1987

Three Dimensional Graphics Software in Interactive ELLPACK

John P. Bonomo
Wayne R. Dyksen

Report Number:
87-674
THREE DIMENSIONAL GRAPHICS SOFTWARE
IN INTERACTIVE ELLPACK

John P. Bonomo
Wayne R. Dyksen

CSD-TR-674
April 1987
Three Dimensional Graphics Software in Interactive ELLPACK

John P. Bonomo †
Wayne R. Dyksen ‡

†Supported in part by National Science Foundation grant CCR-8612817
‡Supported in part by National Science Foundation grants CCR-8612878 and DCR-8602385
Three Dimensional Graphics Software in Interactive ELLPACK

John P. Bonomo
Wayne R. Dyksen

CSD-TR 674

Abstract

In this document we describe the implementation and use of 3-dimensional graphics software in the Interactive ELLPACK system. Section 1 gives an introduction and overview of the software. Section 2 gives a brief description of ELLPACK and Interactive ELLPACK. Section 3 describes the method used to represent a 3-dimensional function on a two dimensional surface. Section 4 presents the user interface and describes the various features of the software that are under the control of the user. Section 5 outlines the software itself and Section 6 explains procedures with which the user can tailor the 3D system for a particular application.

1 Introduction

In this document we describe the implementation and use of 3-dimensional graphics software in the Interactive ELLPACK system ([1]). The software presents a high-level interactive interface allowing the user to draw a 3-dimensional graph of any ELLPACK generated or user supplied function of two variables. The interface also allows control over numerous aspects of the graphic output such as the location of the graph origin or the printing of an optional color index table. The 3D graphics software can plot functions defined on both rectangular and non-rectangular domains, possibly with holes in them. The user accesses this plotting software via the Interactive ELLPACK module plot3d.

The plot3d module was designed with two objectives in mind: ease of use and speed. Most of the specifications of the viewing volume are determined automatically by the software – the user

---

1Supported in part by National Science Foundation grant CCR-8619817
2Supported in part by National Science Foundation grants CCR-8612676 and DCR-8602385
needs only provide a function of two variables to be graphed. Once a primary graph is displayed, the user can then modify various parameters of plot3d to obtain different views of the plotted function. In most cases, however, the software makes an acceptable plot of the user's function. This approach is especially useful in an environment where the user wants to view a large number of functions in order to select the most informative for further study.

Section 2 gives a brief description of both ELLPACK and Interactive ELLPACK. Section 3 describes the method which the plot3d module uses to represent a 3-dimensional function on a two dimensional surface. Section 4 presents the user interface and describes the various features of the 3D software that can be controlled interactively by the user. Section 5 describes the software itself, and Section 6 explains procedures with which the user can tailor the 3D system for a particular application.

2 ELLPACK and Interactive ELLPACK

ELLPACK is a very-high level computer language for solving elliptic partial differential equations (PDEs) ([3]). The basic building blocks in an ELLPACK program are segments which perform various tasks necessary to define and solve a PDE. The equation and boundary segments define the PDE and the problem domain. The grid, discretization, indexing and solution segments specify the discretization and solution technique to be used in solving the PDE. Special triple segments exist which combine the latter three segments into one. Other segments allow the user to select specific output modules and to include FORTRAN code and subroutines in the ELLPACK control program.

Though ELLPACK was initially developed as an environment for evaluating the performance of algorithms and software for elliptic PDEs, it is now recognized as a very powerful tool for solving a large class of elliptic problems. Besides its basic function of solving second order, linear elliptic PDEs with Dirichlet, Neumann, mixed or periodic boundary conditions, ELLPACK can be used to solve nonlinear problems, time dependent problems and problems involving systems of elliptic equations.

Interactive ELLPACK ([1]) is an extension of ELLPACK which includes several important new features:

1. a menu segment to build user designed menus to allow the run time selection of ELLPACK modules;

2. an interactive grid module which allows the user to view, specify and change grids via interactive graphical devices throughout the execution;

3. new 3-dimensional color graphics output modules.
A sample Interactive ELLPACK program is shown in Figure 1. It solves the following elliptic problem:

\[ \begin{align*}
    u_{xx} + u_{yy} + 20\pi^2 u &= 0 \\
    u &= 0 \\
    u_y &= 4\pi \sin(2\pi x) \\
    (x, y) &\in (0, 1) \times (0, 1) \\
    x &= 0, 1, \quad y = 0
\end{align*} \]

The first allows the user to define the grid using the interactive grid module and to select one of two solution methods. The second allows the user to graph any of several functions and to control the placement of the resulting plots on the screen. There is no limit to the number of menus that can be included in an Interactive ELLPACK program. A sample display from this program is shown in Figure 2 and an example of the 3D graphics output is shown in Figure 3.

3 Graphics Algorithm

The basic task of any 3D graphics software is to render a function defined in 3-dimensional world coordinates (x-y-z space) into a planar representation displayed in 2-dimensional device coordinates. In performing this conversion, the software makes use of an intermediate viewing coordinate system, the \( u-v-w \) system. The viewplane (defined by \( w = 0 \)) is "parallel" to the terminal screen in which the user views the graph. The exact 3D-to-2D conversion is dependent on several factors:

1. Whether or not a parallel or perspective projection is used.
2. The location and size of the viewing volume in world coordinates.
3. The location and orientation of the 3D surface with respect to the observer.
4. Whether or not hidden surfaces and hidden lines are removed.

The algorithm used by plot3d has the following characteristics:

1. A parallel orthographic transformation is used; i.e., the center of projection is "at" infinity and the direction of projection is identical to the viewplane normal.
2. No front or back clipping is performed, which reduces the problem of specifying the 3D viewing volume to that of specifying a 2D viewing window on the viewplane (the \( uv \) window). The location and size of the \( uv \)-window is determined automatically so that the planar representation of the surface is positioned evenly on the terminal screen viewport.
3. Since an orthographic projection is used, the exact location of the viewplane is unimportant. Plot3d assumes that the \( x-y-z \) origin and the \( u-v-w \) origin are identical. The orientation of
Figure 1: An Interactive ELLPACK program to explore the use of different methods (5 point star and hodie fft) and grids in solving an Helmholtz problem.

```plaintext
options.
  max x points = 33  max y points = 33
  interpolation = splines
  terminal = tek

equation.
  uxx + uyy + (20*pi**2)*u = 0

boundary.
  u = 0 on x = 0
  u = 0 on x = 1
  u = 0 on y = 0
  u = 0 on y = 1

menu. 'Solution Menu'
  'ig:interactive grid'  grid. interactive
  'fd:ordinary finite differences'  sol. band ga
  'hf:hodie fft'  tri. hodie fft

menu. 'Output Menu'
  'ct:contour true'  out. plot(true)
  'cu:contour u'  out. plot(u)
  'ce:contour error'  out. plot(error)
  'ca:contour abserr = abs(error)'  out. plot(abserr)
  'gt:graph true'  out. plot3d(true)
  'gu:graph u'  out. plot3d(u)
  'ge:graph error'  out. plot3d(error)
  'ga:graph abserr = abs(error)'  out. plot3d(abserr)
  'mv:move plot from view to view'  out. move view
  'cv:copy plot from view to view'  out. copy view
  'dv:delete plot from view'  out. delete view
  'ev:enlarge views'  out. enlarge views
  'pp:put function plot to a file'  out. put plot
  'gp:get function plot from a file'  out. get plot
  'og:overlay grid'  out. overlay grid

subprograms.
  function true(x,y)
    common / c1regl / x1reg, y1reg, pi
    true = sin(2*pi*x) * sin(4*pi*y)
    return
  end
  function abserr(x,y)
    abserr = abs(error(x,y))
    return
  end
end.
```
Figure 2: Interactive ELLPACK Display. Nine of the ten viewports are in use, showing calculated function values and absolute errors using two grids.
Figure 3: Sample 3D Graphics Output. This is an enlargement of viewport 10 in the previous...

Color Value
1.03E-01
3.15E-01
3.35E-01
3.68E-01
3.98E-01
2.45E-01
2.10E-01
1.93E-01
1.75E-01
1.58E-01
1.40E-01
1.23E-01
1.06E-01
8.76E-02
7.01E-02
5.26E-02
3.51E-02
1.75E-11
1.58E-11
1.41E-11
1.24E-11
1.07E-11
9.0E-11
7.3E-11
5.6E-11
3.9E-11
2.2E-11
1.5E-11
8.7E-11
1.11E-11
1.21E-11

...
x-y-z space relative to u-v-w space is specified by two angles: $\phi$, the angle about the screen perpendicular (the w-axis) and $\theta$, the angle about the horizontal (the u-axis).

4. Hidden surfaces and lines are removed.

Prior to implementing the graph algorithm, plot3d first sets up an array of $z = f(x,y)$ values. The values of $x$ and $y$ vary over the user-specified ranges $[x_{min}, x_{max}]$ and $[y_{min}, y_{max}]$ (These correspond to the internal ELLPACK variables [r1axgr, r1bxgr] and [r1zygr, r1bygr]). The cross product of these two ranges defines the $xy$ domain. The size of the $z$ array is dependent upon the number of $x$ and $y$ grid lines which the user specifies when calling the plot3d module; these two values determine the fineness of the plot. Simultaneous to filling in this array, plot3d also determines values for $z_{min}$ and $z_{max}$, which are used in later calculations. The plot3d algorithm then proceeds in three steps:

1. It first calculates a $z$ scaling factor $= \sigma_z$, so that the value $\sigma_z(z_{max} - z_{min})$ is the same order of magnitude as the $x$ and $y$ domain dimensions.

2. The algorithm then determines the size and location of the uv-window.

3. Finally, the algorithm applies the 3D-to-2D transformation to the surface and then maps the results from the uv-window to the terminal viewport.

3.1 Framing

Plot3d leaves a certain minimum percentage of space around the edge of the Interactive ELLPACK viewport empty to frame the plotted surface. This also allows space for text to be written on the viewport if necessary (see Section 3.7). The amount of framing performed is controlled by two variables, $\rho_u$ and $\rho_v$, where

$$
\rho_u = \text{minimum percentage indent in the } u \text{ direction},
$$

$$
\rho_v = \text{minimum percentage indent in the } v \text{ direction}
$$

(The user has access to parameters through two internal Interactive ELLPACK variables, rhou and rhov. See Section 6.2). An example of this is shown in Figure 4, where vlen$u$ and vlen$v$ are the lengths (in device coordinates) in the $u$ and $v$ direction of the screen viewport, respectively.

In order that the graph not be distorted when performing the window-to-viewport transformation, both the uv-window and viewport must have the same ratio of $u$ length to $v$ length. This ratio, called the aspect ratio, is given by

$$
\alpha_{opt} = \frac{\text{vlen}_u}{\text{vlen}_v}.
$$
When the $uv$-window is created, it is constrained to have an aspect ratio equal to $\alpha_{vpt}$.

When plot3d calculates the $z$ scaling factor and the size of the $uv$-window it uses the aspect ratio of the framed viewport (the dashed box in Figure 4) which is given by

$$\alpha_{frm} = \frac{(1 - 2\rho_u) vlen_u}{(1 - 2\rho_u) vlen_v}. \quad (1)$$

### 3.2 Location of Axes

When drawing axes on the plot, it is not always desirable to draw them in the plane defined by $z = 0$. For example, assume the user knows that the function values to be plotted lie between $z = 100$ and $z = 200$. If the axes were drawn in the $z = 0$ plane, the plot of the surface would be confined to the upper half of the viewport, since space would have to be made for $z$ values ranging from 0 to 200. If, on the other hand, the axes are drawn in the $z = 150$ plane, then only $z$ values
from 100 to 200 need be plotted, and the surface would more fully fill the viewport.

The location of the axes is controlled by the parameter \( z_{\text{int}} \). Plot3d subtracts this value from all function values prior to plotting them. In the above example, plot3d would plot \( z \) values between -50 and 50, with the axes plotted at (an adjusted) 0. The value of \( z_{\text{int}} \) is controlled interactively by the user (See Section 4).

3.3 3D-to-2D Transformation

In order to specify the orientation of the \( x-y-z \) coordinate system relative to the observer, two directions must be specified:

1. the view plane normal (VPN) which is perpendicular to the viewplane and points away from the observer. The \( w \) axis of the viewing coordinate system is identical to the VPN.

2. The view up vector (VUP), which specifies the \( v \) axis of the viewplane. Although the VUP may, in general, make any angle with the the VPN, plot3d makes the simplifying assumption that the VUP lies perpendicular to the VPN, i.e. the VUP and the \( v \) axis are identical.

Once these two directions have been set, the 3D-to-2D transformation function can be derived. Plot3d provides a simple and intuitive method to specify the orientation of the 3D surface, an example of which is is shown in Figure 5. The left-handed \( u-v-w \) coordinate system is initially considered identical to the initial right-handed \( x-y-z \) coordinate system, save that \( w \equiv -z \). Two rotation angles are then specified \((\phi, \theta)\) and the newly rotated surface is displayed on the screen. Note that the manner in which the rotations are applied is consistent with the assumption that the observer (and the \( u-v-w \) coordinate system) is fixed while the 3D surface is rotated. We feel this is the most natural way to specify the rotation; when a person has an object in his hand and wishes to view the back of it, he would more likely rotate the object then keep it fixed and walk around it. Figure 6 shows the orientation of both the VPN and the VUP with respect to the \( x-y-z \) coordinate system after the \( \phi \) and \( \theta \) rotations have been performed.

We now develop the equations necessary for the 3D-to-2D transformation (these equations are simplifications of the derivation in [2] since plot3d makes several simplifying assumptions about the specification of the viewing volume). With no rotation at all, an arbitrary point \((x, y, z)\) would be represented by the projected point \((x, y)\). In general for a given \( \phi \) and \( \theta \), we seek a projection \( T_{\phi, \theta} : \mathbb{R}^3 \rightarrow \mathbb{R}^2 \) which converts a 3-dimensional point in world coordinates to a 2-dimensional point in the \( u-v \) plane (the viewplane). To find \( T_{\phi, \theta} \) we first change from a right-handed to a left-handed coordinate system and apply the rotation about the screen perpendicular (Figure 5-b) to \((x, y, z)\).
Using an intermediate $u'-v'-w'$ coordinate system for notational convenience, we obtain

\[
\begin{align*}
    u' &= x \cos \phi - y \sin \phi, \\
    v' &= x \sin \phi + y \cos \phi, \\
    w' &= -z,
\end{align*}
\] (2)

and after applying the rotation about the $u$ axis (Figure 5-c) we obtain

\[
\begin{align*}
    u &= u', \\
    v &= v' \cos \theta - w' \sin \theta, \\
    w &= v' \sin \theta + w' \cos \theta.
\end{align*}
\] (3)

Composing (2) and (3) and projecting the result onto the $u-v$ plane we obtain the following relationships which define the 3D-to-2D transformation function $T_{\phi,\theta}$ used by the plot3d module:

\[
\begin{align*}
    u &= x \cos \phi - y \sin \phi, \\
    v &= x \sin \phi \cos \theta + y \cos \phi \cos \theta + z \sin \theta.
\end{align*}
\] (4)

While the transformation for the $w$ direction is not needed for the 2D transformation, it is needed for the determination of the orientation of the graph relative to the observer (Section 3.6). We call this transformation $W_{\phi,\theta}$ and using (2) and (3) we define it as

\[
w = x \sin \phi \sin \theta + y \cos \phi \cos \theta - z \cos \theta.
\] (5)

It should be noted that the values of $z$ used in (2), (3) and (4) are not identical to the values of the plotted function $f(x, y)$. The function values are modified using both a $z$ offset, $z_{\text{int}}$, and a $z$ scaling factor, $\sigma_z$, so that the actual plotted values have the form

\[
z = \sigma_z(f(x, y) - z_{\text{int}}).
\]

The software has default values for $\phi$ and $\theta$; after the graph is initially displayed, the user can modify these values to obtain different views of the surface.

3.4 Z Scaling and UV-Window Placement

Once we have a 2D representation of the graph on the viewplane, we must transfer that image to device coordinates. In order to do this, we define a $uv$-window in the viewplane whose contents are translated and scaled to a screen viewport. Figure 7 shows an example of this window-to-viewport transformation, where the 2D representation is mapped onto the lower right viewport on the screen (the screen configuration shown has ten viewports of varying sizes). Plot3d determines the location and size of the $uv$-window so that the 2D representation is centered in the $uv$-window (and therefore in the selected viewport as well). In order to do this exactly, the module must first
Figure 5: 3D rotations

calculate $u_{\text{min}}, u_{\text{max}}, v_{\text{min}}$ and $v_{\text{max}}$, the minimum and maximum values in the $u$ and $v$ directions of the 2D representation. This requires applying the $T_{\phi,\theta}$ transformation to all of the points of the graph before defining the $uv$-window, and then applying the window-to-viewport transformation. However, the determination of these values is complicated by the fact that a $z$ scaling factor, $\sigma_z$, is applied to all function values. The use of $\sigma_z$ is necessary to ensure that the dimension of the function surface is of the same order as the dimensions of the $xy$ domain.

Plot3d determines the value of $\sigma_z$ heuristically as follows: let $z_{\text{min}}, z_{\text{max}}, y_{\text{min}}$ and $y_{\text{max}}$ represent the respective endpoints for the $z$ and $y$ ranges of the domain in which the graphed surface is defined (note that for non-rectangular domains we are actually dealing with the enclosing rectangle). The locations in the $uv$ plane of the extreme points of the $xy$ domain in the plane $z = z_{\text{int}}$ are given by

\[
(u_1, v_1) = T_{\phi,\theta}(x_{\text{min}}, y_{\text{min}}, z_{\text{int}}),
\]
\[
(u_2, v_2) = T_{\phi,\theta}(x_{\text{min}}, y_{\text{max}}, z_{\text{int}}),
\]
\[
(u_3, v_3) = T_{\phi,\theta}(x_{\text{max}}, y_{\text{max}}, z_{\text{int}}),
\]
\[
(u_4, v_4) = T_{\phi,\theta}(x_{\text{max}}, y_{\text{min}}, z_{\text{int}}).
\]

Let

\[
u_{\text{min}}' = \min(u_1, u_2, u_3, u_4),
\]
Figure 6: Orientation of the view plane normal (VPN) and the view up vector (VUP) with respect to the x-y-z coordinate axes after $\phi$ and $\theta$ rotations. In (a) the z axis is coming out of the page, and in (b) the x axis is going into the page.

\[ u_{\text{max}}' = \max(u_1, u_2, u_3, u_4), \]
\[ v_{\text{min}}' = \min(v_1, v_2, v_3, v_4), \]
\[ v_{\text{max}}' = \max(v_1, v_2, v_3, v_4), \]

and

\[ \Delta_u' = u_{\text{max}}' - u_{\text{min}}', \]
\[ \Delta_v' = v_{\text{max}}' - v_{\text{min}}'. \]

Figure 4 shows an example of $\Delta_u'$ and $\Delta_v'$. Due to the order in which the two rotation angles are applied, one can show that all the $u$ values of the plotted surface fall between $u_{\text{min}}'$ and $u_{\text{max}}'$ (in other words, $u_{\text{min}} = u_{\text{min}}'$ and $u_{\text{max}} = u_{\text{max}}'$). Plot3d next calculates worst case values for $v_{\text{min}}$ and $v_{\text{max}}$. Let $z_{\text{min}}$ and $z_{\text{max}}$ be the minimum and maximum points on the surface to be plotted. Using both $\sigma_z$ and $z_{\text{int}}$ to adjust the $z$ values, the lowest and highest $v$ values possible are

\[ v_{\text{min}}'' = v_{\text{min}}' + \sigma_z \min(z_{\text{min}} - z_{\text{int}}, 0) \cos \theta. \]
\[ v_{\text{max}}'' = v_{\text{max}}' + \sigma_z \max(z_{\text{max}} - z_{\text{int}}, 0) \cos \theta. \]

Equation (8) gives the $v$ value assuming $z_{\text{min}}$ occurred at the boundary vertex corresponding to $v_{\text{min}}'$, while (9) gives the $v$ value assuming $z_{\text{max}}$ occurred at the boundary vertex corresponding to $v_{\text{max}}'$. The use of the max and min functions in these two equations is necessary to handle the cases when $z_{\text{int}}$ is less than $z_{\text{min}}$ or greater than $z_{\text{max}}$. Given these values for $u_{\text{min}}'$, $u_{\text{max}}'$, $v_{\text{min}}''$ and $v_{\text{max}}''$, plot3d selects a $\sigma_z$ so that the dimensions of the worst-case graph have the same aspect ratio as the framed viewport; that is so

\[ \frac{u_{\text{max}}' - u_{\text{min}}'}{v_{\text{max}}'' - v_{\text{min}}''} = \alpha_{\text{frm}}. \]
Substituting (6), (7), (8) and (9) into (10) and simplifying we obtain

$$\sigma_z = \frac{\Delta'_u/\alpha_{frm} - \Delta'_v}{\left(\max(z_{max}, z_{int}) - \min(z_{min}, z_{int})\right) \cos \theta}.$$ 

Once plot3d has obtained a value for $\sigma_z$ it can calculate values for $v_{ma~x}$ and $v_{mi~n}$. It is then possible to determine the dimensions of the uv-window, $wlen_u$ and $wlen_v$, and the location of the lower left hand corner of the window, $wloc_u$ and $wloc_v$. Let

$$\Delta_u = u_{ma~x} - u_{mi~n},$$

$$\Delta_v = v_{ma~x} - v_{mi~n},$$

$$\alpha_{2D} = \frac{\Delta_u}{\Delta_v}.$$ 

where $\alpha_{2D}$ is the aspect ratio of the 2D representation of the plotted surface. In order to calculate the dimensions of the window correctly, we must ensure that 1) the graph is not distorted due to unequal aspect ratios and 2) enough space is left around the graph to allow for the requested framing. This is accomplished as follows:

if $\alpha_{frm} \geq \alpha_{2D}$ then

$$wlen_u = \alpha_{frm} \frac{\Delta_u}{(1 - 2\rho_u)},$$

$$wlen_v = \frac{\Delta_v}{(1 - 2\rho_v)}.$$
else

\[
\begin{align*}
\text{wlen}_u &= \frac{\Delta_u}{(1 - 2p_u)}, \\
\text{wlen}_v &= \frac{1}{\alpha_{frm}} \cdot \frac{\Delta_u}{(1 - 2p_u)}.
\end{align*}
\]

In both cases it can be shown that the aspect ratio of the \( uv \)-window, \( \text{wlen}_u/\text{wlen}_v \), is equal to that of the viewport, \( \alpha_{\text{opt}} \), using either pair of equations above and (1).

The location of the \( uv \)-window follows easily once the size of the window is known. Since we want the graph centered in the viewport, we set

\[
\begin{align*}
\text{wloc}_u &= \frac{u_{\text{max}} + u_{\text{min}} - \text{wlen}_u}{2}, \\
\text{wloc}_v &= \frac{v_{\text{max}} + v_{\text{min}} - \text{wlen}_v}{2}.
\end{align*}
\]

### 3.5 Color

The use of color in \texttt{plot3d} is an integral part in the presentation of 3D surfaces. Typically colors are accessed through a color map set up on the host graphics terminal. This color map represents a mapping from some continuous subset of the integers to a (usually continuous) set of colors. At the start of an Interactive ELLPACK session, the program determines the type of graphics terminal it is running on, and from this determines \( \text{numcol} \), the number of colors available, and \( \text{begcol} \), the lowest index of the color map. The program then sets up a default color map which associates blue with \( \text{begcol} \) and progresses through green, yellow and orange to red, which is associated with \( \text{begcol} + \text{numcol} - 1 \).

When \texttt{plot3d} is called, it must define a second mapping, one which maps the function \( z \) values to the color map index set. Let the color spacing, \( \Delta_{\text{col}} \), be defined by

\[
\Delta_{\text{col}} = \frac{z_{\text{max}} - z_{\text{min}}}{\text{numcol} - 1}.
\]

Then for any \( z \) value, \texttt{plot3d} uses the following formula to determine its corresponding color map index, \( i_{\text{col}} \):

\[
i_{\text{col}} = \text{begcol} + \left\lfloor \frac{z - z_{\text{min}} + \Delta_{\text{col}}/2}{\Delta_{\text{col}}} \right\rfloor.
\]

Color map indices \( \text{begcol} + 1, \ldots, \text{begcol} + \text{numcol} - 2 \) are associated with ranges of \( z \) values of length \( \Delta_{\text{col}} \); the midpoint of each of these ranges is printed in the color table drawn with each plot. The indices \( \text{begcol} \) and \( \text{begcol} + \text{numcol} - 1 \) are associated with \( z \) ranges of length \( \Delta_{\text{col}}/2 \); the values \( z_{\text{min}} \) and \( z_{\text{max}} \) are printed in the color table for these two indices.

Once a mapping has been made between \( z \) values and colors, \texttt{plot3d} must then determine the colors associated with objects specified by two or more \( z \) values. If four sided panels are being
drawn, plot3d colors the panel with the color associated with the maximum \( z \) value of the four vertices; if lines are being drawn, plot3d colors the line with the color associated with the maximum \( z \) value of the endpoints.

Interactive ELLPACK provides utility routines which allow several pre-defined color maps to be used, as well as the ability for the user to define his own color map. Besides the default color map described above, Interactive ELLPACK also provides a default white-gray-black color map for black-and-white graphics terminals with grey scales (for black-and-white graphics terminals without grey scales, only wireframe graphs should be used). Another utility routine provided by Interactive ELLPACK gives the ability to keep a color map fixed across several plots. By default, the mapping of \( z \) values to colors changes with each graph; e.g., blue is always associated with \( z_{\text{min}} \) whose value can vary from plot to plot. However, in some instances the user would like to keep the color mapping constant. To do this, Interactive ELLPACK queries the user for dummy values for \( z_{\text{min}} \) and \( z_{\text{max}} \) which will supercede all actual \( z_{\text{min}} \) and \( z_{\text{max}} \) values. Thus, blue will always be associated with the dummy \( z_{\text{min}} \) value and red will always be associated with the dummy \( z_{\text{max}} \) value.

3.6 Hidden Line and Surface Removal

The plot3d module provides two types of 3D graphs: wireframe and solid panel. In the former, hidden line removal is performed, and in the latter hidden surface removal is performed. The method used in drawing a wireframe graph is a simple modification of the algorithm developed by Watkins in [4] and the reader is directed there for further details. In short, the algorithm works as follows: plot lines are drawn closest to farthest. While this is done, the program keeps a running track of what area in space is hidden from the observer because of previously drawn lines; if a new plot line falls in this area, it is not drawn; if a new plot line does not fall in this area, it is drawn and the hidden space information is updated.

When solid color panels are chosen for the graph, the panels are drawn farthest to closest. Thus, hidden surface removal is taken care of automatically since the closer panels overlap and obscure the farther panels.

In determining which points on the surface are farthest from the observer plot3d considers the \( w \) values of the transformed unit square in the \( z = z_{\text{int}} \) plane. Using (5), we let

\[
\begin{align*}
  w_{00} &= W_{\phi,\theta}(0,0,z_{\text{int}}) = 0, \\
  w_{10} &= W_{\phi,\theta}(1,0,z_{\text{int}}) = \sin \phi \sin \theta, \\
  w_{01} &= W_{\phi,\theta}(0,1,z_{\text{int}}) = \cos \phi \sin \theta, \\
  w_{11} &= W_{\phi,\theta}(1,1,z_{\text{int}}) = (\sin \phi + \cos \phi) \sin \theta,
\end{align*}
\]
and

\[ w_{\text{max}} = \max(w_{00}, w_{10}, w_{01}, w_{11}). \]

It can now determine the farthest point from the observer as follows:

\[
\begin{align*}
\text{if } w_{00} &= w_{\text{max}} \text{ then } (x_{\min}, y_{\min}) \text{ is farthest,} \\
\text{else if } w_{10} &= w_{\text{max}} \text{ then } (x_{\max}, y_{\min}) \text{ is farthest,} \\
\text{else if } w_{01} &= w_{\text{max}} \text{ then } (x_{\min}, y_{\max}) \text{ is farthest,} \\
\text{else } (x_{\max}, y_{\max}) \text{ is farthest.}
\end{align*}
\]

3.7 Text

Besides the graph lines and panels, plot3d will also print out text to the screen, specifically:

1. The name of the function plotted is printed in the upper left hand corner.

2. The date is printed in the lower left hand corner.

3. Labeled \( x \) and \( y \) axes may be printed.

4. An optional color index table may be printed. Due to the size of the color index table, the
   size of the uv-window must be enlarged to make room for it. A side effect of this enlarging is
   that the 3D graph appears smaller in the viewport.

All text is printed in the viewplane inside the uv-window. The exact location and size of text are
defined to be certain percentages of the uv-window size, and these percentages are values built into
the plot3d module. They can be changed by the user (see Section 6).

3.8 Nonrectangular Domains

In order to handle nonrectangular domains, plot3d makes use of the ELLPACK routine \texttt{q2osmn},
which determines whether any given point is inside, outside or on a domain boundary. When
setting up the \( z \) value array, plot3d calls \texttt{q2osmn} for each grid point and stores a special flag in
the array for any grid point which is outside of the domain. When any graphics output routine
recognizes one of these flags, it skips that point, and all subsequent ones until a valid \( z \) value is
found.
4 User Interface

The plot3d module has three parameters associated with it:

fname (required) the name of the function to be plotted.

nx, ny (optional) the maximum number of x and y grid lines in the graph. These numbers control the number of panels in the wireframe or solid graph. The number of grid lines can be changed during the run of an Interactive ELLPACK program. The default values for nx and ny is 33.

Once this module is invoked, the user is presented with the menu shown in Figure 8. The letters to the left of the colon are character strings which the user types in order to modify various parameters of the plot. Immediately to the right of the colon are descriptions of the plot parameters and to the far right are the current values of those parameters. A detailed description of each selection follows:

ph - controls the value of the rotation angle \( \phi \) about the screen perpendicular (the w-axis). When \( \phi = 0 \) the x axis is pointing to the right and the y axis is pointing upwards (i.e. the observer is located on the positive x axis and is looking towards the origin). If the \( \theta \) rotation is 0, then setting \( \phi = 90 \) results in the x axis pointing upwards and the y axis pointing out to the left. Setting \( \phi = 180 \) and \( \phi = 270 \) result in the x axis pointing to the left and down, respectively (with the corresponding movement of the y axis). Setting \( \phi = -90 \) is the same as \( \phi = 270 \). The initial value for \( \phi \) is 45.
th - controls the value of the rotation angle $\theta$ about the horizontal axis in the viewplane (the $u$-axis). When $\theta = 0$ the $z$ axis is pointing to the right and the $y$ axis is pointing upwards. If the rotation about the vertical ($\phi$ described above) is held at $0$, then setting $\theta = 90$ results in the $y$ axis pointing into the screen while the $x$ axis remains unchanged. Setting $\theta = 180$ and $\theta = 270$ result in the $y$ axis pointing downwards and out of the screen, respectively (with the $z$ axis remaining unchanged). Setting $\theta = -90$ is the same as $\theta = 270$. The initial value for $\theta$ is 45.

$x_0$, $y_0$ - allows the modification of the placement of the graph origin on the screen. The graph origin is defined as the point $(x_{\text{min}}, y_{\text{min}}, z_{\text{int}})$. By setting these values, the user overrides the automatic placement of the viewing window as described in Section 3.4. In order to be more convenient to the user, the interface presents the $u$ and $v$ screen lengths as running from 0.0 to 100.0; thus to place the graph origin at the center of the screen, the user would set $x_0 = 50.0$ and $y_0 = 50.0$. Note that the values for $x_0$ and $y_0$ can then be viewed as percentages of screen lengths. If not set by the user, the word UNSET is printed as the selection value and the program will automatically determine reasonable values for both $x_0$ and $y_0$ (see Section 3.4). If the user modifies either $x_0$ or $y_0$ while the other is UNSET, the program will prompt the user for a value for the remaining unset origin location. The initial value for both $x_0$ and $y_0$ is UNSET.

$z_s$ - allows the modification of the scaling factor for plot function values. If not set by the user, the word UNSET is printed as the selection value and the program will automatically determine a reasonable value for $z_s$ (see Section 3.4). The initial value for $z_s$ is UNSET.

$z_i$ - controls the location of the $z$-intercept for the plot axes. There are two ways to specify the $z$-intercept: 1) The user can input a specific numeric value or 2) he can specify that the minimum $z$ value be used as the intercept. This latter choice is indicated on the main menu by the word ZMIN. Note that if ZMIN is selected, the $z$-intercept may vary between different plots. The initial value for $z_i$ is 0.0.

$g_l$ - controls the number of $x$ and $y$ plot lines. The maximum number of plot lines is determined by the plot3d parameters $n_x$ and $n_y$. The number of $x$ and $y$ plot lines is originally set to these maximum values.
gt – controls the type of graphics plot. After typing gt the user is presented with the menu shown in Figure 9. The three types of graphs are:

1. PANEL WITH BOUNDARIES: solid colored panels with white boundary lines.
2. PANELS WITHOUT BOUNDARIES: solid colored panels with no boundary lines.
3. WIREFRAME: colored wireframe with hidden line removal.

The initial graph type is PANEL WITH BOUNDARIES.

pl – controls the printing of the plot axes. Typing pl toggles between true and false. If false, the axes will not be plotted. The initial setting is true.

c – controls the printing of the color index table. Typing ct toggles between true and false. If true, the color index table will be printed. The initial setting is false.

d – draws the 3D graph onto the currently active view.

r – returns to the Interactive ELLPACK control program.

Whenever the user changes φ, θ or the z-intercept, the values of the x origin, y origin and z scale are automatically reset to UNSET. This is due to the fact that changing any of the first three values can drastically effect the orientation of the graph and almost always requires a recalculation of the the latter three values. The user can, if he wishes, manually reset the graph origin location and z scale values, but it is usually best if he lets the graphics software determine reasonable values for these parameters first, and then afterwards modifies them for his needs.
Most graph parameters remain unchanged over repeated calls to plot3d. For example, if a user sets $\phi = 45$ while plotting and then returns to the Interactive ELLPACK control program, upon return to the plot3d module, $\phi$ still has the value 45. This property holds for all the plot parameters except for $x_0$, $y_0$ and $z_0$, which are always reset to UNSET.

5 3D Software

5.1 q83d00

Subroutine q83d00 initializes all of the interactive parameters described in Section 4 as well as many internal variables which control various aspects of the graphics output, such as the placement and formatting of labels. The Interactive ELLPACK control program calls q83d00 once at the beginning of the execution. For more detailed information on the variables initialized by q83d00 see Section 6.

5.2 q83dmn

Subroutine q83dmn sets the number of grid lines and initializes a table of $z$ values to be plotted. The table is a two dimensional array representing the various grid points and is located in the unnamed common block set up by the Interactive ELLPACK control program. While filling in this table, the subroutine calculates the minimum and maximum $z$ values for the particular grid specified and, once it has these two values, determines the spacing of colors over the $z$ value range. In addition, q83dmn performs some minor initialization tasks.

This subroutine is called from one of two places: 1) the Interactive ELLPACK control program calls it immediately after the plot3d module has been selected by the user. 2) subroutine q83din (see below) calls q83dmn whenever the "gl" option is selected from the main menu, in which case a new table of $z$ values must be computed.

When the function to be plotted is defined over a non-rectangular domain, q83dmn flags any grid points which are not in the domain by storing the largest positive real number representable by the machine into the corresponding location in the table. Later subroutines which actually create plot files will ignore these grid points.

5.3 q83din

Subroutine q83din provides the interface for the plot3d module. For more details on the menus and options presented, see Section 4.
5.4 q83dpl

Subroutine q83dpl is the driving routine for the creation of the 3D plots. It decides what the relative size of the plot will be on the current view, and then prepares and sends data to one of the two graph routines (q83dpn and q83dwf below). The tasks it performs are done in the following order:

1. Determines the viewing window size around the function to be plotted. This specifies how large the \( x \) and \( y \) axes will appear on the screen.

2. Determines the \( z \) scaling factor. This specifies the vertical range of the 3D plot on the screen.

3. Determines the exact location of the viewing window relative to the function to be plotted. This specifies the location of the graph origin on the screen.

4. Determines the order in which data is sent to the graph routines.

For more details on determining the viewing window size and location see Section 3.4.

5.5 q83dpn

Subroutine q83dpn outputs a strip of panels for a given set of \( x, y \) and \( z \) values. Given any two adjacent \( x \) grid points \((x_1, x_2)\) and adjacent \( y \) grid points \((y_1, y_2)\), a solid panel is drawn connecting the four points \((x_1, y_1, z(x_1, y_1)), (x_1, y_2, z(x_1, y_2)), (x_2, y_2, z(x_2, y_2)), (x_2, y_1, z(x_2, y_1))\). The subroutine requires the following data:

\( xdata: \) a vector of \( x \) grid points.

\( y_1, y_2: \) two adjacent \( y \) grid points.

\( zdata_1, zdata_2: \) corresponding \( z \) values along grid lines \( y = y_1 \) and \( y = y_2 \), respectively.

Panels are graphed parallel to the \( x \) axis. The color of each panel is determined by the maximum \( z \) value of the four grid points determining the panel. For non-rectangular domains and domains with holes, only those panels are printed whose four grid points lie inside the domain.

5.6 q83dwf

Subroutine q83dwf outputs a wireframe graph of the user specified function. The method used is a modification of the algorithm developed by Watkins in [4] and the reader is directed there for further details.
5.7 q83dax

Subroutine q83dax draws the projected z and y axis onto the screen. These axes are located in the plane defined by the value of the z-intercept of the graph (see Section 4). The subroutine prints a numeric label at 5 locations along each axis: at each endpoint, and at the one-quarter, one-half and three-quarters point. The subroutine also draws a vertical line from these labeled axes points to their corresponding z values; this gives the user a better idea how the graph is located over the x-y domain.

Subroutine q83dax uses many built in constants to determine the relative sizes and locations of the axes and their labels. These constants all represent percentages of the wlenw, the horizontal length of the graph window (see Section 3.4). These constants are further described in Section 6 along with instructions as to how they may be redefined by the user.

6 Modifying the Software

The plot3d module contains many parameters which determine how the finished graph will look on the screen viewport. Most of these parameters are under user control and fall into one of two categories:

Run-time parameters: These parameters can be changed by the user while the program is running. An example of this type is the parameter which controls the horizontal rotation angle of the graph.

Compile-time parameters: These parameters are set at compile time and thus are fixed throughout the execution of the plot3d module. To change these parameters the user must modify the code of one of the plot3d subroutines described in Section 5 and then recompile and replace it in the Interactive ELLPACK library. An example parameter of this type is the parameter which controls the location of the axes labels.

The run-time parameters are described in Section 4; this section will describe how to modify the compile-time parameters.

6.1 Default Initial Values for Run-time parameters

Section 4 described the run-time parameters which can be changed during the execution of the plot3d module. The default initial values for these parameters are set in the q83d00 subroutine; if the user wants new initial values for these run-time parameters he must modify the assignment statements in this subroutines and then recompile it and replace it in the Interactive ELLPACK
archive library pltlib.a. The parameters which fall into this category and their default initial values are the following:

**phi, theta**: the initial values for $\phi$ and $\theta$, the rotation angles about the vertical and horizontal, respectively. The default initial value for both $\phi$ and $\theta$ is 45.0.

**zint**: the initial value for the $z$-intercept. Its default initial value is 0.0.

**mxdfit, mydfit**: the initial values for the number of $x$ and $y$ grid lines in the graph. They both have a default initial value of 33.

**lpltax**: the initial value for the logical flag controlling the printing of the $xy$-domain axes. Its default initial value is .true.

**lclrtb**: the initial value for the logical flag controlling the printing of the color index table. Its default initial value is .false.

**igtflg**: the initial graph type selection. $igtflg$ can have one of three values which indicate the following:

- 1 = panels with boundaries printed
- 2 = panels without boundaries printed
- 3 = wireframe

Its default initial value is 1.

These values are only used the first time that plot3d is called; thereafter, $\phi$, $\theta$, the $z$-intercept, the number of gridlines, the plot flags and the grid type parameter retain the values they had at the end of the previous call of the module. The one exception to this rule occurs when the user specifies the number of $x$ and $y$ gridlines in the plot3d calling statement; in this case the number of gridlines will always be reset to these values in subsequent calls to plot3d (see Section 4).

### 6.2 Plot Layout Parameters

Subroutine q83d00 also sets the default values for various parameters which control the positioning of the graph on the screen and the labeling of the graph and its axes.

**xlab, ylab**: the default strings used to label the $x$ and $y$ axes. Their default initial values are the two strings 'x' and 'y'.

**xlabsz, ylabsz**: the sizes of the strings $xlab$ and $ylab$. They both have default initial values of 1.

**xnfmt, ynfmt**: the default FORTRAN format specification for the numeric labels of the graph. They both have default initial string value 'f5.2'.

23
xnumsz, ynumsz: the field width of the znfmt and ynfmt format specifications. They both have
default initial values of 5.

The values of the following parameters represent percentages of the horizontal viewing window
length.

rhou, rhov: the default values for \( \rho_u \) and \( \rho_v \) described in Section 3.4. The default initial value for
\( rhou \) is 0.1. \( Rhou \) is then set to be:

\[
Rhou = \frac{1}{2} \left( 1 - \frac{1 - 2rhov}{\alpha_{opt}} \right),
\]

where \( \alpha_{opt} \) is the viewport aspect ratio (see Section 3.1).

rnameu, rnamev: the default u and v location of the name of the function being graphed. Their
default initial values are 0.03 and 0.95.

Many of these parameters are related to each other, so care must be taken when changing them.
Specifically:

- if new strings are used for xlab and/or ylab, then the values of xlabels and ylabels may have to
  be changed to reflect the new string lengths.

- if a new format is used for znfmt and/or ynfmt, then the values of xnumsz and ynumsz may
  have to be changed to reflect the length of the output resulting from the new format.

References

J. Ribbens).

Wesley, 1983.

1985.