We have seen that the Lagrange polynomial can oscillate wildly, except where contained between data points that are in close proximity.

We want our interpolating polynomial to have the same "shape" as the function at the data points — in the sense that the tangent lines to the polynomial and to the function agree at (x_i, f_i)

In other words, we seek polynomials that agree with f at $x_0, x_1, \ldots, x_m$ and that have first derivatives that agree with those of f at (x_i, f_i).

Thm: If $f \in C^1[a,b]$ and $x_0, \ldots, x_n \in [a,b]$ are distinct,

the interpolating polynomial of least degree agreeing with f and f' at $x_0, \ldots, x_n$ is the Hermite polynomial of degree at most $2n+1$ given by

$$H(x) = \sum_{j=0}^{n} f(x_j) H_j(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_j(x)$$

where $H_j(x) = [1 - 2(x - x_j) L_j'(x_j)] L_j^2(x)$

and $\hat{H}_j(x) = (x - x_j) L_j^2(x)$

where $L_j(x)$ denotes the j^{th} Lagrange coefficient polynomial of degree n.

Proof. See text, pgs 134-135. for a derivation showing $H(x)$ agrees with f and $H'(x)$ agrees with f' at $x_0, x_1, \ldots, x_n$.

To show the uniqueness of this polynomial:

Suppose that $P(x)$ is ~~and~~ another polynomial with

$$P(x_k) = f(x_k), \; P'(x_k) = f'(x_k) \quad k = 0, \ldots, n$$

and that the degree of $P(x)$ is at most $2n+1$.

Let $D(x) = H(x) - P(x)$.

Then $D(x)$ is a polynomial of degree at most $2n+1$ with $D(x_k) = 0$ and $D'(x_k) = 0$

Then D has zeros of multiplicity 2 at each x_k so

$$D(x) = (x-x_0)^2 \cdots (x-x_n)^2 Q(x)$$

Either $D(x)$ is of degree $2n+2$ or more which would be a contradiction.

or $\quad Q(x) \equiv 0 \Rightarrow D(x) \equiv 0 \Rightarrow P(x) = H(x)$

$\Rightarrow$ the polynomial is unique.

Unfortunately, a direct application of the theorem requires us to evaluate the Lagrange polynomials and their derivatives.

This is tedious even for small values of n.

Instead we use the Newton interpolatory divided difference formula:

Select a new sequence of nodes $z_0, z_1, \ldots, z_{2n+1}$:

$$z_{2i} = z_{2i+1} = x_i$$

and use $f'(x_0), f'(x_1), \ldots, f'(x_n)$ in place of the undefined first divided differences

$$f[z_0, z_1], \; f[z_2, z_2], \ldots, f[z_{2n}, z_{2n+1}]$$

The Hermite polynomial is then given by

$$H(x) = f[z_0] + \sum_{k=1}^{2n+1} f[z_0, \ldots, z_k](x - z_0) \cdots (x - z_{k-1})$$

Thus the derivative values are used in place of the undefined first divided differences, otherwise Newton's divided differences are produced as usual.

z	$f(z)$	first divided differences	second divided differences
$z_0 = x_0$	$f[z_0] = f(x_0)$	$f[z_0, z_1] = f'(x_0)$	
$z_1 = x_0$	$f[z_1] = f(x_0)$		
$z_2 = x_1$	$f[z_2] = f(x_1)$		
$z_3 = x_1$	$f[z_3] = f(x_1)$		

Q The following table lists data for the function described by $f(x) = e^{.1x^2}$

x	$f(x)$	$f'(x)$
$x_0 = 1$	1.105170918	0.221034186
$x_1 = 1.5$	1.252322716	0.375696814

What is the Hermite polynomial of degree 3 that agrees with the function $f(x)$ and its derivative at the nodes x_0 and x_1?

Ans. The divided difference table for the Hermite polynomial $H_3(x)$ is

z_i	$f[z_i]$	....		
1	1.105170918			
		0.2210...		
1	1.105170918		0.1465...	
		0.2943...		0.0324...
1.5	1.252322716		0.1627...	
		0.3756...		
1.5	1.252322716			

first divided differences

second divided differences

third divided differences

$$\therefore H_3(x) = 1.105170918 + 0.2210341f36\,(x-1) + 0.14653ff25\,(x-1)^2 + 0.0324952256\,(x-1)^2(x-1.5)$$

Next Question: What sort of error is generated by Hermite Interpolation?

Rolle's Theorem (case of MVT).

Suppose $f \in C[a,b]$ and f is differentiable on (a,b). If $f(a) = 0 = f(b)$ then a number c in (a,b) exists with $f'(c) = 0$.

Generalized Rolle's Theorem

Suppose $f \in C[a,b]$ is n times differentiable on (a,b). If $f(x)$ is zero at the $n+1$ distinct numbers $x_0, \ldots, x_n$ in $[a,b]$, then a number ξ in (a,b) exists with $f^{(n)}(\xi) = 0$.

Thm: If $f \in C^{2n+2}[a,b]$ then

$$f(x) = H(x) + \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi)$$

for some ξ with $a < \xi < b$.

Proof: The statement is trivially true if $x = x_k$.

Suppose $x \neq x_k$ & define

$$g(t) = f(t) - H_{2n+1}(t) - \frac{(t-x_0)^2 \cdots (t-x_n)^2}{(x-x_0)^2 \cdots (x-x_n)^2} \left[f(x) - H_{2n+1}(x) \right]$$

Note that $g(x_k) = 0$
$g(x) = 0$. $\Rightarrow$ g has $n+2$ distinct zeros in $[a,b]$

By Rolle's Theorem, g' has $n+1$ distinct zeros $\xi_0, \ldots, \xi_n$ which are between the numbers $x_0, \ldots, x_n, x$.

In addition $g'(x_k) = 0$ for $k = 0, \ldots, n$ so g' has $2n+2$ distinct zeros

$$\xi_0, \ldots, \xi_n, x_0, \ldots, x_n$$

Since g' is $2n+1$ times differentiable, the Generalized Rolle's Theorem implies that a number α in $[a,b]$ exists with $g^{(2n+2)}(\alpha) = 0$

However $g^{(2n+2)}(t) = f^{(2n+2)}(t) - \frac{d^{2n+2} H_{2n+1}(t)}{dt^{2n+2}}$

$$- \left\{ \frac{1}{(x-x_0)^2 \cdots (x-x_n)^2} \left[f(x) - H_{2n+1}(x) \right] \right\} \frac{d^{2n+2} (t-x_0)^2 \cdots (t-x_n)^2}{dt^{2n+2}}$$

Note: $H_{2n+1}(t)$ is a polynomial of degree $2n+1$, its derivative of this order is zero.

$$\frac{d^{2n+2} (t-x_0)^2 \cdots (t-x_n)^2}{dt^{2n+2}} = (2n+2)!$$

$$\Rightarrow 0 = g^{(2n+2)}(\alpha) = f^{(2n+2)}(\alpha) - (2n+2)! \frac{[f(x) - H_{2n+1}(x)]}{(x-x_0)^2 \cdots (x-x_n)^2}$$

So $$f(x) - H_{2n+1} = \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!} f^{(2n+2)}(\alpha)$$

for some α between $x_0, x_1, \ldots, x_n$ & x.

Algorithms similar to the Newton divided difference algorithm may also be applied to other <u>osculating polynomials</u>

polynomials of least degree that agrees with the function f and all its derivatives of order less than or equal to m_i at x_i.

See text for a reference.

Previously, we saw how to compute an approximation to a function over some finite interval using a single polynomial.

We increased the degree of the polynomial if we wanted more accuracy.

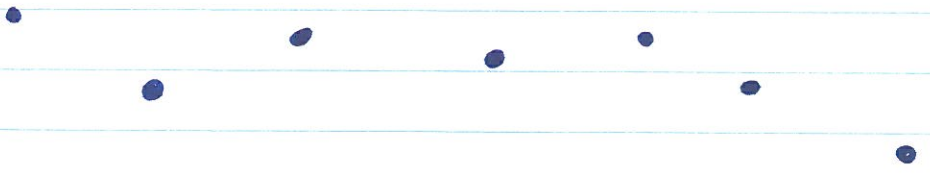
However, high degree polynomials are often "oscillatory" in nature and they have the property that a fluctuation over a small portion of the interval can induce large fluctuations over the entire range.

An alternative approach is to divide the interval up into subintervals and use a different (low order) polynomial in each subinterval.

These polynomials are patched together to give a piecewise polynomial approximation.

Possible Choices

Piecewise Linear Interpolation



Join a set of data points by a series of straight lines.

* A disadvantage of this approach is that the approximation is typically not smooth (more precisely, not differentiable) at these points which may not be satisfactory physically (e.g., we may need the slope) or visually (in computer graphics).

Piecewise Polynomials of Hermite Type.

If we know the value of a function and its derivative at each of the points, then we could use a Hermite cubic polynomial on each interval $[x_i, x_{i+1}]$

Often, however, the derivative of the function is not known at the data points.

Piecewise Quadratic Polynomials

Alternatively, we could join a set of points with quadratic polynomials.

This gives us 3 arbitrary constants.

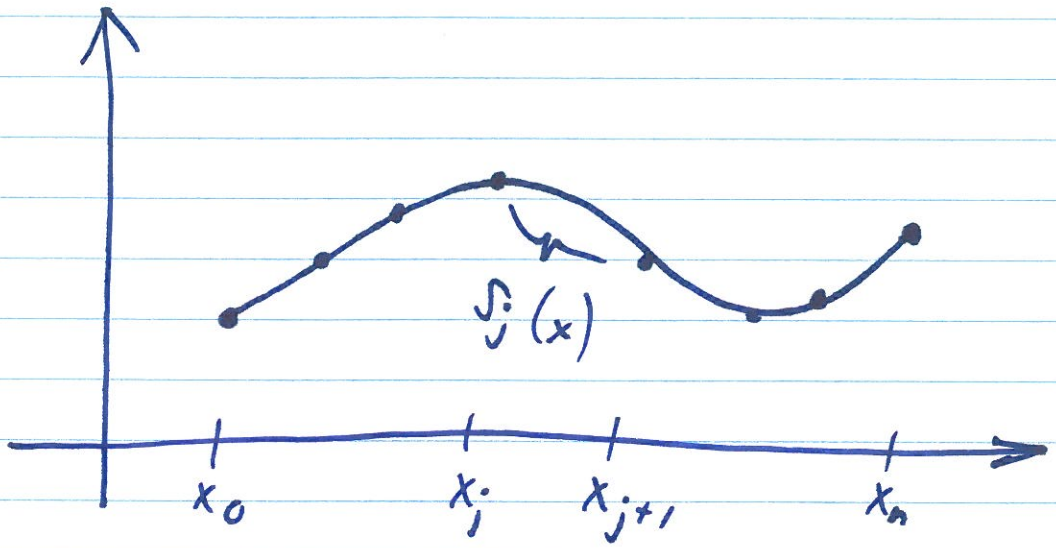
There will be 2 conditions to fit the curve through the endpoints of each interval, but there isn't enough flexibility to set conditions on the derivative at both endpoints x_0 and x_n.

Cubic Spline Interpolation

If we use cubic polynomials between each successive pair of nodes then there are four constants and it is possible to ensure that the interpolant

- agrees with the function at all the nodes
- is continuously differentiable
- has continuous second derivatives

Given a function f defined on $[a,b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a cubic spline interpolant S for f is a function that satisfies the following:



* $S(x) = S_j(x)$ on $[x_j, x_{j+1}]$
* $S_j(x_{j+1}) = f(x_{j+1}) = S_{j+1}(x_{j+1})$ — Spline must pass through the data pts
* $S_j'(x_{j+1}) = S_{j+1}'(x_{j+1})$ — first derivative is continuous
* $S_j''(x_{j+1}) = S_{j+1}''(x_{j+1})$ — second derivative is continuous.

We also need a boundary condition. Typically one of the following hold in the problems we will consider:

* $S'(x_0) = f'(x_0)$ & $S'(x_n) = f'(x_n)$

"Clamped Boundary Conditions"

* $S''(x_0) = 0 = S''(x_n)$

"Free or Natural Boundary Conditions"

Natural BC's tell us that S is not curved at the endpoints.

Of course, for the clamped case we need derivative information, either from the physics or from some assumption.

If there are $(n+1)$ points the number of intervals and the number of $S_i(x)$'s are n:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

Letting $h_i = x_{i+1} - x_i$

We can simplify by substituting into $S_i(x)$ $S_i'(x_{i+1})$ and $S_i''(x_i)$.

Eliminating the b_i and d_i gives a linear system of equations

$$h_{i-1}c_{i-1} + 2(h_{i-1} + h_i)c_i + h_i c_{i+1} = \frac{3}{h_i}(a_{i+1} - a_i) - \frac{3}{h_{i-1}}(a_i - a_{i-1})$$

$$1 \le i \le n-1$$

where the a_i are known

$$a_i = f(x_i) \qquad 0 \le i \le n-1$$

and we define $a_n \equiv f(x_n)$
$b_n \equiv f'(x_n)$
$c_n \equiv f''(x_n)/2$

The b_i and d_i are easily found:

$$b_i = \frac{1}{h_i}(a_{i+1} - a_i) - \frac{h_i}{3}(2c_i + c_{i+1})$$

$$d_i = \frac{c_{i+1} - c_i}{3h_i} \qquad 0 \le i \le n-1$$

We still need to impose BC's:

Consider Natural BC's: $S''(x_0) = 0$, $S''(x_n) = 0$

$\therefore c_n = 0$ (by defn)

$S_0''(x) = 2c_0 + 6d_0(x - x_0)$ $\therefore$ $S''(x_0) = 2c_0 + 6d_0(x_0 - x_0)$, $= c_0$

So we have a matrix equation for the $[c_i]$:

$$Ax = b$$

where

$$A = \begin{bmatrix} 1 & 0 & 0 & \cdots & \cdots & 0 \\ h_0 & 2(h_0+h_1) & h_1 & \ddots & & \vdots \\ 0 & h_1 & 2(h_1+h_2) & h_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ 0 & \cdots & & 0 & 0 & 1 \end{bmatrix}$$

$$x = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} \qquad b = \begin{bmatrix} 0 \\ \frac{3}{h_1}(a_2 - a_1) - \frac{3}{h_0}(a_1 - a_0) \\ \vdots \\ \frac{3}{h_{n-1}}(a_n - a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1} - a_{n-2}) \\ 0 \end{bmatrix}$$

Is there a solution for the c_i's?

Is the solution unique?

Yes and Yes!

The matrix is strictly diagonally dominant so, it is invertible and a unique solution for the c_i's exists.

First we need a definition and a theorem.

Defn: The $n \times n$ matrix A is strictly diagonally dominant if

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|$$

holds for each $i = 1, 2, \ldots, n$

Thm: A strictly diagonally dominant matrix is invertible.

Also note that the matrix A is TRIDIAGONAL: all the entries are zero except for a band which is 3 entries wide centred on the main diagonal.

Solutions to tridiagonal linear systems can be found very efficiently:

Only $O(n)$ operations are needed using methods we shall discuss later.

We also want to treat clamped BC's.

$$f'(a) = S'(a) = S_0'(x_0) = b_0 + 2c_0(x_0 - x_0) + 3d_0(x_0 - x_0)^2$$
$$= b_0$$

but $$b_{i-1} = \frac{a_i - a_{i-1}}{h_{i-1}} - \frac{h_{i-1}}{3}(2c_{i-1} + c_i)$$

$$\therefore \quad b_0 = \frac{a_1 - a_0}{h_0} - \frac{h_0}{3}(2c_0 + c_1)$$

$$\Longrightarrow \quad 2h_0 c_0 + h_0 c_1 = \frac{3}{h_0}(a_1 - a_0) - 3f'(a)$$

Similarly, $$h_{n-1} c_{n-1} + 2h_{n-1} c_n = 3f'(b) - \frac{3}{h_{n-1}}(a_n - a_{n-1})$$

Once again we obtain a strictly diagonally dominant system.

$\Longrightarrow$ a unique solution exists.

Furthermore, the system is tridiagonal so it can be solved efficiently for the coefficients of the spline.

We have seen that Lagrange interpolating polynomials can oscillate wildly, except where contained between data points that are in close proximity.

To avoid this serious problem we developed splines. Splines divide the interval up into subintervals and use a different (low order) polynomial in each subinterval.

See Figures 3.9 → 3.12 for an interesting illustration of this property.

We are very interested in knowing an error bound formula for the cubic spline.

For a cubic spline with clamped boundary conditions we have the following result:

Thm: Let $f \in C^4[a,b]$ with

$$\max_{a \le x \le b} |f^{(4)}(x)| = M$$

If S is the unique clamped cubic spline interpolant to f with respect to the nodes

$$a = x_0 < x_1 < \cdots < x_n = b \quad \text{then}$$

$$\max_{a \le x \le b} |f(x) - S(x)| \le \frac{5M}{384} \max_{0 \le j \le n-1} (x_{j+1} - x_j)^4$$

A fourth order error bound also exists for the case of natural boundary conditions.

Typically clamped boundary conditions will be more accurate unless $f''(x_0) = 0 = f''(x_n)$

Parametric Curves

The interpolating polynomials and splines that we have previously discussed can only be used to interpolate <u>functions</u>.

The technique can be extended to represent general curves in space.

e.g.

Suppose we wish to determine a polynomial or a piecewise polynomial to connect the points

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

in the order given.

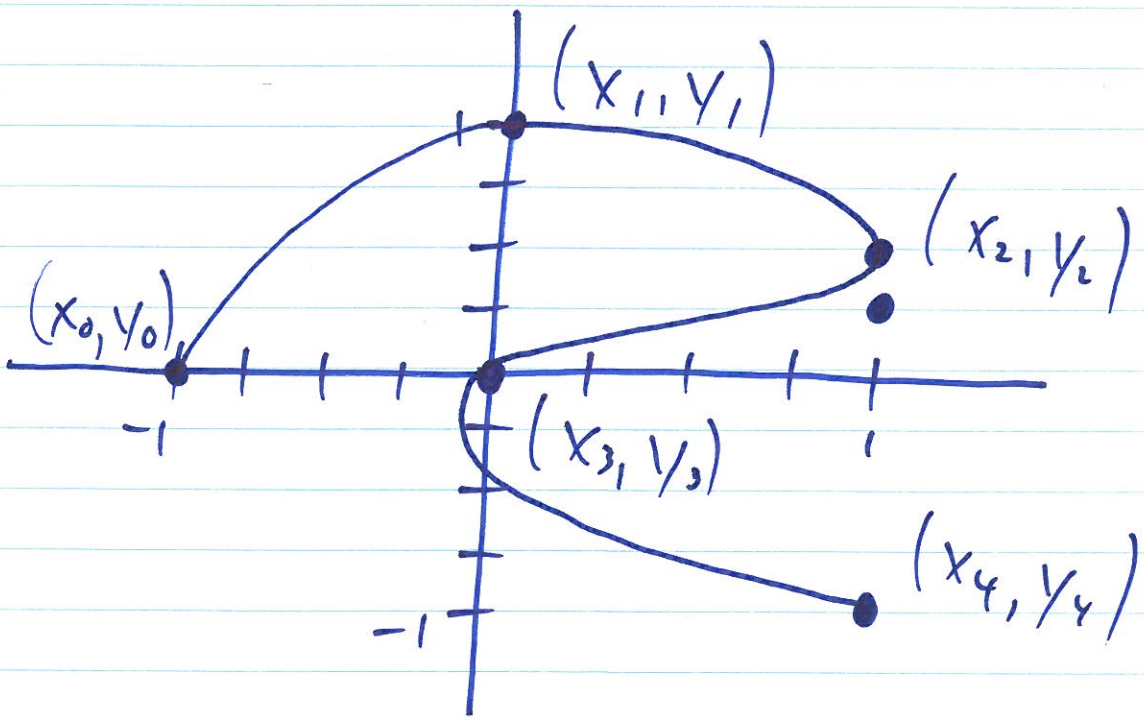
We can define a parameter t on the interval $[t_0, t_n]$ with

$$t_0 < t_1 < \cdots < t_n$$

and construct approximation functions for x and y separately:

$$x_i = x(t_i), \quad y_i = y(t_i)$$

Ex



There is flexibility in choosing the points t_i.

Suppose we choose them to be evenly spaced over $[0, 1]$.

Then

i	0	1	2	3	4
t_i	0	.25	.5	.75	1
x_i	-1	0	1	0	1
y_i	0	1	.5	0	-1

Now the Lagrange Interpolating polynomials for x & y are

$$x(t) = \left(\left(\left(64t - \frac{352}{3}\right)t + 60\right)t - \frac{14}{3}\right)t - 1$$

& $$y(t) = \left(\left(\left(-\frac{64}{3}t + 48\right)t - \frac{116}{3}\right)t + 11\right)t.$$

Alternatively we could use a spline-based interpolation for x and y.

Both of these approaches have the disadvantage that moving a single data point effects the entire curve.

We want the geometric property that changing one of the points on the curve only changes one portion of the curve — a local effect.

One possibility:

Use a piecewise cubic Hermite polynomial.

In other words, we specify each portion of the curve by specifying its end points and the derivatives at these endpoints. (Thus changing a data point will only change the 2 portions adjacent to the point, which means that smooth curves can be easily and quickly modified)

Is this a unique representation?

No. There are 6 conditions
- the curve must pass through both end points
- the derivatives dy/dx must match up at the endpoints with the specified values.

But each cubic polynomial has 4 parameters for a total of 8.

Suppose that the endpoints are at $t=0$ and $t=1$, then we only need to satisfy the conditions on the quotients

$$\frac{dy}{dx}(t=0) = \frac{y'(0)}{x'(0)}, \quad \frac{dy}{dx}(t=1) = \frac{y'(1)}{x'(1)}$$

The actual values of $x'(0)$ and $y'(0)$ can be scaled by a common factor and still satisfy these conditions.

The larger the scaling factor, the closer the curve comes to satisfying the tangent line near $(x(0), y(0))$.

A similar situation holds for $(x(1), y(1))$

To simplify the process of specifying the slopes and to obtain a unique curve, commercial software commonly specifies a second point called a guidepoint which lies on the desired tangent line.

Suppose the endpoints are

$$(x_0, y_0) \quad \text{and} \quad (x_1, y_1)$$

and let the guidepoints be

$$(x_0 + \alpha_0, y_0 + \beta_0) \quad \& \quad (x_1 - \alpha_1, y_1 - \beta_1)$$

We will insist that x satisfies

$$x(0) = x_0 \qquad x'(0) = \alpha_0$$
$$x(1) = x_1 \qquad x'(1) = \alpha_1$$

and that y satisfies

$$y(0) = y_0 \qquad y'(0) = \beta_0$$
$$y(1) = y_1 \qquad y'(1) = \beta_1$$

Now there is a unique solution for x and y:

$$x(t) = [2(x_0 - x_1) + (\alpha_0 + \alpha_1)]\,t^3 + [3(x_1 - x_0) - (\alpha_1 + 2\alpha_0)]\,t^2 + \alpha_0 t + x_0$$

$$y(t) = [2(y_0 - y_1) + (\beta_0 + \beta_1)]\,t^3 + [3(y_1 - y_0) - (\beta_1 + 2\beta_0)]\,t^2 + \beta_0 t + y_0$$

Popular graphics programs will typically use a slightly modified form for x and y. They use Bézier polynomials which scale the derivatives x' & y' by a factor of 3 at the endpoints:

$$x(t) = [2(x_0 - x_1) + 3(\alpha_0 + \alpha_1)]\,t^3 + [3(x_1 - x_0) - 3(\alpha_1 + 2\alpha_0)]\,t^2 + 3\alpha_0 t + x_0$$

$$y(t) = [2(y_0 - y_1) + 3(\beta_0 + \beta_1)]\,t^3 + [3(y_1 - y_0) - 3(\beta_1 + 2\beta_0)]\,t^2 + 3\beta_0 t + y_0$$

Bezier curves are named after the French engineer P. Bezier of the Renault Automobile Company.

He developed them in the early 1960's to fill a need for curves whose shape can be readily controlled by changing a few parameters.

Bezier's application was to construct pleasing surfaces for car bodies.

Bezier also developed another curve representation (also called Bezier curves) which is commonly used. These have the property that they do not normally pass through the data points.

See e.g. Applied Numerical Analysis
Gerald / Wheatley