HIGH-LEVEL CONTROL LANGUAGE FOR SCIENTIFIC AND ENGINEERING GRAPH GENERATION

Martin H. Dost, Wai Mun Syn and Nolan N. Turner, Jr.
IBM Corporation, General Products Division, San Jose, CA.

ABSTRACT

High-level programming languages of all kinds, especially simulation languages, continue to make it easier to solve problems from many fields with digital computers without requiring much knowledge of mathematics or programming. In many cases graphic display is the only efficient way of digesting the results. Numerous programs have been written for many plotting devices which require, however, too much effort for the occasional user to produce custom-tailored graphs.

A high-level control language, called GRAFAEL,* has evolved over a number of years at the development laboratory of IBM, San Jose. This language bridges the gap between the ever increasing volume of information and its display on an open-ended array of plotting devices.

Through the use of simple, unified syntax in a universal post-processor, the data to be plotted is channeled from any given program to one or more available plotters or printers. This arrangement allows the user to specify the format of the desired graph with little or no concern for procedures or device requirements.

Simple statements, such as,

```
GRAPH TIME, X,Y
LIST TIME,A,B,U,V,W,X
```

allow graphs and listings to be generated with standard features, assumed by default. Any special attributes of each variable may, however, be specified. Among the options available through GRAFAEL are the type of plotting device, axis type (linear or log), length and position of any axis, type of line or marker, spacing between and number of divisions, scale factors, starting value and dimensional units. An example of a partly specified variable may be

```
TIME(LE=9, SC=0.05, UN=MSEC, LO=0.1)
```

which results in a 9" long axis, scaled for 0.05 msec per unit graph length, and starting at t=0.1. The user also has the option of displaying curves selectively from different runs, overlaying graphs, choosing line types, and labeling each graph independently.

INTRODUCTION

Every day scientists and engineers solve more complex and new types of problems on the digital computer. High-level programming languages, especially simulation languages, have made it progressively easier for mathematically unskilled people to use computers to good advantage. Yet, some skepticism exists. For example, do the results of all these programs get fully digested, and what is being done to expedite evaluation? Why do so many people still tabulate results and then try to interpret them laboriously? Isn't it common knowledge that graphic presentation of data is more efficient and that plenty of plotting devices are commercially available?

Much sophisticated drafting and plotting is being done automatically, but the programming effort, typically required to obtain customized graphs, is substantial. When graph requirements change almost constantly, as is the case in exploratory simulation studies, a high degree of flexibility in specifying graphs is needed. That, however, usually conflicts with the desired simplicity for the unsophisticated computer user.

Over a number of years, a program has evolved at the IBM development laboratory in San Jose, California, which gives high-level control to the users of many application programs in specifying a great many types of graphs to be drawn on any of several plotters. The concept originated in the late 60's when it was found that a major reason for the popularity of the simulation language DSL/360 (like CSMP[1] an advanced version of DSL/90[2]) was the great ease with which computed results would be plotted. Simple statements, such as

```
PREPAR 0.1, TIME, X, Y, YDOT
GRAPH  X, Y, YDOT
```

sufficed to specify the variables to be prepared for plotting and the composition of the graph picture. The first variable in the GRAPH statement was the independent variable; all others following were the dependent variables. The data would be scaled automatically to fill a 12" x 8" area, unless a specific scale factor and graph dimensions were given. For example,

```
GRAPH 2.5, 10, 6 TIME, X, Y
```

specified a scale factor of 2.5 problem units per inch for the dependent variables X and Y, and a 10" x 6" graph area.

As useful as they were, these capabilities were still too restrictive for many applications, and users kept asking for more flexibility. It became apparent that such capabilities as independent scaling of each variable, positioning and labeling of each axis, choice of linear and logarithmic scale, superposition of selected curves, etc., must be provided if a useful general-purpose engineering graphic program is the objective. To accomplish this, the GRAPH statement of the evolving graph control language was expanded to the following format:

```
GRAPH(---) TIME(---), X(---), Y(---)
```

As before, the first variable "TIME" is the independent variable, followed by the dependent variables "X" and "Y".

Each plot variable may be qualified by an optional list of attributes enclosed in parentheses. These are "local" attributes pertaining to that variable exclusively. There are also "global" attributes which affect the dependent variables as a group or the entire picture as a whole. These options are placed immediately following the "GRAPH" label. As an illustration, consider the following two GRAPH statements:

```
GRAPH TIME, X(LENGTH=6,SCALE=0.5), Y(LENGTH=6)

GRAPH(LENGTH=6), TIME, X(SCALE=0.5), Y
```

They produce identical pictures. X and Y will be plotted against TIME. The vertical axes are 6 inches long and the horizontal axis will be 12 inches by default. The X curve will be drawn with

*An experimental program.

a scale factor of 0.5 unit per inch, whereas the Y values will be automatically scaled to fit a 6-inch axis. By exercising selected options, a highly complex graphic picture may be composed. On the other hand, meaningful defaults are built into the system so that a reasonable picture will be drawn if no optional attributes have been specified. Thus,

```
GRAPH TIME, X, Y
```

will produce an 8" x 12" graph of X and Y versus TIME.

The language which was actually implemented is called GRAFAEL. It provides a graph-generation capability which is at the same time very versatile and very simple to use. To meet the requirement for <u>versatility</u>,

1. GRAFAEL accepts data from many sources without restricting either the programming language used to generate the data, or the storage device used to contain the data.

2. The user is allowed a high degree of flexibility in specifying the content and the appearance of his graphs.

3. The graphs may be produced on a variety of plotting devices.

To meet the requirement for <u>simplicity</u> of use,

1. The only graph related function performed in the user's application program is the preparation of a data set containing the values to be graphed. Subroutines are provided which assist in this process.

2. The GRAFAEL language with which the user specifies graphs employs highly mnemonic terms within a simple consistent syntax.

3. The language is a descriptive language, and not procedural, to free the user from the need to know the steps involved in making a graph.

4. A full set of default assumptions is included. As a result the simplest form of graph statement will produce a usable graph.

The Structure of GRAFAEL

GRAFAEL represents the common link between all the application programs, which generate plottable data, and the various plotting devices available at any particular computer installation. Figure 1 shows schematically how plots are obtained empirically and, by comparison, how GRAFAEL is used to process raw data into a form acceptable directly by plotters. The program which is a stand-alone post-processor, was written in PL/I; it will run in a 104 K byte core region.

An interface routine, SAVE, collects the raw data from the application program and puts it in a desirable format. GRAPH commands serve to communicate specifications about the desired plot appearance. GRAFAEL then sorts the data and determines specifically what each graph is to contain, based on the data itself, the options exercised, and the built-in default assumptions. This graph-generation portion of GRAFAEL deals only with picture-oriented functions, and is relatively ignorant of the capabilities and limitations of the individual plotters. It identifies the device or devices to be used, the start of a frame or picture, the line segments which comprise the frame, and the end of the frame. These functions are common to all graphs, regardless of the plotter used.

For the actual graph production, GRAFAEL uses a collection of plotter interface subroutines which

Process	Empirical		Numerical
Data generation	physical system under test		mathematical model in application program
Interface	measurement (manual/automatic)		formatted points data set command data set
Graph generation	data display specification (type of graph, medium and details)		GRAFAEL (a) picture composition (device independent
	plot production (drafting/recording)		(b) data set generation for plotting procedure
Typical plotting devices	drafting instruments	recorders display devices	pen plotters (drum, flatbed) photo plotters (light, E-beam) printers (line, non-impact)

Fig. 1 Comparison of graph generation processes

deal with the vagaries of the individual plotter. (This collection of routines, called SPLINTER, for <u>S</u>an Jose <u>plot</u> <u>inter</u>face, is also used for plotting applications other than GRAFAEL.) They convert the picture description into one or more data sets which will control the selected plot devices. Only the subroutines within SPLINTER are concerned with characteristics of the individual plotters such as resolution, maximum allowable drawing size, etc.

Aside from two types of printer plots for line printers, several pen and photo plotters have been supported through SPLINTER: CalComp* 566, Gerber 562 (drum type), Gerber 522, 523 (flatbed pen plotters), Gerber 532 (large photo plotter) and Information International, Inc. FR80 (microfilm).

Breaking up the job of producing graphs automatically is analogous to the situation when graphs are produced from empirical data by a team of workers (see Fig. 1). Preparation of the raw dataset can be compared to measurement of analog variables or event counting in continuous or discrete form from the real systems (as simulated in the application programs), while the SPLINTER routines can be compared to draftsmen or technicians who take numerical and analog data and produce plots. They need to know what the desired graphs should look like and on what media to put the curves or markers.

Note that the engineer or scientist in charge of the experiment is not concerned with the sequence of drafting operations or the drafting instruments. However, he will have to specify whether linear or log paper is used, which data are to be selected for plots, what resolution is desired, how to label and position axes and entire plots, and he may have preferences about line and marker types. In this way the responsible experimenter is relieved of the chores which are routine but often time consuming, and he can concentrate on the problem of data evaluation toward understanding and communicating the results. Through the simple mechanism of exercising the various options, the user of GRAFAEL has the tools to tailor his graphs more and more precisely while his computer study progresses and the results become better known.

*CalComp is a registered trademark of California Computer Products.

Joining GRAFAEL to the Data Generating Program

To produce a graphic picture, GRAFAEL must have as input two groups of data:

1. the raw data to be plotted, and
2. the commands and graph options that describe the raw data and specify the picture details.

The raw data must be placed into the following prescribed GRAFAEL format.

data record	identifier	n	k	d_1	d_2	d_3	···	d_n

Each record is of variable length and begins with a unique group identifier. Immediately following the identifier is an integer (n), indicating the number of data values in the record (d_1,d ,···, d_n). Obviously, n must be greater than or equal to 2 since two coordinates are required to plot a point in a graph. Next, another integer, k, serves as a run counter. All data records having the same identifier and the same run number form a logical plot set which must be terminated by an end-of-run (also referred to as an end-of-curve) record. This terminator record contains only the first three fields of a normal data record and uses the record field as a flag by setting n=1. The end-of-run record also increases the run counter k by one.

Any number of groups of data items with any number of runs per group may be generated and accumulated in one job without restriction as to the order in which the data records are collected.

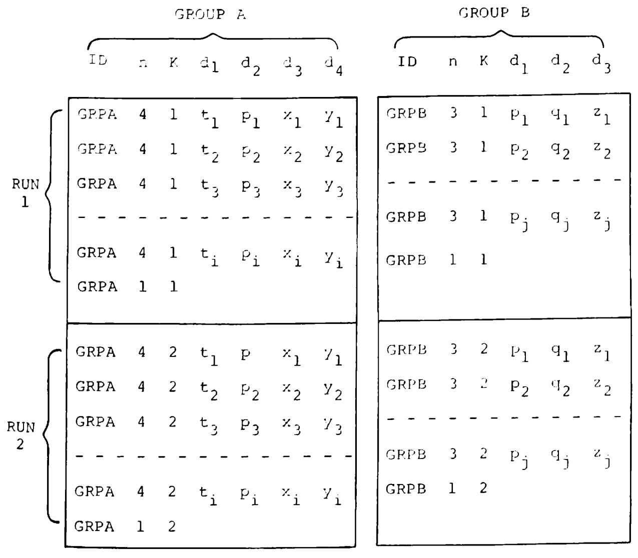
As an illustration, suppose an application program computes data values for the variables p, q, x, y, and z as a function of t for two values of a parameter C (say C1 and C2). Plot points are to be collected in two groups at different frequencies.

group 1 named GRPA containing t, p, x, y

group 2 named GRPB containing p, q, z

Four plot sets will result as depicted in Fig. 2.

To assist the user in collecting data for GRAFAEL, general interface routines have been written for use in both FORTRAN and PL/I application programs. For large production type application programs, the data collection process can be integrated into the system by simply writing on some file the data values to be plotted in accordance with the prescribed format.



GROUP A

	ID	n	K	d_1	d_2	d_3	d_4
RUN 1	GRPA	4	1	t_1	p_1	x_1	y_1
	GRPA	4	1	t_2	p_2	x_2	y_2
	GRPA	4	1	t_3	p_3	x_3	y_3
	- - -						
	GRPA	4	1	t_i	p_i	x_i	y_i
	GRPA	1	1				
RUN 2	GRPA	4	2	t_1	p	x_1	y_1
	GRPA	4	2	t_2	p_2	x_2	y_2
	GRPA	4	2	t_3	p_3	x_3	y_3
	- - -						
	GRPA	4	2	t_i	p_i	x_i	y_i
	GRPA	1	2				

GROUP B

ID	n	K	d_1	d_2	d_3
GRPB	3	1	p_1	q_1	z_1
GRPB	3	1	p_2	q_2	z_2
- - -					
GRPB	3	1	p_j	q_j	z_j
GRPB	1	1			
GRPB	3	2	p_1	q_1	z_1
GRPB	3	2	p_2	q_2	z_2
- - -					
GRPB	3	2	p_j	q_j	z_j
GRPB	1	2			

Fig. 2 Data array for two groups and two runs

After the processing of raw data into GRAFAEL acceptable format, graphic pictures may be produced by specifying appropriate commands and graph options. Continuing with the above illustration, GRAFAEL must be informed as to the names and groupings of the saved data. This is accomplished by means of the "SAVE" command:

```
SAVE (GRPA) T, P, X, Y
SAVE (GRPB) P, Q, Z
```

Now, adding the appropriate GRAPH statements, the user may compose as many graphic pictures as he desires. For example, he may plot X and Y against t for both runs from data identified by GRPA:

```
GRAPH (G1/GRPA) T, X, Y
```

or, p from run 1 and y from run 2 versus t:

```
GRAPH (G2/GRPA) T, P(RUN=1), Y(RUN=2)
```

or, p and q against z for run 2 from GRPB:

```
GRAPH (G3/GRPB,RUN=2) Z, P, Q
```

The Options in GRAFAEL

The most often used options and their two-letter abbreviations of GRAFAEL are listed alphabetically in Fig. 3. Of these, DEVICE and OVERLAY

AXISTYPE	(AX)	axis type (linear or log)
DEVICE	(DE)	plotting device
LENGTH	(LE)	length of axis
LINETYPE	(LI)	type of line to be drawn
LOWVALUE	(LO)	low value of variable range
MARKER	(MA)	type of markers to be drawn at data point
NBRINT	(NB)	number of divisions (or decades)
OVERLAY	(OV)	superimpose on previous graph
POSITION	(PO)	position of graph or axis origin
RUN	(RU)	run number of data to be plotted
SCALE	(SC)	scale factor (variable units/graph unit)
TICINT	(TI)	interval between tick marks (decade length)
UNIT	(UN)	dimensional units (attached to axis label)

Fig. 3 The principal options in GRAFAEL

are strictly graph options, by which the plotting device is picked and spatial overlapping of plots is specified; e.g.,

```
GRAPH (DE = SPRINT, OV)
```

This causes the short (i.e., single page) type of printer plot to be chosen, as well as consolidation with the previous plot. Most of the options listed in Fig. 3 are "curve options," i.e., they pertain to the dependent axes or their associated curves; e.g.,

```
Y (LO = -0.1, SC = 0.02, LI = 3, LE = 6)
```

In this case, the variable Y is to be plotted by a curve using line type 3 (short dashes) over a

6" vertical distance, starting at a low value of -0.1 and scaled for 0.02 unit of Y per inch. If these options are put into the graph option field, they would apply to all dependent variables, causing alignment, uniform scaling, etc. Some options clearly don't belong in the option field of the independent variable; these are line type (LI), marker type (MA), and run number (RU). An example of a horizontal axis specification might be:

```
     TIME (LE = 35, TI = 0.5, UN = SECONDS)
```

which lays down a 35" long axis with a tick mark interval of 0.5" and the unit SECONDS printed in parentheses behind the variable TIME.

A logarithmic scale to the base 10 can simply be obtained by use of the axis type option (AX). The number of decades (NB) is assumed to be 3 by default and the spacing between decade lines (TI) 2.5". For example:

```
 Z (AX = LOG, NB = 6, TI = 1.5, LO = 100, MA = 4)
```

will cause the variable Z to be plotted by markers of type 4 (triangles) over 6 decades and 9" distance, starting at 10^2 and ending at 10^8 , but excluding events outside of that range.

Flexibility in spatial arrangement of axes or entire graphs is provided by the position option (PO), which has two coordinates: X and Y. PO = 0,5 in the graph option field will cause the entire graph to be translated vertically by 5". In an option field associated with either an independent (or dependent) variable, it will cause the left (or lower) end of the axis, i.e., its low value point, to be shifted to the right by an amount X, and up by an amount Y. If only one number is given, it is assumed to be X, and $Y \equiv 0$. The position option, together with the length option enables X-Y plots to appear like strip charts on time base recorders; this avoids overlap of curves, if desired. Positioning of dependent variable axes overrides side-by-side axis arrangement of successive dependent variables, starting at (0,0) and followed by (-1.5,0), (-3,0), etc.; axis omission is possible by the command AX = OMIT.

Graph Samples

To illustrate the use of GRAFAEL, primarily the options, several sample plots of results from problems solved by a high-level simulation language are shown in Figs. 4-9.

Figure 4 resulted from a SAVE statement for the variables TIME, I, V2, SAW, FF, and PC plus the statement

```
GRAPH (DE=CALCOMP, LI=1, LE=3) TIME (UN=SEC)...

      I, V2 (LI = 2) SAW (PO = 0,3.5, LE = 1) ...

      FF (PO = 0,5, LE = 0.5) PC (PO = 0,6, LE = 2)
```

Such a graph is the equivalent of a strip chart recording with variable strip width and overlap capability for several variables vs. time.

Figure 5 shows families of curves vs. time and angular displacement (TH). This graph was obtained by saving the variables TIME, TH, THDOT, EPOT, and EKIN for three runs and placing these two GRAPH statements before the final END:

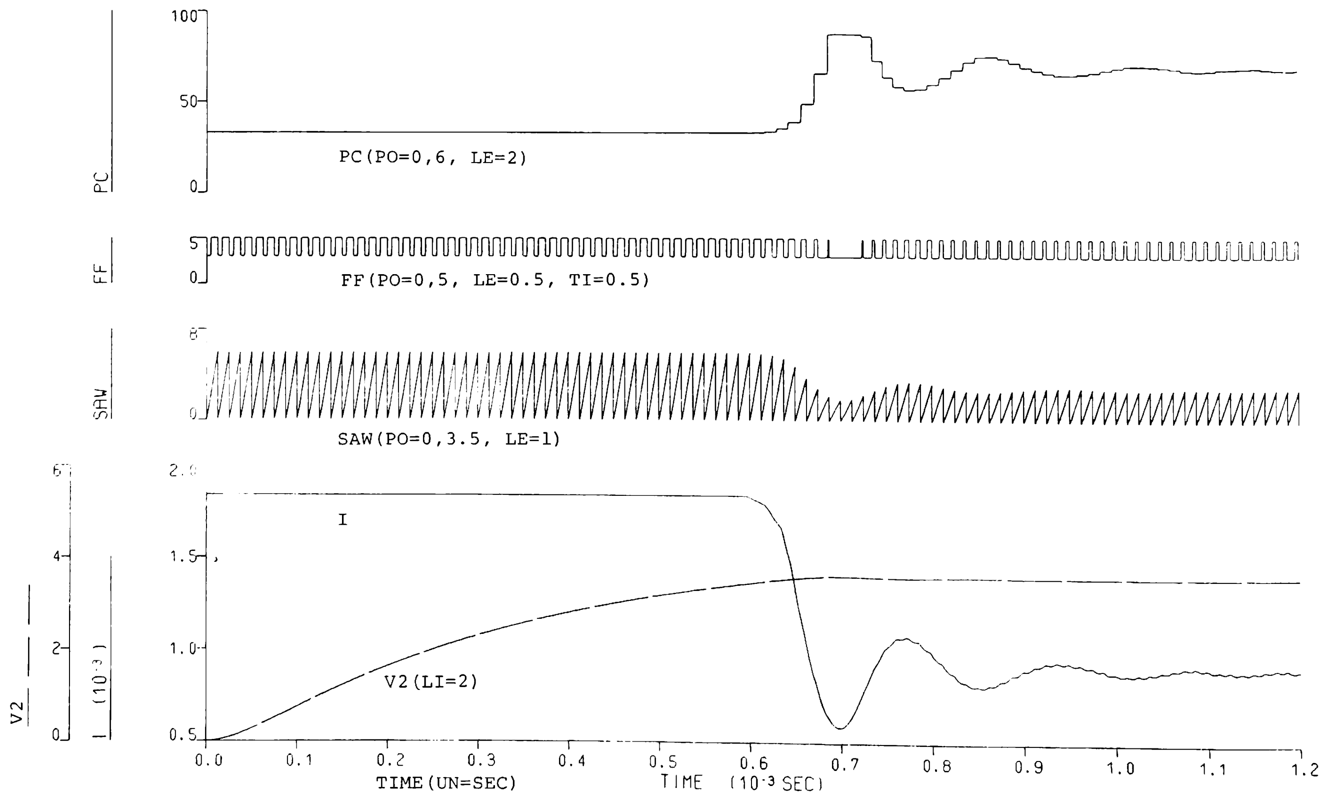
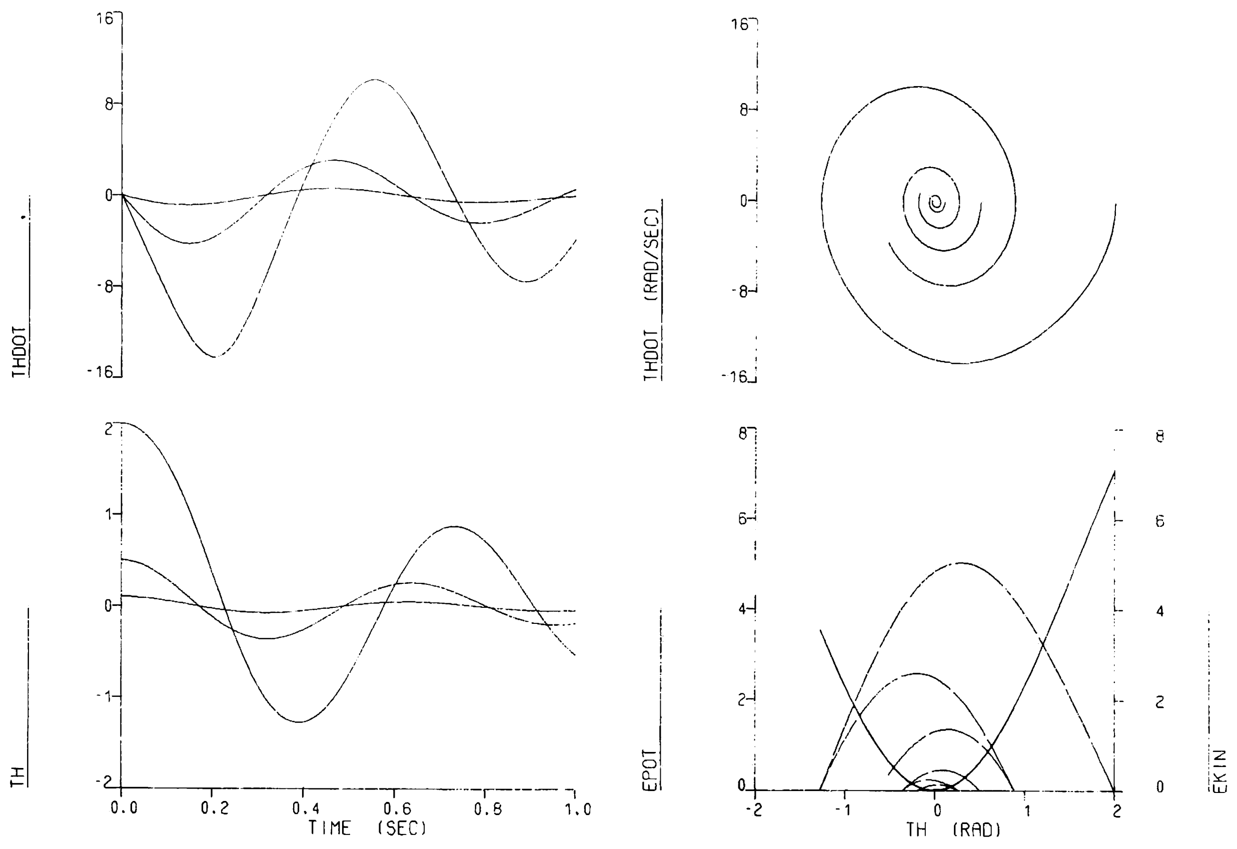


Fig. 4 The strip chart effect.



```
GRAPH (DE=CALCOMP,LE=4) TIME(LE=5,UN=SEC) TH, THDOT(PO=0,4.5,LI=1)
GRAPH (DE=CALCOMP,OV,PO=7,LE=4) TH(LE=4,UN=RAD) EPOT, EKIN(PO=4) ...
                                THDOT(PO=0,4.5,LI=1,UN='RAD/SEC')
```

Fig. 5 Curve families vs time and X-Y plots overlaid

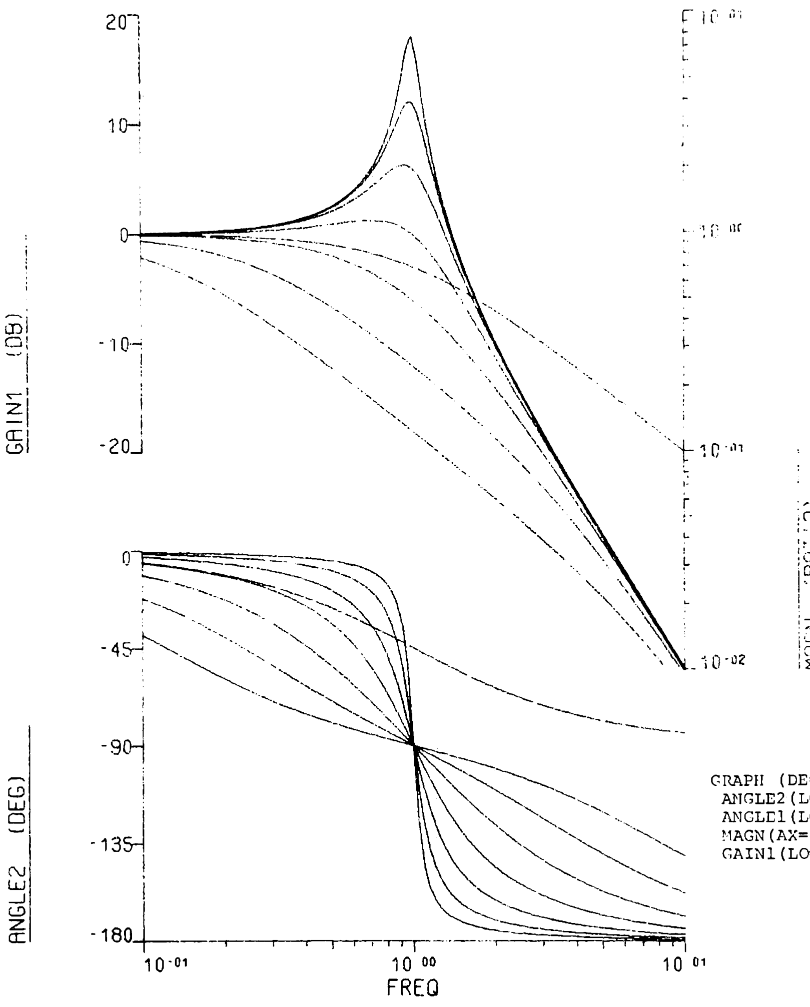
```
GRAPH (DE=CALCOMP, LE=4) TIME(LE=5, UN=SEC) ...

 TH, THDOT(PO=0,4.5, LI=1)

GRAPH (DE=CALCOMP,LE=4,OV,PO=7) TH(LE=4,UN=RAD) ...

 EPOT, EKIN(PO=4) THDOT(PO=0,4.5,LI=1,UN='RAD/SEC')
```

The first one shows angular displacement and velocity of a pendulum for three different amounts of damping. The spatial separation vertically is accomplished by the position command for the THDOT axis. To place the other two pictures immediately to the right of the first ones, overlay (OV) is used for the second GRAPH command. In the lower picture, potential energy (EPOT) and kinetic energy (EKIN) are plotted together; by placing the axis for EKIN at the right-hand margin, the framing effect was obtained and equality of the two energy scales was demonstrated. The upper right picture, finally, shows velocity vs. displacement, i.e., a phase plane plot or "Lissajous" figure, as put on X-Y plotters or oscilloscopes.

Figure 6 is a set of semi-log and log-log plots, familiar to many: gain and phse angle vs. frequency for linear first and second order systems. Note the spatial overlap of the two plot fields due to judicial choice of axis positions and length, scales and low values.

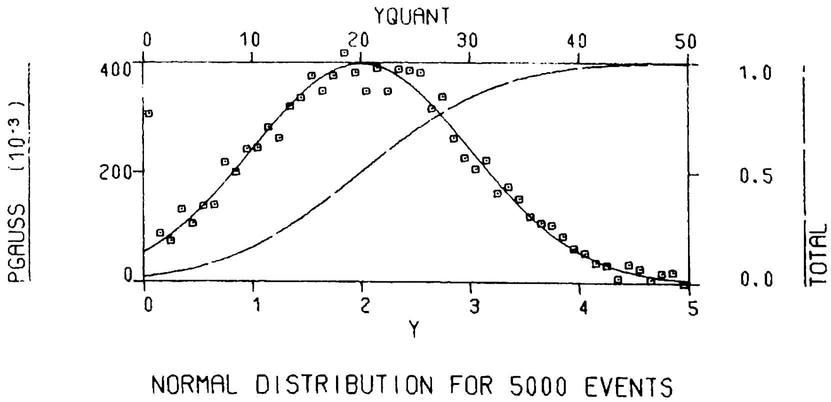
Figure 7 is an example of a graph obtained from more than one SAVE list. A Gaussian distribution function (PGAUSS), given analytically, and its integral (TOTAL), obtained by numerical integration of PGAUSS with respect to Y, were computed in one part of the program, and the result saved separately from probablistic ones, collected in SAVE(Q). Through overlay and axis positioning, a four-sided frame was obtained.

An even more complex graph is shown in Fig.8a, namely line and point graphs from five SAVE lists. The nameless set of SAVE and GRAPH statements produced the solid lines which represent the constant damping curves in the z-plane, where $z = x + jy$ is complex. Superimposed are four sets of points, denoted by different markers as obtained from a special root locus routine, which finds the roots of polynominals for different gain constants in a sampled data control system: crosses for open loop poles, squares for open loop zeros, triangles for closed loop poles at nominal gain, and asterisks for running gain values. Note the axis translation into the origin and the omission of axes for all overlaid graphs.



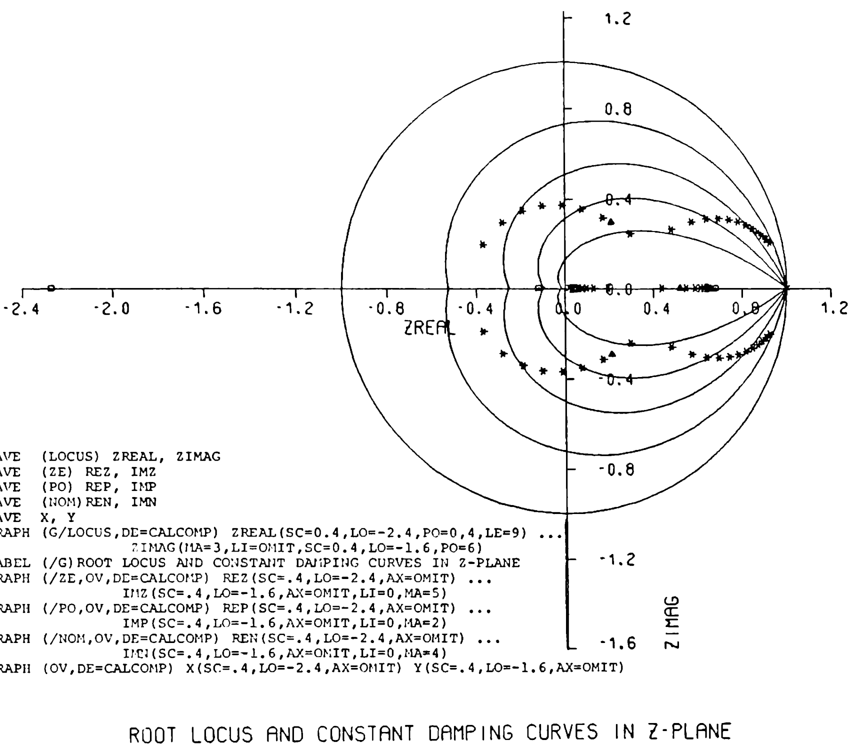
```
GRAPH (DE=CALCOMP) FREQ(AX=LOG,NB=2) ...
 ANGLE2(LO=-180,SC=45,UN=DEG,TI=0.9,LE=3.6) ...
 ANGLE1(LO=-180,SC=45,AX=OMIT,TI=0.9,RUN=1) ...
 MAGN(AX=LOG,LO=.01,TI=2,PO=5,2.5,LI=1,UN=RATIO)...
 GAIN1(LO=-20,SC=10,UN=DB,PO=0,4.5,LI=2,LE=4,RUN=1)
```

Fig. 6 Semi-log and log-log plots



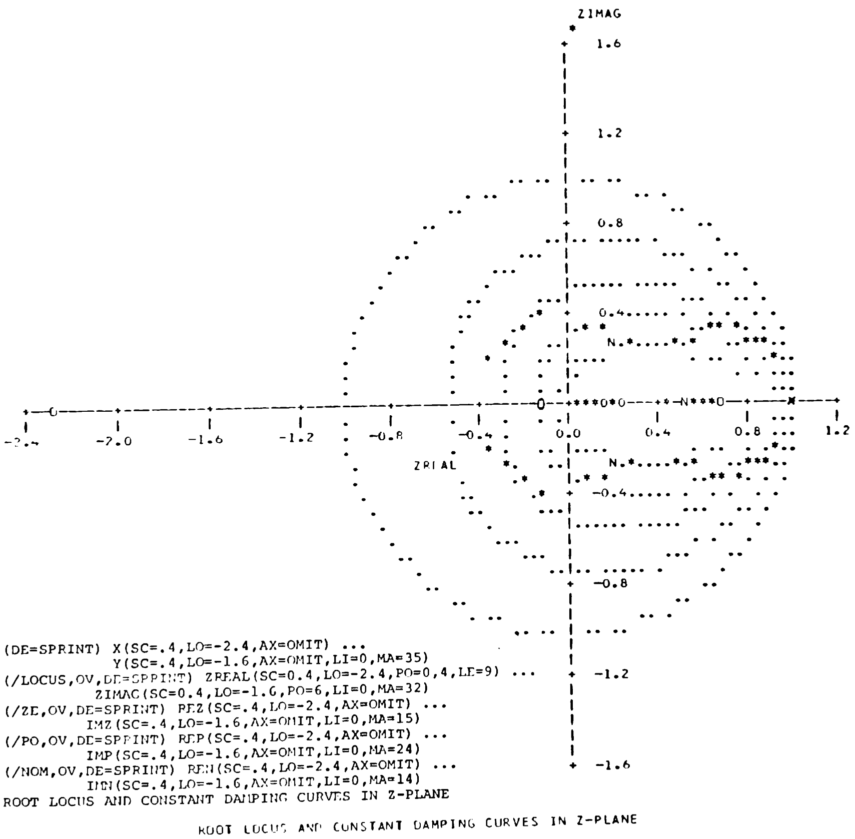
```
SAVE  Y, PGAUSS, TOTAL
SAVE  (Q) YQUANT, PROBAQ
GRAPH (DE=CALCOMP,LE=2) Y(LE=5) PGAUSS, TOTAL(PO=5)
GRAPH (/Q,OV,DE=CALCOMP) YQUANT(LE=5,PO=0,2) ...
      PROBAQ(LE=2.5,LI=0,MA=5,AX=OMIT)
LABEL NORMAL DISTRIBUTION FOR 5000 EVENTS
```

Fig. 7 Line and point plots superimposed



```
SAVE  (LOCUS) ZREAL, ZIMAG
SAVE  (ZE) REZ, IMZ
SAVE  (PO) REP, IMP
SAVE  (NOM)REN, IMN
SAVE  X, Y
GRAPH (G/LOCUS,DE=CALCOMP) ZREAL(SC=0.4,LO=-2.4,PO=0,4,LE=9) ...
                 ZIMAG(MA=3,LI=OMIT,SC=0.4,LO=-1.6,PO=6)
LABEL (/G)ROOT LOCUS AND CONSTANT DAMPING CURVES IN Z-PLANE
GRAPH (/ZE,OV,DE=CALCOMP) REZ(SC=.4,LO=-2.4,AX=OMIT) ...
                IMZ(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=5)
GRAPH (/PO,OV,DE=CALCOMP) REP(SC=.4,LO=-2.4,AX=OMIT) ...
                IMP(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=2)
GRAPH (/NOM,OV,DE=CALCOMP) REN(SC=.4,LO=-2.4,AX=OMIT) ...
                IMN(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=4)
GRAPH (OV,DE=CALCOMP) X(SC=.4,LO=-2.4,AX=OMIT) Y(SC=.4,LO=-1.6,AX=OMIT)
```

Fig. 8a Superposition of data points from several SAVE lists



```
GRAPH (DE=SPRINT) X(SC=.4,LO=-2.4,AX=OMIT) ...
                  Y(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=35)
GRAPH (/LOCUS,OV,DE=SPRINT) ZREAL(SC=0.4,LO=-2.4,PO=0,4,LE=9) ...
                ZIMAG(SC=0.4,LO=-1.6,PO=6,LI=0,MA=32)
GRAPH (/ZE,OV,DE=SPRINT) REZ(SC=.4,LO=-2.4,AX=OMIT) ...
               IMZ(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=15)
GRAPH (/PO,OV,DE=SPRINT) REP(SC=.4,LO=-2.4,AX=OMIT) ...
               IMP(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=24)
GRAPH (/NOM,OV,DE=SPRINT) REN(SC=.4,LO=-2.4,AX=OMIT) ...
               IMN(SC=.4,LO=-1.6,AX=OMIT,LI=0,MA=14)
LABEL ROOT LOCUS AND CONSTANT DAMPING CURVES IN Z-PLANE
```

Fig. 8b Printer plot equivalent to Fig. 8a

Figure 8b, finally, is the equivalent printer plot of Fig. 8a, which was produced on a pen plotter. By specifying the device to be SPRINT, the short print format (limited to one page), and choosing desired marker types differently, a crude, but useful, picture was obtained that resembles Fig. 8a in many ways. It shows all the different curves from 5 runs and 5 groups, drawn in an X-Y sense. It demonstrates the same axis positioning, omission and labeling power (within the capabilities of a liner printer) as for plotters. If spatial conflicts exist in the positioning of characters, the last symbol replaces any previous ones, rather than overstriking them. The sequence of statements may, therefore, matter in certain applications. In order for special points on the real axis in this example to show up, the GRAPH statements were rearranged.

Typically, printer plots are used for less crowded graphs, and often the limitless long print format is required to give the necessary resolution, if pen or photographic plotters are not available. But for expediency or economic reasons, printer plotting may well be preferable.

SUMMARY

As demonstrated through a number of examples in this paper, a high-level control language for the production of engineering and scientific graphs is not only feasible but quite practical. The apparent conflicts between a high degree of flexibility in graphing requirements, desired for many computer studies, and simplicity in coding can be resolved. Much like the highest level simulation languages, GRAFAEL gives the user both the power of tailoring graphs to his needs and the simplicity in nonprocedural syntax to state his wishes with a minimum of programming effort.

Through separation of the general picture composition aspects from the plot generation aspects for any particular plotter, this program not only serves as a postprocessor to many types of application programs, but it can prepare data for fully automatic plotting of digital data on any of several given plotters. The data origin may be experimental or a computer program, regardless in which language. By simplifying the programming for specifying graphs to the necessary minimum, a tool has been created with which the user can be highly productive in displaying data effectively, using the best suited plotting device, yet with little or no concern for its idiosyncrasies.

REFERENCES

1. Continuous System Modeling Program III (CSMP III) and Graphic Feature, (Program Number 5734-XS9) GH19-7000-1 and SH19-7001-2, 1972, IBM World Trade Corporation, 821 United Nations Plaza, New York, N.Y. 10017.

2. Syn, W. M. and Linebarger, R. N., "DSL/90 -- A Digital Simulation Program for Continuous System Modeling," 1966 Spring Joint Computer Conference, April 26-28, 1966.