1 Introduction

Suppose one wishes to draw a curve in three dimensions. Then with most graphics systems one must approximate the curve as a series of straight line segments and give function calls to the graphics library for each vertex (at the beginning of the curve and at each joint between straight line segments). In most cases it would be excessive to store all these vertices in a graphics program. Instead, one wishes to store the information to draw the curve by giving a set of control points which appear the same to the user as vertices. Then a curve drawing function uses these control points to draw the curve as a set of short, straight line segments.

There are several possibilities for using control points. If the curve passes through all the control points then it is said to interpolate the points. The mathematical function used to interpolate must be appropriate for what is being drawn. The most commonly used interpolation methods in computer graphics involve piecewise continuous cubics.

For the sake of smoothness in the interpolated curve, one not only wants the curve to be continuous but to have continuous derivatives. In order to represent curves in three dimensions a parametric form is used. Then derivatives are taken with respect to the parameter. Continuity of some order of derivative with respect to the parameter is not necessarily the same as continuity of derivatives like $dy/dx$ in the variables themselves.

In the following, several ways of representing curves by control points will be examined. For each the questions of smoothness and continuity will be addressed, along with how close the representation of the desired curve is in other ways.

1.1 Mathematical representation

The $x$, $y$, and $z$ coordinates of the curve to be drawn are given by parametric functions:

$$ x = x(t), \quad y = y(t), \quad z = z(t), $$

(1)

where the parameter $t$ ranges over a prescribed set of values. Derivatives are then,

$$ \frac{d^n x}{dt^n}, \quad \frac{d^n y}{dt^n}, \quad \frac{d^n z}{dt^n}, $$

(2)

for the $n$'th derivative. When all three of these are continuous at a specific value of $t$ then the curve is said to have $C^n$ continuity. The largest $n$ for which this is true is said to be the continuity of the curve (on a parametric basis).
2 Polynomial interpolation

When a set of N points is to be interpolated by a curve, one way to do it is to use a N-1 degree polynomial in \( t \) for each coordinate. This is usually unsatisfactory in appearance because of oscillations when N is greater than four. However there are special cases when N can be large because of other constraints on the polynomial. (Bézier curves)

2.1 Cubic polynomial interpolation through four points

If the polynomial is to be cubic then we have for the equations in matrix form:

\[
X = [x(t) \ y(t) \ z(t)] = [1 \ t \ t^2 \ t^3] \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \\ c_{41} & c_{42} & c_{43} \end{bmatrix} = TC. \tag{3}
\]

To find the coefficient matrix \( C \) one needs the values on the curve at four points through which it passes. The conventional way to choose these points is for parameter values \( t = 0, 1/3, 2/3, 1 \). Obviously other values could be used and unequal spacing in \( t \) could be used. If these points are \( p_1 \) at \( t = 0 \), \( p_2 \) at \( t = 1/3 \), etc., then a matrix can be made of the control point coordinates:

\[
P = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \\ x_4 & y_4 & z_4 \end{bmatrix} \tag{4}
\]

Then one can write the equation for \( C \) as

\[
P = M_I^{-1} C \tag{5}
\]

where the matrix \( M_I^{-1} \) is made up of \( T \) for the values of \( t = 0, 1/3, 2/3, 1 \) to give

\[
M_I^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1/3 & 1/9 & 1/27 \\ 1 & 2/3 & 4/9 & 8/27 \\ 1 & 1 & 1 & 1 \end{bmatrix}. \tag{6}
\]

Now the coefficient matrix for the cubic is \( C = M_I P \). where the inverse of \( M_I^{-1} \) is

\[
M_I = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -11/2 & 9 & -9/2 & 1 \\ 9 & -45/2 & 18 & -9/2 \\ -9/2 & 27/2 & -27/2 & 9/2 \end{bmatrix}. \tag{7}
\]

Thus the graphics code to draw the curve loops on \( t \) in small increments from 0 to 1 calling the vertex function with vertices given by \( X = TM_I P \).

2.2 Matrix inverse for \( M_I \)

One can find matrix inverses by Gaussian elimination, or by using a computer tool such as \texttt{octave}:
elmo{khc}˜/classes/636/f99/prob:1> octave
This is free software with ABSOLUTELY NO WARRANTY.
For details, type 'warranty'.

octave:1> A = [1 0 0 0; 1 1/3 1/9 1/27; 1 2/3 4/9 8/27; 1 1 1 1]
A =
1.00000 0.00000 0.00000 0.00000
1.00000 0.33333 0.11111 0.03704
1.00000 0.66667 0.44444 0.29630
1.00000 1.00000 1.00000 1.00000

octave:2> M = A\eye(4)
M =
1.00000 0.00000 0.00000 0.00000
-5.50000 9.00000 -4.50000 1.00000
9.00000 -22.50000 18.00000 -4.50000
-4.50000 13.50000 -13.50000 4.50000

octave:3> M*A
ans =
1.00000 0.00000 0.00000 0.00000
-0.00000 1.00000 0.00000 0.00000
0.00000 -0.00000 1.00000 0.00000
0.00000 0.00000 0.00000 1.00000

octave:4> exit

2.3 Cubic spline interpolation

When there are more than four points to be interpolated with cubics, one cannot just put a cubic through successive sets of four without having discontinuities in slope at the points where different cubic curves connect (at the so-called “knots” in the interpolation). But one can have not only continuous first derivatives but also continuous second derivatives at the knots if all the points are used in determining coefficients for all the needed cubics – one for each section between points to be interpolated, in general. The result is called the cubic spline interpolant.

Suppose there are N points to be interpolated and that their coordinates are labeled as \(x_i, y_i, z_i\) with \(i = 1, 2, \ldots N\). For the cubic between point \(i\) and point \(i + 1\) let the parametric formula for \(x\) be

\[
x(t) = a_i + b_i t + c_i t^2 + d_i t^3, \quad 0 \leq t \leq 1.
\]

There will be similar equations for \(y(t)\) and \(z(t)\). There are \(N - 1\) segments between given points, so there are \(4(N - 1)\) coefficients to be found for the \(x(t)\) equations. \(2(N - 1)\) equations come from the
requirement that the cubics interpolate all the points:

\begin{align}
    x_i &= a_i, \ i = 1, 2, \ldots N - 1 \\
    x_{i+1} &= a_i + b_i + c_i + d_i, \ i = 1, 2, \ldots N - 1.
\end{align}

(9) (10)

There are another \( N - 2 \) equations for equating slopes at the interior points and another \( N - 2 \) equations for equating 2nd derivatives at interior points:

\begin{align}
    b_{i+1} &= b_i + 2c_i + 3d_i, \ i = 1, 2, \ldots N - 2 \\
    c_{i+1} &= c_i + 3d_i, \ i = 1, 2, \ldots N - 2.
\end{align}

(11) (12)

Thus eqs.(9) through (12) provide \( 4N - 6 \) equations for the \( 4N - 4 \) unknown coefficients. The additional two values needed are the slopes at the first and the last point. These can be provided in several manners:

1. Give data specifying the slope at each end.
2. Take the slope at each end to be the slope of the straight line from the end point to the adjacent point.
3. Treat the points adjacent to the end points as not knots, that is make the same cubic interpolate the first three and last three points. This gives the needed two extra equations (the interpolation of the points next to the ends) for a spline with only \( N - 3 \) segments.

The matrix that would result from setting up these equations for solution would be large \((4(N−1))\) square, but it would be sparse and banded. There are special numerical methods to handle such cases of systems of equations.

The main drawback to the cubic spline interpolation is that changing one of the points interpolated will affect the appearance of the curve in segments far removed from the point changed. This lack of locality limits the usefulness of cubic spline interpolation in computer graphics.

3 Hermite cubic interpolation

The problem with simple cubic polynomial interpolation through four points is that it only works well for four control points. With more than four, there will be only \( C^0 \) continuity at at least one of the joining points, called knots, between different cubics. This problem can be eliminated by the cubic spline interpolation of many points, but then locality of influence of points on the curve is lost. An alternative which provides locality and \( C^1 \) continuity is to specify both the points to be interpolated and the slope at the knots. One way to do this is with Hermite cubics.

Hermite cubics are formed from four control numbers, but only two of them are the points to be interpolated. The other two are the slopes of the lines at these points. Thus one has to specify not only \( x(t) \) but also \( dx/dt \) at each point to be interpolated. The interpolation only goes between pairs of points before the cubic must be changed. But by specifying the same coordinate and slope for the end of one cubic and the beginning of the next, the complete curve has \( C^1 \) continuity. The matrix equation needed this time is \( X = T M_H P_H \) where \( P_H \) is made up as follows:

\[
P_H = \begin{bmatrix}
    x_1 & y_1 & z_1 \\
    x_2 & y_2 & z_2 \\
    \frac{dx}{dt} & \frac{dy}{dt} & \frac{dz}{dt} \\
    \frac{dx}{dt} & \frac{dy}{dt} & \frac{dz}{dt}
\end{bmatrix}.
\]

(13)
To find $\mathbf{M}_H$, proceed as above for $\mathbf{M}_I$, but use $d\mathbf{I}/dt$ at point 1 and point 2 for the 3rd and 4th rows of $\mathbf{M}^{-1}_H$:

$$
\mathbf{M}^{-1}_H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 \\
0 & 1 & 2 & 3
\end{bmatrix}
$$

(14)

and thus

$$
\mathbf{M}_H = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
-3 & 3 & -2 & -1 \\
2 & -2 & 1 & 1
\end{bmatrix}.
$$

(15)

### 4 Bézier curves

The main difficulty with the Hermite cubic curves is the need to specify derivatives at points as well as coordinates. One can calculate derivatives by numerical differentiation if values are given for additional coordinates to be used in the numerical derivatives. The simplest version of this yields a cubic Bézier curve.

#### 4.1 Cubic Bézier curves

Suppose one gives four control points. The first and last are to be interpolated by a cubic, as with the Hermite curve. But the second and third are used to find the slopes at the first and last. The line from the first to second control point is assumed to correspond to changing parameter $t$ from 0 to $1/3$. The line from the third control point to the last is assumed to correspond to changing parameter $t$ from $2/3$ to 1. Thus we get the matrix of values needed for $\mathbf{P}_H$ for the Hermite cubic curve from the control points $\mathbf{P}$ of the cubic Bézier curve (which is the same $\mathbf{P}$ as in eq.(4)) by multiplying them by a matrix to take the numerical derivative: $\mathbf{P}_H = \mathbf{D}_B \mathbf{P}$ where

$$
\mathbf{D}_B = \begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 \\
-3 & 3 & 0 & 0 \\
0 & 0 & -3 & 3
\end{bmatrix}.
$$

(16)

Thus the parametric equations for the cubic Bézier curve are $\mathbf{X} = \mathbf{T}\mathbf{M}_B \mathbf{P}$ with $\mathbf{M}_B = \mathbf{M}_H\mathbf{D}_B$ or

$$
\mathbf{M}_B = \begin{bmatrix}
1 & 0 & 0 & 0 \\
-3 & 3 & 0 & 0 \\
3 & -6 & 3 & 0 \\
-1 & 3 & -3 & 1
\end{bmatrix}.
$$

(17)

#### 4.2 Blending polynomials

Any polynomial representation in the form $\mathbf{X} = \mathbf{T}\mathbf{M}\mathbf{P}$ may be expressed as $\mathbf{X} = \mathbf{B}\mathbf{P}$ where $\mathbf{B} = \mathbf{TM}$ is the matrix of blending polynomials. Thus one interprets the generation of the parametric equations for the curve as multiplying control points times blending polynomials. For the particular case of the cubic Bézier curve we have

$$
\mathbf{B}_B = \begin{bmatrix}
(1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3
\end{bmatrix}.
$$

(18)

These curves are shown in Fig. 1.
4.3 Convex hulls and bounding boxes

Note that the peculiar set of blending polynomials for the Bézier cubic, given in eq.(18), just happen to be the terms of the binomial expansion of \((1 - t) + t\)^3. (This is like \((x + y)^3\) with \(x = 1 - t\) and \(y = t\).) Because of this, the sum of the blending polynomials is always one for all \(t\). This has the result that the curve, for all values of \(t\), must lie within the convex hull defined by the control points.

By definition, the convex hull for a parametric curve is the enclosing convex polyhedron having control points as vertices. Not all of the control points will necessarily be vertices of the convex polyhedron forming the convex hull. In two-dimensions, the convex hull is the convex polygon having control points as vertices and enclosing all control points. One can visualize the convex hull as the polygon formed by stretching a rubber band around pins placed on a drawing board at the locations of the vertices.

Mathematically, one can show the convex hull property for a set of vertices as follows. Let the control points be represented by \(p_i = [x_i, y_i, z_i]\). Now let \(X_m\) be the control point having the minimum value of \(x = x_m\). When a parametric curve is obtained by taking a sum of control points times corresponding blending polynomials, \(X(t) = BP\), with each blending polynomial being non-negative, and with the sum of the blending polynomials being one, then for any \(t\) the \(x\) that results will be a weighted sum of the control point \(x_i\)'s and must always be greater or equal to \(x_m\). This argument can be repeated for the control point with the largest \(x\) value, and for the largest and smallest \(y\) and \(z\) values as well. This establishes that the curve lies in a rectangular bounding box. To change from this bounding box to the convex hull, one only need rotate coordinates until another control point has the maximum or minimum for such variables and find a new bounding box. The intersection of these bounding boxes will be the convex hull.
4.4 Bernstein basis functions and general Bézier curves

The binomial expansion is

\[(\alpha + \beta)^n = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} \alpha^{n-i} \beta^i. \quad (19)\]

Substitute \(\alpha = 1 - t\) and \(\beta = t\) in eq.(19) and we get

\[\sum_{i=0}^{n} b^n_i(t) = 1, \quad b^n_i = \frac{n!}{i!(n-i)!}((1 - t)^{n-i} t^i. (20)\]

The \(b^n_i(t)\) are the Bernstein basis functions of order \(n\). Each is a \(n\)th degree polynomial. Thus the sum

\[x(t) = \sum_{i=0}^{n} x_i b^n_i(t) \quad (21)\]

is an \(n\)th degree polynomial parametric equation for coordinate \(x\). Thus, the general Bézier curve for \(n + 1\) control points is

\[X(t) = \sum_{i=0}^{n} b^n_i(t) p_i. (22)\]

As with the Bézier cubic, the parametric curve interpolates the first and last control points and lies within the convex hull defined by the control points.

5 B-splines

B-splines are a general classification of parametric curves defined by control points. The control is localized, and, in general, the curve does not interpolate any of the points. Cubic B-splines are widely used. They are mathematically similar to cubic Bézier curves but do not interpolate control points unless the same control point is repeated twice. They are formed by multiplying each control point by a basis function. The way they differ from the curves given in previous sections is that the parameter \(t\) is allowed to run over 0 to \(n\), with \(n\) being related to the number of control points. The basis functions are defined to make them zero for \(t\) outside the range for which its control point is to have an effect. The knots of the cubics formed are at different coordinates than the control points. Further discussion of splines is beyond the scope of this introduction. See, Kenneth H. Carpenter, “B-Splines,” http://www.eece.ksu.edu/~khc/classes/636/bsplpdf.pdf, for more on this topic.

6 X-splines

While B-Splines are widely used in computer graphics, especially in the form to “NURBS,” a more general form called X-splines may offer some advantages over B-splines. For a detailed description of X-Splines, see Carole Blanc and Christophe Schlick, “X-Splines: A Spline Model Designed for the End-User,” SIGGRAPH’95, Computer Graphics, pp. 377-386. (Currently this paper is available on the Internet at URL: http://www.acm.org/pubs/citations/proceedings/graph/218380/p377-blanc/)
7 Surface interpolation and approximation

When a surface is desired instead of a curve, then there must be two parameters for the parametric representation: \( x(u,v), y(u,v), z(u,v) \). The method of forming such parametric functions is a straightforward extension of that used with curves.

7.1 Bi-cubic patches

Hermite and Bézier cubics can be extended for surface as bi-cubic parametric functions. That is, for blending polynomials \( b_i(t) \) one forms the parametric equations for the surface as sums of products \( b_i(u)b_j(v) p_{i,j} \).

7.2 Quadrics

While cubics are most often used to get pleasing curves, for solids, use of quadratic functions in all three variables, called quadrics, are simpler and often suitable for representing solids. A quadric surface is not represented in parametric form but as a function of three variables set to zero:

\[
0 = f(x, y, z) = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}.
\] (23)

A quadric requires ten constants to define it. (Since the upper and lower off diagonal terms in the matrix multiply the same combinations of the variables, either the upper or lower constants may be taken to be zero.) A quadric can be a sphere, a cylinder, an ellipsoid, etc.

8 Exercises

1. Given two points: \( p_1 = (1.1, 2.1, 3) \) and \( p_2 = (3.1, 0, -1) \), and given that the derivatives, \( dx/dt, \) etc., are all zero at these points, find the Hermite cubic interpolant curve, as a formula for all of \( x(t), y(t), \) and \( z(t) \), which passes through these points with the specified derivative. What can be said about the physical slopes such as \( dy/dx \) when the parametric slopes are all zero?

   Find the numerical values of the coordinates on the interpolated curve for \( t = 1/3 \) and \( t = 2/3 \).

2. Given two points: \( p_1 = (1.1, 2.1, 3) \) and \( p_2 = (3.1, 0, -1) \), and given that the derivatives, \( dx/dt, \) etc., are all one at these points, find the Hermite cubic interpolant curve, as a formula for all of \( x(t), y(t), \) and \( z(t) \), which passes through these points with the specified derivative.

3. Plot, in the \( z = 0 \) plane, the projection of the curves (for \( 0 \leq t \leq 1 \)) for exercises 1 and 2. (Use gnuplot, ignore the \( z \) function. Put both curves on the same axes.)