Three-Dimensional Plotting on a Two-Dimensional Surface

S. K. Skedzeleski
Communications Systems Research Section

Data collected over a two-dimensional surface are often test-displayed as a surface whose height above the plane represents the function's value at the corresponding coordinates. One algorithm for producing such a plane is described, using horizontal lines to define the surface. It has been implemented as a FORTRAN subroutine on the SDS 930 computer.

I. Introduction

It is often desirable to display data which are a function of two variables as a surface whose height above the coordinate plane represents the function's value at the corresponding coordinates. A very simple algorithm has been used for several years to display the results of radar mapping, but it has been done ad hoc and built directly into each program as it was needed. A FORTRAN subroutine which is very brief, both in time and space, has been written to do this plotting on the SDS 930 computer.

The subroutine, DDD, takes data one line at a time (in the constant Y direction) and outputs line segments to the plotting routines that are available on this system. One initialization call is needed to set the boundaries of the plot, and such information is passed in blank COMMON. It would be better to logically separate it in a labeled COMMON block, but RTFTRAN on the 930 computer does not have that capability.

II. The Algorithm

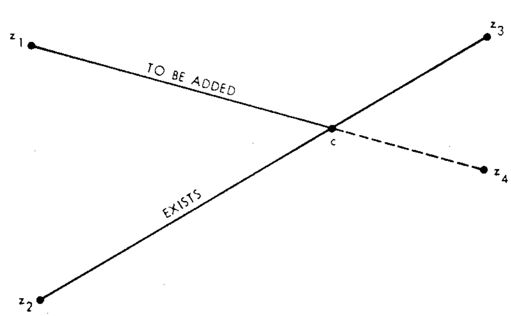
For reading clarity, the following notation is used:

$f_{x,y}$	the function value at (x,y)
$z_{x,y}$	$y + f_{x,y}$
NX	number of points in the x direction
Δx	spacing between sample points in the X direction
Δy	spacing between sample points in the Y direction

The algorithm used in DDD is very simple. For each point (x,y) a line segment is drawn from $(x - \Delta x, z_{x-\Delta x,y})$ to $(x, z_{x,y})$. The first point of each line is handled differently from the rest of the points since the pen is merely positioned there. This algorithm does not conceal hidden lines. A first-order improvement is easy to implement by keeping track of the maximum value of $z_{x',y'}$ for each $x' = x_{min}, x_{min} + \Delta x, \cdots, x_{max}$, over $y' = y_{min}, y_{min} + \Delta y, \cdots, y_{current}$ (initially zero). This is done in the array ZMAX (length NX). If the current point is hidden (i.e., $z_{x,y} < ZMAX_x$), then do not draw the line segment, but move the pen to $(x, z_{x,y})$ with the pen up. This produces acceptable plots when the grid size is not "too large."

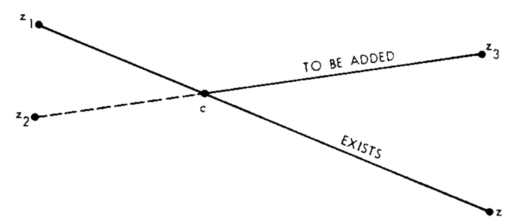
An improvement that is used in DDD is to try to determine at what point a line disappears and to draw the visible portion of the line segment $(x - \Delta x, z_{x-\Delta x,y}), (x, z_{x,y})$. This is implemented by keeping an extra array of length NX, called ZMAXPREDECESSOR. ZMAXPREDECESSOR gives the position of the other end of the line segment drawn to ZMAX, for each x position. (This is not totally true. If the line segment just drawn is a partial line segment—because the previous point was a hidden point—the value is still $z_{x-\Delta x,y}$, not the point at which this line segment became visible). One more value, $ZLAST$ $(= z_{x-\Delta x,y})$, is needed to calculate the point at which lines became visible and hidden. There are two cases which must be considered: a line disappearing and a line reappearing.

In Case I, the current point is hidden, but the last point was visible (line disappearing). In the drawing below, the line segment (z_2, z_3) was drawn previously; z_1 was the last point, and z_4 is the current point. We wish



to add the line segment (z_1, c) and then move the pen to z_4 with the pen up. Setting $z_1 = ZMAX(x - \Delta x)$, $z_2 = ZMAXPREDECESSOR_x$, $z_3 = ZMAX_x$ and $z_4 = z_{x,y}$, we see that we must have $z_1 + \alpha(z_4 - z_1) = z_2 + \alpha(z_3 - z_2)$, where α is the fraction of Δx from $x - \Delta x$ to Δx where the line segments cross. Solving for α we get $\alpha = z_1 - z_2 / z_1 - z_2 + z_3 - z_4$, which gives the crossing point $c = (x - \Delta x + \alpha \cdot x, z_2 + \alpha(z_3 - z_2))$.

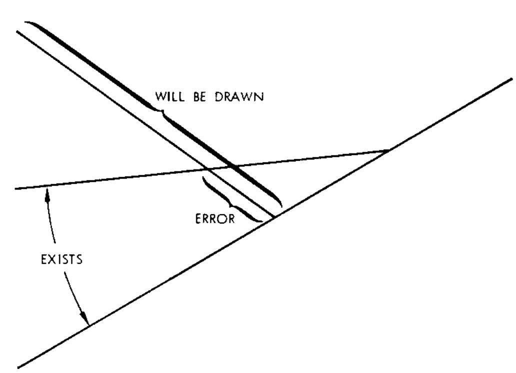
In Case II, the current point is visible, but the last point was hidden (line reappearing). In the drawing below, the line segment (z_1, z_4) was drawn previously; z_2 was the last point, and z_4 is the current point. We wish



to move the pen (still up) to point c, then move to the point z_3 with the pen down. Setting $z_1 = ZMAX_{x-\Delta x}$, $z_2 = ZLAST$, $z_3 = z_{x,y}$ and $z_4 = ZMAX_x$, we can substitute into the formula in Case I and determine point c.

A. Caveat

It should be noted that this method of removing hidden lines is not foolproof. There are situations in which it will cause a line segment to extend past another already drawn. Specifically, if between $x - \Delta x$ and x the line segment which ends at $(x, ZMAX_x)$ is not above all other line segments in that interval, an error can occur (see following sketch). The line segment drawn should be hidden,



yet DDD will draw it. In practice, this situation does not seem to occur often enough or with enough visual impact to justify doing a better job than is now done. However, it was found that by simplifying this procedure further by not keeping track of ZMAXPREDECESSOR and using ZLAST instead, obnoxiously visible crossings were occurring on almost all transitions from visible to hidden points. Hence the current degree of detail.

B. Sideviewing

The algorithm described so far is for a view of the surface which is directly in front of and slightly above the surface. The angle of view depends on rate of change of Y with vertical height and hence is essentially programmable. With certain kinds of data it is necessary to view the surface from a slight angle instead of head-on. One method of doing this is to offset each line in constant Y by one or more Δx positions. It is then possible to "peek around" tall parts of the surface. It is easy to modify the algorithm described so far to allow this. It suffices to keep ZMAX indexed on the position of the plotter, rather than having it coincident with a given x position. The points are then plotted at $x + x_{offset}$, and x_{offset} is incremented as each line is finished.

C. Edge Lines

If only horizontal lines are drawn, as in DDD, the line segments will be dangling on each end. It is particularly annoying on the near edge when side viewing is used, since this is where the eye expects to see the effects of "slicing" this portion of the surface from the entire surface. By adding a line segment from the last point of the current line to the last point of the previous line, the edges are effectively tied down. The edge on the far side is more difficult to connect since there are noticeable (bad) effects from connecting hidden points.

III. User's Guide

To draw a surface containing NX by NY points, $NY + 1$ calls are made to DDD. The first call initializes variables used by DDD, and the remaining calls actually draw the surface. Before the first call is made, the following variables in COMMON must be given values.

COMMON XLIM,YLIM,NX,XSCALE, IYDELT,IROUTE,IERR

XLIM,YLIM the boundaries of the plotting area (in inches for the plotter)

NX the number of points across a line

XSCALE scaling factor applied to $f_{x,y}$. It is used to adjust the height of each value above the baseline

IYDELT the number of Δx positions to shift each line, for side viewing

IROUTE must be set to 1

Calling Sequence:

Call DDD (*ARRAY,NY*) *ARRAY* is the array of *NY* data points to be plotted.

The first call to DDD returns the following values:

IROUTE

is set to 2

IERR

returns to 0 if the arrays in DDD are large enough to handle this plot, 1 otherwise. Currently there can be 512 Δx positions across the entire page (not across just one line of the plot).

Each succeeding call to DDD draws one line across the plot.

To restart DDD, set *IROUTE* to 1, and reset any variables that need to be changed. DDD does not automatically advance to a new sheet of paper or do any "nonstandard" centering of the plot on the page.