CGTM Number 204
August 1981

Revised:
October 1983
November 1985

THE UNIFIED GRAPHICS SYSTEM FOR FORTRAN 77

GRAPHIC ALGORITHMS MANUAL

ROBERT C. BEACH
COMPUTATION RESEARCH GROUP
STANFORD LINEAR ACCELERATOR CENTER
STANFORD, CALIFORNIA 94305

TABLE OF CONTENTS

SECTION 1: AN INTRODUCTION TO THE GRAPHIC ALGORITHMS

This document describes a group of subroutines that incorporate a number of graphic algorithms. These subroutines may be called from FORTRAN 77 and simplify the production of certain classes of pictures. Among the types of pictures that these subroutines will help in producing are:

1. Point or parallel projections of three-dimensional objects.
2. The drawing of smooth curves from a few points on the curve.
3. Graphs of functions with either linear or logarithmic scaling on the axes.
4. Contour and surface plots of a function of two independent variables.

The subroutines that are described here are most conveniently used in conjunction with the basic Unified Graphics System as described in [Bea81] and the following descriptions will assume that the reader is familiar with that document. Although these subroutines are described in conjunction with the basic Unified Graphics System subroutines, they are not very strongly linked to that system. The only link between these subroutines and the basic Unified Graphics System subroutines is their use of the common options scanning and error processing modules. Other than this, the subroutines described here could be used with any plotting package.

These subroutines are available in the same libraries that contain the basic Unified Graphics System and the user should refer to [Bea81] for information on linking to the proper library. The error messages are produced in the same manner as those for the basic Unified Graphics System subroutines.

SECTION 2: A DETAILED DESCRIPTION OF THE SUBROUTINES

This section gives a complete description of each of the subroutines associated with the graphic algorithms in the Unified Graphics System.

In the following descriptions of the subroutines, floating point or character string arguments are always described as such; if nothing is said about the data type of a parameter, it is fixed point. All arguments described as character strings must be character string literals or of type CHARACTER. Almost all arguments represent input to these subroutines; when an argument is an output variable, it will be explicitly described as such and will be underlined in the list of parameters in the calling sequence.

SECTION 2.1: CONVERTING NUMBERS TO CHARACTER STRINGS

This section describes a subroutine that may be used to convert a number to a character string. This operation is a necessary prelude to plotting a number as, for example, a label on an axis.

SECTION 2.1.1: SUBROUTINE UGCNVF

This subroutine may be used to convert a floating point number to a character string. The conversion is similar to the conversion done by a Fw.d format item on a FORTRAN write. The character string may then, for example, be passed to subroutine UGTEXT or UGXTXT.

The principal advantage of this subroutine over the in-memory conversion available in FORTRAN is that this subroutine returns the actual number of characters in the converted number; this value can be useful in centering labels under tic marks.

The calling sequence is:

```
CALL UGCNVF(NUMBER,FDEC,STRING,NBLANK)
```

The parameters in the calling sequence are:

- NUMBER — The floating point value which is to be converted to a character string.
- FDEC — A fixed point value giving the number of places to the right of the decimal point. If FDEC has the value zero, then no decimal point will appear in the string. The value of FDEC must be between 0 and 10.
- STRING — The resulting character string. The string must be at most 12 characters long.
- NBLANK — The number of non-blank characters in the converted number. The non-blank characters are right adjusted

in STRING.

No error messages are produced by this subroutine. If, for any reason, the number cannot be converted, the character string will be filled with asterisks.

Numerous examples of the use of this subroutine will be found in the sample programs in the sections on axis plotting and contour plotting.

SECTION 2.2: PROJECTING THREE-SPACE INTO TWO-SPACE

This section describes a method of transforming three-dimensional data into two-dimensional data so that it may be plotted. Subroutine UGTRAN will generate a transformation representing a point projection or parallel projection. Subroutine UGPROJ uses this transformation to project a three-dimensional point into two dimensions.

SECTION 2.2.1: SUBROUTINE UGTRAN

This subroutine may be used to define a point projection or parallel projection from three-dimensional space into two-dimensional space. The transformation may be used with subroutine UGPROJ to project a point in three-dimensional space into two-dimensional space. The meaning of the input to this subroutine is illustrated in Figure 2.2.1.

First, consider a point projection. To begin, a reference point, REFP, and a view direction, VDIR, are given. REFP and VDIR are given in the three-dimensional coordinate system of the object being viewed. The reference point may be thought of as a point on the forehead of a person and the view direction as the direction in which the person is looking. Along the view direction, at a distance of SCRD, is a projection screen. This screen contains a square with a side length of SCRZ and its horizontal axis is defined by the vector HDIR. The vector UDIR gives an upward direction and serves to further define the orientation of the projection screen. Finally an eye point, E, is defined by moving a distance EYED from REFP in the direction of HDIR. REFP is a point in the three-dimensional coordinate system; VDIR, HDIR, and UDIR are direction numbers of vectors in that system; and SCRD, SCRZ, and EYED are also measured in that system. The transformation takes a point P in three-dimensional space and projects it by means of a straight line from E through P onto the screen to define the point Q. The point Q is the projective transform of P and its coordinates are (t,u). The purpose of the EYED parameter is the generation of stereo pairs; a right eye transformation has a positive value for EYED while a

left eye transformation has a minus value. If only a single view is being produced, that is, if EYED is zero, then only the ratio of SCRD to SCRZ is critical; multiplying both by the same constant will not change the transformation. No scissoring of data is done when the points are projected; projected points may lie outside the square area on the projection screen.

For a parallel projection, the point P is projected onto the screen parallel to the vector VDIR. The user will notice that the position of REFP is not critical for a parallel projection; it may be moved anywhere along VDIR without changing the transformation. Also, the value of SCRD is completely redundant. Nevertheless, the programmer should supply reasonable values for these parameters because this information is saved in TRANS and is utilized by some other subroutines.

A good description of geometric projections may be found in [Car78]. The subroutine described here may be used to produce any of the transformations described in that document.

The calling sequence is:

```
CALL UGTRAN(OPTIONS,REFP,VDIR,HDIR,UDIR,SCRD,SCRZ,TRANS)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- PARALLEL Indicates that a parallel projection is to be generated instead of the default point projection.
- EYED=<value> One-half the eye separation for making stereo pairs. The default value is 0.0.
- XLO=<value> The X coordinate at the left hand side of the screen. The default value is 0.0.
- XHI=<value> The X coordinate at the right hand side of the screen. The default value is 1.0.
- YLO=<value> The Y coordinate at the bottom of the screen. The default value is 0.0.
- YHI=<value> The Y coordinate at the top of the screen. The default value is 1.0.
- TOLER=<value> A tolerance that is used to detect singular situations. The default value is 0.0001.

REFP A floating point array of dimension 3 giving the projective reference point.

VDIR A floating point array of dimension 3 giving the view direction.

HDIR A floating point array of dimension 3 giving the horizontal direction of the projection screen. If this vector is given as (0.0, 0.0, 0.0), then a vector parallel to the X-Y plane and perpendicular to VDIR will be supplied. Notice that this vector cannot be given as all zeros if VDIR points in the Z direction.

UDIR — A floating point array of dimension 3 giving the upward direction of the projection screen. If this vector is given as (0.0, 0.0, 0.0), then a vector perpendicular to VDIR and HDIR is supplied. If the given vector is not perpendicular to HDIR, it is rotated, within the plane of HDIR and UDIR, until it is perpendicular to HDIR.

SCRD — A floating point value giving the distance from REFP to the screen.

SCRZ — A floating point value giving the size of the screen.

TRANS — A floating point array of dimension 31 which will be set to the transformation. This array will contain a 3 by 4 projection matrix as well as normalized copies of the given parameters.

The errors detected by this subroutine are:

1(3): The requested transformation is singular and cannot be generated.

Examples of the use of point and parallel projections will be found in the sample programs in the sections on the mesh surface and the two-dimensional histograms.

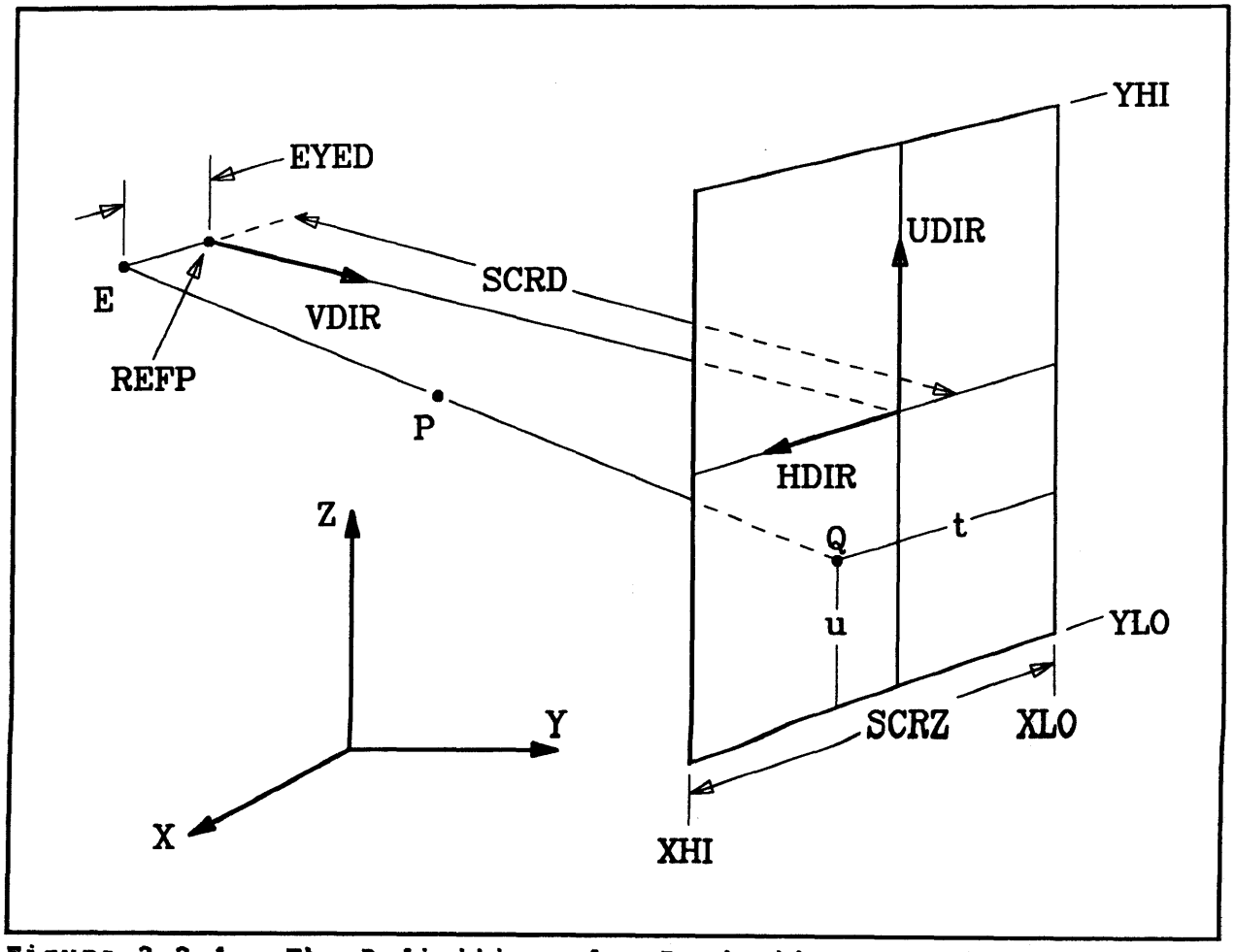

Figure 2.2.1: The Definition of a Projective Transformation.

SECTION 2.2.2: SUBROUTINE UGPROJ

This subroutine uses a transformation defined by subroutine UGTRAN to project a three-dimensional point into two dimensions.

The calling sequence is:

```
CALL UGPROJ(TRANS,PT3D,PT2D)
```

The parameters in the calling sequence are:

TRANS	A floating point array of dimension 31 containing the transformation.
PT3D	A floating point array of dimension 3 giving the coordinates of the three-dimensional point.
PT2D	A floating point array of dimension 2 which will be set to the projected point.

No error messages are produced by this subroutine.

SECTION 2.3: SMOOTH CURVE INTERPOLATION

This section describes a subroutine which can be used to interpolate a smooth curve through a given sequence of points.

SECTION 2.3.1: SUBROUTINE UGSCIN

In some applications, a user has a few points and needs a smooth curve drawn through them. This subroutine will satisfy that need. The interpolation curve is represented parametrically and therefore may be multiple-valued in either X or Y, or in both. The curve is local; that is, the curve between two points depends only on those two points and the points on either side of them. The curve is adjustable in that a variable analogous to tension may be assigned at each point. Finally, the curve is independent of the coordinate system in which it is represented.

The subroutine works by first assigning a parameter value to each given point. This parameter value may be assigned uniformly or non-uniformly. In either case, the parameter is zero at the first point. In the uniform case, it increases by one as the curve passes through each successive point. In the non-uniform case, the parameter increases by an amount equal to the distance between the points. The uniform assignment of parameter values is computationally simpler, but only works well when the points to be interpolated are nearly equally spaced. Next, the curve is defined by determining X and Y as cubic polynomial functions of this parameter. Finally, the equations for X and Y are evaluated and the resulting points are passed to a user supplied subroutine. The equations for X and Y may be evaluated to give an equal number of points between each given point, or the user

may choose to have the equations evaluated at equally spaced values of the parameter. These two options for evaluating the curve give essentially identical results in the case of a uniform curve.

The user may apply additional constraints to the interpolation curve at its beginning and terminal points. At these points, the user must specify one of three possible constraints. The first possibility is to force the second derivative to be zero at an end point. A zero for a second derivative will be apparent by forcing the curve to have zero curvature at it ends. The second possibility is to specify an additional point out beyond the ends of the interpolation curve to control the shape of the curve at its ends. The third possibility is to specify the direction of the tangent vector at the ends of the curve. The conditions at the beginning or terminal ends of the interpolation curve may be applied independently.

The calling sequence is:

```
CALL UGSCIN(OPTIONS,LINSUB,XARRAY,YARRAY,NPTS,TARRAY,NTENS,
            BFLAG,BXVAL,BYVAL,TFLAG,TXVAL,TYVAL)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- UNIFORM Indicates that the parameter values are to be assigned uniformly. The default is to use non-uniform parameter values.
- GENTAN Indicates that the user supplied subroutine is to be called with the derivative vector as well as a point on the curve. The default is to call the user supplied subroutine with the point on the curve only.
- XFACT=<value> The X factor to be used when computing the parameter values on non-uniform curves. The default value is 1.0.
- YFACT=<value> The Y factor to be used when computing the parameter values on non-uniform curves. The default value is 1.0.

The use of XFACT and YFACT will be described below.

- NPARM=<value> The number of interpolated line segments to be generated for each given line segment.
- DPARM=<value> The parameter spacing between computed points.

If neither NPARM or DPARM is given, then the default is NPARM=8.

- TOLER=<value> An internal tolerance which is used to test for point coincidence. The default value is 0.001.

LINSUB The entry point of the line end point subroutine.

XARRAY A floating point array containing the X coordinates of the points to be interpolated.

YARRAY A floating point array containing the Y coordinates of

the points to be interpolated.

NPTS	The number of points in XARRAY and YARRAY.
TARRAY	A floating point array containing the tension values. Additional help is assigning tension values will be given below.
NTENS	The number of values in TARRAY. If NTENS is less than NPTS, then the values will be selected from TARRAY cyclically.
BFLAG	A flag which specifies the type of control to be applied at the beginning end of the interpolated curve. A value of 0 means zero curvature, a value of 1 means BXVAL and BYVAL contain a control point, and a value of 2 means BXVAL and BYVAL contain a tangent vector.
BXVAL	The floating point X value for BFLAG equal 1 or 2.
BYVAL	The floating point Y value for BFLAG equal 1 or 2.
TFLAG	A flag which specifies the type of control to be applied at the terminal end of the interpolated curve.
TXVAL	The floating point X value for TFLAG equal 1 or 2.
TYVAL	The floating point Y value for TFLAG equal 1 or 2.

The errors detected by this subroutine are:

1(3):	The number of points to be interpolated must be at least two.
2(3):	The number of tension values must be at least one.
3(3):	The initial or terminal control of the interpolating curve is invalid.
4(3):	The distance between two given points is too small.

The skeleton for the line end point subroutine when the GENTAN option is not given is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT)
      REAL          XCRD,YCRD
      INTEGER       BBIT
        ...
      END
```

The value of BBIT will be 0 for the first point of an interpolation curve and 1 for each point thereafter. When GENTAN is given the subroutine skeleton is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT,DELX,DELY)
      REAL          XCRD,YCRD
      INTEGER       BBIT
      REAL          DELX,DELY
        ...
      END
```

The purpose of the XFACT and YFACT options items is to allow the user to compensate for dissimilarities in the scaling in the current window. Suppose, for example, the view port is square but the scaling in the window is such that X runs from 0.0 to 100.0 while Y runs from 0.0 to 1.0. If the standard formula for distance is used in this case, the differences in X coordinates would overwhelm the differences in Y coordinates. As a result, the distance computed would really only represent a difference in X coordinate and, on a non-uniform curve, the results could be

very strange. In the case described here, this problem can be overcome by either setting XFACT=0.01 or YFACT=100.0. When XFACT or YFACT must be used, it is best to use NPARM instead of DPARM.

The easiest end point condition to handle is the specification of an extra point (BFLAG and TFLAG equal one). When tangent vectors are given with the default values of XFACT and YFACT, the vector should be approximately a unit vector. When XFACT and/or YFACT are not one, the tangent vector should be scaled appropriately.

The specification of tension values can be a difficult problem, fortunately, only the simplest situations usually arise. Tension values may take on any value but only a narrow range is very useful. The "natural" value is unity. As tension is reduced toward zero, the curve becomes taught and when the tension becomes zero the curve becomes a straight line joining the points to be interpolated. As the tension value is increased from unity, the curve relaxes and, for large values of tension, can form loops. Useful values of tension range from 0.5 to 1.5 with values around 0.9 to 1.1 generally giving the best results. While this subroutine allows tension to be applied independently at each point, the usual scheme is to apply a single value to the entire interpolation curve. Defining a single value of tension for the entire curve is done by setting TARRAY(1) to that value and setting NTENS to one. Setting the tension independently for each point can usually only be done in interactive settings, however, in that case, the user has a large amount of control over the interpolating curve.

Smooth closed curves may be interpolated with this subroutine. To do this, the point (XARRAY(1),YARRAY(1)) and (XARRAY(NPTS),YARRAY(NPTS)) should be identical. The easiest way to then assure that the end points join smoothly is to use BFLAG and TFLAG equal to one and set (BXVAL,BYVAL) to (XARRAY(NPTS-1),YARRAY(NPTS-1)) and (TXVAL,TYVAL) to (XARRAY(2),YARRAY(2)). The local nature of the interpolation scheme assures that the entire curve is smooth.

An example of a smooth curve interpolated by this subroutine is shown in Figure 2.4.1. The boundary of the cross-hatched region is defined by the marked points. The smooth closure of the curve was assured by the scheme described above. A value of 1.05 was used for the tension.

SECTION 2.4: CROSS-HATCHING A POLYGONAL REGION

This section describes a subroutine which can generate cross-hatching for any arbitrary polygonal region.

SECTION 2.4.1: SUBROUTINE UGXHCH

This subroutine may be used to generate data describing a cross-hatched region. This subroutine calls a user supplied subroutine to process the line end point data. The region to be cross-hatched is specified by giving a sequence of points which describe a closed curve. If the curve is a simple closed curve, that is, one that does not intersect itself, then it is exactly the interior that is cross-hatched. For curves that intersect themselves, the areas that are cross-hatched are those areas that are reached from the exterior by crossing the curve an odd number of times. Figure 2.4.1 shows an example of cross-hatching with a boundary curve that intersects itself. One intended purpose of this subroutine is to provide a simple means of cross-hatching histograms, but there are many other possible uses.

The calling sequence is:

```
CALL UGXHCH(OPTIONS,LINSUB,XARRAY,YARRAY,NPTS,WKAREA,LDIM)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- SPACING=<value> The spacing between the lines of cross-hatching. The default value is 0.02.
- ANGLE=<value> The angle at which the lines of cross-hatching are drawn. The default value is 45.0.
- X=<value> The X coordinate of a point which will have a line of cross-hatching pass through it. The default value is XARRAY(1).
- Y=<value> The Y coordinate of a point which will have a line of cross-hatching pass through it. The default value is YARRAY(1).
- XFACT=<value> The X factor to be used when processing the ANGLE and SPACING values. The default value is 1.0.
- YFACT=<value> The Y factor to be used when processing the ANGLE and SPACING values. The default value is 1.0.

The use of XFACT and YFACT will be described below.

- TOLER=<value> An internal tolerance which may have to be adjusted in unusual situations. The default value is 0.0001.

LINSUB The entry point of the line end point subroutine.

XARRAY A floating point array containing the X coordinates of the end points of line segments which bound the region to be cross-hatched.

YARRAY A floating point array containing the Y coordinates of the end points of the line segments.

NPTS The number of end points in XARRAY and YARRAY.

WKAREA A floating point array which will be used as a work area. A dimension at least as large as the maximum number of intersections between a line of cross-hatching and the region boundary is required.

LDIM The dimension of WKAREA.

The errors detected by this subroutine are:

1(3): Either the first and last points in the region definition are not the same, or there are fewer than three points given.

2(3): The work area array is not large enough.

The skeleton for the line end point subroutine is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT)
      REAL          XCRD,YCRD
      INTEGER       BBIT
        ...
      END
```

The value of BBIT may be 0 or 1 and is the blanking bit.

The purpose of the XFACT and YFACT options items is to allow the user to compensate for dissimilarities in the scaling in the current window. Suppose, for example, the view port is square but the scaling in the window is such that X runs from 0.0 to 100.0 while Y runs from 0.0 to 1.0. If the standard formula for angle is used in this case, the result will not be as expected. In the case described here, this problem can be overcome by either setting XFACT=0.01 or YFACT=100.0. If XFACT is used, the SPACING value is measured along the Y axis; if YFACT is used, the SPACING value is measured along the X axis.

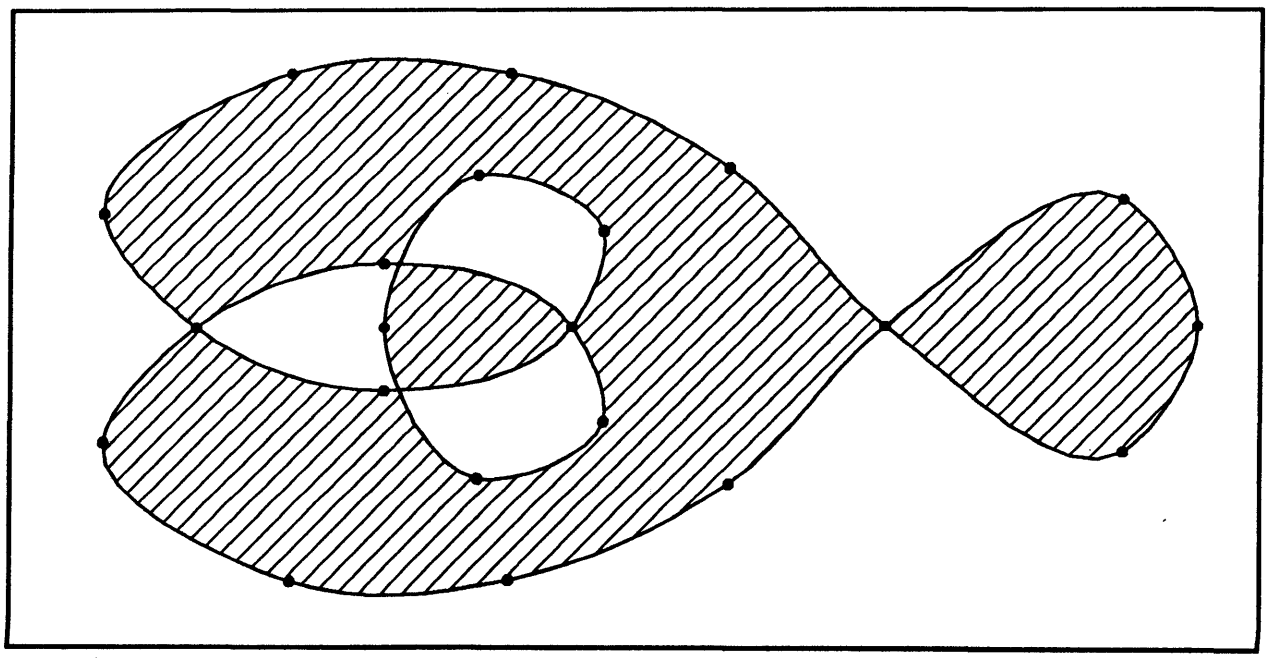

Figure 2.4.1: Cross-Hatching with Self-Intersecting Boundaries.

The boundary of the cross-hatched region in Figure 2.4.1 was interpolated from the marked points using subroutine UGSCIN. Another example of the use of the cross-hatching subroutine will be found in the sample program in the section on plotting a mesh surface.

SECTION 2.5: AXIS GENERATION

This section describes a number of subroutines which may be used to plot axes with linear or logarithmic labeling. The axis may be drawn in any orientation; horizontally, vertically, or at any arbitrary angle. Subroutine UGLNAX generates an axis with linear labels and subroutine UGLNDX is an aid in obtaining "round numbers" on the axes. Subroutine UGLGAX generates an axis with logarithmic labels and subroutine UGLGDX can help in labeling the axes. Finally, a sample program is shown which reads card images and plots a complete graph.

SECTION 2.5.1: SUBROUTINE UGLNAX

This subroutine may be used to generate the description of an axis with linear labels and tic marks. This subroutine calls user supplied subroutines to process the graphic data; one subroutine is called to process line data and a second subroutine is called to process the label information. The reader should refer to subroutine UGLNDX for help in assigning values to the parameters LOLAB, HILAB, and NLAB which result in "round numbers" being used for the labels.

The calling sequence is:

```
CALL UGLNAX(OPTIONS,LINSUB,TXTSUB,TXTFLG,XLO,YLO,XHI,YHI,
            LOLAB,HILAB,NLAB)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- LSTM=<value> The length of the labeled tic marks on the left side of the axis. The default value is approximately 0.01 times the length of the axis.
- RSTM=<value> The length of the labeled tic marks on the right side of the axis. The default value is approximately 0.01 times the length of the axis.
- NSTM=<value> The number of secondary tic marks between the labeled tic marks. The default value is 0. These tic marks are three-fourths the length of the labeled tic marks.

LINSUB The entry point of the line end point subroutine.

TXTSUB The entry point of the label subroutine.

TXTFLG A flag that will be passed to TXTFLG.

XLO The floating point X coordinate of the low end of the axis.

YLO The floating point Y coordinate of the low end of the axis.

XHI The floating point X coordinate of the high end of the axis.

YHI The floating point Y coordinate of the high end of the axis.

LOLAB	A floating point value giving the label to be placed at the low end of the axis.
HILAB	A floating point value giving the label to be placed at the high end of the axis.
NLAB	The number of labels and primary tic marks to be put on the axis.

The errors detected by this subroutine are:

1(3): The number of labels and primary tic marks is too small. It must be at least two.

The skeleton for the line end point subroutine is:

```
      SUBROUTINE     LINSUB(XCRD,YCRD,BBIT)
      REAL           XCRD,YCRD
      INTEGER        BBIT
        ...
      END
```

The value of BBIT may be 0, 1, 2, or 3. The values of 0 or 1 are the blanking bits for the axis or its tic marks. The values 2 or 3 are the blanking bits, incremented by 2, of the secondary tic marks.

The skeleton for the label subroutine is:

```
      SUBROUTINE     TXTSUB(XCRD,YCRD,VALUE,TXTFLG)
      REAL           XCRD,YCRD,VALUE
      INTEGER        TXTFLG
        ...
      END
```

The X and Y coordinates are those of the axis-tic mark intersection. It is the duty of TXTSUB to convert VALUE to a character string and plot that string at the proper offset from (XCRD,YCRD). The user may use TXTFLG to indicate to TXTFLG what it is to do. For example, the value of TXTFLG can indicate if a horizontal or vertical axis is being drawn and TXTSUB can offset X or Y the appropriate amount.

SECTION 2.5.2: SUBROUTINE UGLGAX

This subroutine may be used to generate the description of an axis with logarithmic labels and tic marks. This subroutine calls user supplied subroutines to process the graphic data; one subroutine is called to process line data and a second subroutine is called to process the label information. The reader should refer to subroutine UGLGDX for help in assigning values to the parameters LOLAB, HILAB, and NLAB which result in "round numbers" being used for the labels. This subroutine will produce the best results when the extent of the axis represents an integral number of full cycles.

The calling sequence is:

```
  CALL UGLGAX(OPTIONS,LINSUB,TXTSUB,TXTFLG,XLO,YLO,XHI,YHI,
              LOLAB,HILAB,NLAB)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

LSTM=<value> The length of the labeled tic marks on the left side of the axis. The default value is approximately 0.01 times the length of the axis.

RSTM=<value> The length of the labeled tic marks on the right side of the axis. The default value is approximately 0.01 times the length of the axis.

NSTM=<value> The number of secondary tic marks between the labeled tic marks. The default value is 0. These tic marks are three-fourths the length of the labeled tic marks.

LINSUB The entry point of the line end point subroutine.

TXTSUB The entry point of the label subroutine.

TXTFLG A flag that will be passed to TXTFLG.

XLO The floating point X coordinate of the low end of the axis.

YLO The floating point Y coordinate of the low end of the axis.

XHI The floating point X coordinate of the high end of the axis.

YHI The floating point Y coordinate of the high end of the axis.

LOLAB A floating point value giving the label to be placed at the low end of the axis.

HILAB A floating point value giving the label to be placed at the high end of the axis.

NLAB The number of labels and primary tic marks to be put on the axis.

The errors detected by this subroutine are:

1(3): The number of labels and primary tic marks is too small. It must be at least two.

The skeleton for the line end point subroutine and the label subroutine are the same as those for subroutine UGLNAX.

SECTION 2.5.3: SUBROUTINE UGLNDX

This subroutine is an aid in using subroutine UGLNAX. Consider the following problem: suppose the extent of the data in one direction can only be determined at execution time and suppose that the program has determined that the data extends from 2.637 to 7.913. Usually the programmer does not want these values to label the axis but would prefer "round numbers" to label the axis. In this case, for instance, it is preferable to have the axis run from 2.00 to 8.00 with 7 labeled tic marks, or perhaps from 2.50 to 8.00 with 12 labeled tic marks. This subroutine accepts as its input the extent of the data and limits on the number of labeled tic marks and produces values for the

parameters LOLAB, HILAB, and NLAB in subroutine UGLNAX which assures that all labeled tic marks are labeled with "round numbers".

Algorithms of this nature have been described in [Gia64, Dix65, and Lew73]. The algorithm used in UGLNDX is primarily based on the information in [Dix65].

The calling sequence is:

```
CALL UGLNDX(LODATA,HIDATA,MINLAB,MAXLAB,LOLAB,HILAB,NLAB)
```

The parameters in the calling sequence are:

LODATA	A floating point value giving the low extent of the data.
HIDATA	A floating point value giving the high extent of the data.
MINLAB	The minimum acceptable number of labeled tic marks.
MAXLAB	The maximum acceptable number of labeled tic marks.
LOLAB	A computed floating point value which will be LODATA reduced to a "round number".
HILAB	A computed floating point value which will be HIDATA increased to a "round number".
NLAB	A computed value which will make all of the labels "round numbers".

No error messages are produced by this subroutine.

SECTION 2.5.4: SUBROUTINE UGLGDX

This subroutine is an aid in using subroutine UGLGAX. This subroutine accepts as its input the extent of the data and limits on the number of labeled tic marks and produces values for the parameters LOLAB, HILAB, and NLAB in subroutine UGLGAX which assures that all labeled tic marks are labeled with "round numbers". The difference between LOLAB and HILAB will always represent an integral number of full cycles. The user should give a wider range to the limits on the number of tic marks than is necessary in subroutine UGLNDX. In particular, it is recommended that MINLAB have a value of 2 or 3.

The calling sequence is:

```
CALL UGLGDX(LODATA,HIDATA,MINLAB,MAXLAB,LOLAB,HILAB,NLAB)
```

The parameters in the calling sequence are:

LODATA	A floating point value giving the low extent of the data.
HIDATA	A floating point value giving the high extent of the data.
MINLAB	The minimum acceptable number of labeled tic marks.
MAXLAB	The maximum acceptable number of labeled tic marks.
LOLAB	A computed floating point value which will be LODATA reduced to a "round number".
HILAB	A computed floating point value which will be HIDATA

increased to a "round number".

NLAB — A computed value which will make all of the labels "round numbers".

The errors detected by this subroutine are:

1(3): Round numbers could not be found for the axis within the imposed constraints.

SECTION 2.5.5: AN EXAMPLE

This example shows how the preceding subroutines may be used to produce graphs of functions in conjunction with the basic Unified Graphics System subroutines. The program shown below reads data, determines the extent of the data, and produces a graph. The scaling on the horizontal axis is linear while that of the vertical axis is logarithmic. In this example, the same label subroutine is used to plot both sets of axis labels; the TXTFLG argument is used to distinguish the two cases. The program is:

```
      PROGRAM        AGAXIS
C
C  SAMPLE PROGRAM:   A SIMPLE GRAPH PLOTTER
C
      EXTERNAL       LSUB,TSUB
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(1000)
C
      REAL           VPRT(2,2),WDOW(2,2)
      INTEGER        TTLN,XAXN,YAXN
      CHARACTER*50   TTLP,XAXP,YAXP
      CHARACTER*50   TTLS,XAXS,YAXS
      REAL           XARY(100),YARY(100)
      REAL           XDLO,XDHI,YDLO,YDHI
      REAL           XALO,XAHI,YALO,YAHI
      INTEGER        NPTS,NLAB,INT1
C
      DATA           VPRT/ 2.6, 1.5,11.7, 8.5/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  SELECT THE DUPLEX CHARACTER GENERATOR.
      CALL UGOPEN('VEP12FF,GENIL',99)
      CALL UGFONT('DUPLEX')
C
C  GET THE DATA:  FIRST THE TITLE, AXIS LABELS, AND THE X
C  AND Y COORDINATES ARE READ AND THEN THE EXTENT OF THE
C  DATA IS DETERMINED.
      READ(5,101) TTLN,TTLP,TTLS,
     X            XAXN,XAXP,XAXS,
     X            YAXN,YAXP,YAXS
  101 FORMAT(I10,A50/10X,A50)
      READ(5,102) NPTS
  102 FORMAT(I10)
```

```
      READ(5,103) (XARY(INT1),YARY(INT1),INT1=1,NPTS)
  103 FORMAT(6F10.5)
      XDLO=XARY(1)
      YDLO=YARY(1)
      XDHI=XARY(1)
      YDHI=YARY(1)
      DO 104 INT1=2,NPTS
        XDLO=MIN(XDLO,XARY(INT1))
        YDLO=MIN(YDLO,YARY(INT1))
        XDHI=MAX(XDHI,XARY(INT1))
        YDHI=MAX(YDHI,YARY(INT1))
  104 CONTINUE
C
C  PLOT TITLES AND AXES:  FIRST A FRESH PLOTTING SPACE IS
C  REQUESTED, THEN THE INITIAL DRAWING SPACE IS CREATED,
C  THE SEGMENT IS CLEARED, AND THE TITLES ARE ADDED TO THE
C  SEGMENT FOLLOWED BY THE AXES.  FINALLY THE SEGMENT IS
C  TRANSMITTED TO THE DEVICE.
      CALL UGPICT('CLEAR',0)
      CALL UGDSPC('PUT',13.0,10.0,1.0)
      CALL UGINIT('CLEAR',SEGM,1000)
      CALL UGXTXT('CENTER,SIZE=0.4',6.5,9.25,
     X  TTLP(1:TTLN),TTLS(1:TTLN),SEGM)
      CALL UGXTXT('CENTER,SIZE=0.3',
     X  0.5*(VPRT(1,1)+VPRT(1,2)),0.8,
     X  XAXP(1:XAXN),XAXS(1:XAXN),SEGM)
      CALL UGXTXT('CENTER,SIZE=0.3,ANGLE=90',
     X  1.0,0.5*(VPRT(2,1)+VPRT(2,2)),
     X  YAXP(1:YAXN),YAXS(1:YAXN),SEGM)
      CALL UGLNDX(XDLO,XDHI,7,10,XALO,XAHI,NLAB)
      CALL UGLNAX('RSTM=0',LSUB,TSUB,1,
     X  VPRT(1,1),VPRT(2,1),VPRT(1,2),VPRT(2,1),
     X  XALO,XAHI,NLAB)
      CALL UGLNAX('LSTM=0',LSUB,TSUB,0,
     X  VPRT(1,1),VPRT(2,2),VPRT(1,2),VPRT(2,2),
     X  XALO,XAHI,NLAB)
      CALL UGLGDX(YDLO,YDHI,3,10,YALO,YAHI,NLAB)
      CALL UGLGAX('LSTM=0,NSTM=4',LSUB,TSUB,2,
     X  VPRT(1,1),VPRT(2,1),VPRT(1,1),VPRT(2,2),
     X  YALO,YAHI,NLAB)
      CALL UGLGAX('RSTM=0,NSTM=4',LSUB,TSUB,0,
     X  VPRT(1,2),VPRT(2,1),VPRT(1,2),VPRT(2,2),
     X  YALO,YAHI,NLAB)
      CALL UGWRIT(' ',0,SEGM)
C
C  PLOT THE DATA:  FIRST A WINDOW IS DEFINED SO THAT THE
C  DATA WILL MATCH THE AXES, THEN THE SEGMENT IS CLEARED,
C  THE DATA IS ADDED TO THE SEGMENT, AND THE SEGMENT IS
C  TRANSMITTED.
      WDOW(1,1)=XALO
      WDOW(1,2)=XAHI
      WDOW(2,1)=LOG10(YALO)
      WDOW(2,2)=LOG10(YAHI)
      CALL UGWDOW('PUT',VPRT,WDOW)
      CALL UGINIT('CLEAR',SEGM,1000)
```

```
      DO 201 INT1=1,NPTS
        CALL UGLINE(' ',XARY(INT1),LOG10(YARY(INT1)),1,
     X     SEGM)
  201 CONTINUE
      CALL UGWRIT(' ',0,SEGM)
C
C  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C  AND THE PROGRAM STOPS.
      CALL UGCLOS(' ')
      STOP
C
      END
C*****************************************************************
      SUBROUTINE    LSUB(XCRD,YCRD,FLAG)
C
C  LINE SEGMENT SUBROUTINE:  THE AXIS AND NORMAL TIC MARKS
C  ARE PLOTTED AT THE STANDARD INTENSITY LEVEL.  THE
C  SECONDARY TIC MARKS ARE PLOTTED IN THE "VERY DIM" MODE.
C
      REAL          XCRD,YCRD
      INTEGER       FLAG
C
      COMMON        /PLOT/SEGM
      INTEGER*4     SEGM(1000)
C
      IF (FLAG.LE.1) THEN
        CALL UGLINE(' ',XCRD,YCRD,FLAG,SEGM)
      ELSE
        CALL UGLINE('VDIM',XCRD,YCRD,FLAG-2,SEGM)
      END IF
      RETURN
C
      END
C*****************************************************************
      SUBROUTINE    TSUB(XCRD,YCRD,VALU,FLAG)
C
C  TEXT SUBROUTINE:  IF FLAG=1, A LABEL IS GENERATED FOR A
C  HORIZONTAL AXIS.  IF FLAG=2, A LABEL IS GENERATED FOR A
C  VERTICAL AXIS.
C
      REAL          XCRD,YCRD,VALU
      INTEGER       FLAG
C
      COMMON        /PLOT/SEGM
      INTEGER*4     SEGM(1000)
C
      CHARACTER*10  STRG
      INTEGER       LENG
C
      IF (FLAG.EQ.1) THEN
        CALL UGCNVF(VALU,3,STRG,LENG)
        CALL UGTEXT('SIZE=0.15,CENTER',XCRD,YCRD-0.2,
     X    STRG(11-LENG:10),SEGM)
      ELSE IF (FLAG.EQ.2) THEN
        CALL UGCNVF(VALU,0,STRG,LENG)
```

```
         CALL UGTEXT('SIZE=0.15,RIGHT',XCRD-0.2,YCRD,
     X     STRG(11-LENG:10),SEGM)
       END IF
       RETURN
C
       END
```

The data supplied to this program is shown below. The numerical data used in this example was supplied in a private communication from the SLAC-LBL group working with the SPEAR storage ring at SLAC. This data is the earliest available data and does not represent the current state of knowledge. The possible error in the cross-section measurement is as much as 350 nb near the peak. The cross-section is therefore only known to 2 or 3 significant figures and not the 6 figures that a naive reading of the data might suggest. This is one of the reasons why a logarithmic vertical axis is more appropriate than a linear one.

```
         24THE DISCOVERY OF Y(3095)
            LL  LLLLLLLL LL G
         12ENERGY (GEV)
            LLLLL   L
         18CROSS SECTION (NB)
            LLLL  LLLLLL  LL
         91
```

3.08702	26.64	3.08748	23.88	3.08914	19.12
3.08934	28.29	3.08938	24.32	3.08988	18.36
3.09008	18.29	3.09020	27.48	3.09090	34.18
3.09106	27.47	3.09120	29.80	3.09134	43.95
3.09150	35.34	3.09222	126.90	3.09274	260.20
3.09278	276.67	3.09296	293.51	3.09312	279.88
3.09380	761.47	3.09384	1121.26	3.09396	1124.99
3.09414	1401.76	3.09416	1237.14	3.09418	1787.48
3.09420	2006.46	3.09424	2072.16	3.09424	2116.46
3.09426	1576.38	3.09428	1474.64	3.09428	2159.31
3.09440	1426.50	3.09448	1755.02	3.09448	1715.47
3.09450	1864.18	3.09460	1648.21	3.09464	2309.35
3.09470	2254.99	3.09470	2390.31	3.09472	2375.27
3.09476	1693.46	3.09476	2015.47	3.09478	2043.66
3.09482	1798.43	3.09484	2560.76	3.09484	1808.28
3.09490	3163.26	3.09492	2376.27	3.09494	2329.92
3.09498	2614.89	3.09498	2272.29	3.09502	2436.82
3.09504	2812.99	3.09506	1969.30	3.09510	2432.32
3.09510	2301.46	3.09510	1933.90	3.09512	2769.29
3.09512	2881.40	3.09514	3344.97	3.09520	2053.62
3.09524	1985.32	3.09526	2017.90	3.09528	2367.41
3.09542	2434.28	3.09550	2233.71	3.09576	1869.32
3.09576	2018.33	3.09580	1538.77	3.09584	1591.03
3.09592	2067.37	3.09606	1409.14	3.09606	1762.93
3.09610	1688.89	3.09610	1450.55	3.09618	1307.14
3.09618	1576.81	3.09620	1389.69	3.09628	1139.31
3.09662	1190.66	3.09678	560.21	3.09700	796.24
3.09702	580.49	3.09750	476.03	3.09778	291.48
3.09780	295.05	3.09810	331.34	3.09898	141.51
3.10082	102.40	3.10198	115.01	3.11122	56.67

3.12954 39.80

When this data is supplied to the program, the result is the graph shown in Figure 2.5.1.

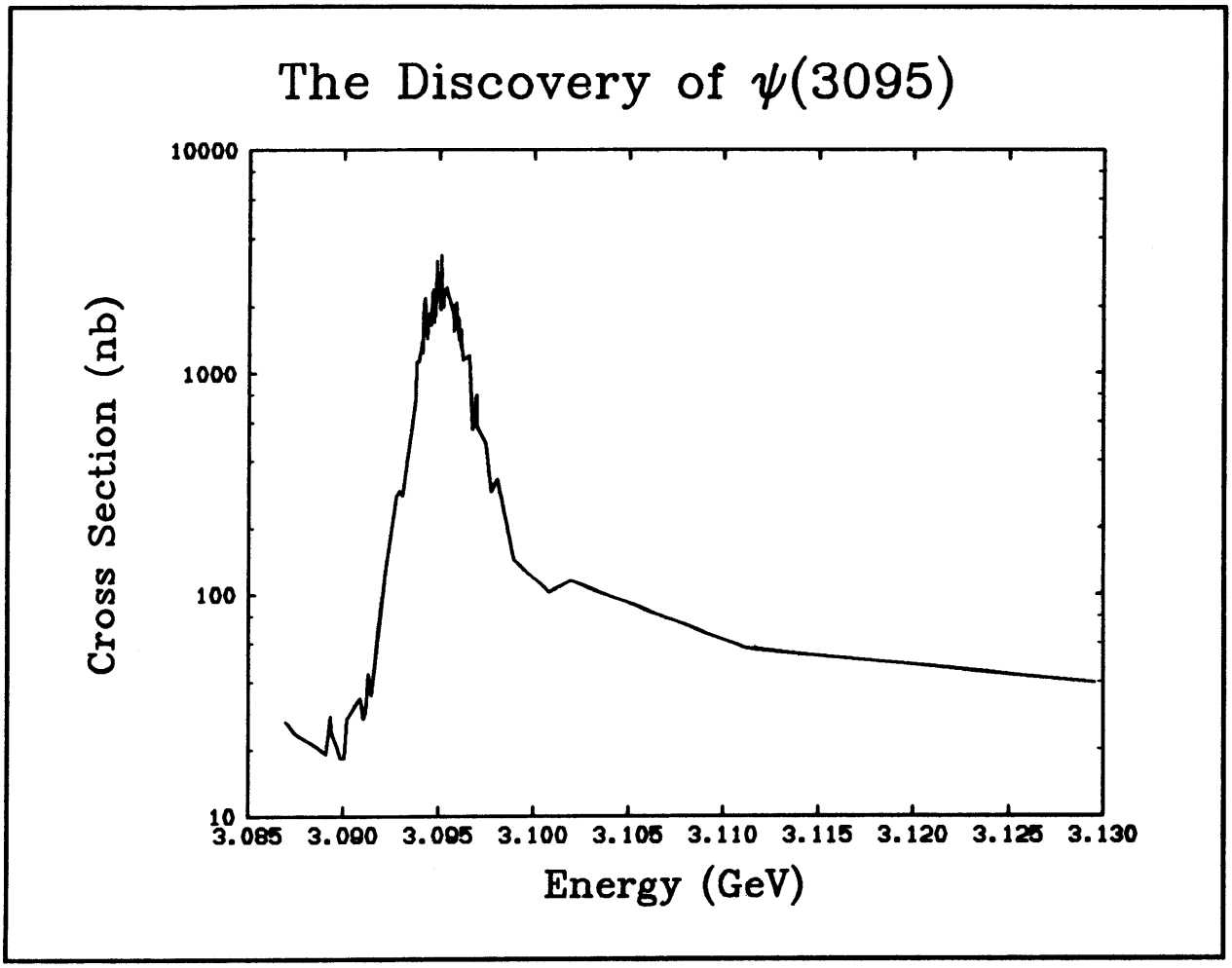

Figure 2.5.1: A Graph Produced by the Program.

SECTION 2.6: CONTOUR PLOTTING

This section describes a subroutine which can generate a contour plot of a three-dimensional surface. A sample program which produces a contour plot is shown.

SECTION 2.6.1: SUBROUTINE UGCNTR

This subroutine may be used to generate the description of a contour plot. This subroutine calls user supplied subroutines to process the graphic data; one subroutine is called to process

line data and a second subroutine is called to process the label information. The surface is given by the Z coordinates of points above a rectangular grid in the X-Y plane and the surface is approximated by these rectangular patches. The contour lines consist of "primary" and "secondary" contours. The primary contours will be labeled where they cross the boundary of the surface. The reader should refer to subroutine UGLNDX for help in assigning values to the parameters CNTRLO, CNTRHI, and NCNTR which result in "round numbers" being used for the contour lines.

This subroutine works best when the function being plotted is reasonably smooth. Very jagged functions give results that are hard to interpret. The two-dimensional histogram plotter that is described later is much better at displaying jagged functions.

A discussion of an algorithm similar to the one used here will be found in [Cot69]. Additional information about contour plotting will be found in [IBM--, Mor68, and War78].

The calling sequence is:

```
CALL UGCNTR(OPTIONS,LINSUB,TXTSUB,ARRAY,MDIM,NDIM,
            CNTRLO,CNTRHI,NCNTR,WKAREA,LDIM)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- NSCL=<value> The number of secondary contour lines between the primary contour lines. The default value is 0.
- TOLER=<value> An internal tolerance which may have to be adjusted if the X, Y, and Z coordinates take on large values. The default value is 0.0001.

LINSUB The entry point of the line end point subroutine.

TXTSUB The entry point of the label subroutine.

ARRAY A floating point two-dimensional array which contains the X, Y, and Z coordinates of the points on the surface. The format of ARRAY is:

```
--  X1   X2  ... XN
Y1  Z11  Z21 ... ZN1
Y2  Z12  Z22 ... ZN2
...      ...
YM  Z1M  Z2M ... ZNM
```

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YM) must be monotonically increasing but do not have to be equally spaced. The entry ARRAY(1,1) is not used and is shown as "--" in the above matrix. See the note at the end of this section for information about how to call this subroutine when the dimensions of ARRAY are not known until execution time.

MDIM The first dimension of ARRAY; that is, M+1.

NDIM The second dimension of ARRAY; that is, N+1.

CNTRLO A floating point value giving the Z value of the lowest primary contour.

CNTRHI A floating point value giving the Z value of the

highest primary contour.

NCNTR The number of primary contours.

WKAREA A full word, fixed point array which will be used as a work area. The amount of space that is needed in this array depends on the dimension of the array ARRAY. A dimension of NDIM*MDIM/15 will be more than sufficient in most cases.

LDIM The dimension of WKAREA.

The errors detected by this subroutine are:

1(3): The bounds of the two dimensional array must be at least 3 by 3 to define a valid surface.

2(3): The work area array is not large enough.

3(4): There is something substantially wrong with the definition of the surface.

The solution to this problem is an approximation in that a unique surface is not defined by the rectangular array. A necessary requirement to produce good contour plots is to have a fine enough mesh that no more than one contour line goes through each rectangular patch in the neighborhood of a saddle point of the surface.

The skeleton for the line end point subroutine is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT)
      REAL          XCRD,YCRD
      INTEGER       BBIT
        ...
      END
```

The value of BBIT may be 0, 1, 2, or 3. The values of 0 or 1 are the blanking bits for the primary contours. The values 2 or 3 are the blanking bits, incremented by 2, of the secondary contours.

The skeleton for the label subroutine is:

```
      SUBROUTINE    TXTSUB(XCRD,YCRD,VALUE,FLAG)
      REAL          XCRD,YCRD,VALUE
      INTEGER       FLAG
        ...
      END
```

The X and Y coordinates are those of the point where the primary contour line crosses the boundary. FLAG can have a value of 0, 1, 2, or 3 and indicates which boundary has been crossed. 0 means the left boundary has been crossed, 1 means bottom, 2 means right, and 3 means top. It is the duty of TXTSUB to convert VALUE to a character string and plot that string at the proper offset from (XCRD,YCRD).

There is a problem associated with the parameter ARRAY in the calling sequence. For some applications, the dimensions values, MDIM and NDIM, are not known until execution time. In such a case, it is not possible to have an array of the correct dimension declared in the source code. The way to get around this problem is to declare a large one-dimensional array in the program and then save the two-dimensional array in FORTRAN's

column-wise order, that is, Y1,...,YM are stored in ARRAY(2),...,ARRAY(MDIM), X1,Z11,...,Z1M are stored in ARRAY(MDIM+1),...,ARRAY(2*MDIM), etc.

SECTION 2.6.2: AN EXAMPLE

This example shows how simple contour plots may be produced by the preceding subroutine. Of particular interest is the way that the same label plotting subroutine is used for both the axis and contour plot labels. The error processing subroutine is necessary because the picture is rather complicated and the graphic segment array is small. The complete program is:

```
      PROGRAM        AGCNTR
C
C  SAMPLE PROGRAM:   A SIMPLE CONTOUR PLOTTER
C
      EXTERNAL       LSB1,LSB2,TSUB
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(500)
C
      REAL           VPRT(2,2),WDOW(2,2)
      REAL           ARRY(20,30)
      INTEGER        WKSP(40)
      REAL           XCRD,YCRD
      INTEGER        INT1,INT2
C
      DATA           VPRT/ 0.00, 0.00, 1.45, 1.20/
      DATA           WDOW/-0.30,-0.20, 1.15, 1.00/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  SELECT THE DUPLEX CHARACTER GENERATOR.
      CALL UGOPEN('VEP12FF,GENIL',99)
      CALL UGFONT('DUPLEX')
C
C  GENERATE THE DATA:  THE DOUBLE LOOP GENERATES A SIMPLE
C  3-DIMENSIONAL SURFACE.
      DO 102 INT1=2,30
        XCRD=(FLOAT(INT1)-2.00)/(30.00-2.00)
        ARRY(1,INT1)=XCRD
        DO 101 INT2=2,20
          YCRD=(FLOAT(INT2)-2.00)/(26.00-2.00)
          ARRY(INT2,1)=YCRD
          ARRY(INT2,INT1)=7.0*XCRD*YCRD-3.0
     X      -2.0*EXP(-15.0*((XCRD-0.50)**2+(YCRD-0.50)**2))
     X      +5.0*EXP(-5.0*(XCRD**2+(YCRD-0.25)**2))
  101   CONTINUE
  102 CONTINUE
C
C  PLOT TITLES AND AXES:  FIRST A FRESH PLOTTING SPACE IS
C  REQUESTED, THEN THE INITIAL DRAWING SPACE IS CREATED,
C  THE SEGMENT IS CLEARED, AND THE TITLES ARE ADDED TO THE
```

```
C  SEGMENT FOLLOWED BY THE AXES.
      CALL UGPICT('CLEAR',0)
      CALL UGDSPC('PUT',VPRT(1,2),VPRT(2,2),1.0)
      CALL UGWDOW('PUT',VPRT,WDOW)
      CALL UGINIT('CLEAR',SEGM,500)
      CALL UGXTXT('CENTER,SIZE=0.05',0.45,0.90,
     X  'CONTOUR PLOT EXAMPLE',
     X  ' LLLLLL  LLL  LLLLLL',SEGM)
      CALL UGLINE(' ',0.00,0.00,0,SEGM)
      CALL UGLINE(' ',1.00,0.00,1,SEGM)
      CALL UGLINE(' ',1.00,0.75,1,SEGM)
      CALL UGLINE(' ',0.00,0.75,1,SEGM)
      CALL UGLINE(' ',0.00,0.00,1,SEGM)
      CALL UGLNAX('NSTM=3',LSB1,TSUB,1,
     X  0.00,-0.10,1.00,-0.10,0.00,1.00,6)
      CALL UGLNAX('NSTM=4',LSB1,TSUB,0,
     X  -0.15,0.00,-0.15,0.75,0.00,0.75,4)
C
C  PLOT THE DATA:  THE CONTOUR PLOTTING SUBROUTINE IS
C  CALLED AND THE SEGMENT IS TRANSMITTED.
      CALL UGCNTR('NSCL=3',LSB2,TSUB,
     X  ARRY,20,30,-3.0,3.0,7,WKSP,40)
      CALL UGWRIT(' ',0,SEGM)
C
C  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C  AND THE PROGRAM STOPS.
      CALL UGCLOS(' ')
      STOP
C
      END
C*********************************************************************
      SUBROUTINE    LSB1(XCRD,YCRD,FLAG)
C
C  LINE SEGMENT SUBROUTINE:  THE AXIS AND NORMAL TIC MARKS
C  ARE PLOTTED AT THE STANDARD INTENSITY LEVEL.  THE
C  SECONDARY TIC MARKS ARE PLOTTED IN THE "VERY DIM" MODE.
C
      REAL          XCRD,YCRD
      INTEGER       FLAG
C
      COMMON        /PLOT/SEGM
      INTEGER*4     SEGM(500)
C
      IF (FLAG.LE.1) THEN
        CALL UGLINE(' ',XCRD,YCRD,FLAG,SEGM)
      ELSE
        CALL UGLINE('VDIM',XCRD,YCRD,FLAG-2,SEGM)
      END IF
      RETURN
C
      END
C*********************************************************************
      SUBROUTINE    LSB2(XCRD,YCRD,FLAG)
C
C  LINE SEGMENT SUBROUTINE:  THE PRIMARY CONTOURS ARE
```

```
C  PLOTTED AT THE STANDARD INTENSITY LEVEL IN THE SOLID
C  MODE.  THE SECONDARY CONTOURS ARE PLOTTED IN THE "VERY
C  DIM" AND "DASHED" MODE.
C
      REAL            XCRD,YCRD
      INTEGER         FLAG
C
      COMMON          /PLOT/SEGM
      INTEGER*4       SEGM(500)
C
      IF (FLAG.LE.1) THEN
        CALL UGLINE(' ',XCRD,YCRD,FLAG,SEGM)
      ELSE
        CALL UGLINE('VDIM,DASHED',XCRD,YCRD,FLAG-2,SEGM)
      END IF
      RETURN
C
      END
C*****************************************************************
      SUBROUTINE    TSUB(XCRD,YCRD,VALU,FLAG)
C
C  TEXT SUBROUTINE:  FLAG MAY HAVE THE VALUE 0, 1, 2, OR 3
C  SIGNIFYING THE LEFT, DOWN, RIGHT, OR UP SIDE.  FOR EACH
C  VALUE, THE COORDINATES ARE OFFSET AN APPROPRIATE AMOUNT.
C
      REAL            XCRD,YCRD,VALU
      INTEGER         FLAG
C
      COMMON          /PLOT/SEGM
      INTEGER*4       SEGM(500)
C
      CHARACTER*10    STRG
      INTEGER         LENG
C
      CALL UGCNVF(VALU,2,STRG,LENG)
      IF (FLAG.EQ.0) THEN
        CALL UGTEXT('SIZE=0.015,RIGHT',XCRD-0.02,YCRD,
     X    STRG(11-LENG:10),SEGM)
      ELSE IF (FLAG.EQ.1) THEN
        CALL UGTEXT('SIZE=0.015,CENTER',XCRD,YCRD-0.02,
     X    STRG(11-LENG:10),SEGM)
      ELSE IF (FLAG.EQ.2) THEN
        CALL UGTEXT('SIZE=0.015,LEFT',XCRD+0.02,YCRD,
     X    STRG(11-LENG:10),SEGM)
      ELSE
        CALL UGTEXT('SIZE=0.015,CENTER',XCRD,YCRD+0.02,
     X    STRG(11-LENG:10),SEGM)
      END IF
      RETURN
C
      END
C*****************************************************************
      SUBROUTINE    UGXERR(LEVL,SNAM,INDX)
C
C  SEGMENT OVERFLOW SUBROUTINE:  THE GRAPHIC SEGMENT
```

```
C  IS TRANSMITTED AND RE-INITIALIZED.
C
      INTEGER        LEVL
      CHARACTER*8    SNAM
      INTEGER        INDX
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(500)
C
      IF (INDX.EQ.11) THEN
        CALL UGWRIT(' ',0,SEGM)
        CALL UGINIT('CONTINUE',SEGM,500)
        LEVL=0
      END IF
      RETURN
C
      END
```

The picture produced by this program is shown in Figure 2.6.1.

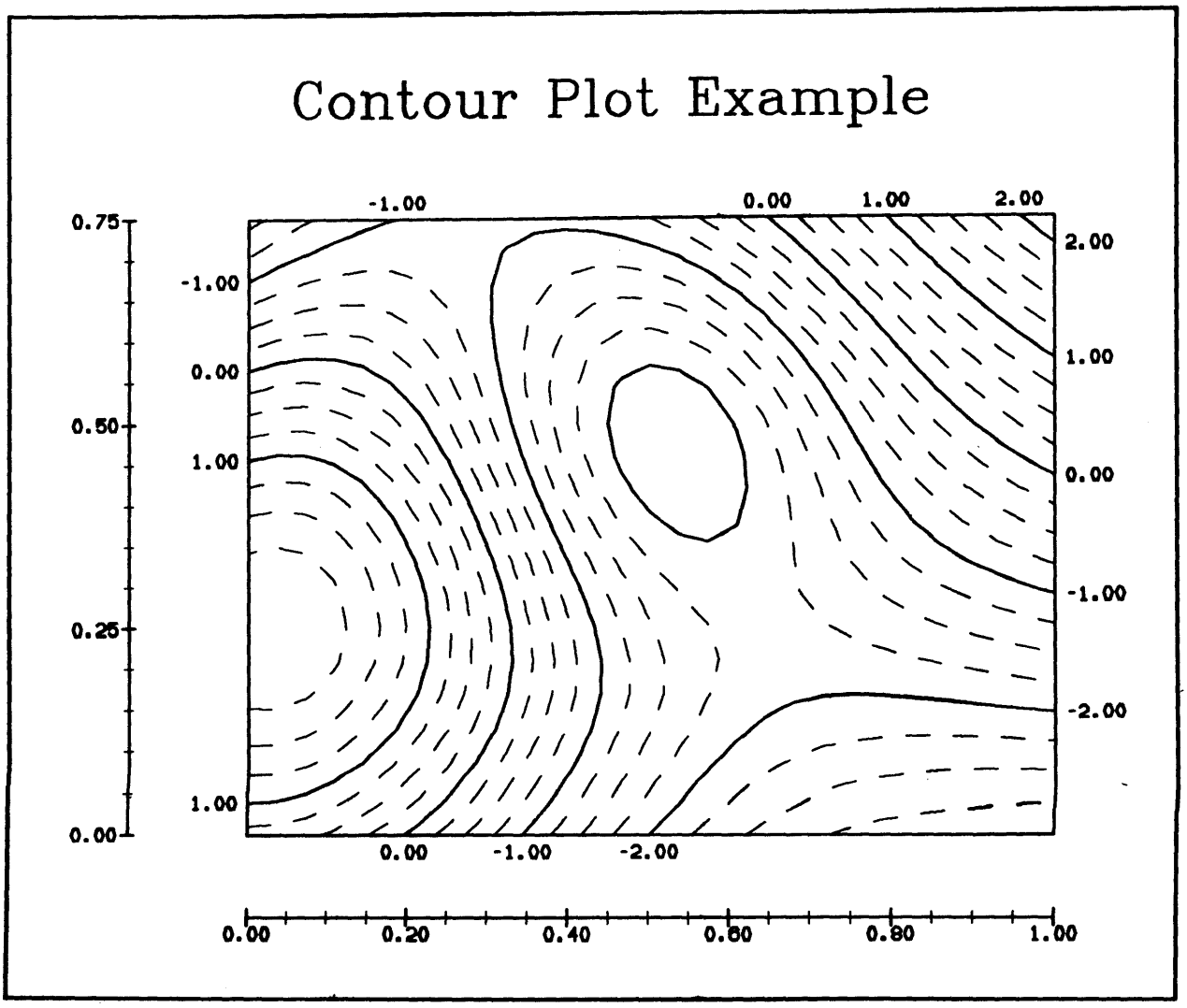

Figure 2.6.1: A Contour Plot Example.

SECTION 2.7: MESH SURFACE GENERATION

This section describes a subroutine which can generate a view of a three-dimensional mesh surface. A sample program which produces a plot of a mesh surface is shown.

SECTION 2.7.1: SUBROUTINE UGMESH

This subroutine may be used to generate the description of a point or parallel projection of a three-dimensional mesh surface with hidden lines eliminated. This subroutine calls a user supplied subroutine to process the line end point data. A mesh surface is a surface which is defined by giving the Z coordinates of points above a rectangular grid in the X-Y plane. The surface description is formed by joining adjacent points with straight lines.

This subroutine works best when the function being plotted is reasonably smooth. Very jagged functions give results that are hard to interpret. The two-dimensional histogram plotter that is described later is much better at displaying jagged functions.

Algorithms of this nature have been described in [Kub68, Wil72, Wri72, Bar72, and Wat74]. The algorithm used in UGMESH is based on the information in [Wri72, and Bar72]. Additional examples of this type of computer generated picture will be found in [Pru73, and Pru75].

The calling sequence is:

```
CALL UGMESH(OPTIONS,LINSUB,ARRAY,MDIM,NDIM,TRANS,WKAREA,LDIM)
```

The parameters in the calling sequence are:

OPTIONS A character string which may contain any of the following items:

- LOWER Indicates that the under side of the surface is to be generated. Normally the upper side of the surface is generated. Two calls to this subroutine are necessary to generate the full view of a surface.
- NOCOMN When both the upper and lower sides of the surface are produced, certain lines (for example, the front edge) will be duplicated. The programmer may use this option to suppress the common lines on either the upper or lower surface.
- TOLER=<value> An internal tolerance which may have to be adjusted if the X, Y, and Z coordinates take on large values. The default value is 0.00005.

LINSUB The entry point of the line end point subroutine.

ARRAY A floating point two-dimensional array which contains the X, Y, and Z coordinates of the points on the surface. The format of ARRAY is:

```
--   X1   X2  ... XN
Y1   Z11  Z21 ... ZN1
Y2   Z12  Z22 ... ZN2
...            ...
YM   Z1M  Z2M ... ZNM
```

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YM) must be monotonically increasing; they do not have to be equally spaced, but better pictures usually result if this is the case. The entry ARRAY(1,1) is not used and is shown as "--" in the above matrix. See the note at the end of the description of subroutine UGCNTR for information about how to call this subroutine when the dimensions of ARRAY are not known until execution time.

MDIM — The first dimension of ARRAY; that is M+1.

NDIM — The second dimension of ARRAY; that is N+1.

TRANS — A floating point array of dimension 31 containing the transformation as produced by subroutine UGTRAN. There are a few restrictions on the views which may be used here. First, the eye point must not be directly above the surface but must be off to one side; and second, the transformation should normally be defined with zero vectors for HDIR and UDIR.

WKAREA — A floating point array which will be used as a work area. The amount of space that is needed in this array depends on many things including the view and the shape of the surface. A dimension of fifteen times the maximum dimension of ARRAY will be more than sufficient in most cases.

LDIM — The dimension of WKAREA. If a value larger than 32767 is specified, only the first 32767 words in WKAREA will be used.

The errors detected by this subroutine are:

1(3): The bounds of the two dimensional array must be at least 3 by 3 to define a valid surface.

2(3): The work area array is not large enough.

The skeleton for the line end point subroutine is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT)
      REAL          XCRD,YCRD
      INTEGER       BBIT
        ...
      END
```

The value of BBIT may be 0 or 1 and is the blanking bit.

This subroutine does not solve the hidden line problem exactly. Instead, it produces an approximate solution. Under normal circumstances, it produces an acceptable picture with relatively fast execution speed. If the surface is very jagged, some line segments which should be eliminated may erroneously appear in the picture. This problem may usually be overcome by using a finer mesh. If the surface is very steep, some line segments which are visible may be erroneously eliminated. This second problem is accentuated by moving the eye position close to the surface when

using a point projection. It can often be overcome by moving to a distant eye position or using a parallel projection.

SECTION 2.7.2: AN EXAMPLE

This example shows how a projective view of a mesh surface may be produced by the preceding subroutine. The cross-hatching subroutine is used to generate an elaborate border for the picture. Two separate line drawing subroutines are used because the upper and lower parts of the surface are to be plotted at different intensity levels. The error processing subroutine is necessary because of the large number of line segments in the picture. The complete program is:

```
      PROGRAM        AGMESH
C
C  SAMPLE PROGRAM:  A SIMPLE MESH SURFACE PLOTTER
C
      EXTERNAL       LSB1,LSB2
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(500)
C
      REAL           XBDR(11),YBDR(11)
      INTEGER*4      BBTS
      REAL           REFP(3),VDIR(3),HDIR(3),UDIR(3)
      REAL           PTRN(31)
      REAL           ARRY(32,52)
      REAL           WKSP(780)
      REAL           XCRD,YCRD,RADU
      INTEGER        INT1,INT2
C
      DATA           XBDR/0.00,1.00,1.00,0.00,0.00,0.05,
     X                    0.05,0.95,0.95,0.05,0.00/
      DATA           YBDR/0.00,0.00,0.80,0.80,0.00,0.05,
     X                    0.75,0.75,0.05,0.05,0.00/
      DATA           BBTS/ZF0000000/
      DATA           REFP/ 50.0, 75.0, 40.0/
      DATA           VDIR/-50.0,-75.0,-30.0/
      DATA           HDIR/  0.0,  0.0,  0.0/
      DATA           UDIR/  0.0,  0.0,  0.0/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  SELECT THE DUPLEX CHARACTER GENERATOR.
      CALL UGOPEN('VEP12FF,GENIL',99)
      CALL UGFONT('DUPLEX')
C
C  GENERATE THE DATA:  THE DOUBLE LOOP CREATES A SIMPLE
C  3-DIMENSIONAL SURFACE.
      DO 102 INT1=2,52
        XCRD=INT1-27
        ARRY(1,INT1)=XCRD
        DO 101 INT2=2,32
```

```
            YCRD=INT2-17
            ARRY(INT2,1)=YCRD
            RADU=SQRT(XCRD**2+YCRD**2)
            ARRY(INT2,INT1)=
     X        ((750.0/(RADU**2+75.0))+5.0)*COS(0.4*RADU)
  101    CONTINUE
  102 CONTINUE
C
C   PLOT BORDER AND TITLE:  FIRST A FRESH PLOTTING SPACE IS
C   REQUESTED, THEN THE INITIAL DRAWING SPACE IS CREATED,
C   THE SEGMENT IS CLEARED, AND THE BORDER IS ADDED TO THE
C   SEGMENT FOLLOWED BY THE TITLE.
       CALL UGPICT('CLEAR',0)
       CALL UGDSPC('PUT',1.0,0.8,1.0)
       CALL UGTRAN(' ',REFP,VDIR,HDIR,UDIR,100.0,75.0,PTRN)
       CALL UGINIT('CLEAR',SEGM,500)
       CALL UGPLIN('VBRIGHT',XBDR,YBDR,10,BBTS,-5,SEGM)
       CALL UGXHCH(' ',LSB2,XBDR,YBDR,11,WKSP,10)
       CALL UGXTXT('CENTER,SIZE=0.04',0.5,0.65,
      X   'MESH SURFACE EXAMPLE',
      X   ' LLL   LLLLLL   LLLLLL',SEGM)
C
C   PLOT THE DATA:  THE MESH SURFACE PROCESSING PROGRAM IS
C   CALLED TO GENERATE THE SURFACE, AND THE SEGMENT IS
C   TRANSMITTED.
       CALL UGMESH('UPPER',LSB1,ARRY,32,52,
      X   PTRN,WKSP,780)
       CALL UGMESH('LOWER,NOCOMN',LSB2,ARRY,32,52,
      X   PTRN,WKSP,780)
       CALL UGWRIT(' ',0,SEGM)
C
C   TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C   AND THE PROGRAM STOPS.
       CALL UGCLOS(' ')
       STOP
C
       END
C*************************************************************
       SUBROUTINE     LSB1(XCRD,YCRD,FLAG)
C
C   LINE SEGMENT SUBROUTINE:  THE UPPER PART OF THE MESH
C   SURFACE IS PLOTTED AT THE STANDARD INTENSITY LEVEL.
C
       REAL           XCRD,YCRD
       INTEGER        FLAG
C
       COMMON         /PLOT/SEGM
       INTEGER*4      SEGM(500)
C
       CALL UGLINE(' ',XCRD,YCRD,FLAG,SEGM)
       RETURN
C
       END
C*************************************************************
       SUBROUTINE     LSB2(XCRD,YCRD,FLAG)
```

```
C
C  LINE SEGMENT SUBROUTINE:  THE LOWER PART OF THE MESH
C  SURFACE AND THE CROSS-HATCHING ARE PLOTTED IN THE
C  "VERY DIM" MODE.
C
      REAL           XCRD,YCRD
      INTEGER        FLAG
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(500)
C
      CALL UGLINE('VDIM',XCRD,YCRD,FLAG,SEGM)
      RETURN
C
      END
C*********************************************************************
      SUBROUTINE     UGXERR(LEVL,SNAM,INDX)
C
C  SEGMENT OVERFLOW SUBROUTINE:  THE GRAPHIC SEGMENT
C  IS TRANSMITTED AND RE-INITIALIZED.
C
      INTEGER        LEVL
      CHARACTER*8    SNAM
      INTEGER        INDX
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(500)
C
      IF (INDX.EQ.11) THEN
        CALL UGWRIT(' ',0,SEGM)
        CALL UGINIT('CONTINUE',SEGM,500)
        LEVL=0
      END IF
      RETURN
C
      END
```

The picture produced by this program is shown in Figure 2.7.1.

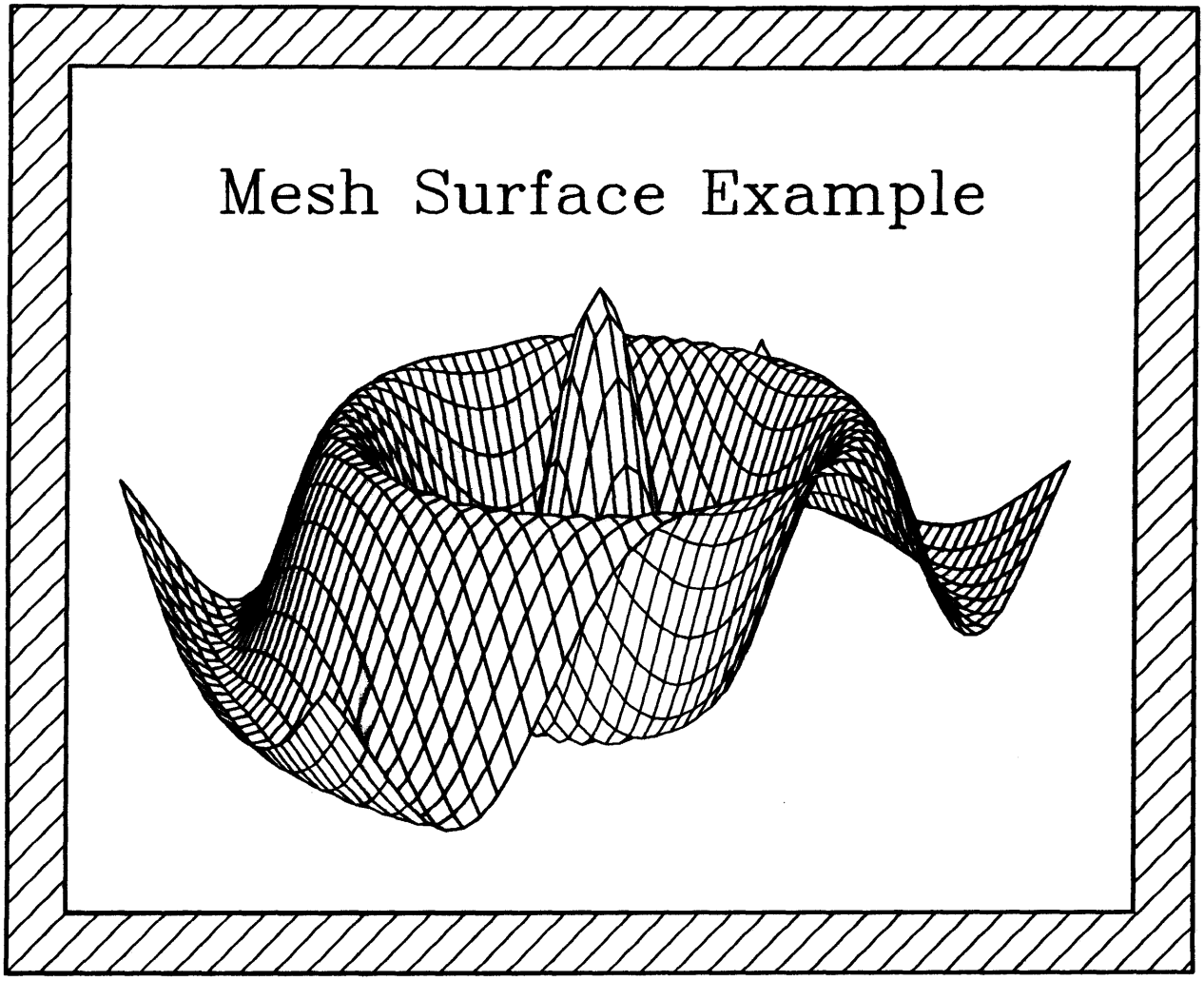

Figure 2.7.1: A Mesh Surface Example.

SECTION 2.8: TWO-DIMENSIONAL HISTOGRAM GENERATION

This section describes a pair of subroutines which can generate a view of a two-dimensional histogram. A two-dimensional histogram is a three-dimensional geometric form which is defined by giving the Z coordinates of the tops of columns above a rectangular grid in the X-Y plane. The first subroutine produces the pictures by drawing lines, the second subroutine produces polygon-fill data. The advantage of the first subroutine is that the pictures can be displayed on any line-drawing graphic device. The advantage of the second subroutine is that it is much faster than the first and does not need any work area; it however, does require a graphic device that can draw polygon-fill data. A sample program which produces a plot of a histogram is also shown.

SECTION 2.8.1: SUBROUTINE UG2DHG

This subroutine may be used to generate a line-drawn description of a parallel projection of a two-dimensional histogram with hidden lines eliminated. This subroutine calls a user supplied subroutine to process the line end point data.

The calling sequence is:

```
CALL UG2DHG(OPTIONS,LINSUB,ARRAY,MDIM,NDIM,TRANS,WKAREA,LDIM)
```

The parameters in the calling sequence are:

OPTIONS — A character string which may contain any of the following items:

TOLER=<value> An internal tolerance which may have to be adjusted if the X, Y, and Z coordinates take on large values. The default value is 0.00005.

LINSUB — The entry point of the line end point subroutine.

ARRAY — A floating point two-dimensional array which contains the X, Y, and Z coordinates of the points on the two-dimensional histogram. The format of ARRAY is:

Z0	X1	X2	...	X(N-1)	XN
Y1	Z11	Z21	...	Z(N-1)1	--
Y2	Z12	Z22	...	Z(N-1)2	--
...			...		...
Y(M-1)	Z1(M-1)	Z2(M-1)	...	Z(N-1)(M-1)	--
YM	--	--	...	--	--

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YM) must be monotonically increasing; they do not have to be equally spaced, but better pictures usually result if this is the case. The value Z0 is the Z coordinate of the base of the columns. The bounds of the (I,J)-th column are XI and X(I+1) in X, YJ and Y(J+1) in Y, and Z0 and ZIJ in Z. This means that the last row and column of ARRAY are almost unused; these unused values are shown as "--" in the above matrix. See the note at the end of the description of subroutine UGCNTR for information about how to call this subroutine when the dimensions of ARRAY are not known until execution time.

MDIM — The first dimension of ARRAY; that is M+1.

NDIM — The second dimension of ARRAY; that is N+1.

TRANS — A floating point array of dimension 31 containing the transformation as produced by subroutine UGTRAN. There are a number of restrictions on the views which may be used here. First, the transformation must be a parallel projection, second, the eye point must not be directly above the surface but must be off to one side, and third, the transformation should normally be defined with zero vectors for HDIR and UDIR.

WKAREA — A floating point array which will be used as a work area. The amount of space that is needed in this array depends on many things including the view and the shape of the surface. A dimension of fifteen times the sum of the dimensions of ARRAY will be more

than sufficient in most cases.

LDIM The dimension of WKAREA. If a value larger than 32767 is specified, only the first 32767 words in WKAREA will be used.

The errors detected by this subroutine are:

1(3): The bounds of the two dimensional array must be at least 3 by 3 to define a valid two-dimensional histogram.

2(3): The work area array is not large enough.

3(3): The transformation is not valid.

The skeleton for the line end point subroutine is:

```
      SUBROUTINE    LINSUB(XCRD,YCRD,BBIT)
      REAL          XCRD,YCRD
      INTEGER       BBIT
        ...
      END
```

The value of BBIT may be 0 or 1 and is the blanking bit.

SECTION 2.8.2: SUBROUTINE UG2DHP

This subroutine may be used to generate a description of a point or parallel projection of a two-dimensional histogram for a graphic device that supports the polygon-fill primitive. This subroutine calls a user supplied subroutine to process the polygon-fill data.

The calling sequence is:

```
CALL UG2DHP(OPTIONS,PFLSUB,ARRAY,MDIM,NDIM,TRANS)
```

The parameters in the calling sequence are:

OPTIONS This parameter is present for consistency with the other calling sequences; no items will be recognized.

PFLSUB The entry point of the polygon-fill subroutine.

ARRAY A floating point two-dimensional array which contains the X, Y, and Z coordinates of the points on the two-dimensional histogram. The format of ARRAY is:

Z0	X1	X2	...	X(N-1)	XN
Y1	Z11	Z21	...	Z(N-1)1	--
Y2	Z12	Z22	...	Z(N-1)2	--
...			...		...
Y(M-1)	Z1(M-1)	Z2(M-1)	...	Z(N-1)(M-1)	--
YM	--	--	...	--	--

The sequences (X1, X2, ..., XN) and (Y1, Y2, ..., YM) must be monotonically increasing; they do not have to be equally spaced, but better pictures usually result if this is the case. The value Z0 is the Z coordinate of the base of the columns. The bounds of the (I,J)-th column are XI and X(I+1) in X, YJ and Y(J+1) in Y, and Z0 and ZIJ in Z. This means that the last row and column of ARRAY are almost unused; these unused values are shown as "--" in the above matrix. See the note

at the end of the description of subroutine UGCNTR for information about how to call this subroutine when the dimensions of ARRAY are not known until execution time.

MDIM The first dimension of ARRAY; that is M+1.

NDIM The second dimension of ARRAY; that is N+1.

TRANS A floating point array of dimension 31 containing the transformation as produced by subroutine UGTRAN. There are a number of restrictions on the views which may be used here. First, the eye point must not be directly above the surface but must be off to one side, and second, the transformation should normally be defined with zero vectors for HDIR and UDIR.

The errors detected by this subroutine are:

1(3): The bounds of the two dimensional array must be at least 3 by 3 to define a valid two-dimensional histogram.

The skeleton for the polygon-fill subroutine is:

```
      SUBROUTINE    PFLSUB(XARY,YARY,FLAG)
      REAL          XARY(5),YARY(5)
      INTEGER       FLAG
        ...
      END
```

The value of FLAG may be 0 through 4. A value of 0 means the polygon being drawn is the top of a column, a value of 1 means it is the low X side of the column, 2 means it is the low Y side, 3 means the high X side, and 4 means the high Y side. The polygons that are passed to this subroutine are always quadrilaterals; the first and last points in XARY and YARY are identical.

The data that is supplied to subroutine PFLSUB can be used in many ways. One simple way to use this information is to simply draw the polygon with the polygon-fill subroutine in a color determined by the parameter FLAG. It has been found that this simple display can be improved if the polygon is also outlined in white or some other distinct color. This outlining makes the columns stand out more distinctly and can be done with a single call to the poly-line subroutine.

SECTION 2.8.3: AN EXAMPLE

This example shows how a parallel view of a two-dimensional histogram may be produced by subroutine UG2DHG. Since no error processing subroutine is used, the graphic segment in the program must be relatively large. The array that defines the histogram is initialized with a data statement; the unused entries in the array are set to 99.0. Also notice that the items in the data statement must be given in FORTRAN's column order, and thus appear as the transpose of the matrix described above under the ARRAY parameter. The complete program is:

```
      PROGRAM        AG2DHG
C
C  SAMPLE PROGRAM:   A SIMPLE 2-DIMENSIONAL HISTOGRAM
C                    GENERATOR
C
      EXTERNAL       LSUB
C
      COMMON         /PLOT/SEGM
      INTEGER*4      SEGM(2000)
C
      REAL           REFP(3),VDIR(3),HDIR(3),UDIR(3)
      REAL           PTRN(31)
      REAL           ARRY(10,12)
      REAL           WKSP(350)
      REAL           CRD1(100),CRD2(100),PT3D(3),PT2D(2)
      INTEGER*4      BBTS(4)
      INTEGER        NCRD,INT1
C
      DATA           ARRY/
     X    0.0, 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0,
     X    0.0, 0.5, 1.0, 0.5, 1.0, 2.0, 3.5, 2.0, 1.0,99.0,
     X    1.0, 1.0, 2.0, 1.0, 2.0, 3.0, 4.0, 3.0, 1.5,99.0,
     X    2.0, 3.0, 3.5, 3.0, 4.0, 4.5, 6.0, 4.5, 2.0,99.0,
     X    3.0, 2.0, 3.0, 3.5, 2.0, 3.0, 4.0, 3.0, 2.5,99.0,
     X    4.0, 2.5, 4.0, 3.0, 2.0, 2.5, 3.0, 2.5, 2.0,99.0,
     X    5.0, 4.0, 5.0, 5.0, 7.0, 4.0, 5.0, 6.0, 5.0,99.0,
     X    6.0, 4.5, 5.5, 9.0, 6.0, 4.5, 9.0, 7.0, 6.0,99.0,
     X    7.0, 4.0, 5.0, 6.0, 7.0, 5.0, 6.0, 6.5, 5.0,99.0,
     X    8.0, 3.0, 4.0, 5.0, 5.5, 4.0, 4.5, 4.0, 3.0,99.0,
     X    9.0, 2.0, 3.0, 4.0, 4.5, 4.0, 4.0, 3.0, 2.0,99.0,
     X   10.0,99.0,99.0,99.0,99.0,99.0,99.0,99.0,99.0,99.0/
      DATA           REFP/-14.0, -9.0, 10.0/
      DATA           VDIR/ 19.0, 13.0, -6.0/
      DATA           HDIR/  0.0,  0.0,  0.0/
      DATA           UDIR/  0.0,  0.0,  0.0/
C
C  INITIALIZE THE PROGRAM:  OPEN THE GRAPHIC DEVICE AND
C  SELECT THE DUPLEX CHARACTER GENERATOR.
      CALL UGOPEN('VEP12FF,GENIL',99)
      CALL UGFONT('DUPLEX')
C
C  INITIALIZE AND PLOT THE TITLE:  FIRST A FRESH PLOTTING
C  SPACE IS REQUESTED, THEN THE INITIAL DRAWING SPACE IS
C  CREATED, THE SEGMENT IS CLEARED, AND THE TITLE IS
C  ADDED TO THE SEGMENT.
      CALL UGPICT('CLEAR',0)
      CALL UGDSPC('PUT',1.0,0.8,1.0)
      CALL UGTRAN('PARALLEL,XLO=0.2,YHI=0.8',
     X  REFP,VDIR,HDIR,UDIR,25.0,15.0,PTRN)
      CALL UGINIT('CLEAR',SEGM,2000)
      CALL UGXTXT('CENTER,SIZE=0.035',0.20,0.70,
     X  'TWO',' LL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.035',0.20,0.60,
     X  'DIMENSIONAL',' LLLLLLLLLL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.035',0.20,0.50,
```

```
     X  'HISTOGRAM',' LLLLLLLL',SEGM)
      CALL UGXTXT('CENTER,SIZE=0.035',0.20,0.40,
     X  'EXAMPLE',' LLLLLL',SEGM)
C
C  PLOT THE AXIS LABELS:  THE LABELS ARE CONVERTED TO LINE
C  SEGMENTS AND TRANSFORMED TO THEIR PROPER POSITION IN
C  THREE DIMENSIONAL SPACE, AND THEN PROJECTED INTO TWO
C  DIMENSIONAL SPACE AND ADDED TO THE GRAPHIC SEGMENT.
      CALL UGCTOL('SIZE=0.5,CENTER',5.0,-1.0,
     X  'X-AXIS','   LLL',100,CRD1,CRD2,NCRD,BBTS)
      DO 101 INT1=1,NCRD
        PT3D(1)=CRD1(INT1)
        PT3D(2)=0.0
        PT3D(3)=CRD2(INT1)
        CALL UGPROJ(PTRN,PT3D,PT2D)
        CRD1(INT1)=PT2D(1)
        CRD2(INT1)=PT2D(2)
  101 CONTINUE
      CALL UGPLIN(' ',CRD1,CRD2,NCRD,BBTS,-NCRD,SEGM)
      CALL UGCTOL('SIZE=0.5,CENTER',4.0,-1.0,
     X  'Y-AXIS','   LLL',100,CRD1,CRD2,NCRD,BBTS)
      DO 102 INT1=1,NCRD
        PT3D(1)=0.0
        PT3D(2)=8.0-CRD1(INT1)
        PT3D(3)=CRD2(INT1)
        CALL UGPROJ(PTRN,PT3D,PT2D)
        CRD1(INT1)=PT2D(1)
        CRD2(INT1)=PT2D(2)
  102 CONTINUE
      CALL UGPLIN(' ',CRD1,CRD2,NCRD,BBTS,-NCRD,SEGM)
C
C  PLOT THE DATA:  THE 2-DIMENSIONAL HISTOGRAM PROCESSING
C  PROGRAM IS CALLED TO GENERATE THE FIGURE, AND THE
C  SEGMENT IS TRANSMITTED.
      CALL UG2DHG(' ',LSUB,ARRY,10,12,PTRN,WKSP,350)
      CALL UGWRIT(' ',0,SEGM)
C
C  TERMINATE THE PROGRAM:  THE GRAPHIC DEVICE IS CLOSED
C  AND THE PROGRAM STOPS.
      CALL UGCLOS(' ')
      STOP
C
      END
C*****************************************************************
      SUBROUTINE LSUB(XCRD,YCRD,FLAG)
C
C  LINE SEGMENT SUBROUTINE:  THE 2-DIMENSIONAL HISTOGRAM
C  IS PLOTTED AT THE STANDARD INTENSITY LEVEL.
C
      REAL          XCRD,YCRD
      INTEGER       FLAG
C
      COMMON        /PLOT/SEGM
      INTEGER*4     SEGM(2000)
C
```

```
      CALL UGLINE(' ',XCRD,YCRD,FLAG,SEGM)
      RETURN
C
      END
```

The picture produced by this program is shown in Figure 2.8.1.

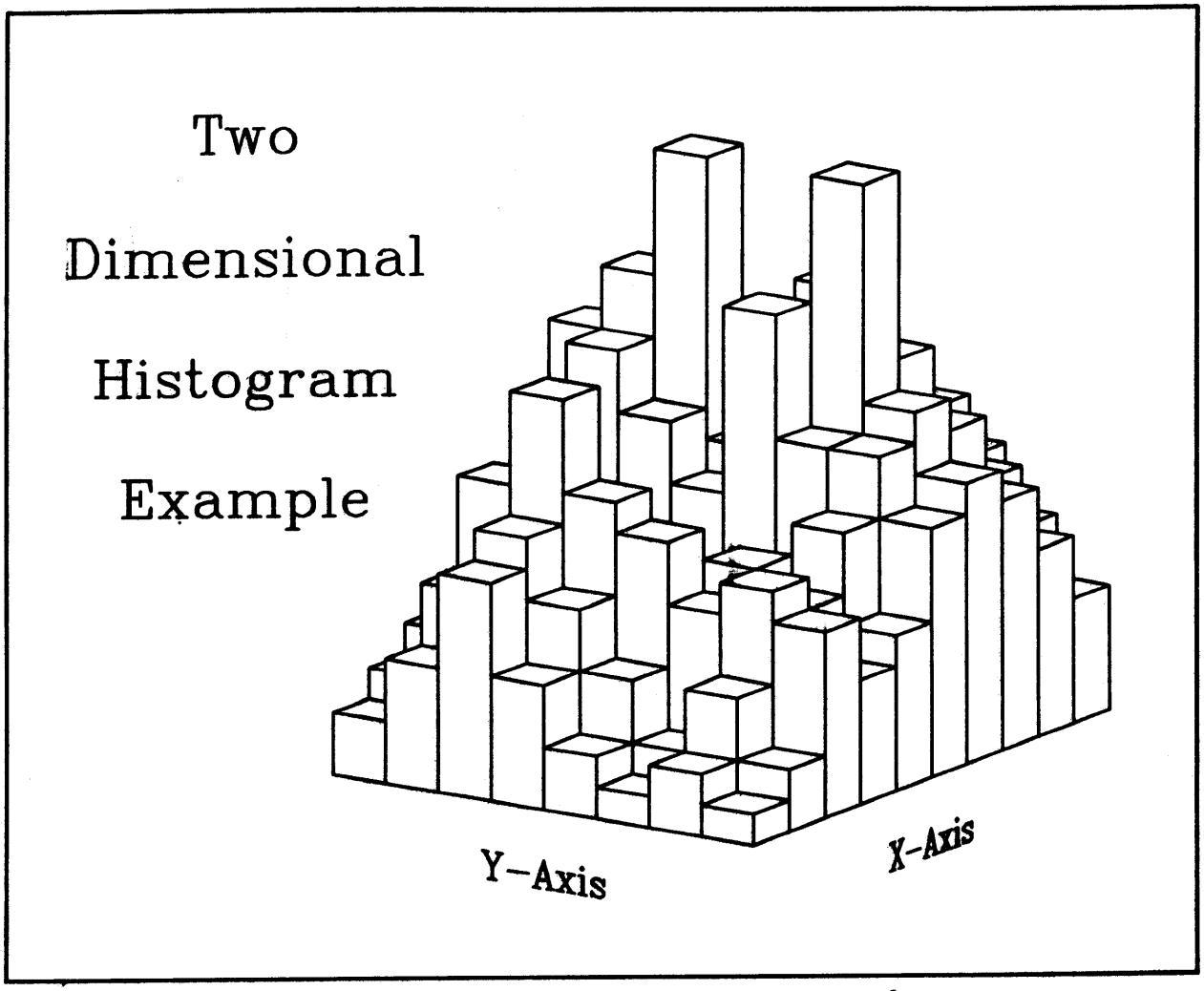

Figure 2.8.1: A Two-Dimensional Histogram Example.

This section contains a list of all of the publications that have been referenced in this document.

[Bar72] J. Barlow and B. Franek, Graphic Representation of Functions of Two Variables, Rutherford High Energy Laboratory, Chilton England, Report Number RHEL/R 259 (August 1972).

[Bea81] R. C. Beach, The Unified Graphics System for FORTRAN 77, Programming Manual, Stanford Linear Accelerator Center, Stanford California 94305, CGTM Number 203 (August 1981, Revised October 1983 and November 1985).

[Car78] I. Carlbom and J. Paciorek, Planar Geometric Projections and Viewing Transformations, Computing Surveys: The Survey and Tutorial Journal of the ACM, Volume 10, Number 4 (December 1978), pages 465-502.

[Cot69] G. Cottafava and G. Le Moli, Automatic Contour Map, Communications of the Association for Computing Machinery, Volume 12, Number 7 (July 1969), pages 386-391.

[Dix65] W. J. Dixon and R. A. Kronmal, The Choice of Origin and Scale for Graphs, Journal of the Association for Computing Machinery, Volume 12, Number 2 (April 1965), pages 259-261.

[Gia64] T. Giammo, A Mathematical Method for the Automatic Scaling of a Function, Journal of the Association for Computing Machinery, Volume 11, Number 1 (January 1964), pages 79-83.

[IBM--] Numerical Surface Techniques and Contour Map Plotting, International Business Machines Corporation, Form Number E20-0117 (undated).

[Kub68] B. Kubert, J. Szabo, and S. Giulieri, The Perspective Representation of Functions of Two Variables, Journal of the Association for Computing Machinery, Volume 15, Number 2 (April 1968), pages 193-204.

[Lew73] C. R. Lewart, Algorithm 463: Algorithms SCALE1, SCALE2, and SCALE3 for Determination of Scales on Computer Generated Plots, Communications of the Association for Computing Machinery, Volume 16, Number 10 (October 1973), pages 639-640.

[Mor68] S. P. Morse, A Mathematical Model for the Analysis of Contour-Line Data, Journal of the Association for Computing Machinery, Volume 15, Number 2 (April 1968), pages 205-220.

[Pru73] M. L. Prueitt, Fantastic Computer Pictures Give Us a New Look at Numbers, Popular Science, Volume 202, Number 2

(February 1973), pages 102-105.

[Pru75] M. L. Prueitt, Computer Graphics, 118 Computer-Generated Designs, Dover Publications Inc., New York 10014, (1975).

[War78] S. A. Ward, Real Time Plotting of Approximate Contour Maps, Communications of the Association for Computing Machinery, Volume 21, Number 9 (September 1978), pages 788-790.

[Wat74] S. L. Watkins, Algorithm 483: Masked Three-Dimensional Plot Program with Rotations, Communications of the Association for Computing Machinery, Volume 17, Number 9 (September 1974).

[Wil72] H. Williamson, Algorithm 420: Hidden-Line Plotting Program, Communications of the Association for Computing Machinery, Volume 15, Number 2 (February 1972), pages 100-103.

[Wri72] T. Wright, A Two-Space Solution to the Hidden Line Problem for Plotting Functions of Two Variables, National Center for Atmospheric Research, Boulder Colorado 80302, Report Number NCAR 72-26 (March 1972).