计算物理

Lecture 8

傅子文

fuziwen@scu.edu.cn

Today's Lecture

- Interpolation & Approximation II
  - cubic splines
    - Clamped versus natural
    - Matrix techniques
    - Tridiagonal solvers
  - Approximating derivatives
  - Richardson Extrapolation

Introduction to Splines

- An alternative approach to using a single (n-1)th order polynomial to interpolate between n points is to apply lower-order polynomials in a piecewise fashion to subsets of data points.
- These connecting polynomials are called *spline functions*.
- Splines minimize **oscillations** and reduce **round-off error** due to their lower-order nature.

Higher Order vs. Splines

- Splines eliminate oscillations by using small subsets of points for each interval rather than every point. This is especially useful when there are jumps in the data:

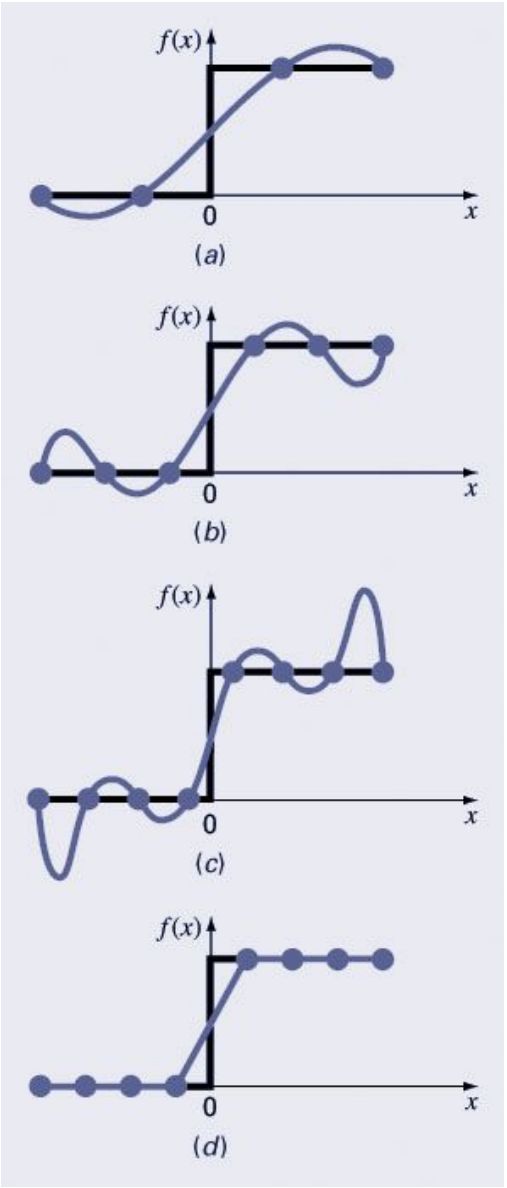
a) 3rd order polynomial

– 5th order polynomial

– 7th order polynomial

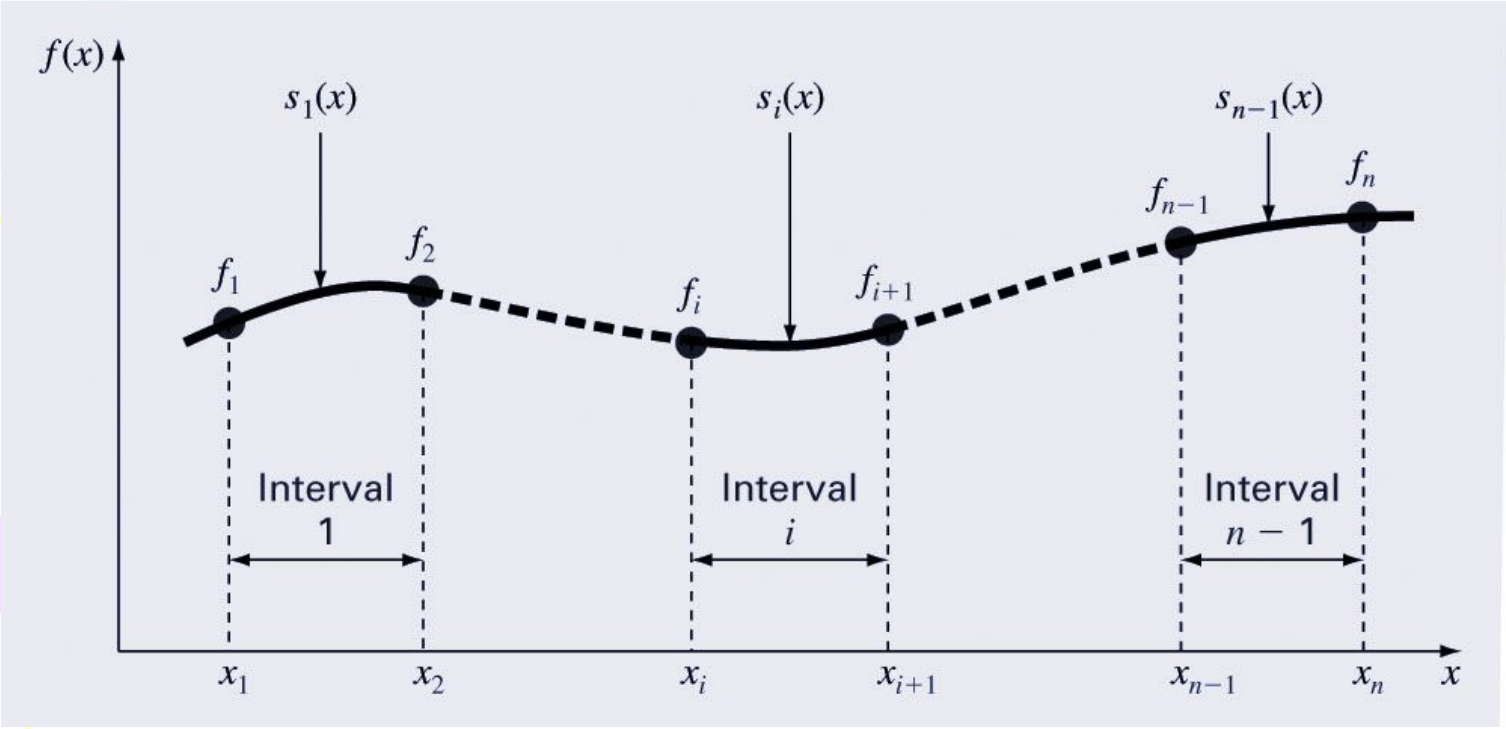
– Linear spline

  - seven 1st order polynomials generated by using pairs of points at a time



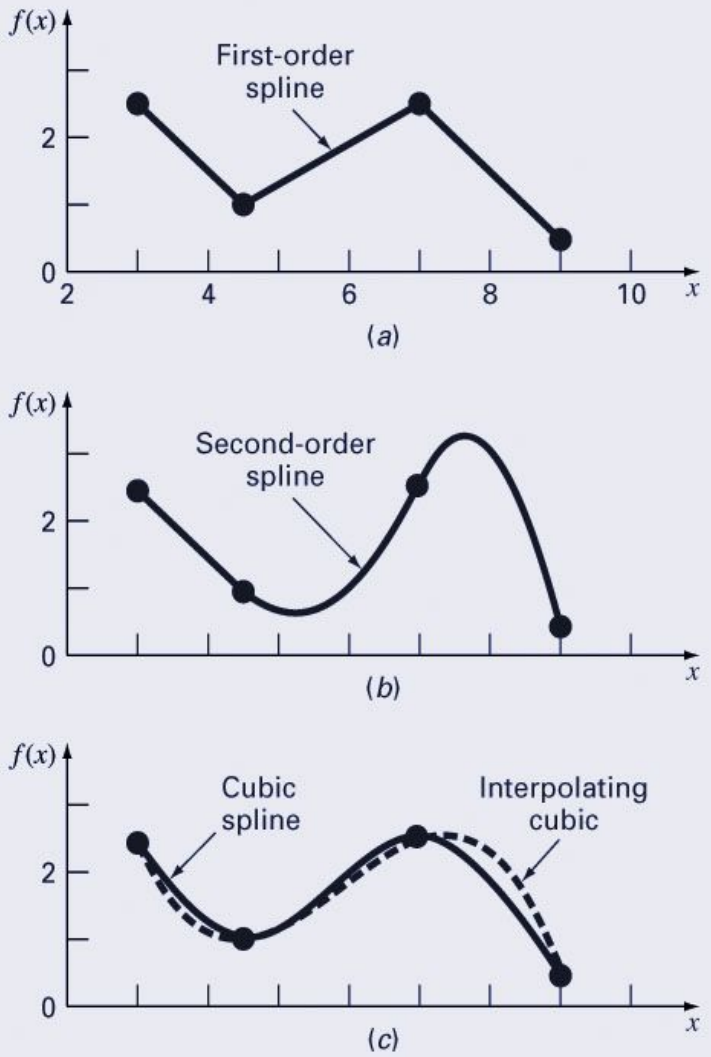
Spline Development

- Spline function ($s_i(x)$)coefficients are calculated for each interval of a data set.
- The number of data points (f_i) used for each spline function depends on the order of the spline function.



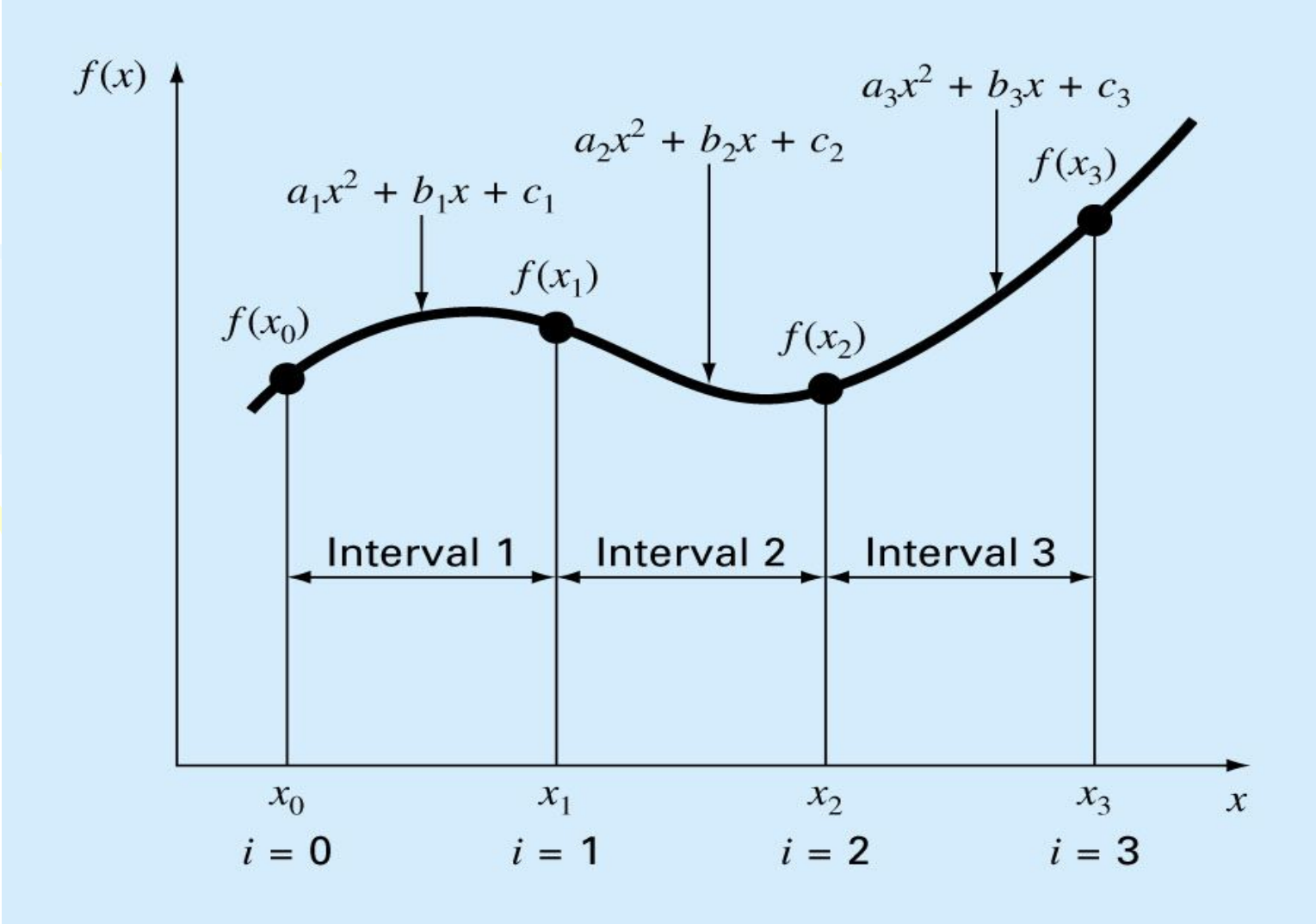
Spline Development

- First-order splines find straight-line equations between each pair of points that
  - Go through the points
- Second-order splines find quadratic equations between each pair of points that
  - Go through the points
  - Match first derivatives at the interior points
- Third-order splines find cubic equations between each pair of points that
  - Go through the points
  - Match first and second derivatives at the interior points



Note that the results of cubic spline interpolation are different from the results of an interpolating cubic

Quadratic spline



Cubic Splines

- While data of a particular size presents many options for the order of spline functions, cubic splines are preferred because they provide the simplest representation that exhibits the desired appearance of smoothness.
  - Linear splines have discontinuous first derivatives
  - Quadratic splines have discontinuous second derivatives and require setting the second derivative at some point to a pre-determined value
    but
  - Quartic or higher-order splines tend to exhibit the instabilities inherent in higher order polynomials (ill-conditioning or oscillations)

Cubic Splines

- Let's examine how we can create a series of cubic polynomials with equal derivatives at the datums
- Choose x, such that $x_j<x<x_{j+1}$ where $j = 1,\ldots,n$
- Around x, we may write $p(x)$ as

$$p(x) = a_j(x-x_j)^3 + b_j(x-x_j)^2 + c_j(x-x_j) + d_j \qquad (1)$$

as in the discussion on Lagrange interpolation set $x = x_j$ then

$$p(x_j) \equiv p_j = d_j = f(x_j) \qquad (2)$$

If we set $x = x_{j+1}$ then

$$p(x_{j+1}) \equiv p_{j+1} = a_j(x_{j+1}-x_j)^3 + b_j(x_{j+1}-x_j)^2 + c_j(x_{j+1}-x_j) + p_j$$

where we have substituted for $d_j = p_j$ using (2)

If we let $x_{j+1} - x_j = h_j$ then

$$p(x_{j+1}) \equiv p_{j+1} = a_j h_j^3 + b_j h_j^2 + c_j h_j + p_j \qquad (3)$$

Next consider derivatives

- We just derived two equations (3) & (2) for the value of the cubic fit at x_j and x_{j+1}

- Taking the first & second derivative of $p(x)$ we find

$$p'(x) = 3a_j(x - x_j)^2 + 2b_j(x - x_j) + c_j \quad \text{and}$$
$$p''(x) = 6a_j(x - x_j) + 2b_j$$

- If we again equate at x_j and x_{j+1} but using 2nd deriv:

$$p''(x_j) \equiv p''_j = 2b_j \; so \; b_j = p''_j/2 \qquad (4) \qquad \text{and}$$
$$p''(x_{j+1}) \equiv p''_{j+1} = 6a_j h_j + 2b_j$$
$$\Rightarrow p''_{j+1} = 6a_j h_j + p''_j \qquad \text{solve for } a_j \text{ to get}$$
$$\therefore a_j = \frac{1}{6}\frac{p''_{j+1} - p''_j}{h_j} \qquad (5)$$

Next find the c_j

- We've just shown

$$a_j = \frac{1}{6}\frac{p''_{j+1} - p''_j}{h_j} \quad (5), \quad b_j = \frac{p''_j}{2} \quad (4)$$

we substitute into

$$p(x_{j+1}) \equiv p_{j+1} = a_j h_j^3 + b_j h_j^2 + c_j h_j + p_j \quad (3)$$

to get

$$p_{j+1} = \frac{1}{6}(p''_{j+1} - p''_j)h_j^2 + p''_j \frac{h_j^2}{2} + c_j h_j + p_j$$

$$\therefore \quad c_j = \frac{p_{j+1} - p_j}{h_j} - \frac{1}{6}\left(h_j p''_{j+1} + 2h_j p''_j\right) \quad (6)$$

Substitute back to get $p(x)$

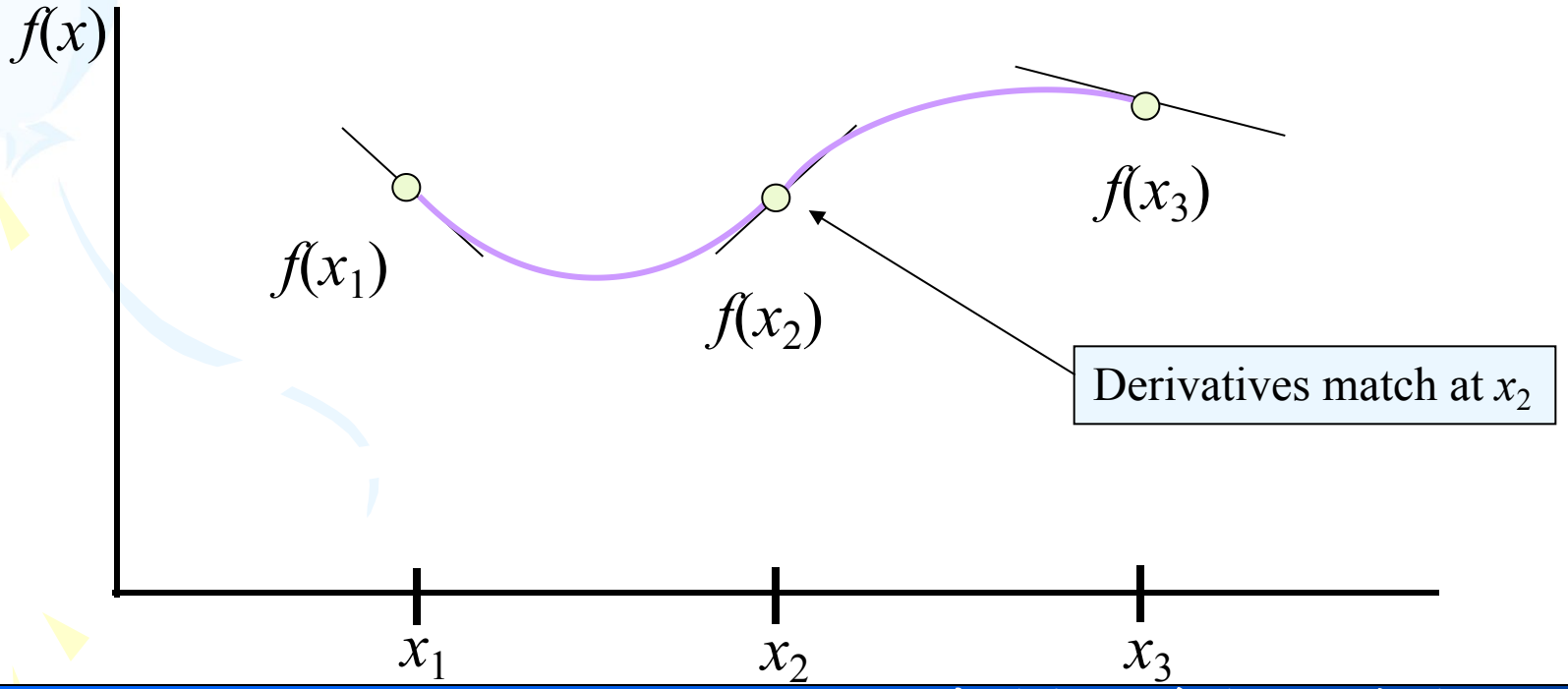
- We now substitute (2),(4),(5) and (6) back into our first equation for $p(x)$ to get

$$p(x) = \frac{p''_{j+1} - p''_j}{6h_j}(x - x_j)^3 + \frac{p''_j}{2}(x - x_j)^2 + \left[\frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3}\right](x - x_j) + p_j \qquad (7)$$

To get any further we now need some information about p''_{j+1} *and* p''_j

Setting p''_{j+1} and p''_j

- We have no information about the derivative of $f(x)$ though
  - We just have the series of $f(x)$ values
- What requirements can we impose?
- Recall we want the derivatives of neighbouring polynomials to match at the datums (x_j)



Apply continuity in the derivative

- We now have to consider adjacent polynomials
- Label:
  - $p_{j,j+1}$ to be the fit between x_j and x_{j+1}
  - $p_{j-1,j}$ to be the fit between x_{j-1} and x_j
- Continuity of the derivative implies that $p'_{j,j+1}$ *and* $p'_{j-1,j}$ should be equal at x_j, the general p' forms are:

$$p'_{j,j+1}(x) = \frac{p''_{j+1} - p''_j}{2h_j}(x - x_j)^2 + p''_j(x - x_j) + \frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3} \qquad (8)$$

$$p'_{j-1,j}(x) = \frac{p''_j - p''_{j-1}}{2h_{j-1}}(x - x_{j-1})^2 + p''_{j-1}(x - x_{j-1}) + \frac{p_j - p_{j-1}}{h_{j-1}} - \frac{h_{j-1} p''_j}{6} - \frac{h_{j-1} p''_{j-1}}{3} \qquad (9)$$

Equate at x_j

- Setting $p'_{j,j+1}(x_j) = p'_{j-1,j}(x_j)$ the first few terms of (8) vanish, while terms in $(x\text{-}x_{j-1})$ in (9) cancel with h_{j-1}:

$$p'_{j,j+1}(x_j) = \frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3}$$

$$p'_{j-1,j}(x_j) = \frac{p''_j - p''_{j-1}}{2} h_{j-1} + p''_{j-1} h_{j-1} + \frac{p_j - p_{j-1}}{h_{j-1}} - \frac{h_{j-1} p''_j}{6} - \frac{h_{j-1} p''_{j-1}}{3}$$

Equate and rearrange to get

$$h_{j-1} p''_{j-1} + 2(h_j + h_{j-1}) p''_j + h_j p''_{j+1} = 6\left[\frac{p_{j+1} - p_j}{h_j} - \frac{p_j - p_{j-1}}{h_{j-1}}\right] \qquad (10)$$

Where j=2,3,…,n-1

Recap: Origin of clamped versus natural

- Setting $p'_{j,j+1}(x_j) = p'_{j-1,j}(x_j)$ in the last lecture we found that

$$p'_{j,j+1}(x_j) = \frac{p_{j+1} - p_j}{h_j} - \frac{h_j p''_{j+1}}{6} - \frac{h_j p''_j}{3}$$

$$p'_{j-1,j}(x_j) = \frac{p''_j - p''_{j-1}}{2} h_{j-1} + p''_{j-1} h_{j-1} + \frac{p_j - p_{j-1}}{h_{j-1}} - \frac{h_{j-1} p''_j}{6} - \frac{h_{j-1} p''_{j-1}}{3}$$

Equate and rearrange to get

$$h_{j-1} p''_{j-1} + 2(h_j + h_{j-1}) p''_j + h_j p''_{j+1} = 6\left[\frac{p_{j+1} - p_j}{h_j} - \frac{p_j - p_{j-1}}{h_{j-1}}\right] \qquad (10)$$

Where j=2,3,…,n-1

Finding the p''_j values

- We now have a system of n-2 equations
  - Recall limits of j=2,3,…,n-1
- There are n unknown values though
  - The p''_j values range through j=1,…,n
- To have enough information to solve for all the values we have to provide information about the end points – two cases
  - Specify derivatives at end points
    - "Clamped spline"
  - Set $p''_1 = 0$ *and* $p''_n = 0$
    - "Natural spline"

Case 1: Clamped Spline

- Specify first derivatives at end points only
  - p_1' *and* p_n' are known
- Go back to equation for p' (8) at x_1 to get

$$p'_{1,2}(x_1) = \frac{p_2'' - p_1''}{2h_j}(x_1 - x_1)^2 + p_1''(x_1 - x_1) + \frac{p_2 - p_1}{h_1} - \frac{h_1 p_2''}{6} - \frac{h_1 p_1''}{3}$$

$$= \frac{p_2 - p_1}{h_1} - \frac{h_1 p_2''}{6} - \frac{h_1 p_1''}{3}$$

$$\Rightarrow 2h_1 p_1'' + h_1 p_2'' = 6\frac{p_2 - p_1}{h_1} - 6p_1' \qquad (11)$$

Similarly, for $j = n$:

$$h_{n-1} p_{n-1}'' + 2h_{n-1} p_n'' = 6\frac{p_n - p_{n-1}}{h_{n-1}} - 6p_n' \qquad (12)$$

Collect equations

- We now have a complete set of n equations in the n unknowns p_j'' :

$$h_{j-1}p''_{j-1} + 2(h_j + h_{j-1})p''_j + h_j p''_{j+1} = 6\left[\frac{p_{j+1} - p_j}{h_j} - \frac{p_j - p_{j-1}}{h_{j-1}}\right] \qquad (10)$$

$$\text{where } j = 2,3,4,\ldots,n\text{-}1$$

$$2h_1 p''_1 + h_1 p''_2 = 6\frac{p_2 - p_1}{h_1} - 6p'_1 \qquad (11)$$

$$h_{n-1}p''_{n-1} + 2h_{n-1}p''_n = 6p'_n - 6\frac{p_n - p_{n-1}}{h_{n-1}} \qquad (12)$$

- We now write this in matrix form

Matrix form for clamped splines

This matrix is said to have ***tridiagonal*** form: the only non zero elements are between the two dotted lines.

$$\begin{bmatrix} 2h_1 & h_1 & 0 & 0 & & & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & & & \\ 0 & h_2 & 2(h_2+h_3) & h_3 & & & \\ & & & & & & \\ 0 & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & & 0 & h_{n-1} & 2h_{n-1} \end{bmatrix} \begin{bmatrix} p_1'' \\ p_2'' \\ p_3'' \\ \\ \\ p_{n-1}'' \\ p_n'' \end{bmatrix} = 6 \begin{bmatrix} \frac{1}{h_1}(p_2-p_1)-p_1' \\ \frac{1}{h_2}(p_3-p_2)-\frac{1}{h_1}(p_2-p_1) \\ \frac{1}{h_3}(p_4-p_3)-\frac{1}{h_2}(p_3-p_2) \\ \\ \\ \frac{1}{h_{n-1}}(p_n-p_{n-1})-\frac{1}{h_{n-2}}(p_{n-1}-p_{n-2}) \\ p_n'-\frac{1}{h_{n-1}}(p_n-p_{n-1}) \end{bmatrix}$$

Eqn. (13)

Note typo in DeVries, equation 3.38

Case 2: Natural Spline

- $p_1'' = 0$ *and* $p_n'' = 0$ are both set to zero
  - Provides two additional equations to the set (10)

- Matrix form is then

$$\begin{bmatrix} 1 & 0 & 0 & 0 & & & 0 \\ 0 & 2(h_1+h_2) & h_2 & 0 & & & \\ 0 & h_2 & 2(h_2+h_3) & h_3 & & & \\ & & & & & & \\ & & & & & & \\ 0 & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & 0 & \\ & & & 0 & 0 & 1 & \end{bmatrix} \begin{bmatrix} p_1'' \\ p_2'' \\ p_3'' \\ \\ \\ p_{n-1}'' \\ p_n'' \end{bmatrix} = 6 \begin{bmatrix} 0 \\ \frac{1}{h_2}(p_3-p_2) - \frac{1}{h_1}(p_2-p_1) \\ \frac{1}{h_3}(p_4-p_3) - \frac{1}{h_2}(p_3-p_2) \\ \\ \\ \frac{1}{h_{n-1}}(p_n-p_{n-1}) - \frac{1}{h_{n-2}}(p_{n-1}-p_{n-2}) \\ 0 \end{bmatrix}$$

Eqn. (14)

How to calculate $p(x)$

- Given either (13) or (14) we have all the p_j necessary to ensure continuous derivatives at the datums

- Algorithm to find $f(x)$ given x and set of p_j:
  - Find j such that $x_j<x<x_{j+1}$
  - Warn user if $x<x_1$ or $x>x_n$ (cannot extrapolate outside region)
  - Solve tridiagonal system to find all p''_j from the p_j values
  - Once all p''_j are found then construct $p(x)$ using equation (7) and calculate value of $p(x)$ ($=f(x)$)

Tridiagonal Linear Systems

- Consider the following matrix equation

$$\begin{bmatrix} b_1 & c_1 & 0 & 0 & & 0 & \\ a_2 & b_2 & c_2 & 0 & & & \\ 0 & a_3 & b_3 & c_3 & & & \\ & & & & & & \\ & & & & & & \\ & 0 & & & a_{n-1} & b_{n-1} & c_{n-1} \\ & & & & 0 & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \\ \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \\ \\ r_{n-1} \\ r_n \end{bmatrix}$$

b_j $j=1,\ldots,n$ occupy the ***Main Diagonal***
a_j $j=2,\ldots,n$ occupy the ***Sub-diagonal***
c_j $j=1,\ldots,n$-1 occupy the ***Super-diagonal***

Consider Gaussian Elimination

- Let's perform Gaussian elimination on the second row:

$$\left[\begin{array}{cccccc|c}
b_1 & c_1 & 0 & 0 & & & r_1 \\
a_2 & b_2 & c_2 & 0 & & 0 & r_2 \\
0 & a_3 & b_3 & c_3 & & & r_3 \\
 & & & & & & \\
 & & & & & & \\
 & 0 & & & a_{n-1} & b_{n-1} & c_{n-1} & r_{n-1} \\
 & & & & 0 & a_n & b_n & r_n
\end{array}\right]$$

Step 1: Need to subtract a_2/b_1 of row 1 from row 2 to eliminate a_2 from the second row

"(2)=(2)- $(a_2/b_1)\times$(1)"

After performing first elimination

$$\left[\begin{array}{cccc|c}
b_1 & c_1 & 0 & 0 & r_1 \\
0 & b_2-\frac{a_2}{b_1}c_1 & c_2 & 0 & r_2-\frac{a_2}{b_1}r_1 \\
0 & a_3 & b_3 & c_3 & r_3 \\
 & & & & \\
 & & 0 & & \\
 & & & & \\
 & & a_{n-1} & b_{n-1} & c_{n-1} \;\; r_{n-1} \\
 & & 0 & a_n & b_n \;\; r_n
\end{array}\right]$$

To make notation easier, let

$$\beta_1 = b_1, \quad \rho_1 = r_1$$
$$\beta_2 = b_2 - (a_2/\beta_1)c_1$$
$$\rho_2 = r_1 - (a_1/\beta_1)\rho_1$$

$$\left[\begin{array}{cccc|c}
\beta_1 & c_1 & 0 & 0 & \rho_1 \\
0 & \beta_2 & c_2 & 0 & \rho_2 \\
0 & a_3 & b_3 & c_3 & r_3 \\
 & & & & \\
 & & 0 & & \\
 & & & & \\
 & & a_{n-1} & b_{n-1} & c_{n-1} \;\; r_{n-1} \\
 & & 0 & a_n & b_n \;\; r_n
\end{array}\right]$$

Next perform elimination on the third line

"(3)=(3)-(a_3/ β_2)×(2)"

Further substitution

$$\left[\begin{array}{cccc|c|ccc||c}
\beta_1 & c_1 & 0 & 0 & & & & & \rho_1 \\
0 & \beta_2 & c_2 & 0 & & 0 & & & \rho_2 \\
0 & 0 & \beta_3 & c_3 & & & & & \rho_3 \\
\hline
 & & & & & & & & \\
 & & & & & & & & \\
 & 0 & & & & a_{n-1} & b_{n-1} & c_{n-1} & r_{n-1} \\
 & & & & & 0 & a_n & b_n & r_n
\end{array}\right]$$

Where we have substituted

$$\beta_3 = b_3 - (a_3/\beta_2)c_2$$
$$\rho_3 = r_3 - (a_3/\beta_2)\rho_2$$

It should be clear that we can do the same pattern of elimination and then relabelling throughout the matrix.

After completing elimination steps

$$\left[\begin{array}{cccc|c} \beta_1 & c_1 & 0 & 0 & \\ 0 & \beta_2 & c_2 & 0 & \\ 0 & 0 & \beta_3 & c_3 & \\ & & & & \\ & & & & \\ & & & & \\ 0 & & & & \\ & & 0 & \beta_{n-1} & c_{n-1} \\ & & 0 & 0 & \beta_n \end{array}\right] \left\|\begin{array}{c} \rho_1 \\ \rho_2 \\ \rho_3 \\ \\ \\ \\ \\ \rho_{n-1} \\ \rho_n \end{array}\right.$$

We have reduced the tridiagonal system to a very simple form.

The very last row can immediately be solved.

Writing out the last row explicitly:

$$\beta_n x_n = \rho_n \Rightarrow x_n = \rho_n / \beta_n$$

We can now use *back substitution* to solve for all the remaining values.

Applying back substitution

- Writing out the penultimate line:

$$\beta_{n-1}x_{n-1} + c_{n-1}x_n = \rho_{n-1}$$

$$\Rightarrow x_{n-1} = \frac{1}{\beta_{n-1}}\left(\rho_{n-1} - c_{n-1}x_n\right)$$

- It should be clear that in general

$$x_{n-j} = \frac{1}{\beta_{n-j}}\left(\rho_{n-j} - c_{n-j}x_{n-j+1}\right), \qquad j = 1,\ldots,n-1$$

Summary of substitution process

- Once we have the form of the initial triadiagonal matrix:

$$\beta_1 = b_1 \;\&\; \rho_1 = r_1$$

- Substitute a_j then

$$\beta_j = b_j - \frac{a_j}{\beta_{j-1}} c_{j-1} \quad \text{and} \quad \rho_j = r_j - \frac{a_j}{\beta_{j-1}} \rho_{j-1} \qquad j = 2,\ldots,n$$

$$x_n = \rho_n / \beta_n$$

- Once this is done, solve and then use back-substitution to get remaining x_j

Basic algorithm for Cubic Spline interpolation

- (1) Use data $(x_j, f(x_j))$ to evaluate $p''(x_j)$ where $j=1,\ldots,n$ depending on whether user wants clamped or natural spline

- Calculate the interpolation polynomial around the two points that bracket the desired x value using the polynomial form (7)

Approximating derivatives

- If we take the algebraic definition of a derivative

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

- We would naturally generalize this to numerical approximation using

$$f_h'(x) = \frac{f(x+h) - f(x)}{h} \qquad \text{(A)}$$

- Where h would be a separation between points
- Note this is calculating a series of derivatives at the points $f(x)$
- Just how good is this estimate?

Estimating the accuracy

- If we Taylor expand about the point x, then

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2!}f''(x) + \frac{h^3}{3!}f'''(x) + \ldots$$

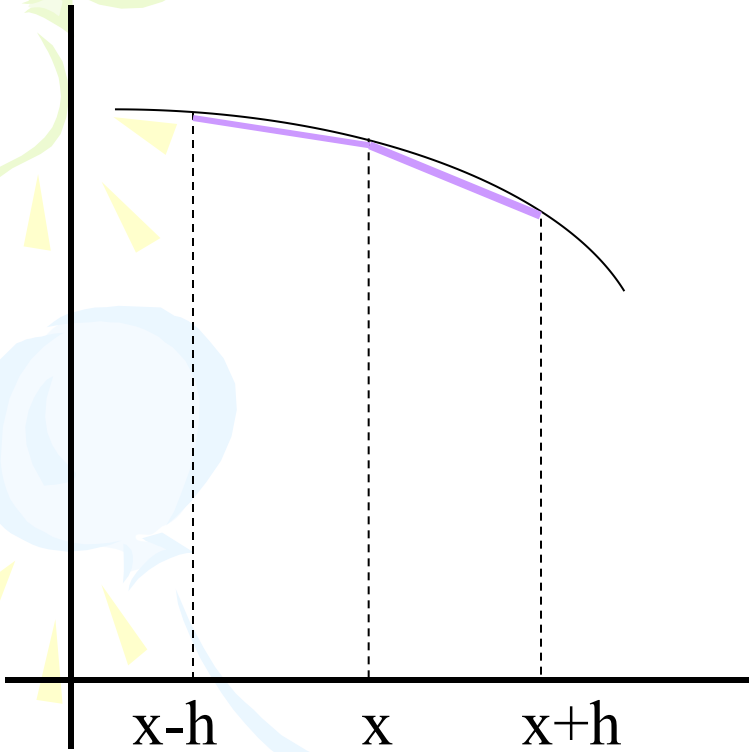
Rearrange to define

$$f'_f(x) = \frac{f(x+h) - f(x)}{h} - \underbrace{\frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) - \ldots}$$ (B)

Take this to be the error term which we write as E(h). In this case the lowest power that appears is h, so $E(h) \sim h$, or in "Big-O" notation $E(h)$ is O(h).

Remember h is small, so that $h^2<h$, hence $O(h^2)$ is more accurate than $O(h)$

Forwards vs Backwards difference



- The difference defined in (A) is called a *forward difference*
  - The derivative is clearly defined in the direction of increasing x
    - Using x & $x+h$
- We can equivalently define a *backwards difference*
  - The derivative is defined in the direction of decreasing x
    - Using x & $x-h$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2!}f''(x) - \frac{h^3}{3!}f'''(x) + \ldots$$

$$\Rightarrow f_b'(x) = \frac{f(x) - f(x-h)}{h} + \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \ldots \quad \text{(C)}$$

Improving the error

- It should be clear that both the forward & backward differences are $O(h)$
- However, by averaging them we can eliminate the term that is $O(h)$ there by defining the *centered difference*:

$$f_c'(x) = \frac{1}{2}\left(f_f'(x) + f_b'(x)\right)$$

$$= \frac{f(x+h) - f(x) + f(x) - f(x-h)}{2h} - \frac{h}{2}f''(x) + \frac{h}{2}f''(x) - \frac{h^2}{6}f'''(x) + \ldots.$$

$$f_c'(x) = \frac{f(x+h) - f(x-h)}{2h} - O(h^2)$$

Note that all odd ordered terms cancel too. Further the central difference is symmetric about x, forward & backward differences are clearly not.

Higher order differences

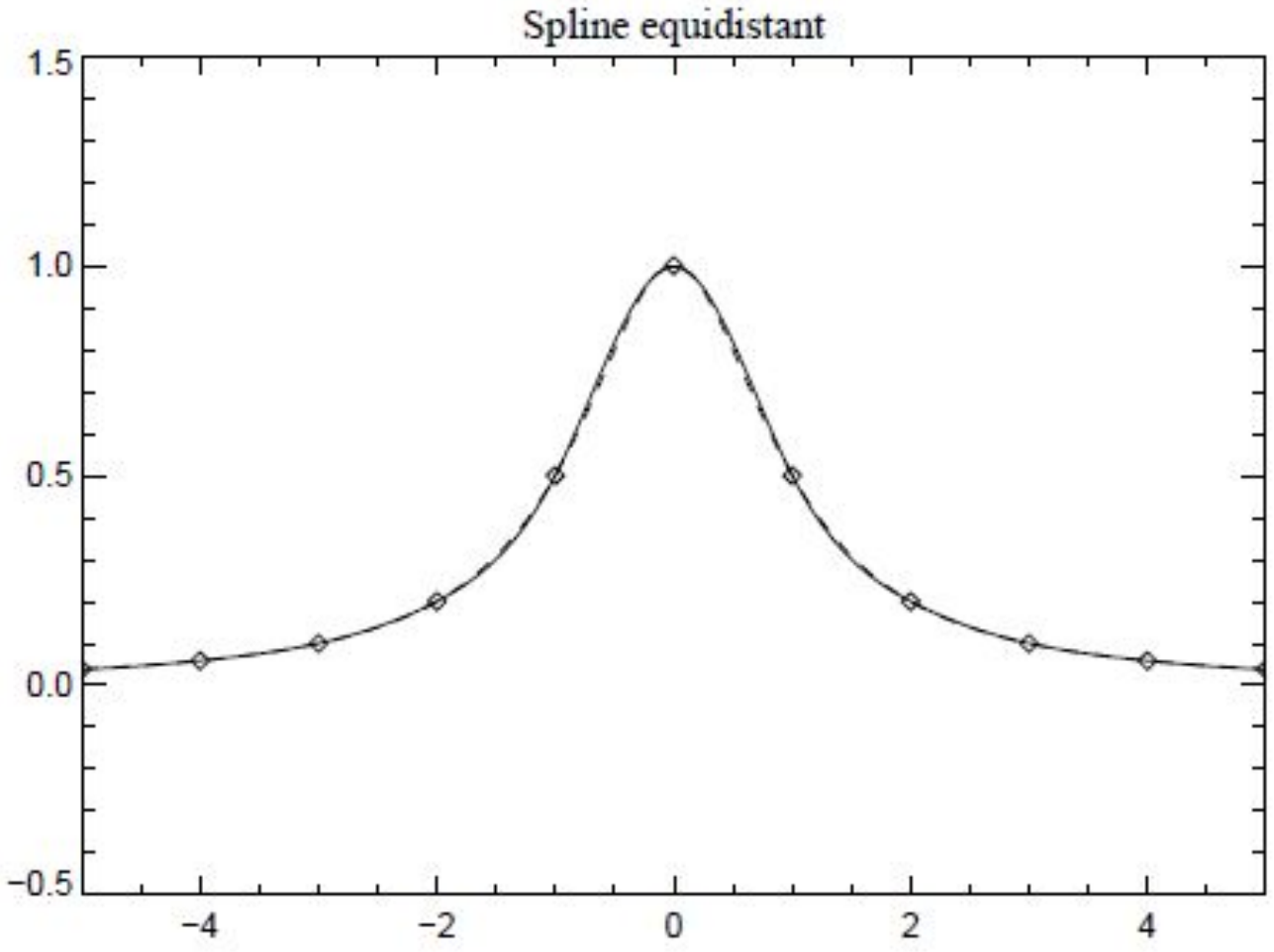
- We can also derive approximations to the second derivative by subtracting the backward difference from the forward (*i.e.* setting $f'_f(x) - f'_b(x) = 0$):

$$0 = f'_f(x) - f'_b(x) = \frac{f(x+h) - f(x) - f(x) + f(x-h)}{h} - \frac{h}{2} f''(x) - \frac{h}{2} f''(x) + O(h^3)$$

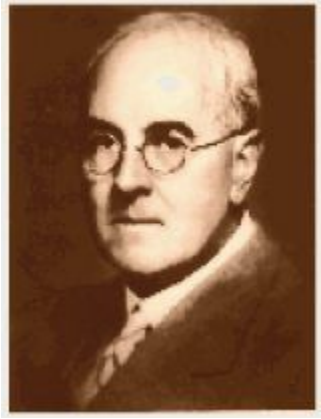
$$\Rightarrow f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2)$$

- This formula is also accurate to $O(h^2)$

Cubic spline fit to Runge Function $f(x)=\frac{1}{1+x^2}$



Richardson Extrapolation



- This is a very widely applicable idea due to L.F. Richardson (1881-1953)
  - We will see the idea again in the section on numerical integration of functions
  - The algorithm may seem somewhat unusual at first, but it is deceptively simple and elegant
- The idea is that if we know the order of the underlying error we can systematically improve the accuracy of a numerical estimate
  - Makes sequences converge faster too

Richardson made seminal contributions to weather forecasting and turbulence

Outline

- Let's consider the "three point"* central difference formula

$$f'_h(x) = \frac{f(x+h) - f(x-h)}{2h} - \frac{h^2}{6} f'''(x) + O(h^4) \quad (1)$$

The same formula applies if we use a step size 2h :

$$f'_{2h}(x)(x) = \frac{f(x+2h) - f(x-2h)}{4h} - \frac{4h^2}{6} f'''(x) + O(h^4) \quad (2)$$

- Notice that we can subtract ¼ of (2) from (1) to eliminate the leading term of the error

*Three point since it requires and produces information at three points

Eliminating the leading order error

- We can now define a new estimate of the derivative using these two formulae:

$$\frac{3}{4} f^{(4)\prime}(x) = f_h'(x) - \frac{1}{4} f_{2h}'(x)$$

(Prefactor of 3/4 since $f_h'(x), f_{2h}'(x)$ measure the same thing)

$$\Rightarrow$$

$$f^{(4)\prime}(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4) \quad \textbf{(3)}$$

- These are "five point" central difference formulae, with the furthest points receiving the least weight
- I have added a (4) superscript to denote order h^4 accurate

Eliminating the leading order error

- We can now define a new estimate of the derivative using these two formulae:

Similarly for the second derivative

$$f^{(4)\prime\prime}(x) = \frac{-f(x-2h)+16f(x-h)-30f(x)+16f(x+h)-f(x+2h)}{12h^2} + O(h^4) \quad \textbf{(4)}$$

- These are "five point" central difference formulae, with the furthest points receiving the least weight
- I have added a (4) superscript to denote order h^4 accurate

There is no reason to stop!

- Given our new formula for $f'(x)$ we could apply exactly the same approach again
  - The leading error term is now $O(h^4)$
- Since the factor of 2 in the change of step size will contribute 2^4 we need to subtract $1/2^4$ of $f'_{2h}(x)$ from $f'_h(x)$
  - Notice in the first step we only had to subtract $1/2^2=1/4$ because the error term was $O(h^2)$
- Define $$\frac{15}{16} f^{(6)\prime}(x) = f^{(4)\prime}{}_h(x) - \frac{1}{16} f^{(4)\prime}{}_{2h}(x)$$
$$\Rightarrow \quad f^{(6)\prime}(x) = \frac{16}{15}\left[f^{(4)\prime}{}_h(x) - \frac{1}{16} f^{(4)\prime}{}_{2h}(x) \right]$$
- And so on… but we can't continue indefinitely
  - Expressions rapidly get very complex
  - Better to work numerically…

A note on step size

- The key point is that from one step to the next we let h double in size (we could allow *h* to halve in size as well!)
  - This is independent of the number of points in the difference formula, or the positions at which data are taken from
  - The key point is the relative change in the value of h, and not the value of h itself
    - You can even think of resetting the value of h each time you want to extrapolate to the next highest level of approximation
- *e.g.* in the definition of $f^{(6)\prime}(x)$ we would use

$$f_h^{(4)\prime}(x) = \frac{f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)}{12h} + O(h^4) \quad (3)$$

$$f_{2h}^{(4)\prime}(x) = \frac{f(x-4h) - 8f(x-2h) + 8f(x+2h) - f(x+4h)}{24h} + O((2h)^4) \quad (3)$$

- However the error in $f_{2h}^{(4)\prime}(x)$ is $\propto (2h)^4$ so 2^4 times larger

Generalized notation

- Let $D_1(h)$ be the three point central difference operator we considered earlier (eqn (1))
- Since $E(D_1(h)) \sim h^2$
  - => $D_1(2h)$ has 4 times the error of $D_1(h)$

$$\therefore error = \frac{D_1(2h) - D_1(h)}{3} = \frac{D_1(2h) - D_1(h)}{2^2 - 1}$$

- Thus we can subtract this calculation of the error from $D_1(h)$ to create a more accurate difference operator ($D_2(h)$)

$$D_2(h) \equiv D_1(h) - \left[\frac{D_1(2h) - D_1(h)}{2^2 - 1} \right] = \frac{4D_1(h) - D_1(2h)}{3} \quad (5)$$

i=2 step

- Consider now the error in $D_2(h)$
- Since $E(D_2(h)) \sim h^4$
  - => $D_2(2h)$ has 16 times the error of $D_2(h)$

$$\therefore error = \frac{D_2(2h) - D_2(h)}{15} = \frac{D_2(2h) - D_2(h)}{2^4 - 1}$$

- Thus we can subtract this calculation of the error from $D_2(h)$ to create a more accurate difference operator ($D_3(h)$)

$$D_3(h) \equiv D_2(h) - \left[\frac{D_2(2h) - D_2(h)}{2^4 - 1} \right] = \frac{16D_2(h) - D_2(2h)}{15} \quad (6)$$

*i*th step

- Clearly this process generalizes

$$E(D_i(h)) \sim h^{2i} \Rightarrow D_i(2h) \text{ contains } 2^{2i} \text{ times error of } D_i(h)$$

$$\therefore error = \frac{D_i(2h) - D_i(h)}{2^{2i} - 1}$$

$$\Rightarrow D_{i+1}(h) \equiv D_i(h) - \left[\frac{D_i(2h) - D_i(h)}{2^{2i} - 1} \right] = \frac{2^{2i} D_i(h) - D_i(2h)}{2^{2i} - 1} \quad (7)$$

Numerical implementation

- In practice, we implement the scheme by actually *halving* the step size rather than doubling it
  - That allows us to use old values of D_i
- For example, consider $f(x)=xe^x$, and we want to find $f'(x)$ at $x=2$
  - By hand: $f'(x)=e^x+xe^x=e^x(1+x)$
  - Thus $f'(2)=3e^2=22.16716830\ldots$
  - This is solely used as a check of our convergence

Answer=22.16716830

Start with h=0.4

- Step 1: D_1=23.16346429, used $f(1.6)$ and $f(2.4)$
- Step 2: Set h=0.2
  - D_1=22.41416066, used $f(1.8)$ and $f(2.2)$
  - D_2=22.16439278, where we have used D_1 from step 1 in eqn (5)
- Step 3: Set h=0.1
  - D_1=22.22878688, used $f(1.9)$ and $f(2.1)$
  - D_2=22.16699562, where we have used D_1 from step 2 in eqn (5)
  - D_3=22.16716914, where we have used D_2 from step 2 in eqn (6)
- Step 4: Set h=0.05
  - D_1=22.18256486, used $f(1.95)$ and $f(2.05)$
  - D_2=22.16715752, where we have used D_1 from step 3 in eqn (5)
  - D_3=22.16716831, where we have used D_2 from step 3 in eqn (6)
  - D_4=22.16716830, where we have used D_3 from step 3 in eqn (7)

Notice that D_2 on step 2 (h=0.2) is a better estimate of the derivative than D_1 from step 3 (h=0.1). This shows the power of error removal process

Answer=22.16716830

Extrapolation table

- These results can be neatly expressed as follows

Step	h	D_1	D_2	D_3	D_4
1	0.4	23.16346429			
2	0.2	22.41416066	22.16439278		
3	0.1	22.22878688	22.16699562	22.16716914	
4	0.05	22.18256486	22.16715752	22.16716831	22.16716830

I have added arrows to indicate how values used in one calculation contribute to the next.

Deciding when to stop

- When we use R.E. we are using it because we don't know the answer
- We must therefore decide on the number of digits of accuracy we require
  - Let nacc = number of digits of accuracy required
  - Need to consider both absolute and relative errors
- On step i, define the difference between current value and previous to be the absolute error
  - abserr=$|D_i-D_{i-1}|$
  - If (-log10(abserr).ge.real(nacc)) then done (A)

Also consider relative error

- Define the relative/fractional error by
  - fracerr=$|(D_i-D_{i-1})/D_i|$
  - If (-log10(fracerr).ge.real(nacc)) then done (B)
- Need to consider both of these conditions (A & B) at the same time
  - The relative error ensures convergence when D_i is large
  - The absolute error is tripped when D_i are small and the relative error can potentially blow-up

The standard approach

- For a linear system of n equations with n unknowns Gaussian elimination is the standard technique

$$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$$
$$a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2$$

$$a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n$$

Or in matrix form:

$$\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ & & \\ a_{n1} & a_{n2} & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \\ b_n \end{bmatrix}$$

Short form:

$$Ax = b$$

As we all know G.E. is an effective but time consuming algorithm – applying the algorithm on an $n \times n$ matrix takes $\sim n^3$ operations ($O(n^3)$).

While we can't change the order of the calculation we can reduce the overall number of calculations.

The *LU* Decomposition method

- A non-singular matrix can be written as a product of lower and upper diagonal matrices

$$A = LU \qquad \text{where,}$$

$$L = \begin{bmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ & & \\ l_{n1} & l_{n2} & l_{nn} \end{bmatrix} \qquad U = \begin{bmatrix} 1 & u_{12} & u_{1n} \\ 0 & 1 & u_{2n} \\ & & \\ 0 & 0 & 1 \end{bmatrix}$$

- The simple nature of *L* & *U* means that it is actually fairly straightforward to relate the values to *A*
  - Then can find the l_{ij} and u_{ij} fairly easily

By examination

- If $A=LU$, multiply to get

$$\begin{bmatrix} a_{11} & a_{12} & a_{1n} \\ a_{21} & a_{22} & a_{2n} \\ & & \\ a_{n1} & a_{n2} & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{1n} \\ l_{21} & l_{21}u_{12}+l_{22} & l_{21}u_{1n}+l_{22}u_{2n} \\ & & \\ l_{n1} & l_{n1}u_{12}+l_{n2} & \sum_{i=1}^{n-1} l_{ni}u_{in}+l_{nn} \end{bmatrix}$$

Equate first col:

$$a_{11}=l_{11}$$
$$a_{21}=l_{21}$$
$$\vdots$$
$$a_{n1}=l_{n1}$$

Now we know all l_{i1}, $i=1,..,n$

Equate first row:

$$a_{11}=l_{11}$$
$$a_{12}=u_{12}l_{11}$$
$$\vdots$$
$$a_{1n}=u_{1n}l_{11}$$

Use l_{11} vals to get all u_{1j} $j=2,\ldots,n$

Equate second col:

$$a_{12}=l_{11}u_{12}$$
$$a_{22}=l_{21}u_{12}+l_{22}$$
$$\vdots$$
$$a_{n2}=l_{n1}u_{12}+l_{n2}$$

Use previous steps to get l_{j2} $i=2,\ldots,n$

Alternate from columns to rows

- Thus expanding *each column of A* allows us to equate a *column of values in L*
- Similarly, expanding a *row of A* allows to equate to a *row of values in U*
- However, most rows will be combinations of U and L entries and we must *alternate* between columns and rows to ensure each expression contains only 1 unknown

General Formulae (Exercise)

- Provided we consider column values on or below the diagonal then we have for the *k*-th column

$$a_{ik} = l_{ik} + \sum_{j=1}^{k-1} l_{ij} u_{jk} \Rightarrow l_{ik} = a_{ik} - \sum_{j=1}^{k-1} l_{ij} u_{jk} \quad \text{where} \quad i = k, k+1, \ldots, n$$

- Similarly, for row values to the right of the main diagonal then we have for the *k*-th row

$$a_{kj} = u_{kj} l_{kk} + \sum_{m=1}^{k-1} l_{km} u_{mj} \Rightarrow u_{kj} = \frac{1}{l_{kk}} \left[a_{kj} - \sum_{m=1}^{k-1} l_{km} u_{mj} \right] \quad \text{where} \quad j = k+1, k+2, \ldots, n$$

- Convince yourself of these two relationships…

Utilizing the LU decomposition

Since $A\vec{x} = \vec{b} \quad \Rightarrow \quad LU\vec{x} = \vec{b}$

Define $\vec{z} = U\vec{x}$ (4), then we have $L\vec{z} = \vec{b}$ (5)

- We need to solve eqn (5) for z first
- Once z is known we can then solve eqn (4) for x
- Sounds like a lot of work! Why is this useful?
  - The triangular matrices are already in a Gauss-Jordan eliminated form so *both can be solved by substitution*

Consider solving $L\vec{z} = \vec{b}$

- If we write out the equations:

$$l_{11} z_1 = b_1 \quad \Rightarrow \quad z_1 = b_1 / l_{11}$$

$$l_{21} z_1 + l_{22} z_2 = b_2 \quad \Rightarrow \quad z_2 = \frac{1}{l_{22}} \left(b_2 - l_{21} z_1 \right)$$

$$l_{n1} z_1 + l_{n2} z_2 + \ldots + l_{nn} z_n = b_n$$

- Which is solved via "forward substitution" – exactly analogous to the back substitution we saw for triadiagonal solvers

$$z_i = \frac{1}{l_{ii}} \left[b_i - \sum_{k=1}^{i-1} l_{ik} z_k \right] \qquad i = 2, \ldots, n$$

Next solve $U\vec{x} = \vec{z}$

- Again if we right out the equations then:

$$x_1 + u_{12}x_2 + u_{13}x_3 + \ldots + u_{1n}x_n = z_1$$

$$x_2 + u_{23}x_3 + \ldots + u_{2n}x_n = z_2$$

$$x_{n-1} + u_{n-1,n}x_n = z_{n-1}$$

$$x_n = z_n$$

- We solve by back substitution again:

$x_n = z_n$ then substitute into $x_{n-1} + u_{n-1,n}x_n = z_{n-1}$

to get $x_{n-1} = z_{n-1} - u_{n-1,n}x_n$ and in general

$$x_{n-i} = z_{n-i} - \sum_{k=n-i+1}^{n} u_{n-i,k}x_k \qquad \text{where } i = 1,\ldots,n\text{-}1$$

Summary

- Cubic spline interpolation gives a smooth piecewise interpolation scheme but requires conditions to be set for the end points
- Fitting cubic splines involves solving a triadiagonal system
  - Tridiagonal solvers are efficient, and use back substitution to calculate solutions
- The accuracy of finite difference approximations to derivatives can be greatly improved using Richardson Extrapolation
  - R.E. relies upon cancelling the leading order error in any expansion

Next Lecture

- Least squares polynomial fitting

Homework 8: 04/17/2019

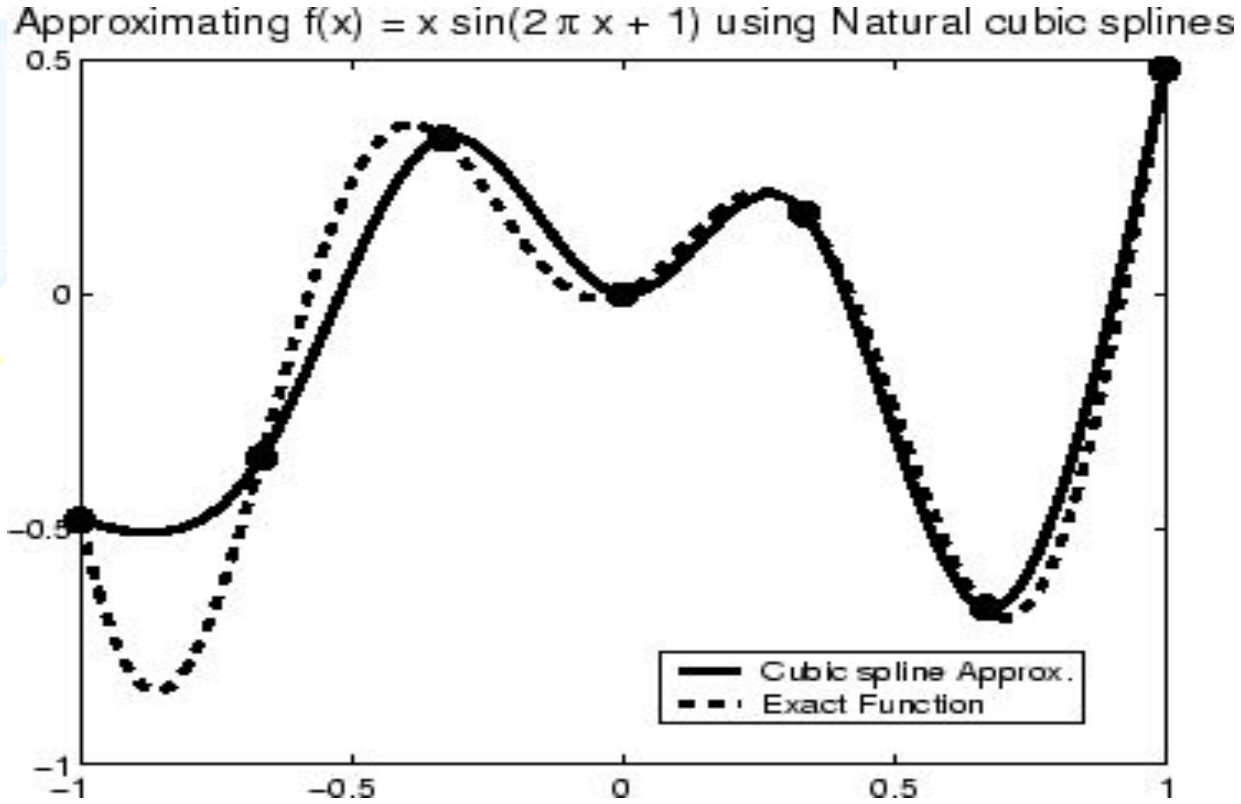
Problem 1: Given a set of seven data points for a voltage-current characteristic of a zener diode

Voltage	*-1.00*	*0.00*	*1.27*	*2.55*	*3.82*	*4.92*	*5.02*
Current	*-14.58*	*0.00*	*0.00*	*0.00*	*0.00*	*0.88*	*11.17*

In Homework 7, we approximate it using Lagrange interpolation. Here we redo it using natural cubic spline and clamped cubic spline, can we still see the "swings" between the data points.

Problem 2: For $f(x) = x\sin(2\pi x + 1)$

Please produce seven equidistantly spaced points in [-1,1], and Approximate f(x) by using natural cubic spline. On same picture, plot this fit and exact function. Redo for 9,17,19,21 nodes



Problem 3: For $f(x) = x\sin(2\pi x + 1)$

Please produce seven equidistantly spaced points in [-1,1], and Approximate f(x) by using clamped cubic spline. On same picture, plot this fit and exact function.Redo for 9,17,19,21 nodes