LA-UR 87-1270 (Final)

CONF-8704165--3

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

LA--UR--87-1270

DE87 008997

TITLE: DYNAMIC RUN-TIME GRAPHICS

AUTHOR(S): James Tabor, C-10

SUBMITTED TO: To be presented at the Spring Cray User Group Meeting, New York, April 22-26, 1987.

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.

By acceptance of this article, the publisher recognizes that the U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes.

The Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy

MASTER

Los Alamos National Laboratory
Los Alamos, New Mexico 87545

FORM NO 836 R4
ST NO 2629 5/81

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DYNAMIC RUN-TIME GRAPHICS

by

James Tabor*
Los Alamos National Laboratory
Los Alamos, NM 87545

ABSTRACT

The major components of the Instant Graphics Package (IGP) have been integrated with the Dynamic Debugging Tool (DDT). IGP and DDT can produce 1D, 2D, and 3D plots; 2D contour plots; and array-versus-array plots at run time for a CTSS controllee memory image. These plots can be used to graphically inspect any array and to satisfy many of the user community's routine graphics requirements. A brief overview of the USERPORT feature, which can dynamically interface to any fragment of a CTSS controllee, will be presented.

1. INTRODUCTION

The Instant Graphics Package (IGP),[1] developed in 1985, is a user friendly run-time graphics package that relies on descriptive information that resides in a symbol table. Since the development of IGP, it was realized that the IGP concept needed to be promoted into a more comprehensive software package such as an interactive debugger. That goal has now been accomplished by merging the graphic components of IGP with the Dynamic Debugging Tool (DDT).[2]

DDT was originally developed at Lawrence Livermore National Laboratory and has been used at Los Alamos for over a decade. During the past two years, extensive local modifications and enhancements has been made to DDT for the purpose of recognizing the Los Alamos multitasking environment and to meet the needs of the 3D code project.[3]

Many of the new features implemented in DDT were designed mainly to differentiate a uniprocess and multitasking system. However, two distinct features have surfaced that have important universal applications in the computer software area. The first is run time graphics, which is the main subject of this paper, and the second is the USERPORT feature,[4] which will be discussed briefly in Section 2.

In this paper the details of IGP and its merger with DDT will be discussed. In most instances the terms IGP and DDT can be used interchangeably.

2. THE USERPORT FEATURE

Though USERPORT is not directly related to this paper, its importance needs to be mentioned at every opportunity. The USERPORT feature in DDT provides the mechanics to access information in software fragments (libraries, memory managers, etc.) that compose a software entity such as a CTSS controllee. The developer of a specific software fragment can design characterization routines that provide information that otherwise is very difficult to obtain or understand. In turn, DDT can dynamically port to these routines

* This work was performed under the auspices of the U. S. Department of Energy.

at any point during execution and display the current state (condition) of these software fragments and their relationship to the software entity. The USERPORT feature is open ended and can be applied to all types of software.

3. THE IGP

Before the development of IGP, computer graphics, as used in an applications code, required many resources, and the process that generated them often could be tedious and redundant. Even to produce a simple one-dimensional plot required knowledge of a graphics library and required several compile and link passes before the final product was reached. More complex plots, such as 2D contours and 3D perspectives, presented a much more difficult task.

The concept and implementation of IGP represented a giant stride forward by making available a comprehensive run time graphics package that required no graphics knowledge to produce a plot and would satisfy many graphics requirements of the general user community. In addition, graphic constructs are effectively decoupled from a code, which reduces compiles, links, and memory requirements and increases overall performance and efficiency of a software system.

IGP was designed to be a stand-alone utility that could produce a variety of plots, at run time, for any array in a CTSS controllee. Much like a debugger, IGP relies on symbol table information such as array addresses, types (integer, real), and classes (pointers, stack, normal).

A lot of flexibility has been built into IGP, especially in the axes manipulation and curve drawing areas. The x, y, and z axes can be scaled linear or log. Curves can be drawn as solid lines, dots, chains, dashes, chain dots, and chain dashes connecting every point or at specified intervals. Also, instead of linearly connecting points, there are bar chart and histogram options. Although there is default axes labeling, each axis label may be altered to suit the viewer. All plots will appear in full color if output is sent to a 35-mm device such as the FR80. In the case of 2D contour plots there is a one-to-one correspondence between the colors of numbers and symbols to the plot curve. Three-dimensional images can be rotated in horizontal and vertical directions.

The graphics routines used in IGP were constructed from the DISSPLA graphics library.[5] Each routine was compartmentalized to produce a specific type of plot. These routines include a graphics driver that decides what type of plot has been requested and calls one or more of the following primary routines:

SCALIT	(scale axes)
MPLOT	(plot array versus array)
LINLIN	(linear x-axis, linear y-axis)
LOGLIN	(log x-axis, linear y-axis)
LINLOG	(linear x-axis, log y-axis)
LOGLOG	(log x-axis, log y-axis)
CONTOUR	(2D contour plots)
PLOT3D	(3D perspective plots)
AOPER	(perform arithmetic on array before plotting)

Scaling for the z-axis (vertical) on 3D plots may also be either linear or log.

For device initialization, termination, plot disposal, and color, the Common Graphics System library[6] is used.

Table 1. summarizes the commands used in IGP.

Table 1. IGP COMMAND SUMMARY

COMMAND	FUNCTION
3D	Plots 2D array as a 3D perspective plot.
CONTOUR	Enables contour plot mode.
CROSAXES=on/off	Crosses axes at zero values.
DATE=on/off	Writes load date of controllee on plot title.
DIMI d1 d2	Presets upper pointer dimensions for arrays.
DIMI1D d1 d2 ad	Sets 2D strides for a 1D array at address.
LCURVE=interp	Sets interpolation mode for plot curve--linear, step, or bar.
LIN=x, y, or z	Scales x, y, or z axis linear.
LINETYPE=type	Sets plot curve type--solid, dot, dash, chndot, or chndsh.
LOG=x, y, or z	Scales x, y, or z axis log.
LSPACE=ispace	Sets point interval for plot curve.
MARKER=isym	Sets symbol marker for plot curve.
PHI=angle	Sets horizontal view angle for 3D plots.
PLOTFILE=name	Names the user-specified metafile.
POUT	Directs plots to specified metafile.
PTTY	Plots at graphics terminal.
TITLE3D="label"	Creates user-specified label for 3D plot title.
THETA=angle	Sets vertical view angle for 3D plots.
VPLOT	Plots array versus array.
XLABEL="label"	Creates user-specified label for x-axis.
YLABEL="label"	Creates user-specified label for y-axis.
ZLABEL="label"	Creates user specified label for 2D plot title.
ZSTEP interval	Indicates contour interval value.

4. IGP MECHANICS

IGP performs on what is known as a drop file, which contains the memory image of the last execution interval (swapped out). Every variable or array has a name. Normally with debuggers, when a variable or array name is entered on the input line, the value or values of the variable or array will be displayed. IGP takes this one step further for arrays. If an array name is entered on the IGP input line, all elements of that array are displayed as a graphics plot. This assumes a graphics terminal is being used. If not, the plots can be directed to a device-independent metafile for postprocessing.

The type of plot produced by IGP is dependent on options invoked. At start-up time the default is linear-linear. In all cases, save for array-versus-array plotting, the x-axis is scaled by the array indicies and the values of the array are plotted on the y-axis.

If the plot mode (POUT or PTTY) is set in IGP, the mere mention of an array name will produce a plot. This is the simplest plot mode in IGP. The examples[7] that follow demonstrate many graphics capabilities and the mechanics of how to use IGP.

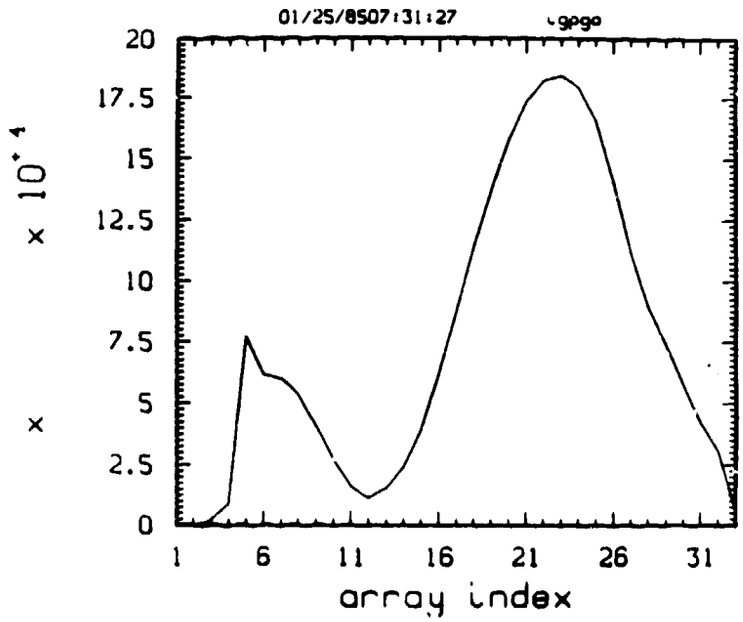
```
subroutine mix
dimension x(33), y(33,23), a(33)

    .
    .
    .
end
```

Assume that subroutine mix is part of an executable controllee called xfile and that arrays a, x, and y have been arithmetically massaged.

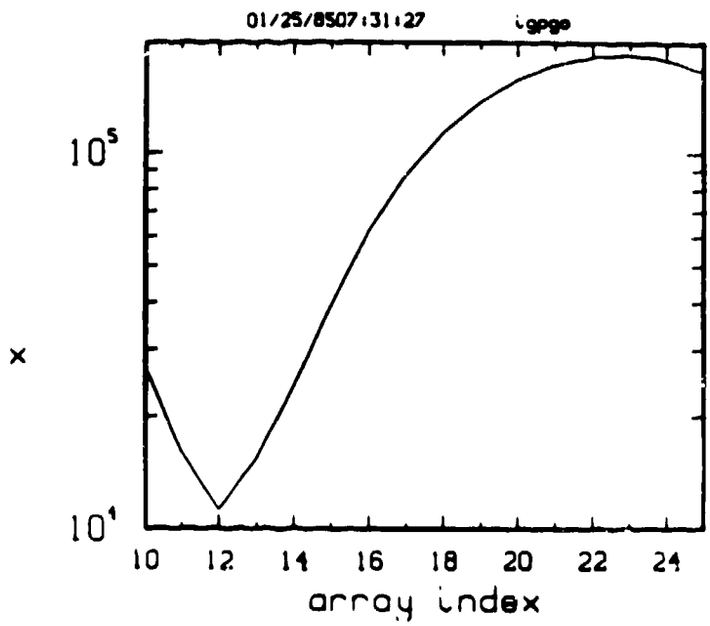
```
IGP XFILE

?PTTY   (or POUT)

?SUB MIX
?X 1 33            (example 1)

?LOG=Y
?X 10 25           (example 2)

?LIN=Y
?Y 1 1 1 23        (example 3)

?LOG=Y
?Y 7 26 15 15      (example 4)

?PHI=180
?THETA=10
?Y 1 33 1 23 3D    (example 5)

?PHI=90
?THETA=30
?Y 5 20 10 20 3D   (example 6)
```

```
?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y 1 1 33 1 23        (example 7)

?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y -1 1 33 1 23       (example 8)

?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y 2 1 33 1 23        (example 9)

?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y -2 1 33 1 23       (example 10)

?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y 3 1 33 1 23        (example 11)

?CONTOUR
Enter Contour Interval
?4.5e+4
Enter 2D Array Name to be Contoured
?Y -3 1 33 1 23       (example 12)

?VPLOT
Enter Array Name to be plotted on x-axis
?X 1 33
Enter Array Name to be plotted on y-axis
?Y 1 33 10 10         (example 13)

?X / A 1 33           (example 14)

?X * 10 1 33          (example 15)

?Y + A 1 33 1 1       (example 16)

?Y - X 1 33 1 1       (example 17)
```

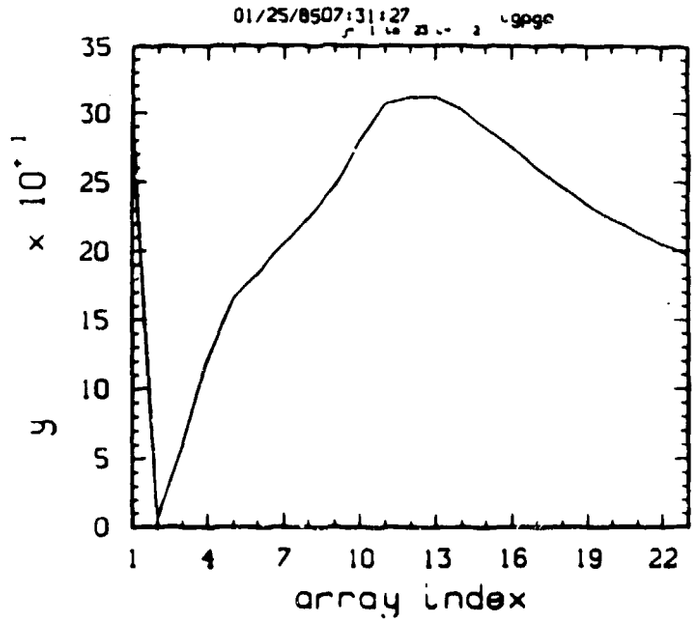
Notice examples 14 through 17, which allow arithmetic massaging before plotting.



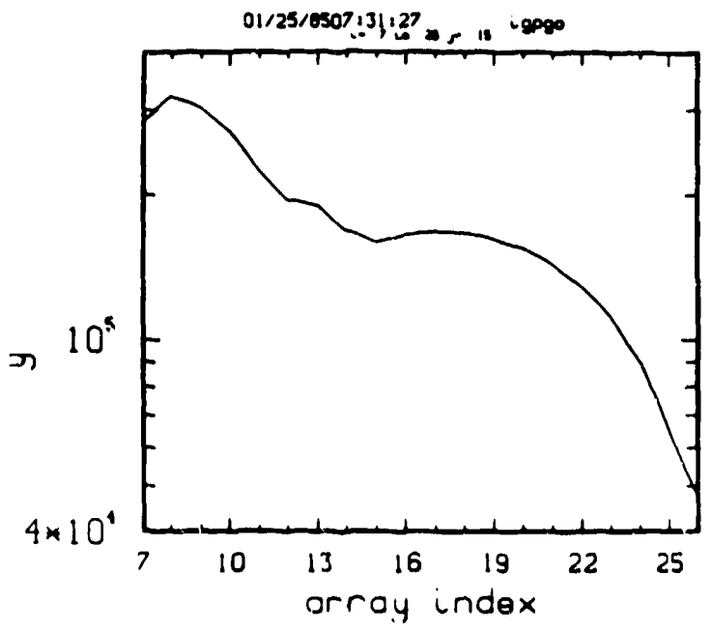
Example 1.



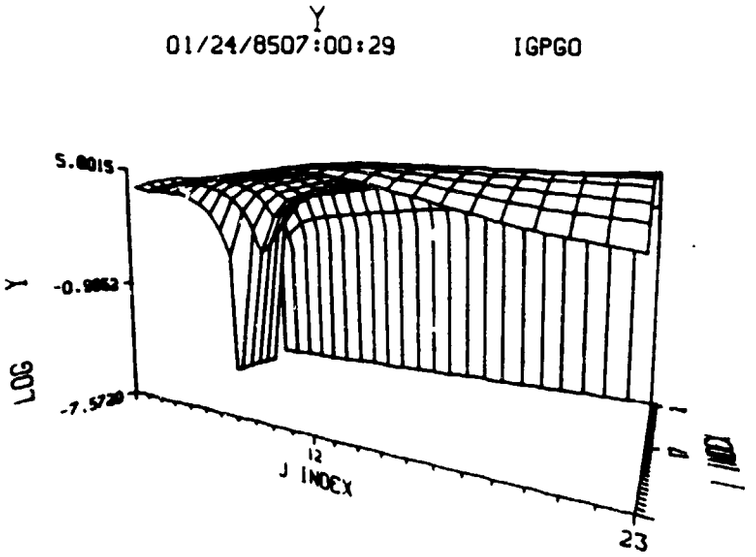
Example 2.



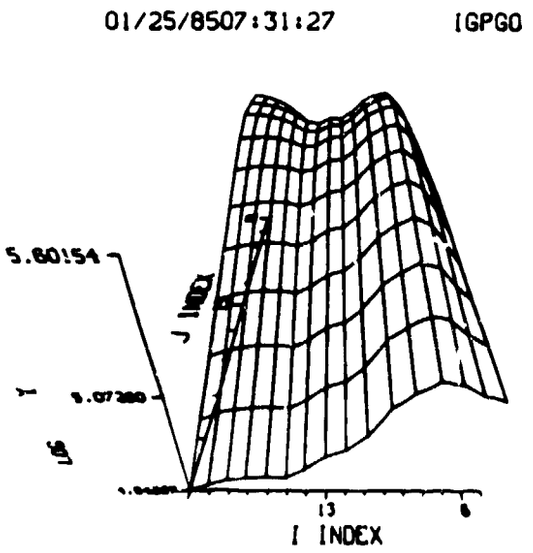
Example 3.



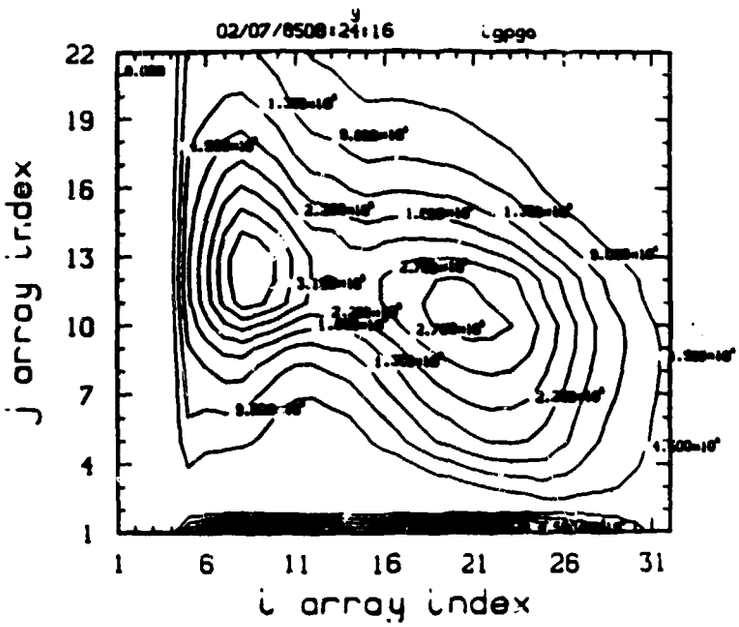
Example 4.



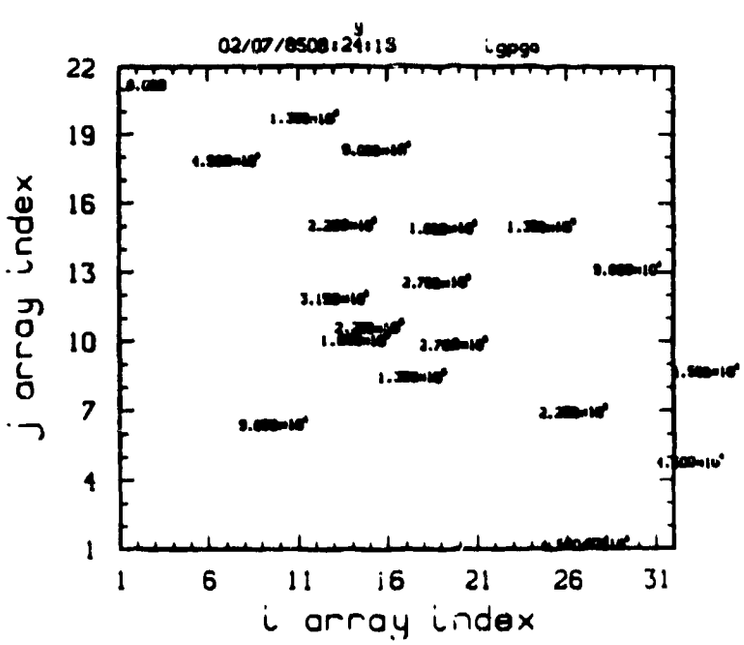
Example 5.



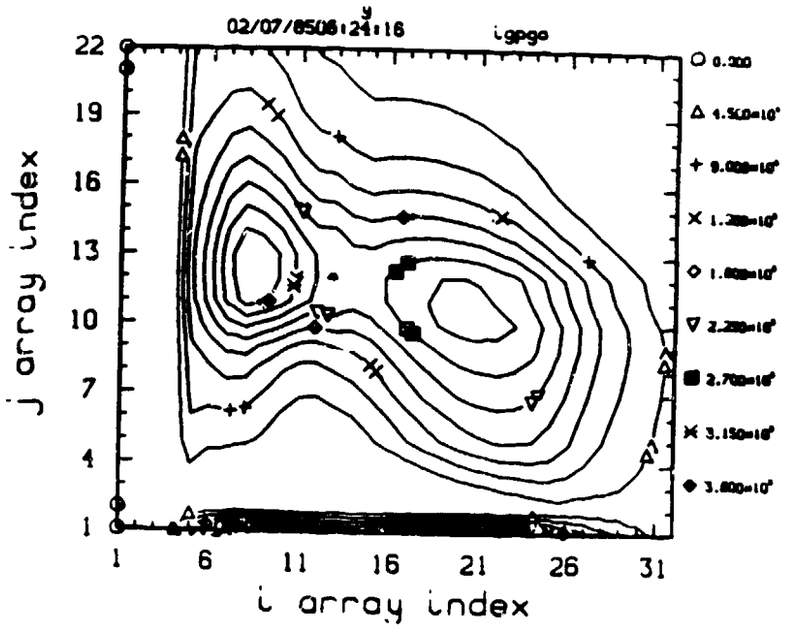
Example 6.



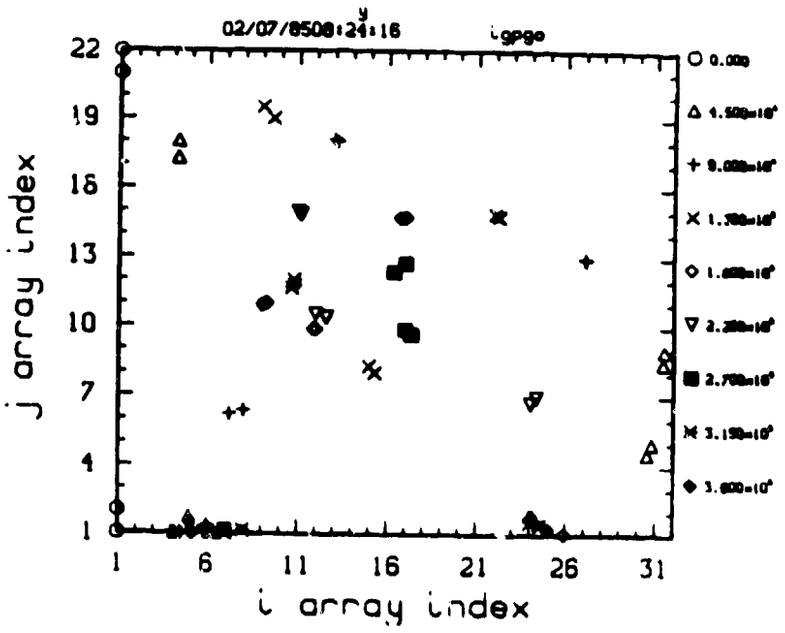
Example 7.



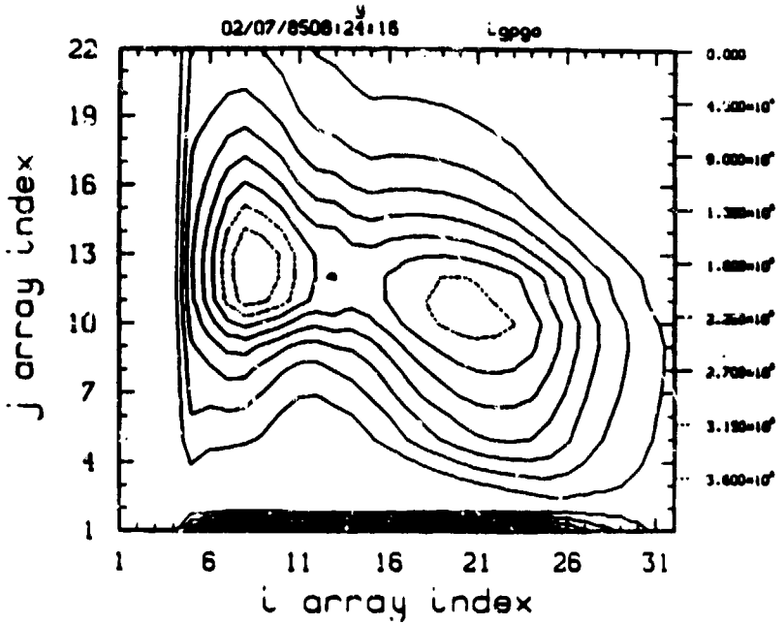
Example 8.



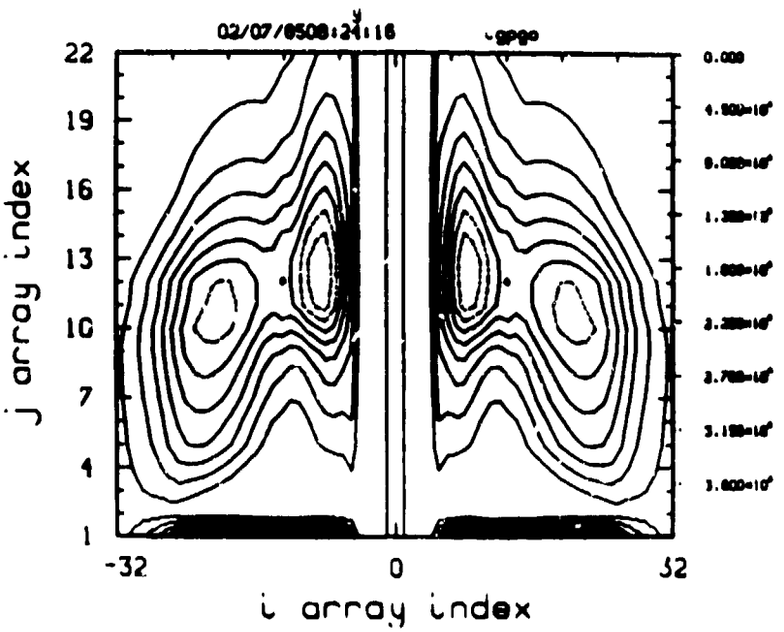
Example 9.



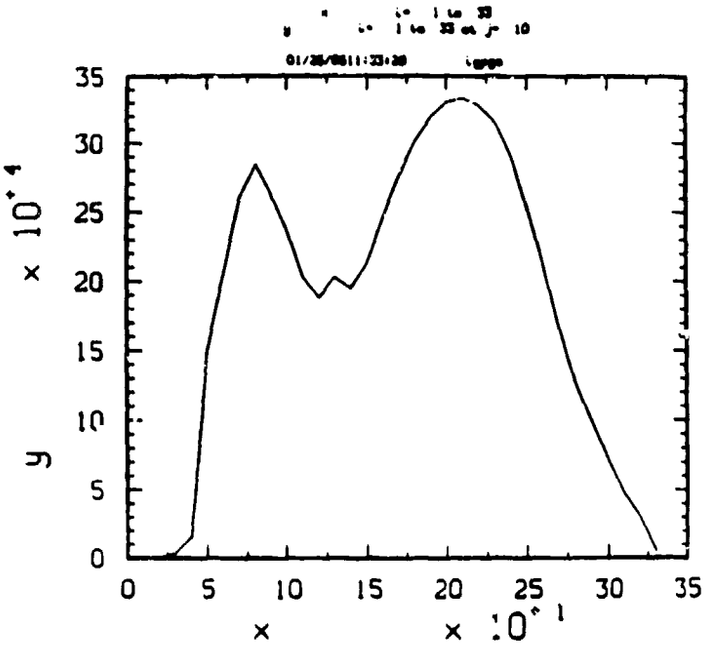
Example 10.



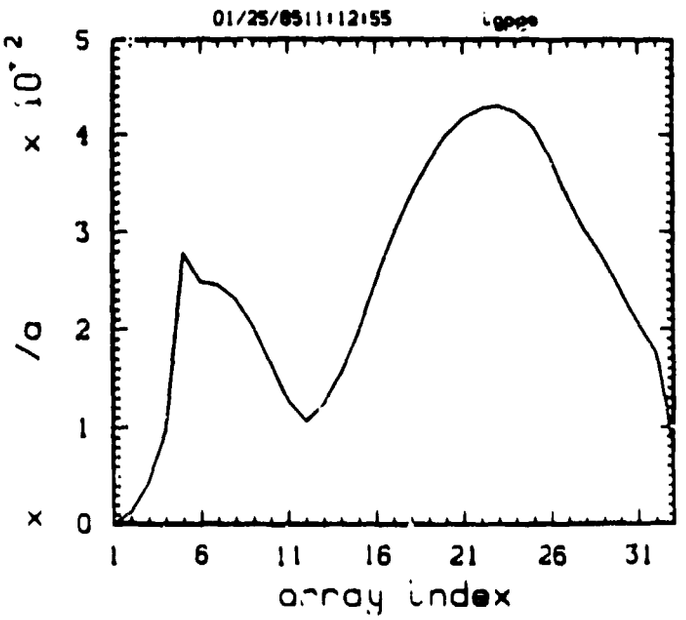
Example 11.



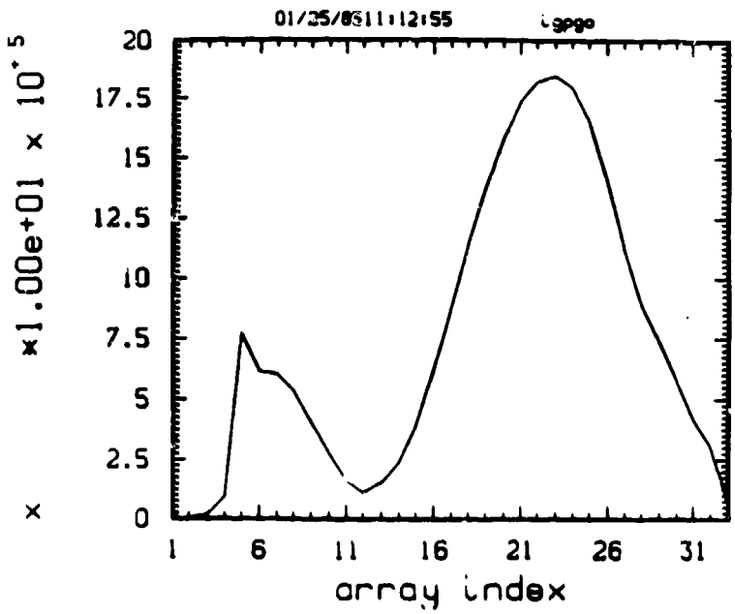
Example 12.



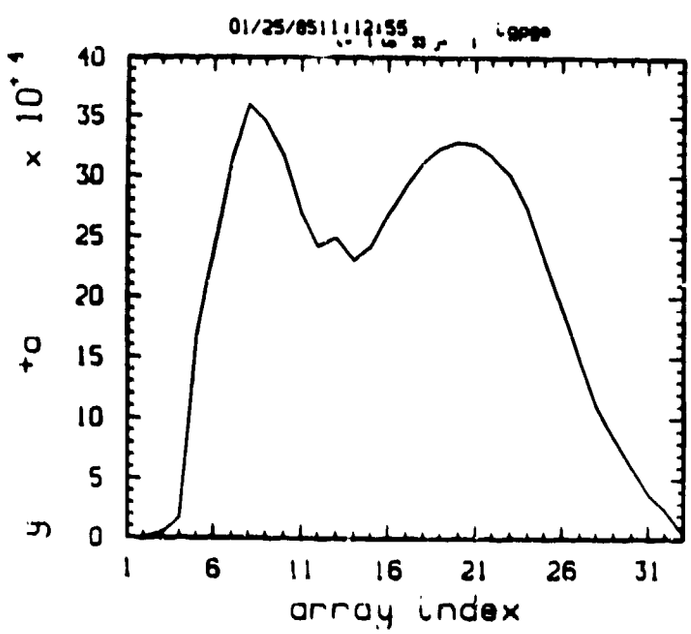
Example 13.



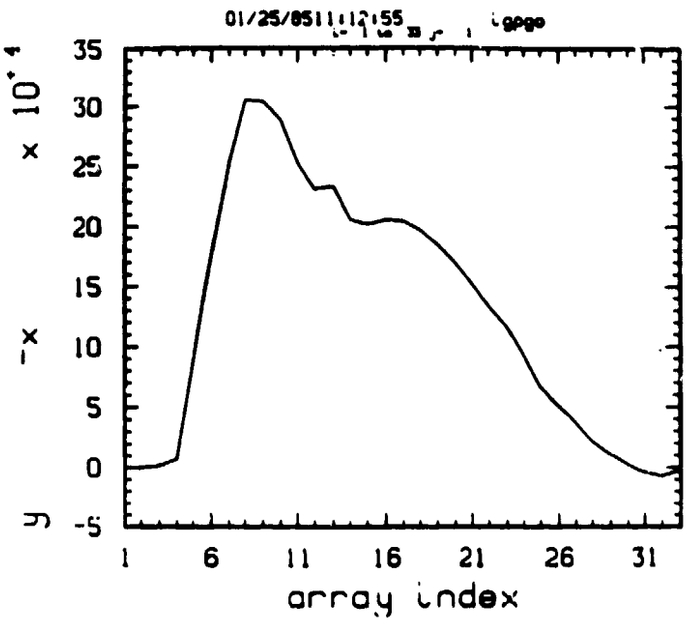
Example 14.



Example 15.



Example 16.



Example 17.

5. IGP USES

Scientists, engineers, code developers, and practically all applications can benefit from IGP. Examples of the kind of information that can be plotted by IGP include static cross-section data from a Monte Carlo application; grid variables such as density, pressure, and temperature from a time-dependent hydrodynamics code; and atomic physics data.

6. IGP TO DDT EVOLUTION

It was known from the beginning that if IGP was to be optimally used, its features must be combined with a commonly used debugger such as DDT. DDT, as it stood, was an extremely powerful tool, and functions such as breakpointing were necessary to use the graphics features effectively.

Rather than redesign a graphics package that would be customized for DDT, the initial approach was to merge IGP with DDT with the least amount of effort. This required removing all redundant features, rewriting memory management, including IGP commands in the DDT grammar, and converting the DDT symbol table data structure to the IGP data structure.

Now that the merger is complete, DDT can be viewed as more than just a debugger. The total package represents a comprehensive, user-friendly, run-time controller that avails most aspects of the CTSS environment.

7. SUMMARY

The merger of IGP and DDT represents a quantum jump not only in graphical debugging capability but also in user-friendly software in general. In the future, more software will be designed, based on symbol

table information, that condenses the effort to produce a product and increases the overall efficiacy of a computing environment such as CTSS.

REFERENCES

1. Reference for the Instant Graphics Package (IGP), Los Alamos National Laboratory, CIC #778, December 1985.
2. Dynamic Debugging Tool (DDT) Reference with the Instant Graphics Package (IGP), Los Alamos National Laboratory, CIC #221, January 1987.
3. James E. Tabor, "Dynamic Debugging for Multitasking," Los Alamos National Laboratory unclassified release LA-UR-86-1435, Proceedings of Spring 1986 Cray User Group.
4. James E. Tabor, "Advances in Dynamic Debugging at Los Alamos," Los Alamos National Laboratory unclassified release LA-UR-87-1269, to be published in the Proceedings of Spring 1987 Cray User Group.
5. DISSPLA User's Manual, ISSCO Graphics.
6. CGS Graphics, User Manual, Los Alamos National Laboratory, December 1985.
7. Learning to Use the Instant Graphics Package (IGP), Los Alamos National Laboratory, CIC #779, July 1985.