Polynomial Interpolation

Larry Caretto
Mechanical Engineering 309

Numerical Analysis of Engineering Systems

March 12-17, 2014

California State University Northridge

Outline

- Review numerical differentiation
- Unique nth order polynomial for n+1 data points, regardless of form
- Exercise on Newton Polynomials and Divided Difference Table
- Lagrange Interpolation
- Cubic splines
- APPENDIX: Thomas algorithm for solving tridiagonal matrix equations

California State University Northridge 2

Review Numerical Differentiation

- Finite difference expressions for derivatives derived from Taylor series
- Derivative = (Finite difference expression) + (Truncation Error) *Use this* *Information only*
- Use uniform grid with points $x_i = x_0 + ih$
  - $f_i = f(x_i)$ and h = step size = $(x_N - x_0)/N$
  - Have N + 1 points from x_0 to x_N
- Can derive expressions for first, second, third, etc. derivatives

California State University Northridge 3

Review Numerical Differentiation II

- Classify finite-difference expressions as forward, backwards and central
  - Forward difference: data for derivative at $x = x_i$ come from points at $x \geq x_i$
  - Backwards difference: data for derivative at $x = x_i$ come from points at $x \leq x_i$
  - Central difference: data for derivative at $x = x_i$ come from points at $x_{i-k} \leq x \geq x_{i+k}$
- Order of the error: truncation error proportional to h^n written as $O(h^n)$

California State University Northridge 4

Review Derivative Expressions

$$f_i' = \frac{f_{i+1} - f_i}{h} + O(h) \qquad f_i' = \frac{f_i - f_{i-1}}{h} + O(h)$$

$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2) \qquad f_i' = \frac{f_{i-2} - 4f_{i-1} + 3f_i}{2h} + O(h^2)$$

$$f_i' = \frac{-f_{i+2} + 4f_{i+1} - 3f_i}{2h} + O(h^2)$$

$$f_i'' = \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} + O(h^2)$$

What are (1) order of derivative, (2) order of error, and (3) direction (forward, central, or backward)?

Central differences easiest to use for a given order of error (if data available)

California State University Northridge 5

In-Class Exercise

- What is order of derivative, order of error and direction of expression below?

$$f_i'' = \frac{-f_{i+2} + 16f_{i+1} - 30f_i + 16f_{i-1} - f_{i-2}}{12h^2} + O(h^4)$$

– Central difference expression for the second derivative, with fourth-order error

t(s)	s(m)
0.7	0.644
0.8	0.717
0.9	0.783
1.0	0.841
1.1	0.891
1.2	0.932
1.3	0.964
1.4	0.985
1.5	0.997

- Find acceleration at t = 1 s for position (s in meters) data

$$s_i'' = \frac{-0.932 + 16(0.891) - 30(0.841) + 16(0.783) - 0.717}{12(0.1^2)}$$

s'' = –0.792 m/s²

California State University Northridge 6

Sum of Coefficients

- Why is the sum of the coefficients in each derivative expression is zero?

$$f_i' = \frac{f_{i+1} - f_i}{h} + O(h) \qquad f_i' = \frac{f_{i-2} - 4f_{i-1} + 3f_i}{2h} + O(h^2)$$

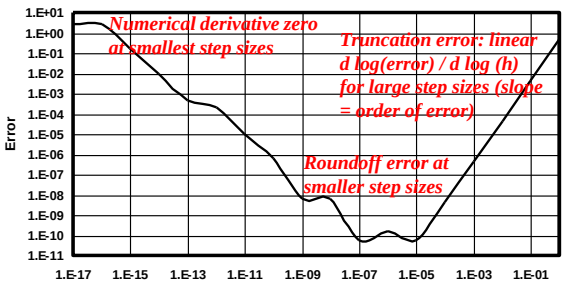
$$f_i' = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2) \qquad f_i' = \frac{-f_{i+2} + 4f_{i+1} - 3f_i}{2h} + O(h^2)$$

$$f_i' = \frac{f_i - f_{i-1}}{h} + O(h) \qquad f_i'' = \frac{f_{i+1} + f_{i-1} - 2f_i}{h^2} + O(h^2)$$

$$f_i'' = \frac{-f_{i+2} + 16f_{i+1} - 30f_i + 16f_{i-1} - f_{i-2}}{12h^2} + O(h^4)$$

- If all f_i are the same any derivative is zero

Figure 2-1. Effect of Step Size on Error



Review Interpolation

x_i	y_i
0	0
10	10
20	30
30	60
40	100
50	150

- Example problem: What is the value of y when x = 34?
- Can use different numbers of x-y data pairs from 2 to 6
- For polynomial p(x), the basic rule is that $p(x_i) = y_i$
  - E.g. p(20) = 30
  - What is p(40)? p(40) = 100

Uniqueness of Polynomial

- With n+1 data points we can fit a polynomial of order n
- This polynomial is unique
- It can take different forms, second order polynomials as an example
  - $y = a + bx + cx^2 = (cx + b)x + a$
  - Newton: $y = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$
  - Lagrange:

$$p(x) = y_1 \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} + y_2 \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} + y_3 \frac{(x - x_1)(x - x_2)}{(x_3 - x_2)(x_3 - x_1)}$$

Example

- $y = 3 + 2x + x^2$ has values y = 3 at x = 0, y = 6 at x = 1 and y = 11 at x = 2
- $y = 3 + 3(x - 0) + 1(x - 0)(x - 1)$ has same values as does polynomial below

$$y = 3\frac{(x-1)(x-2)}{(0-1)(0-2)} + 6\frac{(x-0)(x-2)}{(1-0)(1-2)} + 11\frac{(x-0)(x-1)}{(2-1)(2-0)}$$

- Regardless of the form we use there is only one (unique) n^{th} order polynomial between n + 1 data points

Review Newton Polynomials

- $p_0(x) = a_0$
- $p_1(x) = a_0 + a_1(x - x_0)$
- $p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$
- $p_3(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2)$
- $p_4(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + a_3(x - x_0)(x - x_1)(x - x_2) + a_4(x - x_0)(x - x_1)(x - x_2)(x - x_3)(x - x_4)$
- $p_n(x) = p_{n-1}(x) + a_n \prod_{k=0}^{n-1}(x - x_k)$

```
Function newtonPoly(a() As Double, x() As _
 Double, xFit As Double, n As Long) As Double

 'Evaluates Newton polynomial of order n
  Dim product As Double    'Ongoing product
  Dim sum As Double        'Ongoing sum
  Dim k As Long            'Loop counter

  sum = a(0)
  product = 1
  For k = 1 To n
    product = product * (xFit - x(k-1))
    sum = sum + a(k) * product
  Next k
  newtonPoly = sum
End Function
```

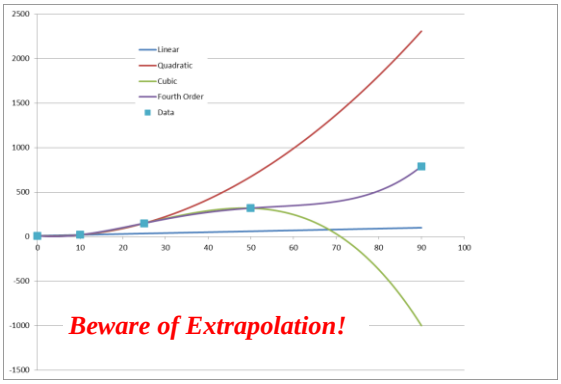
California State University Northridge

13

Review Divided Difference Table

- Start with x and y data in first two columns
- Take differences of previous columns divided by delta-x terms
- Shaded cells show coefficients in Newton polynomials

0	y_0=10	←a_0		
		$F_0=\frac{y_1-y_0}{x_1-x_0}$	←a_1	
10	y_1=20		$S_0=\frac{F_1-F_0}{x_2-x_0}$	←
		$F_1=\frac{y_2-y_1}{x_2-x_1}$		T_0
25	y_2=150		$S_1=\frac{F_2-F_1}{x_3-x_1}$	
		$F_2=\frac{y_3-y_2}{x_3-x_2}$		T
50	y_3=320		$S_2=\frac{F_3-F_2}{x_4-x_2}$	
		$F_3=\frac{y_4-y_3}{x_4-x_3}$		
90	y_4=790			



15

Sample Divided Difference Table

0	y_0=10	←a_0			
		$F_0=\frac{y_1-y_0}{x_1-x_0}$	←a_1		
10	y_1=20		$S_0=\frac{F_1-F_0}{x_2-x_0}$	←a_2	
		$F_1=\frac{y_2-y_1}{x_2-x_1}$		$T_0=\frac{S_1-S_0}{x_3-x_0}$	←a_3
25	y_2=150		$S_1=\frac{F_2-F_1}{x_3-x_1}$		$R_0=\frac{T_1-T_0}{x_4-x_0}$
		$F_2=\frac{y_3-y_2}{x_3-x_2}$		$T_1=\frac{S_2-S_1}{x_4-x_1}$	↑a_4
50	y_3=320		$S_2=\frac{F_3-F_2}{x_4-x_2}$		
		$F_3=\frac{y_4-y_3}{x_4-x_3}$			
90	y_4=790				

Follow same a_k pattern for other starting points

Divided difference table gives polynomial coefficients (next page)

16

Newton Polynomials from x_{start}

- $p_0(x) = a_0$ and $p_1(x) = a_0 + a_1(x - x_{start})$
- $p_2(x) = a_0 + a_1(x - x_{start}) + a_2(x - x_{start})\ (x - x_{start+1})$
- $p_3(x) = a_0 + a_1(x - x_{start}) + a_2(x - x_{start})\ (x - x_{start+1}) + a_3(x - x_{start})\ (x - x_{start+1})(x - x_{start+2})$
- $a_0 = y_{start}$, $a_1 = F_{start} = (y_{start+1} - y_{start}) / (x_{start+1} - x_{start})$. $a_2 = S_{start} = (F_{start+1} - \ldots$

$$p_n(x)=\sum_{m=0}^{n-1} a_m \prod_{k=0}^{m-1}(x-x_{start+k}) \qquad p_n(x)=p_{n-1}(x)+a_n\prod_{k=0}^{n-1}(x-x_{start+k})$$

California State University Northridge

17

General Use of Table

- Start at any point in divided-difference table to get different interpolations
  - Corresponds to using different set of data points and different order for fit
- Generally want to get interpolated value at a point that is in the middle of n+1 data points for n<u>th</u>-order polynomial
- What x points would you use for second order (quadratic) interpolation at x = 30 if you had data at x = [0 10 25 50 90]?

California State University Northridge

10 25 50

18

Divided Difference Exercise

0	y_0=10				
		$F_0 = \frac{y_1 - y_0}{x_1 - x_0}$			
10	y_1=20		$S_0 = \frac{F_1 - F_0}{x_2 - x_0}$		
		$F_1 = \frac{y_2 - y_1}{x_2 - x_1}$		$T_0 = \frac{S_1 - S_0}{x_3 - x_0}$	
25	y_2=150		$S_1 = \frac{F_2 - F_1}{x_3 - x_1}$		$R_0 = \frac{T_1 - T_0}{x_4 - x_0}$
		$F_2 = \frac{y_3 - y_2}{x_3 - x_2}$		$T_1 = \frac{S_2 - S_1}{x_4 - x_1}$	
50	y_3=320		$S_2 = \frac{F_3 - F_2}{x_4 - x_2}$		
		$F_3 = \frac{y_4 - y_3}{x_4 - x_3}$			
90	y_4=790				

Compute divided-difference table for quadratic polynomial through points at x = 10, 25, 50

Use divided-difference table to generate the quadratic polynomial

19

Divided Difference Table for Solution

0	y_0=10				
		$F_0 = \frac{y_1 - y_0}{x_1 - x_0}$			
10	y_1=20	←a_0	$S_0 = \frac{F_1 - F_0}{x_2 - x_0}$		
		$F_1 = \frac{y_2 - y_1}{x_2 - x_1}$ ←a_1		$T_0 = \frac{S_1 - S_0}{x_3 - x_0}$	
25	y_2=150		$S_1 = \frac{F_2 - F_1}{x_3 - x_1}$ ←a_2		$R_0 = \frac{T_1 - T_0}{x_4 - x_0}$
		$F_2 = \frac{y_3 - y_2}{x_3 - x_2}$		$T_1 = \frac{S_2 - S_1}{x_4 - x_1}$	
50	y_3=320		$S_2 = \frac{F_3 - F_2}{x_4 - x_2}$		
		$F_3 =$			
90	y_4=790				

$$F_1 = \frac{150-20}{25-10} = \frac{130}{15} = \frac{26}{3}$$

$$F_2 = \frac{320-150}{50-25} = \frac{170}{25} = \frac{34}{5}$$

$$S_1 = \frac{\frac{34}{5} - \frac{26}{3}}{50-10} = \frac{\frac{102}{15} - \frac{130}{15}}{50-10} = \frac{-28}{(15)(40)} = -\frac{7}{(150)}$$

Quadratic Polynomial

- Usual form for initial point at x = x_0
  - $p_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$
- For starting at x = x_1
  - $p_2(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2)$
  - Using data from previous chart gives

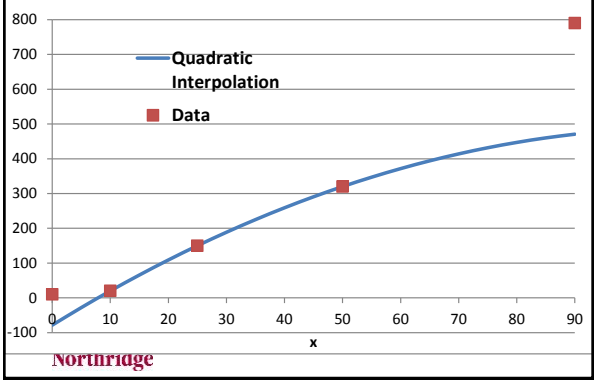
$$p_2(x) = 20 + \frac{26}{3}(x-10) - \frac{7}{150}(x-10)(x-25)$$

  - This gives $p(x_k) = y_k$ for x_k = 10, 25, and 50
  - Interpolated value for x = 30 is 189

California State University Northridge

21

Quadratic Interpolation *versus* Data



Northridge

Lagrange Interpolation

- Alternative form to write interpolation polynomial without need for detailed coefficient computations
- Linear polynomial with two points

$$p(x) = y_{i-1}\frac{x - x_i}{x_{i-1} - x_i} + y_i\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

- Can you see that this is a linear polynomial with $p(x_{i-1}) = y_{i-1}$ and $p(x_i) = y_i$

California State University Northridge

23

Lagrange Interpolation II

- Quadratic polynomial with three points

$$p(x) = y_{i-1}\frac{(x - x_i)(x - x_{i+1})}{(x_{i-1} - x_i)(x_{i-1} - x_{i+1})} + y_i\frac{(x - x_{i-1})(x - x_{i+1})}{(x_i - x_{i-1})(x_i - x_{i+1})} + y_{i+1}\frac{(x - x_{i-1})(x - x_i)}{(x_{i+1} - x_i)(x_{i+1} - x_{i-1})}$$

- Can you see that this is a quadratic polynomial with $p(x_{i-1}) = y_{i-1}$, $p(x_i) = y_i$, and $p(x_{i+1}) = y_{i+1}$

California State University Northridge

24

General Lagrange Polynomial

- For n+1 data points from (x_0, y_0) to (x_n, y_n) we have the following n^{th} order polynomial

$$p(x)=\sum_{i=0}^{n}\left\{\frac{y_i\prod_{j=1,j\neq i}^{n}(x-x_j)}{\prod_{j=1,j\neq i}^{n}(x_i-x_j)}\right\}$$

- At any data point, x = x_k, p(x_k) = y_k

California State University Northridge 25

Splines

- **Definition:** a piecewise polynomial where each piece is a low order polynomial
- **Continuity:** Depending on the order of the spline one can maintain continuity of the piecewise polynomial and one or more of its derivatives
  - Linear spline: continuity of polynomials
  - Quadratic spline: continuity of first derivative
  - Cubic spline: continuity of second derivative

California State University Northridge 26

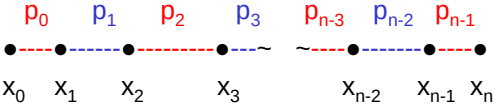
Cubic Splines

- Have n+1 data points from (x_0, y_0) to (x_n, y_n) sometimes called "knots"
- Between each pair of knots, (x_k, y_k) to (x_{k+1}, y_{k+1}), we pass a cubic polynomial
  - $p_k(x) = a_k + b_k x + c_k x^2 + d_k x^3 \quad k = 0,\ldots,n-1$
- This set of n polynomials has a total of 4n coefficients (a_0 to a_{n-1}, b_0 to b_{n-1}, c_0 to c_{n-1}, and d_0 to d_{n-1}) that we have to determine
  - We need 4n equations to determine these coefficients

California State University Northridge 27

Knots and Polynomials

- Diagram of n+1 knots and n polynomials



- Each point (knot) has known x_k and y_k
  - At end points $p_0(x_0) = y_0$ and $p_{n-1}(x_n) = y_n$
  - At each interior point $p_k(x_k) = p_{k+1}(x_k) = y_k$
  - Have continuity of first and second derivatives $p'_k(x_k) = p'_{k+1}(x_k)$ and $p''_k(x_k) = p''_{k+1}(x_k)$ at knots

California State University Northridge 28

Conditions at Knots

- Piecewise polynomial must be continuous at each internal knot from x_1 to x_{n-1} and must match the y value at the knot
  - This means $p_k(x_k) = p_{k+1}(x_k) = y_k$

$$y_k = p_k(x_k) = a_k + b_k x_k + c_k x_k^2 + d_k x_k^3$$
$$y_k = p_{k+1}(x_k) = a_{k+1} + b_{k+1} x_k + c_{k+1} x_k^2 + d_{k+1} x_k^3$$

- This applies from k = 1 to k = n – 1 giving us 2n – 2 equations

California State University Northridge 29

Conditions at Knots II

- At the ends of the interval, the polynomial values must match the data
  - This gives two more equations for a total of 2n – 2 + 2 = 2n

$$y_0 = p_1(x_0) = a_1 + b_1 x_0 + c_1 x_0^2 + d_1 x_0^3$$
$$y_n = p_n(x_n) = a_n + b_n x_n + c_n x_n^2 + d_n x_n^3$$

- At the interior knots we want to have continuity of first and second derivatives
  - We have no data to match, just continuity

California State University Northridge 30

Continuity at Knots III

- n – 1 first derivative continuity conditions at each interior knot (i = 1, …, n – 1)
  - $dp_k/dx = b_k + 2c_kx + 3d_kx^2$ at any x

$$b_k + 2c_k x_k + 3d_k x_k^2 = b_{k+1} + 2c_{k+1} x_k + 3d_{k+1} x_k^2$$

- n – 1 second derivative continuity conditions at internal knots (i = 1, …, n – 1)
  - $d^2p_k/dx^2 = 2c_k + 6d_kx$ at any x

$$2c_k + 6d_k x_k = 2c_{k+1} + 6d_{k+1} x_k$$

Counting Equations

- Have 2n + (n – 1) + (n – 1) = 4n – 2 equations to determine 4n coefficients
- Need some approximations to get an additional two equations
- Need to develop a solution procedure to solve for all coefficients necessary to have n splines for our n – 1 data points
  - See Rao, pp 394-395 for derivation of equations to solve for spline equations in terms of second derivatives

Overview of Solution Process

- Know that each polynomial has second derivative given by $d^2p_k/dx^2 = 2c_k + 6d_kx$
- Shows that second derivative varies linearly with x between knots
- Use linear interpolation for second derivative to derive a set of equations for unknown second derivatives, $p_k'' = \sigma_k$
  - Solve these equations for σ_k then use these values to reformulate the cubic spline equations

Solving for the σ_k

- Define $h_k = x_{k+1} - x_k$ as the variable step size between x values (k = 0, … n – 1)
- Define $\Delta y_k = (y_{k+1} - y_k)/h_k$
- Derived equations for σ_k

$$h_{k-1}\sigma_{k-1} + 2(h_{k-1} + h_k)\sigma_k + h_k\sigma_{k+1} = 6[\Delta y_k - \Delta y_{k-1}]$$

- This is known as a tridiagonal set of equations because, when written as a matrix only three diagonals have nonzero terms

Tridiagonal Equations for σ_k

- Example of σ_k matrix equations (for n+1 = 8) before adding two other equations

$$\begin{bmatrix} h_0 & 2(h_0+h_1) & h_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & h_3 & 2(h_3+h_4) & h_4 & 0 & 0 \\ 0 & 0 & 0 & 0 & h_4 & 2(h_4+h_5) & h_5 & 0 \\ 0 & 0 & 0 & 0 & 0 & h_5 & 2(h_5+h_6) & h_6 \end{bmatrix} \begin{bmatrix} \sigma_0 \\ \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \end{bmatrix} = \begin{bmatrix} 6(\Delta y_1 - \Delta y_0) \\ 6(\Delta y_2 - \Delta y_1) \\ 6(\Delta y_3 - \Delta y_2) \\ 6(\Delta y_4 - \Delta y_3) \\ 6(\Delta y_5 - \Delta y_4) \\ 6(\Delta y_6 - \Delta y_5) \end{bmatrix}$$

- Have six equations in eight unknowns
  - Assume some end conditions to get equations for $\sigma_0 = f(\sigma_0,\sigma_1)$ and $\sigma_7 = f(\sigma_5,\sigma_6)$

Assuming Extra Equations

- Use assumptions about σ_0 and σ_n
  - Assumptions for σ_0 and σ_n can be different
  1. Assume a value (typically $\sigma = 0$, called the "natural" boundary condition)
  2. Assume constant ($\sigma_0 = \sigma_1$; $\sigma_{n-1} = \sigma_n$)
  3. Assume linear behavior resulting in following equations at either end

$$\sigma_0 = \frac{1}{h_1}\left[(h_0 + h_1)\sigma_1 - h_0\sigma_2\right]$$

$$\sigma_n = \frac{1}{h_{n-2}}\left[(h_{n-2} + h_{n-1})\sigma_{n-1} - h_{n-1}\sigma_{n-2}\right]$$

Assuming Extra Equations II

4. Specify the value of p'(x) at the endpoint which gives the following results

$$\sigma_0 = \frac{3}{h_0}\left[\Delta y_0 - p'(x_0)_{assumed}\right] - \frac{\sigma_1}{2}$$

$$\sigma_n = \frac{1}{h_{n-1}}\left[p'(x_n)_{assumed}\sigma_{n-1} - \Delta y_{n-1}\right] - \frac{\sigma_{n-1}}{2}$$

- Assumptions gives equations for σ_0 and σ_n that can be used to replace these variables in tridiagonal matrix equations
- Reduces solution for σ_k to n – 1 equations in n – 1 unknowns

Assuming Extra Equations III

3. Not-a-knot condition assumes continuity of third derivative at second and second-from-last data point: $p_1'''(x_1) = p_2'''(x_1) \quad p_n'''(x_{n-1}) = p_{n-1}'''(x_{n-1})$
   - Same as linear σ vs. x behavior
   - Gives same equations for σ_0 and σ_n

$$-\sigma_0 + \left(1 + \frac{h_0}{h_1}\right)\sigma_1 - \frac{h_0}{h_1}\sigma_2 = 0$$

$$\frac{h_{n-1}}{h_{n-2}}\sigma_{n-2} - \left(1 + \frac{h_{n-1}}{h_{n-2}}\right)\sigma_{n-1} + \sigma_n = 0$$

Assuming Extra Equations IV

- Not-a-knot Equations

$$-\sigma_0 + \left(1 + \frac{h_0}{h_1}\right)\sigma_1 - \frac{h_0}{h_1}\sigma_2 = 0$$

$$\frac{h_{n-1}}{h_{n-2}}\sigma_{n-2} - \left(1 + \frac{h_{n-1}}{h_{n-2}}\right)\sigma_{n-1} + \sigma_n = 0$$

- Linear σ vs. x Equations

$$\sigma_0 = \frac{1}{h_1}\left[(h_0 + h_1)\sigma_1 - h_0\sigma_2\right]$$

$$\sigma_n = \frac{1}{h_{n-2}}\left[(h_{n-2} + h_{n-1})\sigma_{n-1} - h_{n-1}\sigma_{n-2}\right]$$

- Common Form

$$-h_1\sigma_0 + (h_1 + h_0)\sigma_1 - h_0\sigma_2 = 0$$

$$h_{n-1}\sigma_{n-2} - (h_{n-2} + h_{n-1})\sigma_{n-1} + h_{n-2}\sigma_n = 0$$

Modifying Equations I

1. Assume a value for the second derivative, $\sigma_{Assumed}$ (often $\sigma_{Assumed} = 0$)
   - First equation with σ_0

$$h_0\sigma_0 + 2(h_0 + h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

   - If $\sigma_0 = \sigma_{0,Assumed}$ first equation becomes

$$2(h_0 + h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right] - h_0\sigma_{0,Assumed}$$

   - Similarly with $\sigma_n = \sigma_{n,Assumed}$ the last equation becomes

$$h_{n-2}\sigma_{n-2} + 2(h_{n-2} + h_{n-1})\sigma_{n-1} = 6\left[\Delta y_{n-1} - \Delta y_{n-2}\right] - h_{n-1}\sigma_{n,Assumed}$$

Modifying Equations II

2. Assume second derivative is constant at endpoints so that $\sigma_0 = \sigma_1$ or $\sigma_n = \sigma_{n-1}$
   - First equation before and after substituting $\sigma_0 = \sigma_1$ into first equation

$$h_0\sigma_0 + 2(h_0 + h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

$$(3h_0 + 2h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

   - Similarly with $\sigma_n = \sigma_{n-1}$ last equation is

$$h_{n-2}\sigma_{n-2} + (2h_{n-2} + 3h_{n-1})\sigma_{n-1} = 6\left[\Delta y_{n-1} - \Delta y_{n-2}\right]$$

Modifying Equations III

3. Not-a-knot: second derivative is linear near the endpoints $\sigma_0 = \frac{1}{h_1}\left[(h_0 + h_1)\sigma_1 - h_0\sigma_2\right]$
   - First equation before and after substituting $\sigma_0 = \sigma_1$ into this equation

$$h_0\sigma_0 + 2(h_0 + h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

$$\left(2 + \frac{h_0}{h_1}\right)(h_0 + h_1)\sigma_1 + \left(h_1 - \frac{h_0^2}{h_1}\right)\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

   - Last equation with same assumption is

$$\left(h_{n-2} - \frac{h_{n-1}^2}{h_{n-2}}\right)\sigma_{n-2} + \left(2 + \frac{h_{n-1}}{h_{n-2}}\right)(h_{n-2} + h_{n-1})\sigma_{n-1} = 6\left[\Delta y_{n-1} - \Delta y_{n-2}\right]$$

Modifying Equations IV

4. Specify first derivative $\sigma_0 = \frac{3}{h_0}\left[\Delta y_0 - p'_{0,assumed}\right] - \frac{\sigma_1}{2}$
   - First equation shown below before and after substitution

$$h_0\sigma_0 + 2(h_0 + h_1)\sigma_1 + h_1\sigma_2 = 6\left[\Delta y_1 - \Delta y_0\right]$$

$$\left(\frac{3}{2}h_0 + 2h_1\right)\sigma_1 + h_1\sigma_2 = 6\Delta y_1 - 9\Delta y_0 + 3p'_{0,assumed}$$

   - Similar operation on final equation gives

$$h_{n-2}\sigma_{n-2} + \left(2h_{n-2} + \frac{3h_{n-1}}{2}\right)\sigma_{n-1} = 9\Delta y_{n-1} - 6\Delta y_{n-2} - 3p'_{n,assumed}$$

Tridiagonal Equations for σ_k

- Example of σ_k matrix equations ***after*** adding two missing equations

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & 0 & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & 0 & 0 \\ 0 & h_2 & 2(h_2+h_3) & h_3 & 0 & 0 \\ 0 & 0 & h_3 & 2(h_3+h_4) & h_4 & 0 \\ 0 & 0 & 0 & h_4 & 2(h_4+h_5) & h_5 \\ 0 & 0 & 0 & 0 & A_6 & B_6 \end{bmatrix} \begin{bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \end{bmatrix} = \begin{bmatrix} D_1 \\ 6(\Delta y_2 - \Delta y_1) \\ 6(\Delta y_3 - \Delta y_2) \\ 6(\Delta y_4 - \Delta y_3) \\ 6(\Delta y_5 - \Delta y_4) \\ D_6 \end{bmatrix}$$

- B_1, C_1, D_1, $A_6 = A_{n-1}$, $B_6 = B_{n-1}$, and $D_6 = D_{n-1}$ depend on end condition assumptions

B_1, C_1, D_1, A_{n-1}, B_{n-1}, D_{n-1} Part I

Variable	Assume Second Derivative at Ends	Assume Constant Second Derivative at Ends
B_1	$2(h_0 + h_1)$	$3h_0 + 2h_1$
C_1	h_1	h_1
D_1	$6[\Delta y_1 - \Delta y_0] - h_0\sigma_{0,Assumed}$	$6[\Delta y_1 - \Delta y_0]$
A_{n-1}	h_{n-2}	h_{n-2}
B_{n-1}	$2(h_{n-2} + h_{n-1})$	$2h_{n-2} + 3h_{n-1}$
D_{n-1}	$6[\Delta y_{n-1} - \Delta y_{n-2}] - h_{n-1}\sigma_{n,Assumed}$	$6[\Delta y_{n-1} - \Delta y_{n-2}]$

B_1, C_1, D_1, A_{n-2}, B_{n-2}, D_{n-2} Part II

Variable	Not-a-knot: Linear Second Derivatives	Assumed first derivative at ends
B_1	$h_0 + 2h_1$	$3h_0/2 + 2h_1$
C_1	$h_1 - h_0$	h_1
D_1	$6[\Delta y_1 - \Delta y_0]h_1 / (h_1 + h_0)$	$6\Delta y_1 - 9\Delta y_0 + 3p'_{0,Assumed}$
A_{n-1}	$h_{n-2} - h_{n-1}$	h_{n-2}
B_{n-1}	$2h_{n-2} + h_{n-1}$	$2h_{n-2} + 3h_{n-1}/2$
D_{n-1}	$6[\Delta y_{n-1} - \Delta y_{n-2}]h_{n-2} / (h_{n-2} + h_{n-1})$	$9\Delta y_{n-1} - 6\Delta y_{n-2} - 3p'_{n,Assumed}$

Tridiagonal Matrix Equation

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 \\ \vdots & 0 & A_4 & B_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

- Solve this with Thomas algorithm
  - See appendix for derivation

Tridiagonal (Thomas) Algorithm

- Loop over all rows from k = 1 to k = N-1; for each k value compute the following

$$B_{k+1} \leftarrow B_{k+1} - A_{k+1}C_k / B_k \quad D_{k+1} \leftarrow D_{k+1} - A_{k+1}D_k / B_k$$

- Compute $x_N = D_N / B_N$
- Loop over all rows from k = N – 1 to k = 1 in reverse order. For each row, k, compute x_k from the following equation

$$x_k = \frac{D_k - C_k x_{k+1}}{B_k}$$

The Spline Polynomial

- Once the σ_k values are known we have the following equations for $p_k(x)$ to be applied between x_k and x_{k+1}

$$p_k(x) = \frac{\sigma_k}{6}\left[\frac{(x_{k+1}-x)^3}{h_k} - h_k(x_{k+1}-x)\right] + \frac{\sigma_{k+1}}{6}\left[\frac{(x-x_k)^3}{h_k} - h_k(x-x_k)\right] + y_k\frac{x_{k+1}-x}{h_k} + y_{k+1}\frac{x-x_k}{h_k}$$

Cubic splines used to program computer driven machines because there is no discontinuity in force

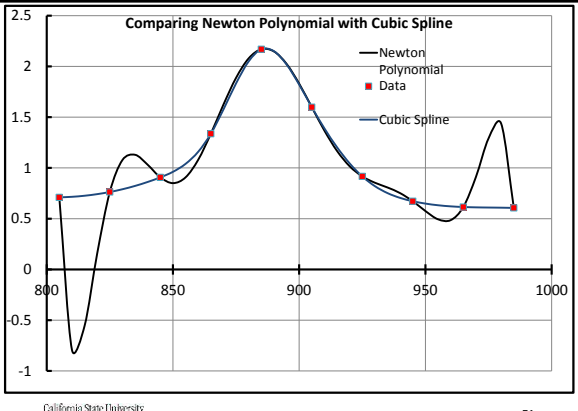
California State University Northridge 49

Spline *vs.* High Order Newton

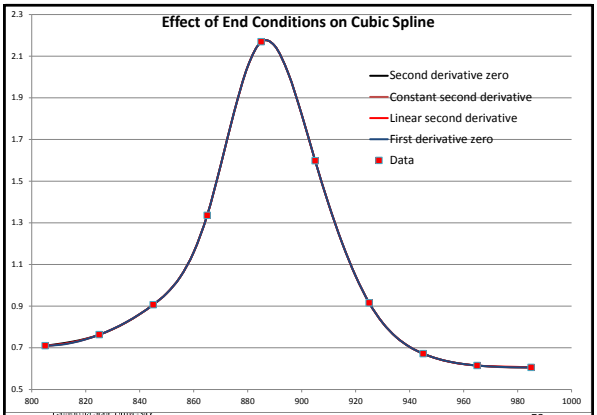
x	y
805	0.71
825	0.763
845	0.907
865	1.336
885	2.169
905	1.598
925	0.916
945	0.672
965	0.615
985	0.606

- Two fits for ten data points at left
  - Ninth order Newton polynomial
  - Cubic spline with nine different cubic formulae
- Results compared on following plot
- Which do you think is better fit?

California State University Northridge 50



California State University Northridge 51



California State University Northridge 52

APPENDIX: Thomas Algorithm

- Application of Gaussian elimination to matrix with only three diagonals
  - Called a tridiagonal matrix
  - Algorithm is sometimes called TDMA for Tridiagonal Matrix Algorithm
  - Actually applies Gaussian elimination, but results are much simpler due to simple structure of the tridiagonal matrix
  - Basic equation for one row of matrix

$$A_i x_{i-1} + B_i x_i + C_i x_{i+1} = D_i$$

California State University Northridge 53

Tridiagonal Matrix Equation

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ A_2 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 \\ \vdots & 0 & A_4 & B_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

- First and last equation have only two terms

California State University Northridge 54

Data Storage

- Tridiagonal matrix zeros are not stored
- Instead of usual A(i,j) matrix structure, only the three diagonal terms are stored
- These are stored as one-dimensional arrays: A(i), B(i), C(i)
- The right-hand-side column is stored as the D(i) array
- Apply the usual Gaussian elimination to get all zeroes below the main diagonal

California State University Northridge 55

Zeroes in Column One

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & B_2 - \frac{A_2}{B_1}C_1 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & A_3 & B_3 & C_3 & \cdots & 0 & 0 \\ \vdots & 0 & A_4 & B_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 - \frac{A_2}{B_1}D_1 \\ D_3 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

Already zero

- Changes required only for B_2 and D_2 when row 1 is the pivot row
  - Factor A_2/B_1 gives $A_2 \leftarrow A_2 - (A_2/B_1)B_1 = 0$

California State University Northridge 56

Zeroes in Column Two

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & B_3 - \frac{A_3}{B_2}C_2 & C_3 & \cdots & 0 & 0 \\ \vdots & 0 & A_4 & B_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 - \frac{A_3}{B_2}D_2 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

Already zero

- Changes required only for B_3 and D_3 when row 2 is the pivot row
  - B_2 and D_2 changed from original values in previous step with row 1 as pivot row

California State University Northridge 57

General Formula

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & B_3 - \frac{A_3}{B_2}C_2 & C_3 & \cdots & 0 & 0 \\ \vdots & 0 & A_4 & B_4 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & A_N & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 - \frac{A_3}{B_2}D_2 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

- Tridiagonal structure changes only B_{k+1} and D_{k+1} when row k is pivot row
  - Formulas: $B_{k+1} \leftarrow B_{k+1} - \frac{A_{k+1}}{B_k}C_k \quad D_{k+1} \leftarrow D_{k+1} - \frac{A_{k+1}}{B_k}D_k$

California State University Northridge

Back Substitution

- Matrix at end of elimination steps

$$\begin{bmatrix} B_1 & C_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & B_2 & C_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & B_3 & C_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & B_{N-1} & C_{N-1} \\ 0 & 0 & 0 & \cdots & 0 & 0 & B_N \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{N-1} \\ x_N \end{bmatrix} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_N \end{bmatrix}$$

- Back-substitution formulas

$$x_N = D_N / B_N$$

$$x_k = \frac{D_k - C_k x_{k+1}}{B_k} \quad \text{for } k = N-1, N-2, \cdots 2,1$$

California State University Northridge

Thomas Algorithm

- Loop over all rows from k = 1 to k = N-1; for each k value compute the following

$$B_{k+1} \leftarrow B_{k+1} - A_{k+1}C_k / B_k \quad D_{k+1} \leftarrow D_{k+1} - A_{k+1}D_k / B_k$$

- Compute $x_N = D_N / B_N$
- Loop over all rows from k = N – 1 to k = 1 in reverse order. For each row, k, compute x_k from the following equation

$$x_k = \frac{D_k - C_k x_{k+1}}{B_k}$$

California State University Northridge 60

Assignment Three Grades

- Number of students: 20
- Maximum possible score: 40
- Mean score: 28.2
- Median score: 28.5
- Standard deviation: 7.67
- Distribution of scores:

```
13 13 19 22 22 26 27
 27 27 28 29 29 29
31 34 34 37 38 38 40
```

California State University Northridge 61

Assignment 3 Comments

- Functions to solve f(x) = 0 should have the name of the functions to compute f(x) as an input arguments to the function not as part of the function
  - Need f' function name for Newton's Method
  - See solutions for correct way to do this
  - You will have to do this in later assign-ments on numerical integration and numerical solution of ordinary differential equations

California State University Northridge 62

Assignment 3 Comments II

- Codes to solve f(x) = 0 should check **both** error for convergence and excess iterations to avoid infinite loop
  - Return answer if required tolerance met before iterations exceeded
  - Return error message if tolerance not met
- Can always save workbook as macro-enabled workbook to avoid losing macros: File | Options | Save | "Save Files in this Format" (first item in menu)

California State University Northridge 63

Assignment 3 Comments III

- Do not place unneeded worksheets in your Excel workbooks
  - Use File | Options | General to set initial number of worksheets to one
- The fewer-less Rule
  - Important in formal writing, not in typical conversation
  - Use less for items that you cannot count
    - Example: I drink less coffee now.
  - Use fewer for items that you can count
    - I drank fewer cups of coffee today

California State University Northridge 64

Assignment 3 Comments IV

- Basic ideas
  - fzero and Goal Seek are like finding a root of f(x) = 0 on your calculator. They are easy to implement once you know how
  - Normally do not have to program solution of f(x) = 0 unless part of a larger program
    - Could find VBA code for doing so on line and could use fzero as part of MATLAB code
    - False position method is slow, but sure, once initial guesses are made
    - Newton's method fast, but may not converge to correct root, and requires derivative

California State University Northridge 65

Assignment 3 Comments IV

- Modify or eliminate dummy text in MATLAB functions
  - Original dummy text

```
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
```

  - Possible modification

```
%FALSEPOS False Position solution of f(x) = 0
%   This function solves f(x) = 0 using the
%   false position method; inputs are …
```

- Set error in Goal Seek in Excel Options
- Name files <lastName>_<assmt no.>

California State University Northridge 66

Interpolation Exercise

x	y
1	1.221
2	1.492
3	1.822
4	2.226
5	2.718
6	3.320
7	4.055
8	4.953
9	6.050
10	7.389

- For the data table at the left, find an estimate of the value of y at x = 6.6 using a cubic polynomial
- You may use notes, ask questions, consult with fellow students, *etc*.
- You should be prepared to solve this problem on a quiz given the form of the Newton polynomial

California State University Northridge

67

Exercise Solution

5	2.718	← a_0		
		0.6018	← a_1	
6	3.320		0.06662	← a_2
		0.7351		-0.004917
7	4.055		0.08137	↑
		0.8978		a_3
8	4.953			

- Divided difference table for cubic polynomial about x = 6.6

- $p_3(x) = a_0 + a_1(x - x_{start}) + a_2(x - x_{start})(x - x_{start+1}) + a_3(x - x_{start})(x - x_{start+1})(x - x_{start+2}) = 2.718 + 0.6018(x - 5) + 0.06662(x - 5)(x - 6) - 0.004917(x - 5)(x - 6)(x - 7) = 3.747$ for x = 6.6

California State University Northridge

68