July 7, 2003

A Method for Incremental Generation of Parametric Curves

1 Background

A number of high performance algorithms have been devised to draw lines, circles and ellipses on raster scan display devices. The best of these involve incremental, all integer processes which step along the desired curve using a *nearest pixel* strategy.

For parametric curves, the conventional approach has been to calculate points on the curve and join the points with straight lines or curved splines. If the points are closely spaced, this technique may be acceptable. But for many curves there are regions with rapidly changing direction which require very closely spaced points and regions of slowly changing direction which require far fewer points. Devising a general parametric curve generator which adapts to local curvature, such as is done by Mathematica, is a nontrivial problem.

In this paper, another solution is presented. Although parametric curves are conceptually traced by increasing the parameter smoothly from one end of the line to the other and updating x and y, we will advance incrementally using expressions derived from x and/or y and update the parameter accordingly.

2 Parametric Curves

We are concerned with 2D parametric curves of the form,

$$x = f(t) \tag{1}$$
$$y = g(t) \tag{2}$$

where x and y are the coordinates of a point and t is the parameter.

For many curves t is set to zero for the start of the curve, and is increased monotonically to some maximum value at the end. Lines, circles and ellipses

can be defined and plotted parametrically, but many other curves including splines, and trig functions can also be put in parametric form.

Some examples of parametric curves follow.

GENERAL LINE

$$x = at + c \tag{3}$$
$$y = bt + d \tag{4}$$

GENERAL CIRCLE

$$x = V\cos(\theta) + c \tag{5}$$
$$y = V\sin(\theta) + d \tag{6}$$

ORIGIN CENTERED ELLIPSE

$$x = a\cos(\theta) \tag{7}$$
$$y = b\sin(\theta) \tag{8}$$

GENERAL CUBIC CURVE

$$x = at^3 + bt^2 + ct + d \tag{9}$$
$$y = pt^3 + qt^2 + rt + s \tag{10}$$

3 The Algorithm

To understand the proposed algorithm, it is convenient to examine an actual implementation. The following C code fragment illustrates the major elements of the algorithm.

```
t = 0;
while (t < tmax) {
      x = floor(f(t)+0.5);
      y = floor(g(t)+0.5);
      plot(x,y);
      dx = fabs(df(t));
      dy = fabs(dg(t));
      if (dx > dy) t += 1.0/dx;
      else t += 1.0/dy;
}
```

where $df(t)$ and $dg(t)$ return $f'(t)$ and $g'(t)$, respectively. For example, if $f(t) = at^2 + bt + c$ then, $df(t) = 2at + b$. $tmax$ is the maximum requested value of t.

The plotting loop always terminates because t increases monotonically at every iteration. This follows from the fact that dx and dy cannot both be zero during any iteration and the magnitude of the reciprocal of the larger of the two quantities must be some positive number.

In operation, the program loop shown above will plot the pixel which is nearest to the current computed x and y values. It then determines the direction of greatest motion relative to the two axes. A one pixel step is made in the direction of greatest motion and the parameter is updated for this movement. For example, if the slope of the curve at the current point is such that the magnitude of dx/dt is greater than that of dy/dt, a step in the x direction will be made. From the equation

$$\frac{dx}{dt} = df(t)$$

we are setting $dx = 1.0$ so that $dt = 1.0/|f(t)|$. t is updated by adding dt and the iteration is repeated until the curve is complete.

4 Caveats

Scan conversion of parametric curves has much in common with the solution of differential equations. One annoying similarity is that regions with rapidly changing derivatives (curvature) may result in unstable behavior or complete failure of the algorithm. These regions are associated with sharp corners in the plotted curve where the calculated derivative at the current point is inappropriate for determining the next point.

Solution strategies for pathological curves may involve subpixel calculations or adaptive plotting. In any case, additional work needs to be done to modify the above method for unconditionally robust behavior.

5 Comments

The method presented in this paper is so simple it has certainly been discovered by others. It is very likely that an exposition of the method has

been published elsewhere. Nevertheless, I have found neither any discussion nor any references to it in the literature. I welcome any citations that are brought to my attention.