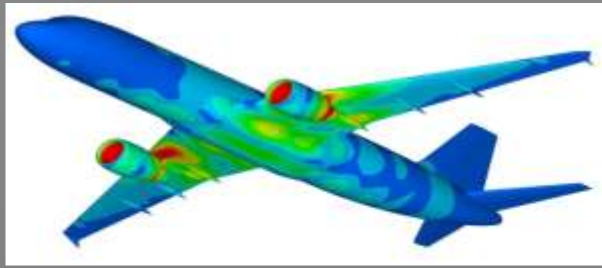
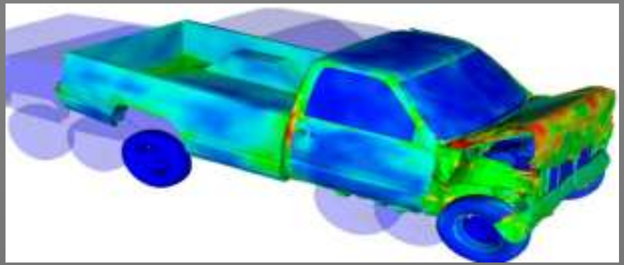
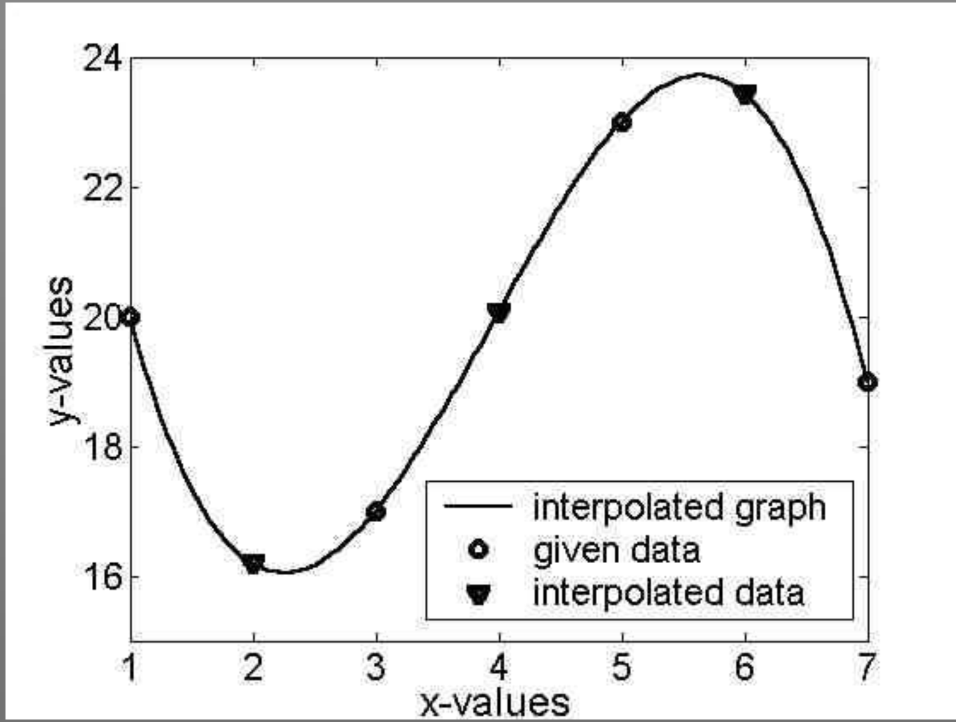
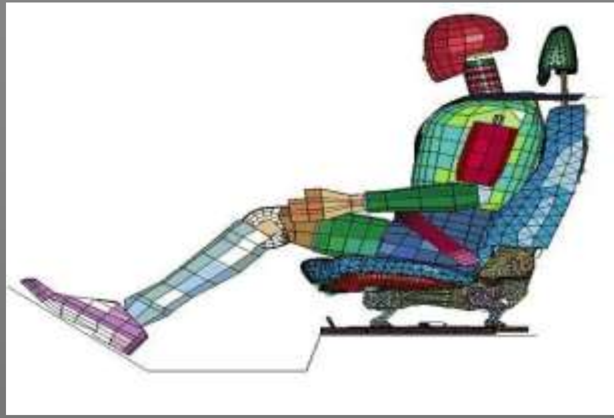
Finite Difference and Interpolation









Dr. Himanshu Pathak
himanshu@iitmandi.ac.in

Taylor Series

- Taylor series are expansions of a function f(x) by some finite distance dx to f(x+dx).
- In essence, the Taylor series provides a means to predict a function value at one point in terms of the function value and its derivatives at another point.
- In particular, the theorem states that any smooth function can be approximated as a polynomial.

$$f(x_{i+1}) \cong f(x_i)$$ 0th order Approximation

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$ 1st order Approximation

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2$$ 2nd order Approximation

$$\boxed{f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{n}(x_i)}{n!}h^n + R_n}$$ Nth order Approximation

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$ Remainder

Taylor Series

Example: Use zero- through fourth-order Taylor series expansions to approximate the function. $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$

$x_i = 0$ with h = 1. That is, predict the function's value at $x_{i+1} = 1$

By Taylor Series

0^{th} order Approximation

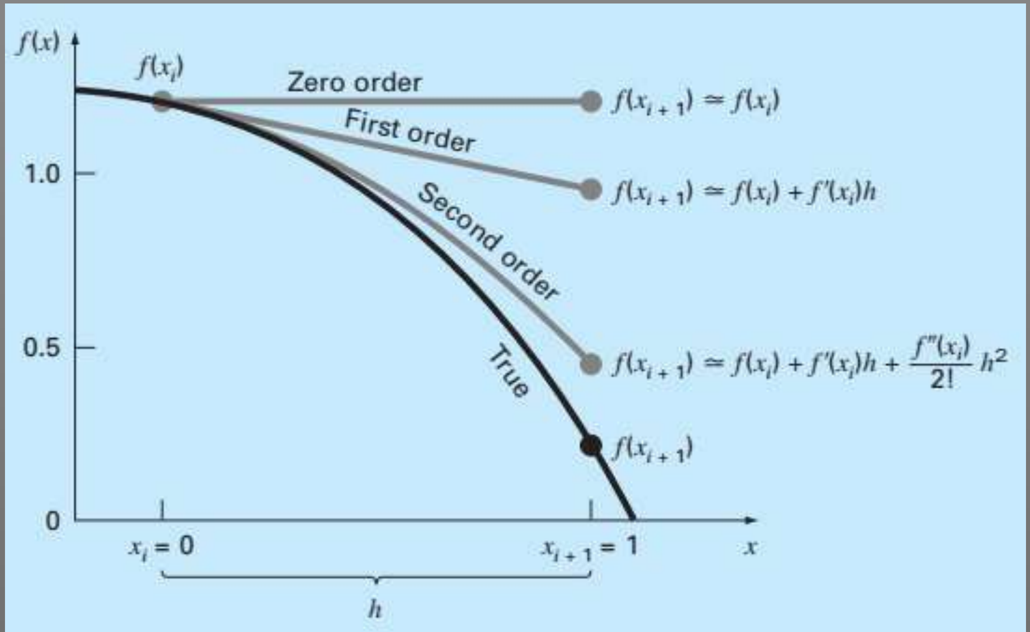
$$f(x_{i+1}) \cong f(x_i)$$ $$f(x_{i+1}) \simeq 1.2$$

1^{st} order Approximation

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i)$$ $$f(1) = 0.95$$

2^{nd} order Approximation

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2$$ $$f(1) = 0.45$$

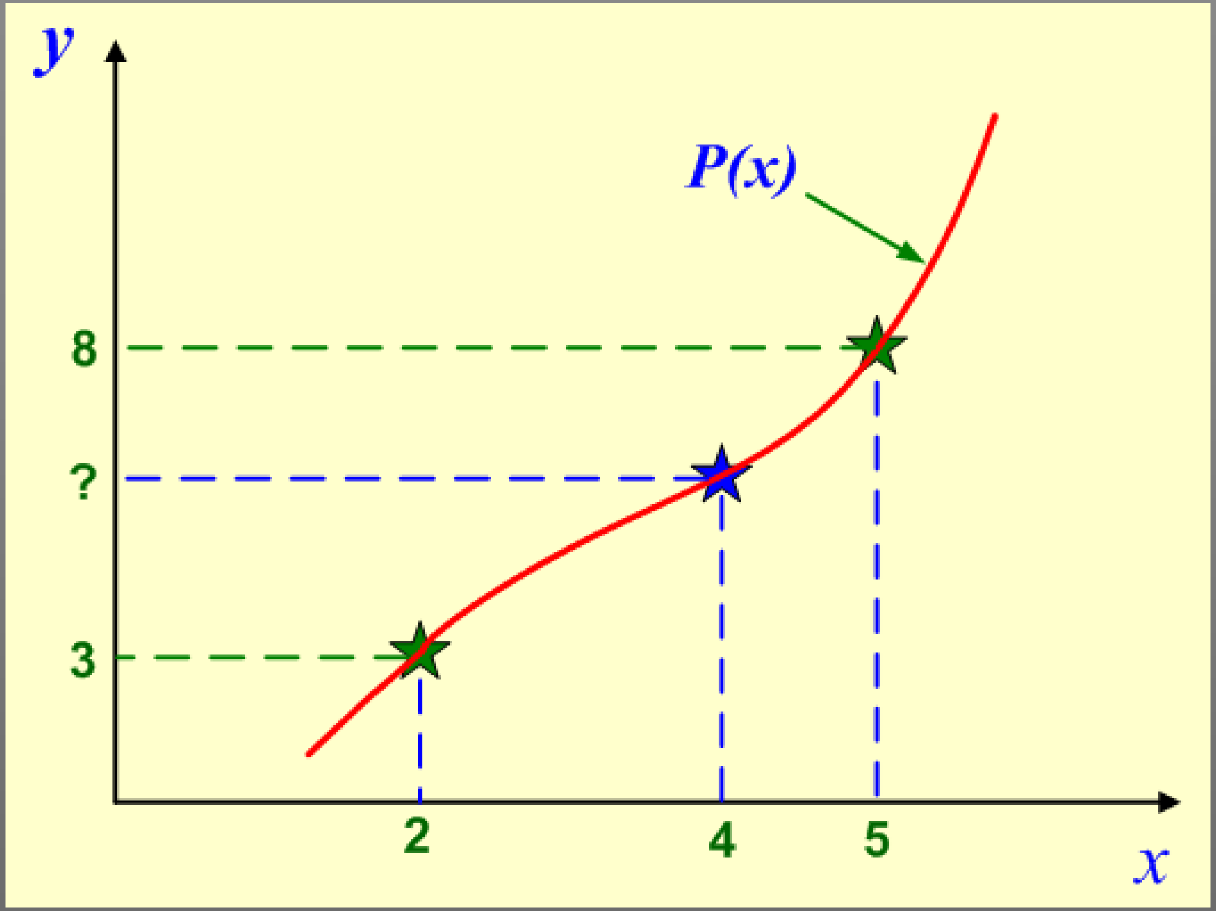


Interpolation

- Interpolation produces a function that matches the given data exactly.
- The function then can be utilized to approximate the data values at intermediate points.

Given data points: at $x_0 = 2$, $y_0 = 3$ and at $x_1 = 5$, $y_1 = 8$

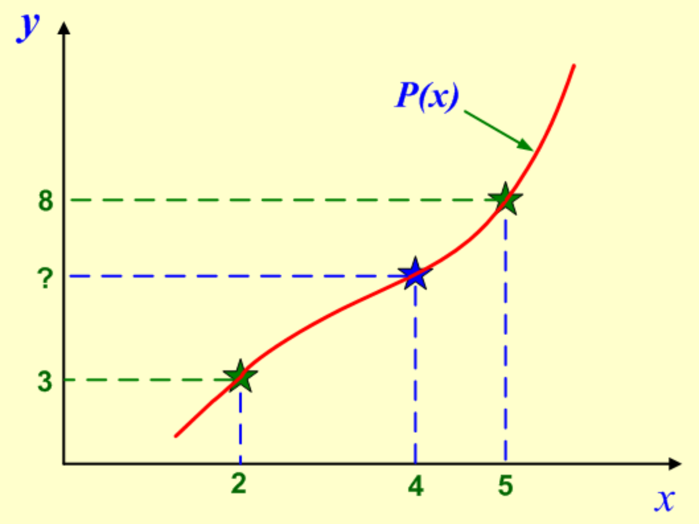
Find the following: at $x = 4$, $y = ?$



Introduction

- **If interpolation method provides a polynomial function with appropriate degree to exactly match a given set of data, polynomial interpolation is employed.**
  - **Lagrange Interpolation**
  - **Newton Interpolation**
- **Hermite interpolation method interpolate function as well as first derivative of function values.**

Lagrange Interpolation



Given data points: at $x_0 = 2, y_0 = 3$ and at $x_1 = 5, y_1 = 8$

Find the following: at $x = 4, y = ?$ Using Lagrange interpolation technique.

$P(x)$ should satisfy the following conditions:

$P(x = 2) = 3$ and $P(x = 5) = 8$.

Assume Function $P(x) = 3L_0(x) + 8L_1(x)$

P(x) can satisfy the above conditions if

at $x = x_0 = 2$, $L_0(x) = 1$ and $L_1(x) = 0$ and

at $x = x_1 = 5$, $L_0(x) = 0$ and $L_1(x) = 1$

The conditions can be satisfied if $L_0(x)$ and $L_1(x)$ are defined in the following way:

$$L_0(x) = \frac{x-5}{2-5} \quad \text{and} \quad L_1(x) = \frac{x-2}{5-2}$$

$$L_0(x) = \frac{x-x_1}{x_0-x_1} \quad \text{and} \quad L_1(x) = \frac{x-x_0}{x_1-x_0}$$

Lagrange Interpolation Contd…

$$P(x)=3L_0(x)+8L_1(x) \Longleftrightarrow L_0(x)=\frac{x-x_1}{x_0-x_1} \quad \text{and} \quad L_1(x)=\frac{x-x_0}{x_1-x_0}$$

$$\Downarrow$$

$$P(x)=L_0(x)y_0+L_1(x)y_1$$

Lagrange Interpolating Polynomial $P(x)=L_0(x)f(x_0)+L_1(x)f(x_1)$

$$P(x)=\left(\frac{x-x_1}{x_0-x_1}\right)(y_0)+\left(\frac{x-x_0}{x_1-x_0}\right)(y_1)$$

P(x = 4) can be obtained by:

$$P(x=4)=\left(\frac{4-5}{2-5}\right)(3)+\left(\frac{4-2}{5-2}\right)(8) \Longrightarrow P(x=4)=6.333$$

Lagrange Interpolation Contd…

Lagrange Interpolating Polynomial for three points:

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

$$P(x)=L_1y_1+L_2y_2+\ldots..+L_ny_n$$

***$L_k(x)$* can be defined by:**

$$L_k(x)=\frac{(x-x_1)\ldots..(x-x_{k-1})(x-x_{k+1})\ldots..(x-x_n)}{(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)}$$

Numerator

$$N_k(x)=(x-x_1)\ldots..(x-x_{k-1})(x-x_{k+1})\ldots..(x-x_n)$$

Denominator

$$D_k(x)=(x_k-x_1)\ldots(x_k-x_{k-1})(x_k-x_{k+1})\ldots(x_k-x_n)$$

MATLAB© Script

Simplest MATLAB Program for Coefficient of Lagrange Polynomial

```
clear all
clc
x = input('Define x vector = ');
y = input('Define y vector = ');
n=length(x);
for k = 1:n
    d(k) = 1;
    for i = 1: n
        if i ~= k
            d(k) = d(k) * (x(k) – x(i));
        end
    c(k) = y (k) / d(k);
    end
end
```

$$P(x) = c_1 N_1 + c_2 N_2 + \ldots\ldots + c_n N_n$$

$$c_k = \frac{y_k}{D_k} = \frac{y_k}{(x_k - x_1)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n)}$$

$$N_k(x) = (x - x_1)\ldots\ldots(x - x_{k-1})(x - x_{k+1})\ldots\ldots(x - x_n)$$

MATLAB© Script

Simplest MATLAB Program for Lagrange Polynomial

```
t=input('Enter point x at which polynomial get evaluated = ');
for i = 1:length(t)
    p(i) = 0;
    for j = 1:n
        N(j) = 1;
        for k = 1:n
            if j ~= k
                N(j) = N(j) * (t(i) - x(k));
            end
        end
        p(i) = p(i) + N(j) * c(j);
    end
end
```

$$P(x) = c_1 N_1 + c_2 N_2 + \ldots\ldots + c_n N_n$$

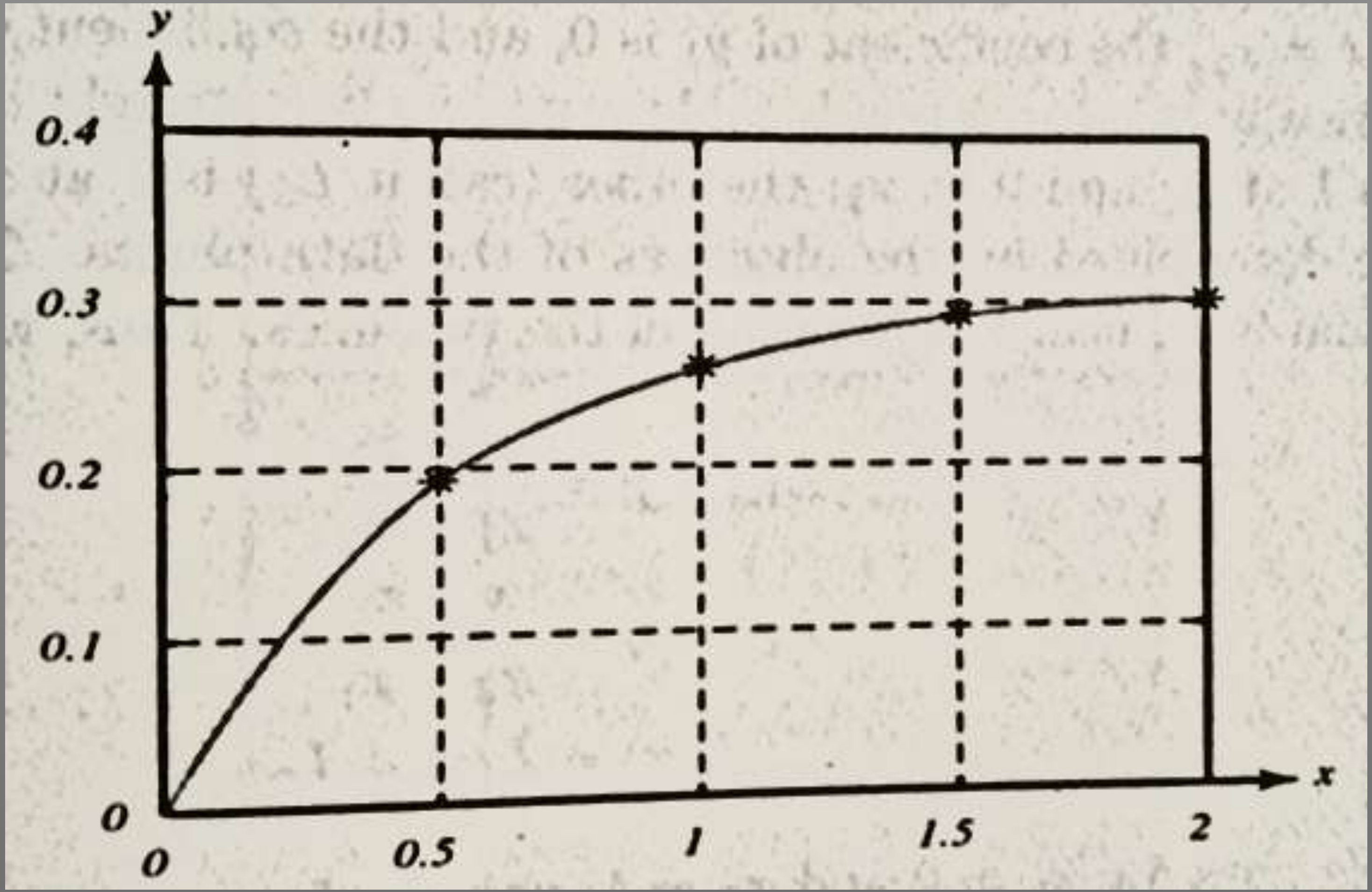
$$c_k = \frac{y_k}{D_k} = \frac{y_k}{(x_k - x_1)\ldots(x_k - x_{k-1})(x_k - x_{k+1})\ldots(x_k - x_n)}$$

$$N_k(x) = (x - x_1)\ldots\ldots(x - x_{k-1})(x - x_{k+1})\ldots\ldots(x - x_n)$$

p=interp1([x], [y], [t])

MATLAB© Script

Comparision of result obtained by developed and in-built function



p=interp1([x], [y], [t])

p=interp1([0 0.5 1 1.5 2],[0 0.19 0.26 0.29 0.31],[0.75 1.25])

Why Newton Interpolation

- ✌ The Lagrange formula is popular because it is well known and is easy to code.
- ✌ Also, the data are not required to be specified with x in ascending or descending order.
- 👎 If we decide to add a point to the set of nodes, we have to completely re-compute all of the polynomial functions.
- ☺ Here we introduce an alternative form of the polynomial: the Newton form (Divided-Difference Interpolating Polynomials).

Introduction

- **Newton interpolation assume a complete polynomial starts from lower degree to higher degree :**
  - **For two point Interpolation** $(x_1, y_1)\ and\ (x_2, y_2)$

$$P(x) = a_1 + a_2(x - x_1)$$

  - **For three point Interpolation** $(x_1, y_1), (x_2, y_2)\ and\ (x_3, y_3)$

$$P(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2)$$

  - **For n-point Interpolation** $(x_1, y_1), (x_2, y_2), \ldots\ldots, (x_n, y_n)$

$$P(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + \ldots. + a_n(x - x_1)\ldots(x - x_{n-1})$$

Newton Interpolation

Given data points: $(x_1, y_1), (x_2, y_2)\ and\ (x_3, y_3)$

Find the coefficient of polynomial.

$P(x)$ for 3 data set: $P(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2)$

At $x = x_1$ and $y = y_1$ $P(x) = y_1 = a_1 + a_2(x_1 - x_1) + a_3(x_1 - x_1)(x_1 - x_2)$

$$y_1 = a_1$$

At $x = x_2$ and $y = y_2$ $P(x) = y_2 = a_1 + a_2(x_2 - x_1) + a_3(x_2 - x_1)(x_2 - x_2)$

$$y_2 = y_1 + a_2(x_2 - x_1) \Longrightarrow a_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

At $x = x_3$ and $y = y_3$ $a_3 = \dfrac{\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1}$

Newton Interpolation Example

For given data points: $(x_1, y_1) = (-2,4); (x_2, y_2) = (0,2)$ *and* $(x_3, y_3) = (2,8)$

Find the Newton interpolation polynomial and value at x =1 .

For 3-data set: $P(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2)$

Coefficients of polynomial: $a_1 = y_1 = 4$

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

$$a_3 = \frac{\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1} = \frac{\dfrac{8-2}{2-0} - \dfrac{2-4}{0-(-2)}}{2-(-2)} = 1$$

Put these value in polynomial $P(x) = 4 - (x - x_1) + (x - x_1)(x - x_2)$

Impose co-ordinate in polynomial $P(x) = 4 - (x - (-2)) + (x - (-2))(x - 0)$

$P(x) = 4 - (x + 2) + (x + 2)x \Longrightarrow P(x) = x^2 + x + 2 \Longrightarrow P(1) = 4$

MATLAB© Script

MATLAB Program for Coefficient of Newton Polynomial

```
clear all
clc
x = input('Define x vector = ');
y = input('Define y vector = ');
n=length(x);
a(1) = y(1);
for k = 1 : n - 1
   d(k, 1) = (y(k+1) - y(k))/(x(k+1) - x(k));
end
for j = 2 : n - 1
   for k = 1 : n - j
      d(k, j) = (d(k+1, j - 1) - d(k, j - 1))/(x(k+j) - x(k));
   end
end
```

```
for j = 2 : n
   a(j) = d(1, j-1);
end
```

$$a_1 = y_1$$

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

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

MATLAB© Script

MATLAB Program for Newton Interpolation

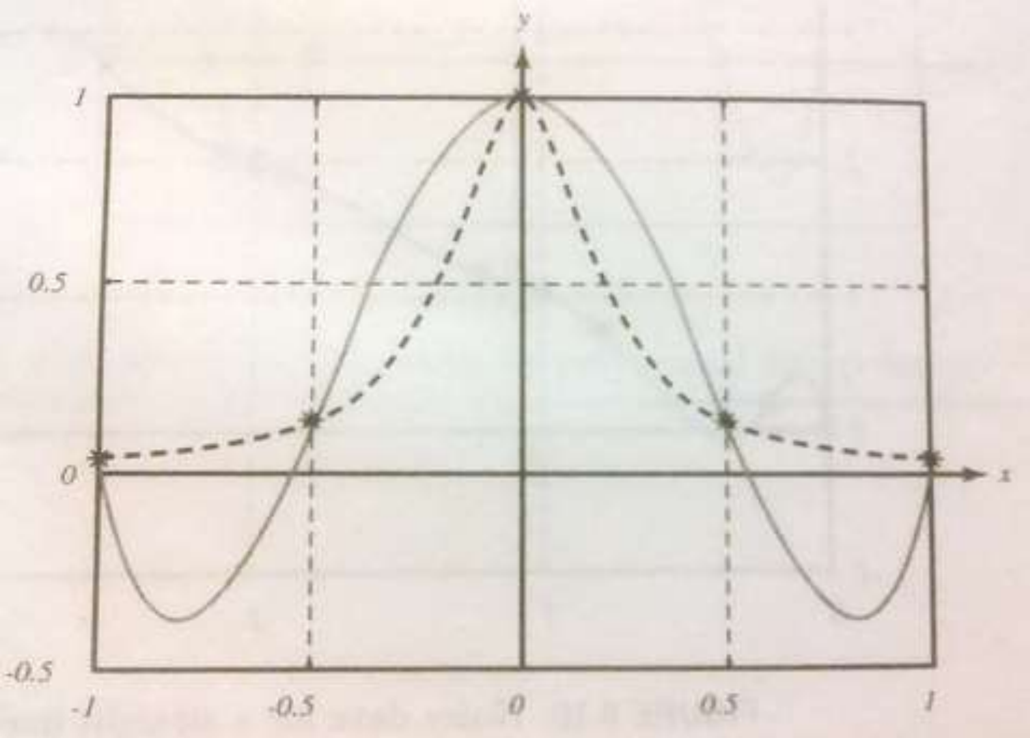
```
t=input('Enter point x at which polynomial get evaluated = ');
for i = 1:length(t)
    d(1) = 1;
    N = a(1);
     for j = 2:n
        d(j) = (t(i) – x(j-1)) * d(j-1);
         N(i) = N(i) + a(j) * d(j);
      end
 end
```

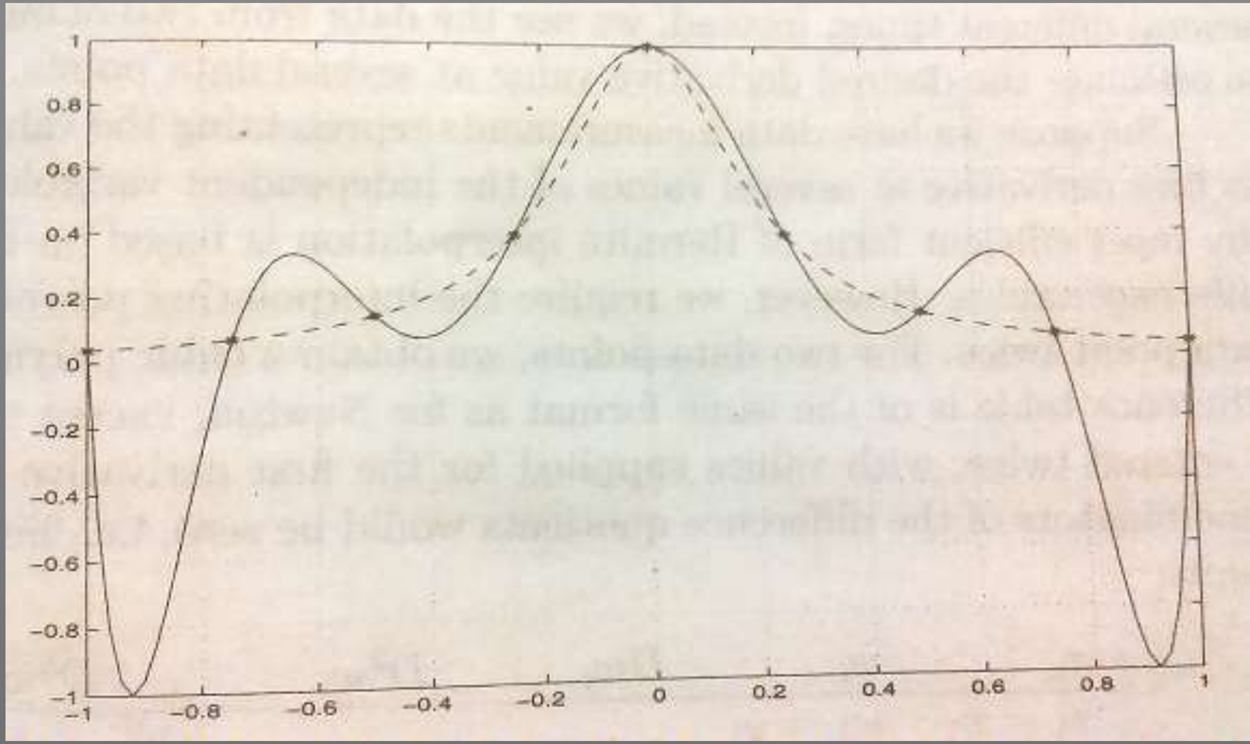
$$P(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2)$$

Piecewise Interpolation

Runge Function

$$f(x) = \frac{1}{1+25x^2}$$





Piecewise Interpolation

Picewise linear interpolation for four data points:

$$\left(x_1, y_1\right), \left(x_2, y_2\right), \left(x_3, y_3\right) \; and \; \left(x_4, y_4\right)$$

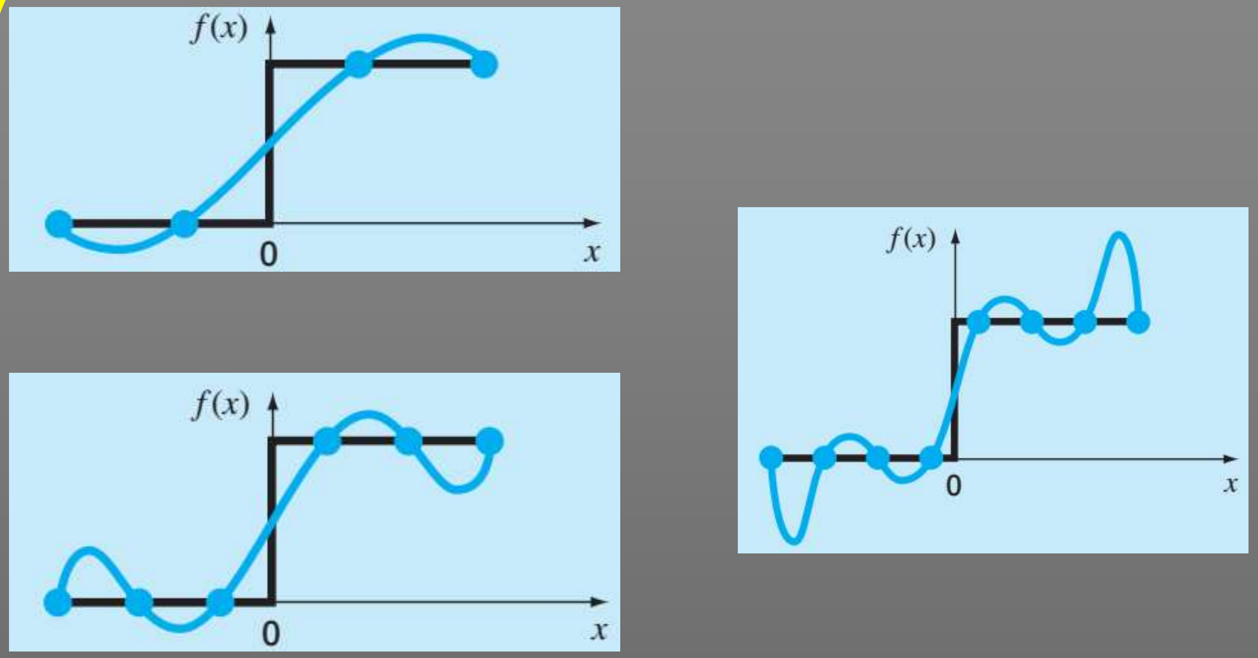
Provided $x_1 < x_2 < x_3 < x_4$

Sub-domain $I_1 = \left[x_1, x_2\right], \quad I_2 = \left[x_2, x_3\right], \quad I_3 = \left[x_3, x_4\right]$

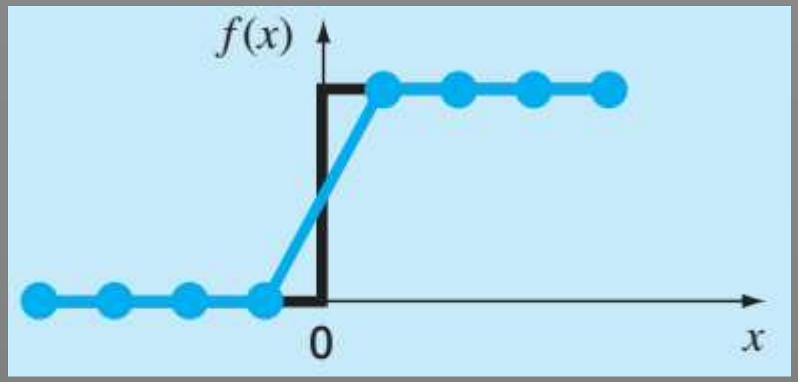
$$P(x) = \begin{cases} \left(\dfrac{x - x_2}{x_1 - x_2}\right)(y_1) + \left(\dfrac{x - x_1}{x_2 - x_1}\right)(y_2); & x_1 \le x < x_2 \\ \left(\dfrac{x - x_3}{x_2 - x_3}\right)(y_2) + \left(\dfrac{x - x_2}{x_3 - x_2}\right)(y_3); & x_2 \le x < x_3 \\ \left(\dfrac{x - x_4}{x_3 - x_4}\right)(y_3) + \left(\dfrac{x - x_3}{x_4 - x_3}\right)(y_4); & x_3 \le x \le x_4 \end{cases}$$

Spline Interpolation

- **A spline is a special function defined piecewise by polynomials.**
- **In essence, spline interpolation is a special type of polynomial interpolation where each section/range is interpolated by different polynomial functions.**



Polynomial Interpolation



Linear Spline Interpolation

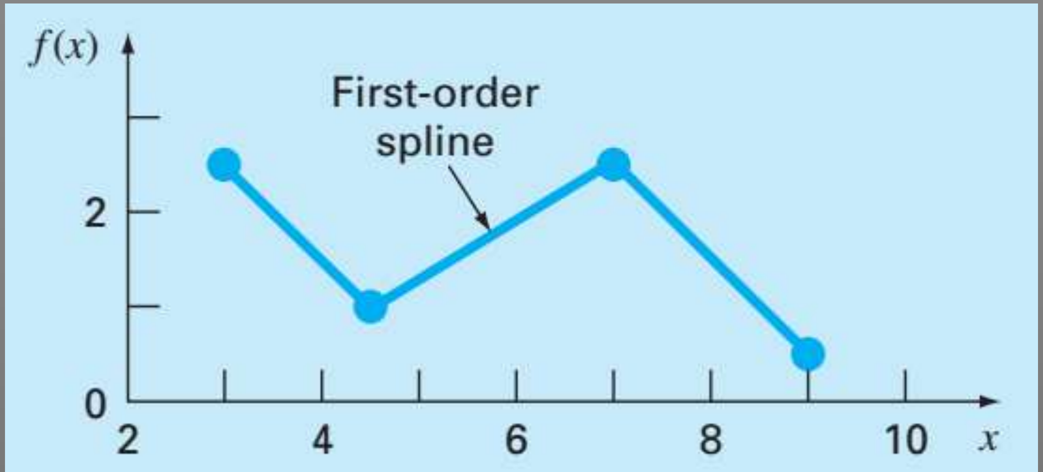
$$
\begin{aligned}
f(x) &= f(x_0) + m_0(x - x_0) & x_0 \le x \le x_1 \\
f(x) &= f(x_1) + m_1(x - x_1) & x_1 \le x \le x_2 \\
&\vdots \\
f(x) &= f(x_{n-1}) + m_{n-1}(x - x_{n-1}) & x_{n-1} \le x \le x_n
\end{aligned}
$$

$$m_i = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}$$

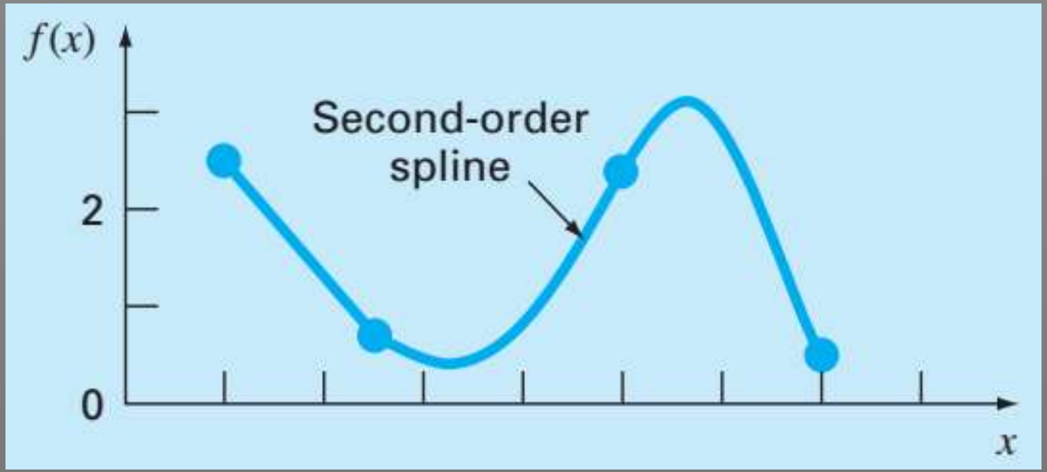
Quadratic Spline Interpolation

- A spline interpolation using second-order polynomials.
- In essence, quadratic splines have continuous first derivatives at the knots.

$$f_i(x) = a_i x^2 + b_i x + c_i$$



Linear Spline Interpolation



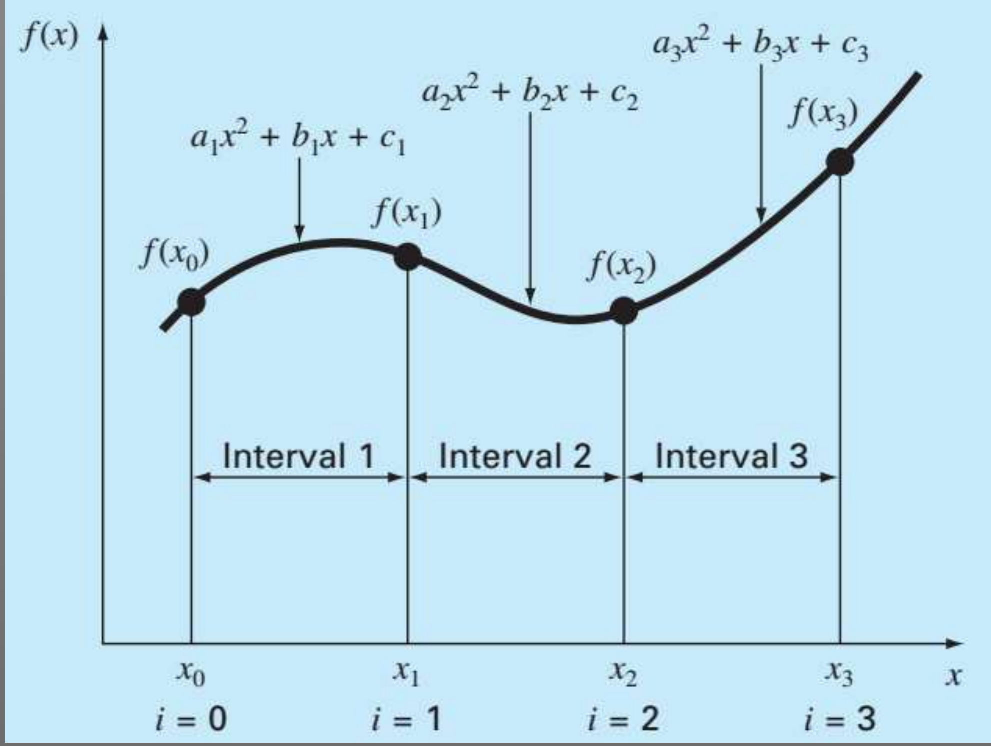
Quadratic Spline Interpolation

Quadratic Spline Interpolation

Quadratic Spline interpolation for four data points:

Polynomial for each interval: $f_i(x) = a_i x^2 + b_i x + c_i$

For n+1data points, there are n intervals and, 3n unknown constants to be find for interpolation functions.



Quadratic Spline Interpolation

Interpolation conditions (3n):

01. The function values of adjacent polynomials must be equal at the interior knots.

$$a_{i-1}x_{i-1}^2 + b_{i-1}x_{i-1} + c_{i-1} = f(x_{i-1})$$
$$a_i x_{i-1}^2 + b_i x_{i-1} + c_i = f(x_{i-1})$$

02. The first and last functions must pass through the end points.

$$a_1 x_0^2 + b_1 x_0 + c_1 = f(x_0)$$
$$a_n x_n^2 + b_n x_n + c_n = f(x_n)$$

Quadratic Spline Interpolation

For n+1data points, there are n intervals and, 3n unknown constants to be find for interpolation functions.

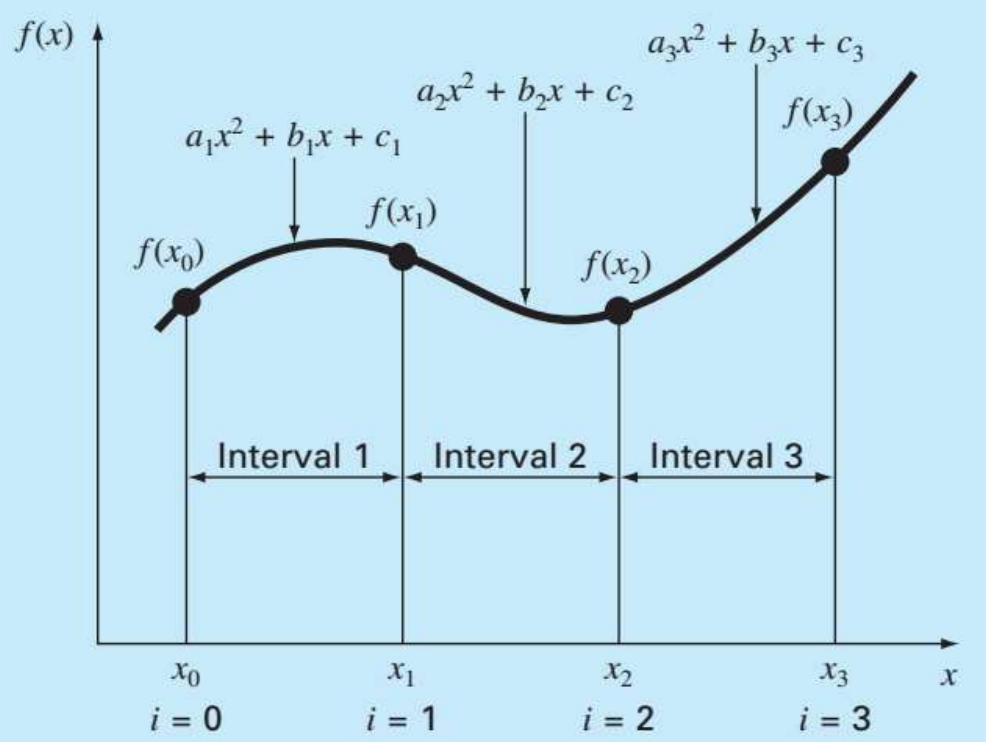
Interpolation conditions continue…

03. The first derivatives at the interior knots must be equal.

$$f'(x) = 2ax + b$$

$$2a_{i-1}x_{i-1} + b_{i-1} = 2a_i x_{i-1} + b_i$$

04. Assume that the second derivative is zero at the first point. $a_1 = 0$



Quadratic Spline Interpolation

Quadratic Spline Interpolation

Given data points:

We have four data points and n = 3 intervals.
Therefore, 3×3=9 unknowns must be determined.

Data to be fit with spline functions.

x	f(x)
3.0	2.5
4.5	1.0
7.0	2.5
9.0	0.5

$$a_{i-1}x_{i-1}^2 + b_{i-1}x_{i-1} + c_{i-1} = f(x_{i-1})$$
$$a_i x_{i-1}^2 + b_i x_{i-1} + c_i = f(x_{i-1})$$

$\Rightarrow$

$$20.25a_1 + 4.5b_1 + c_1 = 1.0$$
$$20.25a_2 + 4.5b_2 + c_2 = 1.0$$
$$49a_2 + 7b_2 + c_2 = 2.5$$
$$49a_3 + 7b_3 + c_3 = 2.5$$

$$a_1x_0^2 + b_1x_0 + c_1 = f(x_0)$$
$$a_nx_n^2 + b_nx_n + c_n = f(x_n)$$

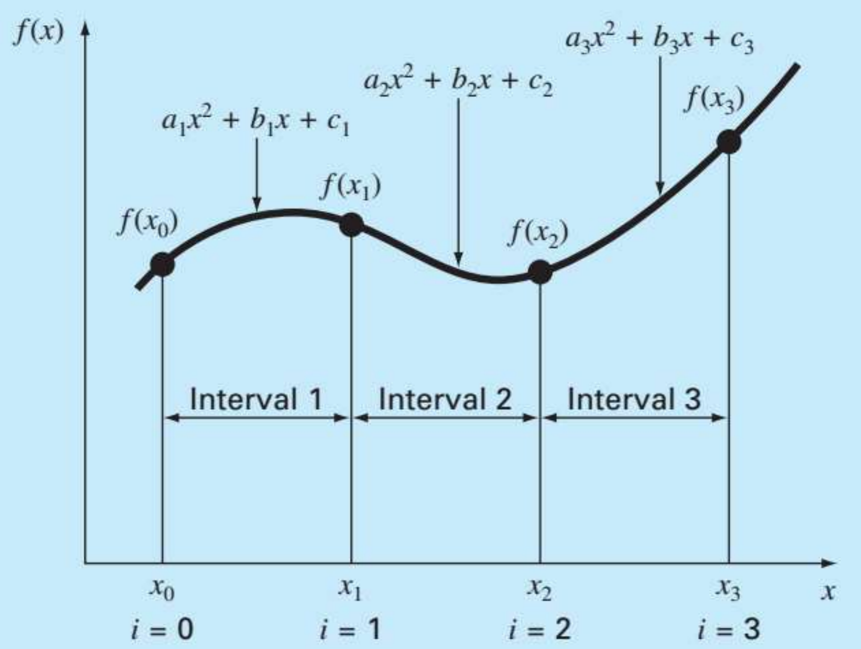
$\Rightarrow$

$$9a_1 + 3b_1 + c_1 = 2.5$$
$$81a_3 + 9b_3 + c_3 = 0.5$$

$$2a_{i-1}x_{i-1} + b_{i-1} = 2a_ix_{i-1} + b_i$$

$\Rightarrow$

$$9a_1 + b_1 = 9a_2 + b_2$$
$$14a_2 + b_2 = 14a_3 + b_3$$



Quadratic Spline Interpolation

Quadratic Spline Interpolation

Obtained Equations:

$$20.25a_1 + 4.5b_1 + c_1 = 1.0$$
$$20.25a_2 + 4.5b_2 + c_2 = 1.0$$
$$49a_2 + 7b_2 + c_2 = 2.5$$
$$49a_3 + 7b_3 + c_3 = 2.5$$

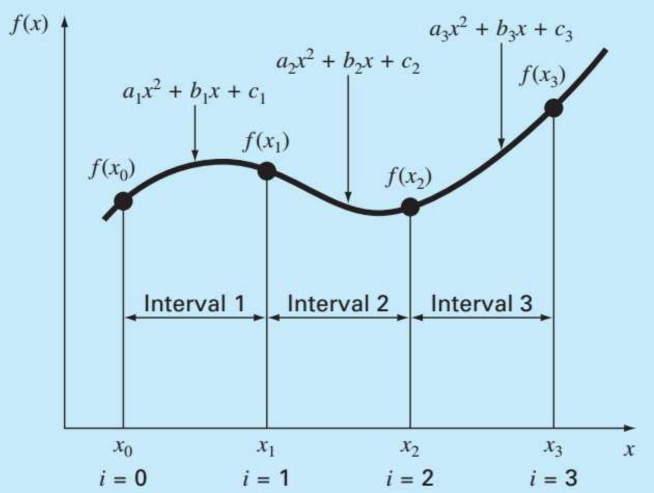
$$9a_1 + 3b_1 + c_1 = 2.5$$
$$81a_3 + 9b_3 + c_3 = 0.5$$

$$9a_1 + b_1 = 9a_2 + b_2$$
$$14a_2 + b_2 = 14a_3 + b_3$$

$$\begin{bmatrix} 4.5 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 20.25 & 4.5 & 1 & 0 & 0 & 0 \\ 0 & 0 & 49 & 7 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 49 & 7 & 1 \\ 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 81 & 9 & 1 \\ 1 & 0 & -9 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 14 & 1 & 0 & -14 & -1 & 0 \end{bmatrix} \begin{Bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \end{Bmatrix} = \begin{Bmatrix} 1 \\ 1 \\ 2.5 \\ 2.5 \\ 2.5 \\ 0.5 \\ 0 \\ 0 \end{Bmatrix}$$

$$\begin{array}{lll} a_1 = 0 & b_1 = -1 & c_1 = 5.5 \\ a_2 = 0.64 & b_2 = -6.76 & c_2 = 18.46 \\ a_3 = -1.6 & b_3 = 24.6 & c_3 = -91.3 \end{array}$$

$$\begin{array}{ll} f_1(x) = -x + 5.5 & 3.0 \le x \le 4.5 \\ f_2(x) = 0.64x^2 - 6.76x + 18.46 & 4.5 \le x \le 7.0 \\ f_3(x) = -1.6x^2 + 24.6x - 91.3 & 7.0 \le x \le 9.0 \end{array}$$



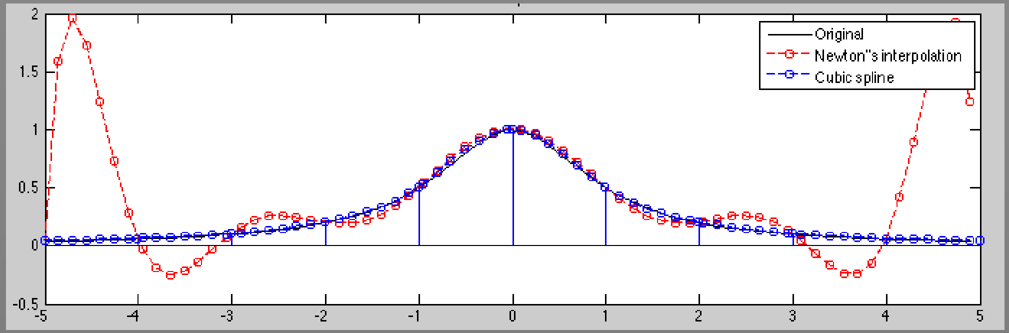
Quadratic Spline Interpolation

Cubic Spline Interpolation

To define third-order polynomial for each interval between knots:

Polynomial for each interval: $f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$

For n+1data points, there are n intervals and, 4n unknown constants to be find for interpolation functions.



Cubic Spline Interpolation

Interpolation conditions (4n):

1. *The function values of adjacent polynomials must be equal at the interior knots.*
2. *The first and last functions must pass through the end points.*
3. *The first derivatives at the interior knots must be equal.*
4. *The second derivatives at the interior knots must be equal.*
5. *The second derivatives at the end knots are zero (natural spline).*

Cubic Spline Interpolation

Interpolation equations:

$$f_i''(x) = f_i''(x_{i-1})\frac{x - x_i}{x_{i-1} - x_i} + f_i''(x_i)\frac{x - x_{i-1}}{x_i - x_{i-1}}$$

$$\begin{aligned} f_i(x) = {} & \frac{f_i''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f_i''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1}) \\ & + \left[\frac{f(x_{i-1})}{x_i - x_{i-1}} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6}\right](x_i - x) \\ & + \left[\frac{f(x_i)}{x_i - x_{i-1}} - \frac{f''(x_i)(x_i - x_{i-1})}{6}\right](x - x_{i-1}) \end{aligned}$$

To find f''(x)

$$\begin{aligned} & (x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + (x_{i+1} - x_i)f''(x_{i+1}) \\ & = \frac{6}{x_{i+1} - x_i}[f(x_{i+1}) - f(x_i)] + \frac{6}{x_i - x_{i+1}}[f(x_{i+1}) - f(x_i)] \end{aligned}$$

Cubic Spline Interpolation

Given data points:

We have four data points and n = 3 intervals. Therefore, 4×3=12 unknowns must be determined.

Data to be fit with spline functions.

x	$f(x)$
3.0	2.5
4.5	1.0
7.0	2.5
9.0	0.5

$$x_0 = 3 \qquad f(x_0) = 2.5$$
$$x_1 = 4.5 \qquad f(x_1) = 1$$
$$x_2 = 7 \qquad f(x_2) = 2.5$$

To find f''(x)

$$(x_i - x_{i-1})f''(x_{i-1}) + 2(x_{i+1} - x_{i-1})f''(x_i) + (x_{i+1} - x_i)f''(x_{i+1}) = \frac{6}{x_{i+1} - x_i}[f(x_{i+1}) - f(x_i)] + \frac{6}{x_i - x_{i+1}}[f(x_{i+1}) - f(x_i)]$$

$$(4.5 - 3)f''(3) + 2(7 - 3)f''(4.5) + (7 - 4.5)f''(7) = \frac{6}{7 - 4.5}(2.5 - 1) + \frac{6}{4.5 - 3}(2.5 - 1)$$

$$f''(3) = 0$$

$$8f''(4.5) + 2.5f''(7) = 9.6$$
$$2.5f''(4.5) + 9f''(7) = -9.6$$

$$f''(4.5) = 1.67909$$
$$f''(7) = -1.53308$$

Cubic Spline Interpolation

To find $f_i(x)$

$$f_i(x) = \frac{f_i''(x_{i-1})}{6(x_i - x_{i-1})}(x_i - x)^3 + \frac{f_i''(x_i)}{6(x_i - x_{i-1})}(x - x_{i-1})^3 + \left[\frac{f(x_{i-1})}{x_i - x_{i-1}} - \frac{f''(x_{i-1})(x_i - x_{i-1})}{6}\right](x_i - x) + \left[\frac{f(x_i)}{x_i - x_{i-1}} - \frac{f''(x_i)(x_i - x_{i-1})}{6}\right](x - x_{i-1})$$

$$f_1(x) = 0.186566(x - 3)^3 + 1.666667(4.5 - x) + 0.246894(x - 3)$$

$$f_2(x) = 0.111939(7 - x)^3 - 0.102205(x - 4.5)^3 - 0.299621(7 - x) + 1.638783(x - 4.5)$$

$$f_3(x) = -0.127757(9 - x)^3 + 1.761027(9 - x) + 0.25(x - 7)$$

Finite Divided Difference Method

- **Divided differences is a recursive division process. The method can be used to calculate the coefficients in the interpolation polynomial in the Newton form.**
- **The Taylor series used to approximate divided differences.**

Taylor Series

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{n}(x_i)}{n!}h^n + R_n$$

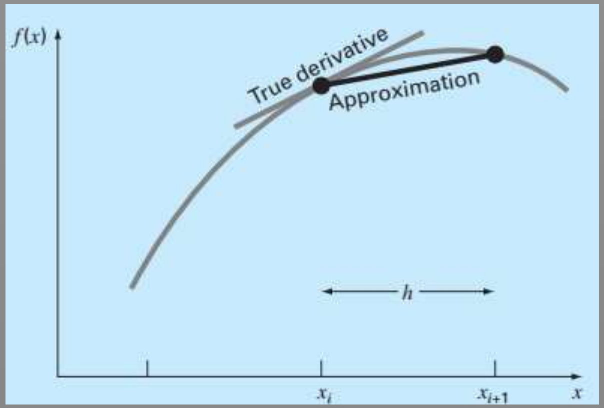
$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + O(x_{i+1} - x_i) \Rightarrow f'(x_i) = \frac{\Delta f_i}{h} + O(h)$$

First Forward Difference

Δf_i is referred to as the first forward difference and h is called the step size

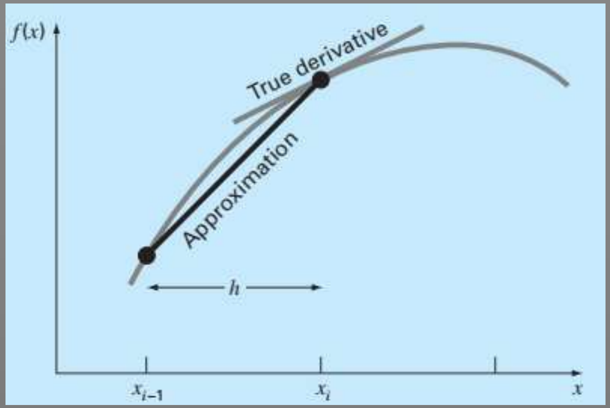
Finite Divided Difference Method

$$\partial_x f^+ \approx \frac{f(x+dx)-f(x)}{dx}$$ **forward difference**



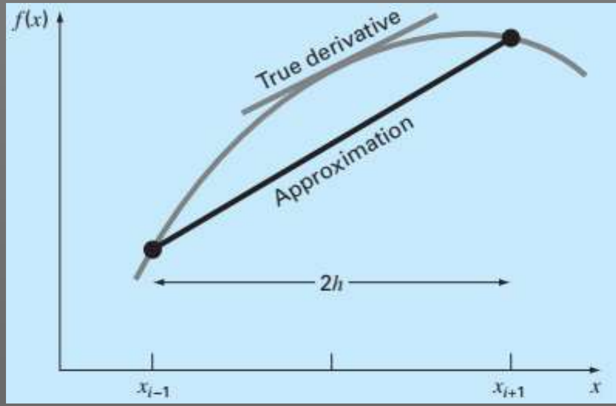
It utilizes data at *i* and *i* $_{1+1}$ to estimate the derivative.

$$\partial_x f^- \approx \frac{f(x)-f(x-dx)}{dx}$$ **backward difference**



To calculate a previous value on the basis of a present value

$$\partial_x f \approx \frac{f(x+dx)-f(x-dx)}{2dx}$$ **centered difference**



It utilizes data at *i* $_{1-1}$ and *i* $_{1+1}$ to estimate the derivative.

Finite Divided Difference Table

i	x_i	$f[x_i]$	First order differences	Second order differences	Third order differences	Fourth order differences	Fifth order differences
0	x_0	$f[x_0]$					
			$f[x_0, x_1]$				
1	x_1	$f[x_1]$		$f[x_0,x_1,x_2]$			
			$f[x_1,x_2]$		$f[x_0,x_1,x_2,x_3]$		
2	x_2	$f[x_2]$		$f[x_1,x_2,x_3]$		$f[x_0,x_1,x_2,x_3,x_4]$	
			$f[x_2,x_3]$		$f[x_1,x_2,x_3,x_4]$		$f[x_0,x_1,x_2,x_3,x_4,x_5]$
3	x_3	$f[x_3]$		$f[x_2,x_3,x_4]$		$f[x_1,x_2,x_3,x_4,x_5]$	
			$f[x_3,x_4]$		$f[x_2,x_3,x_4,x_5]$		
4	x_4	$f[x_4]$		$f[x_3,x_4,x_5]$			
			$f[x_4,x_5]$				
5	x_5	$f[x_5]$					

i	x_i	$f[x_i]$	1st order differences	2nd order differences	3rd order differences	4th order differences
0	0	0				
			$\frac{1-0}{1-0} = 1$			
1	1	1		$\frac{7-1}{2-0} = 3$		
			$\frac{8-1}{2-1} = 7$		$\frac{6-3}{3-0} = 1$	
2	2	8		$\frac{19-7}{3-1} = 6$		$\frac{1-1}{4-0} = 0$
			$\frac{27-8}{3-2} = 19$		$\frac{9-6}{4-1} = 1$	
3	3	27		$\frac{37-19}{4-2} = 9$		
			$\frac{64-27}{4-3} = 37$			
4	4	64				

Finite Divided Difference Table

Example: Compute f(0.3) for the data using Newton's divided difference formula.

x	0	1	3	4	7
f	1	3	49	129	813

Divided difference table

x_i	f_i			
0	1			
		2		
1	3		7	
		23		3
3	49		19	
		80		3
4	129		37	
		228		
7	813			

$$f(x) = f[x_0] + (x - x_0) f[x_0, x_1] + (x - x_0)(x - x_1) f[x_0, x_1, x_2] + (x - x_0)(x - x_1)(x - x_2) f[x_0, x_1, x_2, x_3]$$

$$f(0.3) = 1 + (0.3 - 0)\,2 + (0.3)(0.3 - 1)\,7 + (0.3)(0.3 - 1)(0.3 - 3)\,3$$

$$=1.831$$

THANK YOU

Questions??