Speakeasy—A window into a computer*

by STAN COHEN
Argonne National Laboratory
Argonne, Illinois

ABSTRACT

Speakeasy is a system that enables a user to harness the power of a computer as a tool for problem solving. This paper describes the structure of that system and demonstrates some of its capabilities. The examples show some of the power that results when relatively straightforward graphical facilities are added to this general purpose system.

INTRODUCTION

A modern computer complex is an extremely fast information processor with a vast amount of stored information available to it. Machines and their associated software are becoming ever faster, ever larger and ever more complex. This growth in capability brings with it increasing responsibility to provide the normal user with adequate access to available information. Unfortunately a sense of isolation from advances in computer science is common in many disciplines and results in the people most in need of the capabilities provided by a computer being forced to rely on secondary sources for solutions to their problems or to ignore advances entirely and to program using first generation computer techniques.

One answer to this problem of information exchange can be found within the structure of the computer itself. A generalized modular system that is designed to access and operate with libraries of stored information can make advanced algorithms and general information available to users in a sensible way. Properly structured, such a system acts as an interface between user and the computer, matching the user's needs to capabilities available to him. Inadequacies in such a system are answered by adding new capabilities to satisfy existing needs. A system designed for growth provides the environment to answer the needs of users; it can at the same time provide researchers in computational science with an audience for their work.

This paper describes a system of this type that has been in use and under development for over a decade. Originally designed as a tool for research scientists, the generalized and easily used structure of this processor has now led to its acceptance by a large and varied user community. Although Speakeasy[1-4] has been in existence for many years, it is the widespread availability of time-sharing systems such a TSO and VM/CMS that accounts for the recent rapid growth in acceptance. The Speakeasy processor is currently available for use on IBM 360 or 370 computers operating under OS, TSO, VM/CMS or MTS. It has been translated for use on FACOM-230 and PDP-10 computers. Translations to several other computer systems are now being carried out.

There is an active user group involving some 70 different installations and over a thousand users. The user communities already include physical scientists, engineers, econometricians, government agencies and universities. The growth in the capabilities of Speakeasy in recent years is partially in response to the diverse nature of its user group and partially due to contributions from it.

This paper is intended as a general description of the Speakeasy system. It is divided into two major sections. The first describes the structure of the system. The general form of this system should be of interest to a computer scientist since it is here that the extensibility and growth capabilities reside. The second section is a sampling of the capabilities found in actual operational versions of the system. The features illustrated are only a few of those that make this system valuable to its users. They show that a truly user oriented system can at the same time be a powerful one.

THE OVERALL DESCRIPTION

The structure of the Speakeasy system is illustrated in Figure 1. The system consists of a language processor that interprets the requests of the user and breaks them down into basic syntactical components. The language structure is a straightforward one modeled on conventional mathematics. A scratch-pad storage facility available to the processor enables it to maintain transient information for the duration of the

* Work performed under the auspices of the U.S. Energy Research and Development Administration.

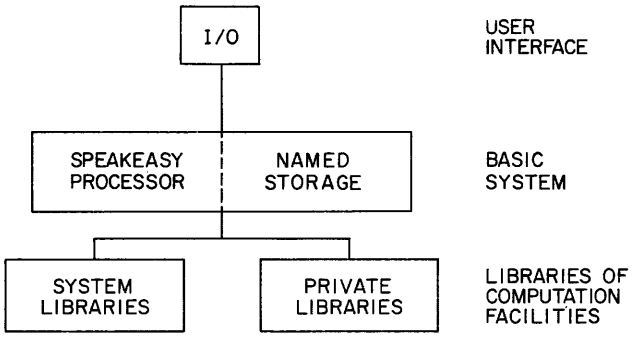

Figure 1—A schematic view of the Speakeasy system. The bulk of the computational facilities are contained in the modules that make up the libraries attached to the system

particular application. The processor and this scratch pad provide the mechanism for a user to define and address structured objects that represent transient information in the tasks being assigned to the system.

Libraries of pre-compiled routines are attached to the processor to provide the system with its working vocabulary. When a specific operator is requested of the system, all of the libraries attached to the system are searched. If the requested operator is found, it is dynamically linked to the system so that it may perform its function. In most cases such operations involve operands previously defined and saved in the transient scratch-pad area. New results are also saved there and are therefore available for later use by other operations.

This operator-operand formulation with a wide variety of operators on structured objects is the heart of the Speakeasy system. The ability to define structured objects such as arrays, vectors and matrices makes it meaningful to develop a large set of operators tailored to specific data structures. The use of attached libraries makes it possible to provide an ever increasing set of such operators without creating a system of immense size and complexity.

The modularity of this system, the use of libraries and the simplicity of its control language combine to provide a powerful system that is easily used and capable of sustained growth.

Design of the language

The Speakeasy system can be viewed as a repository. The communication language for the system (also called Speakeasy) is intended to make this repository usable. In designing the language and its syntactical rules the foremost considerations were naturalness, ease of use and tolerance for trivial mistakes. The basic specification for the language is a simple one. If a request by a user is unambiguous and looks correct to him then it should be accepted by the system.

This design philosophy is somewhat different from that of normal computer languages that provides features to enable the user to exploit the capabilities of the machinery itself. Such languages are specified so that a large number of decisions that relate only indirectly to the calculation must be made by the programmer. To give just a few examples, fixed point versus floating numbers, dimension statements for quantities that occur only as intermediate results, and input and output formats are concepts only indirectly related to the statement of the problem itself. Frequently such relatively trivial specifications constitute the bulk of a program; the parts related to the calculation are but a small part of the material written by the user. It is the volume of this extraneous information that accounts for many of the errors encountered in conventional programming.

In Speakeasy most of such decisions are considered to be part of the internal functions of the system. The user formulates his problem in a brief and natural manner, the system translates this formulation into an executable form, relieving the user of as many trivial decisions as possible. In fact, the Speakeasy language specifications bear little relationship to the structure of a computer and only very indirectly reflect the form of calculation as executed in the computer.

The Speakeasy language is designed to operate with structured objects. Scalars, arrays of numbers, mathematical vectors and true matrices are among the many classes of structured objects that a user may define and use in his calculation. Implicit algebraic rules that are class dependent provide a variety of tools for formulating calculations while operating on defined objects as single entities. Complete matrix algebra is provided in a natural notation. This combined with the array processing capabilities, enables one to write a directive program that does not involve the loops and branches that make up most of the computational steps in conventional programs.

The availability of structured objects means that an operator-operand language with great richness can be developed (one of the limitations of usual languages is that the scalar structure of the languages limit functional values to scalars). Each of the operators in Speakeasy is designed as a self-contained module whose operations are dependent upon the structure of its operands. In this way all of the decisions normally necessary in invoking routines in conventional programming techniques are internally contained in the operators of the system. Each operator is clearly definable in terms of what it does and contains many checks to see that it is being properly applied. By placing decision processes such as these within the system the user is relieved of most of the mundane parts of programming. The user can therefore concentrate his effort on formulating the overall logic of his problem and still be assured that the large number of trivial decisions are being properly made.

The overall language designed around an operator-operand concept combined with conventional algebraic tools is powerful, easily learned and easily used. It is logically complete and is extensible in both the types of structure supported and the operators available for manipulating objects.

The transient scratch-pad—named storage

A special dynamic storage scheme is an important component of the Speakeasy system. This storage facility provides the mechanism for defining and retrieving the structured objects discussed previously. In this scheme each defined object has associated with it a complete description of its structure along with particular values for its elements. Most importantly, the object and its descriptive information can be referenced by name. The name is all that is needed to locate any defined object, to determine everything about its structure and to use it in a calculation.

Named Storage[5] was developed for use with calculations in nuclear physics.[1] It is this storage technique that led to the development of Speakeasy. Individual modules of the Speakeasy system (the operators available to the language processor) are designed to operate on objects defined in Named Storage and to produce new objects there. The extreme modularity of the Speakeasy system is a consequence of this rather straightforward storage scheme.

Each defined object in Named Storage has descriptive information appended to it which designates its class, the type of data in its elements and its dimensionality. The allowed ranges for these designators has been made large so that new data types, new classes, etc. can be supported in later developments. Since no computational capabilities are contained in the storage package it is possible to extend the capabilities of Speakeasy entirely through additions to the operators attached to the system.

The libraries—linkules

In most computer systems each new addition brings with it increases in complexity, more overhead and a larger processor. Even with a very modular system this remains the case. The benefit of each addition must be weighed against the consequence that it will have on the overall system performance. It is therefore unlikely that a feature of great benefit to only a few users will ever be adopted if it degrades the system's use for others.

The library orientation of the Speakeasy system solves this problem in a particularly interesting way. The use of attached libraries containing operators such as those described earlier provides a system with easily expandable capability in a way that neither constrains the growth nor increases the basic structure of the system. Operators that are dynamically attached to the processor, called linkules,[3] provide the means for adding any desired capability for those who need it, without others even being aware of its existence. Private libraries of linkules can be used to give each user community a processor tailored to its own needs.

The freedom provided by a library oriented system is obvious. Growth by the accumulation of new information is automatic. Maintenance is particularly easy since replacement, modification and additions are made on members of libraries and not on the processor itself.

Each new operator added to the system brings with it capabilities that are enhanced by the presence of other operators in the system. A system such as Speakeasy thus reaches a critical stage where the interrelationship between the operators begins to make itself felt. After this threshold is reached the growth of the capabilities of the system are often based on finding new ways to interconnect existing facilities. Speakeasy has passed that threshold and it is no longer possible to clearly define the limits of the capabilities of the existing system.

The user interface

The user's first introduction to a system such as Speakeasy can be through a variety of devices. It is important that a system such as Speakeasy functions equally well for each type of device and that it be able to exploit the particular capabilities of each. In this system this is accomplished by isolating all input and output to specific components of the system, designing interfaces to classes of devices, and providing other general facilities that can be selectively made available to users of specific devices on demand.

This ability to interface correctly to every type of terminal in a time sharing environment is of particular psychological importance since the intent is to make this system appear natural to the casual user. If the introductory session with the processor is spent describing how a particular device is used or why a multitude of peculiar keys are used then it is unlikely that the user will ever be convinced of the naturalness of the language.

The currently operational versions of Speakeasy can be used with card readers, printers, plotters, ASCII terminals, IBM 2741's, Tektronix 4000 series graphics terminals, or any combinations of these devices. In each case the adaptation is one in which the device is natural to use and one where all of its hardware capabilities can be exploited.

EXAMPLES OF SPEAKEASY

Speakeasy is best demonstrated in a time sharing environment, for it is there that the ease of use and

natural form of the language becomes most obvious. Complete novices can begin to carry out calculations with only a few minutes instruction. Built in documentation and teaching aids enable such users to learn about the wide variety of capabilities within the system and to soon begin to utilize the system as a tool in their daily work. As their demands grow they find within the system a wide variety of facilities that can be merged together into an extremely powerful tool.

It is obviously not possible in this short paper to convey the feeling associated with an interactive system. Neither is it possible to demonstrate more than a few of the capabilities of this large system. No attempt is made, for instance, to illustrate the use of stored programs or of the editor that is available to construct such programs. Neither is there any discussion of the means of storing and retrieving information. The examples given here must therefore be viewed as a demonstration of only a few of the available features and perhaps as a tantalizing taste of what can be found within the system.

Examples of the arithmetic capabilities

The bulk of the computational capabilities in this system are related to numerics. Many of these facilities were provided by simply designing interfaces to existing Fortran coded mathematical subroutines and functions. Every attempt is made to find the best available technique and to provide as sensible and as general a facility for the users as is possible. It is in this way that the novice computer user is able to make use of advanced computational techniques.*

The system contains the basic operations of numerical calculus, statistics, matrix algebra and most of the usual special functions. Many of the operations are available for both real and complex numbers.

Figure 2 shows the processor being used as a desk calculator. The two characters :_ are the prompt symbols for the manual mode of Speakeasy. The user's request is typed on this remainder of the line. A carriage return indicates that the request is complete and that processing should take place. If the request elicits a response it is almost immediate and it is followed by a new prompt. A special implied print convention echoes simple input requests and prints the result. This result is also given the name ANSWER so that it may be used in later calculations. Users may define variables by straightforward assignment statements such as those illustrated. Standard mathematics notation is clearly demonstrated by the use of absolute value signs and factorials, and by the normal hierarchy for the evaluation of involved expressions.

* For example, EISPACK is a general purpose eigen-analysis package. In Speakeasy the words EIGENVALUE and EIGENVECTOR are used to invoke this package. Decisions relating to the particular choice of path through EISPACK are made by the linkule interface. The novice user is thus provided with a powerful computational package in a simple way.

```
:_2+2
2+2 =  4
:_3*SQRT(3)
3*SQRT(3) =  5.1962
:_3**3
3**3 =  27
:_ANSWER+20
ANSWER+20 =  47
:_X=9 ; Y=18
:_X*Y-5
X*Y-5 =  157
:_4*X+|3*5!-7!|
4*X+|3*5!-7!| =  4716
:_(3*LOG(4))*(SQRT(3.5*8/6.555)+17.6)
(3*LOG(4))*(SQRT(3.5*8/6.555)+17.6) =  81.792
:_ACOS(-1)
ACOS(-1) =  3.1416
:_SIGNIFICANCE 15
:_ANSWER
ANSWER =  3.14159265358979
:_SIGNIFICANCE 5
:_ANGLES IN DEGREES
:_RATIONALIZE
:_2/3+1/7*SIN(30)
2/3+1/7*SIN(30) =  31/42
:_(-32)**(-3/5)
(-32)**(-3/5) = -1/8
:_(-2)**(3/2)
IN LINE " (-2)**(3/2)  "  ENTERED COMPLEX DOMAIN.
:_DOMAIN COMPLEX
:_RETRY
(-2)**(3/2)      = -2.8284I
:_(2-3I)**2
(2-3I)**2 = -5-12I
:_SQRT(3-4I)
SQRT(3-4I) =  2-1I
:_ANSWER**2
ANSWER**2 =  3-4I
```

Figure 2—A sample of the use of Speakeasy in the manual or desk calculator mode of operation. Note the natural form of the directive language

Figure 3 illustrates the definition and use of arrays of numbers. It is of course not possible to show more than a few of the available capabilities. The compactness and the direct form of the language should be apparent. The rather natural and readable form of output should also be noted. It should be realized that this output form is the default form. Further tailoring is of course possible.

Figure 4 shows the means for defining and operating with matrices. The computational power that is hidden within these few statements should be apparent to those familiar with this field of numerical analysis. Some of the available tailoring commands are also illustrated here to demonstrate the flexibility of the system.

In Speakeasy two dimensional arrays and matrices belong to different families. The algebraic rules for operating on such objects are different. In Figure 5 some of the operations shown in the previous figure are repeated using arrays instead of matrices. The differences are apparent.

```
:_ANGLES IN DEGREES
:_X=GRID(0,360,15)
:_SINE=SIN(X)
:_COSINE=COS(X)
:_TABULATE X SINE COSINE

   X      SINE      COSINE          X      SINE      COSINE

  0      0          1             195   -.25882   -.96593
  15     .25882     .96593        210   -1/2      -.86603
  30     1/2        .86603        225   -.70711   -.70711
  45     .70711     .70711        240   -.86603   -1/2
  60     .86603     1/2           255   -.96593   -.25882
  75     .96593     .25882        270   -1         0
  90     1         -0             285   -.96593    .25882
  105    .96593    -.25882        300   -.86603    1/2
  120    .86603    -1/2           315   -.70711    .70711
  135    .70711    -.70711        330   -1/2       .86603
  150    1/2       -.86603        345   -.25882    .96593
  165    .25882    -.96593        360    0         1
  180   -0         -1

:_X=7,3,4,SQRT(3),1,8
:_AVERAGE(X)
AVERAGE(X) =  4.122
:_CUMSUM(X)

CUMSUM(X) (A 6 COMPONENT ARRAY)
  7        10        14        15.732   16.732   24.732

:_ORDERED(X)

ORDERED(X) (A 6 COMPONENT ARRAY)
  1        1.7321   3         4        7        8
```

Figure 3—The compact form of the Speakeasy language is shown here. There are a large number of functions such as CUMSUM and AVERAGE. The explicit looping so common in other languages is rarely used in Speakeasy

Documentation

Any computer system must be properly documented in order that it be usable. Documentation for a major system is not an easy task. If approached solely by conventional means it would be even more difficult in a system like Speakeasy that is designed for growth. Any printed documentation would be outdated before it was published. Fortunately, the system itself is capable of providing facilities that not only supply the documentation but do so in a way that is guaranteed to be self-sustained.

A library of documents referred to as the Speakeasy HELP documents is attached to the system. This library is addressed by a linkule in the normal library. Each word, concept or facility available in the system is described in a member of the HELP document library. The library is designed as a tree structure so that a user probing the library in an interactive session is led quickly to the specific concept of interest.

Figures 6-8 illustrate the HELP documents and show how the tree structure provides quick access to information about a specific operation. Each of the over 500 documents in this library is available in a similar way. The intent of these short documents is to inform the user of the operational definition of each word. No attempt is made to explain specifics of the techniques used. A second library containing larger documents

```
:_X=MATRIX(3,3:1,2,3,5,2,7,3,1,6)
:_X

  X (A 3 BY 3 MATRIX)
  1  2  3
  5  2  7
  3  1  6

:_1/X

1/X (A 3 BY 3 MATRIX)
 -5/16   9/16 -1/2
  9/16   3/16 -1/2
  1/16  -5/16  1/2

:_NORATIONALIZE
:_1/X

1/X (A 3 BY 3 MATRIX)
 -.3125   .5625 -.5
  .5625   .1875 -.5
  .0625  -.3125  .5

:_X*ANSWER

X*ANSWER (A 3 BY 3 MATRIX)
  1             6.9389E-17   0
  0             1            0
 -2.7756E-17    0            1

:_PRINTNULL(1E-10)
:_ANSWER

ANSWER (A 3 BY 3 MATRIX)
  1  0  0
  0  1  0
  0  0  1

:_EIGENVALUES(X)

EIGENVALUES(X) (A VECTOR WITH 3 COMPONENTS)
 -1.5484   1.0928   9.4556
```

Figure 4—Built-in matrix algebra is available in a natural way

is available for that purpose and is equally easily accessed.

A complete teaching facility is also built into the library of the standard system. A series of tutorial sessions provide a novice with step by step instructions on the use of special facilities. One of these sessions is an introduction to Speakeasy graphics. A few pages of this are shown in Figure 9.

Interactive graphics

A graphics display terminal in a time-sharing environment greatly increases the potentials for true interactive computing. This is particularly evident for the exploratory types of computation that are so common in many problem solving and research environments. Because large amounts of information can be rapidly and sensibly displayed it is possible for the user to quickly interpret the effects of various choices.

```
:_X=ARRAY(3,3:1,2,3,5,2,7,3,1,6)
:_X

  X (A 3 BY 3 ARRAY)
  1   2   3
  5   2   7
  3   1   6

:_1/X

1/X (A 3 BY 3 ARRAY)
  1         .5        .33333
  .2        .5        .14286
  .33333    1         .16667

:_X*ANSWER

X*ANSWER (A 3 BY 3 ARRAY)
  1   1   1
  1   1   1
  1   1   1

:_SUMROWS(X)

SUMROWS(X) (A 3 COMPONENT ARRAY)
  6    14   10
```

Figure 5—The algebraic operations in Speakeasy are class dependent. The array algebra in these examples should be contrasted to the algebra shown in Figure 4

```
:_HELP
  HELP explains how to use Speakeasy.
QUIT          is the command to leave Speakeasy.
INOUT         lists words dealing with input and output.
MATH          lists mathmatical functions.
PHYSICS       are functions of interest primarily to physicists.
STATISTICS    lists words related to statistics.
OBJECTS       lists words dealing with structured objects.
PROGRAMS      lists words used in writing programs.
DATATYPES     lists words about types of data used in Speakeasy.
MISCELLANEOUS lists other Speakeasy words.
BUGS          gives the known errors and stage of correction.
NEWS          gives recent modifications and new features.
TUTORIAL      tells how to use the Speakeasy tutorial.
VOCABULARY    lists all the words in Speakeasy.
HELP XX       gives an explanation of the word XX.
                XX is any vocabulary word.


:_HELP GRAPHICS
  GRAPHICS are words which deal with graphical output.
CALCWORDS     lists words used to plot on a CALCOMP plotter.
GRAPHWORDS    lists graphics words usable with most graphic devices.
OLDTEK        lists words used with the old Tektronix package.
PRINTGRAPHS   lists words used to plot on a line printer or a
                non-graphics terminal.
TEKWORDS      lists words used to plot on a Tektronix terminal.

  Note:  An attempt is being made to, wherever possible, make graphics
words independent of the graphic output device being used to create
the plot.  At present only a few meet that goal (type HELP GRAPHWORDS
for the list).  In the future, the words listed in the TEKWORDS Help
Document will form the basis of the new graphics vocabulary.

  To obtain a description of a given word XXX, enter
 HELP XXX
```

Figure 6—Every word known to the Speakeasy processor is described by a short HELP document. These documents are arranged in a tree structure, part of which is shown here

```
:_HELP GRAPHWORDS
  GRAPHWORDS lists graphics words usable with most graphic devices.
ADDGRAPH      adds a graph to a previous graph.
GRAPH         plots a graph.


:_HELP GRAPH
  GRAPH(Y:X) plots Y as a function of X.
  X and Y are one dimensional arrays of equal length.  The scale, if
not defined, is computed automatically, the curve is plotted, and the
axes are drawn and numbered.  The GRAPH command may be used alone or
else combined with other graphics words to tailor the graph format
to the user's requirements.
  GRAPH(Y1,Y2, ... YN:X) is an alternate form.  It generates a multiple
plot.  Each variable, Y1 through YN, is plotted as a function of X.
Note that a colon must be inserted before the dependent variable.
  GRAPH(Y) is an alternate form.  The points are assumed to be equally
spaced and are plotted as a function of the integers, 1 2 3 ... N, where
N is the number of elements of Y.
  GRAPH(Y1,Y2, ... ,YN:) is an alternate form.  Each variable is
plotted as a function of the integers.  The arguments must have the
same number of elements.
  A generalization allows a dependent (vertical) variable to have
a two dimensional structure.  In this case each row of the array is
treated as if it were a separate variable.  The number of elements in
a row must therefore be equal to the number of elements in X.  The
array may have any number of rows.  Note that an array with this
structure can be prepared rather easily by taking advantage of the
HIWIDE convention (see the HIWIDE Help Document).
  The GRAPH command is available with all graphics packages with
the exception of the option that allows a plot versus the integers if
no horizontal variable is given.  This option is not available with
the old Tektronix graphics package.
```

Figure 7—A continuation of the tree search shown in Figure 6. These documents are short and supply operational definitions

```
:_HELP TEKWORDS
  TEKWORDS are words used to obtain a graph on the Tektronix terminal.
There are seven classes:
  1. Initialization instructions:
TEKRESET      resets the graph description to its default status.
TEKTRONIX     initializes the Tektronix graphics package.

  2. Instructions used to specify or describe the plot:
GRAPHS OFF    suppresses graphic output.
GRAPHS ON     restores graphic output.
OVERLAY       merges several graphs on the same display.
SETTITLE      specifies the graph title.
SETXAXIS      describes the format of the horizontal axis.
SETXLABEL     specifies the label used on the horizontal axis.
SETXSCALE     specifies the horizontal scale.
SETYAXIS      describes the format of the vertical axis.
SETYLABEL     specifies the label used on the vertical axis.
SETYSCALE     specifies the vertical scale.

  3. Commands used to generate output on the terminal:
ADDGRAPH      adds curves to an existing graph.
ADDSCALE      adds scale labels.
BELL          generates an audible signal.
ERASE         erases the screen.
GRAPH         plots a graph.
HARDCOPY      copies the display on the hard copy unit.
WAIT          suspends processing.

  4.  Functions which return information about the graph:
GRAPHLOC      returns the location of the graph.
CURSOR        returns an indicated location on the display.
SHOWGRAPH     lists the current graph description.

  5.  Words which deal with text output:
ANNOTATE      writes text on a graphic display.
PRINTSIZE     specifies the character size used for print output.
SETCHAR       specifies the character size used for graphic output.
TEXTHEIGHT    returns the height of a text object.
TEXTWIDTH     returns the width of a text object.

  6.  Words describing special purpose scales:
BETASCALE     describes use of Reciprocal Absolute Temperature scales.
PROBSCALE     describes how to generate Normal Probability scales.

  7.  Variables used to contain parameters and auxiliary data:
LINECODE      its value controls the format of the plotted curves.
PLOTPARM      a common storage area which can be used to store and
                retrieve the current plot specification.
NULLPOINT     a special value used to omit points from a curve.    JR

  To obtain a description of a given word XXX, enter
 HELP XXX
```

Figure 8—The list of HELP documents that relate to the Tektronix graphics package

```
:_TUTORIAL TEKTRONIX
TEKTRONI PAGE 0
TEKTRONIX
 Table of contents
   1  Introduction
   2  Plotting a simple graph
   3  More about GRAPH
   4  Labels
   5  Controlling the line format
   6  ADDGRAPH
   7  HARDCOPY, WAIT, and ERASE
   8  Setting the scale
   9  Converting to new units
  10 Setting the graph size and location
  11 Conclusion

:_MORE
TEKTRONI PAGE 1
INTRODUCTION
     A graphics support package is available which allows you
to plot graphs on a Tektronix terminal.  This Tutorial will
introduce you to some of its features.  It will first show you how
to plot a simple graph using only a few Speakeasy statements. Then
It will show you how to add scale labels and a title and how to
change the format of the plotted curves.  It will show you two
methods of making a copy of the display.  Finally it will explain
some of the facilities which allow you to set up your own graph
layout and to create quite elaborate displays.  The Tutorial should
be used while you are logged on to a Tektronix terminal.  If you
are not,  wait until you are.
     Type MORE to start the actual session.

:_MORE
TEKTRONI PAGE 2
PLOTTING A SIMPLE GRAPH
     You must be familiar with only two Speakeasy graphics
statements to begin using Tektronix graphics.  The first,
USE TEKTRONIX, initializes the system and must always be typed
before using any graphics words. GRAPH is then available to
actually plot a graph.  For example, examine the following
series of statements:
     X=GRID(0,10);  Y=SIN(X)
     USE TEKTRONIX
     GRAPH(Y:X)
Try this yourself.  (At your installation,  the USE TEKTRONIX
statement may result in a request for information about the
terminal you are using.)  The GRAPH statement automatically
gives you a complete graph.  Unless you have specified otherwise,
it takes care of the details of planning, scaling, and laying out
your graph.
     Try plotting other functions using the GRAPH command.  When
you are interested in more information, type MORE.
```

Figure 9—The start of the tutorial session on the Tektronix graphics package

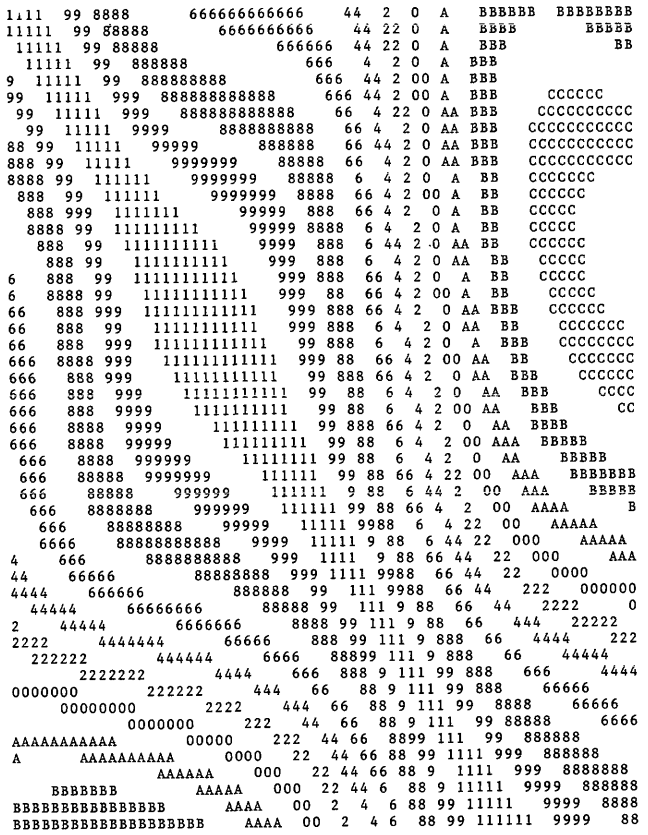

Figure 10—A sample of the output of the contour plotter for non-graphics terminals

He is then in a position to conjecture sensible alternative problems. It is in such situations that the wealth of available facilities combined with the concise and natural directed language of Speakeasy plays its most powerful role. The system provides an environment that matches closely the needs of such a user. He may quickly formulate and carry out alternative calculations and see their results. The flexibility and power of the system becomes most apparent when the results of one calculation lead to ideas that can be tested on the spot by applying new techniques built from tools provided by the system.

The graphical capabilities in Speakeasy are straightforward ones. They are designed to provide the user with the means of displaying computed functions in easily interpreted ways. The discussion here is limited to the facilities currently available for display terminals such as the Tektronix 4012 or 4014. Alternatives exist for other graphical devices and crude graphics are even available on printers and terminals limited to character displays. Figure 10 shows a simple contour plot that can be obtained on such a device. Figure 11 shows a sample of graphical output on a printer.

Perhaps the most dramatic example is illustrated in Figure 12. This example is the result of entering the

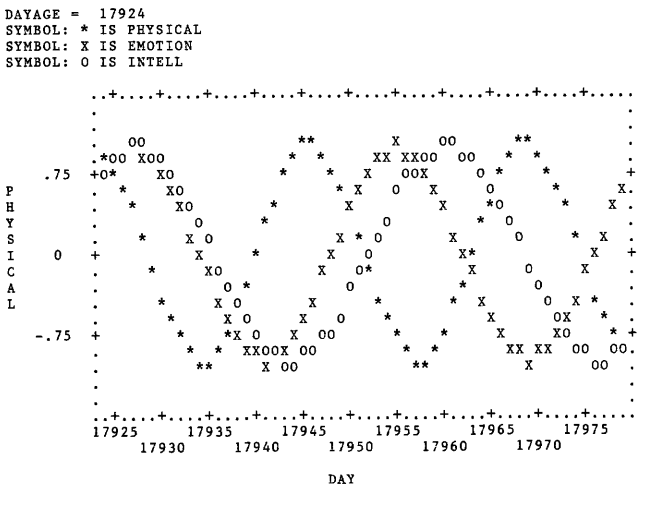

Figure 11—It is often desirable to produce graphical output on a non-graphics terminal. This is an example of one form of such output available in Speakeasy

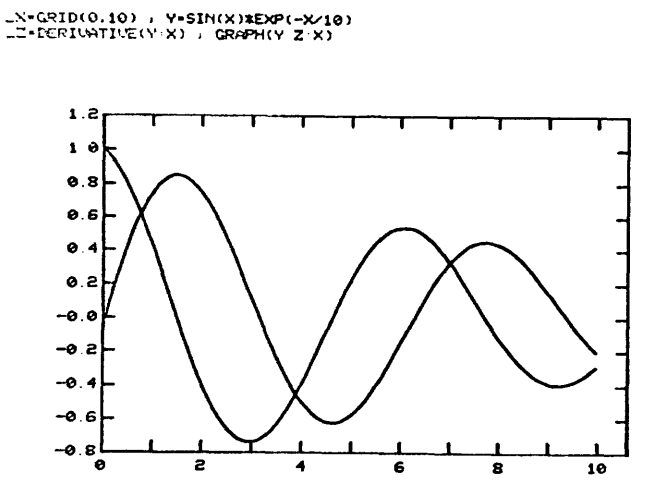

Figure 12—A simple graph and the statements that produced it

four statements:

```
X=GRID(0,10)
Y=SIN (X) *EXP (-X/10)
Z=DERIVATIVE (Y:X)
GRAPH (Y,Z : X)
```

This example makes use of a device-independent graphics package being developed by John H. Reynolds at Comsat Laboratory that is distributed as part of the standard Speakeasy system.

As indicated by this example, automatic scaling and grid generation are provided as a default. A large number of control facilities enable the user to further tailor the graphical output to his specific needs. Figure

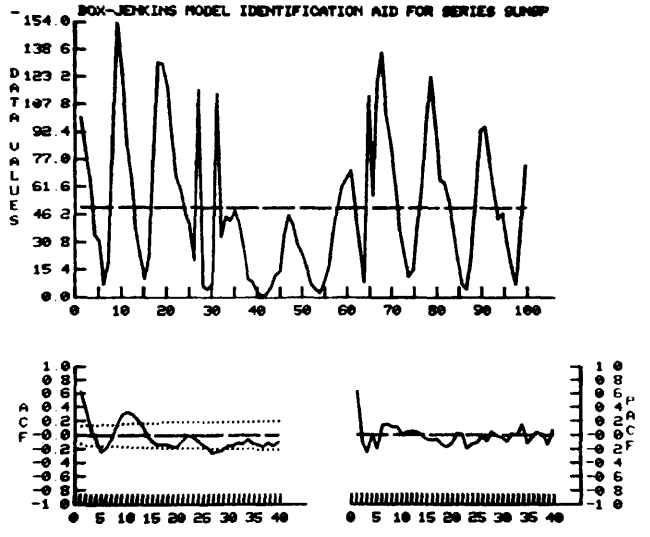

Figure 13—A set of graphs that were produced by a Speakeasy program

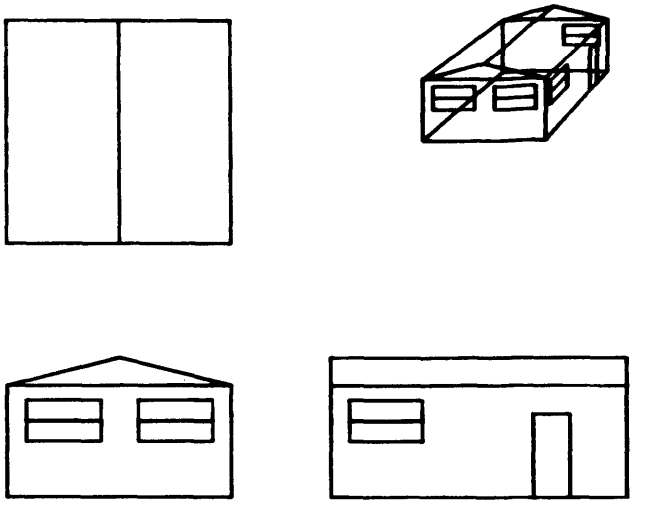

Figure 14—Another use of the three-dimensional package showing its use for a variety of projections of an object

13 is a graph that illustrates some of the other features of this package. This graph also shows some of the statistical capabilities of the system. It was contributed by R. A. Stack of The First National Bank of Chicago.

Basic three-dimensional graphics have recently been added to the facilities available[6]. The approach taken has been to define a three-dimensional object by creating a matrix of its vertex coordinates and then describing the edges through a connectivity array involving these verticies. The power of the arithmetic capabili-

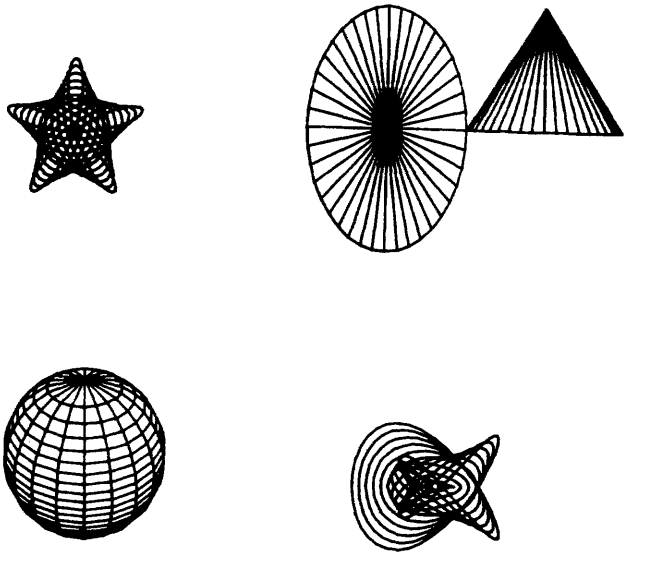

Figure 15—A sampling of some of the capabilities that are available in a system that combines general mathematical tools with simple graphics

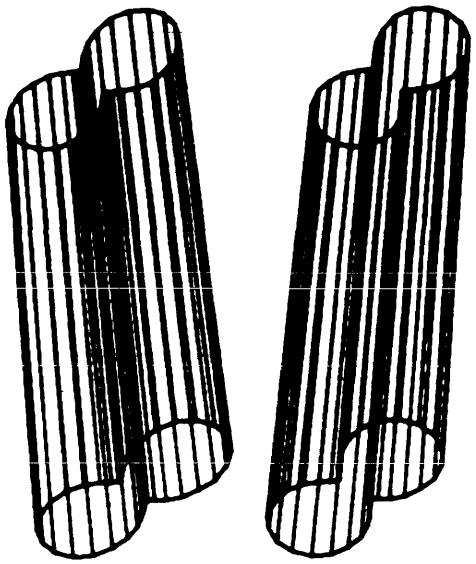

Figure 16.—An example of a steroscopic pairs produced by the three-dimensional graphics package

ties of Speakeasy can be used to manipulate the defined objects. The built-in matrix algebra, in particular, makes it easy to rotate, move or distort objects. Several special linkules were provided to help in the generation of objects made up of several components. The simple command DRAW3D is used to create the visual display. Options for perspective projections are also provided. Figures 14 through 16 show some of the possibilities.

REFERENCES

1. Cohen, S., "The DELPHI-SPEAKEASY System," *Computer Phys. Commun.* 2(1), pp. 1-10, January 1971.
2. Cohen, S., "Speakeasy: RAKUGO," *First USA-Japan Computer Conference Proceedings*, Tokyo, 1972.
3. Cohen, S., *The Speakeasy-3 Reference Manual*, Argonne National Laboratory Report ANL-8000, May 1973.
4. Cohen, S., "Speakeasy," *Sigplan Notices* 9(4), pp. 118-126, April 1974.
5. Cohen, S., *Named Storage*, Argonne National Laboratory Report ANL-7021, April 1964.
6. Blackmond, K., "Three Dimensional Graphics in Speakeasy," unpublished, July 1975.