Lecture 5

Interpolation

Ye Ding (丁 烨)

Email: y.ding@sjtu.edu.cn

School of Mechanical Engineering

Shanghai Jiao Tong University

Interpolation

- **References for Interpolations**

[1] Timothy Sauer, Numerical analysis (2nd ed.), Pearson Education, 2012. Chapter 3

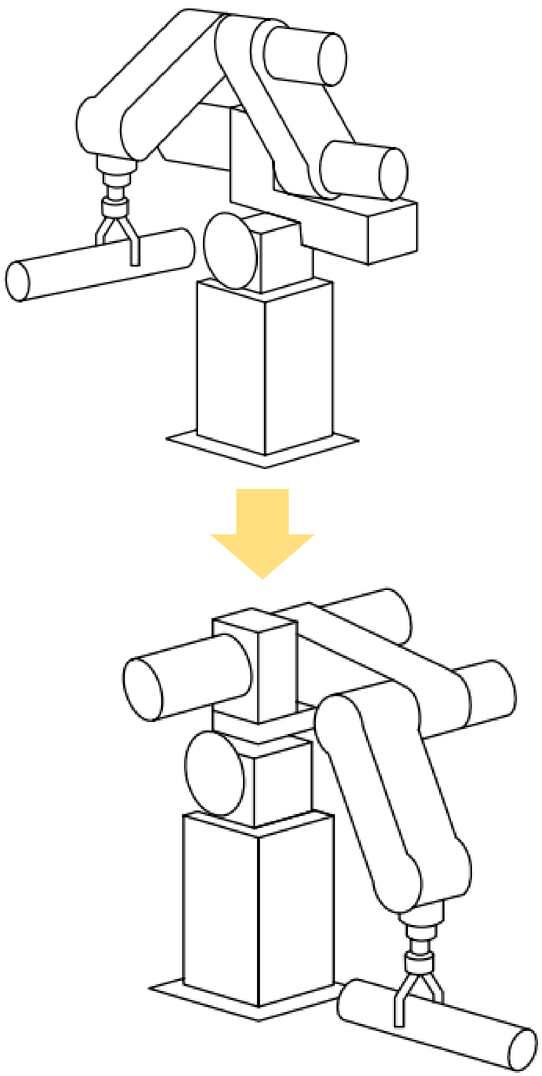
[2] Cleve Moler, Numerical Computing with MATLAB, Society for Industrial and Applied Mathematics, 2004. Chapter 3

[3] Richard L. Burden, J. Douglas Faires, Numerical analysis (9th ed.), Brooks/Cole, 2011. Chapter 3

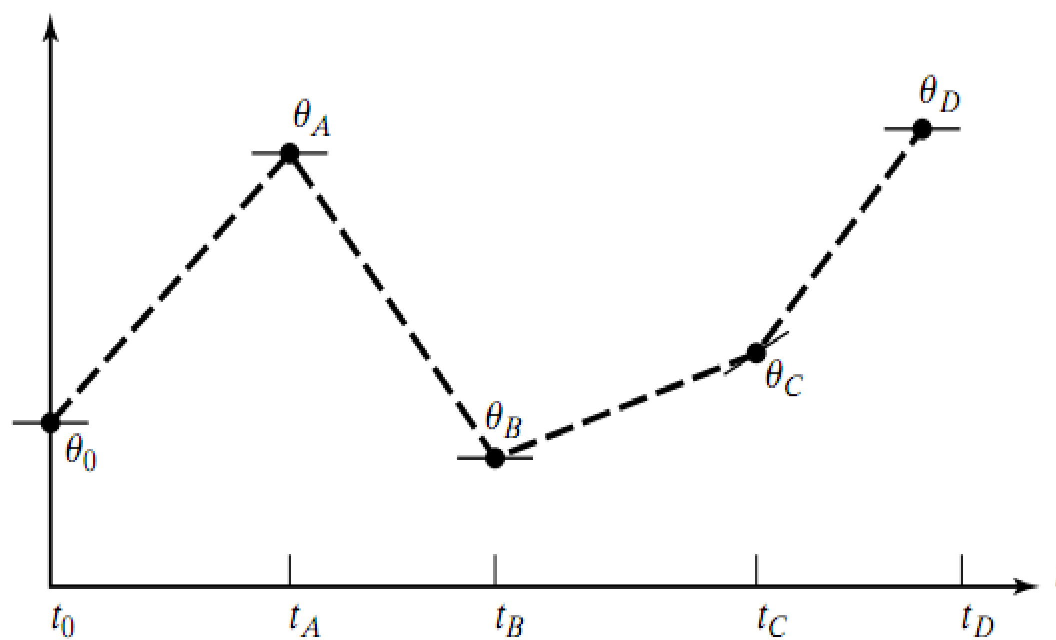
[4] 李庆扬等，数值分析（第5版），清华大学出版社，2008. 第二章

Interpolation

- **Why Interpolation? Robotics**

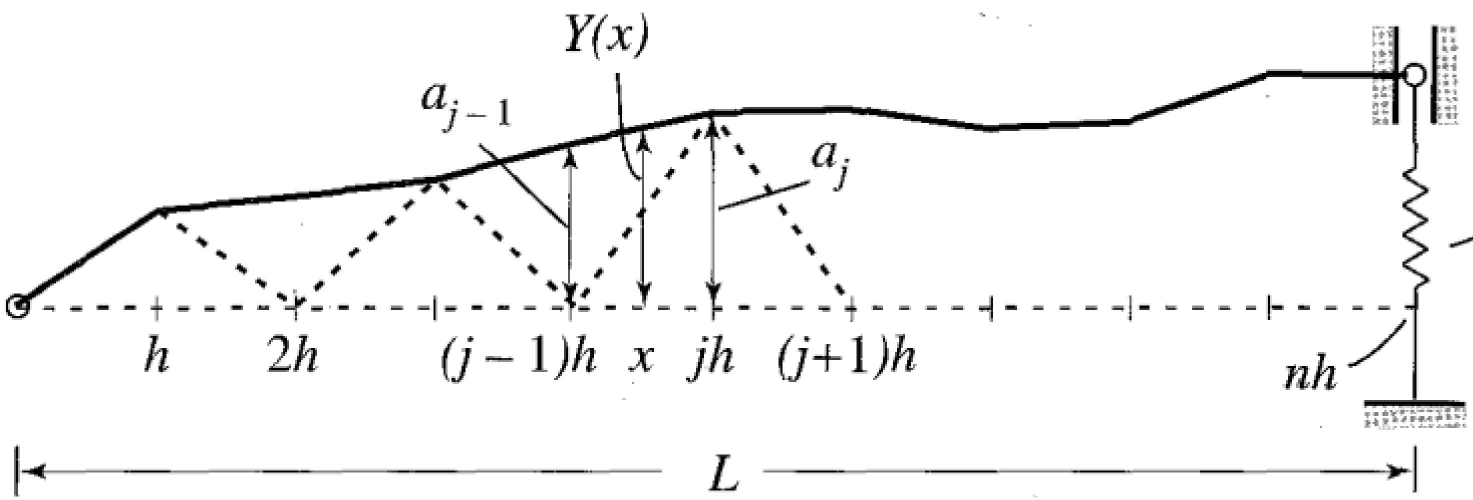


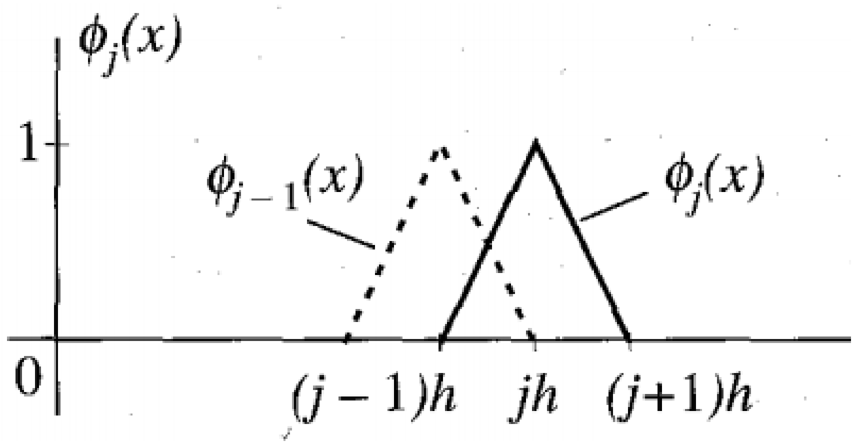
A manipulator moves from its initial position to a desired goal position in a smooth manner.

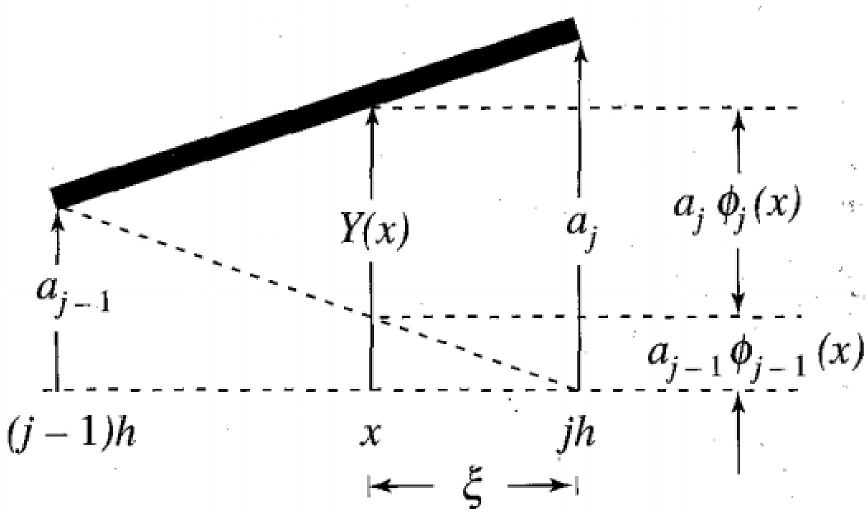


Interpolation

- **Why Interpolation? Finite Element Method**



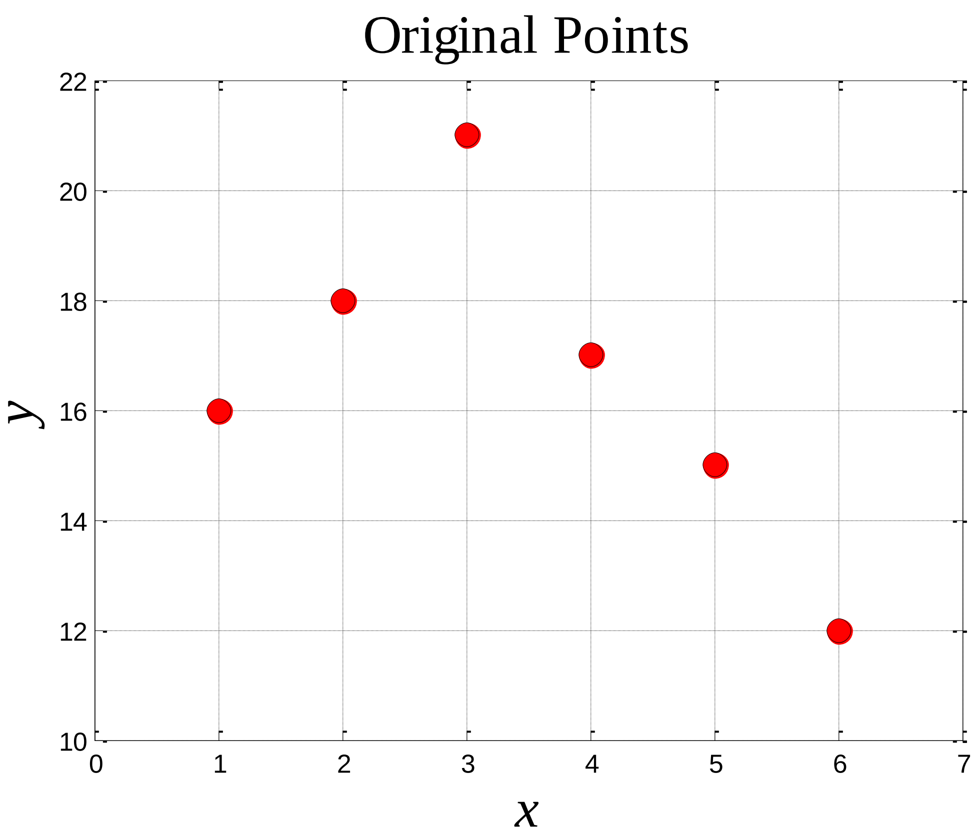




Interpolation Using MATLAB

Why Interpolation?

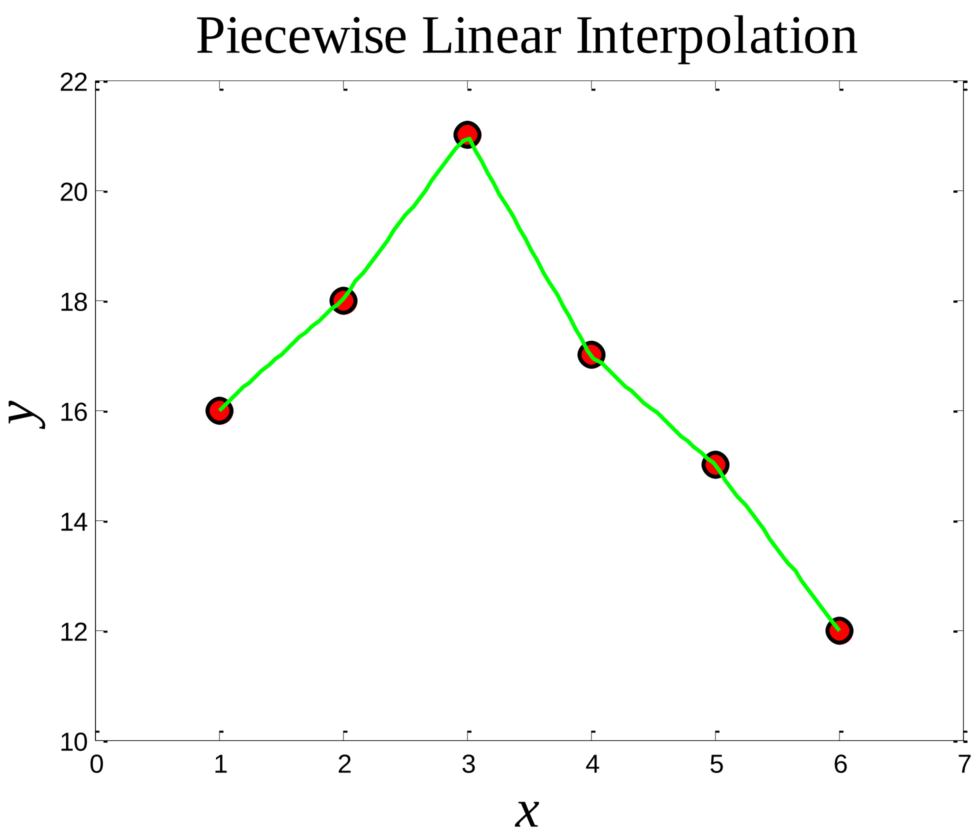
```
x = 1:6;
y = [16, 18, 21, 17, 15, 12];
plot(x,y,'o')
```



Interpolation Using MATLAB

Why Interpolation?

```
x = 1:6; y = [16, 18, 21, 17, 15, 12];
u = linspace(1,6,100); v = interp1(x,y,u,'linear');
plot(x,y,'o',u,v,'g-')
```



Interpolation Using MATLAB

- **Aim of Interpolations**

  Given a set of pairs of values (x_i , y_i), $i = 0,1,\ldots,n$, we construct a continuous function $y = P(x)$ that in some sense represents an underlying function implied by the data points.

The function $y = P(x)$ interpolates the data points $(x_0,y_0),\ldots,(x_n,y_n)$, if $P(x_i) = y_i$ for each $0 \le i \le n$.

Interpolation Using MATLAB

- **Popular Methods for Interpolations**
  - **Lagrange Interpolation Method**
  - **Newton's Divided Differences**
  - **Hermite Interpolation**
  - **Cubic Spline Interpolation**

Lagrange Interpolation

Basic Idea

The function $y = P(x)$ **interpolates** the data points $(x_0,y_0),\ldots,(x_n,y_n)$ if $P(x_i) = y_i$ for each $0 \le i \le n$.

We want to find the **coefficients** of an *n*th-degree polynomial function to match them:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Lagrange Interpolation

- **Basic Idea: Direct Method**

The function $y = P(x)$ interpolates the data points $(x_0,y_0),\ldots,(x_n,y_n)$ if $P(x_i) = y_i$ for each $0 \le i \le n$.

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

$$\begin{cases} a_0 + x_0 a_1 + x_0^2 a_2 + \cdots + x_0^n a_n = y_0 \\ a_0 + x_1 a_1 + x_1^2 a_2 + \cdots + x_1^n a_n = y_1 \\ \vdots \\ a_0 + x_n a_1 + x_n^2 a_2 + \cdots + x_n^n a_n = y_n \end{cases}$$

Lagrange Interpolation

- In matrix form, the system is

$$\mathbf{Xa} = \mathbf{y}$$

where

$$\mathbf{X} = \left[x_i^j \right], i, j = 0, 1, \cdots, n$$

$$\mathbf{a} = \begin{bmatrix} a_0 & \cdots & a_n \end{bmatrix}^T, \mathbf{y} = \begin{bmatrix} y_0 & \cdots & y_n \end{bmatrix}^T$$

- The matrix **X** is known as the Vandermonde matrix.
- Solving the system **X*a* = y** is equivalent to solving the polynomial interpolation problem.

Lagrange Interpolation

Lagrange Interpolating Polynomial: Example 1

Suppose that we are given two points $(x_k, y_k), (x_{k+1}, y_{k+1})$.

The Lagrange polynomial of degree 1 in the variable x for these points:

$$P_1(x) = y_k \frac{x - x_{k+1}}{x_k - x_{k+1}} + y_{k+1} \frac{x - x_k}{x_{k+1} - x_k}$$

$$\triangleq y_k \cdot \ell_k(x) + y_{k+1} \cdot \ell_{k+1}(x)$$

$$\begin{cases} \ell_k(x_k) = 1, \ \ell_k(x_{k+1}) = 0 \\ \ell_{k+1}(x_k) = 0, \ell_{k+1}(x_{k+1}) = 1 \end{cases}$$

Lagrange Interpolation

Lagrange Interpolating Polynomial: Example 2

Suppose that we are given three points $(x_1, y_1), (x_2, y_2), (x_3, y_3)$.

The Lagrange polynomial of degree 2 in the variable x for these points:

$$P_2(x) = y_1 \underbrace{\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}}_{\ell_1(x)} + y_2 \underbrace{\frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}}_{\ell_2(x)} + y_3 \underbrace{\frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}}_{\ell_3(x)}$$

$$\begin{cases} \ell_1(x_1)=1 \\ \ell_1(x_2)=0 \\ \ell_1(x_3)=0 \end{cases} \qquad \begin{cases} \ell_2(x_1)=0 \\ \ell_2(x_2)=1 \\ \ell_2(x_3)=0 \end{cases} \qquad \begin{cases} \ell_3(x_1)=0 \\ \ell_3(x_2)=0 \\ \ell_3(x_3)=1 \end{cases}$$

Lagrange Interpolation

- **Lagrange Interpolating Polynomial: General Case**

Suppose that we are given *n+1* points $(x_0,y_0), (x_1,y_1),\ldots,(x_n,y_n)$

The Lagrange polynomial of degree *n* in the variable *x* for these points:

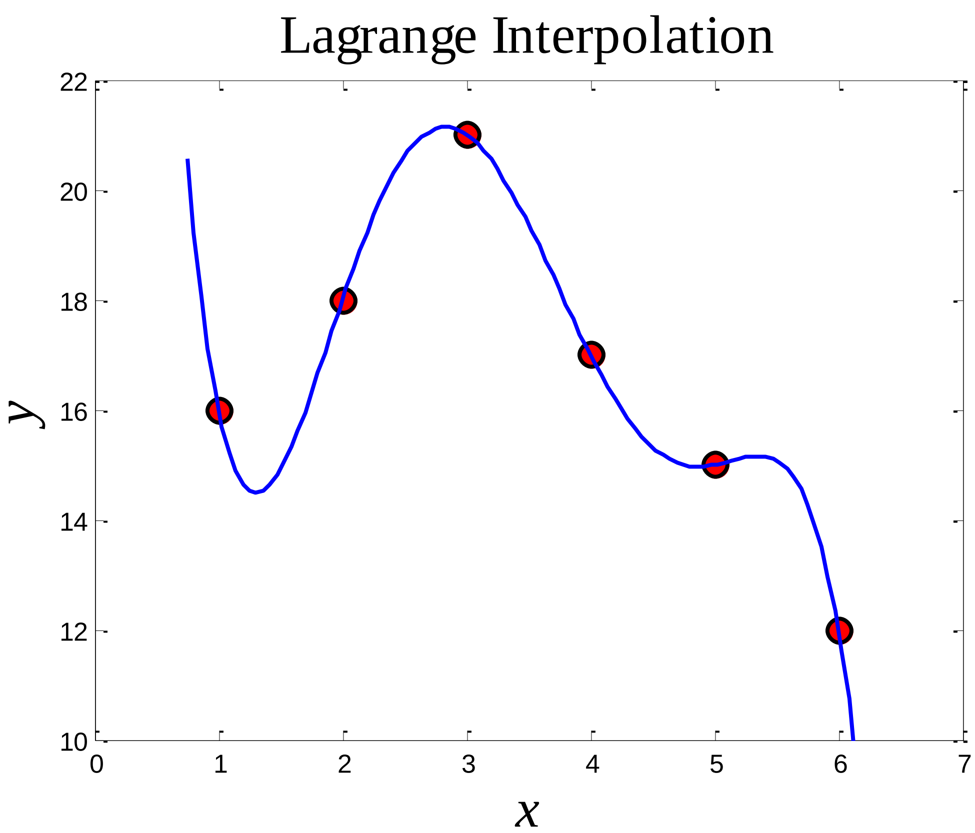
$$\boxed{P_n(x)=\sum_{k=0}^{n} y_k\cdot \ell_k(x)}$$

$$\ell_k(x)=\frac{(x-x_0)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)}=\prod_{\substack{j=0\\ j\neq k}}^{n}\frac{(x-x_j)}{(x_k-x_j)}$$

Lagrange Interpolation

Numerical Example 1

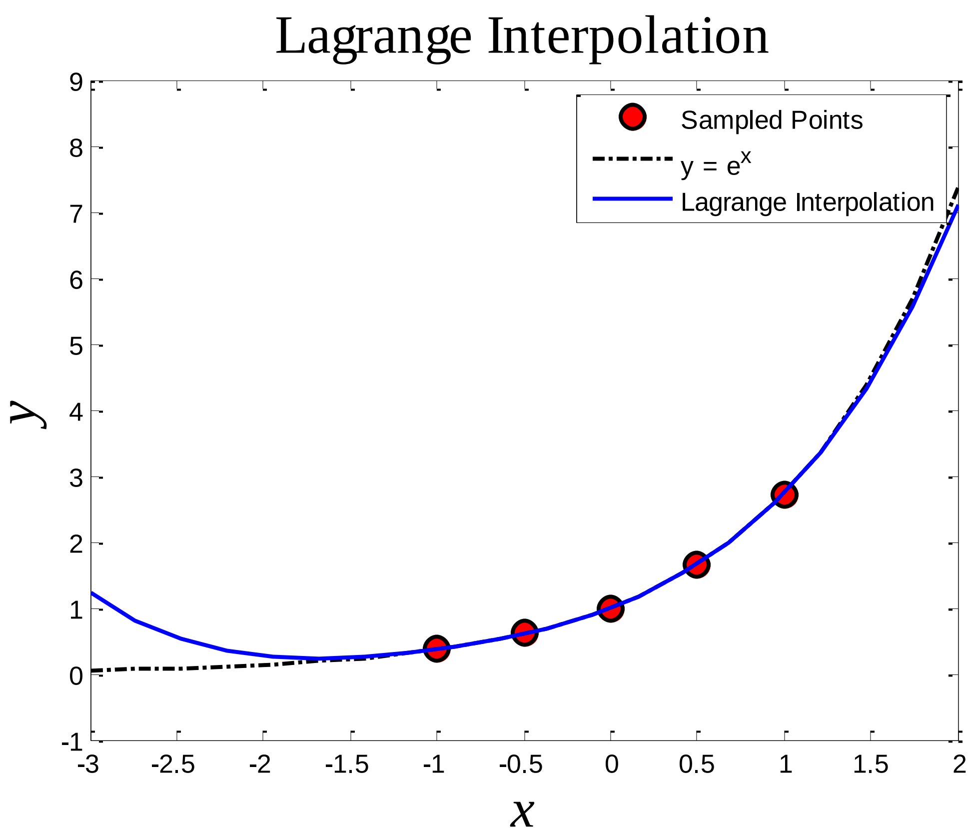
```
x = 1:6; y = [16, 18, 21, 17, 15, 12];
u = linspace(0.75,6.25,100); v = Lagrange(x,y,u);
plot(x,y,'o',u,v,'b-')
```



Lagrange Interpolation

Numerical Example 2

```
x = [-1,-0.5,0,0.5,1]; y = exp(x);
u = linspace(-3,2,20); v = Lagrange(x,y,u);
plot(x,y,'o',u,exp(u),'k-.',u,v,'b-')
```



Lagrange Interpolation

- **Lagrange Interpolating Polynomial: Theorems**

Main Theorem of Polynomial Interpolation. Let $(x_0,y_0),\ldots,(x_n,y_n)$ be $n+1$ points in the plane with distinct x_k. Then there exists one and only one polynomial P of degree n or less that satisfies $P(x_k) = y_k$ for $k = 0,\ldots,n$.

Proof. *cf.* P. 141 in Ref. [1]

Lagrange Interpolation

- **Lagrange Interpolating Polynomial: Theorems**

Assume that $P(x)$ is the (degree n or less) interpolating polynomial fitting the n+1 points $(x_0,y_0),\ldots,(x_n,y_n)$ sampled from $f(x)$. The interpolation error is

$$f(x) - P_n(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{(n+1)}(\xi)$$

***cf.* P. 152 in Ref. [1]**

Interpolation Using MATLAB

- **Popular Methods for Interpolations**
  - **Lagrange Interpolation Method**
  - **Newton's Divided Differences**
  - **Hermite Interpolation**
  - **Cubic Spline Interpolation**

Newton's Divided Differences

Basic Idea

The function $y = P(x)$ interpolates the data points $(x_0,y_0),\ldots,(x_n,y_n)$ if $P(x_i) = y_i$ for each $0 \le i \le n$.

Consider the first two data points (x_0,y_0) and (x_1,y_1):

$$P_1(x) = a_0 + a_1(x - x_0)$$

$$a_0 + a_1(x_0 - x_0) = y_0$$

$$a_0 + a_1(x_1 - x_0) = y_1$$

$$\Rightarrow$$

$$a_0 = y_0$$

$$a_1 = \frac{y_1 - a_0}{x_1 - x_0} = \boxed{\frac{y_1 - y_0}{x_1 - x_0}}$$

$$\boxed{f[x_0,x_1]}$$

Newton's Divided Differences

- **Basic Idea**

Consider the first three data points (x_0,y_0) , (x_1,y_1), and (x_2,y_2):

$$P_2(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1)$$

$$a_0 + a_1(x_2 - x_0) + a_2(x_2 - x_0)(x_2 - x_1) \equiv y_2$$

$$a_2 = \frac{y_2 - a_0 - a_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{y_2 - y_0 - \frac{y_1 - y_0}{x_1 - x_0}(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)}$$

$$= \frac{\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_1 - y_0}{x_1 - x_0}}{x_2 - x_0} \boxed{\triangleq \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0} \triangleq f[x_0, x_1, x_2]}$$

Newton's Divided Differences

General Formula

The function $y = P(x)$ interpolates the data points $(x_0, y_0), \ldots, (x_n, y_n)$ if $P(x_i) = y_i$ for each $0 \leq i \leq n$.

$$
\begin{aligned}
P(x) = f[x_0] &+ \underline{f[x_0, x_1]}(x - x_0) \\
&+ \underline{f[x_0, x_1, x_2]}(x - x_0)(x - x_1) \\
&+ \underline{f[x_0, x_1, x_2, x_3]}(x - x_0)(x - x_1)(x - x_2) \\
&+ \cdots \\
&+ \underline{f[x_0, x_1, \cdots, x_n]}(x - x_0)(x - x_1)\cdots(x - x_{n-1})
\end{aligned}
$$

Newton's Divided Differences

- **General Formula: the Divided Differences**

List the data points in a table:

$$
\begin{array}{c|c}
x_1 & f(x_1) \\
x_2 & f(x_2) \\
\vdots & \vdots \\
x_n & f(x_n)
\end{array}
$$

Divided Differences:

$$f[x_k] = f(x_k)$$

$$f[x_k\ x_{k+1}] = \frac{f[x_{k+1}] - f[x_k]}{x_{k+1} - x_k}$$

$$f[x_k\ x_{k+1}\ x_{k+2}] = \frac{f[x_{k+1}\ x_{k+2}] - f[x_k\ x_{k+1}]}{x_{k+2} - x_k}$$

$$f[x_k\ x_{k+1}\ x_{k+2}\ x_{k+3}] = \frac{f[x_{k+1}\ x_{k+2}\ x_{k+3}] - f[x_k\ x_{k+1}\ x_{k+2}]}{x_{k+3} - x_k}$$

Newton's Divided Differences

General Formula: the Divided Differences

Recursive Table:

$$
\begin{array}{c|ccccc}
x_0 & f[x_0] & & & & \\
x_1 & f[x_1] & f[x_0, x_1] & & & \\
x_2 & f[x_2] & f[x_1, x_2] & f[x_0, x_1, x_2] & & \\
\vdots & \vdots & & \vdots & \ddots & \\
x_n & f[x_n] & f[x_{n-1}, x_n] & f[x_{n-2}, x_{n-1}, x_n] & \ldots & f[x_0, \ldots, x_n]
\end{array}
$$

Newton's Divided Differences

Numerical Example 3

Use divided differences to find the interpolating polynomial passing through the points (0,1), (2,2), (3,4).

x_i	$f(x_i)$	$f[x_{i-1},x_i]$	$f[x_{i-2},x_{i-1},x_i]$
0	1		
2	2	1/2	
3	4	2	1/2

$$P(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2)$$

Newton's Divided Differences

Numerical Example 3

Use divided differences to find the interpolating polynomial passing through the points (0,1), (2,2), (3,4), and (1,0).

x_i	$f(x_i)$	$f[x_{i-1},x_i]$	$f[x_{i-2},x_{i-1},x_i]$	$f[x_{i-3},x_{i-2},x_{i-1},x_i]$
0	1			
2	2	1/2		
3	4	2	1/2	
1	0	2	0	-1/2

$$P_3(x) = 1 + \frac{1}{2}(x-0) + \frac{1}{2}(x-0)(x-2) - \frac{1}{2}(x-0)(x-2)(x-3)$$

Newton's Divided Differences

- **Evaluating a Polynomial**

$$P(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

Method 1: The most straightforward approach

Arithmetic operations:

Method 2: Nested multiplication

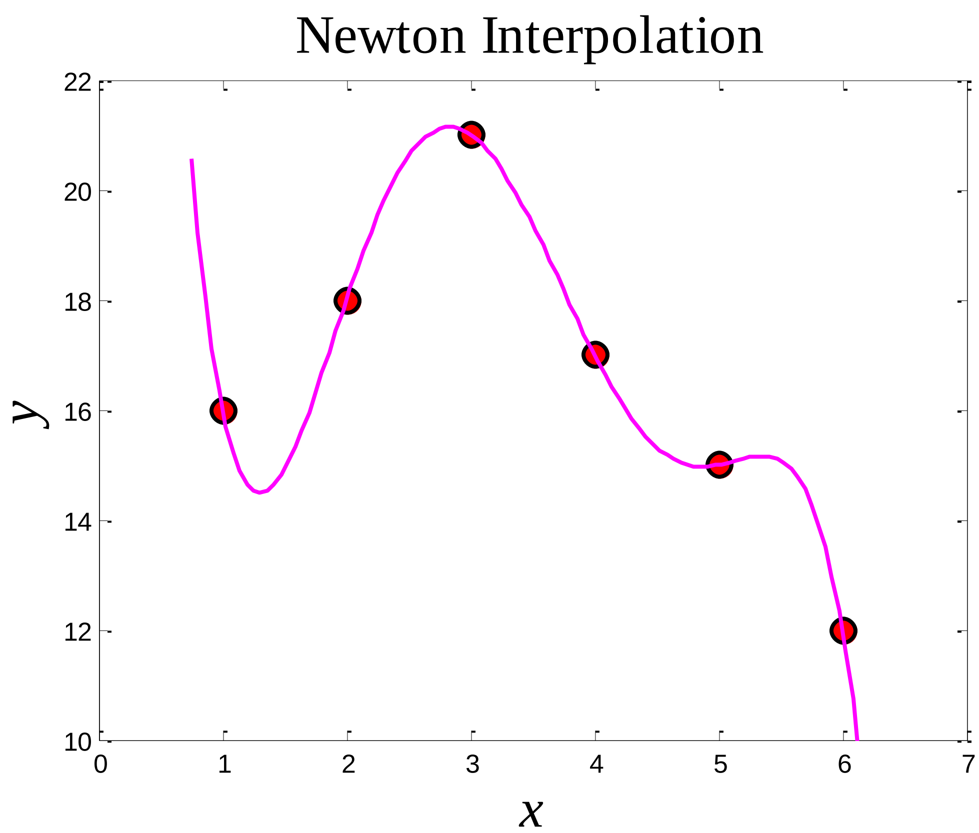
$$P(x) = a_0 + x(\ldots + x(a_{n-2} + x(a_{n-1} + a_n x)) \ldots)$$

Arithmetic operations:

Newton's Divided Differences

Numerical Example 4

```
x = 1:6; y = [16, 18, 21, 17, 15, 12];
u = linspace(0.75,6.25,100); v = Newton(x,y,u);
plot(x,y,'o',u,v,'m-')
```



Interpolation Using MATLAB

- **Popular Methods for Interpolations**
  - **Lagrange Interpolation Method**
  - **Newton's Divided Differences**
  - **Hermite Interpolation**
  - **Cubic Spline Interpolation**

Hermite Interpolation

- **Motivation**

We want to find the polynomial function that not only passes through the given points, but also has the specified derivatives at every data point.

The function $y = H(x)$ interpolates the data points $(x_0,y_0),\ldots,(x_n,y_n)$ if $H(x_i) = y_i$ and $H'(x_i) = y'_i$ for each $0 \le i \le n$.

$$H(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3$$

Hermite Interpolation

Basic Idea

Consider just two points (x_0,y_0), (x_1,y_1) and having the specified first derivatives y'_0, y'_1 at the points.

We want to find the coefficients of the *3rd*-degree polynomial function to match them:

$$H(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3$$

Hermite Interpolation

- **Basic Idea: Direct Method**

Consider just two points (x_0,y_0), (x_1,y_1) and having the specified first derivatives y'_0, y'_1 at the points.

$$H(x) = h_0 + h_1 x + h_2 x^2 + h_3 x^3$$

$$\begin{cases} h_0 + x_0 h_1 + x_0^2 h_2 + x_0^3 h_3 = y_0 \\ h_0 + x_1 h_1 + x_1^2 h_2 + x_1^3 h_3 = y_1 \\ \qquad h_1 + 2x_0 h_2 + 3x_0^2 h_3 = y'_0 \\ \qquad h_1 + 2x_1 h_2 + 3x_1^2 h_3 = y'_1 \end{cases} \Rightarrow \mathbf{Ah} = \mathbf{b}$$

Hermite Interpolation

Alternative Method

Consider just two points (x_k,y_k), (x_{k+1},y_{k+1}) and having the specified first derivatives y'$_k$ = m_k, y'$_{k+1}$ = m_{k+1} at the points.

$$\boxed{H(x) = \alpha_k(x)y_k + \alpha_{k+1}(x)y_{k+1} + \beta_k(x)m_k + \beta_{k+1}(x)m_{k+1}}$$

$$\begin{cases} \alpha_k(x_k) = 1, \alpha_k(x_{k+1}) = 0, \alpha_k'(x_k) = 0, \alpha_k'(x_{k+1}) = 0; \\ \alpha_{k+1}(x_k) = 0, \alpha_{k+1}(x_{k+1}) = 1, \alpha_{k+1}'(x_k) = 0, \alpha_{k+1}'(x_{k+1}) = 0; \\ \beta_k(x_k) = 0, \beta_k(x_{k+1}) = 0, \beta_k'(x_k) = 1, \beta_k'(x_{k+1}) = 0; \\ \beta_{k+1}(x_k) = 0, \beta_{k+1}(x_{k+1}) = 0, \beta_{k+1}'(x_k) = 0, \beta_{k+1}'(x_{k+1}) = 1. \end{cases}$$

Hermite Interpolation

Alternative Method

$$\alpha_k(x_k)=1, \alpha_k(x_{k+1})=0, \alpha_k'(x_k)=0, \alpha_k'(x_{k+1})=0;$$

Let $$\alpha_k(x)=(ax+b)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2$$

$$\begin{cases}\alpha_k(x_k)=ax_k+b=1\\ \alpha_k'(x_k)=2\dfrac{ax_k+b}{x_k-x_{k+1}}+a=0\end{cases} \Rightarrow \begin{cases}a=-\dfrac{2}{x_k-x_{k+1}}\\ b=1+\dfrac{2x_k}{x_k-x_{k+1}}\end{cases}$$

$$\alpha_k(x)=\left(1+2\frac{x-x_k}{x_{k+1}-x_k}\right)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2$$

Hermite Interpolation

Alternative Method

$$\alpha_{k+1}(x_k)=0,\ \alpha_{k+1}(x_{k+1})=1,\ \alpha'_{k+1}(x_k)=0,\ \alpha'_{k+1}(x_{k+1})=0$$

$$\Downarrow$$

$$\alpha_{k+1}(x)=\left(1+2\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)\left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2$$

Hermite Interpolation

- **Alternative Method**

$$\beta_k(x_k)=0,\ \beta_k(x_{k+1})=0,\ \beta_k'(x_k)=1,\ \beta_k'(x_{k+1})=0$$

Let $$\beta_k(x)=a(x-x_k)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2$$

$$\beta_k'(x_k)=1 \quad \Rightarrow \quad a=1$$

$$\beta_k(x)=(x-x_k)\left(\frac{x-x_{k+1}}{x_k-x_{k+1}}\right)^2$$

Hermite Interpolation

Alternative Method

$$\beta_{k+1}(x_k)=0,\ \beta_{k+1}(x_{k+1})=0,\ \beta'_{k+1}(x_k)=0,\ \beta'_{k+1}(x_{k+1})=1$$

$$\Downarrow$$

$$\beta_{k+1}(x)=(x-x_{k+1})\left(\frac{x-x_k}{x_{k+1}-x_k}\right)^2$$

Hermite Interpolation

Alternative Method

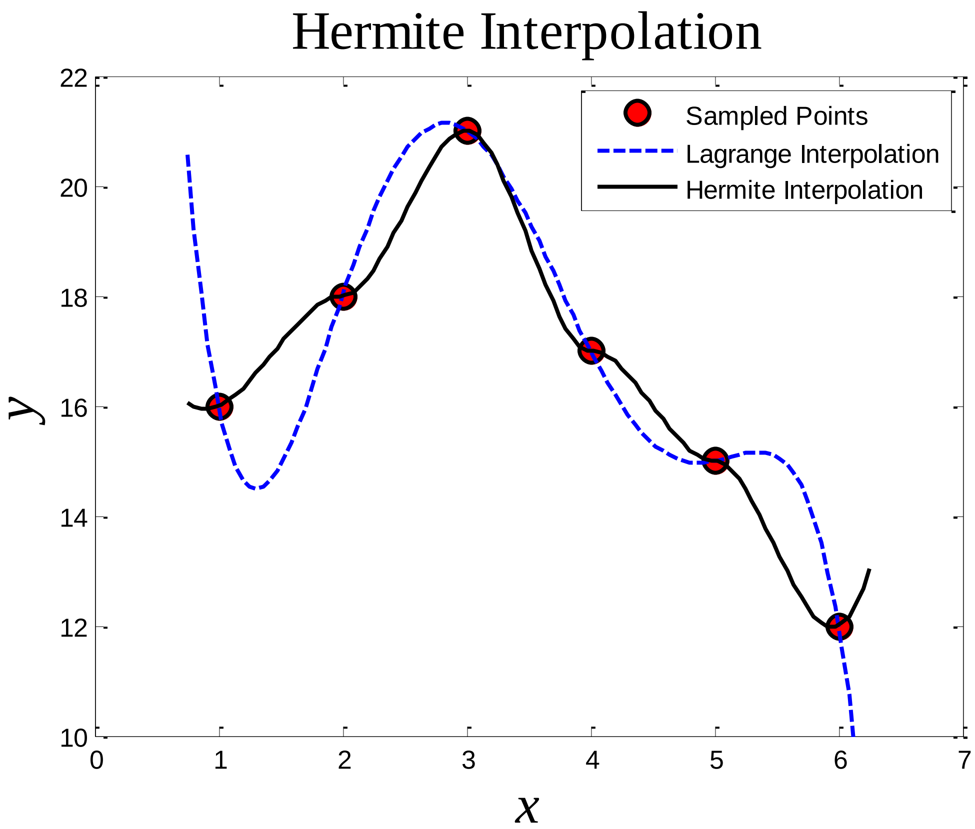
Consider just two points (x_k, y_k), (x_{k+1}, y_{k+1}) and having the specified first derivatives $y'_k = m_k$, $y'_{k+1} = m_{k+1}$ at the points.

$$
\begin{aligned}
H(x) &= \alpha_k(x)y_k + \alpha_{k+1}(x)y_{k+1} + \beta_k(x)m_k + \beta_{k+1}(x)m_{k+1} \\
&= \left(1 + 2\frac{x - x_k}{x_{k+1} - x_k}\right)\left(\frac{x - x_{k+1}}{x_k - x_{k+1}}\right)^2 y_k + \left(1 + 2\frac{x - x_{k+1}}{x_k - x_{k+1}}\right)\left(\frac{x - x_k}{x_{k+1} - x_k}\right)^2 y_{k+1} \\
&\quad + (x - x_k)\left(\frac{x - x_{k+1}}{x_k - x_{k+1}}\right)^2 m_k + (x - x_{k+1})\left(\frac{x - x_k}{x_{k+1} - x_k}\right)^2 m_{k+1}
\end{aligned}
$$

Hermite Interpolation

Numerical Example 5

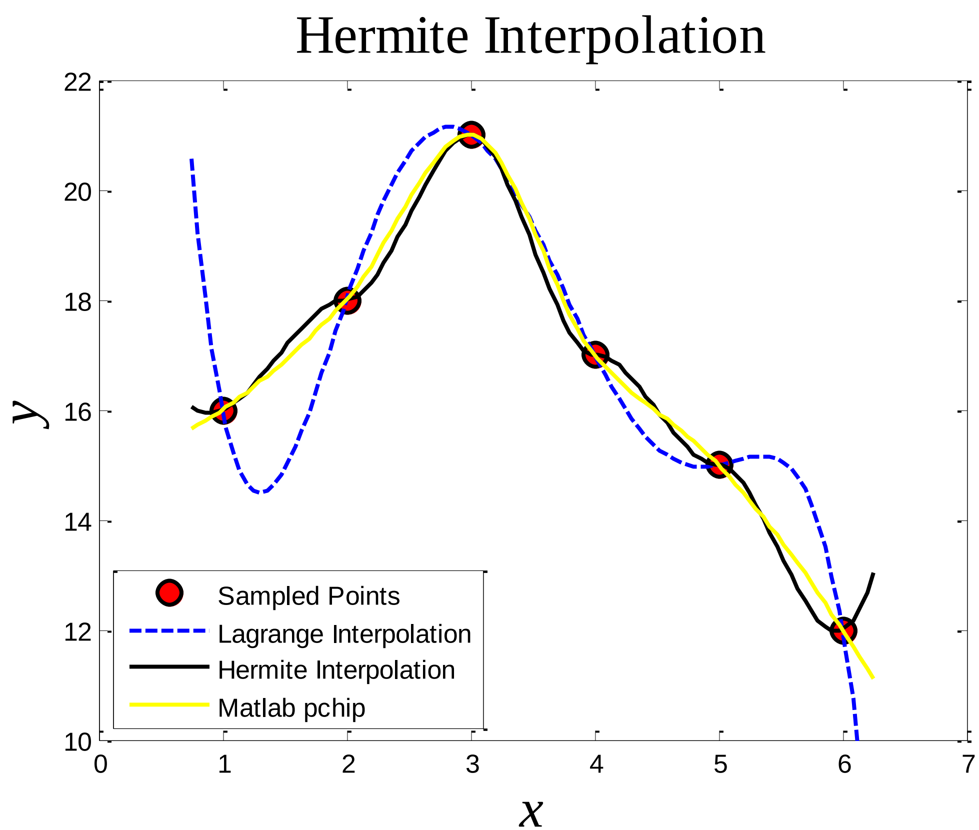
```
x = 1:6;y = [16, 18, 21, 17, 15, 12];dy = [1,0,0,0,0,1];
u = linspace(0.75,6.25,100);v_Her = Hermite(x,y,dy,u);
plot(x,y,'o',u,v_Her, 'k-')
```



Hermite Interpolation

Numerical Example 5: *VS*. PCHIP

```
x = 1:6;y = [16, 18, 21, 17, 15, 12];dy = [1,0,0,0,0,1];
u = linspace(0.75,6.25,100); v_pchip = pchip(x,y,u);
hold on; plot(u,v_pchip,'y-','LineWidth',2)
```



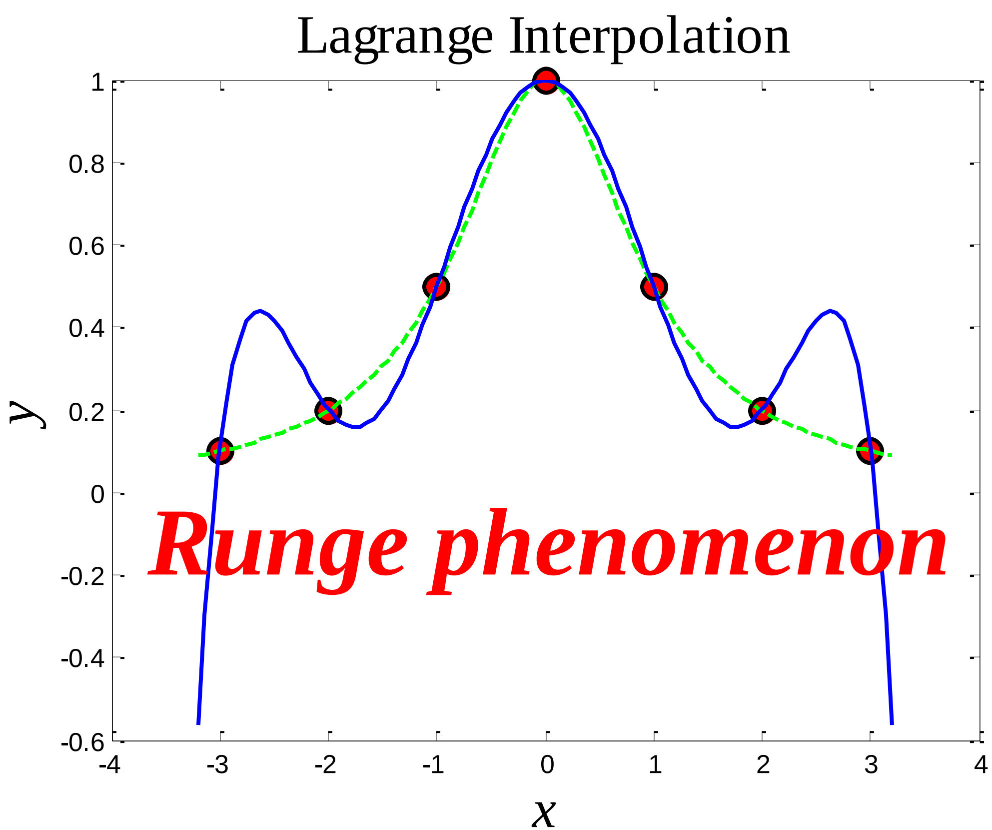
Interpolation Using MATLAB

- **Popular Methods for Interpolations**
  - **Lagrange Interpolation Method**
  - **Newton's Divided Differences**
  - **Hermite Interpolation**
  - **Cubic Spline Interpolation**

Spline Interpolation

- **Motivation:** Consider a function $f(x) = 1/(1 + x^2)$

```
x = -3:1:3;y = 1./(1+x.^2);
u = linspace(-3.2,3.2,100);v = Lagrange(x,y,u);
plot(x,y, 'o', u,1./(1+u.^2),'g--',u,v,'b-')
```



Spline Interpolation

Basic Idea

Given the data points $(x_0, y_0), \ldots, (x_n, y_n)$, $(x_0 < x_1 < \ldots < x_n)$

In each subinterval $[x_i, x_{i+1}]$, ($i = 0,1,\ldots,n-1$) we want to construct the cubic spline:

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

with $S_i(x_i) = y_i$, $S_i(x_{i+1}) = y_{i+1}$, ($i = 0,1,\ldots,n-1$)

$S'_{i-1}(x_i) = S'_i(x_i)$, ($i = 1,\ldots,n-1$)

$S''_{i-1}(x_i) = S''_i(x_i)$, ($i = 1,\ldots,n-1$)

Spline Interpolation

- **Basic Idea: Endpoint conditions**

(1) Natural spline

$$S''_0(x_0) = 0;\ S''_{n-1}(x_n) = 0$$

(2) Clamped cubic spline

$$S'_0(x_0) = v_0;\ S'_{n-1}(x_n) = v_n$$

(3) Not-a-knot cubic spline (MATLAB's default *spline* command)

$$S'''_0(x_1) = S'''_1(x_1)\ ;$$
$$S'''_{n-2}(x_{n-1}) = S'''_{n-1}(x_{n-1})$$

Spline Interpolation

- **Solution Procedure: Method 1**

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

on $[x_i,\ x_{i+1}]$, $(i = 0,1,\ldots,n\text{-}1)$

(1) Constraint #1:

$$S_i(x_{i+1}) = y_{i+1},\ (i = 0,1,\ldots,n\text{-}1)$$

$$\delta_i = x_{i+1} - x_i,\quad \Delta_i = y_{i+1} - y_i$$

$$\Delta_i = \delta_i b_i + \delta_i^2 c_i + \delta_i^3 d_i$$

$$\Delta_i / \delta_i = b_i + \delta_i c_i + \delta_i^2 d_i \qquad (*)$$

Spline Interpolation

- **Solution Procedure: Method 1**

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

on $[x_i,\ x_{i+1}]$, $(i = 0,1,\ldots,n\text{-}1)$

(2) Constraint #2: $S'_{i-1}(x_i) = S'_i(x_i)$, $(i = 1,\ldots,n\text{-}1)$

$$S_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$$

$$x = x_i$$

$$S_i'(x_i) = b_i$$

Spline Interpolation

- **Solution Procedure: Method 1**

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

on $[x_i,\ x_{i+1}]$, $(i = 0,1,\ldots,n\text{-}1)$

(2) Constraint #2: $S'_{i\text{-}1}(x_i) = S'_i(x_i)$, $(i = 1,\ldots,n\text{-}1)$

$$S'_{i-1}(x) = b_{i-1} + 2c_{i-1}(x - x_{i-1}) + 3d_{i-1}(x - x_{i-1})^2$$

$$\Big\downarrow \quad x = x_i$$

$$S'_{i-1}(x_i) = b_{i-1} + 2\delta_{i-1}c_{i-1} + 3\delta_{i-1}^2 d_{i-1} = S'_i(x_i) = b_i \qquad (**)$$

Spline Interpolation

Solution Procedure: Method 1

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

on $[x_i,\ x_{i+1}]$, $(i = 0,1,\ldots,n\text{-}1)$

(3) Constraint #3: $S''_{i-1}(x_i) = S''_i(x_i)$, $(i = 1,\ldots,n\text{-}1)$

$$S_i''(x) = 2c_i + 6d_i(x - x_i)$$

$$\Downarrow \quad x = x_i$$

$$S_i''(x_i) = 2c_i$$

Spline Interpolation

Solution Procedure: Method 1

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$$

on $[x_i,\ x_{i+1}]$, $(i = 0,1,\ldots,n\text{-}1)$

(3) Constraint #3: $S''_{i\text{-}1}(x_i) = S''_i(x_i)$, $(i = 1,\ldots,n\text{-}1)$

$$S''_{i-1}(x) = 2c_{i-1} + 6d_{i-1}(x - x_{i-1})$$

$$\Downarrow\ x = x_i$$

$$S''_{i-1}(x_i) = 2c_{i-1} + 6\delta_{i-1}d_{i-1} = S''_i(x_i) = 2c_i$$

$$\Rightarrow\ d_{i-1} = \frac{1}{3\delta_{i-1}}(c_i - c_{i-1}) \qquad \Rightarrow\ \boxed{d_i = \frac{1}{3\delta_i}(c_{i+1} - c_i)}$$

Spline Interpolation

- **Solution Procedure: Method 1**

Substituting $\boxed{d_i = \frac{1}{3\delta_i}(c_{i+1} - c_i)}$ **into (*)**

$$\Delta_i / \delta_i = b_i + \delta_i c_i + \delta_i^2 d_i$$

$$\underline{b_i} = \Delta_i / \delta_i - \delta_i c_i - \frac{\delta_i}{3}(c_{i+1} - c_i)$$

$$\underline{= \Delta_i / \delta_i - \frac{2}{3}\delta_i c_i - \frac{\delta_i}{3} c_{i+1}}$$

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Spline Interpolation

Solution Procedure: Method 1

Substituting $\boxed{d_i = \dfrac{1}{3\delta_i}(c_{i+1} - c_i)}$ **into (**)**

$$\boxed{b_i = \Delta_i / \delta_i - \frac{2}{3}\delta_i c_i - \frac{\delta_i}{3} c_{i+1}}$$

$$b_{i-1} + 2\delta_{i-1}c_{i-1} + 3\delta_{i-1}^2 d_{i-1} = b_i$$

$$\Downarrow$$

$$\delta_{i-1}c_{i-1} + 2\left(\delta_{i-1} + \delta_i\right)c_i + \delta_i c_{i+1} = 3\left(\frac{\Delta_i}{\delta_i} - \frac{\Delta_{i-1}}{\delta_{i-1}}\right)$$

Spline Interpolation

- **Solution Procedure: Method 1**

$$\delta_{i-1} c_{i-1} + 2\left(\delta_{i-1} + \delta_i\right) c_i + \delta_i \, c_{i+1} = 3\left(\frac{\Delta_i}{\delta_i} - \frac{\Delta_{i-1}}{\delta_{i-1}}\right)$$

(i = 1,…,n-1)

$$\begin{bmatrix} \delta_0 & 2(\delta_0+\delta_1) & \delta_1 & & & \\ & \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & & \\ & & & \ddots & \ddots & \\ & & & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 3 \begin{bmatrix} \frac{\Delta_1}{\delta_1} - \frac{\Delta_0}{\delta_0} \\ \frac{\Delta_2}{\delta_2} - \frac{\Delta_1}{\delta_1} \\ \vdots \\ \frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}} \end{bmatrix}$$

(n-1)×(n+1)

Spline Interpolation

Solution Procedure: Method 1

$$\begin{bmatrix} \delta_0 & 2(\delta_0+\delta_1) & \delta_1 & & & \\ & \delta_1 & 2(\delta_1+\delta_2) & \delta_2 & & \\ & & & \ddots & \ddots & \\ & & & \delta_{n-2} & 2(\delta_{n-2}+\delta_{n-1}) & \delta_{n-1} \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix} = 3 \begin{bmatrix} \frac{\Delta_1}{\delta_1} - \frac{\Delta_0}{\delta_0} \\ \frac{\Delta_2}{\delta_2} - \frac{\Delta_1}{\delta_1} \\ \vdots \\ \frac{\Delta_{n-1}}{\delta_{n-1}} - \frac{\Delta_{n-2}}{\delta_{n-2}} \end{bmatrix}$$

(n-1)×(n+1)

➔ **Natural spline**

$$S''_0(x_0) = 0; \; S''_{n-1}(x_n) = 0$$

$$\begin{cases} 2c_0 = 0 \\ 2c_n = 0 \end{cases}$$

Spline Interpolation

Solution Procedure: Method 1

➔ Clamped cubic spline

$$S'_0(x_0) = v_0;\ S'_{n-1}(x_n) = v_n$$

$$\begin{cases} 2\delta_0 c_0 + \delta_0 c_1 = 3\left(\dfrac{\Delta_0}{\delta_0} - v_0\right) \\ \delta_{n-1} c_{n-1} + 2\delta_{n-1} c_n = 3\left(v_n - \dfrac{\Delta_{n-1}}{\delta_{n-1}}\right) \end{cases}$$

Spline Interpolation

- **Solution Procedure: Method 2**

Since the spline is of degree 3, its second-order derivative must be continuous. Introduce the following notation:

$$M_j = S''(x_j),\ j = 0,1,\cdots,n$$

On the interval [x_j,x_{j+1}], $S_j''(x)$ **is linear:**

$$\boxed{S_j''(x) = M_j \frac{x_{j+1} - x}{h_j} + M_{j+1} \frac{x - x_j}{h_j}}$$

where $h_j = x_{j+1} - x_j$

Spline Interpolation

Solution Procedure: Method 2

$$\boxed{S_j''(x) = M_j \frac{x_{j+1} - x}{h_j} + M_{j+1} \frac{x - x_j}{h_j}} \qquad j = 0, 1, \cdots, n-1$$

Integrating it twice and use $S_j(x_j) = y_j, S_j(x_{j+1}) = y_{j+1}$

$$\boxed{\begin{aligned} S_j(x) = {} & M_j \frac{(x_{j+1} - x)^3}{6h_j} + M_{j+1} \frac{(x - x_j)^3}{6h_j} + \left(y_j - \frac{M_j h_j^2}{6} \right) \frac{x_{j+1} - x}{h_j} \\ & + \left(y_{j+1} - \frac{M_{j+1} h_j^2}{6} \right) \frac{x - x_j}{h_j} \end{aligned}}$$

Spline Interpolation

Solution Procedure: Method 2

$$S_j(x) = M_j \frac{(x_{j+1}-x)^3}{6h_j} + M_{j+1}\frac{(x-x_j)^3}{6h_j} + \left(y_j - \frac{M_j h_j^2}{6}\right)\frac{x_{j+1}-x}{h_j} + \left(y_{j+1} - \frac{M_{j+1} h_j^2}{6}\right)\frac{x-x_j}{h_j}$$

The first derivatives:

$$S_j'(x) = -M_j \frac{(x_{j+1}-x)^2}{2h_j} + M_{j+1}\frac{(x-x_j)^2}{2h_j} + \frac{y_{j+1}-y_j}{h_j} - \frac{M_{j+1}-M_j}{6}h_j$$

Spline Interpolation

Solution Procedure: Method 2

$$S_j'(x) = -M_j \frac{(x_{j+1} - x)^2}{2h_j} + M_{j+1} \frac{(x - x_j)^2}{2h_j} + \frac{y_{j+1} - y_j}{h_j} - \frac{M_{j+1} - M_j}{6} h_j$$

We obtain:

$$S_j'(x_j) = -\frac{h_j}{3} M_j - \frac{h_j}{6} M_{j+1} + \frac{y_{j+1} - y_j}{h_j}$$

Spline Interpolation

- **Solution Procedure : Method 2**

$$\boxed{S_j'(x_j) = -\frac{h_j}{3}M_j - \frac{h_j}{6}M_{j+1} + \frac{y_{j+1} - y_j}{h_j}}$$

Similarly, we have

$$\boxed{S_{j-1}'(x_j) = \frac{h_{j-1}}{6}M_{j-1} + \frac{h_{j-1}}{3}M_j + \frac{y_j - y_{j-1}}{h_{j-1}}}$$

Spline Interpolation

Solution Procedure: Method 2

Using $S_j'(x_j) = S_{j-1}'(x_j)$ $\qquad f[x_j, x_{j+1}]$

$$S_j'(x_j) = -\frac{h_j}{3}M_j - \frac{h_j}{6}M_{j+1} + \boxed{\frac{y_{j+1}-y_j}{h_j}}$$

$$S_{j-1}'(x_j) = \frac{h_{j-1}}{6}M_{j-1} + \frac{h_{j-1}}{3}M_j + \boxed{\frac{y_j - y_{j-1}}{h_{j-1}}}$$

$f[x_{j-1}, x_j]$

Spline Interpolation

Solution Procedure: Method 2

$$\mu_j M_{j-1} + 2M_j + \lambda_j M_{j+1} = d_j, \quad j = 1, 2, \cdots, n-1$$

where

$$\mu_j = \frac{h_{j-1}}{h_{j-1} + h_j}, \quad \lambda_j = \frac{h_j}{h_{j-1} + h_j}$$

$$d_j = 6 \frac{f[x_j, x_{j+1}] - f[x_{j-1}, x_j]}{h_{j-1} + h_j} = 6 f[x_{j-1}, x_j, x_{j+1}]$$

(n + 1) unknows, while (n – 1) equations.

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Spline Interpolation

Solution Procedure: Method 2

For the clamped cubic spline

$$S'_0(x_0) = v_0;\ S'_{n-1}(x_n) = v_n$$

we have

$$2M_0 + M_1 = \frac{6}{h_0}\left(f[x_0, x_1] - v_0\right) \triangleq d_0$$

$$M_{n-1} + 2M_n = \frac{6}{h_{n-1}}\left(v_n - f[x_{n-1}, x_n]\right) \triangleq d_n$$

Spline Interpolation

- **Solution Procedure: Method 2**

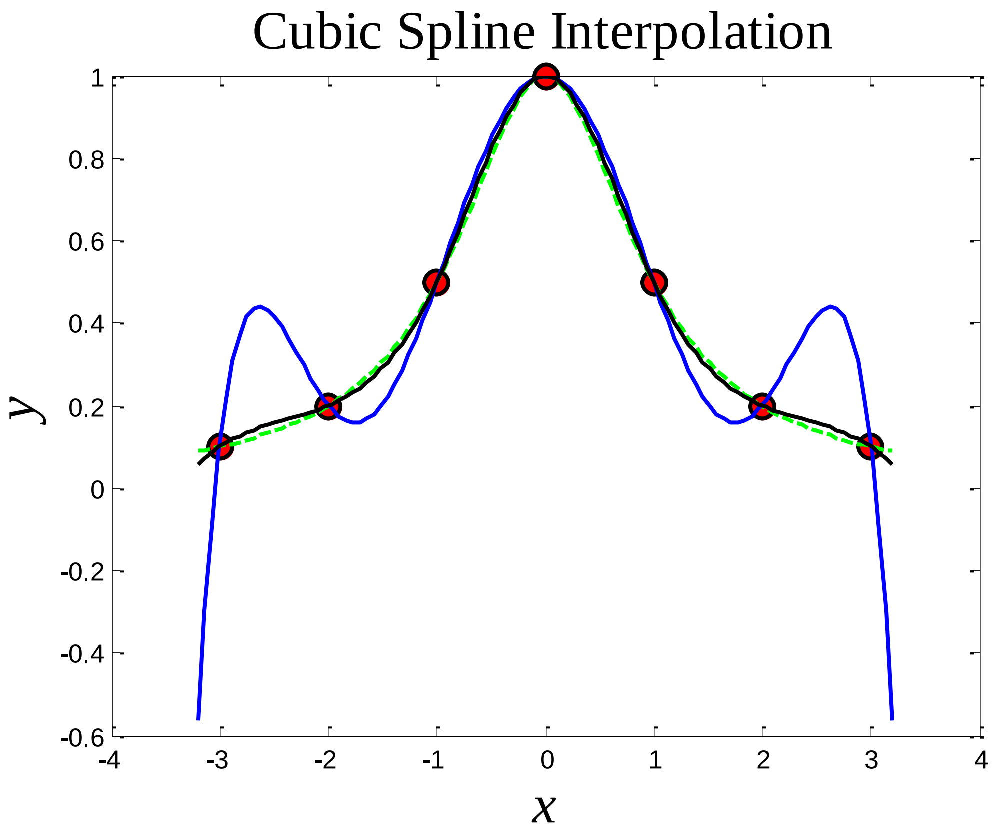
For the clamped cubic spline, the spline interpolation can be obtained from:

$$\begin{bmatrix} 2 & \lambda_0 & & & \\ \mu_1 & 2 & \lambda_1 & & \\ & \ddots & \ddots & \ddots & \\ & & \mu_{n-1} & 2 & \lambda_{n-1} \\ & & & \mu_n & 2 \end{bmatrix} \begin{bmatrix} M_0 \\ M_1 \\ \vdots \\ M_{n-1} \\ M_n \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}$$

Spline Interpolation

- Re-Consider the function $f(x) = 1/(1 + x^2)$

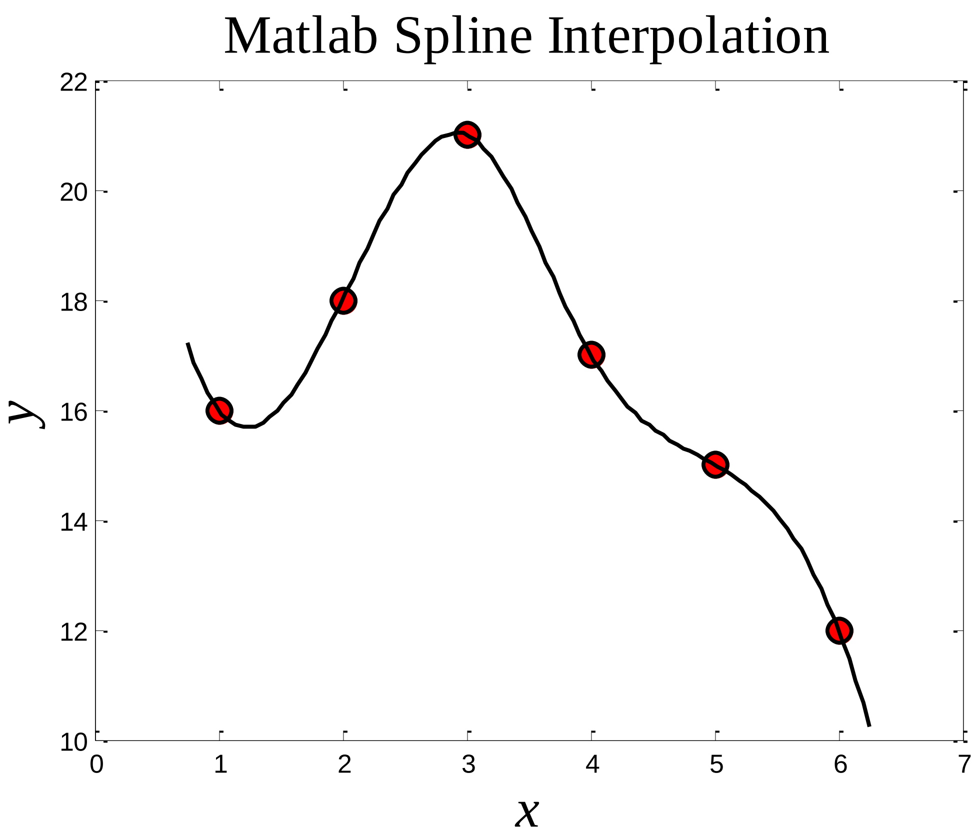
```
x = -3:1:3;y = 1./(1+x.^2);
u = linspace(-3.2,3.2,100); v_Spl = spline(x,y,u);
hold on; plot(u,v_Spl,'k-','LineWidth',2)
```



Spline Interpolation

Numerical Example 6

```
x = 1:6; y = [16, 18, 21, 17, 15, 12];
u = linspace(0.75,6.25,100); v = spline(x,y,u);
plot(x,y, 'o',u,v,‘k-')
```

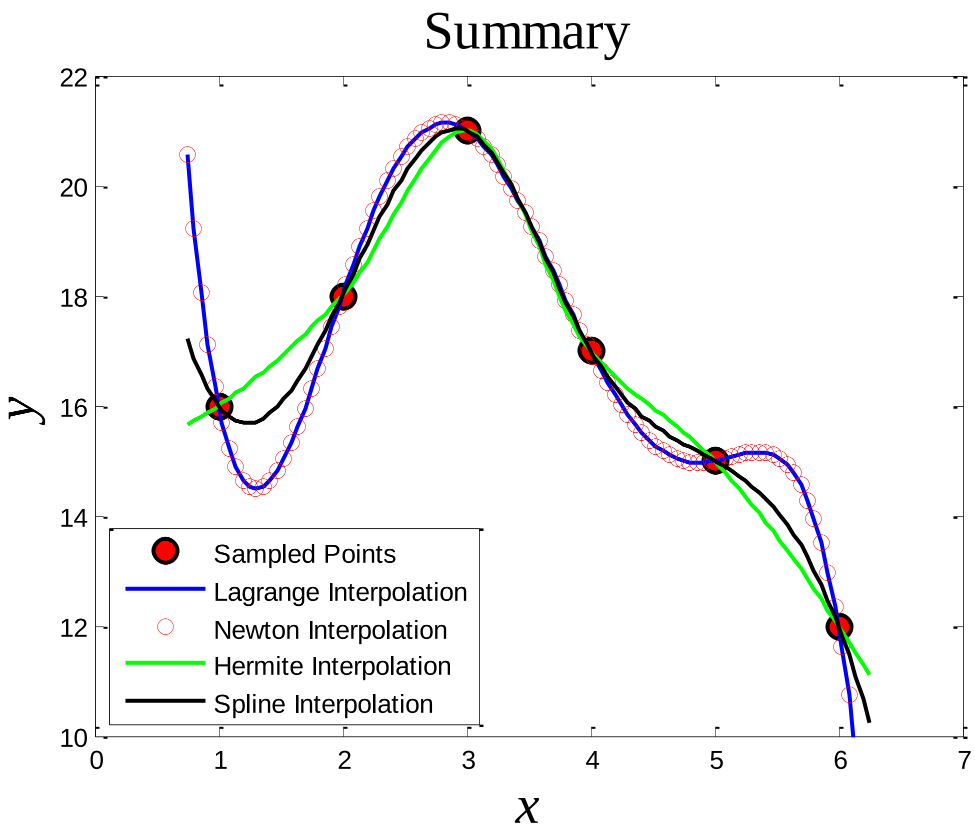


MATLAB Built-in Functions

- **MATLAB Built-in Functions for Interpolation**
  - ✓ **1-D data interpolation: *interp1***
  - ✓ **2-D data interpolation: *interp2***
  - ✓ **Cubic spline data interpolation: *spline***
  - ✓ **Polynomial evaluation: *polyval***

Summary

- ✓ **Lagrange Interpolation Method**
- ✓ **Newton's Divided Differences**
- ✓ **Hermite Interpolation**
- ✓ **Cubic Spline Interpolation**



Thank You !