3 Interpolation

Given: \((x_k, y_k), k = 0, 1, 2, \ldots, N\)

Find: \(y(x^*)\) for some \(x^* \in [\text{Min}(x_k), \text{Max}(x_k)]\)

- simplest method: piecewise linear interpolation

\[ y(x^*) = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_N)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_N)} \]

\[ P_N(x) = \sum_{k=0}^{N} y_k \cdot L_{N,k}(x) \]

Definition: Lagrange interpolating polynomial

\[ L_{N,k}(x) = \prod_{i \neq k}^{N} \frac{x-x_i}{x_k-x_i} \]

\[ \delta_{jk} = \begin{cases} 1 & \text{if } j = k \\ 0 & \text{if } j \neq k \end{cases} \]

\[ P_N(x) = \sum_{k=0}^{N} y_k \cdot \delta_{jk} = y_j, \text{ for all } j \]

\[ \text{Lagrange error} = \frac{1}{2!} |f''(\xi) (x-x_0)(x-x_N)| \leq \frac{\max(f''(x))}{2} \cdot \frac{(x-x_0)(x-x_N)}{2} = \frac{\text{max}(f'') (x-x_0)(x-x_N)}{2} \sim O(\Delta x^2) \]

§ Polynomial Approximation

Theorem (Weierstrass Approximation Theorem --- existence)

If \(f \in C[a,b]\) and given \(\varepsilon > 0\), then there exists a polynomial \(P(x)\) defined on \([a,b]\) with the property that \(|f(x) - P(x)| < \varepsilon\) for all \(x \in [a,b]\).

Given: \((x_k, y_k), k = 0, 1, 2, \ldots, N\)

Solution: find the unique polynomial of degree \(N\), \(P_N(x)\), passing through all these \((N+1)\) points.

\[ P_N(x) = a_0x^N + a_1x^{N-1} + \cdots + a_N \]

\[ P_N(x) = \sum_{k=0}^{N} y_k \cdot L_{N,k}(x) \]

Theorem If \(x_0, x_1, \ldots, x_N\) are distinct numbers in \([a,b]\) and \(f \in C^{N+1}[a,b]\), then for each \(x \in [a,b]\), \(\exists \xi \in [a,b]\) such that

\[ f(x) = P_N(x) + \frac{f^{(N+1)}(\xi)}{(N+1)!} (x-x_0)(x-x_1)\cdots(x-x_N) \]

\[ \text{piecewise linear interpolation} \]

\(\sim\) use \(P_i(x)\) between two adjacent points, i.e. for \([x_i, x_{i+1}]\)

\[ \text{Lagrange error} = \frac{1}{2} |f''(\xi) (x-x_0)(x-x_N)| \leq \frac{\max(f''(x))}{2} \cdot \frac{(x-x_0)(x-x_N)}{2} = \frac{\text{max}(f'') (x-x_0)(x-x_N)}{2} \sim O(\Delta x^2) \]
§ Polynomial approximation

Lagrange error: \[ \left| f(x) - \sum_{i=0}^{N} \frac{f(x_i)}{(x-x_i)^{N+1}} \prod_{j=0}^{N} (x-x_j) \right| \]

dependent on the choices of nodes

Consider \([a,b] = [-1,1]\)

(i) uniformly spaced nodes: \(x_{k+1} - x_k = \text{constant}; x_k = \frac{2k}{N} - 1, k = 0,1,\ldots,N\)

(ii) Chebychev nodes: \(x_k = \text{roots of the } N+1^\text{th} \text{ Chebychev polynomial} \]

\(T_{N+1}(x) = \cos\left((N+1)\cos^{-1} x\right)\)

\((N+1)\cos^{-1} x_k = \frac{\pi}{2} + k\pi, k = 0,1,\ldots,N\)

§ Nevill’s method --- computing higher-order polynomial based on lower-order polynomials

Define \(P_{m_1,m_2,\ldots,m_k}(x) = \text{the polynomial of degree } k-1 \text{ passing through} \)

the points \((x_{m_1}, y_{m_1}), (x_{m_2}, y_{m_2}), \ldots, (x_{m_k}, y_{m_k})\)

e.g. \(\{m_k\}_{k=1}^{3} = \{1,2,4\}\)

\(P_{1,2,4}(x) = \text{the polynomial of degree } 2 \text{ passing through} \)

the points \((x_1, y_1), (x_2, y_2), (x_4, y_4)\)
• Construct the $N^{th}$ order polynomial using the $(N-1)^{th}$ polynomials

$$p_{0,1,\ldots,N}(x) = \frac{(x-x_j)p_{0,1,\ldots,j-1,N}(x) - (x-x_i)p_{0,1,\ldots,i+1,N}(x)}{(x_i - x_j)}$$

$p_j(x) = y_j$

$p_j(x) = y_j$  $p_{j,N+1}(x)$

$p_j(x) = y_j$  $p_{j,N+1}(x)$  $p_{j,N+2}(x)$

$p_j(x) = y_j$  $p_{j,N+1}(x)$  $p_{j,N+2}(x)$  $p_{0,1,\ldots,N}(x)$

§ Newton form of interpolation polynomial

Given: $x^*$

Find: $p_j(x)$

Write: $p_j(x) = \sum_{i=0}^{N} \left( \prod_{j=0}^{i-1} (x-x_j) \right) a_i$

$= a_j + a_j(x-x_0) + a_j(x-x_0)(x-x_1) + a_j(x-x_0)(x-x_1)(x-x_2) + \cdots$

$+ a_j(x-x_0)(x-x_1)(x-x_2)(x-x_{N-1})$

$P_{j,N}(x) = (x-x_j)P_{j-1,N-1}(x) - (x-x_j)P_{j-1,N}(x) (x-x_j)$

Because $P_{0,1,\ldots,j,N}(x) = y_j$ for $k \neq j$

$P_{0,1,\ldots,j,N}(x) = y_j$ for $k \neq i$

- $P_{0,1,\ldots,j,N}(x) = \frac{(x_j-x_j)y_j-(x_i-x_j)y_i}{(x_i-x_j)} = y_{j}$ for $k \neq i, j$

- $P_{0,1,\ldots,j,N}(x) = \frac{(x_i-x_j)y_j-(x_i-x_i)y_i}{(x_i-x_j)} = y_{j}$

- $P_{0,1,\ldots,j,N}(x) = \frac{(x_j-x_j)y_j-(x_i-x_i)y_i}{(x_i-x_j)} = y_{j}$

$P_j(x) = a_j + a_j(x-x_0) + a_j(x-x_0)(x-x_1) + a_j(x-x_0)(x-x_1)(x-x_2) + \cdots$

$+ a_j(x-x_0)(x-x_1)(x-x_2)(x-x_{N-1})$

$= a_j + \left[ a_j + a_j(x-x_0) + a_j(x-x_0)(x-x_1) + a_j(x-x_0)(x-x_1)(x-x_2) + \cdots \right]$

$+ a_j(x-x_0)(x-x_1)(x-x_2)(x-x_{N-1})$

$= a_j + (x-x_0) \left[ a_j + (x-x_0) \left[ a_j + (x-x_0) \left[ a_j + \cdots \right] \right] \right]$

$P_x(x) = a_j + (x-x_0) * \left[ a_j + (x-x_0) * \left[ a_j + \cdots \right] \right]$

nested multiplication
§ Newton form of interpolation polynomial

\[ P_n(x') = a_n + (x' - x_1)^* \left[ a_1 + (x' - x_2)^* \left[ a_2 + (x' - x_3)^* \left[ a_3 + \cdots \right] \right] \right] \]

**STEP1:** compute all \( a_k, k = 0, 1, \ldots, N \)

**STEP 2:** \( b_0 = a_0 \)

\[ b_{y_{i+1}} = a_y + a_y (x' - x_{y+1}) = a_{y+1} + b_y (x' - x_{y+1}) \]

\[ b_{y_{i+1}} = a_y + a_y (x' - x_{y_{i+1}}) b_{y_{i+1}} \]

\[ b_k = a_k + b_{k+1} (x' - x_k), \quad k = N-1, N-2, \ldots, 1, 0 \]

\[ P_k(x') = b_k \]

**STEP1:** compute all \( a_k, k = 0, 1, \ldots, N \)

Define Newton’s divided difference:

\[ f[x_i] = y_i \]

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

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

\[ a_x = f[x_{y_{i+1}}, x_{y_{i+1}}, \ldots, x_{y_{i+1}}] \]

\[ f[x_x] = y_x \]

\[ f[x_{y_{i+1}}, x_{y_{i+1}}] = a_{x} \]

\[ f[x_{y_{i+1}}, x_{y_{i+1}}, x_{y_{i+1}}, x_{y_{i+1}}] = a_{x} \]

\[ f[x_{y_{i+1}}, x_{y_{i+1}}, x_{y_{i+1}}, x_{y_{i+1}}] = a_{x} \]

\[ P_k(x) = \sum_{x_{k=0}}^{x_{x}} \prod_{x_{k=0}}^{x_{x}} (x - x_k) \]

\[ P_k(x') = b_k \]

\[ b_x = a_x \]

\[ b_k = a_x + b_{x+1} (x' - x_k), \quad k = N-1, N-2, \ldots, 1, 0 \]
§ Hermite interpolation

Given: \( f(x_k) \) and \( f'(x_k) \), \( k = 0, 1, \ldots, N \)
Find: a polynomial passing all the points with the given slops

Theorem: If \( f \in C^{n}[a,b] \) and \( x_0, x_1, \ldots, x_N \) are distinct, the unique polynomial of least degree agreeing \( f \) and \( f' \) at \( x_0, x_1, \ldots, x_N \) is the polynomial of degree \( 2N + 1 \) given by

\[
H_{2N+1}(x) = \sum_{j=0}^{N} f(x_j) H_{N,j}(x) + \sum_{j=0}^{N} f'(x_j) \hat{H}_{N,j}(x)
\]

\[
H_{N,j}(x) = \left[ 1 - 2(x - x_j) L_{N,j}^{k}(x) \right] * L_{N,j}^{k}(x)
\]

\[
\hat{H}_{N,j}(x) = (x - x_j) * L_{N,j}^{k}(x)
\]

§ Newton’s form of Hermite polynomial

Define \( \hat{x}_k = \hat{x}_{2N+1} = x_k \), i.e. \( \{ \hat{x}_k \}_{k=0}^{2N} = \{ x_0, x_1, x_2, \ldots, x_N \} \)

\[
H_{2N+1}(x) = a_0 + a_1 (x - \hat{x}_0) + a_2 (x - \hat{x}_0)^2 + a_3 (x - \hat{x}_0) (x - \hat{x}_1) + a_4 (x - \hat{x}_0) (x - \hat{x}_1) (x - \hat{x}_2) + \cdots
\]

\[
= a_0 + a_1 (x - x_0) + a_2 (x - x_0)^2 + a_3 (x - x_0)^3 + \cdots
\]

\[
= a_0 + \left[ (x - \hat{x}_0) + (x - \hat{x}_1) + (x - \hat{x}_2) + \cdots \right]
= \left[ a_{2N+1} + a_{2N+1} \right] *
\]

Given: \( x' \in [a,b] \)

Want: \( f(x') = H_{2N+1}(x') \) = ?

STEP 1: find \( a_k \), \( k = 0, 1, 2, \ldots, 2N + 1 \)

STEP 2: \( b_{2N+1} = a_{2N+1} \)

\[ b_k = a_k + b_{k+1} \times (x' - x_k) \text{ for } k = 2N, 2N - 1, \ldots, 1, 0 \]

\[ b_0 = H_{2N+1}(x') \]
§ Newton’s form of Hermite polynomial

Define $\hat{x}_k = x_{2k+1} = x_k$, i.e. $\{ \hat{x}_k \}_{k=0}^{N}$

Define $f[\hat{x}_k] = f[\hat{x}_{2k+1}] = f(x_k)$

Define $f[\hat{x}_k, \hat{x}_{k+1}] = f'(x_k)$

Define

$$
\begin{align*}
&f[\hat{x}_k, \hat{x}_{k+1}, \hat{x}_{k+2}, \ldots, \hat{x}_{2k+1}] = \\
&\quad \frac{f(x_k) - f(x_{k+1})}{x_k - x_{k+1}}
\end{align*}
$$

Define

$$
\begin{align*}
&f[\hat{x}_k, \hat{x}_{k+1}, \ldots, \hat{x}_{k+d}] = \\
&\quad \frac{f[\hat{x}_k, \hat{x}_{k+1}, \ldots, \hat{x}_{k+d}]}{\hat{x}_{k+d+1} - \hat{x}_k}
\end{align*}
$$

for $k \geq 3$

§ Cubic spline interpolation ~ piecewise cubic polynomials

Given: $(x_k, y_k = f(x_k))$, $k = 0$

high-order polynomials $\Rightarrow 0$

Alternative choice: piecewise polynomial approximation

Piecewise linear interpolation:
disadvantage: discontinuous first derivatives at nodes in general

Piecewise cubic interpolation:
Goal: continuous 1st and 2nd derivatives at nodes

Use a cubic polynomial to fit for each subinterval, $S_k(x)$ for $x \in [x_k, x_{k+1}]$

$$
\begin{align*}
&\text{C}^0: \quad S_0(x) = y_0 \quad S_0(x) = y_1 \\
&\quad S_1(x) = y_1 \quad S_1(x_2) = y_2
\end{align*}
$$

6 constraints

$$
\begin{align*}
&\text{C}^1: \quad S_0'(x) = S_1'(x) \\
&\quad S_1'(x) = S_2'(x)
\end{align*}
$$

8 degrees of freedom
Cubic Spline Interpolation:

Let \( a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b \). A cubic spline interpolant \( S(x) \) for \( f(x) \) is a function that satisfies the following conditions:

(i) \( S(x) \) is a cubic polynomial on the interval \([x_i, x_{i+1}]\), denoted by \( S_i(x) \), for \( k = 0, 1, 2, \ldots, N-1 \)

(ii) \( S_i(x_i) = y_i \) and \( S_i(x_{i+1}) = y_{i+1} \), for \( k = 0, 1, 2, \ldots, N-1 \)

(iii) \( S_i''(x_i) = S_i''(x_{i+1}) \), for \( k = 0, 1, 2, \ldots, N-2 \)

(iv) \( S_i''(x_{i+1}) = S_i''(x_i) \), for \( k = 0, 1, 2, \ldots, N-2 \)

4\(N\) degrees of freedom!

\[ S(x) = \sum_{i=0}^{N-1} S_i(x) \phi_i(x) \]

for \( k = 0, 1, 2, \ldots, N-1 \)

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

\[ S_i'(x) = b_i + 2c_i (x-x_i) + 3d_i (x-x_i)^2 \]

\[ S_i''(x) = 2c_i + 6d_i (x-x_i) \]

define \( h_i = x_{i+1} - x_i \) for \( k = 0, 1, \ldots, N-1 \)

\[ C^0: \quad S_i(x_i) = a_i = y_i \]

\[ S_i(x_{i+1}) = a_i + b_i h_i + c_i h_i^2 + d_i h_i^3 = y_{i+1} \]

for \( k = 0, 1, \ldots, N-1 \)

boundary conditions:

(a) free or natural spline: \( S_i''(x_i) = S_i''(x_{i+1}) = 0 \)

(b) clamped spline: \( S_i''(x_i) = f''(x_i) \) and \( S_i''(x_{i+1}) = f''(x_{i+1}) \)

\[ C^1: \quad \text{for} \ k = 0, 1, \ldots, N-2 \]

\[ S_i'(x) = b_i + 2c_i (x-x_i) + 3d_i (x-x_i)^2 \]

\[ S_i''(x) = b_i + 2c_i h_i + 3d_i h_i^2 \]

\[ S_i''(x_i) = b_i \]

\[ C_i^1: \quad \text{for} \ k = 0, 1, \ldots, N-2 \]

\[ S_i''(x) = 2c_i + 6d_i (x-x_i) \]

\[ S_i''(x) = 2c_i + 6d_i h_i = 2c_i + 6d_i h_i^3 = b_i \]
Summary:

(i) \( y_k = a_k \quad (k = 0, 1, 2, \ldots, N-1) \)

(ii) \( y_{k+1} = a_k + b_k h_k + c_k h_k^2 + d_k h_k^3 \quad (k = 0, 1, \ldots, N-1) \)

(iii) \( b_k + 2c_k h_k + 3d_k h_k^2 = h_{k+1} \quad (k = 0, 1, \ldots, N-2) \)

(iv) \( 2c_k + 6d_k h_k = 2c_{k+1} \quad (k = 0, 1, \ldots, N-2) \)

From (iv) : \( d_k = d_1 \quad (v) \)

Substitute (v) into (ii) : \( b_k = b_1 (d, \check{c}) \quad (vi) \)

Substitute (v)(vi) into (iii) : \( c_k = c_1 (\check{a}, \check{c}) \)

For \( k = 1, 2, \ldots, N-1 \)

\[
h_{k+1} c_{k+1} + 2(h_k + h_1) c_k + h_k c_{k+1} = \frac{3}{h_k} (a_{k+1} - a_k) - \frac{3}{h_{k+1}} (a_k - a_{k+1})
\]

\[
a_k = S_{x^{-1}}(x_k) = y_k = a_{x^{-1}} + b_{x^{-1}} h_{x^{-1}} + c_{x^{-1}} h_{x^{-1}}^2 + d_{x^{-1}} h_{x^{-1}}^3
\]

\[
2c_k = S_{x^{-1}+1}(x_k) = 2c_{x^{-1}} + 6d_{x^{-1}} h_{x^{-1}}
\]

\( \sim (N-1) \) equations for \((N+1)\) unknowns \( \{c_k\}_{k=0}^N \)

boundary conditions:

(a) free or natural spline: \( S_k^*(x_k) = S_{k+1}^*(x_k) = 0 \)

\( c_0 = c_N = 0 \)

(b) clamped spline: \( S_k^*(x_k) = f'(x_k) \) and \( S_{k+1}^*(x_k) = f''(x_k) \)

\( S_k^*(x_k) = b_k = f'(x_k), \quad S_{k+1}^*(x_k) = b_{k+1} \)

\( a_k = a_{x^{-1}} + b_{x^{-1}} h_{x^{-1}} + c_{x^{-1}} h_{x^{-1}}^2 + d_{x^{-1}} h_{x^{-1}}^3 = f'(x_k) \)

For \( k = 0, 1, 2, \ldots, N-1 \)

\[h_k a_k + (h_k + h_{k+1}) a_{k+1} + h_{k+1} a_{k+2} = \frac{3}{h_k} (a_{k+1} - a_k) - \frac{3}{h_{k+1}} (a_k - a_{k+1})
\]

\[
a_k = S_{x^{-1}}(x_k) = y_k = a_{x^{-1}} + b_{x^{-1}} h_{x^{-1}} + c_{x^{-1}} h_{x^{-1}}^2 + d_{x^{-1}} h_{x^{-1}}^3
\]

\[
2c_k = S_{x^{-1}+1}(x_k) = 2c_{x^{-1}} + 6d_{x^{-1}} h_{x^{-1}}
\]

\( \sim (N-1) \) equations for \((N+1)\) unknowns \( \{c_k\}_{k=0}^N \)

boundary conditions:

(a) free or natural spline: \( S_k^*(x_k) = S_{k+1}^*(x_k) = 0 \)

\( c_0 = c_N = 0 \)

(b) clamped spline: \( S_k^*(x_k) = f'(x_k) \) and \( S_{k+1}^*(x_k) = f''(x_k) \)

\( S_k^*(x_k) = b_k = f'(x_k), \quad S_{k+1}^*(x_k) = b_{k+1} + 2c_{x^{-1}} h_{x^{-1}} + 3d_{x^{-1}} h_{x^{-1}}^2 = f'(x_k) \)

\[h_k a_k + (h_k + h_{k+1}) a_{k+1} + h_{k+1} a_{k+2} = \frac{3}{h_k} (a_{k+1} - a_k) - \frac{3}{h_{k+1}} (a_k - a_{k+1})
\]

\[
a_k = S_{x^{-1}}(x_k) = y_k = a_{x^{-1}} + b_{x^{-1}} h_{x^{-1}} + c_{x^{-1}} h_{x^{-1}}^2 + d_{x^{-1}} h_{x^{-1}}^3
\]

\[
2c_k = S_{x^{-1}+1}(x_k) = 2c_{x^{-1}} + 6d_{x^{-1}} h_{x^{-1}}
\]

\( \sim (N-1) \) equations for \((N+1)\) unknowns \( \{c_k\}_{k=0}^N \)

boundary conditions:

(a) free or natural spline: \( S_k^*(x_k) = S_{k+1}^*(x_k) = 0 \)

\( c_0 = c_N = 0 \)

(b) clamped spline: \( S_k^*(x_k) = f'(x_k) \) and \( S_{k+1}^*(x_k) = f''(x_k) \)

\( S_k^*(x_k) = b_k = f'(x_k), \quad S_{k+1}^*(x_k) = b_{k+1} \)

\( a_k = a_{x^{-1}} + b_{x^{-1}} h_{x^{-1}} + c_{x^{-1}} h_{x^{-1}}^2 + d_{x^{-1}} h_{x^{-1}}^3 = f'(x_k) \)
§ Triagonal linear system: $Ax = r$

$$
\begin{pmatrix}
a_{11} & a_{12} & 0 & 0 & \cdots & 0 \\
a_{21} & a_{22} & a_{23} & 0 & \cdots & 0 \\
0 & a_{32} & a_{33} & a_{34} & \cdots & 0 \\
0 & 0 & a_{4,3} & a_{4,4} & \cdots & 0 \\
0 & 0 & 0 & a_{5,4} & \cdots & 0 \\
0 & 0 & 0 & 0 & \cdots & a_{6,5} \\
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5 \\
x_6 \\
\end{pmatrix}
= 
\begin{pmatrix}
h_1 \\
h_2 \\
h_3 \\
h_4 \\
h_5 \\
h_6 \\
\end{pmatrix}
$$

Given: $x^*$

Want: an approximation of $f(x^*)$

Step 1: find the subinterval that $x^*$ belongs to, say $x^* \in [x_i, x_{i+1}]$

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

$$
f(x^*) \approx S_i(x^*)
$$

Theorem: Let $f \in C^4[a,b]$ with $\max_{a \leq x \leq b} |f^{(4)}(x)| = M$. If $S(x)$ is the unique clamped cubic spline interpolant to $f$ with respect to the nodes $a \leq x_0 < x_1 < \cdots < x_N = b$. Then

$$
\max_{a \leq x \leq b} |f(x) - S(x)| \leq \frac{5M}{384} \max_{a \leq x \leq b} |x_{i+1} - x_i|^4
$$

c.f. natural spline: 4th order too.

§ Boundary conditions for spline interpolations

(i) Natural spline

$$
\begin{align*}
    f^*(x_0) &= 0 \\
    f^*(x_N) &= 0
\end{align*}
$$

This choice minimizes the value of $\int_a^b f^*(x) \, dx$.

(ii) Parabolic runout

$$
\begin{align*}
    f^*(x_0) &= f^*(x_1) \\
    f^*(x_N) &= f^*(x_{N-1})
\end{align*}
$$

implying

$$
\begin{align*}
    f^*(x_0) &= f^*(x_N) = 0 \\
    f^*(x_0) &= f^*(x_N) = 0
\end{align*}
$$

(parabolic ending curves)

(iii) Clamped spline

$$
\begin{align*}
    f^*(x_0), f^*(x_N), f^*(x_0) & \text{ known} \\
    f^*(x_0), f^*(x_N), f^*(x_0) & \text{ known}
\end{align*}
$$

(iv) Fit a cubic polynomial $C(x)$ to the four ending points.

$$
\begin{align*}
    f^*(x_0) &= C^*(x_0) = f^*(x_0) - f^*(x_N) \\
    f^*(x_N) &= C^*(x_N) = f^*(x_N) - f^*(x_{N-1})
\end{align*}
$$

(v) Periodic boundary condition

$$
\begin{align*}
    f(x_0) &= f(x_N) \\
    f^*(x_0) &= f^*(x_N) \\
    f^*(x_0) &= f^*(x_N)
\end{align*}
$$
§ Tension spline interpolation

Let \( a = x_0 < x_1 < \cdots < x_{N-1} < x_N = b \). A tension spline interpolant \( S(x) \) for \( f(x) \) is a function that satisfies the following conditions:

(i) \( S(x) \) is a cubic polynomial on the interval \([x_i, x_{i+1}]\), denoted by \( S_i(x) \), for \( k = 0, 1, 2, \cdots, N-1 \)

(ii) \( S_i(x_i) = y_i \) and \( S_i(x_{i+1}) = y_{i+1} \), for \( k = 0, 1, 2, \cdots, N-1 \)

(iii) \( S_i'(x_{i+1}) = S_i''(x_{i+1}) \), for \( k = 0, 1, 2, \cdots, N-2 \)

(iv) \( S_i'(x_{i-1}) = S_i''(x_{i-1}) \), for \( k = 0, 1, 2, \cdots, N-2 \)

(v) \( S_i(x) \) is a solution of \( S^{(iv)} - \tau S'' = 0 \).

\( \tau = 0 \) implies cubic polynomials \( \Rightarrow \) cubic spline!

\( \tau = \infty \) implies approximately linear fitting (still \( C^2 \))

\((N-1)\) equations for \((N+1)\) unknowns \( \{z_k = S''(x_k)\}, k = 0, 1, 2, \cdots, N \)

\( S_i(x) \) is a solution of \( S^{(iv)} - \tau S'' = 0 \).

\( \tau = 0 \Rightarrow \) cubic polynomials \( \Rightarrow \) cubic spline!

\( \tau = \infty \Rightarrow \) approximately linear fitting (still \( C^2 \))

\( (N-1)\) equations for \((N+1)\) unknowns \( \{z_k = S''(x_k)\}, k = 0, 1, 2, \cdots, N \)

§ Two dimensional interpolations

- Cartesian Product and Grid

Given: \( f_{ij} = f(x_i, y_j) \), \( 0 \leq i \leq M \), \( 0 \leq j \leq N \)

Solution: 2D interpolation \( \Rightarrow \) two-layer 1D interpolations
First, interpolate in one direction

\[ f(x, y) = \sum_{i=0}^{M} f(x_i, y_i) \times L_{M,j}(x) = \prod_{k=0}^{M} \frac{(x-x_i)}{(x_i-x_j)} \]

\[ f(x, y) = \sum_{j=0}^{N} f(x_j, y_j) \times L_{N,j}(y) = \prod_{k=0}^{N} \frac{(y-y_j)}{(y_j-y_k)} \]

\[ f(x, y) = \sum_{j=0}^{N} f(x_j, y_j) \times L_{N,j}(y) = \sum_{j=0}^{N} f_j \times L_{N,j}(y) \]

\[ f(x, y) = \sum_{i=0}^{M} \sum_{j=0}^{N} f_{ij} \times L_{M,j}(x) \times L_{N,j}(y) \]

### Example: \( N+1 = 3 = (m+1)(m+2)/2 \) or \( m=1 \)

\[ P_1(x, y) = C_{00} + C_{10}x + C_{01}y \]

\[ P_1(x_i, y_i) = C_{00} + C_{10}x_i + C_{01}y_i = f_i \] for \( k = 0, 1, 2 \)

\[
\begin{pmatrix}
1 & x_0 & y_0 \\
1 & x_1 & y_1 \\
1 & x_2 & y_2
\end{pmatrix}
\begin{pmatrix}
C_{00} \\
C_{10} \\
C_{01}
\end{pmatrix}
=
\begin{pmatrix}
f_0 \\
f_1 \\
f_2
\end{pmatrix}
\]

### Example: \( N+1 = 6 = (m+1)(m+2)/2 \) or \( m=2 \)

\[ P_2(x, y) = C_{00} + C_{10}x + C_{01}y + C_{20}x^2 + C_{11}xy + C_{02}y^2 \]

\[
\begin{pmatrix}
1 & x_0 & y_0 & x_0^2 & x_0y_0 & y_0^2 \\
1 & x_1 & y_1 & x_1^2 & x_1y_1 & y_1^2 \\
1 & x_2 & y_2 & x_2^2 & x_2y_2 & y_2^2 \\
1 & x_3 & y_3 & x_3^2 & x_3y_3 & y_3^2 \\
1 & x_4 & y_4 & x_4^2 & x_4y_4 & y_4^2 \\
1 & x_5 & y_5 & x_5^2 & x_5y_5 & y_5^2
\end{pmatrix}
\begin{pmatrix}
C_{00} \\
C_{10} \\
C_{01} \\
C_{20} \\
C_{11} \\
C_{02}
\end{pmatrix}
=
\begin{pmatrix}
f_0 \\
f_1 \\
f_2 \\
f_3 \\
f_4 \\
f_5
\end{pmatrix}
\]

Matrix invertable? Not always!

**Theorem**: Interpolation of arbitrary data by the subspace \( \Pi(x, y) \) is possible on a set of \((m+1)(m+2)/2\) nodes if these nodes lie on lines \( L_0, L_1, \ldots, L_m \) in such a way that for each \( L_i \) contains exactly \((i+1)\) nodes.
§ Shepard interpolation method

Write $P=(x,y)$ represents a point in the $R^2$ space. $Q$ is another point.

Let $\Phi(P,Q)$ be a real-valued function on $R^2 \times R^2$ which satisfies

\[ \Phi(P,Q) = 0 \text{ if and only if } P = Q \]

e.g. $\Phi(P,Q) = |P-Q|^2 = (x_p - x_q)^2 + (y_p - y_q)^2$

Define 2D Lagrange interpolation function as

\[ L_{\delta_j}(P) = \prod_{i=0}^{N} \frac{\Phi(P,P_i)}{\Phi(P,P_i)} \]

or written as $L_{\delta_j}(x,y) = \prod_{i=0}^{N} \frac{\Phi(x,y;x_i,y_i)}{\Phi(x_i,y;x_i,y_i)}$

References: Kincaid & Cheny

Lancaster & Salkauskas “Curve & Surface Fitting” (1980)