Programmed Methods for Printer Graphical Output

David Garfinkel*
Johnson Foundation for Medical Physics, University of Pennsylvania

It is frequently desirable to display the results of computation in a graphical form. This is often done through the use of special hardware such as digital X,Y-plotters. Programmed graphical output for standard printers is preferable in several situations: (1) when economic considerations do not justify the expense of special hardware for the purpose, (2) when a combination of graphical output with some other kind, such as explanatory material, is desired, and (3) when some special variety of graphical output is needed which cannot readily be drawn by an analog device.

A number of routines have been prepared (primarily by users rather than manufacturers) to convert numerical data into graphical form for printing by output typewriters or line printers. Virtually nothing on this subject has been published, and this report represents an admittedly incomplete attempt to describe this technique and suggest possibilities for its use.

The graphing routines encountered while collecting information for this report are of two types: generalized routines, usually intended to plot curves, and special purpose routines, for specific jobs which may be as diverse as representing census data or simulating a radar scope.

Generally Applicable Methods

The most straightforward approach to drawing a graph is to treat a sufficiently large portion of the store (initially filled with spaces or background markings) as if it were a piece of graph paper, each character position representing the intersection of two lines. Points may then be plotted as they are received, by placing the right character in the proper position. When the last information has been plotted, this portion of storage may be printed out immediately or stored on tape for subsequent printing. This procedure has the disadvantage of requiring extensive storage space; it has the advantage that the input need not be received in any particular order. This procedure has been implemented (for instance, Share #1085, which provides a rectangular grid background to assist the eye in determining the exact position of points).

A more commonly used method of preparing a graph is to prepare printer instructions line by line (for either immediate printing or tape storage). This requires less storage than the first method but it does require all the data for one line at once; points produced at random must be ordered before graphing. This method is probably also faster than the other (not counting ordering operations), as the routine need work with only one dimension.

The simplest procedure for preparing a graph for line printers is to start with a line of spaces (or spaces with position-indicator and margin markings), add any desired marginal data (e.g., time values), and then insert into the appropriate positions in this line the characters which are to constitute the graph. Routines of this sort have been prepared by the author and associates for the Univac I, the 704, and the 7090. The Univac routine is about 250 instructions in length, requires about 50 words of working-storage plus input and output areas, and is fairly fast.[1, 2] This routine effectively determines the word in which the output character is to be placed by dividing the (two-digit) input number by the word length (in characters), taking the remainder as the proper digit in that word. The problem of two characters in the same space at the same time is handled by making a marginal note of each coincidence.

It is easiest to construct a line of output in this fashion with (or for) a computer in which the word length is one alphanumeric character: if the width of the line is 1.0, and one wishes to represent .86 by the letter A, one simply places an A in the 86th of a line of 100 words. In a computer having words of 10 characters this would become the sixth digit of the eighth word.

Routines intended for line-at-a-time printers do not yield optimal output for typewriters, which are inherently slower, but need not put out a fixed number of characters and can take advantage of tab stops. Specific typewriter output routines are useful for small computers having a typewriter as the principal output device (and possibly where one wants to take the output on punched paper tape and type it on a flexowriter). A routine of this type[2] for the Univac I assumes that a tab stop has been set every tenth space; on the average this uses only half as much typing time as the corresponding routine intended for line printers, but the calculation is slower. Either the inputs must be in order within each line, or a process of replacing spaces by tab orders is required. The delay due to the typewriter can be minimized by interspersing type orders with subsequent calculation.

However, the individual lines are prepared, they must be related to each other. In many applications, input for each line is delivered to the graphing routine in sequence, with the lines equally spaced; in others, it is in sequence but the lines are not uniformly spaced. If it is not in sequence it must be ordered, using as a key the linear dimension of the graph that goes across the lines or down the paper. The number of input items may not coincide with the number

* Research Career Development Fellow, U. S. Public Health Service.

[1] A graph prepared by this routine has been published in these communications (Feb. 1962, p. 116).

[2] The author will furnish flow charts on request.

TABLE 1. REPRESENTATIVE GRAPHING ROUTINES

Description	*Installation and Programmer*	*Machine and Language*
One- or two-curve plot (with scaling); inputs must be in order.	Yale University	650 FORTRANSIT
General plot of up to 4 simultaneous curves; input must be floating-point numbers (in any order).	R. Herbold, National Bureau of Standards	704 SAP-coded FORTRAN
Multi-curve plot (up to 15 curves); inputs in order.	Biostatistics Unit, UCLA	7090
Two-curve plot with computation of RMS difference; inputs must be in order and are assumed equally spaced.	Janet Bramhall, Applied Physics Lab., Johns Hopkins University	7090 FORTRAN, FAP
Histogram plot ($\leqq$ 100 variables, $\leqq$ 2000 cases). Plots a one page histogram for each variable.	Biostatistics Unit, UCLA	7090 FORTRAN, FAP
Cross-tabulation Plot (one-page cross-tabulation plots, specifications)	Biostatistics Unit, UCLA	7090 FORTRAN, FAP
Pictorial representation of statistical data (such as scatter-grams)[a]	RAND Corp.	7090 FORTRAN
Graphical representation of a random distribution of N points in 2 dimensions[a]	RAND Corp.	7090
UR PLOT. Line-at-a-time grapher, for offline printing	SHARE #1118, U.S. Annino, D. Long	709/7090 (FORTRAN, FAP)
UM PLOT. Entire graph in memory simultaneously, for offline printing	SHARE #1085, B. Carnahan, L. Evans	FORTRAN, SAP, or MAD
RW PLOT. Generalized plot with the 7090 doing the calculations for the 1401.	M. Kory, Space Technology Labs.	7090, 1401
Two-curve plot (input must be in order and assumed to represent months). Can be altered to give semilogarithmic plots.	H. Eisenpress, J. L. McPherson, J. Shiskin, Bureau of the Census	UNIVAC I
Nine-curve plot for Flexowriter	Eugene Grabman, Research Labs. For Engineering Sciences, University of Virginia	Burroughs 205
SPLOT. One-curve plot with scaling, one input at time in order.	John H. Welsch, Stanford Computation Center	Burroughs 220 BALGOL
APLOT. One-curve plot with scaling (several inputs at time).	John H. Welsch, Stanford Computation Center	Burroughs 220 BALGOL
JS CODA PLOT. Entire graph in memory simultaneously, for offline printing.	Control Data Corp.	CDC 1604

[a] These are output portions of other routines.

of lines the graph is to have, so that it may be necessary to discard or merge input items, leave blank spaces, or both (if the distribution is irregular enough). The process must also be terminated either by exceeding a numerical limit or a limit on the number of items, by a termination sentinel, or by intervention from a higher level.

A routine to deal with this problem[2] for the UNIVAC I (which assumes that items are received in order and that the numerical value of the length of the graph and the number of lines composing it have been determined) determines the numerical value associated with each printer line and then selects for graphing the input item which is closest to that value. This has as its output instructions to graph an input item as a line of graph or to suppress it; instructions to insert a line of spaces in the graph; instructions to insert marginal data (time values) in every tenth line of the graph; and termination instructions. These instructions may be executed immediately or stored for future reference.

Many routines incorporating these techniques (with authors, machine, language, etc.) are listed in Table 1. The principal variation is in the number of curves that can appear in the final graph (probably about ten curves are as many as can be put on one piece of paper without making it unreasonably cluttered).

From the considerations described above, the computers most suitable for this type of work have online printers, one alphanumeric character per word,[3] and are small enough that being printer-limited is not serious. This is a description of the type of computer (IBM 1401; Philco 2400; RCA 301; Burroughs B280, etc.) which is intended to serve as an input-output satellite to a large computer. As one of the functions of such computers is the reduction of large masses of calculated data to a usable printed form, the preparation of printed graphs would seem to be one of their natural tasks, and it is regrettable that this application is apparently infrequent. Preparation of graphing routines for such machines might well enhance their usefulness.

Specialized Routines of Interest

Specialized routines of interest include one for the IBM 650 (E. Shahn, M. Weiss, and M. Berman, National Institutes of Health), which in conjunction with the graphing operation decomposes a curve which is the sum of a set of exponentials into its component exponential curves. This type of analysis is particularly useful in physiological compartmentation work.

A graphical method for finding roots of polynomials (to be used as a last resort when customary procedures break down) has been prepared by R. P. Rich (Applied Physics Laboratory, Johns Hopkins University). The printer page is used to represent an area in the X,Y-plane of the argument $z = X + iY$ of the function $f(z)$, whose

[3] The CDC 160 can also be used in this mode.

roots are to be found. One of four digits is printed at each character position to indicate which of the four possible combinations of the real and imaginary parts of the function $f(z)$ exists at the corresponding mesh point. The locus of zeros of the real and imaginary part are then drawn in manually. A more detailed representation of the neighborhood of the intersections of these curves is then obtained by taking smaller values of ΔX and ΔY, with reiteration until sufficient precision is reached in locating the roots of the function $f(z)$.

Current Status of IPL-V for the Philco 2000 Computer (June 1962)

STUART S. SHAFFER
System Development Corporation, Santa Monica, California*

The initial version of IPL-V for the Philco 2000 has been completed and is now operating on the computer. This model, IPLT-1, contains the loader, interpreter, output, housekeeping functions, and some of the primitives of the 7090 version.

This system was written mainly in JOVIAL, a procedure-oriented language, and was compiled to produce a Philco TAC binary operating deck. Certain portions were written in Philco TAC language to speed up the operation of the system during execution and to facilitate handling of primitives which in JOVIAL would have been uneconomical. These assembly-language coded sections constitute less than 3 percent of the system.

The present model assembles IPL program cards at the rate of 500 per minute in the listing mode. This compares favorably with the 7090 model, since System Development Corporation's configuration of the Philco computer operates at approximately one-fifth the speed of the 7090. Interpretation time also compares favorably with the 7090 model taking into account the difference in speeds of the two computers. The package of J primitives produced by H. A. Simon is being used and assembled with the IPL program. With listing suppressed, assembly time of these routines is only a few seconds. Input to the Philco 2000 is by card, and the system operates as a program executed in phase three of the SL Philco operating system. The J165, J166 dump and reload features are working, and this technique can be used to save an assembled IPL program on tape for later use. Unused region assignments are not returned to available space; therefore additional routines may be added with new data for reload and run from a dump tape. However, card input is extremely fast (2000 cards per minute) and normal operating procedure at SDC is card input.

J's available in SDC's model include all but a few of those available to the 7090 version. Floating-point arithmetic or the use of auxiliary storage for routines has not been implemented. It is expected that both of these will be added as needed. Since the 7090 version was translated almost literally into JOVIAL, the addition of these features is not an expensive matter.

The total set of J's described in [1] which are not available in the present model include: J102, J105, J106, J107, J108, J122, J123, J124, J126, J127, J128, J138, J140, J141, J142, J143, J144, J145, J146, J160, J161, J167 and J170. Many of these will be added soon. In addition to the standard list of J's, the J190's have been incorporated in this system. These permit the modification and testing of P, Q, SYMB, and LINK of the IPL word and are useful in self-programming applications.

The Philco 2000 used for SDC research has a 16K magnetic core memory, 9 tape units, and a 32K drum. IPLT does not, at present, make use of the drum. The system includes an available-space list which has approximately 7000 cells. There is some unused memory, but with the present Philco 2000 configuration, a high-speed available space list of much more than 8000 cells is not anticipated even in the later models of IPLT.

IPL-V has also been implemented on the 2000 as an expansion of the assembly language (TAC), see page 484 of this issue. These two methods of implementing IPL-V are currently being studied. The result of this study will be presented in the near future.

REFERENCE

1. NEWELL, ALLEN (Ed.) *Information Processing Language-V Manual.* Prentice-Hall, Englewood Cliffs, N. J. (1961).

* Artificial Intelligence Staff, Research Directorate.