of the task force were somewhat unique. A number of ESCS staff members were selected, and were asked to work in a large room; they were charged (commanded) with analyzing and recommending a completely integrated, quick-turnaround information retrieval, analysis, and display system. They were relieved of their normal duties as much as possible so that they could attend the task force meetings. These were held in a different building from the members' office building to minimize distractions. The task force was established with no designated chairperson; no one was charged with maintaining order, or with keeping a cool head amidst the ensuing discussions.

Thus unconstrained, most of the members dredged up from the depths of their knowledge and/or biases, any points that they felt would have relevance to the task at hand. Members often emphasized their ideas by delivering them eyeball to eyeball at maximum volume: they uttered dire, prophetic warnings of bureaucratic retribution; hurled exquisite insults and heroic metaphors; used ways to attract attention or pose precise alternatives and ways to ease tension and cope with the objections of people who were not on the task force; and they knew when and where to adjourn at the end of a hard day. Despite the outward appearance of total chaos, it was a strikingly successful exercise in group dynamics. The task force produced a set of problem statements and recommendations and presented them to Agency management in late 1977 and early 1978.

Recommendations of the Task Force

The task force recommended that a computer-based system be developed which was capable of the quick turnaround necessary to support the outlook and situation process. In brief, the recommended system would consist of a well-maintained data base of historical and forecasted data, the software to effectively analyze and report this data, and a staff within a workable organizational arrangement.

FIGURE 1
The OASIS Computer System



DATA MANAGEMENT AND DATA BASES

A relevant, well-maintained data base is, of course, essential for outlook and situation work. Data management is necessary to assure the integrity of the data and to provide for timely updating and a consistent variable nomenclature. Rapid data access, though, is only one condition for complete data support. Thus, a program of thorough, careful data updating and maintenance is also needed to assure both quality and timeliness.

In addition to the primary OASIS data base, users need access to other data. Detailed data maintained by other ESCS/ECON units, ESCS/STAT units, other USDA agencies, and other outside sources are all helpful for outlook work and they are sometimes necessary for research.[3] Thus, OASIS must be linked to other data management systems. In addition, individual users also need to be able to store their own programs and data. OASIS currently provides facilities to maintain its central data base, to access any other from T-DAM (Time-Series Data Access method), to access data in some commercial data management systems, read and write character image files, and to maintain private libraries for users.

T-DAM

T-DAM is the data management system used for the OASIS data base. It is specifically designed for economic time-series data. In 1976, development began on T-DAM because there was no efficient time-series data management system available to the former Economic Research Service.

T-DAM can contain any number of variables, and it has 50 million observations. Variables are divided into logical groups (each group has a 3-character name and belongs to a specific user or project), plus a 20-character variable name (though OASIS allows only 7 characters for Speakeasy use). OASIS data currently make up about 10 percent of the total data available in T-DAM.

While all OASIS users can read almost all T-DAM data, a logical group's owner can restrict even the reading of his data to specific persons. Users can create new variables and update existing variables only within logical groups that they own. Both password-protection and use-auditing systems are included in T-DAM.

T-DAM allows various periodicities, and the user may specify the beginning and ending periods. All retrievals may be started at the same time period, and the retrieved results will be padded; that is, equivalent dates are alined in columns.

Other salient features of T-DAM appear in table 2. PRAM (Page Relative Access Method) is a recently developed enhancement which improves both space management and user performance.

[3] ESCS/ECON refers to the former Economic Research Service, and ESCS/STAT to the former Statistical Reporting Service. Both are now parts of the Economics, Statistics, and Cooperatives Service formed January 1, 1978.

Table 2—Features of T-DAM-PRAM, data management system used in ESCS

Feature
Ability to interface to any user's program via a "standardized" subroutine call
Almost unlimited amount of data on-line at one time
Data series logically divided into "logical groups", but in one centralized physical file
System operable in both interactive and batch modes
Data access protected so only the owner of a "logical group" (or someone he specifies) can alter his data, and so the owner can restrict even the reading of his data when desired
Completely mnemonic names for the data variables
Capability for maintaining documentation information, such as units of measure, source, description, owner, last updater, and date last updated for each variable
Flexibility of internal data formats, including INTEGER, REAL, and DOUBLE-PRECISION
Automatic handling of periodicities such as annual, quarterly, and monthly
Space-accounting facility which reclaims space from deleted variables, facilitates addition of new observations to an existing variable, and allows each variable to have an almost unlimited number of observations
Retrieval of any desired contiguous subset of possible observations
Auditing facility to log who (which user) is doing what to which data; can be valuable in tracking system or user problems and in supplying management information
Archive facility for holding logical groups of variables not currently required on-line (to reduce on-line disk costs), and to be a system backup
Linkes to various software packages, such as SAS, SPSS, TPL

Other Data Bases and Data Management Packages

As the operating system permits Fortran subroutine calls, it is relatively easy to interface OASIS with any commercially available, or user-developed data management package which has a Fortran interface. Some packages for which interfaces have been developed include Total, Ramis, Starmap, SPSS, and the Federal Reserve's MDL. The Fortran input-output capabilities allow interfacing (with some decrease in efficiency) with any data set or data management package formatted in either character image or Fortran unformatted format.

THE OASIS PRESENTATION SUBSYSTEM

An ounce of image is worth a pound of performance

—Peter's Placebo

The information presentation facilities were designed to have a set of commands for creating and maintaining table, text, and graphics units. Further, these units were to be merged into user-oriented products known as BRIEFs and REPORTs. The T-DAM data management system interacts with these BRIEFs and REPORTs so that data required for tables and graphs can be automatically retrieved for use.

Tables

Speakeasy offers a tabulate command which produces tabular listings of data. While this command is flexible, it does not offer such required features as footnotes, easy dating of periodicities, and multiple labels per data line. Therefore, a series of commands were set up to allow creation, alteration, and printing of table specifications. Clerical and research staff with no computer background have learned to create table specifications within a few hours. Tables are designed to be generated from within REPORTs and BRIEFs. They require a specific environment which is created by the commands REPORT, BRIEF, and SHOWTABLE. A typical table is shown as table 3.

Text

Textual material is required to explain "hard data" presented in tables and graphs. Therefore, OASIS needed facilities for creating, editing, and printing textual information. Speakeasy provides some textual facilities. It was determined that judicious use of these commands within a Speakeasy program would provide the basic facilities required for OASIS' initial operation. However, four new commands were added. Figure 2 shows a BRIEF, including text and a TABULATE table.

Graphics

Graphical displays are a highly convenient way of conveying economic information. An integral part of OASIS, they have two levels of implementation. The first level was developed at Communications Satellite Corporation and is supplied with Speakeasy. The second level is created by combining the first-level graphics commands with other Speakeasy commands into linkule-driven prompting programs. Both levels have unique advantages and limitations.

The graphics package as supplied with Speakeasy is designed to support many different terminal types. Others can be added by the installation without undue difficulty. The main feature is that all of the graphics

U. S. DEPARTMENT OF AGRICULTURE
ECONOMICS, STATISTICS, AND COOPERATIVES SERVICE
OUTLOOK AND SITUATION INFORMATION SYSTEM

TABLE 3.--WORLD GRAIN PRODUCTION BY CROP AND MAJOR PRODUCING REGION

ID	VARIABLE NAME	UNITS	1973 HISTORY	1974 HISTORY	1975 HISTORY	1976 HISTORY
1	WHEAT (TOTAL)	1000 M.T.	372,005	356,141	349,198	415,528
2	U.S.		46,560	48,496	57,751	58,296
3	CANADA		16,159	13,295	17,078	23,587
4	ARGENTINA		6,560	5,970	8,570	11,000
5	USSR		109,784	83,913	66,224	96,882
6	EEC-9		41,393	45,391	38,105	39,539
7	OTH.W. EUROPE		9,372	11,305	10,397	11,561
8	E. EUROPE		31,631	34,107	28,485	34,614
9	AUSTRALIA		11,987	11,357	11,982	11,713
10	COARSE GRAINS	1000 M.T.	661,178	621,536	635,406	693,875
11	U.S.		186,777	150,905	185,057	193,859
12	ARGENTINA		17,935	13,793	12,438	16,860
13	CANADA		20,411	17,436	19,987	21,125
14	AUSTRALIA		24,976	21,968	20,702	16,543
15	BRAZIL		16,851	16,926	18,482	19,381
16	THAILAND		2,520	2,730	3,350	3,000
17	USSR		100,951	99,744	65,820	114,979
18	RICE ROUGH	1000 M.T.	223,469	227,339	243,109	235,402
19	U.S.		3,034	3,667	4,091	3,777
20	THAILAND		9,471	9,570	10,032	10,428

commands are the same for all terminal types; thus, users need learn only one set of graphics commands.

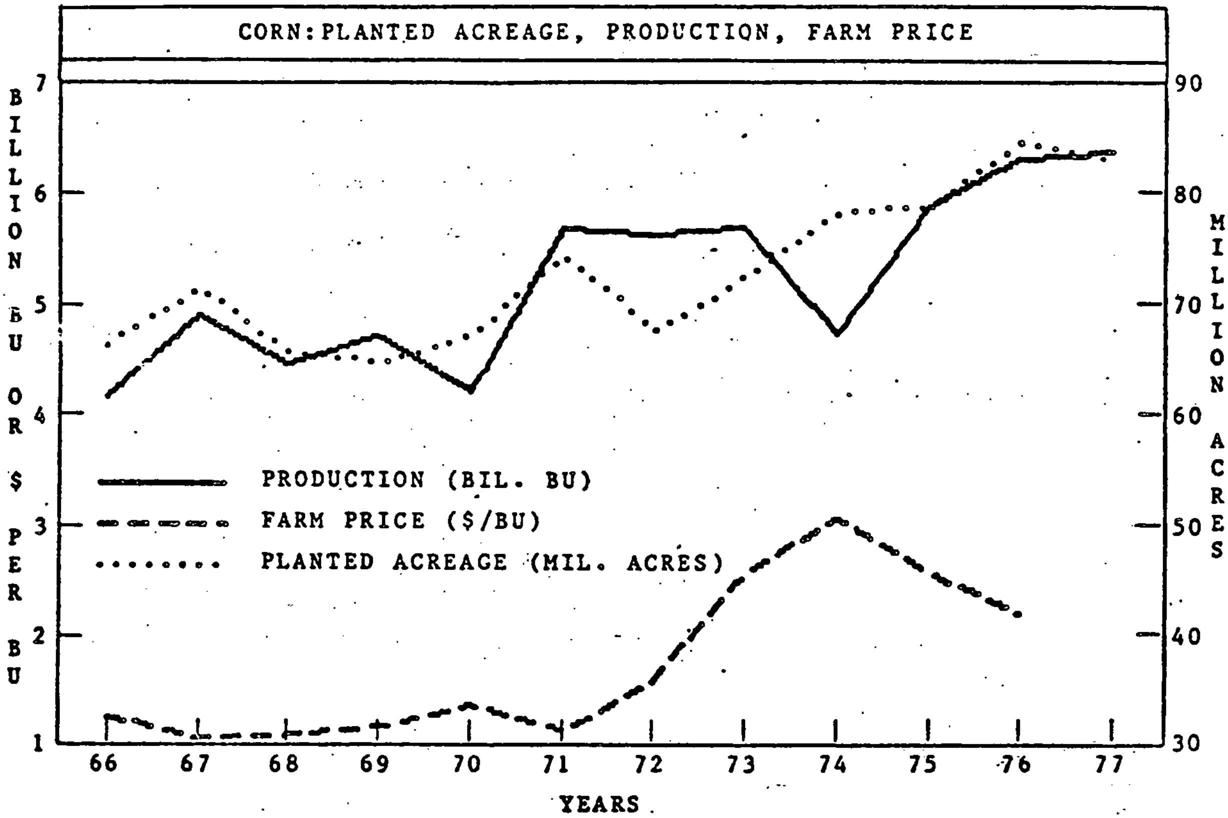
While the standard graphics commands provide easily understood methods for generating tailored graphical output, however, users identified three needs that dictated the second level of graphics capabilities. First, policymakers often must have graphics quickly. These requests cannot be planned; frequently, they must be completed within 1 or 2 working days. Second, standard graphs are needed that can be easily reproduced following updates of the data series. Many ESCS program areas keep files of such graphs for a large volume of data, some of which are updated monthly. Third, many graphs are needed only once, either as an analytical aid or for a briefing. Often, the user drew such graphs by hand.

The OASIS task force had to provide for these needs in a comprehensive yet user-oriented manner, one that could adapt to users' different requirements. In any system adopted, updates had to be handled easily. Also, many potential users of the new system did not have time to write and "debug" a level-one program whenever they needed a complex graph; thus, OASIS had to be able to meet their needs.

The solution was to install in the OASIS system a second graphics level consisting of linkule-driven Speakeasy programs which prompt the user for the appropriate parameters needed to create a graph and produce the desired graph with a single command. The graphics are also designed so that they can be driven by the REPORT and BRIEF commands.

This graphics facility has already demonstrated its worth, as it is quick and easy to use. While being developed, it was used to meet needs of Department and agency level policymakers. Production of a single graph takes about 10 minutes, specification and production included. Because little or no programming is involved, and on-line instructions are available for every portion of

FIGURE 3
An OASIS Graph



ALL FIGURES REPORTED ON CROP YEAR BASIS

the OASIS system, the uninitiated can easily produce useful results in a short time. Several improvements will be made to the graphics at both levels, and the system will be able to provide more varied types of graphics. Figure 3 shows an OASIS graph.

BRIEFS

A BRIEF is a self-contained set of materials not alterable by the final user. It covers specific time periods and data, and it is produced for a specific purpose—generally to describe a current situation or to respond to a specific question.

OASIS briefs are designed to provide information to users who have a minimum knowledge of computer systems. While a report requires the user to specify time periods and some other optional information, a brief is completely self-contained. After typing "BRIEF(name1 ,name2 . . .)", the user receives briefings on the named subjects.

An extension of the briefing facility is the "time-stamped" briefing. Each BRIEF is recorded in a table with the date and time of its storage. Another table is kept in OASIS showing the last date and time that an authorized recipient of briefs received an updated listing of available briefs. When such a user types the OASIS command BRIEFME, an index of new or altered briefs is produced. After seeing this, the user can ask for a print-out of all indexed briefs or terminate the automatic briefing and, using the BRIEF command, request only certain briefs. At this time, the user's profile is altered to reflect that he or she is up to date.

Reports

OASIS reports are predefined combinations of one or more tables, graphs, and text units. The time periods, forecast scenarios, periodicities, and some other environmental attributes are specified at the time the report is requested, allowing flexibility in displaying tables and graphs in response to ad hoc needs of researchers or analysts.

Technically, the REPORT command retrieves each requested report specification and writes its individual lines into one sequential file. The computer sets the environment based upon the parameters included with the REPORT command, and it shifts the control input from the terminal (or card reader) to that file of control cards.

The recommendations of the task force were accepted by the management team, and the task force was instructed to develop (1) an information presentation subsystem for displaying text, tables, and graphs, and (2) an improved data base and data maintenance procedures.

The starting point for OASIS proved to be the decision to combine Speakeasy, a user-oriented, interactive computer language, and T-DAM (Time-series Data Access Method), the ESCS/Economics time-series data management system. High-priority data for OASIS reports are currently maintained in T-DAM. The OASIS operating system is Speakeasy with a few additional routines.

THE PURPOSE FOR OASIS

The basic reason OASIS exists is to enable its users to store, manipulate, and display information as desired with a minimum of effort and wasted time. Within ESCS, however, a substantial amount of research and analysis is used in the outlook and situation process. Thus, the OASIS task force recognized the benefits of a system that would support both research and forecasting activities. With such a system, the end result, the report, would emerge as a natural result of the underlying analysis. While most of the phase one implementation of OASIS involved data management and display components, the marriage of the analysis and research components to them was kept in mind, especially in selecting an operating system.

More specifically, the long-range OASIS goals are: (1) to provide an effective time-series data management system for outlook and situation and research activities; (2) to assist in an integrated, comprehensive flow of information within ESCS; (3) to increase the stock of user-oriented tools; and (4) to provide a central focus for the agency's outlook and situation modeling efforts.

CHOOSING/DESIGNING THE OPERATING SYSTEM

Build a system that even a fool can use, and only a fool will want to use it.
—Shaw's Principle

The basic criterion for OASIS was that it be an interactive, well-documented, user-oriented system capable of providing results quickly. The following additional criteria were also important.

Additional Criteria

OASIS must also operate in a "batch" environment so that large or low-priority requirements can be filled at lower dollar costs. The same commands and syntax should be used in both interactive and "batch" modes.

The system should incorporate facilities for the retrieval of variables with different structures (such as scalars one- and two-dimensional arrays of numbers and literals) into a temporary workspace, the manipulation of these objects, and storage of the results.

The vocabulary should be "free-format" and extendable through standard programming to accomplish additional tasks. Some mechanism for the creation of "programs" of the system's vocabulary should be provided to help with recurring or iterative tasks, and some method should exist for storing and retrieving these "programs".

Individual users should be able to maintain their own libraries of temporary data, program, and vocabulary extensions, so, they can keep their own materials without affecting either the performance or the cost for other users.

Design Alternatives

The OASIS task force had several alternatives available. Excellent software for the individual pieces of the overall computer system exist; there are data base management systems, statistical packages, linear programming systems, and the like, which satisfy very demanding people. But they satisfy only a small part of the requirements for OASIS. The task force could have chosen one system to satisfy the numerical information communication needs, another to satisfy the text communication, a third for econometric modeling, a fourth for linear programming modeling, and so forth, and it could have tried to merge them together.

Ad hoc merging was rejected for several reasons, First, it was felt that an integrated system would serve not only to link the various parts, but as a basis for common communication among users. Second, the time available for the task force to complete the work was very short. Finally, a system that already met several of the needs would allow concentration on only those pieces that were weak or missing.

THE OPERATING SYSTEM

A graphic version of OASIS is shown in figure 1. The function of the operating system, at the center of the computer system, is to provide effective applications control. The user retrieves data from the data bases and navigates it through assorted analyses into display or storage for further use. Table 1 shows some of the capabilities. Their extremely broad range makes it difficult to categorize or list them effectively in the limited available space. Those wishing more detail should contact the Speakeasy Center.

CONCLUSIONS

> Blessed is he who expects nothing for he shall not be disappointed.
>
> Franklin's Rule

While acknowledging that OASIS could not operate without data and the appropriate manipulative tools, the task force realized that the entire process had to be engineered to account for the basic fallibility of the human animal. The software that produces the tables, text, and graphs allows the most unsophisticated user the maximum number of opportunities to make a maximum number of mistakes and still correct the errors. This ability removes the system from the purview of the technicians and places it directly in the hands of users who might want to take advantage of the full range of services under the OASIS umbrella.

REFERENCES

1. Cohen, Stanley and Steven C. Pieper, "The SPEAKEASY-3 Reference Manual, Level MU." Argonne National Lab. rpt. ANL-8000, rev. 2, 1977.
2. Gordon, Gary, "Speakeasy Lectures". Communications Satellite Corp., mimeog., 1977.
3. Havenner, A., R. Herman, and J. Condie, "Model Estimation with FEDEASY". Am. Sta. Assoc. Conf. Proc., Aug. 1977.
4. Condie, James M., "A Speakeasy Language Extension for Economists." Assoc. Computing Machinery Annual Conf., 1977.
5. The OASIS Reference Manual. U.S. Dept. Agr., Econ., Stat., Coop. Serv., forthcoming.
6. Schwartz, Raymond C., "PRAM—Page Relative Access Method". U.S. Dept. Agr., Econ., Stat., Coop. Serv., OASIS staff, mimeog., 1978.

An Approach to Writing Input-Burdened Linkules

by
Roy Conley
Federal Reserve Bank of San Francisco

One of SPEAKEASY's strong points is the ease with which new commands can be added to the language. To add a command, a user merely writes a FORTRAN subroutine, called a linkule, which performs the desired operation. By giving this linkule the proper argument list and placing it in the linkule library, the linkule becomes a new SPEAKEASY command.

Most problems associated with writing SPEAKEASY linkules are common to all programming. One way in which linkule writing differs from other programming is the way in which input is handled. Normally this poses no problems. However, when a linkule requires a great deal of input, finding an effective way to supply that input can be a challenge.

Of course, there is no such thing as a typical linkule. Linkules vary as widely in form and function as do any other programs. However, there are some generalizations which can be made. Most linkules operate on a single SPEAKEASY object or a few related objects. They either modify the object, detect some property of the object, or create a new object related to the old one in some way. The point, here, is that the input object already exists at the time the linkule is used. It is not generally necessary to modify an object or create a new one specifically to provide input to the linkule. For these linkules, input is no problem.

There is a type of linkule for which input does become a problem. It differs from "typical" in two major ways. First, its purpose is to perform involved calculations. It is not intended to modify or create SPEAKEASY objects. It may, in fact, create or modify objects, but this is a by-product of its operation rather than a primary objective. Second, a great deal of input is required. In particular, many dissimilar pieces of information are required as input. Dissimilar in the sense that they would not normally be collected together in a single object. Given the need to implement such a linkule, the difficulty faced by the linkule writer is how to provide all the input to the linkule.

There are many ways to move input to a linkule. The most common way is simply to pass the data as arguments. This assumes that all data required by the linkule already exist in some object or objects. This is normally the case. A SPEAKEASY user is more often concerned with which linkule will provide a particular result, given the data, than with how to modify the data in order to use a particular linkule. When a linkule requires a large amount of input, this method will not work. Too many arguments would be required. Other ways of handling the input must be found.

A linkule may receive its input interactively. Based on what it already "knows", it can prompt the user for more information. This approach is useful for some on-line applications. Such linkules can be run in batch mode, but the input tends to be cumbersome.

Data can be passed to linkules thru the use of keywords. Keywords are names for SPEAKEASY objects in which the linkule will look for data. These names are fixed at the time the linkule is written and cannot be changed later. Since the data will be in a location whose name is known to the linkule, it is not necessary to pass these objects as arguments.

A linkule can obtain data by doing a FORTRAN "READ" of an external dataset. This requires that the external dataset be created in advance. It also requires the user to set up the appropriate linkage prior to entering SPEAKEASY.

Finally, it is possible to combine existing objects into new objects to be passed as arguments. This is similar to the first method of passing data, but differs in one important aspect. That is that it requires the user to prepare an object specifically for use as input to the linkule. The user must give this object a structure imposed by the linkule.

Which input method is best for a given situation? There are no absolute answers. However, some observations can be made about the possible uses of each method.

The use of interactive input generally should be reserved for linkules to be used only on-line. This is particularly true in the case where branching occurs within the linkule and different input is requested depending on what has already been entered. In any case, it is somewhat awkward for someone setting up a batch run to provide a series of answers without seeing the questions. Even for strictly on-line linkules, if a great deal of input is required, interactive input is very slow. The problems of slow response and long transmission times are aggravated by the interaction. Interactive input should be reserved for on-line linkules which require only a few responses or for linkules which actually require decisions to be made by the user during execution.

The use of keywords should be approached with caution. Admittedly, their use may be attractive for certain optional inputs. The use of keywords allows options to be set without altering the calling sequence of the linkule. In addition, options can be set which will apply to several related linkules. However, the benefits do now come without cost. The user must be aware of which names are keywords to the kinkules he is using. This is true even though he does not wish to make use of the keywords. The linkule cannot tell if an object with the required name was created deliberately or was left over from another application. The use of keywords restricts a user's choice of object names. It is common for users to give objects descriptive names. The use of keywords will limit their ability to do so. Keywords also force the linkule writer to be aware of keywords used by other linkules at his site. Failure to do so could result in users having to change the names of objects between calls to various linkules, in order to satisfy the keyword requirements of each. Keywords can be useful, but should be used carefully and in moderation.

Interaction and keywords, then, are not the answer for input-burdened linkules. This leaves FORTRAN "READ"s and passing existing or specially created objects as arguments. The use of FORTRAN "READ" statements in a linkule could be an effective way to pass large quantities of data to a linkule. This is especially true when the linkule is being adapted from a batch-type program. The original input statements of the program could be retained intact. There are several limitations to this method, however. The input file must be created in advance and could not easily be changed during the SPEAKEASY session. This would make it difficult, for example, to run the linkule several times during a session with slightly different inputs. Thus flexibility, one of SPEAKEASY's strong points, is lost. The use of FORTRAN "READ"s also assumes the user to be familiar with the linkage mechanism between FORTRAN and the input data set (JCL,TSO ALLOCATE,...). This may or may not be the case. It may not even be possible to set up this linkage at a given site. If FORTRAN "READ"s are to be considered, then careful consideration must be given to who the users of the linkule will be and to the environment in which SPEAKEASY is being run.

This leaves passing arguments to the linkule. Certainly, this is the way most linkules handle input. But if a linkule requires more than a little input, some combining of objects must be done. Once again, there is no "best" way to do this, but some suggestions can be made.

A very important decision to be made is the number of arguments to be passed. SPEAKEASY has a limit of thirty arguments. Most people have a much smaller limit. The more arguments passed, the greater the risk of transposing two arguments, omitting one, misspelling one or making some similar error. Five to ten arguments seems to be a good range. Most people can handle that many without much trouble.

Limiting the number of arguments means each of those arguments will have to contain a lot of data. Many small objects will have to be combined into a few large ones. Of course, small objects are easier to handle than large ones. But given a choice between passing a few large objects or many small ones, it is generally better to opt for a few large ones.

Given that the input to the linkule will be contained in a few large objects, careful thought must be given as to what each object should contain. Often the way the particular linkule is to be used will provide some clues. As much as possible, all data in a given object should be related in some way. This may be possible only in a very general sense, such as control parameters and raw data. Data which is likely to change from one call to the linkule to another should be seperated from data which is likely not to change. This allows the user to make changes, yet not worry that he has inadvertently altered something which previously worked. It may be desirable to combine individual values into one-dimensional arrays while combining groups of values into two-dimensional arrays. Also, the intended user of the linkule may have strong feelings about how the data should be grouped, and should certainly be consulted.

The goal of any linkule writer should be to produce a linkule which is powerful enough that the user will want to use it, yet flexible enough that the user will be able to use it. The problems associated with input-burdened linkules certainly make this much more difficult. But by considering the peculiarities of each input method, the SPEAKEASY environment, the intended user, and how the linkule is to be used, the linkule-writer should be able to achieve this goal.

WEATHER AND CROP YIELD ANALYSIS SYSTEM

By

Michael D. Weiss

USDA, ESCS, CED, FSG

The Forecast Support Group Weather/Yield System has evolved from two related efforts: First, yield equations based on "weather-up-to-the-present" have been developed for within-the-year forecasting and analysis of U.S. yields for several major crops; second, a large data bank of historical and recent weather and economic data has been created, along with a software system (written in Speakeasy) permitting convenient manipulation and analysis of the data.

The system is now fully operational and permits essentially automatic forecasting of crop yields based on "weather-up-to-the-present."

The weather data bank and associated software have been designed not only to generate weather-variable values for use in the yield equations, but also to serve as a general resource in answering weather-related questions. Monthly records of temperature, precipitation, and the Palmer Drought Index are available for each climatic subdivision of each of the 48 contiguous states. The climatic subdivisions are those used by the National Oceanic and Atmospheric Administration; they correspond somewhat to the Agriculture Department's Crop Reporting Districts. The data bank currently covers January 1931 to present for temperature and precipitation and January 1931-December 1975 for the Palmer Drought Index. The temperature and precipitation series are automatically updated each week during the growing season and each month outside the growing season.

This paper represents the views of the author and not necessarily those of the U.S. Department of Agriculture.

The weather data system is capable of manipulating weather records for any chosen groupings of years, months, states, and state climatic subdivisions. Examples of its use, illustrating actual operation of portions of the system as experienced by the user at the computer terminal, appear on the following pages. The user's typed entires are circled.

Example 1: (cf. pp. 3 - 4).

The user wishes to create a time-series of quarterly precipitation indices for Iowa. The first quarter is to begin with December.

The user enters the weather type ("PRCP"); the state name ("IA"); the climatic subdivisions of Iowa under consideration ("ALL"); the months ("DEC," "JAN," etc.); and the years ("1931," "1975"). He then indicates how the quarters are to be formed (and, additionally, asks for a semi-annual index). The output is shown on page 4.

Example 2: (cf. p. 5).

The user wishes to forecast the 1978 U.S. soybean yield using "weather-up-to-the-present." He enters the Speakeasy manual mode and gives three simple commands. He obtains the forecast table shown.

Example 1: Quarterly Indices

```
SPECIFY WEATHER TYPE (TEMP, PRCP, OR PDI)
WTYPE =PRCP
ENTER STATE NAME ABBREVIATIONS
STATNAMS =IA
ENTER "ALL" OR LIST CHOSEN CLIMATIC SUBDIVISIONS IN
ORDER WITHOUT LEADING ZEROES
SUBDVSIA =ALL
ENTER MONTHS IN ORDER
MONVALS =DEC JAN FEB MAR APR MAY JUN JUL AUG SEP OCT NOV
ENTER STARTING AND ENDING YEARS FOR FIRST MONTH
STARTYR =1931
LASTYR =1975
USING D2IOR VERSION 06-13-77, 1100  (V1.02)
USING DATA FILE OF 02/11/73 AT 122248
```

```
STATCOL IS A 551 ELEMENT REAL ARRAY
ANSWERMT IS A 540 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS?
ANS =YES
ENTER MONTHS TO BE COMBINED IN ORDER
CHOICES =DEC JAN FEB
WITHOUT BLANKS, ENTER NAME OF FORTHCOMING ACCUMULATION MATRIX DECFEB
DECFEB IS A 45 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS AGAIN?
ANS =YES
ENTER MONTHS TO BE COMBINED IN ORDER
CHOICES =MAR APR MAY
WITHOUT BLANKS, ENTER NAME OF FORTHCOMING ACCUMULATION MATRIX MARMAY
MARMAY IS A 45 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS AGAIN?
ANS =YES
ENTER MONTHS TO BE COMBINED IN ORDER
CHOICES =JUN JUL AUG
WITHOUT BLANKS, ENTER NAME OF FORTHCOMING ACCUMULATION MATRIX JUNAUG
JUNAUG IS A 45 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS AGAIN?
ANS =YES
ENTER MONTHS TO BE COMBINED IN ORDER
CHOICES =SEP OCT NOV
WITHOUT BLANKS, ENTER NAME OF FORTHCOMING ACCUMULATION MATRIX SEPNOV
SEPNOV IS A 45 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS AGAIN?
ANS =YES
ENTER MONTHS TO BE COMBINED IN ORDER
CHOICES =DEC JAN FEB MAR APR MAY
WITHOUT BLANKS, ENTER NAME OF FORTHCOMING ACCUMULATION MATRIX DECMAY
DECMAY IS A 45 BY 1 REAL ARRAY
WANT TO COMBINE MONTHS AGAIN?
ANS =NO
:_
```

YEAR	DECFEB	MARMAY	JUNAUG	SEPNOV	DECMAY
1931	5.12	7.40	15.36	5.44	12.52
1932	2.63	8.60	8.00	5.91	11.23
1933	2.32	3.03	10.50	11.41	5.40
1934	2.32	7.99	12.76	8.77	10.81
1935	3.97	4.99	7.03	9.45	8.96
1936	4.72	8.86	10.42	4.27	13.57
1937	2.78	11.51	12.52	9.49	14.29
1938	3.31	5.76	12.94	2.83	9.07
1939	2.57	6.92	14.83	5.71	9.49
1940	3.52	6.76	10.34	15.16	10.28
1941	3.76	7.71	14.00	7.42	11.47
1942	3.09	8.42	15.77	4.88	11.51
1943	2.65	13.08	15.57	4.87	15.73
1944	3.37	13.40	11.13	5.76	16.77
1945	4.16	9.76	12.25	10.04	13.92
1946	2.42	10.42	13.40	7.71	12.84
1947	3.76	7.61	10.10	6.61	11.37
1948	5.20	6.45	11.25	5.62	11.65
1949	4.14	8.70	12.73	3.38	12.33
1950	3.36	14.25	16.87	7.10	13.11
1951	2.98	8.95	13.99	3.67	11.93
1952	3.66	8.40	10.98	2.76	12.06
1953	3.07	10.08	15.20	6.60	13.15
1954	2.93	7.17	8.11	4.73	10.10
1955	1.53	6.12	11.49	4.96	7.65
1956	1.49	9.35	12.15	7.90	10.84
1957	2.41	4.38	13.73	5.55	7.28
1958	2.35	13.65	11.33	9.10	16.01
1959	5.20	10.77	12.54	6.54	15.96
1960	2.64	8.61	11.94	13.31	11.25
1961	4.24	9.04	13.60	5.54	13.28
1962	1.58	8.55	11.95	4.85	10.14
1963	1.21	10.74	12.98	6.09	11.95
1964	3.64	12.18	11.36	12.02	15.32
1965	3.33	7.54	11.48	3.49	10.87
1966	2.33	7.71	13.44	6.56	10.54
1967	1.79	7.99	12.47	9.28	9.78
1968	4.84	8.34	16.19	6.25	13.19
1969	1.96	9.52	10.65	11.38	11.48
1970	4.79	5.69	9.08	9.16	10.48
1971	3.15	9.51	13.74	10.32	12.66
1972	4.32	14.65	11.37	11.73	19.47
1973	4.02	11.40	11.33	5.88	15.42
1974	3.36	9.31	11.73	5.97	13.17
1975	2.50	11.23	7.18	3.15	13.78

Example 2: Soybean Yield Forecast

```
TSO SPEAKEASY 3 MU+ 11:15 AM AUGUST 11, 1978
:_SIZE=175
:_GET SOYPROG
:_SOYPROG
EXECUTION STARTED



◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆

                  U.S. SOYBEAN YIELD FORECAST

            SOYCAST                    SOYCAST                   SOYCAST
 ........   .......       ........    .......     ........     .......
   1951      20.9            1961      23.9          1970        26.5
   1952      20.4            1962      25.3          1971        26.5
   1953      19.2            1963      24.3          1972        28.4
   1954      20.7            1964      23.7          1973        28.6
   1955      20.4            1965      25.3          1974        23.7
   1956      21.8            1966      24.8          1975        27.8
   1957      21.8            1967      24.5          1976        27.2
   1958      24.1            1968      26            1977        29.8
   1959      22.9            1969      27.2          1978        29.1
   1960      23.4


ECONOMIC ASSUMPTIONS:
U.S HARVESTED ACREAGE OF SOYBEANS FOR 1978: 63.178 MIL.
ESTIMATE OF ESCS/STATISTICS FERTILIZER INDEX FOR 1978: 181

THIS FORECAST MADE ON AUGUST 11, 1978
◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆


MANUAL MODE
:_
```

RICHARD A. STACK

Thomas L. Jacobs & Associates
53 W. Jackson
Suite 1339
Chicago, Illinois 60604

Property Analysis System (PAS)

This paper introduces a set of programs which allow the small investor to analyze the financial characteristics of a proposed investment in income property. The computations required to produce such an analysis, though simple, are nonetheless tedious and error-prone. Their length may cause the investor to overlook potentially profitable investments during his initial screening, and the possibility for computational error carries the risk of making a bad investment. The Property Analysis System described in this article addresses these problems in real estate investment in a useful way.

Introduction

The system was designed to provide the individual investor with a financial analysis of an investment (income) property. It is set up for simple purchases and is generally not capable of doing swaps and property exchanges, or of handling complicated partnership arrangements or investment trusts in which different partners share tax and appreciation benefits in different proportions.

PAS is used in SPEAKEASY's interactive mode which allows the investor to change estimates and investment profile characteristics and produce financial analyses reflecting those changes very rapidly. It does not, however, automatically search for an "optimal" strategy in any sense; i.e., no parameters of the analysis are varied over a range to select the "best" investment choice.

The PAS system is reasonably designed from a human engineering standpoint - it generally converses with its user in english, gives intelligent error messages, assumes reasonable defaults, and is forgiving when presented with unexpected or erroneous input.

Overview of Operation

Two sets of data are needed to produce a property analysis: an investor profile, containing money market data and investor-related information; and property data specific to the investment being considered.

When a user signs on to the system, he is prompted for his initials, which are used as an investor identification. If he has previously established his investor profile, it is re-printed for his review (see pg 2 of the attachment) and he is given the opportunity to modify it. If this is his initial use of PAS, he will be prompted on an item-by-item basis to supply information which constructs the profile. Suitable defaults and a method for changing them are provided. When completed, the profile will be filed, to be used in all future property analysis work. This entry path into the system is shown on pg 1 of the attachment.

The contents of the profile are detailed in both examples. As is apparent, some items are related to the mortgage market, some to the investor's financial situation, and some to general assumptions about the property's investment characteristics and transaction costs. Most of this information should remain fairly static, or vary over restricted ranges in the course of a particular analysis.

Some general comments about the content of the investor profile should be made:

1) There is not much error checking done on the reasonability of the information supplied, i.e, one could enter a negative useful life or mortgage rate, and it would be accepted. The financial analysis could probably be carried through with bizarre data values, but would be of course, useless. A system designed for less sophisticated users than those expected to use this version of PAS would require more thorough edits and cross-checks.

2) No legal or IRS rules are applied. For example, no check is made to see whether the useful life of a property qualifies it for the use of accelerated depreciation.

3) Item 11, years of analysis to produce, not only controls the amount of printout produced, but has a major impact on the financial analysis, since the property is assumed to be sold at the end of the last year shown.

4) Item 12, level of analysis, should be left at zero (the default) if a full analysis of cash flows and rate of return information is wanted; setting this to one will eliminate the printout of cash flows.

After the investor profile is retrieved or built, an existing property description is retrieved from the user's file, or built. Page 3 shows how an "index scanning" function is used to print a summary of properties on file, and provides an example of how a previously created property description is retrived, then printed. (An example of creating a new property is shown on page 7.) The property description is free-form and contains descriptive information, e.g., address, physical characteristics, broker name, etc. (at the option of the user) and five numeric data elements related to the property, i.e., purchase price, land value, estimated revenue, operating expense (including taxes), and cost of required capital improvements. Each property in an investor's file is identified by a two-character code, chosen by the user.

Once both investor and property data are present, the financial characteristics of the investment are produced by invoking the analysis function (see page 4). This consists of some descriptive information, a taxable income analysis, cash flow analysis, and an analysis of proceeds, including after tax rate of return on investment. Key elements of this printout are: cash flow before taxes (which shows to what extent the property "carries" itself); cash flow after taxes (which shows net financial impact on the investor); and annual after-tax rate of return (which allows the comparison of returns with alternative investments).

After the intitial pass has indicated that a specific property is worth further analysis, elements of the property description and/or investor profile can be modified to check various financing alternatives and financial projections or to perform crude sensitivity analyses. The examples of pages 5 and 6 show how the depreciation method to be used is modified (from 125% declining balance to straight-line) and the results of this change. Any property description and/or investor profile component can be altered and the analysis program re-run, and this sequence is repeated as often as necessary. Each change to either set of data alters the saved copy and remains in effect until subsequently modified.

```
  UPINIT
YOUR INITIALS? (NEED THREE) ABC

CREATING NEW INVESTOR PROFILE - IF DEFAULTS (IN PARENS)
  ARE OK, SIMPLY HIT RETURN

ENTER Mortgage % of price                                ( 75  ) 70
ENTER Closing expenses (as % of purchase price)          ( 3   )
ENTER Mortgage term (years)                              ( 25  ) 30
ENTER Mortgage annual % rate                             ( 9.75)
ENTER Annual % increase in gross income                  ( 5   )
ENTER Annual % increase in operating exp                 ( 5   )
ENTER Vacancy allowance (as % of gross income)           ( 4   )
ENTER Depreciation method (100 for S/L, 125 or 200)      ( 100 )
ENTER Useful life (in years)                             ( 25  ) 30
ENTER Investor income tax bracket (%)                    ( 42  ) 38
ENTER Number of yrs of analysis to produce               ( 6   )
ENTER Level of analysis to produce                       ( 0   )
ENTER Sales costs (as % of sales price)                  ( 6   )
ENTER Pessimistic est of yrly appreciation (%)           ( 0   )
ENTER Most likely est of yrly appreciation (%)           ( 5   )
ENTER Optimistic  est of yrly appreciation (%)           ( 10  )

INVESTOR ANALYSIS PROFILE


 ITEM                  DESCRIPTION                         VALUE
 ****  ****************************************************  *******
   1    Mortgage % of price                                    70
   2    Closing expenses (as % of purchase price)               3
   3    Mortgage term (years)                                  30
   4    Mortgage annual % rate                                  9.75
   5    Annual % increase in gross income                       5
   6    Annual % increase in operating exp                      5
   7    Vacancy allowance (as % of gross income)                4
   8    Depreciation method (100 for S/L, 125 or 200)         100
   9    Useful life (in years)                                 30
  10    Investor income tax bracket (%)                        38
  11    Number of yrs of analysis to produce                    6
  12    Level of analysis to produce                            0
  13    Sales costs (as % of sales price)                       6
  14    Pessimistic est of yrly appreciation (%)                0
  15    Most likely est of yrly appreciation (%)                5
  16    Optimistic  est of yrly appreciation (%)               10


OK AS IS? (Y OR N) Y
```

```
  VPINIT
YOUR INITIALS? (NEED THREE) RAS


INVESTOR ANALYSIS PROFILE


 ITEM                   DESCRIPTION                          VALUE
 ****  ********************************************         *******
   1   Mortgage % of price                                     80
   2   Closing expenses (as % of purchase price)                3
   3   Mortgage term (years)                                   30
   4   Mortgage annual % rate                                   9.75
   5   Annual % increase in gross income                        5
   6   Annual % increase in operating exp                       5
   7   Vacancy allowance (as % of gross income)                 4
   8   Depreciation method (100 for S/L, 125 or 200)          125
   9   Useful life (in years)                                  25
  10   Investor income tax bracket (%)                         42
  11   Number of yrs of analysis to produce                     6
  12   Level of analysis to produce                             0
  13   Sales costs (as % of sales price)                        6
  14   Pessimistic est of yrly appreciation (%)                 5
  15   Most likely est of yrly appreciation (%)                 7.5
  16   Optimistic  est of yrly appreciation (%)                10


OK AS IS? (Y OR N) Y


Available PAS (Property Analysis System) Commands:

Investor profile commands:
  VPINVPRT - Print current investor analysis profile
  VPINVMOD - Modify current investor analysis profile

Property related commands:
  VPBLDCRE - Create a property description
  VPBLDPRT - Print the current property description
  VPBLDRET - Retrieve a property description
  VPBLDMOD - Modify the current property description
  VPBLDSCN - Print partial descriptions of all investor properties
  VPBLDDEL - Remove a property description from your file

Operating commands:
  VPINIT   - Initialize system
  VPMENU   - Print list of commands and their function
  VPANAL   - Produce property analysis

:+
 VPBLDSCN

PROPERTIES IN FILE:

 PROPERTY ID  1ST DESCRIPTION LINE
 ***********  ********************
     R1         878 ASH ST
     R3         929 12TH ST


:←VPBLDRET
PROPERTY IDENTIFIER? (2 CHARACTERS) R1
PROPERTY R1 RETRIEVED AND MADE CURRENT PROPERTY


:←$PRINT PROPERTY DESCRIPTION
:←VPBLDPRT
 DESCRIPTIVE TEXT               DATA                VALUES
*****************  *************************  *******
878 ASH ST         Purchase Price              125000
SAMPLE PROPERTY 1  Land Value                   15000
                   Annual Gross Income          11520
                   Ann Oper Exp (incl taxes)     2700
                   Capital Improvements             0
:←
```

```
VPANAL


PROPERTY INVESTMENT ANALYSIS: AT  7:04 PM  ON August 16, 1978

 PROPERTY DATA
   ID: R1  ADDRESS: 878 ASH ST
   PURCHASE PRICE:  125000

 MORTGAGE:
   INITIAL VALUE:  100000    INT RATE:  9.75    TERM:  30
   MONTHLY PAYMENT:  859.15    BALANCE AT SALE:  95459

                      ANALYSIS OF TAXABLE INCOME
*************************************************************
YEAR  TOTAL   VACANCY  OPERATING  INTEREST  DEPREC-  TAXABLE
      GROSS   ALLOW    EXPENSE    EXPENSE   IATION   INCOME
      INCOME
****  ******  *******  *********  ********  *******  *******
  1    11520   461       2700       9724      5500    -6865
  2    12096   484       2835       9665      5225    -6112
  3    12701   508       2977       9599      4964    -5347
  4    13336   533       3126       9526      4716    -4565
  5    14003   560       3282       9446      4480    -3765
  6    14703   588       3446       9358      4256    -2945


                      ANALYSIS OF CASH FLOWS
*********************************************************
YEAR  NET OPER  PRINCIPAL   CASH FLOW  INCOME  CASH FLOW
      INCOME    + INTEREST  BEF TAXES  TAX     AFT TAXES
****  ********  **********  *********  ******  *********
  1      8359      10310      -1951    -2883      933
  2      8777      10310      -1533    -2567     1035
  3      9216      10310      -1094    -2246     1152
  4      9677      10310       -633    -1917     1284
  5     10161      10310       -149    -1582     1432
  6     10669      10310        359    -1237     1596


           TITLE                      ANALYSIS OF PROCEEDS
***************************   ***********************************
ASSUMED YEARLY APPRECIATION        5          7.5         10
RESULTING SELLING PRICES      167512       192913     221445
ADJUSTED BASES                105911       107435     109147
CAPITAL GAINS                  60450        84327     111148
TAX LIABILITY                  13845        18859      24492
PROCEEDS BEFORE TAXES          62002        85878     112699
PROCEEDS AFTER TAXES           48156        67019      88207
ANNUAL AFTER TAX RETURN(%)        12.397       18.142     23.242

:←
```

```
  $MODIFY INVESTOR PROFILE--CHANGE DEPRECIATION METHOD
:←VPINVMOD
WANT TO SEE CURRENT PROFILE? (Y OR N) N
WHICH ITEM NUMBER TO CHANGE? 8
NEW VALUE FOR Depreciation method (100 for S/L, 125 or 200) 100
WHICH ITEM NUMBER TO CHANGE?
PROFILE MODIFIED AND RE-FILED

:←
```

```
  $RE-DO ANALYSIS--USING STRAIGHT-LINE DEPRECIATION
:←UPANAL


PROPERTY INVESTMENT ANALYSIS: AT  7:13 PM  ON August 16, 1978

  PROPERTY DATA
    ID: R1  ADDRESS: 878 ASH ST
    PURCHASE PRICE:  125000

  MORTGAGE:
    INITIAL VALUE:  100000    INT RATE:  9.75    TERM:  30
    MONTHLY PAYMENT:  859.15    BALANCE AT SALE:  95459


                  ANALYSIS OF TAXABLE INCOME
 ************************************************************
 YEAR  TOTAL   VACANCY  OPERATING  INTEREST   DEPREC-  TAXABLE
       GROSS   ALLOW    EXPENSE    EXPENSE    IATION   INCOME
       INCOME
 ****  ******  *******  *********  ********   *******  *******
   1    11520   461        2700       9724      4400    -5765
   2    12096   484        2835       9665      4400    -5287
   3    12701   508        2977       9599      4400    -4783
   4    13336   533        3126       9526      4400    -4249
   5    14003   560        3282       9446      4400    -3686
   6    14703   588        3446       9358      4400    -3090


                  ANALYSIS OF CASH FLOWS
 ********************************************************
 YEAR  NET OPER   PRINCIPAL    CASH FLOW  INCOME   CASH FLOW
       INCOME     + INTEREST   BEF TAXES  TAX      AFT TAXES
 ****  ********   **********   *********  ******   *********
   1      8359       10310       -1951     -2421       471
   2      8777       10310       -1533     -2221       688
   3      9216       10310       -1094     -2009       915
   4      9677       10310        -633     -1785      1152
   5     10161       10310        -149     -1548      1399
   6     10669       10310         359     -1298      1656


           TITLE                         ANALYSIS OF PROCEEDS
 ***************************   **********************************
 ASSUMED YEARLY APPRECIATION        5          7.5          10
 RESULTING SELLING PRICES      167512       192913      221445
 ADJUSTED BASES                108651       110175      111887
 CAPITAL GAINS                  58861        82738      109558
 TAX LIABILITY                  12361        17375       23007
 PROCEEDS BEFORE TAXES          62002        85878      112699
 PROCEEDS AFTER TAXES           49641        68504       89692
 ANNUAL AFTER TAX RETURN(%)        12.267       17.927      22.97

:←
```

```
  $CREATE NEW PROPERTY DESCRIPTION
:←VPBLDCRE
PROPERTY ID? (2 CHARACTERS) RX

ENTER LINES OF DESCRIPTIVE TEXT, (ADDRESS ON FIRST LINE)
  HIT RETURN TO QUIT
? 123 MAIN ST
? ANYWHERE, IL
?
Enter Purchase Price              150000
Enter Land Value                  18000
Enter Annual Gross Income         12*875
Enter Ann Oper Exp (incl taxes)   1500+12*175
Enter Capital Improvements        2000
:←


  VPBLDSCN

PROPERTIES IN FILE:

 PROPERTY ID  1ST DESCRIPTION LINE
 ***********  ********************
     RX        123 MAIN ST
     R1        878 ASH ST
     R3        929 12TH ST


:←
```

SPEAKEASY DATABASE COMMANDS

Richard D. Schlichting
Cornell University
Ithaca, New York
August 17, 1978

Section 1 -- GSPEAK

A. Commands

Long term storage of information such as programs and objects defined in Named Storage is available in Speakeasy using the commands KEEP and KEPT. These commands, however, are not practical if a very large number of objects are to be saved since each must be assigned a unique reference name. The Generic Storage programming package is designed to store and catalog many hundreds of objects and to retrieves them indirectly through reference to their associated catalog entries.

The activation of Generic Storage is accomplished by the command GSTART, which is of the form

```
GSTART(name:NOBUFS=n,MAXSIZE=m,RECOVERY=m).
```

All of the parameters are optional, and those after the colon can appear in any order.

The parameter 'name' allows the user to specify the dataset to be opened for Generic Storage. Objects will be saved in the dataset USERID.name.GDATA, which will automatically be created if it does not already exist. However, in Batch Speakeasy, the dataset cannot be created dynamically, so it must exist and be allocated by job control statemants to the ddname 'name'. if the parameter is omitted, the dataset name defaults to USERID.GSPEAK.GDATA.

If 'nobufs=n' appears in the argument list, a maximum of n buffers will be retained in core. The default value is two, which is also the minimum allowable. Although the linkule will run more efficiently for a greater n, a compromise must be made between efficiency and the extra space used in Named Storage since each buffer uses space equal to the BLKSIZE of the dataset(see below). For most applications, two buffers should be sufficient.

The MAXSIZE parameter of GSTART allows the user to set an upper limit on Named Storage space occupied by GSPEAK during execution. The larger the size, the more efficient the linkule will operate. The default size is sixteen. For each extra buffer that is requested to be kept in Named Storage, space equivalent to the block size of the Generic Storage dataset must be added to allow proper operating room for GSPEAK.

The RECOVERY parameter specifies the number of kilobytes of the generic dataset which will be reserved for recovery from errors which involve the dataset running out of room. Without this area, such an error could lead to an unusable dataset. For maximum protection, the RECOVERY size should be as large as MAXSIZE.

A defined object in Speakeasy is put in the Generic Storage dataset by the command SAVE. In using the command, it is necessary to describe the catalog entries that will be used to reference the information at a later time and to specify the name of the object to be saved. In particular

```
SAVE(X:A,I,J)
```

indicates that the object named X is to be copied into the Generic Storage dataset and to be saved for later reference under the generic name A, with case number I and item number J. Executing a SAVE command does not destroy the object; it still exists in Speakeasy's Named Storage. However, a permanant copy is also available for later use.

The command SAVED is used to reference a previously saved object. Once again, the generic name, case number, and item number are used to select a particular object. In Speakeasy the word SAVED produces a temporarily defined object that can itself be used in expressions. Thus,

```
Y=4*SQRT(SAVED(A,I,J))
```

will take the square root of the object saved as (A,I,J), multiply the result by four, and assign the name Y to the final result. To merely recover an object, a statement such as

```
X=SAVED(A,I,J)
```

will suffice.

The SAVE and SAVED commands can be used in abbreviated forms if desired. For example, the statement

```
SAVE(Y)
```

is equivalent to

```
SAVE(Y:Y,0,0)
```

and copies the currently defined object Y into the database under the generic name Y. Table 1 provides a complete list of the available SAVE and SAVED commands.

The command

```
GEND
```

ends a session. If the keyword SAVE is specified, all of the data stored in the Generic Storage dataset during that session is retained, while the keyword NOSAVE will cause the

TABLE 1. Complete list of the available SAVE and SAVED commands usable in GSPEAK

Command	Generic Name	Case Number	Item Number	Speakeasy Object
SAVE(X:A,I,J)	A	I	J	X
SAVE(X:A,I)	A	I	0	X
SAVE(X:A)	A	0	0	X
SAVE(X)	X	0	0	X
SAVED(A,I,J)	A	I	J	-
SAVED(A,I)	A	I	0	-
SAVED(A)	A	0	0	-

data to be discarded. The use of the command without a keyword is the equivalent of a GEND(SAVE) if the SAVE command has been used at least once during the session; otherwise, the equivalent of a GEND(NOSAVE) is done.

The command

```
GRESTORE(name:GENERATION=n)
```

is used to resume generic activity after a GEND command has been used. Both parameters are optional, with 'name' giving the dataset name(see GSTART), and 'GENERATION=n' specifying which generation of the dataset to restore. Each GEND command which results in the saving of data adds a successive generation to a generation stack. Each such generation acts like a 'checkpoint', allowing the user to restore the exact environment in Generic Storage after any of his previous saving GENDs. The number 'n' is the n'th generation of the dataset, so that

```
GRESTORE(:GENERATION=1)
```

would restore the first previous generation, while

```
GRESTORE(:GENERATION=MOSTRECENT)
```

is equivalent to GRESTORE and restores the present generation (i.e. the information saved by the most recent GEND command). For most cases, a simple GRESTORE will suffice.

The generation stack can be displayed with the command

```
GHISTORY(name)
```

which will print out the generation number, date and time for each generation of Generic Storage. The parameter 'name' is optional. If GSPEAK is active, the parameter is ignored and the information is displayed from the currently active dataset, while if GSPEAK is not active, the generations of USERID.name.GDATA will be printed. Omitting the parameter in the latter case implies that the dataset name is USERID.GSPEAK.GDATA.

Two commands are provided to examine the catalog of stored information: GNAMES and GINVENTORY. The command

```
GNAMES
```

will produce a list of currently defined generic names. It can also be used in the form

```
N=GNAMES
```

which will assign the corresponding namelist to N. The GINVENTORY command gives a variety of more detailed information depending on which form is used. A simple

```
GINVENTORY
```

behaves exactly like GNAMES, printing out the currently defined generic names, while

```
GINVENTORY(ALL)
```

displays all of the generic names along with their associated case, item pairs. To obtain case, item pairs for specific generic names,

```
GINVENTORY(gname1,gname2...)
```

is used. If case numbers are all that is needed, then

```
GINVENTORY(gname1,gname2...:CASES)
```

will print out the desired information. Item numbers associated with specific cases can be displayed by

```
GINVENTORY(gname1,gname2...:CASES case1,case2...).
```

It should also be noted that these commands can appear on the right side of an equal sign if only one generic name is used in the argument list. In these cases, an appropriate Speakeasy object will be defined.

The removal of unwanted items from Generic Storage is accomplished through the GDELETE command. If the user specifies

```
GDELETE(gnme)
```

the name and all of its associated case, item pairs are erased. To remove only a single case, item pair, the command

```
GDELETE(gnme,case,item)
```

is used, where case and item are the indices of the element to be deleted.

The command

```
GSIZE(MAXSIZE=n)
```

allows respecification of the MAXSIZE parameter that is set in GSTART. As in that command, the number 'n' is the number of kilobytes within which GSPEAK will restrict itself during execution.

Eventually, the generic dataset will become full of unwanted and inaccessable data. This is especially true since actual removal of the information is not possible with GSPEAK's sequential datasets. Since this represents wasted space, a method for garbage collection has been provided. If the user types

```
GCOMPRESS(readname,writename:GENERATION=n)
```

information currently in Generic Storage dataset USERID.readname.GDATA will be copied into a new dataset, USERID.writename.GDATA. Inaccessible data will not be copied.

If no parameters in the GCOMPRESS command appear before the colon, the name of the read dataset(i.e. the dataset to be compressed) is assumed to be USERID.GSPEAK.GDATA, and that of the write dataset USERID.COMPRESS.GDATA. If only one parameter is used, it is interpreted as the read dataset name, in which case the write dataset name is assumed to be as above. As with GSTART, TSO Speakeasy will create this write dataset if it does not exist, while Batch Speakeasy requires that it exist and be allocated with ddname 'writename'. In any case, the two datasets must have the same specifications; in particular, the BLKSIZEs must be identical.

The third parameter of GCOMPRESS, 'GENERATION=n', allows the specification of the particular generation to be compressed. It is also optional, with the default being the most recent generation.

The GCOMPRESS command can be invoked anytime during a Speakeasy session so that Generic Storage need not have been activated by a GSTART or GRESTORE command. However, if GSPEAK is active, the equivalent of a GEND(SAVE) command will be done to prepare for the compress.

D. Dataset Structure and Programming Considerations

The facilities described here are designed to operate as normal linkules with the Mu+ version of the Speakeasy processor. Therefore, in order to use GSPEAK, no special preparations are necessary for Speakeasy itself. However, to be able to read and write into Generic Storage from GSPEAK, it is necessary to have a dataset available.

The dataset used for Generic Storage is a normal sequential dataset. An appropriate dataset can either be defined by the user, or left up to GSPEAK to create when the GSTART linkule is invoked. Since allowing GSPEAK to create the dataset results in default specifications being used, it is often desireable to create the dataset before activating Generic Storage. This is done prior to entering Speakeasy or within Speakeasy using the TSO submode.

To allocate a new Generic Storage dataset the appropriate TSO commands are as follows:

```
   ATTRIB DCB LRECL(6000) BLKSIZE(6000) RECFM(F)
   ALLOC DA(USERID.name.GDATA) USING(DCB) SPACE(10,5)
      FILE(name) BLOCK(7000)
```

The space allocation of ten tracks for the primary allocation and five tracks for the secondary extent is sufficient for applications where less than ten thousand data elements are saved. Data sets that are filled close to capacity or for which a "full dataset" error has been received can be stripped of their inaccessable data and copied into a new dataset by use of the GCOMPRESS command. If a "full dataset" error has not yet been received and more than one generation is needed, the entire dataset should be copied into a larger one via the TSO-Copy command. The BLKSIZE may be other than 6000, but the RECFM must remain F.

To use a previously created dataset in GSPEAK, one need only specify its name in the GSTART or GRESTORE command. If the dataset has not been allocated to the proper file, the linkule will do so automatically. Alternatively, the user can do the allocation manually in TSO. This method has the advantage of being slightly more efficient. To accomplish this allocation, the command

```
         ALLOC DA(USERID.name.GDATA) FILE(name)
```

is used. The dataset name must still appear in the GSTART or GRESTORE argument list, however.

Three GSPEAK keywords refer directly to opening and closing of the Generic Storage dataset. They are the GSTART command, used to open the dataset; the GRESTORE command, used to reopen a dataset to reference previously stored data or to add new data; and the GEND command, used to close the dataset so that new data added is kept as a permanent part of the dataset. No reference can be made to Generic Storage until either GSTART or GRESTORE has been used to open the file and initialize certain internal elements of the GSPEAK linkule. In addition, because of the way in which the linkule is constructed, no information is saved permanently until the GEND command is used with the SAVE keyword. However, the SAVE keyword is not intended to be used indiscriminately since in the process of closing the dataset, at least one block of data is added to the dataset. This action occurs even if no information had been saved during the session. Therefore, in order to prevent the needless waste of the dataset, the user who simply wants to terminate the GSPEAK session with no retention of data should use the GEND command with the NOSAVE keyword.

Section 2 -- RSPEAK

RSPEAK is an attempt to provide a simple relational database facility for Speakeasy. Commands are provided to create, manipulate, and update relations as well as commands to display database information to the user. The basic syntax of the various RSPEAK commands is described below, while more general information on the relational approach to database management, as well as examples of RSPEAK in use can be found in the longer document, "Database Manipulation in Speakeasy" by R. Schlichting.

1. Creating a Relation

The RSPEAK command

```
CREATERELATION relation-name WITH n COLS
```

provides for the initial creation of a relation. Before this command is executed, however, preparations must be made to define the relation's data in such a way that appropriate input can be provided for the linkule.

These preparations consist of defining both descriptive information about the relation and the data itself. The attribute, or column headings, are defined in a Speakeasy name-literal array. One way of doing this is by using the NAMELIST function in the following manner:

```
attribs=NAMELIST(attrib1,attrib2,...,attribN)
```

There should be as many attribute names as there are columns in the relation. As with all Speakeasy name-literal objects, a maximum of eight characters is permitted; longer names will be truncated.

The attribute types are also defined in a name-literal array, with the keywords NUM and CHAR indicating the type of each column. Thus, the command

```
types=NAMELIST(NUM,CHAR,NUM)
```

would indicate that the first and third columns are numeric, while the second contains character literals. These are the only two types presently supported.

After the descriptive information has been defined, the data itself needs to be put in Named Storage. Each column should be defined as either an array of numbers or a two dimensional character object. The latter type is created by typing

```
EDIT attr-name NEW NONUM
```

which will cause the Speakeasy editor to be invoked. The data elements are then typed, one to a line, while in EDIT

INPUT MODE. After all of the elements have been entered, a null line puts the user in EDIT COMMAND MODE. In this mode, any of the normal Speakeasy editor commands can be used to alter the information as desired. When the elements, which can be a maximum of eighty characters, are fully defined the user types

```
DEFINE attr-name NONUM
```

to have the object entered into Named Storage under the appropriate name. An END command returns the user to MANUAL MODE.

When the information has been properly defined, the CREATERELATION command can be invoked as shown above. The command will then request the name under which the attributes have been defined, and, following that, the namelist which holds the type definitions. The attribute names of the primary keys are then requested. These should be entered with separating commas. Up to four columns can be designated as the primary key.

The user is then asked if all of the columns are stored under their attribute names. If they are, RSPEAK will automatically retrieve the information and store it. Otherwise, the user will have to type in the name under which each column is stored as RSPEAK asks for it. The former method is obviously preferable. After all the data has been input to the linkule, a message will be transmitted to the user indicating that the definition of the relation has been accomplished.

While the process for creating a relation may seem overly complicated, each step is necessary for the proper definition of the relation. There are, however, certain facets of Speakeasy which allow the circumvention of some of the most arduous preparations. For example, all of the required definitions can be performed in a previous Speakeasy session and then stored until the relation is actually created. The KEEPLIST function, which allows the storing of several objects under a single name, is especially useful. Another possibilty is the generation of the appropriate objects from a Fortran program which could then store the objects in a private KEEP library for later retrieval in an interactive Speakeasy session. This alternative means that the data in the relation need not be laboriously entered from a terminal. This is especially nice if the relation contains a large amount of information. All of these devices facilitate the use of the CREATERELATION command.

2. Manipulating Relations

There are two basic operations which can be performed on relations in RSPEAK, combining and projecting.

To combine two relations, the COMBRELATION command is used in the following manner:

```
    COMBRELATION rel1 WITH rel2 OVER common-domain [INTO
                           resultnam]
```

This command merges two relations by first scanning for like elements in the common domain, and then upon finding them, combining those two rows into one. If there are elements in the common domain of one relation but not the other, that row does not participate in the combine operation. Conversely, if an element has many like elements in the other relation, its row is replicated to participate in the combine. The optional keyword INTO allows the specification of a name for the resulting relation. If this keyword is omitted, the name RESULT will automatically be assigned to the answer, allowing it to be used in subsequent commands. The COMBRELATION command is illustrated several times in the sample TSO sessions in the appendix.

The PROJRELATION command isolates a specific subset of columns from the relation. By typing

```
PROJRELATION relnam OVER col1,col2,...colN [INTO resultnam]
```

a result is formed in which only those columns specified will appear. It should also be pointed out that they will be defined in the order given in the command, so that it also can be used to rearrange the columns of a relation. As with the COMBRELATION command, the keyword INTO allows specification of a name for the resulting relation.

Many queries into relational databases involve the repeated use the the COMBRELATION and PROJRELATION commands before a final result is established. When this is the case, the intermediate relations generated by the commands tend to be transitory, existing only for a short time until another operation can be performed on it. In order to save both time and space, special capabilities have been provided in RSPEAK to facilitate these multiple-command queries. They make use of what is called a result relation.

If the user appends the keyword RESULT to the end of either the COMBRELATION or PROJRELATION commands, a result relation is formed instead of a regular relation. This result relaton, which is faster to generate and uses less space than a regular relation, can then by manipulated by special versions of the COMBRELATION and PROJRELATION commands.

The COMBRESULT command allows the last previously generated result relation to be combined with any regular relation in the same way as the COMBRELATION command. The proper format is

```
        COMBRESULT WITH relnam OVER common-domain
```

where 'common-domain' is the attribute heading.

The command

```
PROJRESULT OVER col1,col2,...,colN
```

projects the last generated result relation over the specified columns.

These two commands form, as a consequence of their execution, another result relation which replaces any previously generated result relation. Therefore, since only the last generated result is accessable at any given time, these special relations are truly transitory. One further property should be noted: since the result relations are nameless, the INTO keyword (used with the COMBRELATION and PROJRELATION commands) is incompatable with the use of these special types of relations. Thus, INTO should not be used in conjunction with the RESULT keyword, or on the COMBRESULT or PROJRESULT commands.

The user will generally want to retain the result of a long series of commands. A method of converting a result relation to a regular relation has been provided. The command

```
KEEPRESULT AS relnam
```

will copy the last generated result into the relational database as a regular relation under the name 'relnam'. The result relation still exists, however, and can be used in further computations.

3. Updating Commands

In addition to the manipulative commands, two operations are provided for the alteration of regular relations. To insert a new row, the command

```
ROWADD relnam TO el1,el2,...elN
```

is used. In this case, N is the number of columns in the relation. The elements which constitute the new row may be constants, either numeric or character, or the names of objects defined in Named Storage. The row is added to the end of the specified relation, defining a new relation which replaces the old. It should be noted that this relation is not treated as a result relation, but simply as a modified version of the original relation.

To delete a row, the command

```
ROWDELETE relname OF el1,el2,...elK
```

is used. The number of elements specified in the command should be equal to the number of attributes which serve as the primary key. The row with that key is then searched out and deleted from the relation. As with ROWADD, the new relation replaces the old.

4. Informational Commands

There are three words which can be used to print out relations. The command

DISPRELATION relnam

will cause the specified relation to be printed out. To display a result, the command

DISPRESULT

is used. The names of all currently defined relations can be obtained by the command

RELATIONS.

5. Miscellaneous commands

The user might find it advantageous to be able to use normal Speakeasy commands on certain columns of a relation. To facilitate this type of operation, two commands can be used. The word

USERELATION relnam

will define each column of the specified relation in Named Storage under its column heading (i.e. attribute name). Numeric columns will be defined as one dimensional arrays, while character information will become two dimensional character arrays. The command

USERESULT

performs the same operation on the last generated result relation.

The command

REND

allows the user to end the RSPEAK session in much the same way as the GEND command of GSPEAK. If the keyword SAVE is specified, all of the relations created during that session are retained, while the keyword NOSAVE will cause the relations to be discarded. The use of the command without a keyword is the equivalent of an REND SAVE if at least one relation has been saved during the session; otherwise the equivalent of a REND NOSAVE is done. However, any result relations which were generated are valid only during the session in which they were defined and are never incorporated into the relational database.

Several of the commands which were used in GSPEAK are also necessary for the operation of RSPEAK. These are

GSTART, GRESTORE, and GSIZE. For a description of each, see the previous section.

Appendix I

GLOSSARY OF GSPEAK COMMANDS

GCOMPRESS--Writes all current information from one generation of Generic Storage into a new dataset.

GDELETE--Removes either an entire generic name and its associated case, item pairs or one case, item pair of a specific generic name from Generic Storage.

GEND--Closes the Generic Storage dataset and retains tho data stored during the session.

GHISTORY--Prints out the date, time, and generation number for each generation of Generic Storage.

GINVENTORY--Lists case and item numbers associated with a generic name.

GNAMES--Lists all the Generic Names.

GRESTORE--Opens and initializes a Generic Storage dataset which has previously been written in.

GSIZE--Allows respecification of the MAXSIZE parameter that is set in GSTART.

GSTART--Initiates a new Generic Storage dataset.

SAVE--Writes objects into the Generic Storage dataset. (See Table 1)

SAVED--Reads objects from the Generic Storage dataset. (See Table 1)

Appendix II

GLOSSARY OF RSPEAK COMMANDS

COMBRELATION--Forms a relation by combining two other relations.

COMBRESULT--Forms a result relation by combining a regular relation with the latest result relation.

CREATERELATION--Creates a relation.

DISPRELATION--Displays a relation.

DISPRESULT--Displays the latest result relation.

KEEPRESULT--Creates a regular relation from the latest result relation.

PROJRELATION--Forms a new relation by projecting a relation over specified columns.

PROJRESULT--Forms a new result relation by projecting the latest result relation over specified columns.

RELATIONS--Prints out the names of the currently defined relations.

REND--Terminates an RSPEAK session.

ROWADD--Adds a row to a relation.

ROWDELETE--Deletes a row from a relation.

USERELATION--Defines the columns of a relation in Named Storage under their attribute headings.

USERESULT--Defines the columns of the latest result relation in Named Storage under their attribute headings.

A Preview of some upcoming Developments in RSPEAK

S. Massaquoi August 1978

The evolution of Speakeasy's relational data base facility will involve primarily the introduction of a new set of high-powered words. Soon the relational data base vocabulary will probably consist of the words: RCOMBINE, RAPPEND, RPROJECT, RCREATE, RDELETE, RUPDATE, RSHOW, RCOUNT, RCOPY RDESCRIBE, RDESCRIPTION and RREFERENCE. The capabilities of these words will greatly extend both the power and convenience of data manipulation in RSPEAK.

The impetus for developing these new words comes from the realization that a relational data base is more efficient and powerful when it consists of many different relations, each containg a conceptually different set of data, than when it is made up of a few catch-all relations. Such poorly refined data bases tend to have a great deal of redundancy of information and suffer many types of manipulation anomalies.* With an increased number of relations, however, it becomes much more tedious to execute the basic manipulations necessary to retrieve the widely distributed data. Thus the new RSPEAK words RUPDATE, RSHOW, RCOUNT, RDELETE and RCOPY will be designed to automatically combine, append, and project all of the relations needed to isolate the data. The words RREFERENCE and RDESCRIBE will enable the user to, respectively, specify which relations in the data base are to be included in the automatic searches, and to store short descriptions of the data contained in each relation (RDESCRIPTION will recover this description). Finally, RCREATE, RAPPEND, RCOMBINE and RPROJECT will retain the user's ability to manually manipulate relations if desired.

The tentative drafts of the help documents for these words are given at the end of this handout to better explain their individual functions, but first, an example of how some of this vocabulary might be used.

Imagine for a moment that it is 10:50 A.M. and Sam, an attendant at Speakville airport, has just discovered a fat wallet lying in a corridor apparently having been just dropped by a hurried traveler. Incredibly, the only sources of possible identification are part of a torn social security card displaying the digits: 37-2011 , and a torn portion of a newspaper advertisement which reads: "...fly our wide-bodied jets to Hawaii!" Being uncompromisingly honest and quite resourceful to boot, Sam rushes to the nearest computer terminal and logs onto Speakeasy. Once invoking RSPEAK he has access to FLIGHTINFO, the data base consisting of the following relations:

* For a discussion of optimal relational data base organization and manipulation anomalies, see"Relational Data-Base Management Systems" by Donald D. Chamberlin. ACM Computing Surveys: Vol. 8, #1 March 1976

DEPARTURES					
TIME	FLIGHTNO	AIRLINE	DESTINATION	PLANE	GATE
10:55	202	ALPHA	NEW YORK	727	K1
10:55	139	GAMMA.	HONOLULU	747	C3
11:00	617	GAMMA	BOSTON	DC-10	C1
11:03	4	BETA	NEW YORK	727	J6
11:11	165	GAMMA	PHOENIX	DC-10	A3
11:18	105	ZETA	HONOLULU	707	E1
11:25	218	ZETA	SAN FRANCISCO	747	E6
11:41	201	PHI	ATLANTA	707	H11

ARRIVALS					
TIME	FLIGHTNO	AIRLINE	FROM	PLANE	GATE
10:30	100	ZETA	DALLAS	727	E4
10:38	617	GAMMA	SEATTLE	DC-10	C1
10:45	302	RHO	HICKSVILLE	DC-9	D2
10:53	413	ALPHA	WASHINGTONDC	727	K4
11:02	108	ALPHA	PHILADELPHIA	707	K5
11:11	338	ZETA	LOS ANGELES	DC-10	E5
11:18	119	PHI	MIAMI	707.	H2
11:21	106	BETA	NEW YORK	L-1011	J8
11:25	115	BETA	ALBANY	727	J6

PASSENGERS

NAME	FLIGHTNO	AIRLINE	ADDRESS	S.S.NO
Aaron, L	413	ALPHA	603 N Woodale, N.Y.N.Y.	488-21-4966
Abbott, K	165	GAMMA	13 East Drive, Pit Pa	738-28-7277
Abbate, J	105	ZETA	18½ 43rd, Zv. Id	100-14-8139
Ackerson, Z	401	OMEGA	P.O. Box 21, Woodland Ky	376-45-5104
Addison, M	202	ALPHA	173 W 95th St, Chi Il	218-19-1817
Ardmore, P	139	GAMMA	506 W. Highbrook, Min Min	365-37-2011
Aston, J	139	GAMMA	218 E. Lombard, Ren. Nev	189-27-3604
Astor, V	118	OMEGA	17½ 109th #6, LA Ca.	138-81-8712
Atticus, G	201	PHI	100 Longwood Dive, N.Y.	151-11-0664
Avery, C	106	BETA	37 N. Kedzie, Chi. Ill.	642-01-0080
Baker, C	201	PHI	3109 E 47th St, Ch. Ill.	137-12-3092
Bamberg, D	302	RHO	Ontario NW. Wash. D.C.	418-22-2867
Banson, L	119	PHI	13 East Start, Miami Fla	910-05-5631
Berry, V	302	RHO	Calhoun Sq #8 Boston Ma	218-89-9037
Best, G	302	RHO	Roberts Fld, No. Dak	668-84-0935
Bettler, G	413	ALPHA	601 E Main, Portage, Ind.	471-62-6801
Binks, J	100	ZETA	14 High St. Elgin Ill.	928-35-2019
Bisk, F	4	BETA	#3 Township Ave. Wr. Va	375-63-8076
Ritner, L	139	GAMMA	1922 Cassidy, NY NY	225-24-8172
B			1059 W. 211 Chi, Ill.	606-0

PLANETYPE

PLANE	NO SEATS	BODYTYPE	NOENGINES
DC-9	80	NARROW	2
DC-10	198	WIDEBODY	3
727	78	NARROW	3
707	130	NARROW	4
747	240	WIDEBODY	4
L-1011	200	WIDEBODY	3

He then enters:

```
:_RDESCRIPTION FLIGHTINFO

     DEPARTURES IS A RELATION HAVING 6 ATTRIBUTE HEADINGS:
TIME, FLIGHTNO, AIRLINE, DESTINATION, PLANE, GATE
THERE ARE CURRENTLY 8 RECORDS UNDER THESE HEADINGS.
     DEPARTURES  IS A CONSTANTLY UPDATED RELATION WHICH CONTAINS INFORMA-
TION REGARDING FLIGHTS DEPARTING WITHIN THE NEXT HOUR.

     ARRIVALS IS A RELATION HAVING 6 ATTRIBUTE HEADINGS:
TIME, FLIGHTNO, AIRLINE, DESTINATION, PLANE, GATE
THERE ARE CURRENTLY 9 RECORDS UNDER THESE HEADINGS.
     ARRIVALS IS A CONSTANTLY UPDATED RELATION  WHICH CONTAINS INFORMATION
REGARDING FLIGHTS WHICH HAVE ARRIVED DURING THE PAST HALF-HOUR OR ARE
SCHEDULED TO ARRIVE IN THE NEXT HALF-HOUR.

     PASSENGERS IS A RELATION HAVING 5 ATTRIBUTE HEADINGS:
NAME, FLIGHTNO, AIRLINE, ADDRESS, S.S.NO
THERE ARE CURRENTLY 1,314 RECORDS UNDER THESE HEADINGS.
     PASSENGERS CONTIANS INFORMATION ABOUT ALL OF THE PERSONS WHO CURRENTLY
HAVE RESERVATIONS ON FLIGHTS ARRIVING AT OR DEPARTING FROM SPEAKVILLE AIRPORT.

     PLANETYPE IS A RELATION HAVING 4 ATTRIBUTE HEADINGS:
PLANE, NOSEATS, BODYTYPE, NOENGINES
THERE ARE CURRENTLY 6 RECORDS UNDER THESE HEADINGS.
     PLANETYPE CONTAINS PHYSICAL SPECIFICATIONS OF THE TYPES OF AIRCRAFT
CURRENTLY USED BY AIRLINES OPERATING AT SPEAKVILLE AIRPORT.


$   Note that FLIGHTINFO is merely a namelist of the relations in the
$   data base.
$      He continues:

:_RREFERENCE DEPARTURES, PLANETYPE
:_RSHOW "TIME,FLIGHTNO,AIRLINE,GATE FROM DEPARTURES WHERE DESTINATION IS &
 'HONOLULU' AND BODYTYPE IS 'WIDEBODY' IN PLANETYPE"

   TIME    FLIGHTNO   AIRLINE  GATE
   ******* ********** ******** ********
   10:55   139        GAMMA    C3
```

So Sam rushes off to C3 only to find that flight 139 had left seconds earlier. Not to despair, though, he logs back onto the system and enters:

```
:_RREFERENCE PASSENGERS, DEPARTURES
:_RSHOW "NAME, ADDRESS, S.S.NO FROM PASSENGERS WHERE DESTINATION IS &
 'HONOLULU' IN DEPARTURES"

    NAME          ADDRESS                         S.S.NO
    ************* ****************************** ***********
    Abbate,L      13 East Drive, Pitt. Pa.        738-28-7277
    Ardmore,P     506 W. Highbrook, Minn. Minn.   365-37-2011
    Aston,J       218 E. Lombard, Reno Nev.       189-27-3604
    Bitner,L      1922 Cassidy, N.Y., N.Y.        225-24-8172
        .                    .                          .
        .                    .                          .
        .                    .                          .
```

And voila! the last digits of Mr. Ardmore's social security number match those on the stub--all is not lost! Thanks to Sam and RSPEAK the wallet will be returned!

Finally, the important thing to note is that the real drama of this example lies in the fact that the last simple query alone replaces essentially the following sequence of current commands:

```
:_ATTRIBS=NAMELIST(DESTINATION)
:_TYPES=NAMELIST(CHR)
:_EDIT DESTINATION NEW NONUM
 1.   HONOLULU
:%DEFINE DESTINATION_NONUM
:%END
:_CREATERELATION TEMP1 WITH 1 COL
:_COMBRELATION DEPARTURES WITH TEMP1 OVER DESTINATION INTO TEMP2
:_PROJRELATION TEMP2 OVER FLIGHTNO, AIRLINE INTO TEMP3
:_COMBRELATION TEMP3 WITH PASSENGERS OVER FLIGHTNO RESULT
:_COMBRESULT WITH TEMP3 OVER AIRLINE
:_PROJRESULT OVER NAME, ADDRESS, S.S.NO
:_DISPRESULT
```

Tentative RSPEAK Help Documents

RAPPEND RAPPEND joins relations with the same attribute headings.

RAPPEND(rel1, rel2, ... reln INTO resultname) produces a single relation named 'resultname' which consists of all of the records contained in relations rel1 ... reln (with duplicate records removed). Rel1 ... reln must therefore all have the same attribute headings.

If the keyword INTO and the resultname are omitted, the resulting relation is by default named "RESULT".

RAPPEND is one of several words used with Speakeasy's relational data base facility. See the RSPEAK larger document for more information.

RCOMBINE *

RCOPY RCOPY copies data from a relation into another object.

The form of the command is: RCOPY "qualifier" where the qualifier is always enclosed by a single pair of quotation marks. Within the qualifier, three keywords may be used: FROM, WHERE (or WHEREVER) and AS.

FROM specifies the name of the relation from which the data is to be retrieved.

WHERE introduces a logical condition which must be satisfied by the data search. The condition consists of units of the form:
conditionattribute verb conditionvalue IN relationname
connected together by the logical conjunctions AND, OR or EXCEPT (or UNLESS); optionally, AND WHERE, OR WHERE or EXCEPT WHERE may be used. Further, up to 8 levels of parentheses may be used to logically group the expression. When parentheses are not used the default precedence of the logical conjunctions is: AND before OR before EXCEPT. The verb may be: IS, ISNOT, EQUALS, .EQ., .GE., .LE., .GT., .LT., .NE., ISBEFORE, ISAFTER, ISBETWEEN...AND.., CONTAINS. The conditionvalue may be a list of data items enclosed in apostrophes, a name of an object containing the list of data items (each item in the list is considered a conditionvalue alternative), or an expression bracketed in angle brackets (or at-signs) which will be evaluated by the Speakeasy processor before being used as a value. The specification of a particular relation to be searched by the keyword IN is optional but speeds location of data.

AS specifies the names which the resulting relations will be given. AS ARRAYS may be used to indicate that the results are to be in the form of 1-dimensional arrays with the given names. If the names are omitted, the attribute headings will be used as names by default.

The portion of the qualifier preceeding any keywords indicates what part of the data is to be copied. This may be a list of attribute headings, "RECORDS" if data under all headings is to be copied or a list of relationnames if the entire relations are to be copied (in this case no other keywords follow). A copy of just the attribute headings of a given relation can be obtained by using the syntax:

```
        RCOPY ATTRIBUTES FROM relationname
```

Finally if the syntax   RCOPY :   is used the user will be automatically prompted with the possible keywords. A blank line response causes an advance to the next keyword, a break causes the prompt to be reissued and two consecutive breaks cause termination of the command with nothing done.

RCOPY is one of several words used with Speakeasy's relational data base facility. For more information and examples of the RCOPY syntax, see the RSPEAK larger document.

RCOUNT

RCOUNT counts selected data items in a relation. The form of the command is: RCOUNT "qualifier"   where the qualifier is identical to that used with RSHOW (see separate help document) with the single exception that RCOUNT may be used in a replacement expression:

```
        NUM = RCOUNT "qualifier"
```

which assigns the count obtained to the variable NUM.

RCOUNT is one of several words used with Speakeasy's relational data base facility. See the RSPEAK larger document for more information.

RCREATE

*

RDELETE

RDELETE deletes selected records from a relation. The form of the command is: RDELETE "qualifier"   where the qualifier is english-like and is always enclosed by a single pair of quotation marks. The two syntaxes of the command are:

RDELETE relation1,.... relationN   which destroys relatons 1 thru N.

RDELETE FROM relation WHERE condition   which caused all records which contain data items satisfying the condition to be deleted from relation. The condition consists of units of the form:

conditionattribute verb conditionvalue IN searchrelation

connected together by the logical conjunctions AND, OR or EXCEPT (or UNLESS); optionally, AND WHERE, OR WHERE or EXCEPT WHERE may be used. Further, up to 8 levels of parentheses may be used to logically group the expression. The verb may be: IS, ISNOT, EQUALS, .EQ., .GE., .LE.,

.GT., .LT., .NE., ISBEFORE, ISAFTER, ISBETWEEN...AND..., CONTAINS. The conditionvalue may be a list of data items enclosed in apostrophes, a name of an object containing the list of data items (each item in the list is considered a conditionvalue alternative), or an expression bracketed in angle brackets (or at-signs) which will be evaluated by the Speakeasy processor before being used as a value. The specification of a relation to be searched first by the keyword IN is optional but speeds location of data.

RDELETE is one of several words used with Speakeasy's relational data base facility. For more information see the RSPEAK larger document.

RDESCRIB RDESCRIBE enters a description of a relation.

RDESCRIBE(relationname WITH description) is used to enter a description of the data containted in the relation named "relationname". "description" is the character object which contains the description. This object may be created with a simple replacement statement using quotation marks or with the Speakeasy word TEXT (see help document for TEXT).

Alternatively, the syntax RDESCRIBE(relationname :) may be used to cause the user to be prompted with ":@" to enter lines of the description directly.

RDESCRIBE is one of several words used with Speakeasy's relational data base facility (see larger document: "Relational Data Bases in Speakeasy" for more informaton).

RDESCRIP RDESCRIPTION(relation) lists the description of relation.

The description which the user has stored using the word RDESCRIBE (see separate help document) is printed out preceeded by the message:

```
   "'relation' is a relation having N attribute headings:
     atthead1,atthead2,...attheadn
     There are currently k records in 'relation'."
```

In order that k, the number of records, be always accurate, the number is automatically revised when when changes are made to 'relation'. Also, the entire description is automatically deleted whenever 'relation' is destroyed.

The description can be saved as a character object X, rather than be displayed, via the command:

```
      X=RDESCRIPTION( relation ).
```

RDESCRIPTION is one of several words used with Speakeasy's relational data base facility. See the RSPEAK larger document for more information.

RPROJECT *

RREFEREN RREFERENCE specifies relations for automatic searches.

RREFERENCE rel1,...relN specifies which relations of those currently defined are to be searched automatically by other RSPEAK words which use the keyword WHERE (presently these include: RSHOW, RCOUNT, RCOPY, RDELETE, and RUPDATE). With these words the keyword FROM or IN may be omitted whenever exactly one relation has been referenced.

RREFERENCE ALL references all currently defined relations.

Relations can be freed from being referenced by using the commands RFREE rel1,...relN or RFREE ALL

RREFERENCE is one of several words used with Speakeasy's relational data base facility. For more information, see the RSPEAK larger document.

RSHOW

RSHOW displays selected data items from a relation.

The form of the command is: RSHOW "qualifier" where the syntax of qualifier is english-like and is always enclosed in a single pair of quotation marks. Within the qualifier two keywords may be used: FROM and WHERE.

FROM specifies the name of the relation from which the data is to be retrieved. This keyword may be omitted if there is exactly one relation currently referenced.

WHERE introduces a logical condition which must be satisfied by the data search. The condition consists of units of the form:

conditionattribute verb conditionvalue IN relationname

connected together by the logical conjunctions: AND, OR or EXCEPT (or UNLESS); optionally, AND WHERE, OR WHERE or EXCEPT WHERE may be used. Further, up to 8 levels of parentheses may be used to logically group the expression. The verb may be: IS, ISNOT, EQUALS, .EQ., .GE., .LE., .GT., .LT., .NE., ISBEFORE, ISAFTER, ISBETWEEN...AND..., CONTAINS. The conditionvalue may be a list of data items enclosed in apostrophes, a name of an object containing the list of data items (each item in the list is considered a conditionvalue alternative), or an expression bracketed in angle brackets (or at-signs) which will be evaluated by the Speakeasy processor before being used as a value. The specification of a searchrelation with IN is optional but speeds the location of data.

The portion of the qualifier preceeding any keywords indicates what part of the data is to be displayed. This may be a list of attribute headings, "RECORDS" if data under all headings is to be displayed or a list of relationnames if the entire relations are to be shown (in this case no other keywords follow). A listing of just the attribute headings of a given relation can be obtained by using the syntax:

RSHOW ATTRIBUTES FROM relationname

Finally, if the syntax RSHOW : is used, the user will be automatically prompted with the possible keywords. word, a break causes the prompt to be reissued and two consecutive breaks cause termination of the command with nothing done.

RSHOW is one of several words used with Speakeasy's relational data base facility. For more information and examples of the RSHOW syntax, see the RSPEAK larger document

RUPDATE RUPDATE replaces selected data items in a relation.

The form of the command is: RUPDATE "qualifier" where the qualifier is english like and is always enclosed in a single pair of quotation marks. The two basic syntaxes of the command are:

RUPDATE relationname : which causes new records to be added to the relati on named relationname. This is done by having the user be prompted sequentially with the attribute headings and having the user enter the data items to be stored under these headings. A break character in the middle of an input line causes a prompt to be reissued while while a break immediately following a prompt terminates the command. Note that if the last record was not completed it will not be retained.

RUPDATE atthead1,...attheadN : which causes the user to be prompted with attribute headings 1 thru N . The data which is entered in response to each prompt is used to replace th data under each respective heading.

RUPDATE RECORDS : is the same as above except that the user is promted automatically with all of the attribute headings belonging to the relation.

In addition, 3 keywords may be used with the basic syntaxes above.

IN specifies the name of the relation in which data is to be updated. This keyword may be omitted if exactly one relation is currently referenced.

WHERE introduces a logical condition which must be satisfied by the data search. The condition consists of units of the form:

conditionattribute verb conditionvalue IN relationname

connected to gether by the logical conjunctions AND, OR or EXCEPT (or UNLESS): optionally, AND WHERE, OR WHERE or EXCEPT WHERE may be used. Further, up to 8 levels of parentheses may be used to logically group the expression. When parentheses are not used the default precedence of the logical conjunctions is: AND before OR before EXCEPT. The verb may be: IS, ISNOT, EQUALS, .EQ., .GE., .LE., .GT., .LT., .NE., ISBEFORE, ISAFTER, ISBETWEEN...AND.., CONTAINS. The conditionvalue may be a single data item enclosed in apostrophes, a name of an object containing a data item, a list of conditionvalue alternatives, an object containing a list of conditionvalue alternatives or an expression bracketed in angle brackets (or at-signs) which will be evaluated by the speakeasy processor before being used as a value. The specification of a searchrelation with IN is always optional but speeds location of data.

WITH replaces the colon and specifies directly the data or names of objects containing the data which is to replace the old data. The data is separated by semicolons into fields corresponding to the attribute headings. Each field may contain a single data item enclosed in apostrophes, the name of an object which contains the data item, or an expression in angle brackets or at-signs which will be evaluated by the Speakeasy processor before being used as a data item.

RUPDATE is one of several words used with Speakeasy's relational data base facility. For more information, see the RSPEAK larger document.

* The help documents for these words will be substanitially the same as those now existing for COMBRELATION,CREATERELATION and PROJRELATION.

APPLICATIONS OF RSPEAK: A RELATIONAL DATA MANAGEMENT FACILITY IN SPEAKEASY

by

Bill A. Bavinger
School of Architecture
Rice University

INTRODUCTION

For the past four years the Rice Architecture Computer Lab (RACL) in the School of Architecture at Rice University has used Speakeasy as its primary access to computing capabilities. Utilizing Speakeasy's provision for extension the lab has developed numerous linkules that are specifically related to geographic information processing. This subsystem of linkules is called RAGIS (Rice Architecture Geographic Information System). Speakeasy has provided the capability to integrate under a single information processing system, a number of special purpose functions and operations that jointly allow acceptance, organization, and representation of spatially related data at the regional, urban, and the built form scales. The addition last year of a relational data management facility in Speakeasy (Rspeak) provided the opportunity to integrate a data base management capability into the lab's work. In practice the relational approach to data management greatly extended the lab's research and applications. This paper will briefly describe the concepts of the relational data model and the specific use of Rspeak in four different applications: (1) a Watershed Management Project for the U.S. Army Corps of Engineers, (2) Urban Research at RACL, (3) Information Management for Health Planning and Facility Design, and (4) a management application concerning chemicals and research related to the nation's Enhanced Oil Recovery (EOR) program.

There are three general conceptual models for data management: hierarchical (tree), network (complex), and relational (tabular). Of these three models the relational seems to offer many advantages (Date, 1977). Relations are straightforward, relatively easy to understand, and their uniformity of representation allows an operational simplicity that is much more easily managed than either the network or hierarchical models. Also, since its organization does not not depend on physical analogs, or pointers, to create relations between data the relational model offers a minimum of organizational or informational bias. The network approach, from which heirarchical models can be considered a special case, has many constructs which are essential in the data set. This one fact establishes the difference between relational and network models: relations are simple because they have one uniform construct, networks are complex because they have many operational constructs, all of which are necessary for data management and manipulation.

Since its inception in the early 1970's, RACL has been concerned both with complex problems related to regional and urban analysis and with large amounts of data. Many approaches to data management related to these problems have been explored and developed

RAGIS

SYSTEM DIAGRAM

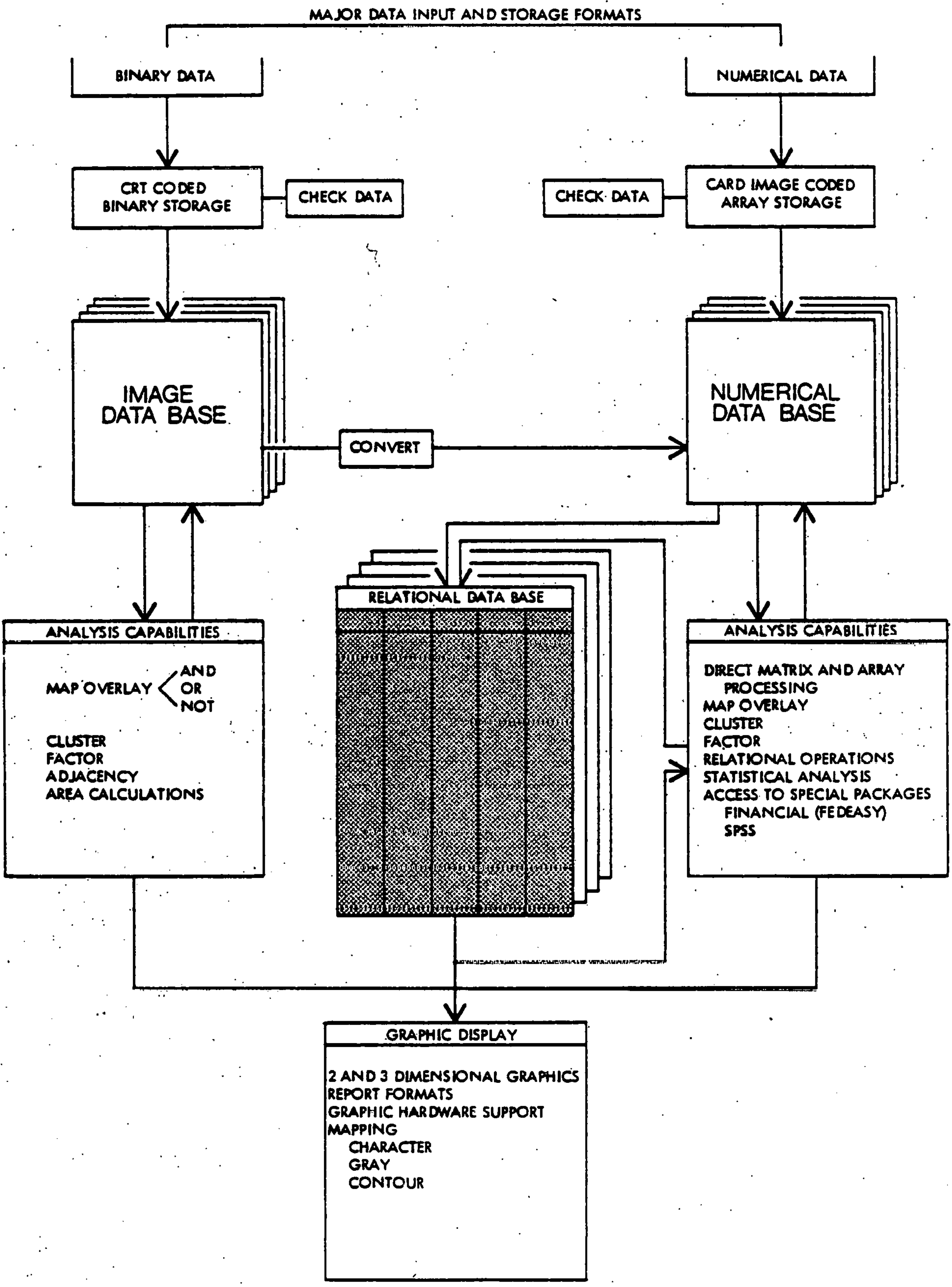


Figure 1

under RAGIS. Included in these approaches are pattern finding capabilities (multi-variate clustering and factoring), data association capabilities (the matrix approach to data organization), and capabilities to determine context (connectivity and adjacency). The introduction of Rspeak and the concepts of the relational data management have allowed the perception of the interrelationships between the results of analysis and the raw data describing the original problem context. These capabilities are implemented very simply by adding attributes (columns) containing the results of various kinds of analysis to tables of attributes in Rspeak containing data (relations). Thus in addition to finding relationships between data, it is possible to find relationships both between various kinds of analysis and data, and among the results of the analyses. The merger of Rspeak with RAGIS analysis capabilities has allowed perceptions through first order data organizations to a second order of relationships describing the first order. For the first time many of the original goals of RACL, which included the integration of diverse information into the design process at all scales, and the view of design as a series of representations in the relationships between information, are manageable.

While at the technical level the relational approach to data management has received much attention in recent articles and books, little has been written concerning actual applications. Work at RACL with Rspeak has revealed a number of operational considerations related to the use of a relational data management model and "relational thinking". In practice Rspeak provides a number of powerful data management capabilities, but more importantly, it introduces a number of concepts for problem organization and solving. An understanding of these concepts has greatly expanded the scope of RACL applications. It is difficult to specifically state these concepts as operational rules at this point in time, but the general influence of relational thinking emerges in the following descriptions of specific applications.

APPLICATIONS

Watershed Management

RAGIS has been used on several flood plain projects (Rowe and Blackburn, 1977; Bedient, 1978) primarily for the purposes of organizing the inventory and analysis of land use and environmental data, and in support of land use and hydrologic modelling activities (Figure 2). On the most recent of these projects (Bear Creek, Texas) Rspeak was explored for its utility to the overall data management requirements. The spatial data base constructed for Bear Creek conformed to the Universal Transverse Mercator (UTM) cardinal referencing system. The data points themselves took the form of a uniform gird cell system with individual grid cells of 1/4km X 1/4km or 15.45 acres. Spatial information was coded as discrete images which are stored as logical bit strings. Thus each data pattern was either present or absent in each grid cell. This raster-like technique for data capture and storage is one of the major aspects of RAGIS. The image data base incorporated land use and ground cover information, soils information by hydrologic sub-group, and other environmental and economic data such as the location of natural hazard conditions and land value. To improve the spatial resolution of the data representation

APPLICATION OF RAGIS TO FLOOD PLAIN MANAGEMENT

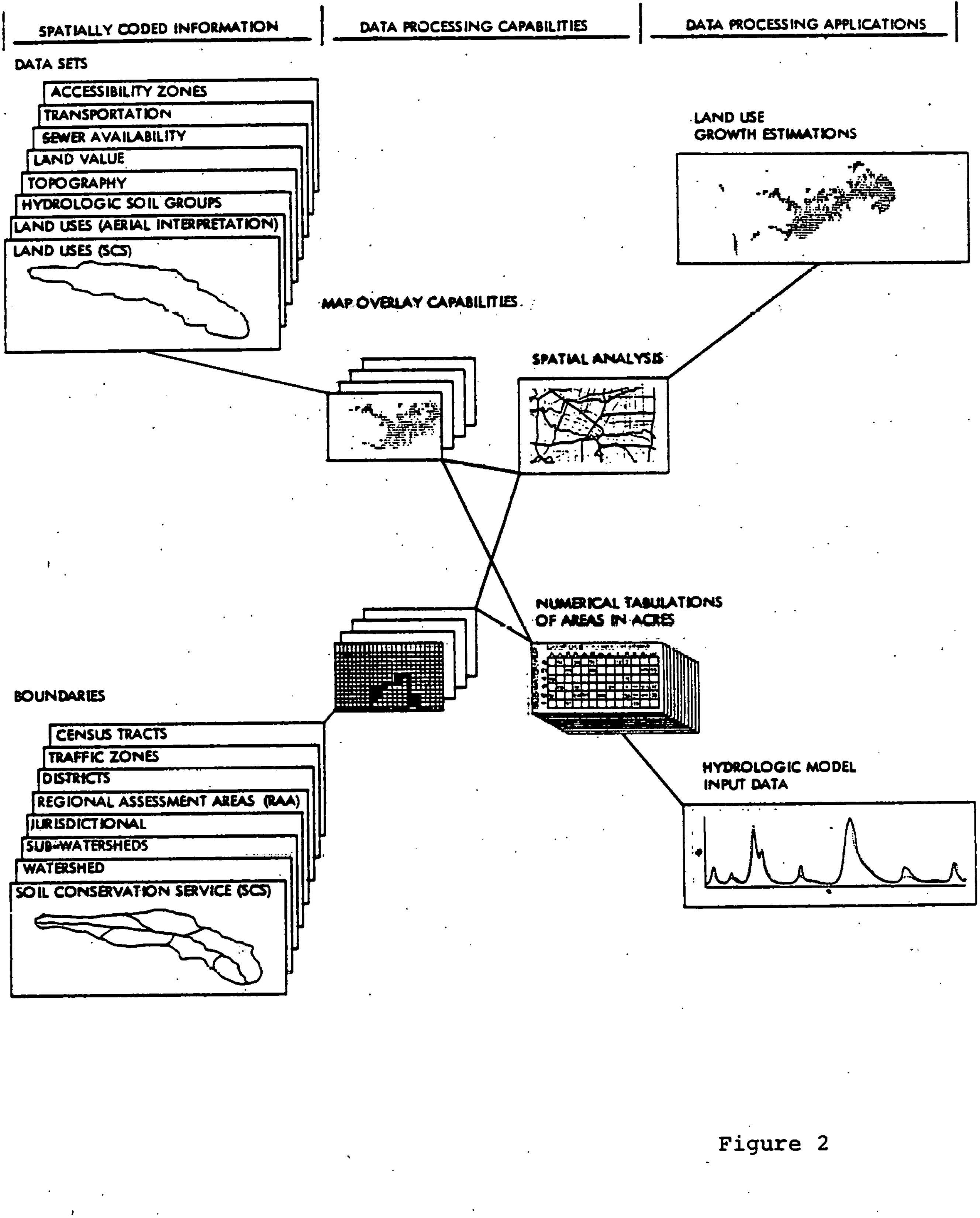
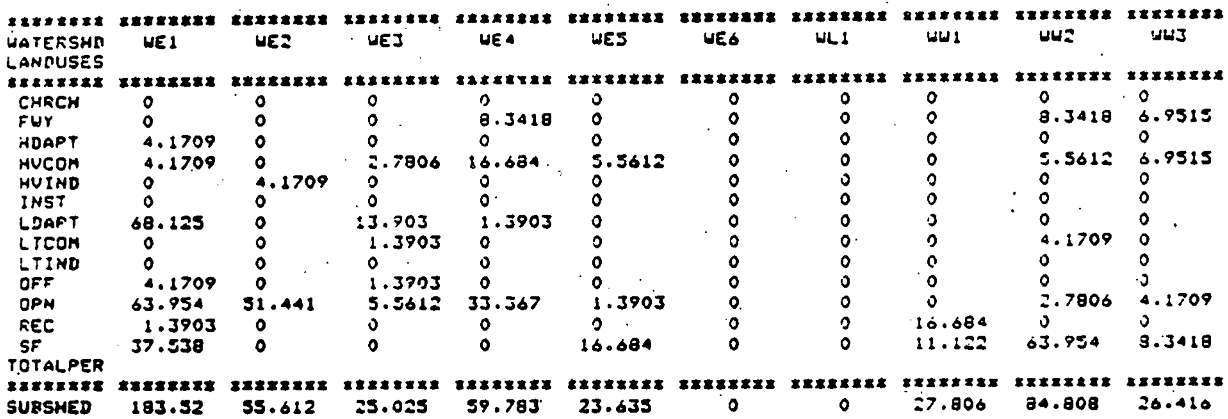


Figure 2

the capability to allow the partial presence of a data feature was incorporated.

Through Speakeasy each data pattern was converted to a numerical array of zeros (for absence) and any interger value desired (for presence). These arrays were combined under their major headings and loaded as attribute columns in Rspeak. Rspeak provided a centralized data management facility to support the numerous uses of the data required in this project. In addition, it allowed the results from one area of the project such as land use growth projections to be related to results from other areas of the project such as the determination of flooding. Thus the end purpose of the project, which was to identify the relationship between these two dynamic and competative systems at any point in time and related to economic criteria was readily accomplished in Rspeak (Figure 3). Any state, or condition, in either land use growth or flooding could be mapped directly from the relational data base in terms of the other system (Figures 4 & 5).



WATERSHD	WE1	WE2	WE3	WE4	WE5	WE6	WL1	WW1	WW2	WW3
LANDUSES										
CHRCH	0	0	0	0	0	0	0	0	0	0
FWY	0	0	0	8.3418	0	0	0	0	8.3418	6.9515
HDAPT	4.1709	0	0	0	0	0	0	0	0	0
HVCOM	4.1709	0	2.7806	16.684	5.5612	0	0	0	5.5612	6.9515
HVIND	0	4.1709	0	0	0	0	0	0	0	0
INST	0	0	0	0	0	0	0	0	0	0
LDAPT	68.125	0	13.903	1.3903	0	0	0	0	0	0
LTCOM	0	0	1.3903	0	0	0	0	0	4.1709	0
LTIND	0	0	0	0	0	0	0	0	0	0
OFF	4.1709	0	1.3903	0	0	0	0	0	0	0
OPN	63.954	51.441	5.5612	33.367	1.3903	0	0	0	2.7806	4.1709
REC	1.3903	0	0	0	0	0	0	16.684	0	0
SF	37.538	0	0	0	16.684	0	0	11.122	63.954	8.3418
TOTALPER										
SUBSHED	183.52	55.612	25.025	59.783	23.635	0	0	27.806	84.808	26.416

Figure 3

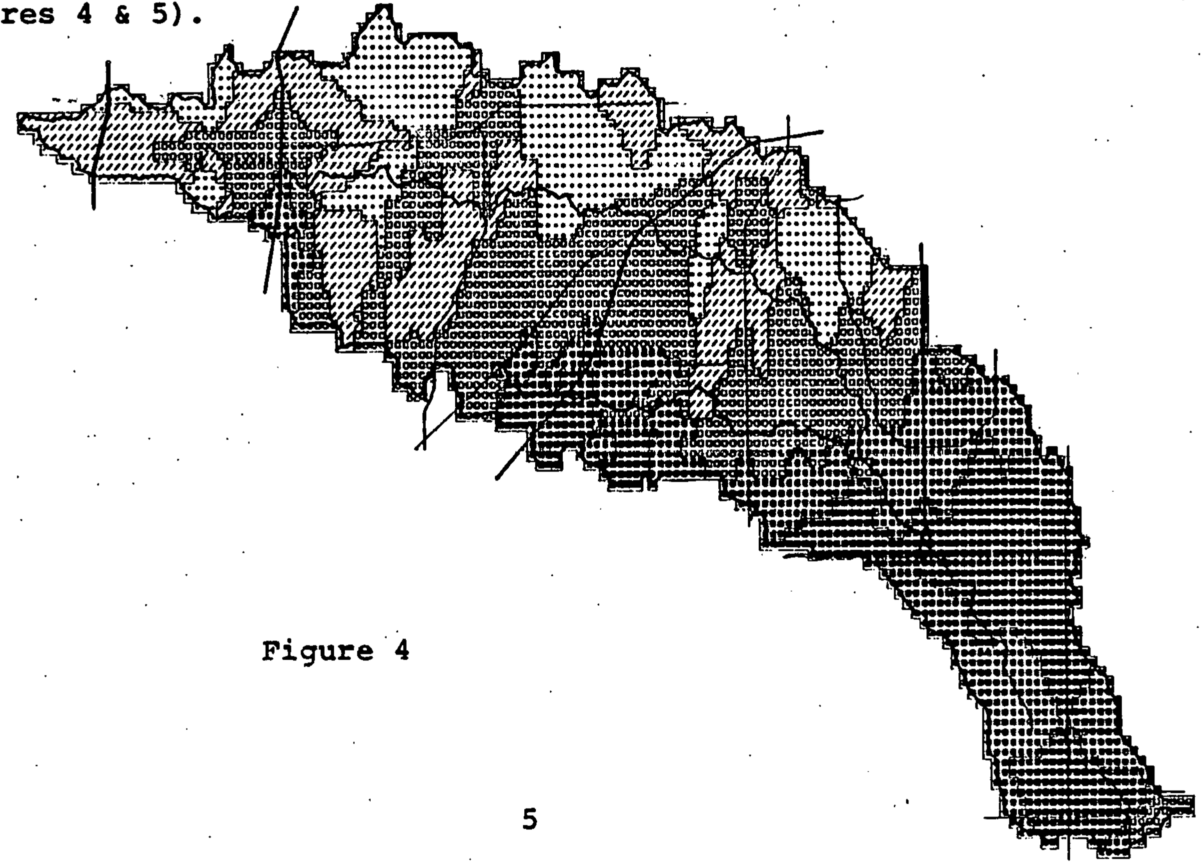


Figure 4

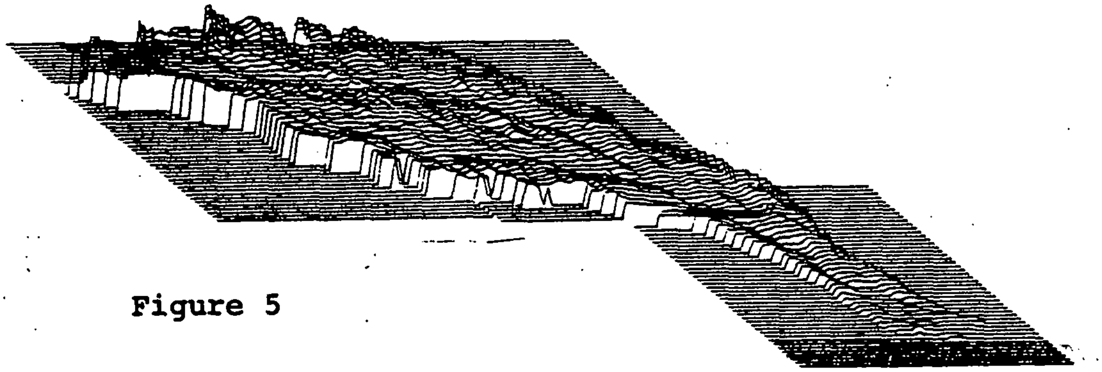


Figure 5

In practice while Rspeak is TSO bound, it was nonetheless possible in a 512K TSO region to construct relations that contained as many as 14 data categories across 4074 grid cells. Queries to these numerical relations were very economical. The advantage of Rspeak in this application was that it provided a unified format for data analysis.

Urban Analysis

The focus of information processing in RAGIS at the urban scale has been to support the exploration of spatial data for the purposes of defining inherent structures and relations in urban spatial distributions and processes. In addition to the processing of data to support theories and models, well managed data processing is beginning to yield new insights into the nature of urban problems. Descriptive data processing (Figure 6) is increasingly becoming a problem identification, solving and monitoring process by iteslf. The traditional sequence of problem identification, model building, data gathering, data processing and conclusion making is breaking up into more integrated relationships with the information descriptive of the problem. Rspeak provided an excellent capability to implement this integrated approach.

DESCRIPTIVE DATA PROCESSING

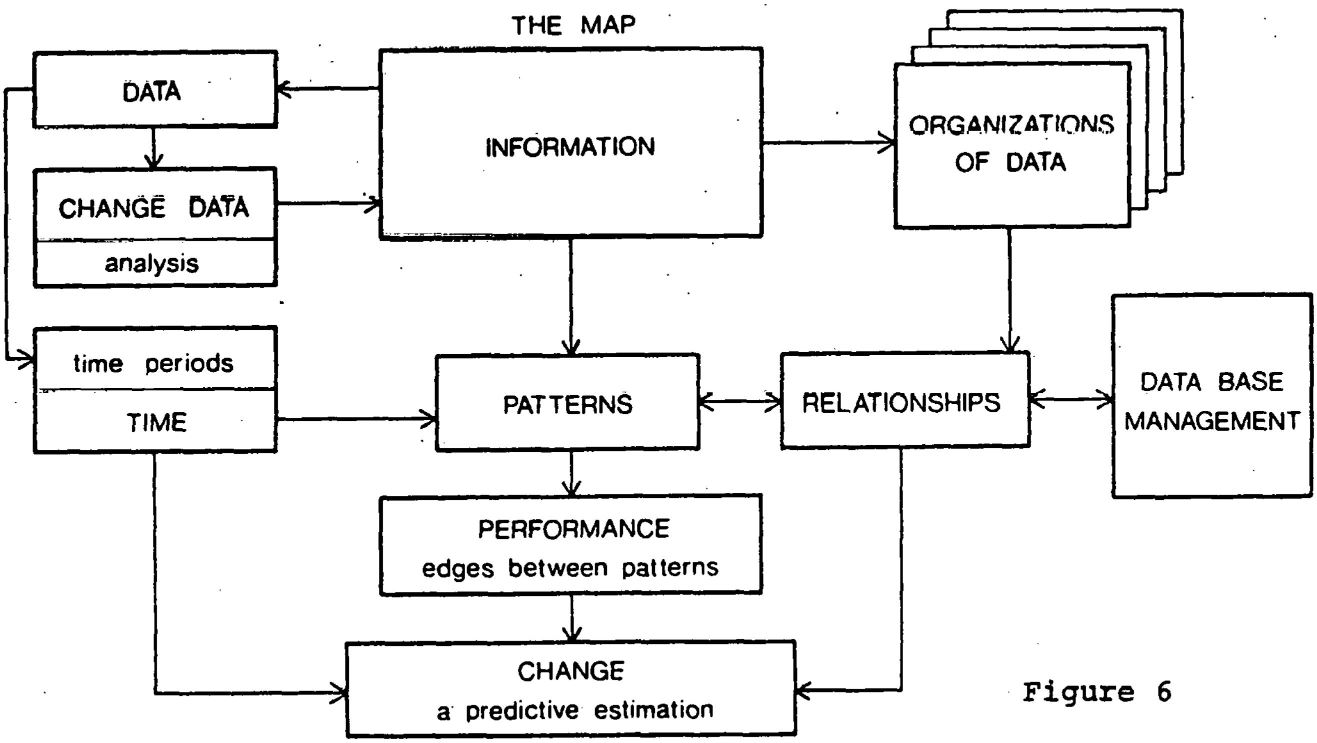


Figure 6

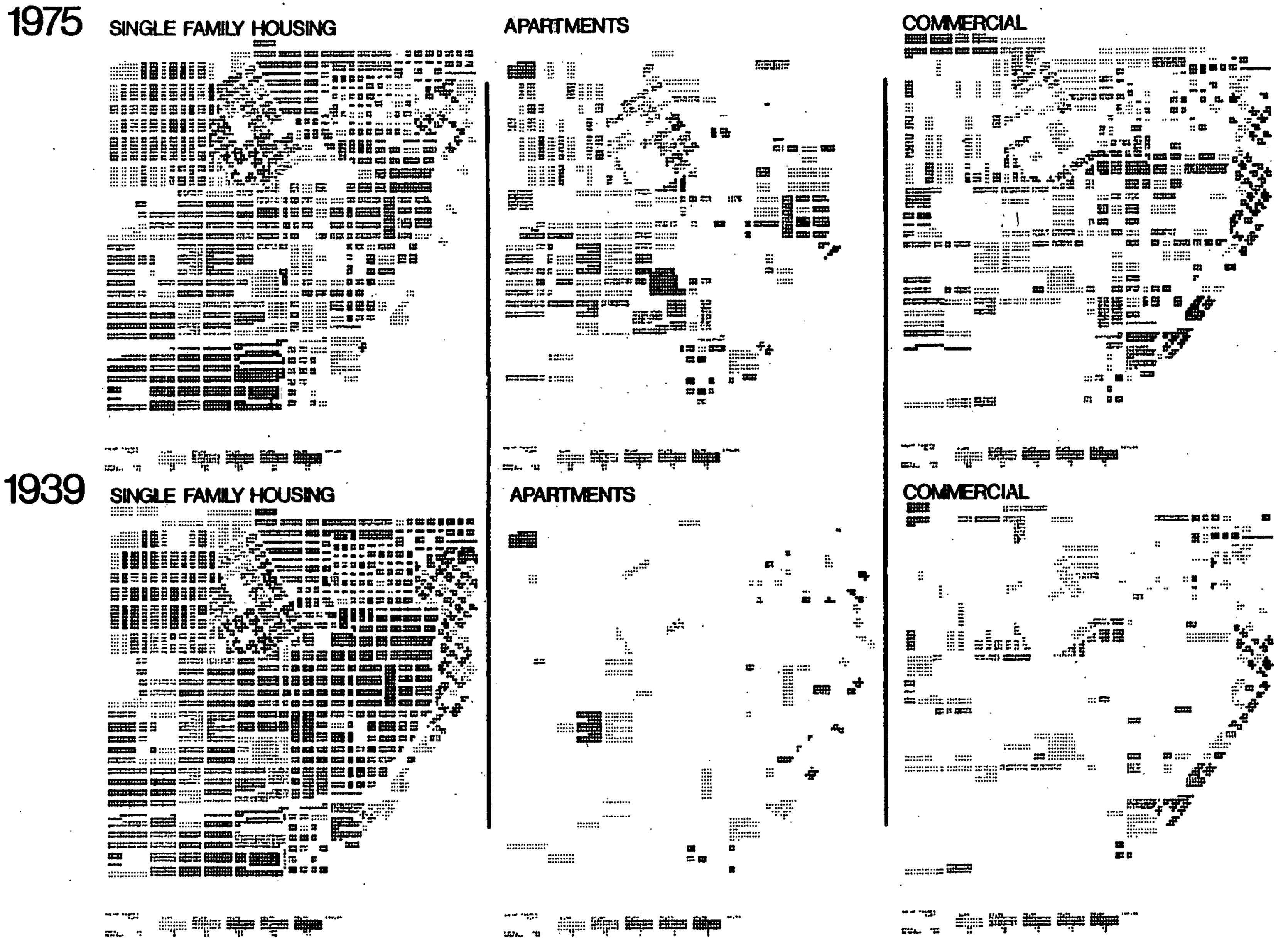
The operative mechanism of descriptive data processing is organization and the capability to reorganize interactively at any time to accomodate the nature of explorative inquiry. RAGIS is designed to relate to the ill-defined class of problems involving urban sociological, economical, and environmental criteria. Rspeak also offered the opportunity to deal with objective and subjective concerns. While a vast amount of information descriptive of urban characteristics and growth has a spatial index, an equally vast amount does not. Generally, information that deals with cause related issues falls into the latter category. Subjective information related to policies, issues, opinions, and feelings is a large subset in this area and is not usually recorded as raw data on a map. Subjectivity generally has objective precedents, but the relationships between the two are neither necessarily direct nor logical. It is essential that any study of cause and effect be capable of not only working in consideration of spatial and non-spatial information, but that it also have the capability for relating the two with sensitivity. By implementing special case relations dealing with issues with relations containing the data descriptive of urban blocks, an operational integration of these concerns was achieved.

Futher capabilities of the relational data base allowed data to be perceived in relationship to time and change. This was accomplished directly with the data and was not dependent on analytical simulation. This capability was implemented through the arrangement of relations in terms of time increments allowing the amount of change to be mapped (Figure 7). Alternatively, when the tables are organized in increments of change, the time spans can be equally perceived. The relationship between the amount of change and units of time is one of the most important perceptions that any planner or decision maker can gain from urban data management. Perceived relationships are gained so directly using Rspeak that nonspecialists can visualize and implement these relationships.

Health Planning

Passage of the National Health Planning Resources Development Act (PL 93-641) has provided this country with a singular framework for health planning with goals written at the macro scale and guidelines written at the mirco scale. The resulting differences in scale among the sources of information needed by health planners frustrated conventional information management approaches unable to deal with such inconsistencies among the data. The relational data approach was used to overcome these problems of scale of information and to explore a data management approach to the understanding of such issues as health status in a clinical and a social sense; incorporating the concepts of illness and wellness indicators and predictive epidemiological planning; the urban and natural environments as a context for for health planning separated from the health and medical care delivery system; the use of the map as the means for seeing not only the spatial distribution of the population and their health status but also the care delivery system available to meet their needs; and facility planning based on the design of environments at the room scale to deliver specific kinds of care. The relational data model has managed successfully the varying scales of information both in original data and in the relationships between diverse data sets, and has proven to be a

SELECTED DATA PATTERNS 1975/1939



1975

SINGLE FAMILY HOUSING

APARTMENTS

COMMERCIAL

1939

SINGLE FAMILY HOUSING

APARTMENTS

COMMERCIAL

Figure 7

most effective mechanism for representing the interrelationships among variously scaled data to the information manager.

The matrix organization of data relationships has proven to be a workable approach to organizing on a fundamental level the information relationships relating to this complex application (Figure 8). The relational data model was used to make functional a multi-dimensional matrix. RACL has been studying an urban area of Houston called the Montrose and a coastal region near Houston for ways in which information relationships could describe the forces which shape the development of an urban or natural envrionment, and with the addition of the Census information organization it is now possible to relate changes in populations to those forces. Joining the health care delivery system organization to the matrix has made it possible to incorporate the principles of demographic-based planning into the nature of growth and decay in the urbanization process, providing a comprehensive population-based planning capability. Spatially representing the management of those diverse data sets in the relational data model provides a very powerful visual aid to health planners.

The flow of relationships expressed in the matrix organization can best be represented in the connectivity of a network diagram. The health status of a population can be studied both clinically and socially, and in terms of the map, which serves as a context for the interrelating of the four major subject areas in the matrix: the urban and natural environments, the census and health (Figure 9). The impacts of changes in health status on services, facilities and manpower planning can be traced throught the series of relations in each of those three health resource areas. Facility design is linked to the network through the International Classification of Disease Index (ICDA-8), making it possible to measure the impact on built form of changes in the health status indicators of a population. To be effective, population-based health planning must be able to incorporate this holistic approach to studying the interrelationships among the population, their environments, and the health care delivery system.

The use of generic facility names by health planners obscures the detail of design data, making it difficult to plan accurately for the allocation of manpower and service resources in the community. RACL has used the relational data base capabilities of Rspeak to discover how relationships fuctionally and structurally between designed environments for the delivery of care can be directly linked to specific population health care needs and related resources required for the delivery of care. Using a data base containing data describing the environment in each of approximately 500 rooms across 167 criteria, tremendous progress has been made into to very nature of design, particularly the potentials for systems design and modular construction. The relational approach has greatly aided research into a process for the design of population-based environments or collections of environments for the delivery of health care without relying on the generic facility planning process.

This approach offers the first workable solution that actually deals with the information associated with these complex areas. Managing the interrelationships between diverse data sets with differently scaled data has enriched the process of population-based health planning, and has allowed the development of a comprehensive analysis and

INFORMATION RELATIONSHIPS FOR PLANNING

- a matrix approach as an index to the organization of data -

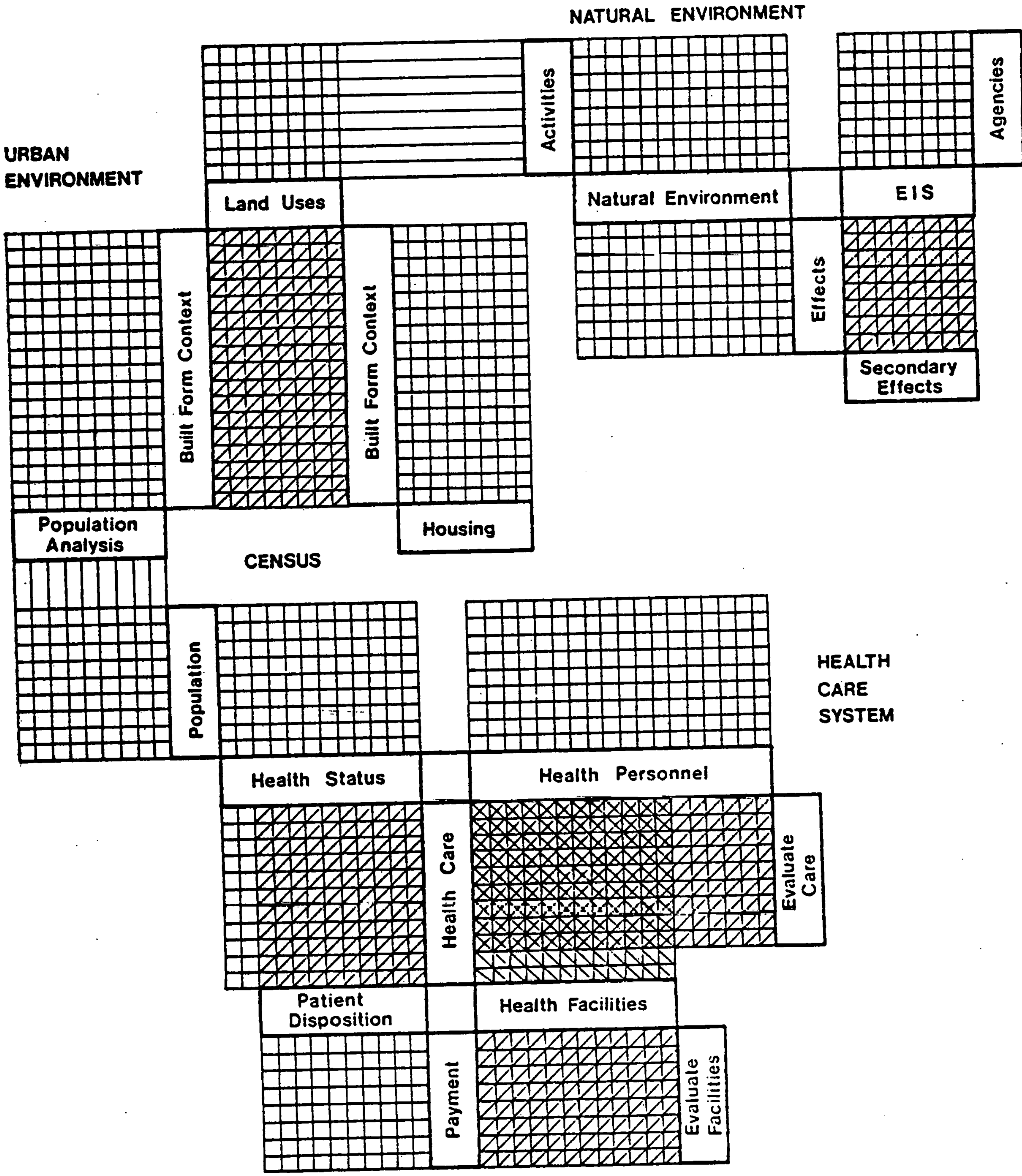
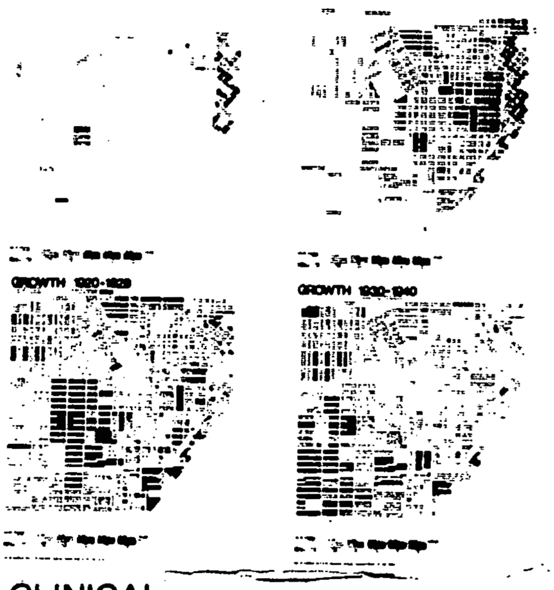
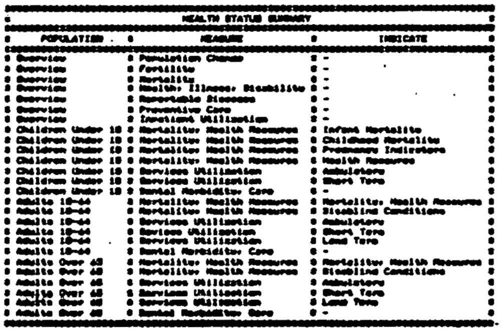


Figure 8

MAPS



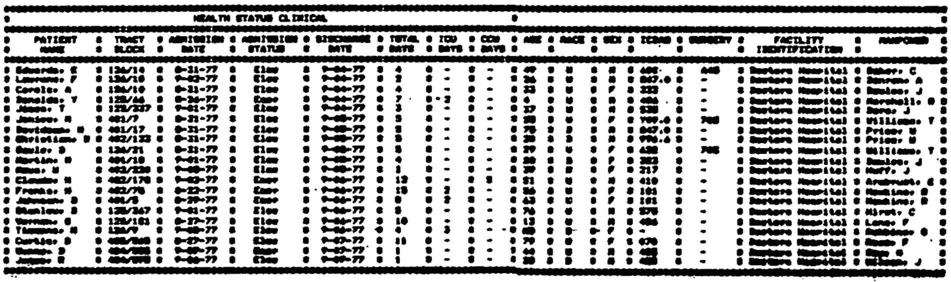
HEALTH STATUS



HEALTH STATUS SUMMARY

POPULATION	MEASURE	INDICATE
Overview	Population Change	-
Overview	Fertility	-
Overview	Mortality	-
Overview	Health, Illness, Disability	-
Overview	Reportable Diseases	-
Overview	Preventive Care	-
Overview	Inpatient Utilization	-
Children Under 18	Mortality, Health Resources	Infant Mortality
Children Under 18	Mortality, Health Resources	Childhood Mortality
Children Under 18	Mortality, Health Resources	Pregnancy Indicators
Children Under 18	Mortality, Health Resources	Health Resources
Children Under 18	Services Utilization	Ambulatory
Children Under 18	Services Utilization	Short Term
Children Under 18	Dental Morbidity, Care	-
Adults 18-64	Mortality, Health Resources	Mortality, Health Resources
Adults 18-64	Mortality, Health Resources	Disabling Conditions
Adults 18-64	Services Utilization	Ambulatory
Adults 18-64	Services Utilization	Short Term
Adults 18-64	Services Utilization	Long Term
Adults 18-64	Dental Morbidity, Care	-
Adults Over 65	Mortality, Health Resources	Mortality, Health Resources
Adults Over 65	Mortality, Health Resources	Disabling Conditions
Adults Over 65	Services Utilization	Ambulatory
Adults Over 65	Services Utilization	Short Term
Adults Over 65	Services Utilization	Long Term
Adults Over 65	Dental Morbidity, Care	-

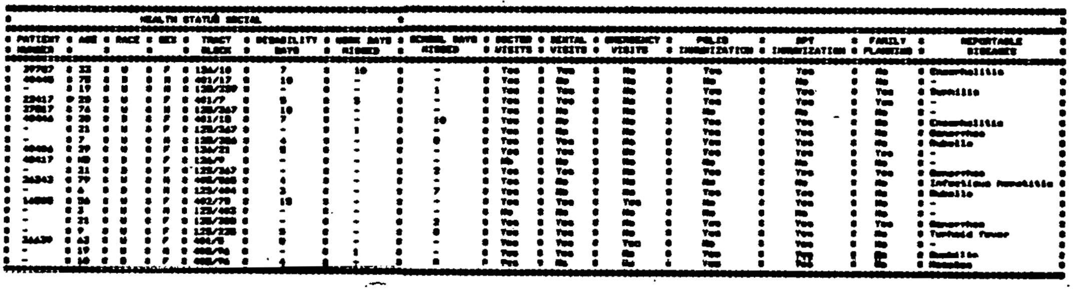
HEALTH STATUS CLINICAL



HEALTH STATUS CLINICAL

PATIENT NAME	TRACT BLOCK	ADMISSION DATE	ADMISSION STATUS	DISCHARGE DATE	TOTAL DAYS	ICU DAYS	CCU DAYS	AGE	RACE	SEX	ICDA8	SURGERY	FACILITY IDENTIFICATION	MANPOWER
Edwards, E	126/14	8-31-77	Elec	9-08-77	4	-	-	40	W	M	465	445	[illegible] Hospital	Baker, C
Lawrence, F	126/10	9-02-77	Elec	9-06-77	2	-	-	26	W	M	847.0	-	[illegible] Hospital	Burrows, A
Carole, A	126/10	8-31-77	Elec	9-06-77	4	-	-	23	W	F	323	-	[illegible] Hospital	Dunlap, J
Donald, Y	125/44	8-30-77	Emer	9-06-77	7	2	-	4	W	M	486	-	[illegible] Hospital	Marshall, B
James, T	125/237	9-01-77	Elec	9-04-77	3	-	-	37	W	M	530	-	[illegible] Hospital	[illegible], J
Jones, H	401/7	8-31-77	Elec	9-05-77	5	-	-	28	W	F	V25.0	[illegible]	[illegible] Hospital	Williams, T
Davidson, M	401/17	8-31-77	Elec	9-05-77	5	-	-	75	B	M	847.0	-	[illegible] Hospital	Price, M
Christiana, B	402/133	8-31-77	Elec	9-05-77	5	-	-	28	B	M	[illegible]	-	[illegible] Hospital	Price, M
[illegible], D	126/21	8-31-77	Elec	9-05-77	5	-	-	29	W	F	423	[illegible]	[illegible] Hospital	Williams, T
[illegible], M	401/10	9-01-77	Elec	9-05-77	4	-	-	20	B	F	323	-	[illegible] Hospital	Dunlap, J
[illegible], M	402/230	9-05-77	Elec	9-05-77	1	-	-	30	B	F	217	-	[illegible] Hospital	Huff, J
Claude, M	402/170	9-02-77	Emer	9-06-77	12	-	5	51	W	M	410	-	[illegible] Hospital	[illegible], C
Franks, M	402/78	8-22-77	Emer	9-06-77	15	2	-	56	W	F	101	-	[illegible] Hospital	[illegible], B
[illegible], D	401/5	8-29-77	Emer	9-06-77	8	2	-	43	B	F	101	-	[illegible] Hospital	[illegible], B
[illegible], D	125/367	9-01-77	Elec	9-06-77	5	-	-	76	W	M	575	-	[illegible] Hospital	Hirst, C
[illegible], G	125/161	8-27-77	Elec	9-06-77	10	-	-	13	W	M	486	-	[illegible] Hospital	Lane, F
Thompson, H	126/9	9-05-77	Elec	9-06-77	4	3	-	60	B	F	-	-	[illegible] Hospital	[illegible], G
Curtis, D	405/[illegible]	8-27-77	Elec	9-07-77	11	-	-	79	W	F	070	-	[illegible] Hospital	[illegible], F
[illegible], D	404/[illegible]	9-05-77	Emer	9-07-77	1	-	-	44	W	M	[illegible]	-	[illegible] Hospital	[illegible], G
[illegible], E	404/[illegible]	9-06-77	Elec	9-07-77	1	-	-	20	B	M	[illegible]	-	[illegible] Hospital	[illegible], J

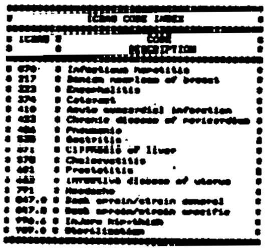
HEALTH STATUS SOCIAL



HEALTH STATUS SOCIAL

PATIENT NUMBER	AGE	RACE	SEX	TRACT BLOCK	DISABILITY DAYS	WORK DAYS MISSED	SCHOOL DAYS MISSED	DOCTOR VISITS	DENTAL VISITS	EMERGENCY VISITS	POLIO IMMUNIZATION	DPT IMMUNIZATION	FAMILY PLANNING	REPORTABLE DISEASES
39767	33	W	F	126/10	7	10	-	Yes	Yes	No	Yes	Yes	No	Encephalitis
40445	75	B	M	401/17	10	-	-	Yes	No	No	No	No	No	-
-	19	W	M	128/230	-	-	1	Yes	Yes	No	Yes	Yes	Yes	Syphilis
22417	28	W	F	401/7	5	5	-	Yes	Yes	No	Yes	Yes	Yes	-
37817	74	W	M	125/367	10	-	-	Yes	No	No	No	No	No	-
40446	20	B	F	401/18	7	-	10	Yes	No	No	Yes	Yes	No	Encephalitis
-	21	W	F	125/367	-	1	-	Yes	No	No	Yes	Yes	No	Gonorrhea
-	7	W	M	128/286	4	-	0	Yes	Yes	No	Yes	Yes	No	Rubella
40486	29	W	F	126/21	5	-	-	Yes	Yes	No	Yes	Yes	Yes	-
40417	60	B	F	126/9	-	-	-	No	No	No	No	No	No	-
-	21	B	F	125/367	-	-	2	Yes	Yes	No	Yes	Yes	Yes	Gonorrhea
26343	79	W	M	405/[illegible]	4	-	-	Yes	No	No	No	No	No	Infectious Hepatitis
-	6	B	M	125/404	3	-	7	Yes	No	No	Yes	Yes	No	Rubella
14888	26	W	F	402/78	13	-	-	Yes	Yes	Yes	No	Yes	No	-
-	3	W	M	123/403	-	-	-	No	No	No	No	No	No	-
-	21	W	F	125/288	-	-	2	Yes	Yes	No	Yes	Yes	Yes	Gonorrhea
-	9	W	F	125/228	5	-	0	Yes	Yes	No	Yes	Yes	No	Typhoid fever
24430	43	W	F	404/5	0	-	-	Yes	Yes	Yes	No	Yes	No	-
-	19	B	M	408/96	-	1	-	Yes	Yes	No	Yes	Yes	No	Rubella
-	10	B	F	408/96	4	-	0	Yes	No	No	Yes	Yes	No	Measles

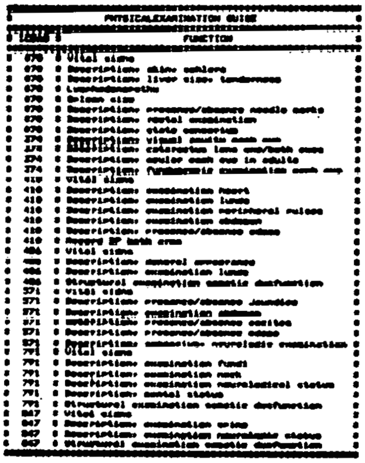
DISEASES



ICDA8 CODE INDEX

ICDA8	CODE DESCRIPTION
070	Infectious hepatitis
217	Benign neoplasm of breast
323	Encephalitis
374	Cataract
410	Acute myocardial infarction
423	Chronic disease of pericardium
486	Pneumonia
535	Gastritis
571	Cirrhosis of liver
575	Cholecystitis
601	Prostatitis
[illegible]	Inflammative disease of uterus
791	Headache
847.0	Back sprain/strain general
847.9	Back sprain/strain specific
[illegible]	Induce birth
V25.0	Sterilization

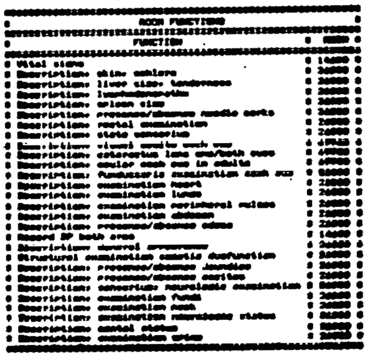
EXAMINATIONS



PHYSICALEXAMINATION GUIDE

ICDA8	FUNCTION
070	Vital signs
070	Description, skin, sclera
070	Description, liver size, tenderness
070	Lymphadenopathy
070	Spleen size
070	Description, presence/absence needle marks
070	Description, rectal examination
070	Description, state conscious
374	Description, visual acuity each eye
374	Description, cataracts one eye/both eyes
374	Description, ocular each eye in adults
374	Description, fundoscopic examination each eye
410	Vital signs
410	Description, examination heart
410	Description, examination lungs
410	Description, examination peripheral pulses
410	Description, examination abdomen
410	Description, presence/absence edema
410	Record BP both arms
486	Vital signs
486	Description, general appearance
486	Description, examination lungs
486	Structural examination somatic dysfunction
571	Vital signs
571	Description, presence/absence jaundice
571	Description, examination abdomen
571	Description, presence/absence ascites
571	Description, presence/absence edema
571	Description, [illegible] neurologic examination
791	Vital signs
791	Description, examination fundi
791	Description, examination neck
791	Description, examination neurological status
791	Description, mental status
791	Structural examination somatic dysfunction
847	Vital signs
847	Description, examination spine
847	Description, examination neurologic status
847	Structural examination somatic dysfunction

ROOMS



ROOM FUNCTIONS

FUNCTION	ROOM
Vital signs	[illegible]
Description, skin, sclera	[illegible]
Description, liver size, tenderness	[illegible]
Description, lymphadenopathy	[illegible]
Description, spleen size	[illegible]
Description, presence/absence needle marks	[illegible]
Description, rectal examination	[illegible]
Description, state conscious	[illegible]
Description, visual acuity each eye	[illegible]
Description, cataracts one eye/both eyes	[illegible]
Description, ocular each eye in adults	[illegible]
Description, fundoscopic examination each eye	[illegible]
Description, examination heart	[illegible]
Description, examination lungs	[illegible]
Description, examination peripheral pulses	[illegible]
Description, examination abdomen	[illegible]
Description, presence/absence edema	[illegible]
Record BP both arms	[illegible]
Description, general appearance	[illegible]
Structural examination somatic dysfunction	[illegible]
Description, presence/absence jaundice	[illegible]
Description, presence/absence ascites	[illegible]
Description, [illegible] neurologic examination	[illegible]
Description, examination fundi	[illegible]
Description, examination neck	[illegible]
Description, examination neurologic status	[illegible]
Description, mental status	[illegible]
Description, examination urine	[illegible]

Figure 9

management capability keyed to performance for health information and health planning.

A Management Application

The U.S. Department of Energy (DOE) for a number of years has sponsored research and field applications in the Enhanced Oil Recovery (EOR) program. In conjunction with a local research group the applicability of Speakeasy and Rspeak to manage data related to this project was explored. EOR processes are both highly technical and touch a number of specialized sciences including chemical engineering,and geology. Across the nation many different approaches have been taken to this problem which further compounds the complexity of gaining insight into the overall problem area. Data related to these many applications and different processes had to be organized and searched both for specific and general relationships.

The major advantage of Speakeasy and Rspeak was that a piece meal approach to the construction of a complex data base could be undertaken. Thus data related to the aspects of the overall problem which were well defined could be organized and used to clarify the problem aspects which were less well defined. There can exist no singular description of the overall problem area which satisfies all criteria. Chemical species, field conditions, research assumptions, recovery processes, government policy, and varying data conditions must all be interrelated and managed across time. Speakeasy and Rspeak supported a "cut and paste" approach to the construction of a complex data base. The relational approach to data management required that neither an overall framework be defined nor all of the specific relationships identified and understood. The result was that an overall schema for data management emerged from fragmented sections of the problem.

CONCLUSION

As computer applications leave the age of the individual data file and enter the age of data base management systems, a unique set of problems begins to appear. One of the major problems with packaged data management systems is that their command languages are generally so weak. As Speakeasy, which is one of the most powerful interactive command languages in use today, incorporates data base management capabilities (Gspeak and Rspeak) and concepts it offers a viable system for today's data management problems.

Through the various applications one theme remained constant. Rspeak was as much a tool for data management as a vehicle for seeing and understanding relational data management concepts. This understanding has greatly expanded RACL's problem organization and solving capabilities. Also, the perception of Speakeasy as a relational structure emerged. This is mainly an aspect of Speakeasy's "umbrella" effect on diverse data processing. The perception of Speakeasy as a growing concept emerged over the perception of Speakeasy as a set of finite capabilities.

REFERENCES

Bedient, Philip B., et. al., Preliminary Feasibiltiy Report for the Bear Creek, Texas, Local Protection Project, The Southwest Center for Urban Research, Houston, Texas, May, 1978.

Date, C.J., An Introduction to Database Systems, Second Edition, Addison-Wesley, Reading, Massachusetts, 1977.

Rowe, P.G. and Blackburn, J.B., et. al., A Preliminary Investigation of Structural and Nonstructural Flood Control Alternatives for Cypress Creek, Texas, School of Architecture, Rice University, Houston, Texas, September, 1977.

Schlichting, Richard D., "Data Base Manipulation in Speakeasy", Speakeasy Computing Corporation, Chicago, Illinois, unpublished, 1977.

SURVEY ANALYSIS WITH SPEAKEASY

J.P. KIEFFER - G. LAPLANCHE

C.E.C.T.I. - UNIVERSITY OF LIEGE - BELGIUM

August 1978.

Two objectives underlie the development of the system described below. They are :

1. To satisfy an immediate need at the University of LIEGE for a specific and user-friendly tool to process survey data.

2. To define the nucleus of a larger system that should be all at once complete, independent and easy to maintain and extend without losing the basic advantages and facilities of interactive work under SPEAKEASY.

The present version of this system must be considered as a first attempt. Its main objectives actually concern its formal characteristics rather than its contents. A lot of extensions are still to be implemented and not presented here.

The data collected by surveys consist of vectors, that is statistical variables defined on the sample of people consulted. Most of these variables consist of categorical data. The processing of these data generally involves two types of statistical methods :

- Multivariate analysis like factor analysis. Such methods process all the data globally looking for general properties of the sample studied;
- Methods of analysis for single variables, pairs or multiplets of chosen variables. This leads, for instance, to cross-classifications or contingency tables.

So far, the present system is primarily concerned with the second approach. In fact, it is the only one where interactive processing is a major constraint as will appear from the example given below. In this approach, the survey analysis itself proceeds by chaining questions translated by specific but restricted statistical techniques.

So, the system appears as a set of functions (linkules). Each linkule matches an elementary function as needed by this kind of survey analysis. But in respect to this context, they work somewhat differently from the basic functions of SPEAKEASY so they may be considered as a subsystem. Their pecularities may be summarized as follows :

- In order to ensure full-compatibility with the basic functions of SPEAKEASY and, at the same time, to optimize their processing, the linkules implicitly create SPEAKEASY objects to contain their results. These objects are automatically checked out and retrieved by further linkule executions. Thus, they behave like classical arguments but need not be predefined and if necessary they are conventionally named by the linkule that creates them.

- All results are automatically displayed on the terminal with explicit titles and in a friendly presentation. The system includes linkules to put them out on a printer in the same fashion or to display them back on the terminal without recalculations.

- All functions basically make allowance for various pecularities of the survey data : they can indifferently process numerical and literal data; they process separately possible missing values; the reading in and writing out of data may be conditional i.e. that only those data are transmitted that correspond to individuals characterized by given responses to some variables ("men only" for instance!).

To be self-complete, the system offers specific functions for all needs encountered in the context of a survey analysis : not only statistical computation functions but also file access functions, for instance. Obviously, SPEAKEASY already contains a lot of basic functions that may be useful in survey analysis. These will not be listed in the classification given below. All of the linkules listed are currently available and most of them are used in the example listed at the end.

In agreement with the general character of SPEAKEASY, the use of each function in the system is described by a corresponding HELP.

Moreover, the system provides for explicit error messages.

<u>List of available linkules</u>

1. <u>Input - Output functions</u>

READFILE : (conditional) reading of data on file
WRITEFILE : (conditional) writing of data on file
PRINTOUT : writing out of results on a printer
SCREEN : displaying back results on the terminal.

2. <u>Manipulations - transformations of variables</u>

VARGROU	: variable transformations by	merging two variables
VARMOD		value changes
VARCLAS		class definition

EXTRACT : Conditional subsample extraction on a set of variables.

GATHER : Merging multiple answers into a single variable

3. Statistical methods

FREQD1
FREQD2
FREQD3 } : single and crossed response distributions

FREQD1 FREQD2 FREQD3	: single and crossed response distributions
PRECON2 PRECON3 INDEP2 QASI2 MARMO2 SYMET2 CONTI3 MULCRO2	Preparing or processing 2-way or 3-way contingency tables with various statistical null-hypotheses

4. System management

FREESURV : Scratching all objects created by the system.

Example

To illustrate the use of the system, we have listed a session using most of the functions above. The original data are taken from an actual survey stored in card-image on a file (FT10F001). The answers of one individual fill three punched cards but only variables from the cards coded 1 in column 80 are used in this example.

Although the listing was obtained by BATCH execution, the session illustrated simulates an actual session at a terminal. This explains why the results of the PRINTOUT and WRITEFILE functions appear on a separate file (FT20F001).

```
      SPEAKEASY 3 MU+  4:43 PM JULY 13, 1978
INPUT... FMT='(1X,F1.0,1X,F1.0,4X,F2.0,1X,A1,41X,3A2,18X,A1)'
INPUT... READFILE(1),FMT,300:CIVSTAT,SEX,PROF,REASON,C1,C2,C3,CODE:CODE,'1')
      END OF FILE ENCOUNTERED
      210  OBS. STORED FOR  630  OBS. PROCESSED ON THE FILE
INPUT... FREQD1(PROF)
      *** FREQ. DIST. OF VAR. PROF
        TOT. NUMBER OF OBS.         210
        NO MIS. VAL. CONSID.
      CAT1  OFTABL    PCTABL
      ****  ******  *********
       11     15      7.1429
       12     36     17.143
       21      9      4.2857
       22     25     11.905
       23     19      9.0476
       32     28     13.333
       33      2       .95238
       41      8      3.8095
       42     26     12.381
       43     29     13.81
       51      7      3.3333
       52      1       .47619
       61      5      2.381
INPUT... CLAS=19 29 39 49
INPUT... VARCLAS(PROF:CLAS)
INPUT... FREQD1(PROF)
      *** FREQ. DIST. OF VAR. PROF
        TOT. NUMBER OF OBS.         210
        NO MIS. VAL. CONSID.
      CAT1  OFTABL   PCTABL
      ****  ******  ********
        1     51     24.286
        2     53     25.238
        3     30     14.286
        4     63     30
        5     13      6.1905
INPUT... FREQD1(REASON,' ')
      *** FREQ. DIST. OF VAR. REASON
        TOT. NUMBER OF OBS.         210
            19 OBS. LOST FOR MIS. VAL.          I.E. %    9.048
        REMAINING SAMPLE SIZE       191
      CAT1  OFTABL   PCTABL
      ****  ******  ********
        1      46    24.084
        2      32    16.754
        3       7     3.6649
        4      10     5.2356
        5      40    20.942
        6       9     4.712
        7      28    14.66
        8      11     5.7592
        9       8     4.1885
INPUT... OLD=' ' '1' '2' '3' '4' '5' '6' '7' '8' '9'
INPUT... NEW=0 1 2 3 3 5 5 7 8 8
INPUT... VARMOD(REASON:OLD,NEW)
INPUT... INDEP2(SEX:REASON,0)
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR      INDEPENDENCE
        VAR. NAMES  : SEX                REASON
        MIS. VALUES :   NONE                C.0
        OBS. LOST   :          19 OUT OF       210 I.E. %    9.048
        SAMPLE SIZE :          191
      DEGFRE   CHIVAL   PROBA1   LOGCHI   PROBA2
      ******  *******  *******  *******  *******
        5      6.0093   .30532   6.1652   .29048
      CAT1  CAT2  OFTABL    TFTABL     CHIELS
      ****  ****  ******  ********  *********
        1     1     23      17.581    1.6702
        1     2     10      12.23      .40674
        1     3      6       6.4974    .038075
        1     5     20      18.728     .086429
        1     7     10      10.702     .045993
        1     8      4       7.2618   1.4651
        2     1     23      28.419    1.0333
        2     2     22      19.77      .25163
        2     3     11      10.503     .023555
        2     5     29      30.272     .053469
        2     7     18      17.298     .028454
        2     8     15      11.738     .90637
INPUT... PRINTOUT
       SCREEN PRINTED
INPUT... NAMES
        CURRENTLY DEFINED NAMES
      OFTABL , MARG2 , PCTABL , NEW , FMT , CIVSTAT , CHIVAL , DEGFRE , LOGCHI , OLD , CAT1 , CAT2 , TFTABL , CHIELS , MARG1 ,
      SEX , PROBA1 , PROBA2 , PROF , CLAS , C1 , C2 , C3 , SCREEN , REASON
INPUT... FREESURV
INPUT... NAMES
        CURRENTLY DEFINED NAMES
      NEW , FMT , CIVSTAT , OLD , SEX , PROF , CLAS , C1 , C2 , C3 , REASON
INPUT... FREQD1(C1)
      *** FREQ. DIST. OF VAR. C1
        TOT. NUMBER OF OBS.          210
        NO MIS. VAL. CONSID.
      CAT1  OFTABL    PCTABL
      ****  ******  *********
             10       4.7619
       11    32      15.238
       12    16       7.619
       13    17       8.0952
       14    43      20.476
       15     4       1.9048
       16     1        .47619
       17    58      27.619
       18     7       3.3333
       19    12       5.7143
       21     6       2.8571
       22     3       1.4286
       33     1        .47619
INPUT... VARCLAS(C1,C2,C3:Y1,Y2,Y3:('11','13','16'),' ')
INPUT... FREQ02(Y2,0:Y3,0)
      *** 2-WAY CROSS-CLASSIFICATION
        VAR. NAMES   : Y2              Y3
        MIS. VALUES :    0.0              0.0
        OBS. LOST    :               30      210 S.E. %   14.286
        SAMPLE SIZE :        180
                 MARG1              MARG2
      ********  *****  ********   *****
          1       25      1        19
          2       40      2        35
          3       53      3        34
          4       62      4        52
      CAT1  CAT2  OFTABL           PCTABL
      ****  ****  ******  ************************
        1     1      0      0          0          0
        1     2      5     20         14.286     2.7778
        1     3      6     24         17.647     3.3333
        1     4     14     56         15.217     7.7778
        2     1      4     10         21.053     2.2222
        2     2      6     15         17.143     3.3333
        2     3     11     27.5       32.353     6.1111
        2     4     19     47.5       20.652    10.556
        3     1      6     11.321     31.579     3.3333
        3     2     14     26.415     40         7.7778
        3     3      3      5.6604     8.8235    1.6667
        3     4     30     56.604     32.609    16.667
        4     1      9     14.516     47.368     5
        4     2     10     16.129     28.571     5.5556
        4     3     14     22.581     41.176     7.7778
        4     4     29     46.774     31.522    16.111
INPUT... SYMET2
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR     SYMMETRY
        STAT. CONCL. DANGER. :         1 THEOR.FREQ. < 1
        STAT. CONCL. DANGER. :  25.000 % TH. FREQ. < 5
      DEGFRE   CHIVAL  PROBA1   LOGCHI   PROBA2
      ******  *******  ******  *******  *******
        6      10.169   .1177   10.361   .11024
      CAT1  CAT2  OFTABL  TFTABL    CHIELS
      ****  ****  ******  ******  *********
        1     1      0       0      0
        1     2      5       4.5    .055556
        1     3      6       6      0
        1     4     14      11.5    .54348
        2     1      4       4.5    .055556
        2     2      6       6      0
        2     3     11      12.5    .18
        2     4     19      14.5    1.3966
        3     1      6       6      0
        3     2     14      12.5    .18
        3     3      3       3      0
        3     4     30      22      2.9091
        4     1      9      11.5    .54348
        4     2     10      14.5    1.3966
        4     3     14      22      2.9091
        4     4     29      29      0
 NPUT... TITLE=ARRAY(2,3:'USE OF SYMET2          ON VARS Y2 AND Y3')
 NPUT... PRINTOUT(TITLE)
       TEXT IN ARG. PRINTED  SCREEN PRINTED
```

```
INPUT... MARH02
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR     MARGINAL HOMOGENEITY
      DEGFRE   CHIVAL   PROBA1
      ******  *******  ********
         3      9.8471   .019911
      CAT1  CAT2  OFTABL
      ****  ****  ******
        1     1      0
        1     2      5
        1     3      6
        1     4     14
        2     1      4
        2     2      6
        2     3     11
        2     4     19
        3     1      6
        3     2     14
        3     3      3
        3     4     33
        4     1      9
        4     2     13
        4     3     14
        4     4     29
INPUT... CONT13(SEX:CIVSTAT:REASON,C:0.3)
      *** 3-WAY CONTINGENCY TABLE WITH HYPOTHESIS...
        INDEP. OF VARS         3
         VAR. NAMES  : SEX              CIVSTAT          REASON
         MIS. VALUES :   NONE             NONE             0.0
         OBS. LOST   :         19 OUT OF       210 I.E. %    9.048
         SAMPLE SIZE :        191
         STAT. CONCL. DANGER. :        2 THEOR.FREQ. < 1
         STAT. CONCL. DANGER. :  68.111 % TH. FREQ. < 5
      DEGFRE   CHIVAL    PROBA1    LOGCHI    PROBA2
      ******  *******   *******   *******   *******
        25     31.127    .18466    32.443    .14559
      CAT1  CAT2  CAT3  OFTABL    TFTABL     CHIELS        CAT1  CAT2  CAT3  OFTABL    TFTABL      CHIELS
      ****  ****  ****  ******  *********  **********      ****  ****  ****  ******  **********  **********
        1     1     1      5     4.8168     .0069713         2     1     1      5      5.2984      .016809
        1     1     2      3     3.3508     .036723          2     1     2      2      3.6859      .77109
        1     1     3      4     1.7801    2.7683            2     1     3      3      1.9581      .55437
        1     1     5      4     5.1309     .24926           2     1     5      2      5.644      2.3527
        1     1     7      3     2.9319     .00158           2     1     7      6      3.2251     2.3875
        1     1     8      1     1.9895     .49216           2     1     8      4      2.1885     1.4955
        1     2     1     16    10.356     3.0759            2     2     1     15     18.304       .59629
        1     2     2      5     7.2042     .67439           2     2     2     14     12.733       .12608
        1     2     3      2     3.8272     .87237           2     2     3      7      6.7644      .008206
        1     2     5     10    11.031      .096435          2     2     5     22     19.497       .32123
        1     2     7      7     6.3037     .076921          2     2     7      8     11.141       .88572
        1     2     8      3     4.2775     .38153           2     2     8     10      7.5602      .78736
        1     3     1      2     2.4084     .069247          2     3     1      3      4.8168      .68523
        1     3     2      2     1.6754     .062893          2     3     2      6      3.3508     2.0945
        1     3     3      0      .89005    .89005           2     3     3      1      1.7801      .34187
        1     3     5      6     2.5654    4.5981            2     3     5      5      5.1309      .003339
        1     3     7      0     1.466     1.466             2     3     7      4      2.9319      .38908
        1     3     8      0      .99476    .99476           2     3     8      1      1.9895      .49216
```

```
INPUT... CONTI3(0)
      *** 3-WAY CONTINGENCY TABLE WITH HYPOTHESIS...
      NO 3-ORDER INTERACTION
        STAT. CONCL. DANGER. :        4 THEOR.FREQ. < 1
        STAT. CONCL. DANGER. :  66.667 % TH. FREQ. < 5
      DEGFRE   CHIVAL   PROBA1   LOGCHI   PROBA2
      ******  *******  *******  *******  *******
        10     9.8451   .45419   11.268   .33701
      CAT1  CAT2  CAT3  CFTABL     TFTABL      CHIELS        CAT1  CAT2  CAT3  OFTABL     TFTABL      CHIELS
      ****  ****  ****  ******  *********  **********        ****  ****  ****  ******  *********  **********
        1     1     1      5      6.1263     .20705          2     1     1      5      3.8778     .32473
        1     1     2      3      2.1427     .34301          2     1     2      2      2.8566     .25686
        1     1     3      4      3.0158     .3212           2     1     3      3      3.9828     .24253
        1     1     5      4      3.2268     .18526          2     1     5      2      2.7747     .21632
        1     1     7      3      4.0594     .27648          2     1     7      6      4.9394     .22774
        1     1     8      1      1.4262     .12738          2     1     8      4      3.5709     .051553
        1     2     1     16     14.675      .11966          2     2     1     15     16.322      .10714
        1     2     2      5      5.6841     .082344         2     2     2     14     13.316      .035154
        1     2     3      2      2.7107     .18631          2     2     3      7      6.2905     .080033
        1     2     5     10     12.744      .59072          2     2     5     22     19.256      .39111
        1     2     7      7      4.7802    1.0308           2     2     7      8     10.22       .48242
        1     2     8      3      2.408      .14552          2     2     8     10     10.594      .033341
        1     3     1      2      2.1989     .017992         2     3     1      3      2.7997     .014327
        1     3     2      2      2.1732     .013797         2     3     2      6      5.8276     .0050958
        1     3     3      0       .27356    .27356          2     3     3      1       .7267      .10278
        1     3     5      6      4.0295     .56367          2     3     5      5      6.9696     .55659
        1     3     7      0      1.1604    1.1604           2     3     7      4      2.8401     .47368
        1     3     8      0       .16575    .16575          2     3     8      1       .83473     .03272
INPUT... FMT='(4F2.0,1X,3A2)'
INPUT... WRITEFILE(20,FMT,10:CIVSTAT,SEX,PROF,REASON,C1,C2,C3:SEX,1)
      10  OBS. WRITTEN ON THE FILE FOR  32  OBS. PROCESSED
INPUT... PRINTOUT('VARS. SEX , CIVSTAT AND REASON')
       TEXT IN ARG. PRINTED  SCREEN PRINTED
INPUT... GATHER(C1,C2,C3:C)
INPUT... WHATIS C1:WHATIS C
      C1 IS NOT DEFINED
      C IS A 210 BY 3 NAME-LITERAL ARRAY
INPUT... MULTVAR(C,' ')
      *** FREQ. DIST. OF MULTIPLE ANS. VAR.   C
        TOT. NUMBER OF OBS. :        210
        MAX. NUMBER OF ANSWERS         3
        WEIGHTS USED :   3.000   2.000   1.000
        MIS. VAL.   :            FREQ.:        58
      CAT1  WGTABL    PCTWGT
      ****  ******  **********
       11     165    14.175
       12      94     8.0756
       13     122    10.481
       14     199    17.096
       15      63     5.4124
       16      26     2.2337
       17     250    21.478
       18      60     5.1546
       19      31     6.5568
       20      28     2.4055
       21      46     3.9519
       22      24     2.0619
       31       1      .085911
       33       3      .25773
       61       2      .17182
```

```
INPUT... EXTRACT(Y1,Y2,Y3,SEX:SEX,1)
      83  CES. EXTRACTEC
INPUT... INCEP2(Y1:Y2)
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR      INDEPENDENCE
        VAR. NAMES  : Y1                  Y2
        SAMPLE SIZE :         83
        STAT. CONCL. DANGER. :          4 THEOR.FREQ. < 1
        STAT. CONCL. DANGER. :  71.C00 % TH. FREQ. < 5
      DEGFRE    CHIVAL     PROBA1      LOGCHI     PROBA2
      ******   *******  **********   *******   **********
        16      57.122   1.4693E-6   44.781   1.4992E-4
      CAT1   CAT2  OFTABL    TFTABL        CHIELS              CAT1   CAT2  OFTABL    TFTAHL        CHIELS
      ****   ****  ******   ********   ***********             ****   ****  ******   ********    ***********
        0      0      4       .43373    29.323                   2      3      8      4.241       3.3319
        0      1      0       .28916      .28916                 2      4      7      7.8566       .0034279
        0      2      0      1.2048     1.2C48                   3      0      1      1.8434       .38586
        0      3      0       .771C8      .771C8                 3      1      0      1.2289      1.2289
        0      4      0      1.3012     1.3C12                   3      2      6      5.1205       .15107
        1      0      0      1.C843     1.0843                   3      3      1      3.2771      1.5823
        1      1      0       .72289      .72289                 3      4      9      5.5301      2.1772
        1      2      2      3.012        .34005                 4      0      3      3.253        .019679
        1      3      1      1.5277       .44646                 4      1      4      2.1687      1.5465
        1      4      7      3.253      4.316                    4      2     13      9.0361      1.7388
        2      0      1      2.3855       .80473                 4      3      6      5.7831       .0081325
        2      1      2      1.55C4       .10551                 4      4      4      9.759       3.3985
        2      2      4      6.6265     1.0411
INPUT... OF=ARRAY(2,2:17 25 10 32)
INPUT... PRECON2(OF)
INPUT... INCEP2
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR      INCEFENDENCE
      DEGFRE    CHIVAL    PRCBA1    LCGCHI   PROBA2
      ******   *******   *******   *******   ******
        1       2.6745    .1C197    2.6515    .1005
      CAT1   CAT2  CFTABL    TFTABL     CHIELS
      ****   ****  ******    ******    *******
        1      1     17       13.5      .9C741
        1      2     25       28.5      .42982
        2      1     10       13.5      .90741
        2      2     32       28.5      .42962
INPUT... SCREEN
      *** 2-WAY CONTINGENCY TABLE
        TEST  FOR      INDEPENCENCE
      DEGFRE    CHIVAL    PROBA1    LCGCHI   PROBA2
      ******   *******   *******   *******   ******
        1       2.6745    .1C197    2.6975    .1C05
      CAT1   CAT2  OFTABL    TFTABL     CHIELS
      ****   ****  ******    ******    *******
        1      1     17       13.5      .90741
        1      2     25       28.5      .42982
        2      1     10       13.5      .90741
        2      2     32       28.5      .42962
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
        SPACE USED . 19 K NOW,   29 K PEAK,   SIZE    80 K
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
****************************************************
*                                                  *
* U221101                          78.194 16.03.50 *
* U221101.FT21F001                                 *
*                                                  *
*                                                  *
****************************************************
*** 2-WAY CONTINGENCY TABLE
  TEST  FOR      INDEPENDENCE
  VAR. NAMES  : SEX                 REASON
  MIS. VALUES :   NONE                 0.0
  OBS. LOST   :         19 OUT OF      210 I.E. %     9.048
  SAMPLE SIZE :         191
DEGFRE   CHIVAL   PROBA1   LOGCHI   PROBA2
******  ******** ******* ******* *******
  5      6.0193   .30532   6.1652   .29048
CAT1  CAT2  OFTABL   TFTABL     CHIELS
****  ****  ******  ********  *********
  1     1     23     17.581    1.6702
  1     2     10     12.23      .40674
  1     3      6      6.4974    .038075
  1     5     20     18.728     .086429
  1     7     10     10.702     .045993
  1     8      4      7.2618   1.4651
  2     1     23     28.419    1.0333
  2     2     22     19.77      .25163
  2     3     11     10.503     .023555
  2     5     29     30.272     .053465
  2     7     18     17.298     .028494
  2     8     15     11.738     .90637

USE OF SYMET2
ON VARS Y2 AND Y3
-----------------------

*** 2-WAY CONTINGENCY TABLE
  TEST  FOR      SYMMETRY
  STAT. CONCL. DANGER. :          1 THEOR.FREQ. < 1
  STAT. CONCL. DANGER. :  25.000 % TH. FREQ. < 5
DEGFRE   CHIVAL   PROBA1   LOGCHI   PROBA2
******  ******** ******* ******* *******
  6      10.169   .1177   10.361   .11024
CAT1  CAT2  OFTABL  TFTABL    CHIELS
****  ****  ******  ******  *********
  1     1      0       0      0
  1     2      5       4.5     .055556
  1     3      6       6      0
  1     4     14      11.5     .54348
  2     1      4       4.5     .055556
  2     2      6       6      0
  2     3     11      12.5     .18
  2     4     19      14.5    1.3966
  3     1      6       6      0
  3     2     14      12.5     .18
  3     3      3       3      0
  3     4     30      22      2.9091
  4     1      9      11.5     .54348
  4     2     10      14.5    1.3966
  4     3     14      22      2.9091
  4     4     29      29      0
```

```
2.1.2.1. 111712
2.1.4.2. 171112
2.1.3.2. 141517
3.1.1.5. 121114
2.1.3.2. 1420
1.1.3.3. 131214
2.1.4.8. 152211
2.1.4.3. 141721
3.1.4.5. 171113
2.1.2.5. 111421

 VARS. SEX ,CIVSTAT AND REASCN
 ------------------------------

 *** 3-WAY CONTINGENCY TABLE WITH HYPOTHESIS...
 NC 3-ORDER INTERACTION
   STAT. CCACL. DANGER. :        4 THEOR.FREQ. < 1
   STAT. CCACL. DANGER. :  66.667 % TH. FREQ. < 5
 DEGFRE   CHIVAL   PROBA1   LOGCHI   PROBA2
 ******  *******  *******  *******  *******
   10     9.1451   .42415   11.266   .33701
 CAT1  CAT2  CAT3  OFTABL    TFTABL     CHIELS
 ****  ****  ****  ******  *********  **********
   1     1     1      5      6.1263     .20705
   1     1     2      3      2.1421     .34301
   1     1     3      4      3.0156     .3212
   1     1     5      4      3.2266     .18526
   1     1     7      3      4.0594     .27640
   1     1     8      1      1.4262     .12738
   1     2     1     16     14.675      .11966
   1     2     2      5      5.6843     .082344
   1     2     3      2      2.7101     .18631
   1     2     5     13     12.744      .55072
   1     2     7      7      4.1002    1.0308
   1     2     8      3      2.408      .14552
   1     3     1      2      2.1969     .017592
   1     3     2      2      2.1732     .013797
   1     3     3      0       .27356    .27356
   1     3     5      6      4.0259     .56367
   1     3     7      0      1.1604    1.1604
   1     3     8      0       .16535    .16535
   2     1     1      5      3.8776     .32473
   2     1     2      2      2.8566     .25666
   2     1     3      3      3.9828     .24253
   2     1     5      2      2.7747     .21632
   2     1     7      6      4.9394     .22774
   2     1     8      4      3.5709     .051553
   2     2     1     15     16.322      .10714
   2     2     2     14     13.316      .035154
   2     2     3      7      6.2905     .080033
   2     2     5     22     19.256      .39111
   2     2     7      8     10.22       .48242
   2     2     8     11     11.594      .030341
   2     3     1      3      2.7997     .014327
   2     3     2      6      5.8270     .0050558
   2     3     3      1       .7267     .10270
   2     3     5      5      6.5656     .55659
   2     3     7      4      2.8608     .47368
   2     3     8      1       .83473    .03272
```

F4STAT and Its Future Under SPEAKEASY

D. Saxe and W. VanHassel

6th Annual SPEAKEASY Conference

August 18, 1978

The Linkules written at ETS are based on our statistical subroutines package, F4STAT. Special statistical matrix operators for performing least squares analysis form the core of both F4STAT and the ETS linkule library. Versions of the F4STAT routines for frequency distributions, crosstab analysis and summary statistics have been added to ETS links during the last year. During the coming year we will be working toward implementation of regression (univariate), analysis of variance (ANOVA), and perhaps other routines (multivariate regression, factor analysis, MANOVA). We hope to work closely with the University of Liege in order to complement and not duplicate efforts toward providing a wider assortment of statistical routines for SPEAKEASY.

So far, our Linkule writing has been at least partly experimentation with different techniques. This process will probably continue through this next year before we decide on a comprehensive approach to our goal of providing interactive statistical support for the novice user. Our help documents are attached for your review. We would welcome your comments on any aspect of our effort.

F4STAT lists words available from the ETS F4STAT library.

ACCUM	see ACCUMULATE
ACCUMULATE	computes weighted frequency distributions.
CROSSTAB	computes two-way weighted & unweighted tables.
DIRPRO	computes the direct product of 2 objects.
MSIG	returns means, standard deviations and ranges.
MSTD	returns a multistandardized square matrix or 2 dim object.
PFROMZ	returns the probability of a score in a normal distribution.
POLY	polytomizes a discrete variable.
SDG	returns a step-diagonalized square matrix or 2 dim object.
STD	returns a standardized square matrix or 2 dimensional object.
SWP	returns a swept square matrix or 2 dimensional object.
TCM	returns a transformed square matrix or 2 dimensional object.
XTAB	see CROSSTAB
ZFROMP	returns the z score associated with a probability.

For more information on F4STAT operators and their uses see: "The Use of Special Matrix Operators in Statistical Calculus" by Albert Beaton, Educational Testing Service, Research Bulletin RB-64-51, Princeton, NJ, 1964. Available on request.

To find out more about a command "XXX" enter "HELP XXX". dhs

ACCUMULATE computes weighted frequency distributions.
ACCUMULATE(W:C) returns three new objects, the first contains the distinct values found in 'C', the second contains the observed frequencies of those values, and the third contains the weighted frequencies of those values using the co-related weight values contained in 'W'.

The default naming convention for the newly created objects is as follows: The original name of 'C' will be truncated to four characters, if necessary, in order to append the letters 'VALU' for the distinct values, 'FREQ' for the observed frequencies and the first four letters of 'W' name for the accumulated weights. These defaults can be overridden by the call ACCUMULATE(W:C:NAME1, NAME2,NAME3).

ACCUMULATE(W1,W2,...WN:C1,C2,...CN) COMPUTES MULTIPLE DISTRIBUTIONS. When nc is the number of 'C' objects and nw is the number of 'W' objects, nc*(nw+2) objects will be created. ACCUMULATE(W1,W2,... Wn:C1,C2,...Cn:NAME1,NAME2,...NAMEn) again overrides the default naming conventions. Names should be entered in the following order, C1VALU, C1FREQ, C1W1, C1W2,...C1Wn, C2VALU, C2FREQ, C2W1, C2W2, ...C2Wn...CnVALU etc.

If the number of 'given' names does not equal the number of objects to be created, the routine either uses or defaults as many names as needed. If an object with a created or 'given' name already exists, it will be replaced. No indication of this action is returned. An attempt to create more than one object with the same name will terminate the routine.

'C' objects must be 1 dimensional real or name-literal, 'W' objects must be 1 dimensional real objects. All objects must have the same number of elements.

When there is only one 'C' object and no names are given, the calling sequence ACCUMULATE(W1,W2,...Wn,C) may be used (ie, the colon may be omitted).

ACCUMULATE W1 W2 ... Wn BY C1 C2 ... Cn is an alternate calling sequence when no names are given.

Unweighted distribution(s) may be computed by any of the following calls:

```
ACCUMULATE(C)
ACCUMULATE(:C1,C2,...Cn)
ACCUMULATE(:C1,C2,...Cn:NAME1,NAME2,...NAMEn)
```

ACCUM is a synonym for ACCUMULATE.

CROSSTAB computes two way weighted frequency distributions. CROSSTAB(R,C) returns three new objects, the first contains the distinct values found in 'R', the second contains the distinct values found in 'C', and the third contains the observed frequencies of those values (using the distinct values of 'R' for the row dimension and the distinct values of 'C' for the column dimension).

The default naming convention for the newly created objects IS AS FOLLOWS: THE ORIGINAL NAMES OF 'R' AND 'C' WILL BE TRUNCATED to four characters, if necessary, in order to append the letters 'VALU' for the first two objects respectively. The third object is named by concatenating the first two letters of 'R' name and 'C' name and appending the letters 'FREQ'. These defaults may be overridden by the call CROSSTAB(R,C:NAME1,NAME2,NAME3).

CROSSTAB(R,C,W1,W2,...Wn) returns the same three objects as above and in addition, returns a two dimensional object for each 'W', containing the two way weighted distribution of the distinct values found in 'R' and 'C' using the co-related values found in 'W'. The default names may be overridden by the call CROSSTAB(R,C,W1, W2,...Wn:NAME1,NAME2,...NAMEn).
Names should be entered in the following order: RVALU, CVALU, RCFREQ, RCW1, RCW2,...RCWn.

If the number of 'given' names does not equal the number of objects to be created, the routine either uses or defaults as many names as needed. If an object with a created or 'given' name already exists, it will be replaced. No indication of this action is returned. Any attempt to create more than one object with the same name will cause the routine to terminate.

'R' and 'C' objects must be 1 dimensional real or name-literal, 'W' objects must be 1 dimensional real. All objects must have the same number of elements.

XTAB is a synonym for CROSSTAB.

DIRPRO RETURNS THE DIRECT PRODUCT OF TWO OBJECTS.
DIRPRO(A,B) RETURNS THE DIRECT PRODUCT OF A AND B. IT IS OFTEN USED TO FORM A CELL MEMBERSHIP VECTOR FOR SEVERAL DISCRETE VARIABLES USING THE RESULT OF A POLY OPERATION. A AND B ARE REAL AND USUALLY VECTORS. THE RESULT IS AN OBJECT OF NOELS(A)*NOELS(B) ELEMENTS CONTAINING ALL THE PAIRWISE PRODUCTS OF THE ELEMENTS OF A AND B. THE ELEMENTS OF B VARY MOST RAPIDLY SO THE RESULT CONTAINS (A(1)*B(1),A(1)*B(2),....). THIS IS EQUIVALENT TO VEC(NOELS(A)*NOELS(B):OUTERPRO(A,B)). IF A AND B ARE VECTORS, THE RESULT IS A VECTOR. IF EITHER A OR B IS A SCALAR, THE RESULT IS THE NORMAL SPEAKEASY PRODUCT. IF EITHER OR BOTH A AND B

ARE 1 DIMENSIONAL ARRAYS, THE RESULT WILL BE A 1 DIMENSIONAL ARRAY. IF BOTH A AND B ARE 2 DIMENSIONAL WITH THE SAME NUMBER OF ROWS, THEN THE RESULT WILL BE 2 DIMENSIONAL WITH THE SAME NUMBER OF ROWS AS A AND B AND NOCOLS(A)*NOCOLS(B) COLUMNS. IN THIS CASE, EACH ROW OF THE RESULT IS THE SAME AS THE DIRPRO EXPLAINED ABOVE; I.E. ROW ONE OF THE RESULT IS THE SAME AS DIRPRO(A(1),B(1)). THE RESULT WILL BE AN ARRAY UNLESS BOTH A AND B ARE MATRICES, IN WHICH CASE THE RESULT WILL BE A MATRIX. SEE ALSO THE HELP DOCUMENTS FOR POLY,OUTERPRO,HIWIDE,F4STAT. DHS

MSIG(A,B,...C) returns means, standard deviations and ranges.

MSIG(A,B,...C) returns the mean, standard deviation and range for each 'column' of the input variable list. Input variables must be real, but may be vectors, arrays or matrices. If an input variable is a two dimensional object, the computations are done on the columns. The output is a two dimensional (nx5) real array containing one row for each 'column' in the input list. The columns of the output array are: (1) number of observations, (2) mean, (3) standard deviation, (4) lowest observed value and (5) highest observed value. MISSING DATA: The routine checks for the presence of an object named MISVAL, if it exists, all observed values equal to its value(s) are eliminated from the computations. The standard deviation is computed as:

SQRT(SUMXSQ-SUMX*MEANX)/(N-1)

MSTD RETURNS A MULTISTANDARDIZED MATRIX OR 2 DIMENSIONAL ARRAY. MSTD(A,LIST) RETURNS THE RESULT OF MULTISTANDARDIZING THE OBJECT A USING A CHOLESKY TRIANGULAR FACTORIZATION OF THE SUBMATRIX, A(LIST,LIST). A IS A REAL, SQUARE, SYMMETRIC OR SKEW SYMMETRIC MATRIX OR 2 DIMENSIONAL ARRAY WITH A(LIST,LIST) SYMMETRIC. THE NUMBER OF ROWS (COLUMNS) OF A MUST BE GREATER THAN ONE. A REMAINS UNCHANGED BY MSTD. THE RETURNED OBJECT WILL HAVE THE SAME SIZE AND CLASS AS A. LIST IS A SCALAR OR 1 DIMENSIONAL OBJECT CONTAINING THE ROW AND COLUMN INDICES OF THE SUBMATRIX TO BE MULTISTANDARDIZED. MORE THAN ONE LIST MAY BE USED AS IN THE ALTERNATE CALLING SEQUENCE: MSTD(A,LIST1,LIST2,...LISTN). THE TRANSFORMATION MATRIX (CHOLESKY TRIANGULAR FACTORIZATION) MAY BE OBTAINED BY THE CALLING SEQUENCE: MSTD(A,LIST1,LIST2,...LISTN:T). UNLESS OTHERWISE SPECIFIED, T WILL BE DEFINED AND INITIALIZED TO AN IDENTITY MATRIX (2 DIMENSIONAL ARRAY IF A IS AN ARRAY) BEFORE THE TRANSFORMATION IS COMPUTED. T WILL HAVE THE SAME SIZE AS A. THE POSITIONAL PARAMETER, ASIS, MAY BE USED TO INDICATE THAT THE USER HAS ALREADY DEFINED AND PLACED INITIAL VALUES IN T. E.G.: MSTD(A,LIST:T,ASIS). USING THE PARAMETER UNIT IN PLACE OF ASIS IS THE SAME AS MSTD(A,LIST). SEE ALSO THE HELP DOCUMENT FOR F4STAT. DHS

PFROMZ RETURNS THE PROBABILITY OF A SCORE IN A NORMAL DISTRIBUTION. PFROMZ(Z) RETURNS THE PROBABILITY OF A STANDARD SCORE IN A NORMAL DISTRIBUTION. THE PROBABILITY IS THE AREA UNDER THE NORMAL CURVE TO THE RIGHT OF THE GIVEN SCORE. THE RETURNED PROBABILITIES WILL HAVE THE SAME CLASS AND STRUCTURE AS Z, WHICH MUST BE REAL. SEE ALSO THE HELP DOCUMENTS FOR ZFROMP AND F4STAT. DHS

POLY POLYTOMIZES A DISCRETE VARIABLE.
POLY(NUMCAT,CATEGORY) POLYTOMIZES THE DISCRETE VARIABLE CATEGORY AND RETURNS A VECTOR FOR USE IN FORMING A CROSS PRODUCTS MATRIX. NUMCAT IS THE MAXIMUM NUMBER OF CATEGORIES AND CATEGORY IS A SCALAR GIVING THE CATEGORY VALUE FOR ONE OBSERVATION. THE RESULT OF POLY WILL BE A VECTOR OF NUMCAT-1 ELEMENTS. THE CONTENTS OF THE VECTOR DEPENDS ON THE VALUE OF CATEGORY:

CATEGORY EQ NUMCAT: ALL THE ELEMENTS OF THE VECTOR ARE SET TO -1.

CATEGORY LE NUMCAT AND GT 0: THE CATEGORY ELEMENT IS SET TO 1 AND ALL OTHER ELEMENTS ARE SET TO 0.

CATEGORY EQ 0: A WARNING MESSAGE IS ISSUED AND THE VECTOR SET TO 0.

CATEGORY LT 0 OR GT NUMCAT: AN ERROR MESSAGE IS ISSUED AND NO RESULT IS DEFINED.

MORE THAN ONE OBSERVATION MAY BE HANDLED BY USING AN ALTERNATE CALLING SEQUENCE: POLY(NUMCAT,CAT1,CAT2,...CATN) WHERE CAT1 TO CATN ARE SCALARS OR ONE DIMENSIONAL OBJECTS. IN THIS CASE, POLY RETURNS A MATRIX WITH NOCOLS = NUMCAT - 1 AND NOROWS = SUM OF THE NUMBER OF ELEMENTS OF CAT1,...CATN. EACH ROW OF THE RESULTING MATRIX REPRESENTS THE RESULT OF A SINGLE POLY OPERATION AS DEFINED IN THE FIRST CALLING SEQUENCE.
SEE ALSO THE HELP DOCUMENTS FOR F4STAT AND DIRPRO. DHS

SDG RETURNS A STEP-DIAGONALIZED SQUARE MATRIX OR 2 DIMENSIONAL OBJECT.
SDG(A,LIST) RETURNS THE RESULT OF STEP-DIAGONALIZING THE OBJECT A USING A MODIFIED JACOBI METHOD. THE ADVANTAGE OF THIS METHOD IS THAT WHILE COMPUTING THE EIGENVALUES VECTORS) IT SIMULTANEOUSLY TRANSORMS THE REST OF THE OBJECT. A IS A REAL, SQUARE, SYMMETRIC OR SKEW-SYMMETRIC MATRIX OR 2 DIMENSIONAL ARRAY WITH A(LIST,LIST) SYMMETRIC. THE NUMBER OF ROWS (COLUMNS) OF A MUST BE GREATER THAN ONE. THE RETURNED OBJECT WILL HAVE THE SAME SIZE AND CLASS AS A. LIST IS A SCALAR OR 1 DIMENSIONAL OBJECT CONTAINING THE ROW AND COLUMN INDICES OF THE SUBMATRIX TO BE DIAGONALIZED. MORE THAN ONE LIST MAY USED IN THE ALTERNATE CALLING SEQUENCE: SDG(A,LIST1,LIST2....LISTN). SEVERAL OPTIONS ARE AVAILABLE AND MUST APPEAR AFTER A COLON, BUT MAY BE IN ANY ORDER. THE OPTIONS HILO, LOHI, NOSORT SPECIFY THAT THE EIGENVALUES AND CORRESPONDING PARTS OF A SHOULD BE SORTED HIGH TO LOW, LOW TO HIGH, OR NOT AT ALL, RESPECTIVELY. HILO IS THE DEFAULT. IF AN ARGUMENT APPEARS WHICH IS NOT A SORT OPTION AND NOT ASIS OR UNIT (EXPLAINED BELOW), THEN THE ARGUMENT WILL BE DEFINED WITH THE SAME STRUCTURE AS A AND WILL CONTAIN THE EIGENVECTORS OF A(LIST,LIST) IN THE CORRESPONDING LOCATIONS. SPECIFYING THE OPTION UNIT INDICATES THAT THE OBJECT TO RECEIVE THE EIGENVECTORS SHOULD BE INITIALIZED AS AN IDENTITY MATRIX. ASIS INDICATES THAT THE OBJECT IS ALREADY DEFINED AND INITIALIZED AND SHOULD BE TRANSFORMED. IF THE OBJECT IS T AND THE EIGENVECTOR MATRIX IS V, THE RESULTING T (USING ASIS) WILL BE T*V WHERE * DENOTES MATRIX MULTIPLICATION. IF NEITHER ASIS NOR UNIT IS GIVEN, UNIT IS ASSUMED. AN EXAMPLE OF THE CALLING SEQUENCE USING OPTIONS IS: SDG(A,1,2,I:ASIS,T,LOHI). SDG USES A CONSTANT, SDGCON, TO CHECK FOR CONVERGENCE DURING ITS ITERATIONS. THE DEFAULT VALUE IS .00001. THIS VALUE MAY BE CHANGED BY SETTING SDGCON TO THE DESIRED VALUE BEFORE USING SDG. NEGATIVE VALUES ARE IGNORED AND THE DEFAULT USED. SEE ALSO THE HELP DOCUMENTS FOR F4STAT, GEIGEN, EIGENSYS, EIGENVALS, AND EIGENVECS. DHS

STD RETURNS A STANDARDIZED SQUARE MATRIX OR 2 DIMENSIONAL ARRAY. STD(A) RETURNS THE RESULT OF STANDARDIZING THE OBJECT A, WHERE A IS A REAL SQUARE MATRIX OR 2-DIMENSIONAL ARRAY WITH NON-NEGATIVE DIAGONAL ELEMENTS. IF T IS A DIAGONAL MATRIX WITH SQRT(DIAGELS(A)) ON ITS DIAGONAL, THEN STD(A) IS THE SAME AS TRANSPOS(T)*A*T WHERE THE * DENOTES MATRIX MULTIPLICATION. IF A DIAGONAL ELEMENT IS NEGATIVE, STD WRITES AN ERROR MESSAGE AND RETURNS NO RESULT. IF A DIAGONAL ELEMENT IS EXACTLY ZERO, THE ELEMENTS OF THAT ROW AND COLUMN WILL BE SET TO ZERO AND A WARNING MESSAGE ISSUED. THE DIAGONAL ELEMENTS OF THE TRANSFORMING MATRIX, T, MAY BE OBTAINED BY AN ALTERNATE CALLING SEQUENCE: STD(A,D). THE RESULT OF STD WILL HAVE THE SAME CLASS AND STRUCTURE AS A. IF A IS A MATRIX, D WILL BE A VECTOR; IF A IS AN ARRAY, D WILL BE AN ARRAY. SEE ALSO THE HELP DOCUMENT FOR F4STAT. DHS

SWP RETURNS A SWEPT SQUARE MATRIX OR 2 DIMENSIONAL OBJECT. SWP(A,LIST) RETURNS THE RESULT OF SWEEPING THE OBJECT A. EACH ROW AND COLUMN SWEPT REPRESENTS ONE STEP IN THE GAUSSIAN ELIMINATION MATRIX INVERSION ALGORITHM. A IS A REAL SQUARE SYMMETRIC OR SKEW SYMMETRIC POSITIVE OR SEMI DEFINITE MATRIX OR TWO DIMENSIONAL ARRAY WITH NOROWS(A)=NOCOLS(A) GREATER THAN 1. THE RETURNED OBJECT WILL HAVE THE SAME CLASS AS A. LIST IS A SCALAR, ONE DIMENSIONAL ARRAY OR VECTOR CONTAINING THE INDICES OF THE ROWS AND COLUMNS TO BE SWEPT. IF THE ELEMENTS OF LIST ARE NOT INTEGERS, THE THE TRUNCATED INTEGER VALUES ARE USED. AN ALTERNATE CALLING SEQUENCE, SWP(A,LIST1,LIST2,...,LISTN), WHERE LIST1 THROUGH LISTN ARE LIKE LIST ABOVE, RETURNS AN OBJECT WITH ALL THE ELEMENTS OF LIST1 THROUGH LISTN SWEPT FROM A. SWP(A) RETURNS THE MATRIX INVERSE OF A. INVERSE(A) SHOULD BE USED INSTEAD OF THIS LAST CALLING SEQUENCE WHEN ANSWERS OF HIGHER NUMERICAL ACCURACY ARE REQUIRED. SWP USES THE VALUE OF ACCURACY (DEFAULT=10E-8) TO CHECK THE MAGNITUDE OF THE DIAGONAL BEFORE SWEEPING. IF THE DIAGONAL IS LESS THAN ACCURACY THEN THE MATRIX IS ASSUMED TO BE SINGULAR, AN ERROR MESSAGE IS WRITTEN, AND SWP RETURNS WITHOUT DEFINING A RESULT. SEE ALSO THE HELP DOCUMENTS FOR INVERSE, ACCURACY, DETERMINANT AND F4STAT.

DHS

TCM RETURNS A TRANSFORMED SQUARE MATRIX OR 2 DIMENSIONAL OBJECT. TCM(A,T) RETURNS TRANSPOS(T)*A*T WHERE THE MULTIPLICATION IS MATRIX MULTIPLICATION. A IS A REAL, SQUARE, SYMMETRIC OR SKEW SYMMETRIC MATRIX OR TWO DIMENSIONAL ARRAY. T IS A REAL OBJECT WITH NROWS(T)=NCOLS(A). NROWS(A)=NCOLS(A) MUST BE GREATER THAN 1. THE RETURNED OBJECT WILL BE N BY N IF T IS M BY N. IF N=1, THEN THE RETURNED OBJECT WILL BE A SCALAR; IF N IS GREATER THAN 1, THEN IT WILL BE OF THE SAME CLASS AS A. SEE ALSO THE HELP DOCUMENT FOR F4STAT.

DHS

ZFROMP RETURNS THE Z SCORE ASSOCIATED WITH A PROBABILITY. ZFROMP(P) RETURNS THE STANDARD SCORE CORRESPONDING TO A PROBABILITY IN A NORMAL DISTRIBUTION. P IS ANY REAL OBJECT CONTAINING PROBABILITIES STRICTLY LESS THAN 1 AND GREATER THAN 0. THE PROBABILITY IS ASSUMED TO BE THE AREA UNDER THE NORMAL CURVE TO THE LEFT OF THE RETURNED STANDARD SCORE. SEE ALSO THE HELP DOCUMENTS FOR PFROMZ AND F4STAT. DHS

SOLVING PARTIAL DIFFERENTIAL EQUATIONS IN SPEAKEASY

by Daniel F. X. O'Reilly(*)
Department of Mathematics and Statistics
Marquette University

The physical world offers a wide range of problems which can be mathematically modeled using partial differential equations (pde's). The heat equation and the wave equation are just two of the many examples which come to mind. However, finding analytic solutions for pde's is much more complicated than for ordinary differential equations. It frequently involves a clever guess regarding the nature of the solution. For example, in a textbook solution, it may look like the author is pulling a rabbit out of the hat when he says something like, "Suppose $u(x,y) = f(x)g(y)$." But there are many situations in which it doesn't matter how clever we are because an analytic solution simply does not exist. In these cases or in the situation where we have run out of clever guesses, we must be satisfied with a numerical solution.

Even numerical methods for the solution of pde's have problems, though. Many methods will exhibit instability if we are not careful as we refine the approximate solution using a finer and finer mesh. And even stable methods can prove totally useless on large problems. For behind every numerical approximation to the world of the continuum for a pde lurks a huge linear system, perhaps involving a 1000 by 1000 matrix. In fact, given present computer core sizes, our codes are severely limited in their ability to solve three-dimensional partial differential equations.

A rather successful subroutine package designed specifically for elliptic boundary value problems was recently developed at the National Center for Atmospheric Research by Swarztrauber and Sweet. Basically, it can solve the two-dimensional modified Helmholtz equation (and hence the more well known equations of Laplace and Poisson) using cartesian, polar, cylindrical or spherical coordinates, when either Dirichlet or Neumann boundary conditions are given. Recently, a large portion of this package was incorporated into a linkule called HELMHOLTZ in the Speakeasy language. It is the purpose of this paper to describe some of the features of this linkule. In addition, there have been some

---

(*) Work done as a faculty research participant at Argonne National Laboratory, June 5 to August 18, 1978.

advances in other directions which will soon make Speakeasy one of the most potent and versatile tools for the numerical solution of pde's.

One of the more significant of these changes is the incorporation of the LINPACK subroutine package into Speakeasy. LINPACK is a collection of state-of-the-art subroutines for the solution of linear systems which has been under development at Argonne and several universities over the past few years. Linkules like INVERSE, SIMEQ and SIMEQUAT, to name a few, will soon be replaced by much more efficient codes which will take advantage of band structure and symmetries in the arguments. There will also be added protection against overflow and underflow problems. Since the typical linear system which results from numerically approximating a partial differential equation is very large and sparse (probably with band structure), it is expected that the more sophisticated linear system solvers will help us considerably in this area.

Finally, I would like to indicate some future directions for development which should greatly enhance our capabilities. Numerical techniques for conformal mappings as well as the addition some additional orthogonal function families, like the Chebyshev polynomials, will enlarge the domain of the problems we can solve and also make the use of finite element and spectral methods quite easy.

The first problem I would like to consider involves the direction of fluid flow around a cylindrical object. For the purpose of simplification, let us assume that the flow is irrotational and that the fluid is incompressible with low viscosity. In such a situation the velocity potential is a harmonic function, ie. it is a solution to Laplace's equation:

$$(d/dx)(du/dx) + (d/dy)(du/dy) = 0$$

The elementary theory of complex variables tells us that this velocity potential is the real part of an analytic function and we know that the level curves of the imaginary part of this function will give us the streamlines associated with the flow. Rather than use some exotic subroutine package to solve this problem, we can make use of an obvious solution to the problem when there is no obstruction to the flow, and then use a conformal mapping to give the solution in the more complicated region. It is easy to see that g(z)=z is an analytic function associated with the unobstructed flow of a fluid across the complex plane from say left to right. Now if we consider another complex plane where the fluid is constrained to flow around the unit circle, it suffices to find a conformal mapping which takes the upper half plane onto the upper half plane minus the unit circle. Such a function is known to exist by the Riemann mapping theorem and in this case it is quite easy and can be written down in closed form:

$$f(z) = z + 1/z$$

Given these elementary results from physics and the theory of complex variables, let us now consider how easy it is to graphically display the streamlines associated with this flow using a handful of Speakeasy commands.

```
:_domain complex
:_x=colarray(grid(-2,2,1/8))
:_y=rowarray(grid(.01,4.01,1/8))
:_z=x + 1i*y
:_w=imag(z+1/z)
:_graphics tek4012
:_h=grid(0,max(w),max(w)/10)
:_contour w h
```

In figure 1 we see the associated streamlines. Note the use of the fact that points inside the unit circle above the real axis are mapped into the lower half plane, when we only consider positive values of w in defining the array h.

Needless to say, the above example is highly specialized and even when the domain is quite simple the nature of the boundary conditions will generally make the solution impossible to "guess". To aid us in this area, we have a new linkule in Speakeasy called HELMHOLTZ. This linkule calls the subroutine package of Swarztrauber and Sweet developed at NCAR. Specifically, it is possible to find numerical solutions to the modified Helmholtz equation (The actual Helmholtz equation is homogeneous.):

$$(\text{Laplacian})u + \text{const}*u = f$$

in an interactive setting. At this time, one is limited to solving the above equation in 2 dimensions with either cartesian or polar coordinates. As an example of how one might use the new linkule, consider the following problem: it is desired to solve

$$(d/dx)(du/dx) + (d/dy)(du/dy) - 2*u = 0$$

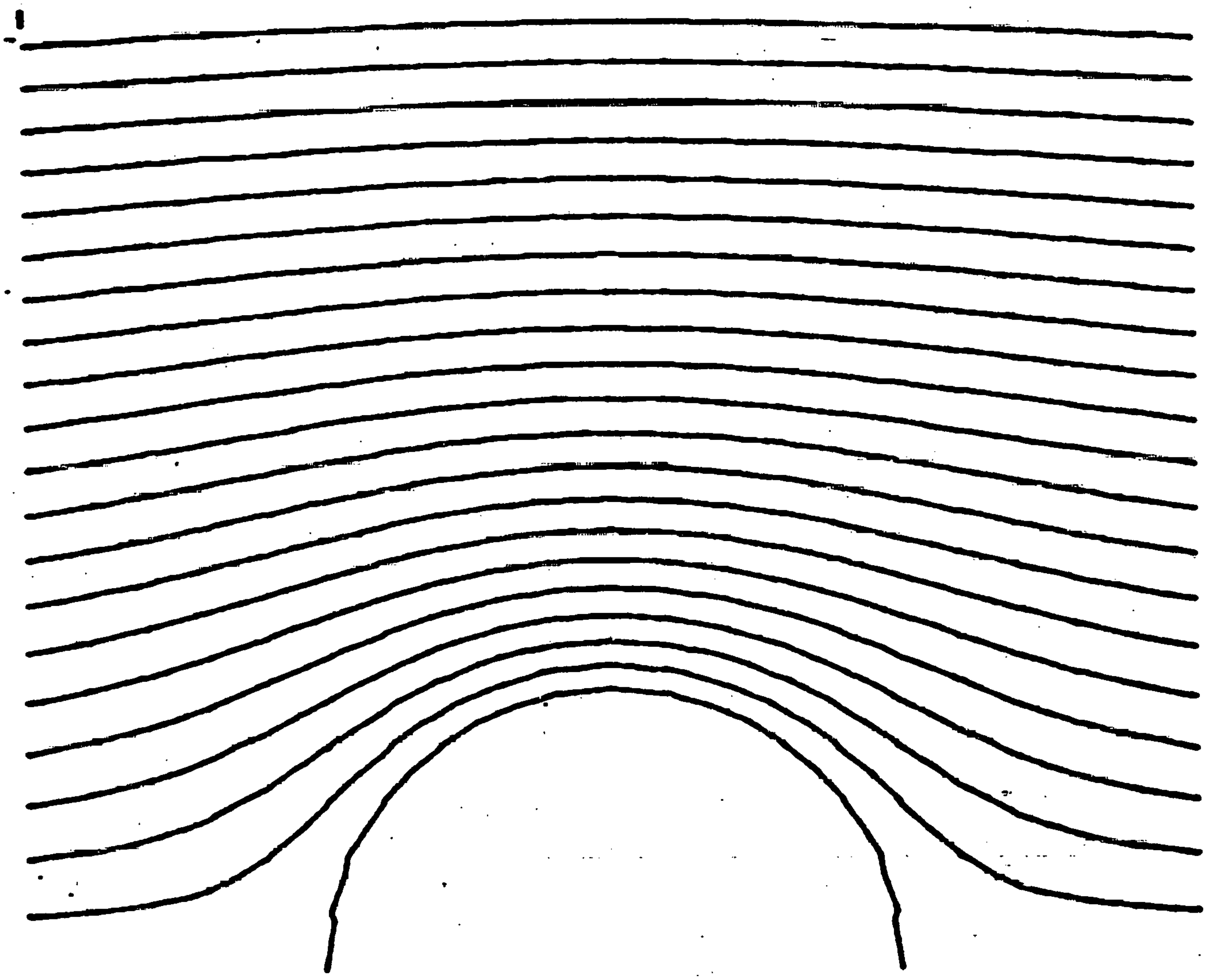
on the unit square

$$\{(x,y): 0 \le x \le 1,\ 0 \le y \le 1\}$$

subject to the boundary conditions:

$$u(0,y) = \exp(y) \qquad u(x,0) = \exp(x)$$

Figure 1. A two dimensional cross section of the streamlines associated with fluid flow around a cylindrical object.



u(1,y) = exp(1+y)   u(x,1) = exp(1+x)

The obvious analytic solution to this problem is

u(x,y) = exp(x+y)

To obtain an approximate numerical solution on an 8 panel by 8 panel mesh using the NCAR package, one would procede as follows in a typical Speakeasy session:

```
:_x=grid(0,1,1/8)
:_bda=exp(x)
:_bdb=exp(1+x)
:_solution=helmholtz(x,x,-2,0:0,bda,bdb,bda,bdb)
```

At this point, "solution" is a 9 by 9 array such that solution(i,j) approximates the value u(x(i),y(j)). To verify this, we might try the following:

```
:_exact=exp(colarray(x)+rowarray(x))
:_max(abs(solution-exact))
```

which will cause the maximum entry of the difference array to be printed. In the first line, notice how the high-wide arithmetic feature of Speakeasy can be very useful in defining a function of two variables which might appear on the right hand side of the Helmholtz equation.

The syntax of the call to this linkule for cartesian coordinates is

```
HELMHOLTZ(x,y,const,f:'boundary data':<PERIODIC>:
<HORIZONTAL|VERTICAL>)
```

where the first four parameters are required.
x is the array of x-coordinates for the mesh points.
y is the array of y-coordinates for the mesh points.
const is the scalar constant which appears in the Helmholtz equation given above.
f is the two-dimensional array which defines the right hand side of the Helmholtz equation.
'boundary data' is a list which

(i) contains 5 elements if the keyword PERIODIC is missing-- the first element is an integer mask, KBD, 1 to 4 digits in length (all of which are 0's or 1's) specifying the kind of boundary data which is being supplied in the next 4 elements of the 'boundary data' list. Basically, if the units digit of KBD is 0, the values of u(x,y) for x=a are known and given in the 2nd element of the 'boundary data' list. If the units digit of KBD is a 1, the values of the partial of u with respect to x are known for x=a (Neumann boundary condition) and given in the 2nd element of the list.

If the ten's digit of KBD is 0, the values of u(x,y) for x=b are known and given in the 3rd element of the list. etc. the hundred's digit corresponding to y=c and the thousand's digit corresponding to y=d.

(ii) contains 3 elements if the keyword PERIODIC and either HORIZONTAL or VERTICAL but not both are specified. The first element in the list is again the integer KBD described above and the next two elements are arrays which specify the boundary data (Dirichlet or Neumann). Note that the two digits of KBD corresponding to the unspecified boundary data will be ignored -- but should still be 0 or 1.

(iii) is empty is all the keywords PERIODIC, HORIZONTAL and VERTICAL are present, indicating periodic boundary conditions in both the horizontal and vertical directions.

When polar coordinates are used, the linkule solves the modified Helmholtz equation:

(1/r)(d/dr)(r*(du/dr)) + (1/r**2)(d/dtheta)(du/dtheta) + const*u = f(r,theta)

on the elementary polar region

$$\{(r,\text{theta}):\ a \le r \le b,\ c \le \text{theta} \le d\}$$

subject to either Dirichlet or Neumann boundary conditions. The syntax of the call to this linkule when using polar coordinates is:

```
HELMHOLTZ(r,theta,const,f:'boundary data':<PERIODIC>:POLAR)
```

where the first four positional parameters and the keyword POLAR are all required.

r is the array of r-coordinates for the mesh points.

theta is the array of theta-coordinates for the mesh points.

"const" is again the scalar constant in the modified Helmholtz equation.

f is the two-dimensional array which defines the right hand side.

'boundary data' is a list which

(i) contains 5 elements if the keyword parameter PERIODIC is not specified and the boundary data for r=a is given. The first of these elements is an integer mask with the same interpretation as the one for rectangular coordinates (see above). The next 4 elements are one-dimensional arrays which contain the boundary data for r=a, r=b, theta=c and theta=d, respectively (either the values of u or its normal derivative depending on the value of the integer mask).

(ii) contains 4 elements if the keyword parameter

PERIODIC is not specified and the boundary conditions for r=a=0 is left unspecified. (This can only be done when a=0.) In this case, we are of course assuming that Neumann boundary data is given for theta=c and theta=d, so the first 2 digits of the integer mask should be 1's. The integer mask is followed by 3 elements which are one-dimensional arrays giving the boundary data for r=b, theta=c and theta=d, respectively.

(iii) contains 3 elements if the keyword parameter PERIODIC is specified and the boundary conditions for r=a are given. In this case, the integer mask is followed by 2 arrays which give either the value of u or its normal derivative for r=a and r=b.

(iv) contains 2 elements if the keyword parameter PERIODIC is specified and the boundary conditions for r=a=0 are left unspecified. (again, this is only permitted when a=0.) in this case the integer mask is followed by a single one-dimensional array giving the boundary data for r=b. The value of the integer mask in this case should be either 0 or 10, since the units, hundreds and thousands digits of the mask are ignored.

In summary, here are the 8 basic ways to call the linkule HELMHOLTZ. The first four are cartesian coordinate variations and the last four are polar coordinate variations.

```
HELMHOLTZ(X,Y,CONST,F:KBD,BDA,BDB,BDC,BDD)
or
HELMHOLTZ(X,Y,CONST,F:KBD,BDA,BDB:PERIODIC:VERTICAL)
or
HELMHOLTZ(X,Y,CONST,F:KBD,BDC,BDD:PERIODIC:HORIZONTAL)
or
HELMHOLTZ(X,Y,CONST,F::PERIODIC:HORIZONTAL,VERTICAL)
or
HELMHOLTZ(R,THETA,CONST,F:KBD,BDA,BDB,BDC,BDD:POLAR)
or
HELMHOLTZ(R,THETA,CONST,F:KBD,BDA,BDB:PERIODIC:POLAR)
or
HELMHOLTZ(R,THETA,CONST,F:KBD,BDB,BDC,BDD:POLAR)
             if the first element of R is zero,
or
HELMHOLTZ(R,THETA,CONST,F:KBD,BDB:PERIODIC:POLAR)
        again only if the first element of R is zero
```

Now let us consider some actual examples of the use of HELMHOLTZ. In each of these examples, except for one, the solution is known and therefore we will be able to compare the accuracy of the numerical solution. It is important to realize that almost all the error in these examples is pure discretization error. That is to say, it results from a finite difference approxmation and not from roundoff. In the first example we will solve Laplace's equation on the domain:

$$D = \{(x,y) \mid -2 \le x,y \le 2\}$$

subject to the boundary conditions

$$u(-2,y) = u(2,y) = 4 - y^2$$
$$u(x,-2) = u(x,2) = x^2 - 4$$

The solution to this problem is obviously

$$u(x,y) = x^2 - y^2$$

As is well known the level curves of this function are the contour lines associated with a saddle. The Speakeasy session follows and the plot is found in Figure 2.

```
:_x=grid(-2,2,1/4)
:_bda=4-x**2
:_bdc=x**2-4
:_solution=helmholtz(x,x,0,0:0,bda,bda,bdc,bdc)
:_hmax=max(solution)
:_hmin=min(solution)
:_heights=grid(hmin,hmax,(hmax-hmin)/10)
:_graphics tek4012
:_contour solution heights
```

The following example demonstrates the use of HELMHOLTZ with periodic boundary conditions. We want to solve

(d/dx)(du/dx) + (d/dy)(du/dy) - 4*u =

(2-(4+pi**2/4)*x**2)*cos((pi/2)*(y+1))

on the rectangle 0 < x < 2, -1 < y < 3, subject to the boundary conditions

u(o,y) = 0   for -1 < y < 3
(du/dx)(2,y) = 4*cos((pi/2)*(y+1))
u is periodic in y.

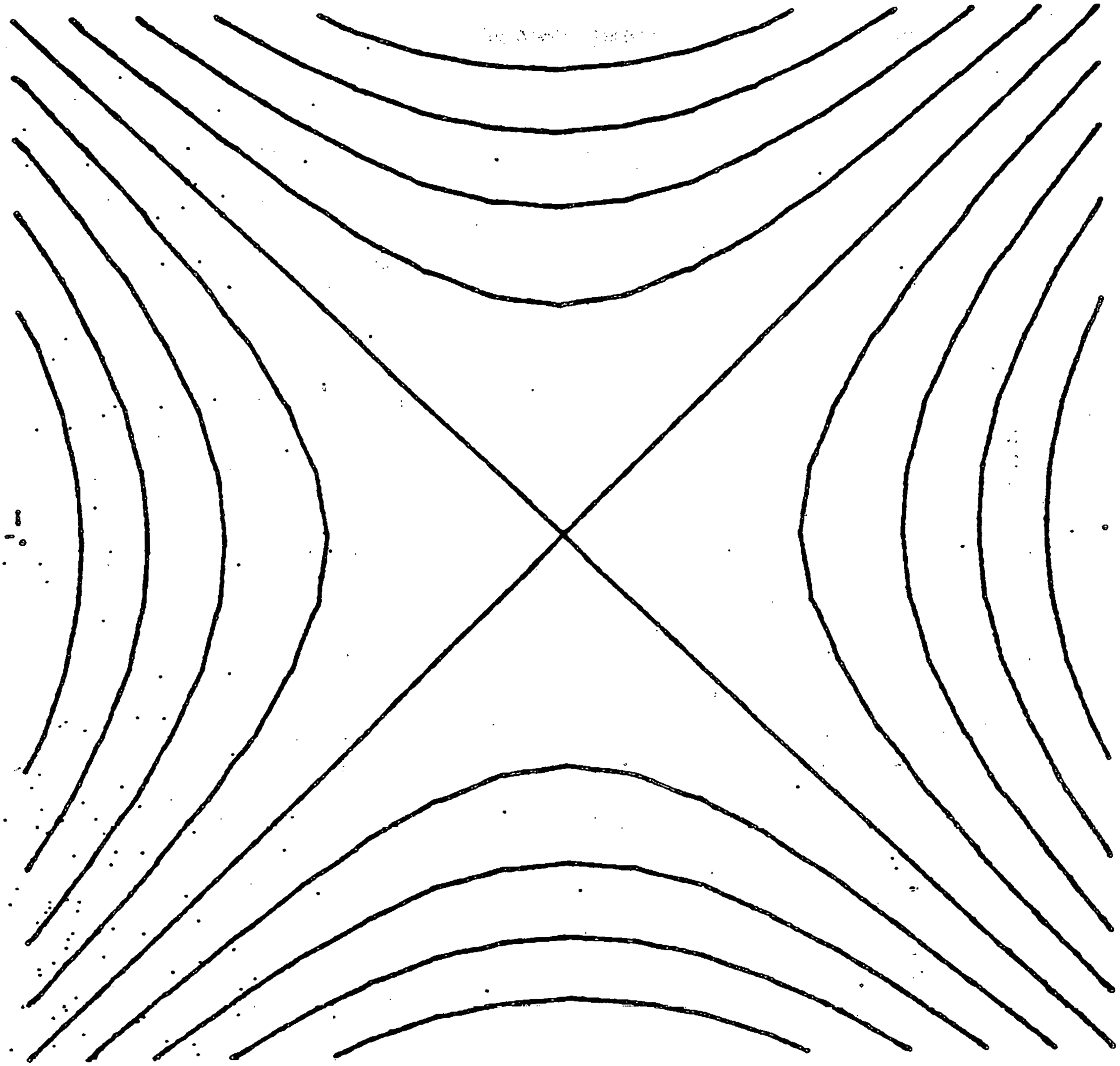
The exact solution to this problem is

u(x,y) = x**2*cos((pi/2)*(y+1)).

Here is the Speakeasy session which attacks this problem.

```
:_pi=4*atan(1)
```

Figure 2. The level curves of $u(x,y)=x^2 - y^2$, a solution to an elliptic boundary value problem found by HELMHOLTZ.



```
:_x=grid(0,2,.25)
:_y=grid(-1,3,.5)
:_fx=colarray(2-(4+pi**2/4)*x**2)
:_bda=array(9:)
:_bdb=4*cos((pi/2)*(y+1))
:_f=fx*rowarray(bdb/4)
:_solution=helmholtz(x,y,-4,f:0010,bda,bdb:periodic:vertical)
```

Now let us determine the discretization error.

```
:_exact = colarray(x**2)*rowarray(bdb/4)
:_max(abs(solution-exact))
MAX(ABS(SOLUTION-EXACT)) =  .054773
```

One might ask how the discretization error depends on the mesh size. For the finite difference approximation used in the NCAR package, it is easy to see that for a step size equal to h in both the x and y direction, the discretization error is on the order of h**2. We can easily demonstrate this fact by repeating the above example with a mesh which is twice as fine (in other words, the arrays x, y and bda should each have 17 elements). The resulting discretization error would be .013478, which is roughly one fourth the discretization error we had before. It should also be pointed out that, even though we have been printing out the absolute error in our examples, the relative error would be more meaningful for those elements of the solution matrix which are very large.

For our next example we will solve an equation involving polar coordinates. The pde is:

(1/r)*(d/dr)(r*(du/dr)) + (1/r**2)*(d/dtheta)(du/dtheta)

= 16*r**2

on the quarter disk 0 < r < 1, 0 < theta < pi/2, subject to the boundary conditions

u(1,theta) = 1-cos(4*theta), 0 ≤ theta ≤ pi/2

(du/dtheta)(r,0) = (du/dtheta)(r,pi/2) = 0, 0 < r < 1.

Note that u(0,theta) need not be specified in this case since the origin is just a vertex of the domain.

```
:_pi=4*atan(1)
:_r=grid(0,1,1/8)
:_theta=grid(0,pi/2,pi/16)
:_f=colarray(16*r**2)
:_bdb=1-cos(4*theta)
:_bdc=array(9:)
```

```
:_solution=helmholz(r,theta,0,f:1100,bdb,bdc,bdc:polar)
```

Again we examine the discretization error.

```
:_exact=colarray(r**4)*rowarray(bdb)
:_max(abs(solution-exact))
MAX(ABS(SOLUTION-EXACT)) =  .02351
```

In looking over the above examples, several limitations of the HELMHOLTZ linkule are immediately obvious.

(1) The domain must be either rectangular or an elementary polar region.
(2) The equations which can be solved involve no time derivatives.
(3) The domain is only two-dimensional.
(4) Certain more exotic boundary conditions are not allowed.

As for the first restriction, we have already seen a technique for handling more complicated regions, viz. conformal mapping. At present there are no built-in facilities in Speakeasy for mapping a mesh of points in one domain conformally onto another domain. Of course this topic has been a subject of considerable interest for numerical analysts, and several excellent algorithms are already available. In the case of Poisson's equation the conformal mapping approach will not work. Some techniques involving imbedding and splitting are discussed in [3]. Basically, the imbedding technique involves solving the equation on a rectangular domain which contains the more complicated region, while the splitting technique involves decomposing the more complicated region (say an L shaped region) into rectangles and piecing together the solutions for each rectangle.

The second restriction may also be avoided in certain cases. For example, suppose we are required to solve the wave equation

(Laplacian)u = (d/dt)(du/dt)

Furthermore, suppose we are looking for periodic solutions. Then it is natural to assume that u has the following form:

u(x,y,t) = exp(-iwt)*v(x,y)

Inserting this in the wave equation, we get

(Laplacian) v = w**2*v

which of course can be solved for v using HELMHOLTZ.

The third restriction can be removed using Fourier Transform techniques. The following example indicates how one should proceed. Suppose we wish to solve:

```
(Laplacian) u = -2*exp(x+y)*sin(2*z)

on 0 < x < 1, 0 < y < 1, 0 < z < 2*pi
```

subject to the following boundary conditions:

```
u(0,y,z) = exp(y)*sin(2*z)      u(1,y,z) = exp(1+y)*sin(2*z)
u(x,0,z) = exp(x)*sin(2*z)      u(x,1,z) = exp(1+x)*sin(2*z)
              u(x,y,0) = u(x,y,2*pi) = 0
```

If we assume that u(x,y,z)=v(x,y)*sin(2*z), then we can conclude quickly from the above pde that v must satisfy

```
(Laplacian)v - 4*v = -2*exp(x+y)
```

which can be solved using HELMHOLTZ. In general of course it is required to solve the more general Poisson equation

```
              (Laplacian)u = f(x,y,z) =
Sum(v(x,y)*cos(n*z)+w(x,y)*sin(n*z))
```

assuming f has a Fourier expansion. When used in conjunction with a Fast Fourier Transform, this method can be quite efficient. Although Speakeasy has some Fourier transform linkules, they do not use the Cooley-Tukey algorithm so the very precise solution of three-dimensional problems still lies in the future.

As for the fourth restriction noted above, there is often very little that can be done. For example if the given boundary data for an elliptic problem involves both the values of u and its normal derivative on some component of the boundary, then it is well known that the resulting problem is not well posed. In other words, if we continually refine the definition of the boundary values, making the mesh finer and finer, it is quite possible that the numerical solutions might actually diverge.

Despite the wide range of problems we have dealt with, there are still many partial differential equations whose format does not match the existing facilities. If we are reasonably careful in replacing the pde by a linear system, we can then use some of the linkules designed to handle linear problems. There are some disadvantages to using the existing linkules when the systems are large and sparse. Linkules like SIMEQ, SIMEQUAT, INVERSE and DETERMINANT fail to take account of the band structure or symmetry which is common in these large linear systems. The incorporation of the LINPACK subroutine package into Speakeasy will greatly enhance the possibility of either direct or iterative solutions to large linear systems. Some other advantages to the LINPACK routines include the avoidance of overflow-underflow problems to a much greater extent, and the ability to solve complex linear least squares problems.

In summary, it is clear that Speakeasy has already become a powerful tool for the solution of partial differential equations, at least for the very important class of elliptic boundary value problems. One of the great advantages of Speakeasy is its ability to incorporate the latest advances in the field of numerical analysis quickly and easily. Perhaps less obvious is the fact that once this is done, the language also becomes a valuable tool for the numerical analyst in the further development of algorithms. We have seen how the graphical capabilities of Speakeasy make possible the quick visualization of a numerical solution. With both the rapid advances in solutions to pde's and the rapid development of Speakeasy itself, we can expect this field to undergo some very big changes in the next few years.

REFERENCES

1. W. F. Ames, Numerical Methods for Partial Differential Equations, 2nd ed., Academic Press, N.Y. (1977).

2. G. Birkhoff, The Numerical Solution of Elliptic Equations, SIAM Regional Conference Series in Applied Mathematics, no. 1, (1972).

3. B. L. Buzbee, F. W. Door, J. A. George & G. H. Golub, "The Direct Solution of the Discrete Poisson Equation on Irregular Regions", SIAM Journal on Numerical Analysis, 8 (1971), pp. 722-736.

4. B. L. Buzbee, G. H. Golub & C. W. Nielson, "On Direct Methods for Solving Poisson's Equation", SIAM Journal on Numerical Analysis, 7 (1970), pp. 627-656.

5. R. V. Churchill, Complex Variables and Applications, McGraw-Hill, N.Y. (1960).

6. S. Cohen & S. C. Pieper, The SPEAKEASY-3 Reference Manual, Level Mu , ANL-8000, Rev. 2, (1977).

7. G. E. Forsythe & W. R. Wasow, Finite Difference Methods for Partial Differential Equations , John Wiley & Sons, N. Y. (1960).

8. P. Swarztrauber & R. Sweet, "The Direct Solution of the Discrete Poisson Equation on the Disk", SIAM Journal on Numerical Analysis, 10 (1973), pp. 900-907.

9. P. Swarztrauber & R. Sweet, "Efficient FORTRAN Subprograms for the Solution of Elliptic Partial Differential Equations", National Center for Atmospheric Research Technical Note, NCAR - TN/IA - 109, (1975).

10. R. Sweet, "Direct Methods for the Solution of Poisson's Equation on a Staggered Grid", Journal of Computational Physics, 12 (1973), pp. 422-428.

11. R. Sweet, "A Generalized Cyclic Reduction Algorithm", SIAM Journal on Numerical Analysis, 11 (1974), pp. 506-520.

******** THE SPEAKEASY COMPILER ********

by Thom Grace, Argonne Speakeasy Center

ABSTRACT.

The concepts of interpretting and compiling are introduced and discussed within the context of Speakeasy, with an introduction to the Speakeasy compiler. The linkule writing and function construction facilities are described, and an example applied to non-linear optimization is presented.

INTRODUCTION.

At the heart of the Speakeasy system is an interpretive processor. This means that each line of input is individually analyzed by the processor which, based on this analysis, performs the actions required. This is a common approach to the design of user-oriented software and command systems, as it allows great flexibility in syntax and quick execution of a user's command.

Many of the actions performed by the processor involve the invocation of linkules [1]: machine executable programs in the form of compiled code from a source language, typically Fortran. Data is passed to a linkule from the processor and used directly, no interpretation of code is necessary since the linkule has already been compiled. Thus, the execution of a typical Speakeasy statement will involve a syntax check of the statement, construction of parameter lists to be passed to the linkule(s) involved, and the access and execution of the linkule(s). This applies also to Speakeasy programs, which are collections of Speakeasy statements. In particular, each line is interpretted every time it is processed.

Interpretting a program has the effect of recompiling a line each time it is executed. Although negligible for programs of moderate length used occasionally, this interpretive overhead becomes slow and expensive in repeated executions. This arises when, for example, a program performs a great many operations (eg. a program with loops) or when a program must be executed a great many times. Operations such as numerical algorithms characteristically require a vast number of evaluations of a function. In these cases, the execution of Speakeasy programs is prohibitively costly.

In order to reduce this overhead, a new linkule may be written and added to the linkule libraries, to be accessed dynamically by the processor as a new Speakeasy word or by another linkule (eg. an optimization routine). The interpretive phase of execution then involves one line: the statement that utilizes the linkule. All subsequent operations, carried out by the linkule are at the machine-instruction level, and the interpretive cost is dramatically reduced.

The incorporation of a new linkule, however, requires in-depth knowledge of several areas concerning the supporting environment [1]: familiarity with another programming language (usually Fortran) and all the processes necessary for its processing (compiling, link-editing, etc.). Since the design philosophy of Speakeasy has been to make such detailed knowledge of extraneous information unnecessary [2], an automatic linkule writer or Speakeasy compiler has been developed.

Originally began in the context of differential equations [3], the compiler facilities now available can be utilized by a user, with only a minimum of extraneous knowledge, to construct personal linkules from a subset of the Speakeasy language. The heart of the compiler is a translator which acts on Speakeasy models (structures similar to programs) to produce Fortran source. The source is optionally compiled and added to the user's linkule library. The choice of Fortran as a target language for the translator allows more sophisticated users easy access to the intermediate stages of compilation, yet does not require any intervention from a casual or novice user.

It should be emphasized that the translator is evolutionary. Although not all Speakeasy statements can be translated, the class of such statements is growing. Even at the early stages of development, the translator-compiler has been used successfully to write linkules and to solve large systems of numerical functions expeditiously.

This document gives an overview of the translator and the translation process in Section A, followed by highlights of the linkule writing and function construction aspects in Sections B and C. Section D contains an example of the construction of a function to be used dynamically by a minimization linkule. Since this is intended to be an expository presentation of the features available, details are given separately in [4], available from the Speakeasy Center.

A. OVERVIEW OF THE TRANSLATOR.

The translator, invoked by a single Speakeasy statement, is used primarily to translate a model into Fortran and optionally to install the result in the user's linkule library. The compiled model may then be invoked by other Speakeasy linkules (eg. DEQ1E) or be used as a new Speakeasy word. A model is defined similarly to a Speakeasy program using the Speakeasy Editor, but using a subset of the extensive Speakeasy program language. Additionally, for use by other linkules, the model may utilize a special symbol set and class of reserved variables.

In general, models to be translated may contain executable and keyword statements. The executable statements may be of any of the following forms:

Comments;
Assignments;
Conditional (IF) statements;
FOR loops;
GOTO's (or GO TO's).

These are quite similar to Speakeasy program statements (eg. comments begin with '$', continuations begin with '&', statements may be labelled, etc.), but are restricted to involving only scalar expressions. Furthermore, identifiers may be at most six characters long, to conform with Fortran naming conventions. Most of the Speakeasy scalar functions may be used in a translatable model.

A model may access Speakeasy Named Storage to retrieve scalar values. This is done by means of a keyword statement. Other keyword statements are provided to allow the user to specify various parameters of the translation, such as the name of the resulting linkule, whether the result will be referenced by another linkule or be used as a new Speakeasy word, etc.

Unlike Speakeasy programs, all variables within a model are local in the sense that an access of a local variable will not alter any part of Named Storage. In this way, true linkules can be formed that depend only on values passed from the processor rather than on objects of specific names defined by the user.

By using options of the translator, the user may specify which portions of the compilation process are to be executed. This may range from a syntax check of the model by the translator, to saving the Fortran source in the user's KEEP library, to submitting the background job that installs the linkule.

Since the translator is in the developmental stages, restrictions and peculiarities of translation are evident. For example, the user must provide JCL if the translated model is to be placed into a linkule library. This is essentially the only knowledge outside the scope of Speakeasy itself that the user must have, and need only be applied once, since the JCL is obtained from the user's KEEP library. The restriction of expressions to scalars, disallowing more complex structures, may be relaxed in future releases of the translator, as efficient and effective methods of analyzing the rich and powerful Speakeasy vocabulary are developed.

B. AUTOMATIC LINKULE WRITER.

The translator may be used to construct a new linkule from a user's model. After translation, the linkule may be used as a new Speakeasy word, just as if the linkule had been written independently in Fortran. The translator provides facilities specifically designed for the automatic construction of linkules. By means of a single option, various keywords are available to facilitate the design of such a linkule.

Although computation in a model is restricted to scalar operations, a linkule translated from a model will be interfaced with the Speakeasy Highwide conventions. In this way, the new Speakeasy word will be usable with any class of variable (scalar, vector, array, etc.) and return a value of corresponding structure. The translator will insert the appropriate Highwide subroutine to enable such an interface.

Since many calculations can be performed only in a certain domain, the translator recognizes a keyword statement which will assure that the arguments to the resulting linkule satisfy specified restrictions. In addition, arguments may be restricted to the integers or allowed to be any real number. This will also override the Fortran naming conventions for local variables passed by the processor. Using these facilities, the translated model can detect improper arguments and respond with an appropriate diagnostic message at execution time.

By using the translator as an automatic linkule writer, virtually any computable scalar function of up to 30 variables may be made into a linkule, with the added power of automatic parameter checking, access to Speakeasy Named Storage, and the Speakeasy Highwide conventions. Thus, even a casual user may design and implement personal linkules of great flexibility.

C. DYNAMIC FUNCTION CONSTRUCTION.

The Speakeasy language has well-developed capabilities for the manipulation of numerical structures. However, the discrete nature of Speakeasy objects makes them unusable by processes which may request functional information at many arbitrary points. Such processes include, for example, optimization algorithms and numerical solution of differential equations. These often require vast numbers of precise evaluations not obtainable from a vector or array of fixed points. The translator may be used to construct a function that can be referenced dynamically by other Speakeasy linkules, so that efficient evaluation of a numerical function at any point is possible.

Linkules have been written that solve systems of ordinary differential equations, minimize real-valued fuctions, and minimize the sum of squares of vector-valued functions. In each case, the function is a translated model installed in the user's linkule library. Keywords are provided that indicate which type of function the model is to define.

In addition, a model to be translated as a dynamic function utilizes a set of reserved variables together with a special symbol set. These reserved variables and symbols define quantities to the model from which the function will be calculated. Although expressions within the model are restricted to scalars, certain reserved variables may be regarded as vectors. In this case, expressions involve elements of these variables, accessed by using the special symbol set.

For example, a typical model defining a system of ordinary differential equations will calculate the value of n derivatives depending on n dependent variables, the independent variable, and any values retrieved from Named Storage. The n derivatives are given by reserved variables Y1', Y2', ..., Yn'; the dependent variables by Y1, Y2, ..., Yn; and the independent variable by X. The number of equations in the system is also available as the reserved variable N.

D. AN OPTIMIZATION EXAMPLE.

This section presents an example of the use of the translator to construct a function to be minimized. The example is in two parts. In the first part, the model (describing a simple quadratic function) is defined. Guided by the HELP documents, the model is checked for syntax errors, then translated and installed in the linkules library. The compiled function is referenced in the minimization linkule and the minimum is found. In the second part, the model is changed to access Named Storage for a parameter, recompiled, and again minimized.

THE SPEAKEASY COMPILER

```
:_$
:_$  FIRST, WE DEFINE THE MODEL WE'LL TRANSLATE.
:_$  NOTE THAT IN A MODEL TO BE OPTIMIZED, THE
:_$  KEYWORD STATEMENT 'MINIMIZE' MUST APPEAR FIRST.
:_$  THE FUNCTION VALUE IS THE RESERVED VARIABLE F,
:_$  INDEPENDENT VARIABLE VARIABLES ARE X1, X2, ..., XN;
:_$  AND N IS THE DIMENSION OF THE SYSTEM.
:_$
:_MODEL FUNCTION
EDIT INPUT MODE
1 MODEL
2 $
3 $  THIS IS A SIMPLE QUADRATIC. THE
4 $  MINIMUM IS AT X1 = X2 = ... = 0.
5 $
6 MINIMIZE
7 F = 0
8 FOR J = 1,N
9   F = F + XJ**2
10 ENDLOOP J
11 F = F/2
12 END
MODEL FUNCTION IS NOW DEFINED
MANUAL MODE

:_HELP TRANSLATE
   TRANSLATE(MNAME:options) translates a model MNAME into a Fortran
 subroutine or function and optionally supplies the highwide con-
 vention for a linkule, and/or submits a batch job to install the
 module as a linkule in the user's KEEP library.  One or more of
 the following options may be specified:
   LIST/NOLIST         LIST will print out on the terminal a copy of the
                       subroutine or function, but not the linkule
                       source if HIWIDE is requested.  NOLIST will only
                       print out error messages.
   SAVE/NOSAVE         Saves the linkule and/or subroutine in the user's
                       KEEP library as member name, where name is either
                       specified in the model, or defaults to the
                       model name
   HIGHWIDE/NOHIGHWIDE   processes the highwide convention to make a
         or              linkule.
   HIWIDE/NOHIWIDE
   BATCH/NOBATCH       Submits a batch job to compile the new linkule.

 The default options are LIST,NOSAVE,NOBATCH,NOHIGHWIDE.
 These can be changed by the user with the command SETTRANSLATE.
 Note that a linkule will be installed only if BATCH is used
 and a member JCL exists in the user's or the system KEEP library.
 TRANSLATE should be used only under TSO.
:_
```

THE SPEAKEASY COMPILER

```
:_$
:_$  NOW WE WILL TRANSLATE THE MODEL TO CHECK
:_$  FOR ERRORS AND SEE THE FORTRAN SOURCE...
:_$
:_TRANSLATE FUNCTION
C
C  THIS IS A SIMPLE QUADRATIC. THE
C  MINIMUM IS AT X1 = X2 = ... = 0.
C
      SUBROUTINE FUNCTI(N,X,F,ISPMER,ISPARM)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION X(1),ISPARM(1)
      IF(ISPARM(1) .EQ. 1) RETURN
      F = 0
      J = 1
      GOTO   14
   11 IF((1)-(N))    12,    12,    13
   12 J=J+1
      IF(J.GT.N)GOTO   15
      GOTO   14
   13 J=(-1)+J
      IF(J.LT.(N))GOTO   15
   14 CONTINUE
        F = F + X(J)**2
      GOTO   11
   15 CONTINUE
      F = F/2
      RETURN
      END
:_
```

THE SPEAKEASY COMPILER

```
:_$
:_$  GOOD...NO ERRORS. NOW, TO INSTALL
:_$  THE FUNCTION IN OUR LINKULE LIBRARY
:_$
:_TRANSLATE FUNCTION:SAVE BATCH NOLIST
JOB MINI     SUBMITTED

:_HELP MINIMIZE
MINIMIZE(MNAME,START,END) minimizes the nonlinear function MNAME.
MINIMIZE(MNAME,START,END) will seek a local minimum of the nonlinear
function defined by previously TRANSLATE'd model MNAME.
START (real vector/array) is the starting point for the search.
END (optional) will be defined, if the search is successful, to be a
real array the same size as START containing the final estimate of the
solution.
MINIMIZE returns a real scalar that is the function value at END.
:_$
:_$  NOW WE'LL MINIMIZE THE FUNCTION,
:_$  STARTING AT A POINT A LONG WAYS AWAY...
:_$
:_START = (10,10,10)
:_MINIMIZE(FUNCTION,START)
MINIMIZE(FUNCTION,START) =  0
:_
```

```
:_$
:_$  NOW WE HAVE CHANGED THE FUNCTION...
:_$
:_EDIT FUNCTION
EDIT COMMAND MODE
:%L
EDITING FUNCTION
  1 MODEL
  2 $  THIS IS THE SAME QUADRATIC, EXCEPT
  3 $  THAT THE MINIMUM IS NOW THE VALUE OF
  4 $   THE PARAMETER 'HEIGHT'. THE PARAMETERS
  5 $   KEYWORD GETS A VALUE FROM THE ALLOCATOR.
  6 $
  7 MINIMIZE
  8 PARAMETERS = (HEIGHT)
  9 F = 0
 10 FOR J = 1,N
 11    F = F + XJ**2
 12 ENDLOOP J
*13 F = F/2 + HEIGHT
EDIT COMMAND MODE
:%END
MANUAL MODE
:_$
:_$ FIRST, WE RETRANSLATE THE FUNCTION
:_$
:_TRANSLATE FUNCTION:NOLIST SAVE BATCH
JOB MINI     SUBMITTED

:_$  NOW WE CAN MINIMIZE THE NEW FUNCTION,
:_$  AFTER DEFINING THE SCALAR IT NEEDS
:_$
:_HEIGHT = -10
:_START = (10,10,20,30)
:_MINIMIZE(FUNCTION,START)
MINIMIZE(FUNCTION,START) =  -10
:_
```

REFERENCES.

[1] Cohen, S., and Peiper, S., The Speakeasy-3 Reference Manual. Argonne National Laboratory Report, ANL-8000.

[2] Cohen, S., Speakeasy - A Window Into a Computer. AFIPS Conference Proceedings, Vol. 45.

[3] Grace, T., Solutions of Ordinary Differential Equations Using EPISODE Through Speakeasy. (unpublished), 1977.

[4] Kane, M.L., and Grace, T., The Automatic Linkule Writer. (unpublished), 1977.