Numerical Interpolation

Overview

- Motivation
- Lagrange Polynomials
- Newton Interpolation (Divided Differences) Method
- Interpolation Using Splines
  - Linear, Quadratic, Cubic
Numerical Interpolation: Definitions

Given a set of measurements (or function values) at a set of points, to estimate the values at unknown sample points.

- Many engineering problems, instruments (scanners, cameras, etc) measure data at a discrete set of points (digitization).
- Subsequent analysis requires evaluation of the function at other points (within the function domain).
- Interpolation methods permit approximate functions to be fit over the known points.
- Interpolants can be linear, quadratic, cubic and multidimensional.
Motivation

Example 8-A Chemical Reaction Product

Consider the observed concentration of the product of a simple chemical reaction as a function of time:

<table>
<thead>
<tr>
<th>Time</th>
<th>0.0</th>
<th>0.5</th>
<th>1.0</th>
<th>1.5</th>
<th>2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>0.0</td>
<td>0.19</td>
<td>0.26</td>
<td>0.29</td>
<td>0.31</td>
</tr>
</tbody>
</table>

In this chapter, we discuss several techniques for approximating the concentration of the product at other times. The data are illustrated in Fig. 8.1.

Besides interpolating the data shown above, we also consider the effect of utilizing a more extensive set of data.
Motivation

Example 8-B: Spline Interpolation for Parametric Functions

Spline interpolation is used in computer graphics to represent smooth curves (in parametric form). Several points are chosen along the curve and are indexed in terms of a parameter, $t$. Interpolation is performed on the $x$ and $y$ coordinates separately (each as a function of $t$). The resulting parametric plot, $(x(t), y(t))$ gives the interpolated curve. For example, we take the points shown in Fig. 8.2 as the data, which gives

$$
\begin{align*}
    t &= [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10 \ 11 \ 12] \\
    x &= [0 \ 1 \ 2 \ 2 \ 1 \ 1 \ 2 \ 3 \ 3 \ 3 \ 4 \ 5] \\
    y &= [0 \ 0 \ 0 \ 1 \ 1 \ 2 \ 2 \ 2 \ 1 \ 0 \ 0 \ 0]
\end{align*}
$$

Data for several other curves are given in the exercises.
Method of Undetermined Coefficients

- Use an $n$th order polynomial as the interpolant:

$$f(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_n x^n$$

- Use the known values $(x, f(x)$ pairs) to determine the coefficients of the polynomial.
Method of Undetermined Coefficients: Linear Interpolant

\[
\frac{f(x) - f(x_i)}{x - x_i} = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}
\]

\[
f(x) = f(x_i) + \frac{x - x_i}{x_{i+1} - x_i}[f(x_{i+1}) - f(x_i)]
\]

- \(x\) is chosen to lie between \(x_i\) and \(x_{i+1}\), the nearest neighbors.
- Linear interpolant uses the product of the derivative (dy/dx) and the deviation \((x - x_i)\) as a correction term to the base function value \(f(x_i)\).
Lagrange Interpolation

- Earlier methods restrict the discrete data to be within a constant interval.
- For many applications, data collection cannot be controlled, and we end up with data at unequal intervals, for instance, \( x_i, f(x_i), i = 1, \ldots N \)

Given a set of values \( x_i, f(x_i), i = 1, \ldots N \), where \( f(x_i) \) is the function value at \( x_i \), Lagrange interpolating equation for any point \( x_0 \) is given by

\[
f(x) = \sum_{i=1}^{n} w_i(x) f(x_i)
\]

where \( w_i \)'s are the weights, and are a function of \( x_0 \), and given by

\[
w_i(x) = \frac{\prod_{j \neq i, j=1}^{n} (x - x_j)}{\prod_{j \neq i, j=1}^{n} (x_i - x_j)}
\]
Lagrange Interpolation (contd)

- Lagrange form for straightline through 2 points:
  \[ p(x) = \frac{(x - x_2)}{(x_1 - x_2)}y_1 + \frac{(x - x_1)}{(x_2 - x_1)}y_2 \]

- Lagrange form of a parabola through 3 points:
  \[ p(x) = w_1(x)y_1 + w_2(x)y_2 + w_3(x)y_3 \]
  \[ = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}y_1 + \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}y_2 + \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}y_3 \]

- Notice that the equations pass through all the points \((x_k, y_k)\)
Lagrange Interpolation: Example

\[
\begin{align*}
\mathbf{x} &= [-2, 0, -1, 1, 2] \\
\mathbf{y} &= [4, 2, -1, 1, 8]
\end{align*}
\]
Lagrange Interpolation: Algorithm

(Left) Compute Coefficients (Right) Evaluate Polynomial

\[ P(x) = c_1(x-x_2)\ldots(x-x_n) + c_2(x-x_1)(x-x_3)\ldots(x-x_n) + \ldots + c_n(x-x_1)\ldots(x-x_{n-1}) \]

**Input**
- \( x \) data vector for independent variable
- \( y \) data vector for dependent variable
- \( n \) number of components in vectors \( x \) and \( y \)

For \( k = 1 \) to \( n \)
- d(\( k \)) = 1
  For \( j = 1 \) to \( n \)
    If \( (j \neq k) \)
      d(\( k \)) = d(\( k \))(x(\( k \)) - x(\( j \)))
    End
  c(\( k \)) = y(\( k \))/d(\( k \))
End

Return c vector of coefficients

**Input**
- \( x \) data vector
- \( c \) coefficient vector
- \( n \) number of components in vector \( x \)
- \( t \) vector of points for interpolation \( t_1, \ldots, t_m \)
- \( m \) number of components in vector \( t \)

For \( i = 1 \) to \( m \)
- \( P(i) = 0 \)
  For \( j = 1 \) to \( n \)
    d(\( j \)) = 1
    For \( k = 1 \) to \( n \)
      If \( (j \neq k) \)
        d(\( j \)) = d(\( j \))(t(\( i \)) - x(\( k \)))
      End
    End
  End
  \( P(i) = P(i) + c(j) \times d(\( j \)) \)
End

Return P vector of interpolated values

Numerical Interpolation
Lagrange Interpolation: Difficulties

⇒ Adding additional points to the function - does not improve the curve shape.

⇒ Also, must redo the entire computation from scratch.
Lagrange Interpolation: Analysis

- Lagrange formulation is of the form (for a straight line)
  \[ L(x) = L_1y_1 + L_2y_2 \]

- Consider \( x_1 = 0, x_2 = 1 \); then the basis lines are
  \[ L_1 : -x + 1, L_2 : y = x \]
Newton Interpolation Polynomials

These methods can successively generate the polynomials to interpolate discrete data.

Let $P_n(x)$ be the $n$th order Lagrange polynomial representing $f(x)$ at $x_1, x_2, \ldots x_n$,

\[ P_n(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + \cdots + a_n(x - x_1)(x - x_1) \cdots (x - x_{n-1}) \]

By evaluating the function at $x_1, x_2, \ldots$ we can solve for $a_1, a_2, \ldots$, etc.

\[ a_1 = P_n(x_1) = f(x_1) = y_1 \]
Newton Interpolation Polynomials (contd.)

To obtain $a_1$, 

$$f(x_1) + a_2(x_2 - x_1) = P_n(x_2) = f(x_2) = y_2$$

$$a_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

Similarly,

$$a_3 = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}$$

$\Rightarrow$ Calculations can be made in a systematic manner using divided differences of the function values.
Divided Differences

Zeroth Divided Difference

\[ f[x_i] = f(x_i) \]

First Divided Difference

\[ f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i} \]

Second Divided Difference

\[ f[x_i, x_{i+1}, x_{i+2}] = \frac{f[x_{i+1}, x_{i+2}] - f[x_i, x_{i+1}]}{x_{i+2} - x_i} \]
Divided Differences (contd.)

**$k$th Divided Difference**

$$f[x_i, x_{i+1}, \ldots, x_{i+k}] = \frac{f[x_{i+1}, \ldots, x_{i+k}] - f[x_i, \ldots, x_{i+k-1}]}{x_{i+k} - x_i}$$

$P_n(x)$ can thus be expressed as

$$P_n(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, x_1, \ldots, x_k](x - x_0)(x - x_1)\ldots(x - x_{k-1})$$
Newton Interpolation: Example

Example 8.5 Newton Interpolation Parabola

Consider again the data from Example 8.1. We can find a quadratic polynomial passing through the points \((x_1, y_1) = (2, 4)\), \((x_2, y_2) = (0, 2)\), and \((x_3, y_3) = (8, 2)\). The Newton form of the equation is

\[ p(x) = a_0 + a_2(x + 2) + a_1(x + 2)(x - 0), \]

where the coefficients are

\[ a_0 = y_1 = 4, \]

\[ a_2 = \frac{y_2 - y_1}{x_2 - x_1} = \frac{2 - 4}{0 - (-2)} = -1, \]

and

\[ a_1 = \frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1} = \frac{8 - 2}{8 - 0} - \frac{2 - 4}{2 - (-2)} = 1. \]

Thus,

\[ p(x) = 4 - (x + 2) + x(x + 2) = x^2 + x + 2, \]

as before.

The calculations can be performed in a systematic manner, using a “divided-difference table”:

<table>
<thead>
<tr>
<th>( x )</th>
<th>( y )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>4</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
</tbody>
</table>

The coefficients of the Newton polynomial are the top entries in this table.

The graph of the interpolation polynomial is shown in Fig. 8.3.
Newton Interpolation: Additional Points

- Additional points added does not require the calculations to be repeated from scratch.

*Example 8.6 Additional Data Points*

If we extend the previous example, adding the points \((x_4, y_4) = (-1, -1)\) and \((x_5, y_5) = (1, 1)\), the polynomial has the form

\[
p(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + a_4(x - x_1)(x - x_2)(x - x_3) + a_5(x - x_1)(x - x_2)(x - x_3)(x - x_4).
\]

The divided-difference table becomes (with new entries shown in bold)

<table>
<thead>
<tr>
<th>(x_i)</th>
<th>(y_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>-2</td>
<td>4</td>
</tr>
<tr>
<td>(\frac{(2 - 4)}{(0 + 2)} = -1)</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>(\frac{(0 - 1)}{(-1 + 2)} = -1)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>(\frac{(2 + 1)}{(1 + 2)} = 1)</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>(\frac{(-1 - 8)}{(-1 - 2)} = 3)</td>
<td></td>
</tr>
<tr>
<td>(\frac{(2 - 0)}{(-1 - 2)} = 2)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>(\frac{(1 + 1)}{(1 + 1)} = 1)</td>
<td></td>
</tr>
</tbody>
</table>
Newton Interpolation: Algorithm

\[ \text{Input} \]
\[
\begin{align*}
  x & \quad \text{data vector for independent variable} \\
  y & \quad \text{data vector for dependent variable} \\
  n & \quad \text{number of components in vectors } x \text{ and } y
\end{align*}
\]

\[ \text{Compute} \]
\[
\begin{align*}
  a(1) & = y(1) \\
  \text{For } k = 1 \text{ to } n - 1 & \quad \text{form first divided difference} \\
  d(k, 1) & = (y(k + 1) - y(k))/(x(k + 1) - x(k)) \\
  \text{End} \\
  \text{For } j = 2 \text{ to } n - 1 & \quad \text{form } j^{th} \text{ divided difference} \\
  \quad \text{For } k = 1 \text{ to } n - j & \\
  d(k, j) & = (d(k + 1, j - 1) - d(k, j - 1))/(x(k + j) - x(k)) \\
  \quad \text{End} \\
  \text{End} \\
  \text{For } j = 2 \text{ to } n & \\
  a(j) & = d(1, j - 1) \\
  \text{End} \\
  \text{Return} \\
  a & \quad \text{vector of coefficients}
\end{align*}
\]

\[ \text{Input} \]
\[
\begin{align*}
  x & \quad \text{data vector} \\
  a & \quad \text{coefficient vector} \\
  n & \quad \text{number of components in vector } x \\
  t & \quad \text{vector of points for interpolation } t_1, \ldots, t_m \\
  m & \quad \text{number of components in vector } t
\end{align*}
\]

\[ \text{Compute} \]
\[
\begin{align*}
  d(1) & = 1 \\
  N(k) & = a(1) \\
  \text{For } j = 2 \text{ to } n & \\
  d(j) & = (t(k) - x(j - 1))d(j - 1) \\
  N(k) & = N(k) + a(j) \cdot d(j) \\
  \text{End} \\
  \text{End} \\
  \text{Return} \\
  N & \quad \text{vector of interpolated values}
\end{align*}
\]
Newton Interpolation: Analysis

- Especially convenient when x-spacing is constant
- Can add new points to compute higher order polynomials without restarting the computation, i.e. incrementally.

- Example: for points \((0, 1), (1, 3), (2, 6)\), basis functions are 
  \[ P_1(x) = 1, \quad P_2(x) = x, \quad P_3(x) = x(x - 1) \]

- Polynomial is 
  \[ N(x) = 1 + 2(x - 0)^2 + 0.5(x - 0)(x - 1) \]
Polynomial Interpolation: Difficulties

Example 8.8 Humped and Flat Data

The data

\[ x = [-2 \ -1.5 \ -1 \ -0.5 \ 0 \ 0.5 \ 1 \ 1.5 \ 2] \]
\[ y = [ 0 \ 0 \ 0 \ 0.87 \ 1 \ 0.87 \ 0 \ 0 \ 0] \]

illustrate the difficulty with using higher order polynomials to interpolate a moderately large number of points, especially when the curve changes shape significantly over the interval (being flat in some regions and not in others). (See Fig. 8.11.)

![Graph of humped and flat data for polynomial interpolation.](image)
Spline Interpolation

Motivation

- Using higher order polynomials can sometimes be inaccurate, especially when a function $f(x)$ has abrupt local changes.
- In these instances, it can be better to use lower order polynomials, termed **splines**.
- Splines are typically **linear, quadratic or cubic**.
- Examples include Hermite, Bezier, B-Spline, Catmull-Rom.
Linear Splines

⇒ Connects adjacent points with linear segments

\[
f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}, \quad x_1 \leq x \leq x_2
\]

\[
\vdots \quad \vdots
\]

\[
f_{n-1}(x) = f(x_{n-1}) + \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}, \quad x_{n-1} \leq x \leq x_n
\]

and satisfies the following condition

\[
f_i(x_{i+1}) = f_{i+1}(x_i), \quad \text{for } i = 1, 2, \ldots
\]

Notes

⇒ Linear splines interpolate (or intersect) all the input discrete points.

⇒ They are thus, discontinuous at the data points.
Quadratic Splines

A quadratic function is defined between each pair of adjacent points,

\[ f_i(x) = a_i x^2 + b_i x + c_i, \text{ for } i = 1, 2, \ldots, n - 1 \]

requiring \(3(n - 1)\) equations to solve for the \(3(n - 1)\) unknowns.
Quadratic Splines

Necessary Conditions

⇒ Splines pass through the data points:

\[ f_i(x_i) = f(x_i) = a_i x_i^2 + b_i x_i + c_i, \text{ for } i = 1, 2, \ldots, n - 1 \]
\[ f_i(x_{i+1}) = f(x_{i+1}) = a_i x_{i+1}^2 + b_i x_{i+1} + c_i, \text{ for } i = 1, 2, \ldots, n - 1 \]

⇒ Splines are continuous at the interior points (equal first derivatives)

\[ 2a_i x_{i+1} + b_i = 2a_{i+1} x_{i+1} + b_{i+1}, \text{ for } i = 1, 2, \ldots, n - 2 \]

⇒ Arbitrary: set second derivative for spline between first 2 data points to zero,

\[ a_1 = 0 \]

resulting in the \(3(n - 1)\) conditions to solve for the coefficients.
Quadratic Splines: Example

- Dataset used

**TABLE 6-11** Linear Spline for Well-water Elevation

<table>
<thead>
<tr>
<th>$x_i$ (ft)</th>
<th>$f(x_i)$ (ft)</th>
<th>Linear Spline $f(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>$14.6 + \frac{10.7 - 14.6}{42 - 15} (x - 15) = 14.6 - 0.1444(x - 15)$</td>
</tr>
<tr>
<td>2</td>
<td>42</td>
<td>$10.7 - 0.0686046 (x - 42)$</td>
</tr>
<tr>
<td>3</td>
<td>128</td>
<td>$4.8 - 0.0164021 (x - 128)$</td>
</tr>
<tr>
<td>4</td>
<td>317</td>
<td>$1.7 - 0.0120689 (x - 317)$</td>
</tr>
<tr>
<td>5</td>
<td>433</td>
<td></td>
</tr>
</tbody>
</table>
Quadratic Splines: Example (contd)

Example 6-10 Quadratic Spline for Well-water Elevation

The data set that was previously used in Examples 6-8 and 6-9 can be utilized to illustrate the quadratic spline. Using Eqs. 6-71a, the following equations can be developed:

\[ 225a_1 + 15b_1 + c_1 = 14.6 \]  \hspace{0.5cm} (6-74a)
\[ 1764a_2 + 42b_2 + c_2 = 10.7 \]  \hspace{0.5cm} (6-74b)
\[ 15384a_3 + 128b_3 + c_3 = 4.8 \]  \hspace{0.5cm} (6-74c)
\[ 100489a_4 + 317b_4 + c_4 = 1.7 \]  \hspace{0.5cm} (6-74d)

Equations 6-71b results in the following conditions:

\[ 1764a_1 + 42b_1 + c_1 = 10.7 \]  \hspace{0.5cm} (6-75a)
\[ 15384a_2 + 128b_2 + c_2 = 4.8 \]  \hspace{0.5cm} (6-75b)
\[ 100489a_3 + 317b_3 + c_3 = 1.7 \]  \hspace{0.5cm} (6-75c)
\[ 187489a_4 + 433b_4 + c_4 = 0.3 \]  \hspace{0.5cm} (6-75d)

Equations 6-72 results in the following conditions:

\[ 2a_1(42) + b_1 = 2a_2(42) + b_2 \]  \hspace{0.5cm} (6-76a)
\[ 2a_2(128) + b_2 = 2a_3(128) + b_3 \]  \hspace{0.5cm} (6-76b)
\[ 2a_3(317) + b_3 = 2a_4(317) + b_4 \]  \hspace{0.5cm} (6-76c)

The last condition comes from Eq. 6-73 as

\[ a_1 = 0 \]  \hspace{0.5cm} (6-77)

Since \( a_1 = 0 \), the resulting system of 11 equations can be summarized in a matrix format as
Quadratic Splines: Example (contd)

\[
\begin{pmatrix}
15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 1,764 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 16,384 & 128 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 100,489 & 317 & 1 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
42 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 16,384 & 128 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 100,489 & 317 & 1 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 187,489 & 433 & 1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
1 & 0 & -84 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 256 & 1 & 0 & -256 & -1 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 634 & 1 & 0 & -634 & -1 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
14.6
\end{pmatrix}
\begin{pmatrix}
b_1
\end{pmatrix}
\begin{pmatrix}
c_1
\end{pmatrix}
\begin{pmatrix}
a_2
\end{pmatrix}
\begin{pmatrix}
b_2
\end{pmatrix}
\begin{pmatrix}
c_2
\end{pmatrix}
\begin{pmatrix}
a_3
\end{pmatrix}
\begin{pmatrix}
b_3
\end{pmatrix}
\begin{pmatrix}
c_3
\end{pmatrix}
\begin{pmatrix}
a_4
\end{pmatrix}
\begin{pmatrix}
b_4
\end{pmatrix}
\begin{pmatrix}
c_4
\end{pmatrix}
\begin{pmatrix}
a_5
\end{pmatrix}
\begin{pmatrix}
b_5
\end{pmatrix}
\begin{pmatrix}
c_5
\end{pmatrix}
\begin{pmatrix}
a_6
\end{pmatrix}
\begin{pmatrix}
b_6
\end{pmatrix}
\begin{pmatrix}
c_6
\end{pmatrix}
The solution of this system of equations can be determined numerically to be

\[
\begin{bmatrix}
\begin{array}{c}
-0.144444 \\
16.766667 \\
8.818583 \times 10^{-4} \\
-0.2185206 \\
18.32227 \\
-1.2506 \times 10^{-4} \\
3.925179 \times 10^{-2} \\
1.824836 \\
2.411243 \times 10^{-4} \\
-0.1929122 \\
38.62283
\end{array}
\end{bmatrix}
\]

The resulting quadratic splines are shown in Fig. 6-3. For example, the first spline is

\[
f(x) = (0)x^2 + (-0.144444)x + (16.766667),
\]

which is valid for \(15 \leq x \leq 42\). The first spline is shown in the figure over its range of applicability (i.e., \(15 \leq x \leq 42\)). Other splines can be similarly expressed. Plotting each spline over its respective range of applicability produces Fig. 6-3. The figure shows that the quadratic spline results in predicting the data points with the appearance of continuity in both magnitude and slope at the measured data points.
Cubic Splines

A cubic function is defined between each pair of adjacent points,

\[ f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i, \text{ for } i = 1, 2, \ldots, n - 1 \]

requiring \(4(n - 1)\) equations to solve for the \(3(n - 1)\) unknowns.
Cubic Splines

Necessary Conditions

⇒ Splines pass through the data points:

\[ f_i(x_i) = f(x_i) = a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i, \text{ for } i = 1, 2, \ldots, n - 1 \]
\[ f_i(x_{i+1}) = f(x_{i+1}) = a_i x_{i+1}^3 + b_i x_{i+1}^2 + c_i x_{i+1} + d_i, \text{ for } i = 1, 2, \ldots, n - 1 \]

⇒ Splines are continuous at the interior points (equal first derivatives)

\[ 3a_i x_{i+1}^2 + b_i x_{i+1} + c_i = 3a_{i+1} x_{i+1}^2 + b_{i+1} x_{i+1} + c_{i+1}, \text{ for } i = 1, 2, \ldots, n - 2 \]

⇒ Set second derivative at first and last end point to zero,

\[ 6a_1 x_1 + b_1 = 0 \]
\[ 6a_{n-1} x_n + b_{n-1} = 0 \]

resulting in the \(4(n-1)\) conditions to solve for the coefficients.
Application: Cubic Splines in Geometric Modeling

Implicit form

\[ F(x, y, z) = 0 \]

Parametric form

\[ x(t) = x(t) \]
\[ y(t) = y(t) \]
\[ z(t) = z(t) \]

Example: Lines

\[ y = mx + c \quad \text{(Implicit)} \]
\[ x(t) = x_0 + t(x_1 - x_0) \quad \text{(Parametric)} \]
\[ y(t) = y_0 + t(y_1 - y_0) \quad \text{(Parametric)} \]
Cubic Polynomials

\[ x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \]
\[ y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \]
\[ z(u) = a_z u^3 + b_z u^2 + c_z u + d_z \]

\[ 0 \leq u \leq 1. \]

which can be written in matrix form as

\[ Q(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} = U \cdot C \]

\[ Q(u) \] is the curve evaluated at \( u \).
Cubic Polynomials (contd.)

\[ Q'(u) = \begin{bmatrix} \frac{d}{dt}x(t) & \frac{d}{dt}y(t) & \frac{d}{dt}z(t) \end{bmatrix} \]

\[ = \frac{d}{dt}[U \cdot C] \]

is the tangent vector (velocity) of the curve at \( u \) and \( C \) is the coefficient matrix.

\[ C = M \cdot G \]

where \( M \) is known as a basis matrix and \( G \), the geometry vector, that describes the constraints of the curve formulation.
Cubic Polynomials (contd.)

Hence

\[ Q(u) = U \cdot M \cdot G \]

or,

\[ Q(u) = [x(u)\ y(u)\ z(u)] \]

\[ = [u^3\ u^2\ u\ 1] \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & m_{23} \\ m_{30} & m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} G_0 \\ G_1 \\ G_2 \\ G_3 \end{bmatrix} \]
Example

P1
P2
P3
P4
Notes:

- Basis (or blending) functions shape the control points into a curve segment.
- Why use control points? Can be interactively ‘pulled’ to alter the curve shape - a key to interactive design and modeling.
- Blending functions can either interpolate or approximate the control points.
- B-Spline and Bezier basis functions are the most popular.
- Cubics are most popular from a computational point of view.
- Easily extended to higher degree curves.
- B-Spline/Bezier surfaces in 3D - straightforward extension of B-Spline/Bezier curves.
Parametric Representation of Cubic Curves

\[ Q(u) = U \cdot M \cdot G \]

**Hermite Formulation**

- **Constraints:** (\( G = P_0, P_3, R_0, R_3 \)), \( P_0, P_3 \) are end points \( (u = 0 \text{ and } u = 1) \), \( R_0, R_3 \) are tangent vectors at \( P_0 \) and \( P_3 \).
- **Parameterization Interval:** 0.0....1.0
- **Basis Functions:** \( B_i = \{2u^3 - 3u^2 + 1, -2u^3 + 3u^2, u^3 - 2u^2 + u, u^3 - u^2\} \)

\[
Q(u) = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_3 \\ R_0 \\ R_3 \end{bmatrix} = U \cdot M_H \cdot G_H
\]
Derivation of $M_H$

$Q(u) = au^3 + bu^2 + cu + d$

$Q(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

$Q'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$

$\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$
Derivation of $\mathbf{M}_H$(contd.)

\[
\begin{bmatrix}
Q(0) \\
Q(1) \\
Q'(0) \\
Q'(1)
\end{bmatrix} =
\begin{bmatrix}
0 & 0 & 0 & 1 \\
1 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 \\
3 & 2 & 1 & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
a \\
b \\
c \\
d
\end{bmatrix}
\]

\[
\begin{bmatrix}
a \\
b \\
c \\
d
\end{bmatrix} =
\begin{bmatrix}
2 & -2 & 1 & 1 \\
-3 & 3 & -2 & -1 \\
0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0
\end{bmatrix}
\cdot
\begin{bmatrix}
Q(0) \\
Q(1) \\
Q'(0) \\
Q'(1)
\end{bmatrix}
\]

\[
= \mathbf{M}_H \cdot
\begin{bmatrix}
Q(0) \\
Q(1) \\
Q'(0) \\
Q'(1)
\end{bmatrix}
\]
Beziers Curve Formulation

- **Constraints:** \( G = (P_0, P_1, P_2, P_3) \) which is the control point mesh, with 
  \( R_0 = Q'(0) = 3(P_1 - P_0), R_3 = Q'(1) = 3(P_3 - P_2) \)

- **Parameterization Interval:** 0.0....1.0

- **Basis Functions:** \( B_i = \{(1 - u)^3, 3u(1 - u)^2, 3u^2(1 - u), u^3\} \)

\[
Q(u) = [x(u) \ y(u) \ z(u)] = [u^3 \ u^2 \ u \ 1] \\
= \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}
\]

Beziers form is advantageous since the curve is controlled completely by points, in contrast to tangent vectors and points.