Spline

- So far, we have only thought about going through the specified points
- If there is large number of data points,
  - Use a high-order polynomial that passes through them all [might show oscillations!]
  - Fit a somewhat high order polynomial to each interval and match derivatives at each point — spline

Cubic Spline

- Spline is a polynomial between each pair of points.
- But coefficients of this polynomial are determined slightly non-locally.
- Smooth (=continuous + differentiable), avoids oscillations
- Ultimate method for piecewise polynomial interpolation of strongly varying data. Very simple local form, but globally flexible and smooth
- Achieved by requiring continuity of function at data points but also for up to the *l*th derivative.
- *l*=2 for cubic spline

– Uses :

(i) interpolation condition for function

(ii) boundary conditions for smoothness for 2nd derivative

(iii) 2 remaining conditions from assuming the 2nd derivative value at edges [Natural spline : set these edge values of p” to be zero] n-2 equations for p”

$$h_{j-1}p''_{j-1} + (2h_j + 2h_{j-1})p''_j + h_j p''_{j+1} = 6\left(\frac{p_{j+1} - p_j}{h_j} - \frac{p_j - p_{j-1}}{h_{j-1}}\right), \qquad j = 2, \dots, n-1. \quad (3.35)$$

2 additional equations for end points, from p’(x)

$$2h_1 p''_1 + h_1 p''_2 = 6\frac{p_2 - p_1}{h_1} - 6p'_1, \quad (3.36)$$

$$h_{n-1}p''_{n-1} + 2h_{n-1}p''_n = -6\frac{p_n - p_{n-1}}{h_{n-1}} + p'_n. \quad (3.37)$$

Along with assuming p''(x_1) and p'' (x_n) = 0
(natural spline)

$$\begin{bmatrix} 1 & & & & & \\ & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & & \ddots & & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & \\ & & & & & 1 \end{bmatrix} \begin{bmatrix} p_1'' \\ p_2'' \\ p_3'' \\ \vdots \\ p_{n-1}'' \\ p_n'' \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 6\frac{p_3-p_2}{h_2} - 6\frac{p_2-p_1}{h_1} \\ 6\frac{p_4-p_3}{h_3} - 6\frac{p_3-p_2}{h_2} \\ \vdots \\ 6\frac{p_n-p_{n-1}}{h_{n-1}} - 6\frac{p_{n-1}-p_{n-2}}{h_{n-1}} \\ 0 \end{bmatrix}. \tag{3.39}$$

Leads to a set of tridiagonal linear equations

To solve *n* Tridiagonal Linear Eqns

- Elimination & Back substitution
- Pattern develops

$$\beta_1 x_1 + c_1 x_2 = \rho_1$$

$$\beta_j x_j + c_j x_{j+1} = \rho_j \qquad j=2, n\text{-}1$$

$$\beta_n x_n = \rho_n$$

- a_j, b_j, c_j, r_j are known
  - First assign β_1 and ρ_1
  - *for loop* j=2,n
  - Evaluate β_j and ρ_j [you will need the (j-1) value in both cases; so go in this sequence]
- Now you have all values of β and ρ
- And you are all set to evaluate x_j values
  - First evaluate $x_n = \rho_n / \beta_n$
  - Evaluation of x_j will require x_{j+1}
  - *for loop* : j = n-1, 1
  - Evaluate $x_j = (\rho_j - c_j x_{j+1})/\beta_j$

For j=2, n $\qquad \beta_j = b_j - \dfrac{a_j}{\beta_{j-1}} c_{j-1}$

$$\rho_j = r_j - \frac{a_j}{\beta_{j-1}} \rho_{j-1}$$

For j=1, $\qquad \beta_1 = b_1 \qquad \rho_1 = r_1$