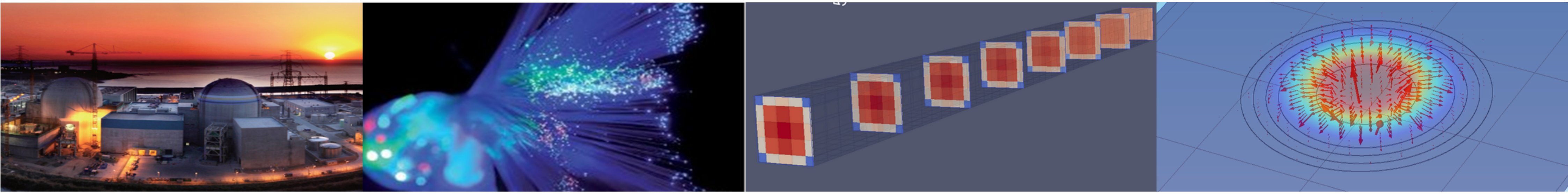


# INTRODUCTION TO NUMERICAL ANALYSIS


***Cho, Hyoung Kyu***

***Department of Nuclear Engineering***

***Seoul National University***

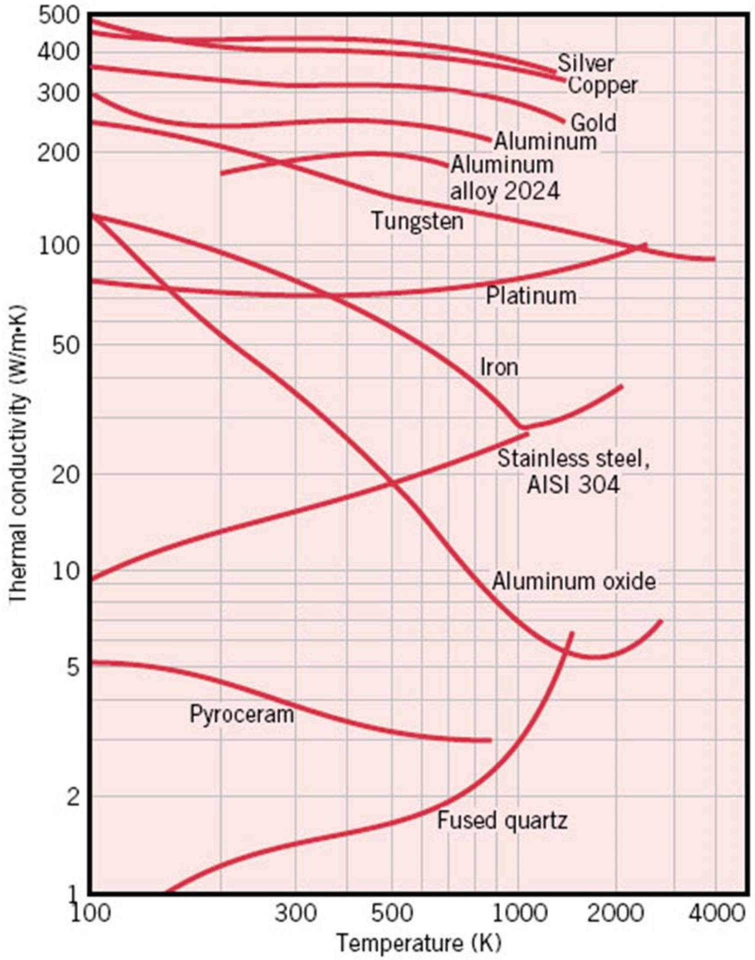
# 6. CURVE FITTING AND INTERPOLATION

## Finding eigenvalues

- QR factorization and iterative methods

## HW3 관련

- 열전도도 vs. 온도
- Iterative method !



## ❖ Crout's method with partial pivoting

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} = \begin{bmatrix} L_{11} & (L_{11}U_{12}) & (L_{11}U_{13}) & (L_{11}U_{14}) \\ L_{21} & (L_{21}U_{12}+L_{22}) & (L_{21}U_{13}+L_{22}U_{23}) & (L_{21}U_{14}+L_{22}U_{24}) \\ L_{31} & (L_{31}U_{12}+L_{32}) & (L_{31}U_{13}+L_{32}U_{23}+L_{33}) & (L_{31}U_{14}+L_{32}U_{24}+L_{33}U_{34}) \\ L_{41} & (L_{41}U_{12}+L_{42}) & (L_{41}U_{13}+L_{42}U_{23}+L_{43}) & (L_{41}U_{14}+L_{42}U_{24}+L_{43}U_{34}+L_{44}) \end{bmatrix}$$

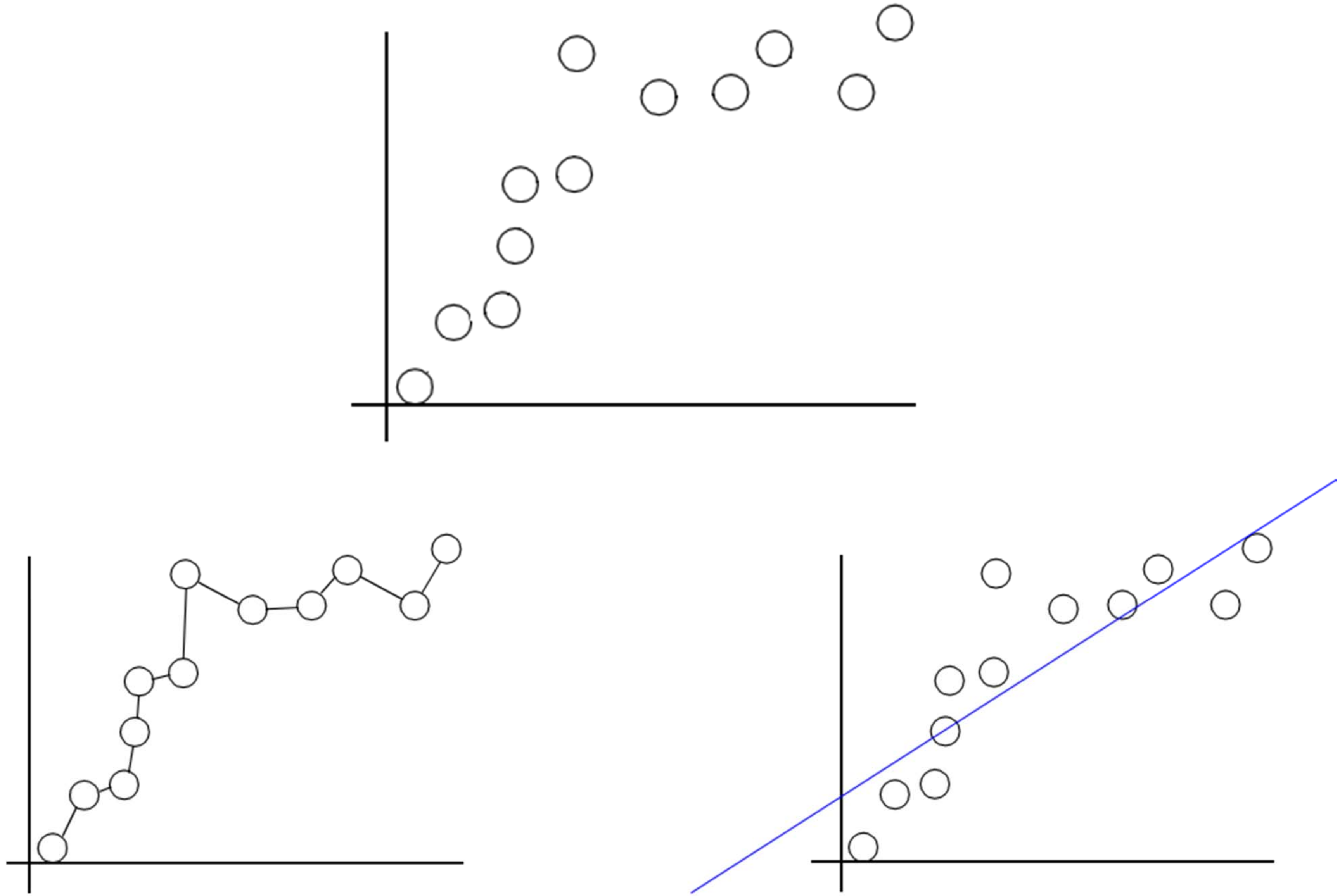
$L_{11} = a_{11}$ $\quad U_{12} = \dfrac{a_{12}}{L_{11}}$ $\quad U_{13} = \dfrac{a_{13}}{L_{11}}$ $\quad U_{14} = \dfrac{a_{14}}{L_{11}}$

$L_{21} = a_{21}$ $\quad L_{22} = a_{22} - L_{21}U_{12}$ $\quad U_{23} = \dfrac{a_{23} - L_{21}U_{13}}{L_{22}}$ and $U_{24} = \dfrac{a_{24} - L_{21}U_{14}}{L_{22}}$

$L_{31} = a_{31}$, $\quad L_{32} = a_{32} - L_{31}U_{12}$, and $L_{33} = a_{33} - L_{31}U_{13} - L_{32}U_{23}$ $\quad U_{34} = \dfrac{a_{34} - L_{31}U_{14} - L_{32}U_{24}}{L_{33}}$

$L_{41} = a_{41}$, $\quad L_{42} = a_{42} - L_{41}U_{12}$, $\quad L_{43} = a_{43} - L_{41}U_{13} - L_{42}U_{23}$, $L_{44} = a_{44} - L_{41}U_{14} - L_{42}U_{24} - L_{43}U_{34}$

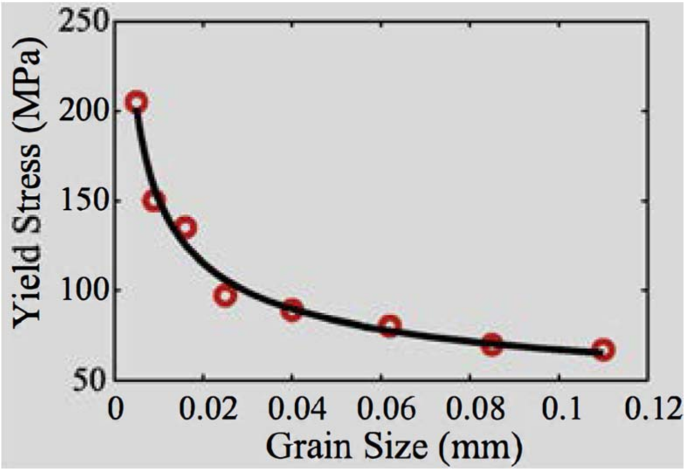
# Curve fitting vs. interpolation



## Curve fitting

- Experimental data

| $d$ (mm) | 0.005 | 0.009 | 0.016 | 0.025 | 0.040 | 0.062 | 0.085 | 0.110 |
|---|---|---|---|---|---|---|---|---|
| $\sigma_y$ (MPa) | 205 | 150 | 135 | 97 | 89 | 80 | 70 | 67 |



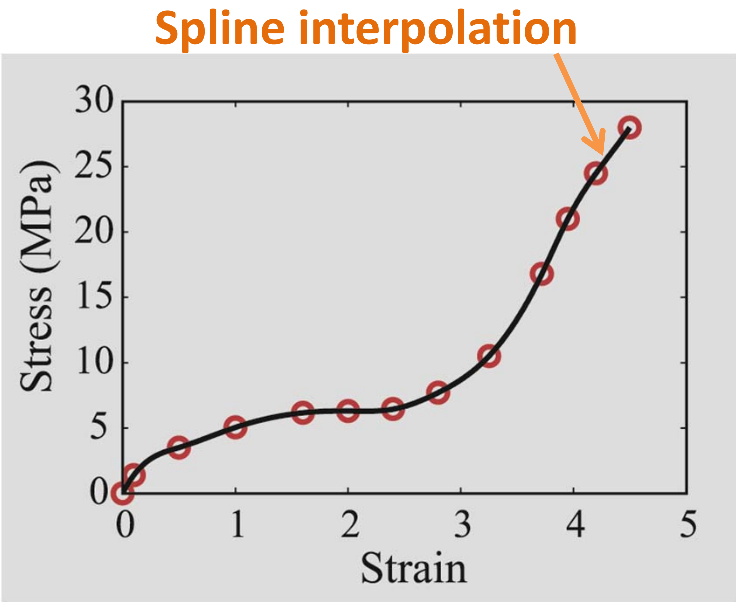

- Objective
  - To find a function that fits the data points overall
  - The function does not have to give the exact value at any single point, but fits the data well overall.
  - Is typically used when the values of the data points have some error or scatter.
    - All experimental measurements have built-in errors or uncertainties.
- Curve fitting can be carried out with many types of functions and with polynomials of various orders.

## Interpolation

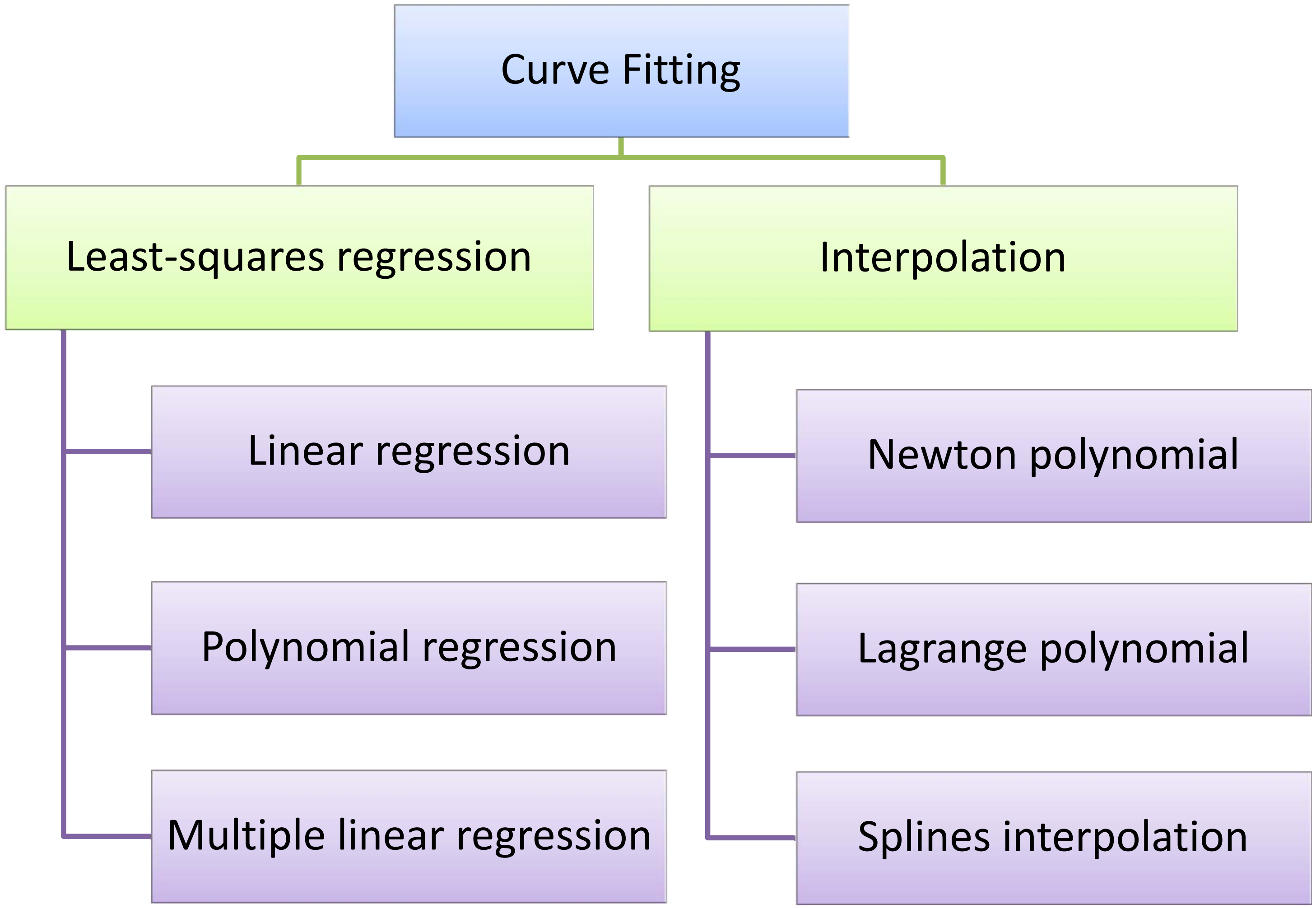
- Procedure for estimating a value between known values of data points.
  - Done by first determining a polynomial that gives the exact value at the data points
- When a small number of points is involved
  - Single polynomial might be sufficient
- When a large number of points are involved
  - Different polynomials are used in the intervals between the points.
  - **Spline interpolation**

## Extrapolation

- To predict how the data might extend beyond the range over which it was measured.

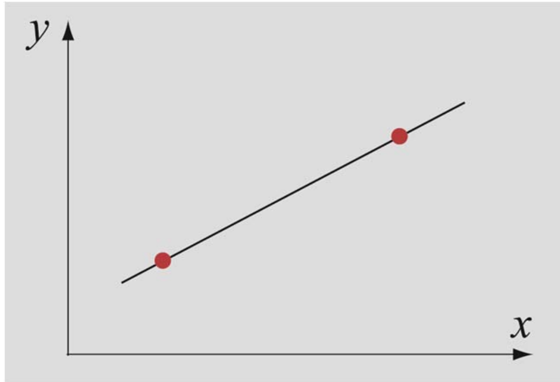


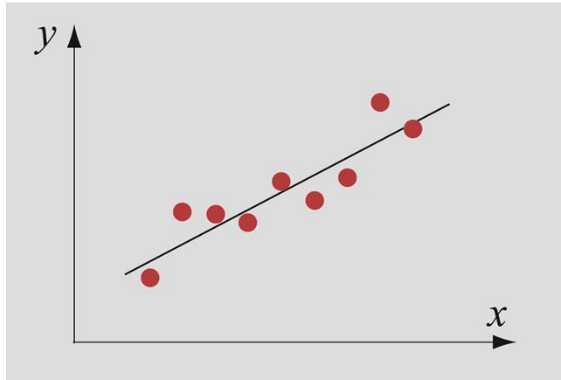
## ❖ Overview of this chapter



## Curve fitting using a linear equation

$$y = a_1 x + a_0$$




- With two data points
  - The constant can be determined that gives the exact values at the points.
- With more than two data points
  - Constant $a_1$ and $a_0$ are determined such that the line has the **best fit overall (?)**.
- The definition of best fit
- A mathematical procedure for deriving the values of the constants

## Measuring how good is fit

- To quantify the overall agreement between the points and the function
- Procedure
  - Calculate the error (**residual,** the difference between a data point and the predicted value)
  - Calculate a total error using the residuals

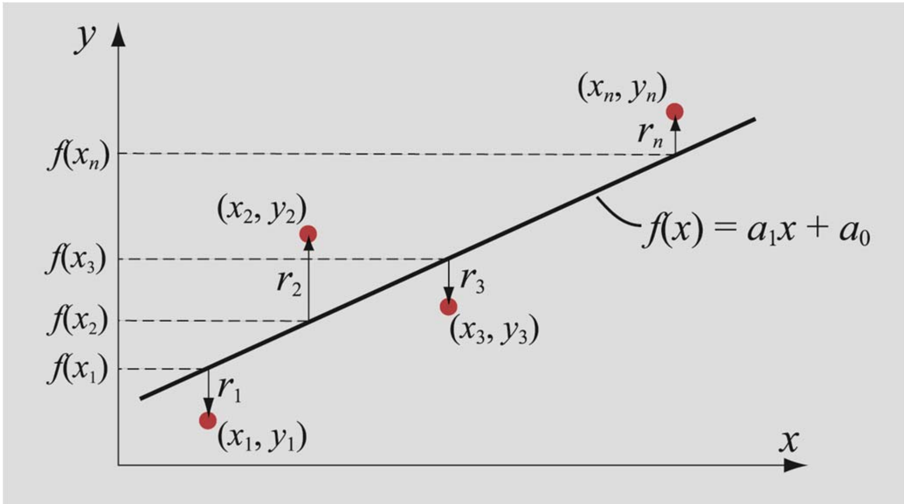
- Residual ($r_i$): $r_i = y_i - f(x_i)$

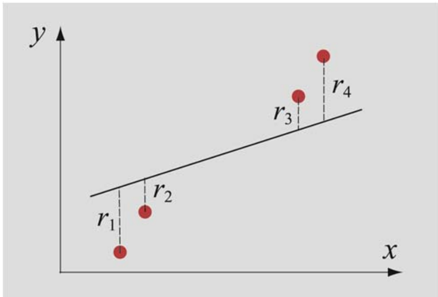
- Total error ($E$)

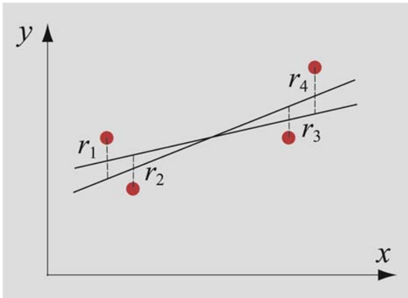
$$E = \sum_{i=1}^{n} r_i = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]$$

$$E = \sum_{i=1}^{n} |r_i| = \sum_{i=1}^{n} |y_i - (a_1 x_i + a_0)|$$

$E =$



## ❖ Linear Least-Squares Regression

- A procedure to determine the coefficients $a_1$ and $a_0$.
- Best fit
  - The smallest possible total error calculated by adding the squares of the residuals

$$E = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]^2$$

- Systems of two linear equations

$$na_0\left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} x_i\right)^2 a_1 = \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

$$na_0\left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} x_i^2\right) na_1 = n\left(\sum_{i=1}^{n} x_i y_i\right)$$

# 6.2 Curve Fitting with a Linear Equation

## ❖ Linear Least-Squares Regression

- Systems of two linear equations

$$na_0\left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} x_i\right)^2 a_1 = \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)$$

$$na_0\left(\sum_{i=1}^{n} x_i\right) + \left(\sum_{i=1}^{n} x_i^2\right) na_1 = n\left(\sum_{i=1}^{n} x_i y_i\right)$$

$$a_1 = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

$$a_0 = \frac{\left(\sum_{i=1}^{n} x_i^2\right)\left(\sum_{i=1}^{n} y_i\right) - \left(\sum_{i=1}^{n} x_i y_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2}$$

- Simplified form with the summations

$$S_x = \sum_{i=1}^{n} x_i, \quad S_y = \sum_{i=1}^{n} y_i, \quad S_{xy} = \sum_{i=1}^{n} x_i y_i, \quad S_{xx} = \sum_{i=1}^{n} x_i^2$$

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} \qquad a_0 = \frac{S_{xx}S_y - S_{xy}S_x}{nS_{xx} - (S_x)^2}$$

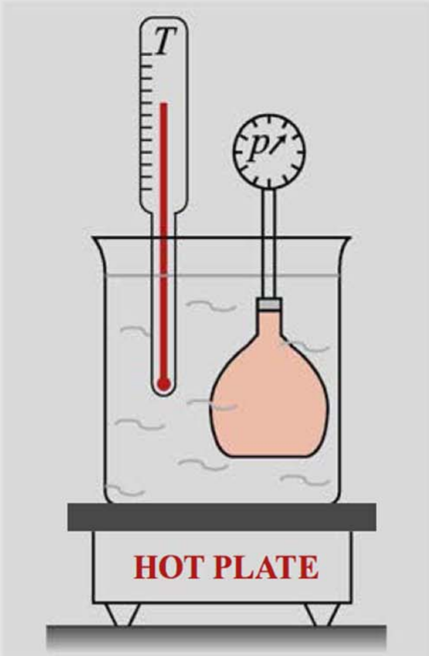
## ❖ Example 6-1

**Example 6-1: Determination of absolute zero temperature.**

According to Charles's law for an ideal gas, at constant volume, a linear relationship exists between the pressure, $p$, and temperature, $T$. In the experiment shown in the figure, a fixed volume of gas in a sealed container is submerged in ice water ($T$ = 0°C). The temperature of the gas is then increased in ten increments up to $T$ = 100°C by heating the water, and the pressure of the gas is measured at each temperature. The data from the experiment is:

| $T$ (°C) | 0 | 10 | 20 | 30 | 40 | 50 | 60 |
|---|---|---|---|---|---|---|---|
| $p$ (atm.) | 0.94 | 0.96 | 1.0 | 1.05 | 1.07 | 1.09 | 1.14 |

| $T$ (°C) | 70 | 80 | 90 | 100 |
|---|---|---|---|---|
| $p$ (atm.) | 1.17 | 1.21 | 1.24 | 1.28 |




Extrapolate the data to determine the absolute zero temperature, $T_0$. This can be done using the following steps:

(*a*) Make a plot of the data ($p$ versus $T$).

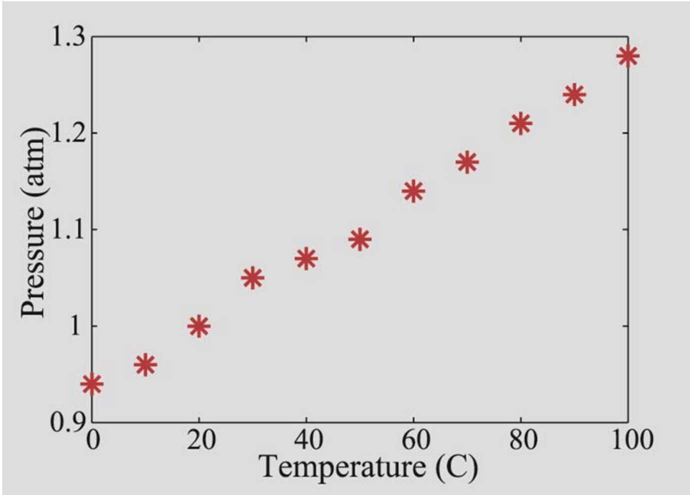
(*b*) Use linear least-squares regression to determine a linear function in the form $p = a_1T + a_0$ that best fits the data points. First calculate the coefficients by hand using only the four data points: 0, 30, 70, and 100°C. Then write a user-defined MATLAB function that calculates the coefficients of the linear function for any number of data points and use it with all the data points to determine the coefficients.

(*c*) Plot the function, and extend the line (extrapolate) until it crosses the horizontal ($T$) axis. This point is an estimate of the absolute zero temperature, $T_0$. Determine the value of $T_0$ from the function.

## Example 6-1

(a)

```
>> T=0:10:100;
p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
>> plot(T,p,'*r')
```




(b) Hand calculation

Data: $(0, 0.94), (30, 1.05), (70, 1.17), (100, 1.28)$

Function form: $p = a_1\, T + a_0$

$$S_x = \sum_{i=1}^{4} x_i = 0 + 30 + 70 + 100 = 200 \qquad S_{xx} = \sum_{i=1}^{4} x_i^2 = 0^2 + 30^2 + 70^2 + 100^2 = 15800$$

$$S_y = \sum_{i=1}^{4} y_i = 0.94 + 1.05 + 1.17 + 1.28 = 4.44 \qquad S_{xy} = \sum_{i=1}^{4} x_i y_i = 0 \cdot 0.94 + 30 \cdot 1.05 + 70 \cdot 1.17 + 100 \cdot 1.28 = 241.4$$

$$a_1 = \frac{nS_{xy} - S_x S_y}{nS_{xx} - (S_x)^2} = \frac{4 \cdot 241.4 - (200 \cdot 4.44)}{4 \cdot 15800 - 200^2} = 0.003345$$

$$a_0 = \frac{S_{xx} S_y - S_{xy} S_x}{nS_{xx} - (S_x)^2} = \frac{15800 \cdot 4.44 - (241.4 \cdot 200)}{4 \cdot 15800 - 200^2} = 0.9428$$

## Example 6-1

(b) MATLAB programming

```
function [a1,a0] = LinearRegression(x, y)
% LinearRegration calculates the coefficients a1 and a0 of the linear
% equation y = a1*x + a0 that best fit n data points.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Output variable:
% a1   The coefficient a1.
% a0   The coefficient a0.

nx = length(x);
ny = length(y);
if nx ~= ny
    disp('ERROR: The number of elements in x must be the same as in y.')
    a1 = 'Error';
    a0 = 'Error';
else
Sx = sum(x);
Sy = sum(y);
Sxy = sum(x.*y);
Sxx = sum(x.^2);
a1 = (nx*Sxy - Sx*Sy)/(nx*Sxx - Sx^2);
a0 = (Sxx*Sy - Sxy*Sx)/(nx*Sxx - Sx^2);
end
```

### Example 6-1

(b) MATLAB programming

```
T=0:10:100;
p=[0.94 0.96 1.0 1.05 1.07 1.09 1.14 1.17 1.21 1.24 1.28];
[a1,a0]=LinearRegression(T,p);
a1
a0
```

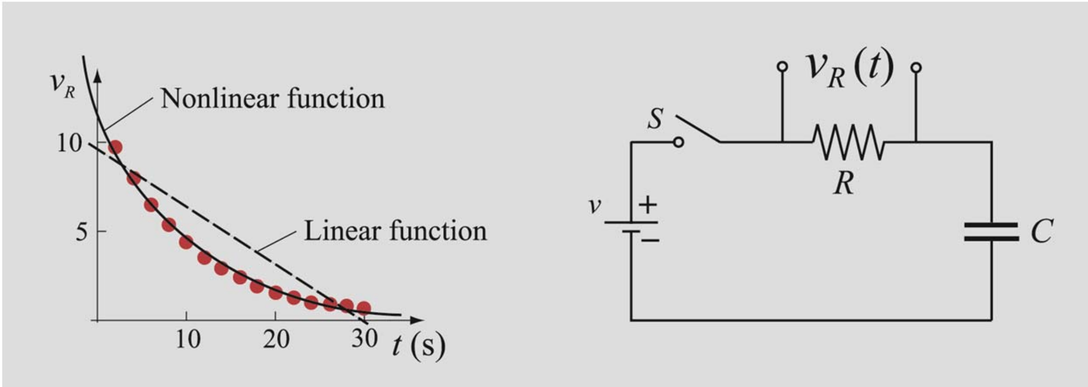
(c) Plots of the function, the points, and calculates the value of $T_0$ from the function

```
Tplot=[-300 100];
pplot=a1*Tplot+a0;
plot(T,p, '*r' ,'markersize' ,12)
hold on
plot(Tplot,pplot, 'k')
xlabel('Temperature (C) ','fontsize' ,20)
ylabel('Pressure (atm)' ,'fontsize' ,20)
T0=-0.9336/0.0034
```

# 6.3 Curve Fitting with Nonlinear Equation

## ❖ Non-linear systems

- Ex) RC circuit




## ❖ Non-linear curve fitting by writing the equation in a linear form

- Curve fitting with nonlinear functions that can be written in a form for which the linear least-squares regression method can be used

$$y = bx^m$$

$$y = be^{mx} \text{ or } y = b10^{mx}$$

$$y = \frac{1}{mx + b}$$

## Writing a non-linear equation in a linear form

- Ex) power function

$$y = bx^m$$

  - Taking the natural logarithm

$$\ln(y) = \ln(bx^m) = m\ln(x) + \ln(b)$$

$$\underbrace{\ln(y)}_{Y} = \underbrace{m}_{a_1}\underbrace{\ln(x)}_{X} + \underbrace{\ln(b)}_{a_0}$$

$a_1$ $and$ $a_0$ can be determined by linear least-square regression

$$m = a_1 \text{ and } b = e^{(a_0)}$$

## ❖ Writing a non-linear equation in a linear form

- Transforming nonlinear equations to linear form

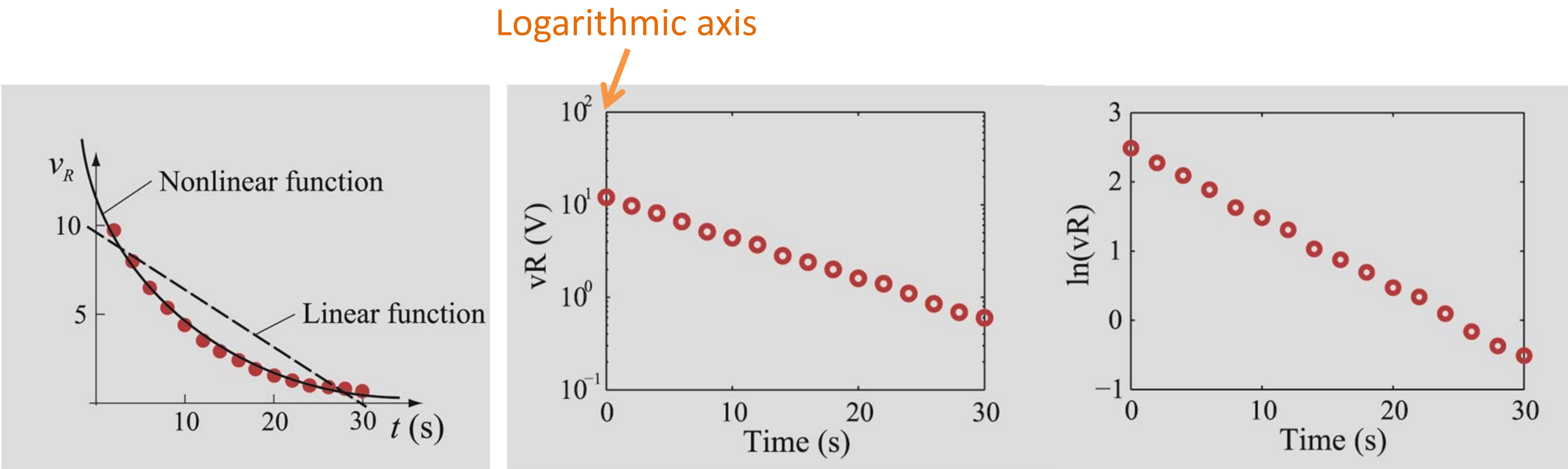
| Nonlinear equation | Linear form | Relationship to $Y = a_1X + a_0$ | Values for linear least-squares regression | Plot where data points appear to fit a straight line |
|---|---|---|---|---|
| $y = bx^m$ | $\ln(y) = m\ln(x) + \ln(b)$ | $Y = \ln(y)$, $X = \ln(x)$<br>$a_1 = m$, $a_0 = \ln(b)$ | $\ln(x_i)$ and $\ln(y_i)$ | $y$ vs. $x$ plot on logarithmic $y$ and $x$ axes.<br>$\ln(y)$ vs. $\ln(x)$ plot on linear $x$ and y axes. |
| $y = be^{mx}$ | $\ln(y) = mx + \ln(b)$ | $Y = \ln(y)$, $X = x$<br>$a_1 = m$, $a_0 = \ln(b)$ | $x_i$ and $\ln(y_i)$ | $y$ vs. $x$ plot on logarithmic $y$ and linear $x$ axes.<br>$\ln(y)$ vs. $x$ plot on linear $x$ and $y$ axes. |
| $y = b10^{mx}$ | $\log(y) = mx + \log(b)$ | $Y = \log(y)$, $X = x$<br>$a_1 = m$, $a_0 = \log(b)$ | $x_i$ and $\log(y_i)$ | $y$ vs. $x$ plot on logarithmic $y$ and linear $x$ axes.<br>$\log(y)$ vs. $x$ plot on linear $x$ and $y$ axes. |
| $y = \dfrac{1}{mx+b}$ | $\dfrac{1}{y} = mx + b$ | $Y = \dfrac{1}{y}$, $X = x$<br>$a_1 = m$, $a_0 = b$ | $x_i$ and $1/y_i$ | $1/y$ vs. $x$ plot on linear $x$ and $y$ axes. |
| $y = \dfrac{mx}{b+x}$ | $\dfrac{1}{y} = \dfrac{b}{m}\dfrac{1}{x} + \dfrac{1}{m}$ | $Y = \dfrac{1}{y}$, $X = \dfrac{1}{x}$<br>$a_1 = \dfrac{b}{m}$, $a_0 = \dfrac{1}{m}$ | $1/x_i$ and $1/y_i$ | $1/y$ vs. $1/x$ plot on linear $x$ and $y$ axes. |

# 6.3 Curve Fitting with Nonlinear Equation

## How to choose an appropriate nonlinear function for curve fitting

- Based on a guiding theory of the physical phenomena
- If there is no knowledge of a possible form of the equation, choosing the most appropriate nonlinear function to curve-fit given data may be more difficult.

- Ex)



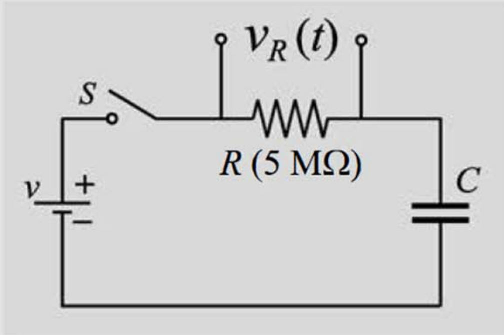

- Exponential functions cannot pass through the origin.
- Exponential functions can only fit data with all positive $y$s, or all negative $y$s.
- Logarithmic functions cannot include $x = 0$ or negative values of $x$.
- For power function $y = 0$ when $x = 0$.
- The reciprocal equation cannot include $y = 0$.

# 6.3 Curve Fitting with Nonlinear Equation

## Example 6-2

**Example 6-2: Curve fitting with a nonlinear function by writing the equation in a linear form.**

An experiment with an *RC* circuit is used for determining the capacitance of an unknown capacitor. In the circuit, shown on the right and in Fig. 6-8, a 5-MΩ resistor is connected in series to the unknown capacitor *C* and a battery. The experiment starts by closing the switch and measuring the voltages, $v_R$, across the resistor every 2 seconds for 30 seconds. The data measured in the experiment is:




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $t$ (s) | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 |
| $v_R$ (V) | 9.7 | 8.1 | 6.6 | 5.1 | 4.4 | 3.7 | 2.8 | 2.4 | 2.0 |

| | | | | | | |
|---|---|---|---|---|---|---|
| $t$ (s) | 20 | 22 | 24 | 26 | 28 | 30 |
| $v_R$ (V) | 1.6 | 1.4 | 1.1 | 0.85 | 0.69 | 0.6 |

Theoretically, the voltage across the resistor as a function of time is given by the exponential function:

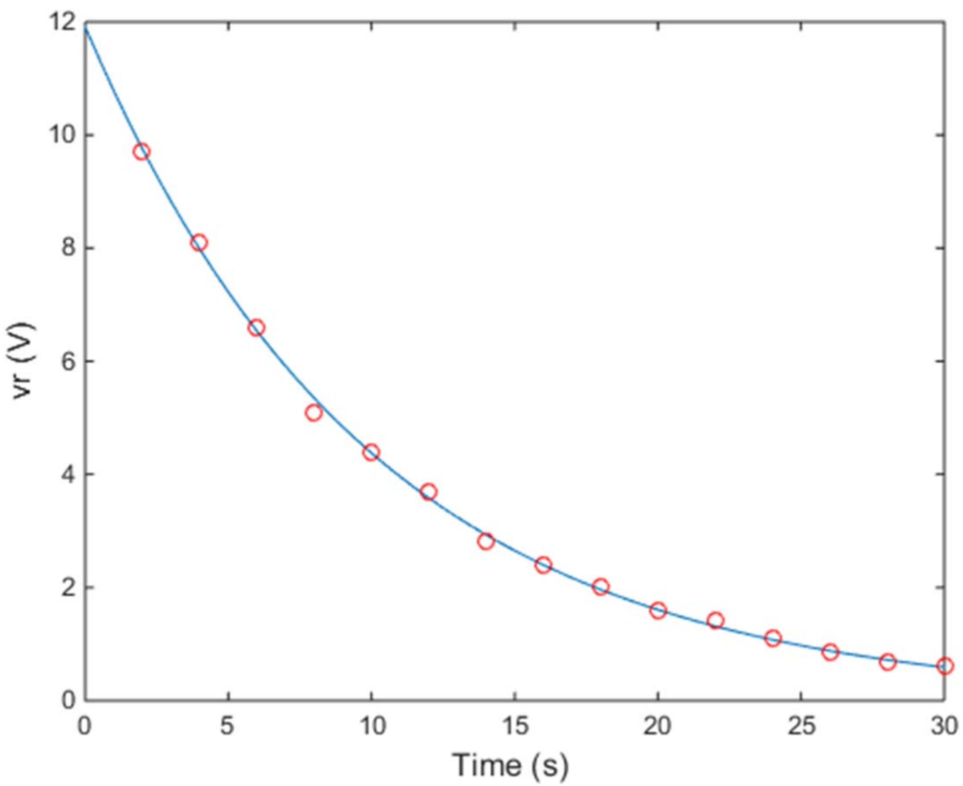
$$v_R = ve^{(-t/(RC))} \qquad (6.17)$$

Determine the capacitance of the capacitor by curve fitting the exponential function to the data.

$$v = be^{mt} \quad \Rightarrow \quad \ln(v) = \ln b + mt \qquad m = -\frac{1}{RC} \rightarrow C = -\frac{1}{Rm}$$

## Example 6-2

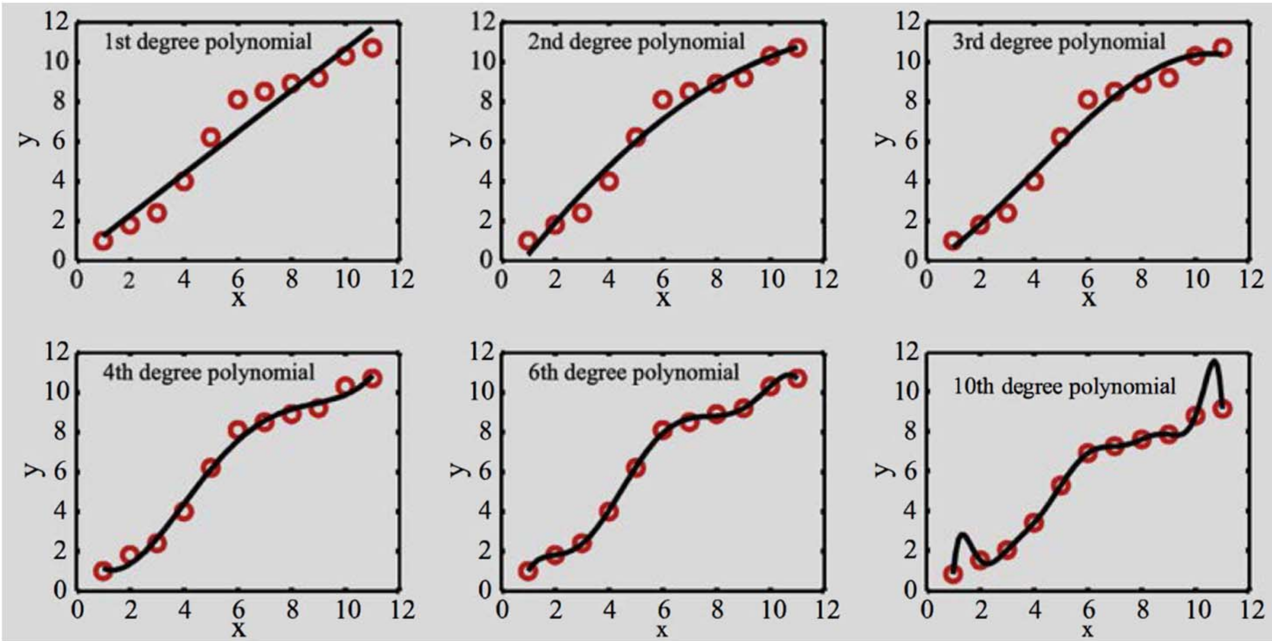
```
texp = 2:2:30;
vexp = [9.7 8.1 6.6 5.1 4.4 3.7 2.8 2.4 2.0 1.6 1.4 1.1 0.85 0.69 0.6];
vexpLOG = log(vexp);
R = 5E6;
[a1,a0] = LinearRegression(texp, vexpLOG)
b = exp(a0)
C = -1/(R*a1)
t = 0:0.5:30;
v = b*exp(a1*t);
plot(t,v,texp,vexp,'or')
```



## 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

### Polynomial function

- $n$ : non-negative integer, degree or order of the polynomial
- As the order of a polynomial increases, its curve can have more “bends”.
- With $n$ data points
  - Curve-fit with polynomials of different order
- Coefficients can be determined by minimizing the error in a least squares sense.



# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## Polynomial curve fitting

- Polynomials order of $(n-1)$
  - The polynomial passes exactly through all of the points.
  - Ex) with 11 data points, $10^{th}$ degree polynomial

- Which one is the best fit?
  - Not a simple answer
  - If data points are very accurate $\Rightarrow$ high order polynomial might be more appropriate
  - If data points are not very accurate $\Rightarrow$ high order polynomial is not recommended

# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## ❖ Polynomial regression

- Procedure for determining the coefficients of a polynomial

$$f(x) = a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$$

- Total error

$$E = \sum_{i=1}^{n}[y_i - (a_m x_i^m + a_{m-1}x_i^{m-1} + \ldots + a_1 x_i + a_0)]^2$$

Total error in linear regression

$$E = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n}[y_i - (a_1 x_i + a_0)]^2$$

$$\frac{\partial E}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_1 x_i - a_0) = 0$$

$$\frac{\partial E}{\partial a_1} = -2\sum_{i=1}^{n}(y_i - a_1 x_i - a_0)x_i = 0$$

# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## ❖ Polynomial regression

- Ex) second order polynomial

$$E = \sum_{i=1}^{n}[y_i - (a_2x_i^2 + a_1x_i + a_0)]^2$$

$$\frac{\partial E}{\partial a_0} = -2\sum_{i=1}^{n}(y_i - a_2x_i^2 - a_1x_i - a_0) = 0$$

$$\frac{\partial E}{\partial a_1} = -2\sum_{i=1}^{n}(y_i - a_2x_i^2 - a_1x_i - a_0)\boxed{x_i} = 0$$

$$\frac{\partial E}{\partial a_2} = -2\sum_{i=1}^{n}(y_i - a_2x_i^2 - a_1x_i - a_0)\boxed{x_i^2} = 0$$

$$na_0 + \left(\sum_{i=1}^{n}x_i\right)a_1 + \left(\sum_{i=1}^{n}x_i^2\right)a_2 = \sum_{i=1}^{n}y_i$$

$$\left(\sum_{i=1}^{n}x_i\right)a_0 + \left(\sum_{i=1}^{n}x_i^2\right)a_1 + \left(\sum_{i=1}^{n}x_i^3\right)a_2 = \sum_{i=1}^{n}x_iy_i$$

$$\left(\sum_{i=1}^{n}x_i^2\right)a_0 + \left(\sum_{i=1}^{n}x_i^3\right)a_1 + \left(\sum_{i=1}^{n}x_i^4\right)a_2 = \sum_{i=1}^{n}x_i^2y_i$$

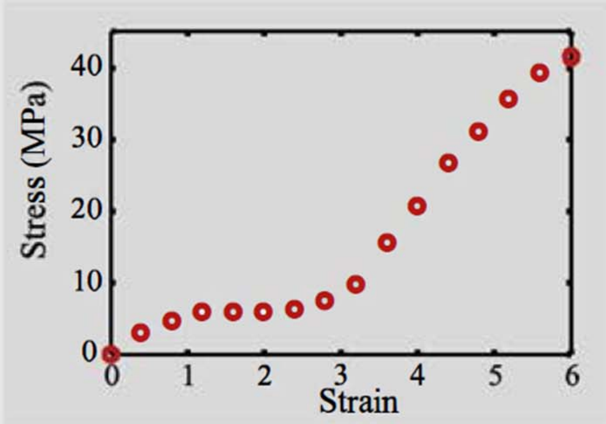
  - Solution of the system of equations: $a_0$, $a_1$, $a_2$
  - Best fits of $n$ data points: $y = a_2x_i^2 + a_1x_i + a_0$

# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## ❖ Example 6-3

**Example 6-3: Using polynomial regression for curve fitting of stress–strain curve.**

A tension test is conducted for determining the stress–strain behavior of rubber. The data points from the test are shown in the figure, and their values are given below. Determine the fourth order polynomial that best fits the data points. Make a plot of the data points and the curve that corresponds to the polynomial.




| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Strain $\varepsilon$ | 0 | 0.4 | 0.8 | 1.2 | 1.6 | 2.0 | 2.4 | | |
| Stress $\sigma$ (MPa) | 0 | 3.0 | 4.5 | 5.8 | 5.9 | 5.8 | 6.2 | | |
| Strain $\varepsilon$ | 2.8 | 3.2 | 3.6 | 4.0 | 4.4 | 4.8 | 5.2 | 5.6 | 6.0 |
| Stress $\sigma$ (MPa) | 7.4 | 9.6 | 15.6 | 20.7 | 26.7 | 31.1 | 35.6 | 39.3 | 41.5 |

$$f(x) = a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## Example 6-3

$$na_0 + \left(\sum_{i=1}^{n} x_i\right)a_1 + \left(\sum_{i=1}^{n} x_i^2\right)a_2 + \left(\sum_{i=1}^{n} x_i^3\right)a_3 + \left(\sum_{i=1}^{n} x_i^4\right)a_4 = \sum_{i=1}^{n} y_i$$

$$\left(\sum_{i=1}^{n} x_i\right)a_0 + \left(\sum_{i=1}^{n} x_i^2\right)a_1 + \left(\sum_{i=1}^{n} x_i^3\right)a_2 + \left(\sum_{i=1}^{n} x_i^4\right)a_3 + \left(\sum_{i=1}^{n} x_i^5\right)a_4 = \sum_{i=1}^{n} x_i y_i$$

$$\left(\sum_{i=1}^{n} x_i^2\right)a_0 + \left(\sum_{i=1}^{n} x_i^3\right)a_1 + \left(\sum_{i=1}^{n} x_i^4\right)a_2 + \left(\sum_{i=1}^{n} x_i^5\right)a_3 + \left(\sum_{i=1}^{n} x_i^6\right)a_4 = \sum_{i=1}^{n} x_i^2 y_i$$

$$\left(\sum_{i=1}^{n} x_i^3\right)a_0 + \left(\sum_{i=1}^{n} x_i^4\right)a_1 + \left(\sum_{i=1}^{n} x_i^5\right)a_2 + \left(\sum_{i=1}^{n} x_i^6\right)a_3 + \left(\sum_{i=1}^{n} x_i^7\right)a_4 = \sum_{i=1}^{n} x_i^3 y_i$$

$$\left(\sum_{i=1}^{n} x_i^4\right)a_0 + \left(\sum_{i=1}^{n} x_i^5\right)a_1 + \left(\sum_{i=1}^{n} x_i^6\right)a_2 + \left(\sum_{i=1}^{n} x_i^7\right)a_3 + \left(\sum_{i=1}^{n} x_i^8\right)a_4 = \sum_{i=1}^{n} x_i^4 y_i$$

- Programming steps
  - Step1: create vectors $x$ and $y$ with data points
  - Step2: create vector $xsum$: $xsum(1)$~$xsum(8)$ $\quad xsum(k) = \sum_{i=1}^{n} x^k$
  - Step3: set up the system of five linear equations

    $[a][p] = [b]$
  - Step4: solve the system of five linear equations
  - Step5: plot the data points and the curve-fitting polynomial

# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## Example 6-3

```
clear all
x = 0:0.4:6;
%x=1:4
y = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1 35.6 39.3 41.5];
n = length(x);
m = 4;
for i = 1:2*m
    xsum(i) = sum(x.^(i));
end
% Beginning of Step 3
a(1,1) = n;
b(1,1)= sum(y);
for j= 2:m+1
    a(1,j)=xsum(j-1);
end
for i = 2:m+1
    for j=1:m+1
        a(i,j)= xsum(j+i-2);
    end
    b(i,1) = sum(x.^(i-1).*y);
end
```

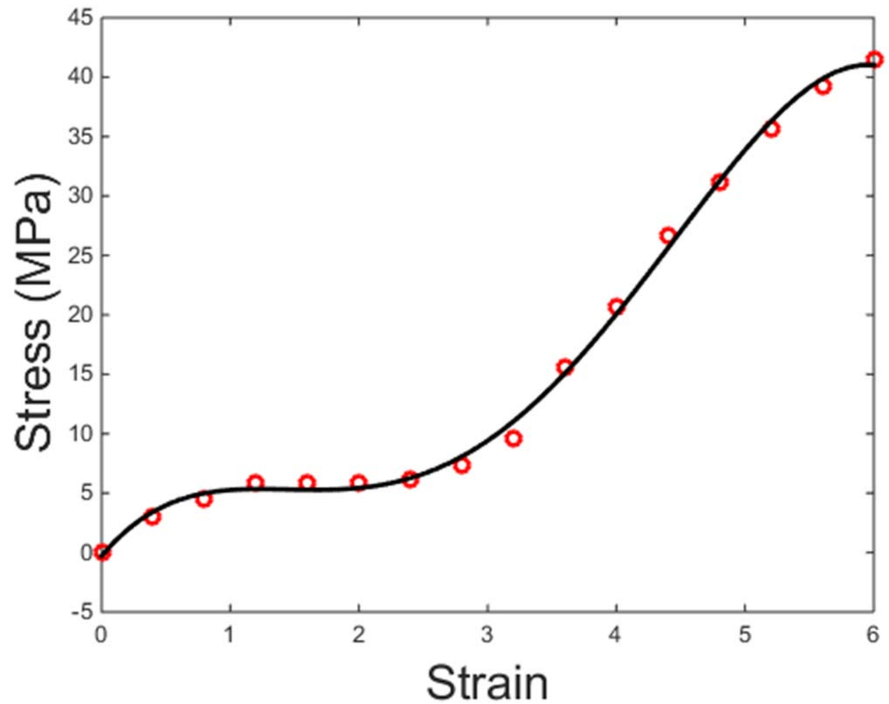
# 6.4 Curve Fitting with Quadratic and Higher-Order Polynomials

## ❖ Example 6-3

```
% Step 4
p=(a\b)'
for i = 1:m+1
    Pcoef(i) = p(m+2-i);
end
epsilon=0:0.1:6;
stressfit=polyval(Pcoef,epsilon);
plot(x,y,'ro',epsilon,stressfit,'k','linewidth',2)
xlabel('Strain','fontsize',20)
ylabel('Stress (MPa)','fontsize',20)
```

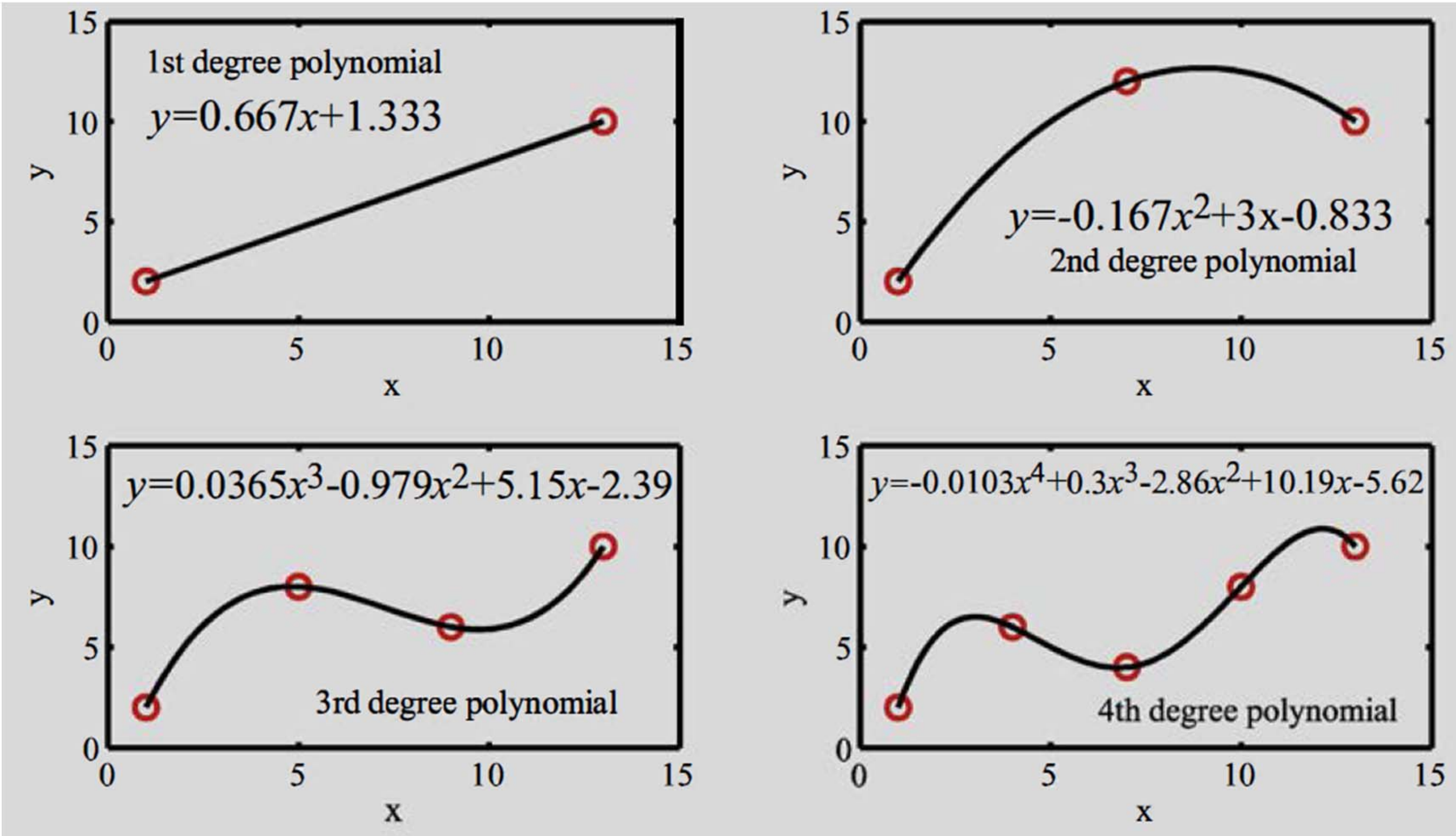
The curve-fitting polynomial is:

$$f(x)=(-0.2644)x^4+3.1185x^3-10.1927x^2+12.878x-0.2746$$



## Interpolation

- Uses a formula which gives the exact values at all the data points and an estimated value between the points.
- For $n$ data points, there is a polynomial of order on $n-1$ that passes through all of the points.



## Different forms of polynomials

- Standard form

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x + a_0$$

- Lagrange form

$$f(x) = y = a_1(x - x_2) + a_2(x - x_1)$$

$$f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2$$

$$f(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)} y_2 +$$
$$\ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} y_i + \ldots +$$
$$\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})} y_n$$

- Newton's form

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

# 6.5 Interpolation Using a Single Polynomial

## ❖ Interpolation with standard form

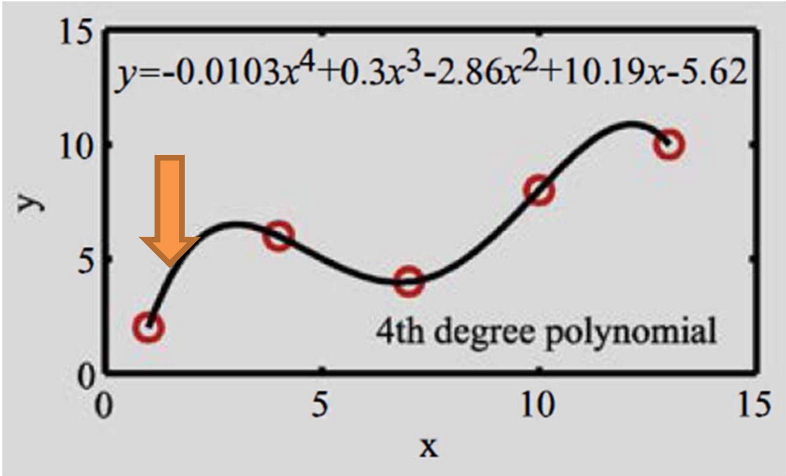
$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x + a_0$$

$$\begin{aligned} a_4 1^4 + a_3 1^3 + a_2 1^2 + a_1 1 + a_0 &= 2 \\ a_4 4^4 + a_3 4^3 + a_2 4^2 + a_1 4 + a_0 &= 6 \\ a_4 7^4 + a_3 7^3 + a_2 7^2 + a_1 7 + a_0 &= 4 \\ a_4 10^4 + a_3 10^3 + a_2 10^2 + a_1 10 + a_0 &= 8 \\ a_4 13^4 + a_3 13^3 + a_2 13^2 + a_1 13 + a_0 &= 10 \end{aligned}$$

- Ex) 4th order polynomial with 5 data points
  - $(1,2), (4,6), (7,4), (10,8), \textit{and } (13,10)$

$$y = -0.0103x^4 + 0.3x^3 - 2.86x^2 + 10.19x - 5.62$$

  - $x = 2$ ?




- $m + 1$ linear equations
  - For high order polynomials ⇒ not efficient
  - Frequently, the matrix of the coefficients is **ill conditioned**!

## Lagrange interpolating polynomials

- First-order Lagrange polynomial

$$f(x) = y = a_1(x - x_2) + a_2(x - x_1)$$

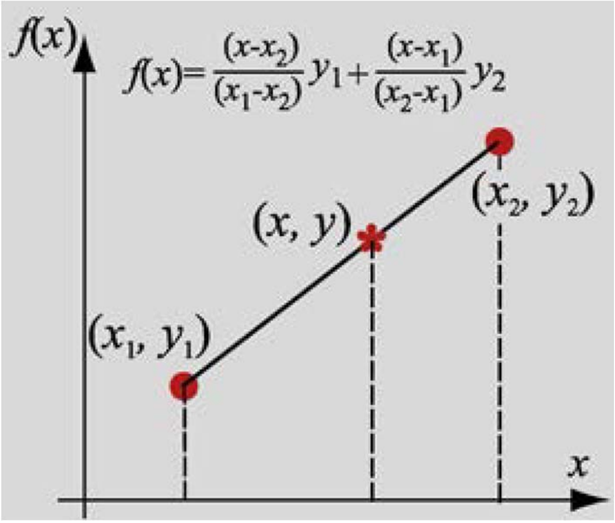
$$y_1 = a_1(x_1 - x_2) + a_2(x_1 - x_1) \quad \text{or} \quad a_1 = \frac{y_1}{(x_1 - x_2)}$$

$$y_2 = a_1(x_2 - x_2) + a_2(x_2 - x_1) \quad \text{or} \quad a_2 = \frac{y_2}{(x_2 - x_1)}$$

$$\boxed{f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2}$$

- $x = x_1 \Rightarrow y_1 \qquad x = x_2 \Rightarrow y_2$

$$f(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)} x + \frac{x_2 y_1 - x_1 y_2}{(x_2 - x_1)}$$

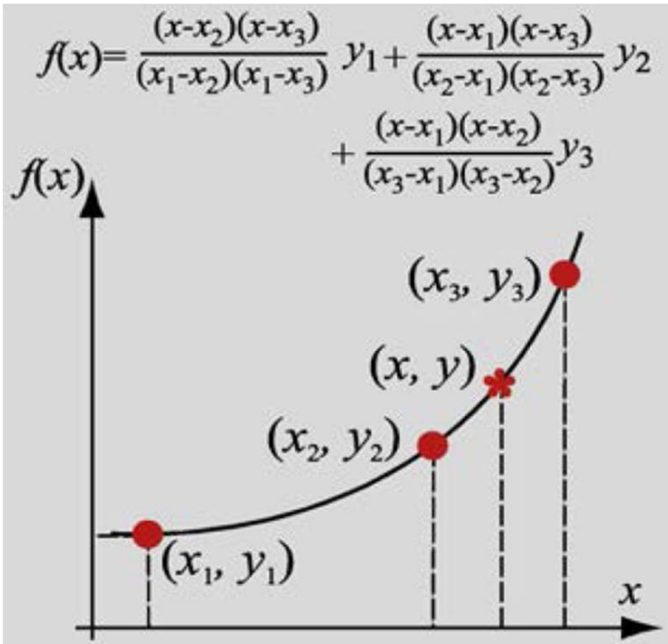


## ❖ Lagrange interpolating polynomials

- Second-order Lagrange polynomial

$$f(x) = y = a_1(x-x_2)(x-x_3) + a_2(x-x_1)(x-x_3) + a_3(x-x_1)(x-x_2)$$

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



## Lagrange interpolating polynomials

- $n-1$ order Lagrange polynomial

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

$$f(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)}y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)}y_2 +$$
$$\ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}y_i + \ldots +$$
$$\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}y_n$$

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$$ : Lagrange functions

## Lagrange interpolating polynomials

- Additional notes
  - For a given set of points, the whole expression of the interpolation polynomial has to be calculated for every value of $x$.
  - In other words, the interpolation calculations for each value of $x$ are independent of others.
  - This is different from other forms where once the coefficients of the polynomial are determined, they can be used for calculating different values of $x$.

## Program: interpolation using a Lagrange polynomial

```
function Yint = LagrangeINT(x,y,Xint)
% LagrangeINT fits a Lagrange polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x   A vector with the x coordinates of the given points.
% y   A vector with the y coordinates of the given points.
% Xint   The x coordinate of the point to be interpolated.
% Output variable:
% Yint   The interpolated value of Xint.

n = length(x);
for i = 1:n
    L(i) = 1;
    for j = 1:n
        if j ~= i
            L(i)= L(i)*(Xint-x(j))/(x(i)-x(j));
        end
    end
end
Yint = sum(y.*L);
```

$$\boxed{f(x) = \sum_{i=1}^{n} y_i L_i(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x - x_j)}{(x_i - x_j)}}$$

## Newton's interpolating polynomials

$$f(x)=a_1+a_2(x-x_1)+a_3(x-x_1)(x-x_2)+\dots+a_n(x-x_1)(x-x_2)\dots(x-x_{n-1})$$

- Determination of the coefficients does not require a solution of a system of $n$ equations.
- Desirable features
  - Data points do not have to be in descending or ascending order
  - Additional data set can be treated easily.

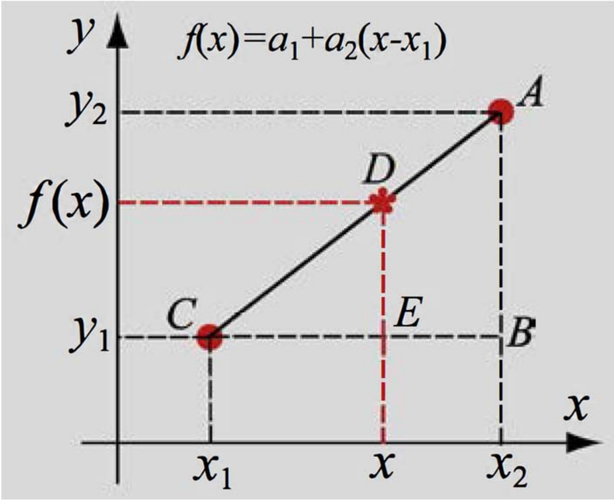
- First-order Newton's polynomial

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

$$\frac{DE}{CE} = \frac{AB}{CB}, \quad \text{or} \quad \frac{f(x)-y_1}{x-x_1} = \frac{y_2-y_1}{x_2-x_1}$$

$$f(x) = y_1 + \frac{y_2-y_1}{x_2-x_1}(x-x_1)$$

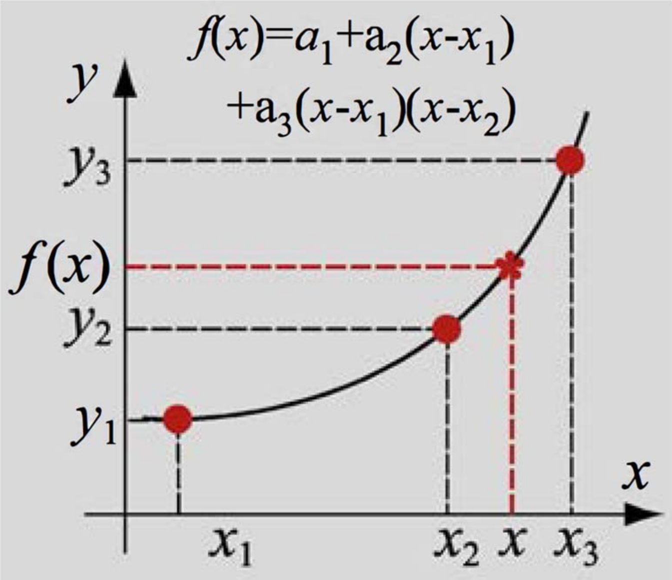
$$\boxed{a_1 = y_1}, \quad \text{and} \quad \boxed{a_2 = \frac{y_2-y_1}{x_2-x_1}}$$



## 6.5 Interpolation Using a Single Polynomial

### Newton's interpolating polynomials

- Second-order Newton's polynomial




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

- $x = x_1 \;\Rightarrow\; \boxed{a_1 = y_1}$

$$x = x_2 \;\Rightarrow\; y_2 = a_1 + a_2(x_2 - x_1) \;\Rightarrow\; \boxed{a_2 = \frac{y_2 - y_1}{x_2 - x_1}}$$

- $x = x_3 \;\Rightarrow\; y_3 = a_1 + a_2(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2)$

$$\Rightarrow\; y_3 = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2)$$

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

$$a_1 = y_1$$

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

## Newton's interpolating polynomials

- Third-order Newton's polynomial

$$f(x) = y = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) + a_4(x - x_1)(x - x_2)(x - x_3)$$

$$a_4 = \frac{\dfrac{\left(\dfrac{y_4 - y_3}{x_4 - x_3} - \dfrac{y_3 - y_2}{x_3 - x_2}\right)}{(x_4 - x_2)} - \dfrac{\left(\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}\right)}{(x_3 - x_1)}}{(x_4 - x_1)}$$

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1}$$

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

$$f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} = \frac{\dfrac{y_3 - y_2}{x_3 - x_2} - \dfrac{y_2 - y_1}{x_2 - x_1}}{(x_3 - x_1)}$$

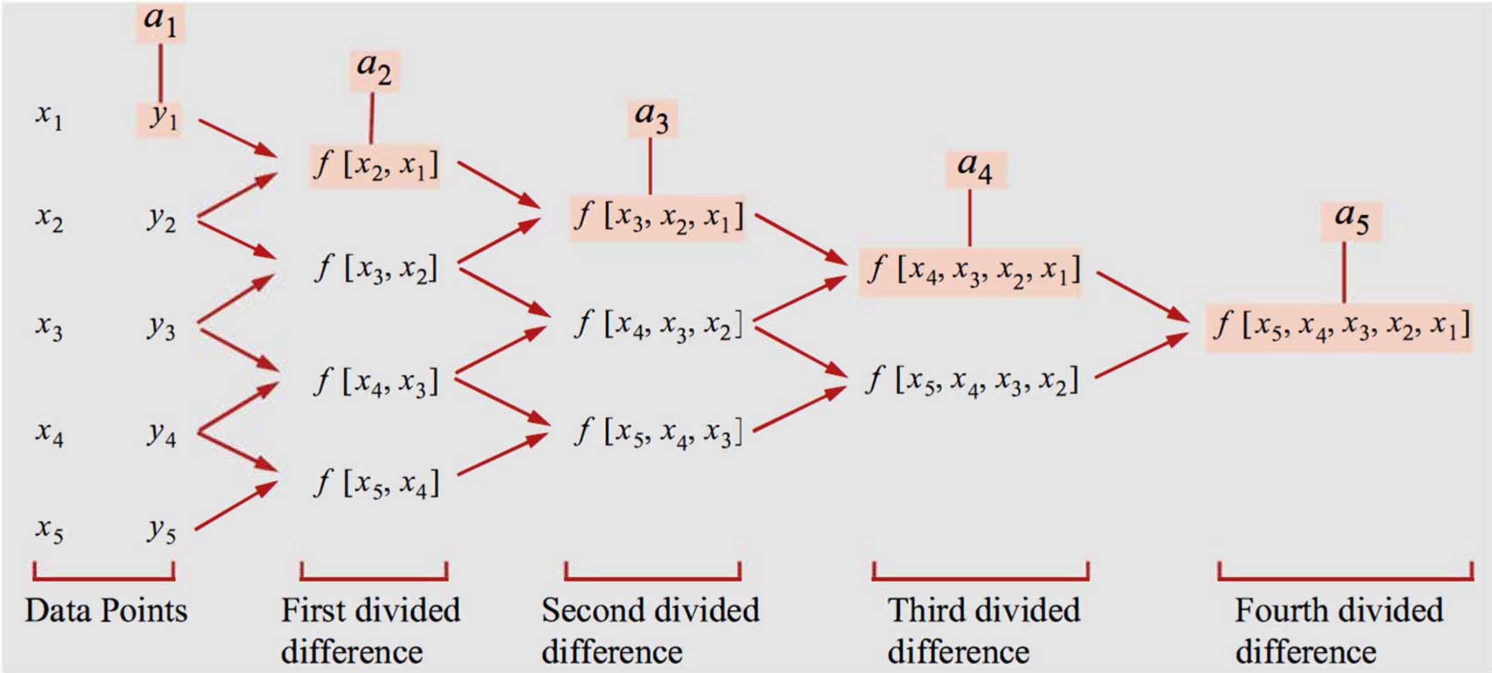
$$a_2 = \frac{y_2 - y_1}{x_2 - x_1}$$

$$f[x_2, x_1] = \frac{y_2 - y_1}{x_2 - x_1} = a_2$$

# 6.5 Interpolation Using a Single Polynomial

## ❖ Newton's interpolating polynomials

- General form of Newton's polynomial




$$f[x_k, x_{k-1}, \ldots, x_2, x_1] = \frac{f[x_k, x_{k-1}, \ldots, x_3, x_2] - f[x_{k-1}, x_{k-2}, \ldots, x_2, x_1]}{x_k - x_1}$$

$$f(x) = y = \underbrace{y_1}_{a_1} + \underbrace{f[x_2, x_1]}_{a_2}(x - x_1) + \underbrace{f[x_3, x_2, x_1]}_{a_3}(x - x_1)(x - x_2) + \ldots + \underbrace{f[x_n, x_{n-1}, \ldots, x_2, x_1]}_{a_n}(x - x_1)(x - x_2)\ldots(x - x_{n-1})$$

## Newton's interpolating polynomials

- Example 6-5

**Example 6-5: Newton's interpolating polynomial.**

The set of the following five data points is given:

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 1 | 2 | 4 | 5 | 7 |
| $y$ | 52 | 5 | –5 | –40 | 10 |

($a$) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
($b$) Use the polynomial obtained in part ($a$) to determine the interpolated value for $x = 3$.
($c$) Write a MATLAB user-defined function that interpolates using Newton's polynomial. The input to the function should be the coordinates of the given data points and the $x$ coordinate of the point at which $y$ is to be interpolated. The output from the function is the $y$ value of the interpolated point.

## Newton's interpolating polynomials

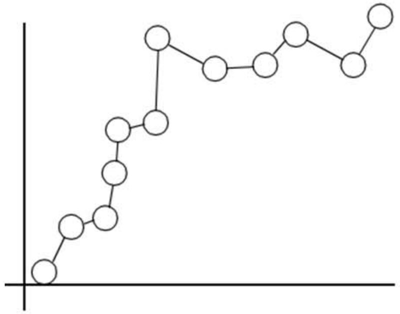
- Example 6-5

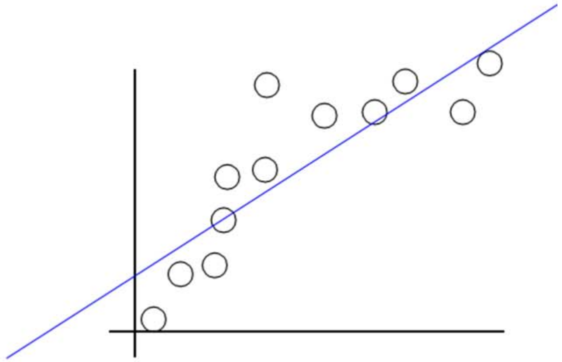
```
function Yint = NewtonsINT(x,y,Xint)
n = length(x);
a(1) = y(1);
for i = 1:n-1
    divDIF(i,1) = (y(i+1)-y(i))/(x(i+1)-x(i));
end
for j = 2:n-1
    for i = 1:n-j
        divDIF(i,j) = (divDIF(i+1,j-1) - divDIF(i,j-1))/(x(j+i) - x(i));
    end
end
for j=2:n
    a(j)=divDIF(1,j-1);
end
Yint=a(1);
xn=1;
for k=2:n
    xn=xn*(Xint-x(k-1));
    Yint=Yint+a(k)*xn;
end
```

```
% NewtonsINT fits a Newtons polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x   A vector with the x coordinates of the given points.
% y   A vector with the y coordinates of the given points.
% Xint   The x coordinate of the point to be interpolated.
% Output variable:
% Yint   The interpolated value of Xint.
```

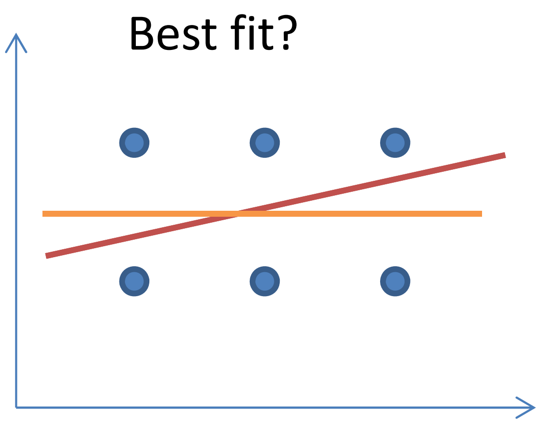
# Review

## Curve fitting and interpolation





- Least square method
  - Linear Least-Squares Regression
  - Curve fitting with nonlinear equation
  - Curve Fitting with Quadratic and Higher-Order Polynomials

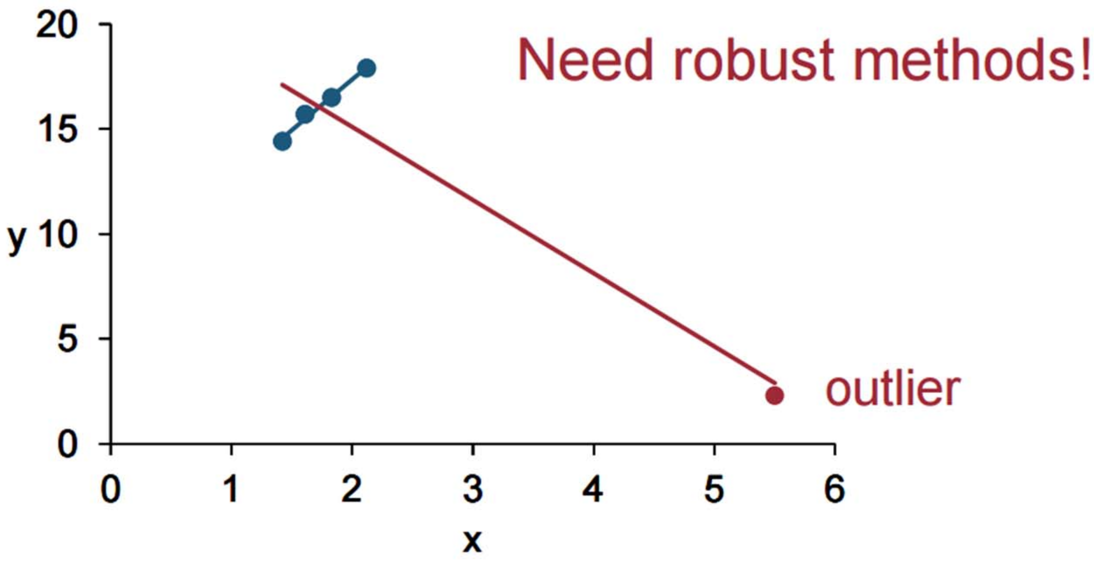



$$E = \sum_{i=1}^{n} |r_i| = \sum_{i=1}^{n} |y_i - (a_1 x_i + a_0)|$$

$$E = \sum_{i=1}^{n} [y_i - (a_1 x_i + a_0)]^2$$

## ❖ Curve fitting and interpolation

- Ex) RANSAC: Random Sample Consensus
  - Outlier



## Curve fitting and interpolation

- Non-linear square method : example

$$f(x; A, x_0, \sigma) = A\, e^{-(x-x_0)^2/(2\sigma^2)}$$

$$E = \sum_{i=1}^{n} [y_i - f(x_i)]^2$$

$$\frac{\partial E}{\partial A} = -2\sum_{i=1}^{n}[y_i - f(x_i)]\frac{\partial f}{\partial A} = 0$$

$$\frac{\partial E}{\partial x_0} = -2\sum_{i=1}^{n}[y_i - f(x_i)]\frac{\partial f}{\partial x_0} = 0$$

$$\frac{\partial E}{\partial \sigma} = -2\sum_{i=1}^{n}[y_i - f(x_i)]\frac{\partial f}{\partial \sigma} = 0$$

$$\frac{\partial f}{\partial A} = e^{-(x-x_0)^2/(2\sigma^2)}$$

$$\frac{\partial f}{\partial x_0} = \frac{A(x-x_0)}{\sigma^2} e^{-(x-x_0)^2/(2\sigma^2)}$$

$$\frac{\partial f}{\partial \sigma} = \frac{A(x-x_0)^2}{\sigma^3} e^{-(x-x_0)^2/(2\sigma^2)}$$

- 3 unknowns, system of 3 non-linear equations!

## Curve fitting

- Solution of over-determined system

$$y = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x) = \sum_{j=1}^{n} c_j f_j(x)$$

  - With five functions $\Rightarrow$ five coefficients need to be determined!
  - With ten data points
  - Five unknowns with ten equations ? $\Rightarrow$ over-determined system

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \\ \\ \\ \\ y_{10} \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) & f_4(x_1) & f_5(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) & f_4(x_2) & f_5(x_2) \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ f_1(x_{10}) & f_2(x_{10}) & f_3(x_{10}) & f_4(x_{10}) & f_5(x_{10}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \mathbf{FC} = \mathbf{Y}$$

# Curve fitting

- Solution of over-determined system
  - Least square method

$$\|\mathbf{e}\|^2 = \mathbf{e}^T\mathbf{e} = (\mathbf{Y}-\mathbf{FC})^T(\mathbf{Y}-\mathbf{FC}) = (\mathbf{Y}^T-\mathbf{C}^T\mathbf{F}^T)(\mathbf{Y}-\mathbf{FC})$$

$$= \mathbf{Y}^T\mathbf{Y} - \mathbf{C}^T\mathbf{F}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{FC} + \mathbf{C}^T\mathbf{F}^T\mathbf{FC}$$

$$= \mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{FC} + \mathbf{C}^T\mathbf{F}^T\mathbf{FC}$$

$$\mathbf{C}^T\mathbf{F}^T\mathbf{Y} = \mathbf{C}^T\left(\mathbf{F}^T\mathbf{Y}\right) = \left(\mathbf{F}^T\mathbf{Y}\right)^T\mathbf{C} = \mathbf{Y}^T\mathbf{FC}$$

  - Derivative with respect to **C**

$$\frac{d}{d\mathbf{C}}\left(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{FC} + \mathbf{C}^T\mathbf{F}^T\mathbf{FC}\right) = -2\mathbf{F}^T\mathbf{Y} + 2\mathbf{F}^T\mathbf{FC} = 0$$

$$\frac{d}{d\mathbf{C}}\left(\mathbf{Y}^T\mathbf{FC}\right) = \left(\mathbf{Y}^T\mathbf{F}\right)^T = \mathbf{F}^T\mathbf{Y}$$

$$\frac{d}{d\mathbf{C}}\left(\mathbf{C}^T\mathbf{F}^T\mathbf{FC}\right) = \mathbf{F}^T\mathbf{FC} + \left(\mathbf{F}^T\mathbf{F}\right)^T\mathbf{C} = \mathbf{F}^T\mathbf{FC} + \mathbf{F}^T\mathbf{FC} = 2\mathbf{F}^T\mathbf{FC}$$

| $\mathbf{y}$ | $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$ |
|---|---|
| $\mathbf{Ax}$ | $\mathbf{A}^T$ |
| $\mathbf{x}^T\mathbf{A}$ | $\mathbf{A}$ |
| $\mathbf{x}^T\mathbf{x}$ | $2\mathbf{x}$ |
| $\mathbf{x}^T\mathbf{Ax}$ | $\mathbf{Ax} + \mathbf{A}^T\mathbf{x}$ |

## Curve fitting

- Solution of over-determined system $\mathbf{FC} = \mathbf{Y}$
  - Least square method

$$\frac{d}{d\mathbf{C}}\left(\mathbf{Y}^{\mathrm{T}}\mathbf{Y} - 2\mathbf{Y}^{\mathrm{T}}\mathbf{FC} + \mathbf{C}^{\mathrm{T}}\mathbf{F}^{\mathrm{T}}\mathbf{FC}\right) = -2\mathbf{F}^{\mathrm{T}}\mathbf{Y} + 2\mathbf{F}^{\mathrm{T}}\mathbf{FC} = 0$$

$$\mathbf{F}^{\mathrm{T}}\mathbf{FC} = \mathbf{F}^{\mathrm{T}}\mathbf{Y}$$

(5x10)(10x5) (5x10)(10x1)

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \\ \\ \\ y_{10} \end{bmatrix} = \begin{bmatrix} f_1(x_1) & f_2(x_1) & f_3(x_1) & f_4(x_1) & f_5(x_1) \\ f_1(x_2) & f_2(x_2) & f_3(x_2) & f_4(x_2) & f_5(x_2) \\ \\ \\ \\ \\ f_1(x_{10}) & f_2(x_{10}) & f_3(x_{10}) & f_4(x_{10}) & f_5(x_{10}) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \mathbf{FC} = \mathbf{Y}$$

## Curve fitting

- Solution of over-determined system
  - Example

$$y = c_1 + c_2 \sin\left(\frac{\pi x}{10}\right) \longrightarrow f_1(x) = 1, \quad f_2(x) = \sin\left(\frac{\pi x}{10}\right)$$

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{Y}$$

| $x$ | 0 | 2.5 | 6.5 | 9.5 | 13 | 15.5 | 20 |
|---|---|---|---|---|---|---|---|
| $y$ | 1.1 | 4.6 | 4.6 | 0.3 | -4.0 | -4.3 | 0 |

## Curve fitting

- Solution of over-determined system
  - Example

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{Y}$$

$$\mathbf{F} = \begin{bmatrix} 1 & \sin\left(\frac{\pi\times 0}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 2.5}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 6.5}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 9.5}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 13}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 15.5}{10}\right) \\ 1 & \sin\left(\frac{\pi\times 20}{10}\right) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1 & 0.7071 \\ 1 & 0.8910 \\ 1 & 0.1564 \\ 1 & -0.8090 \\ 1 & -0.9877 \\ 1 & 0 \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} 1.1 \\ 4.6 \\ 4.6 \\ 0.3 \\ -4.0 \\ -4.3 \\ 0 \end{bmatrix}$$

$$\mathbf{F}^{\mathrm{T}} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.7071 & 0.8910 & 0.1564 & -0.8090 & -0.9877 & 0 \end{bmatrix}$$

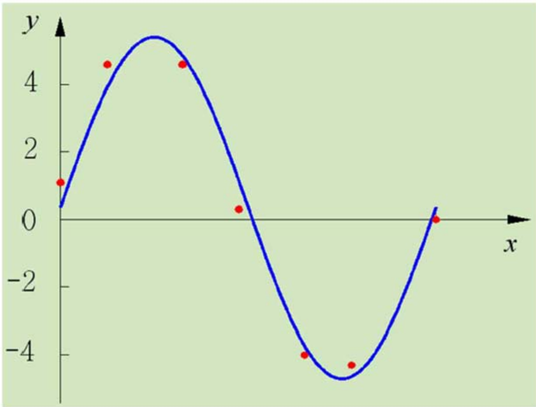
# Curve fitting

- Solution of over-determined system
  - Example $\mathbf{F}^{\mathrm{T}}\mathbf{F}\mathbf{C} = \mathbf{F}^{\mathrm{T}}\mathbf{Y}$

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.7071 & 0.8910 & 0.1564 & -0.8090 & -0.9877 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 1 & 0.7071 \\ 1 & 0.8910 \\ 1 & 0.1564 \\ 1 & -0.8090 \\ 1 & -0.9877 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0.7071 & 0.8910 & 0.1564 & -0.8090 & -0.9877 & 0 \end{bmatrix} \begin{bmatrix} 1.1 \\ 4.6 \\ 4.6 \\ 0.3 \\ -4.0 \\ -4.3 \\ 0 \end{bmatrix}$$

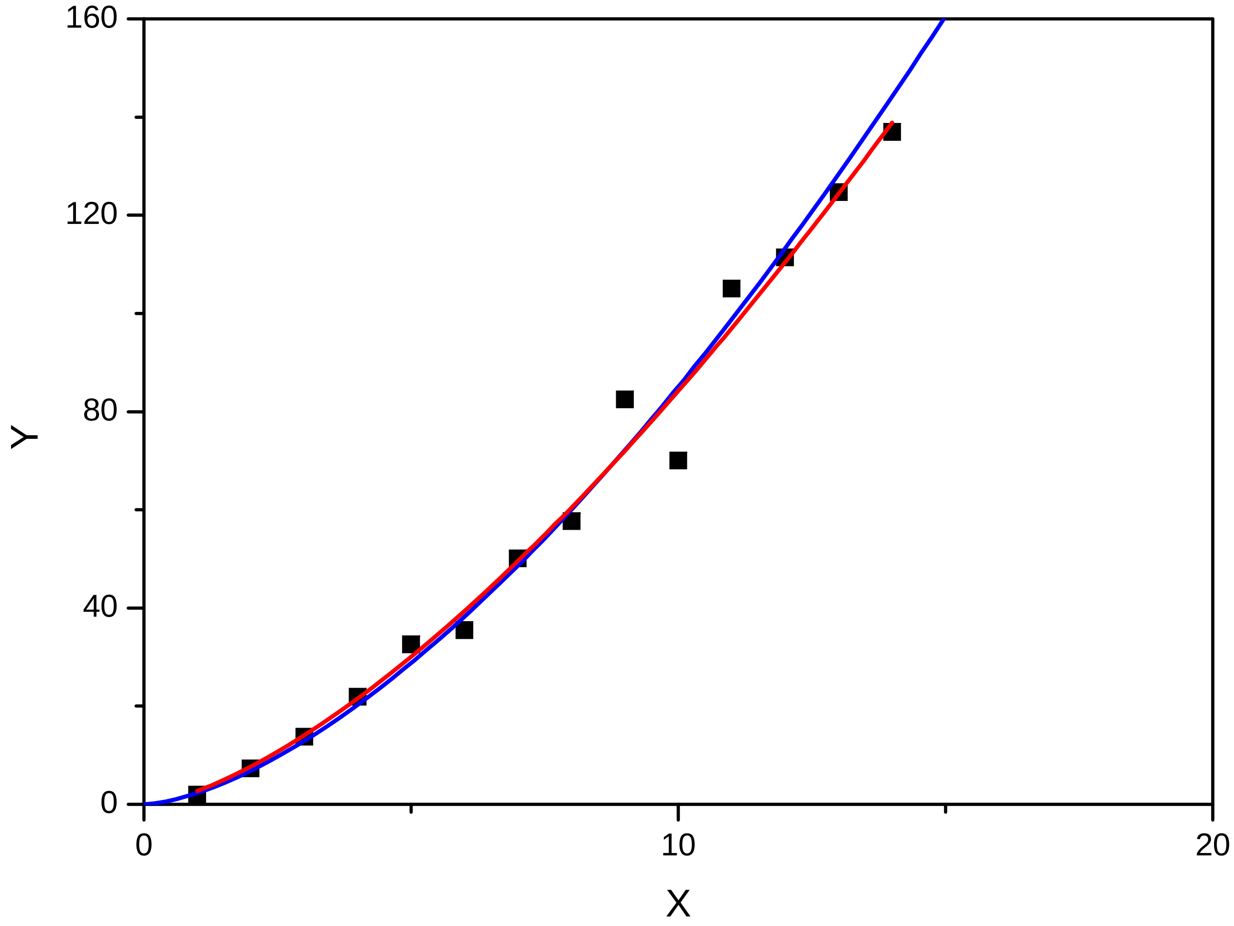
$$\begin{bmatrix} 7 & -0.0422 \\ -0.0422 & 2.9484 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 2.3 \\ 14.8813 \end{bmatrix}$$

$$y = 0.359 + 5.0524 \sin\left(\frac{\pi x}{10}\right)$$



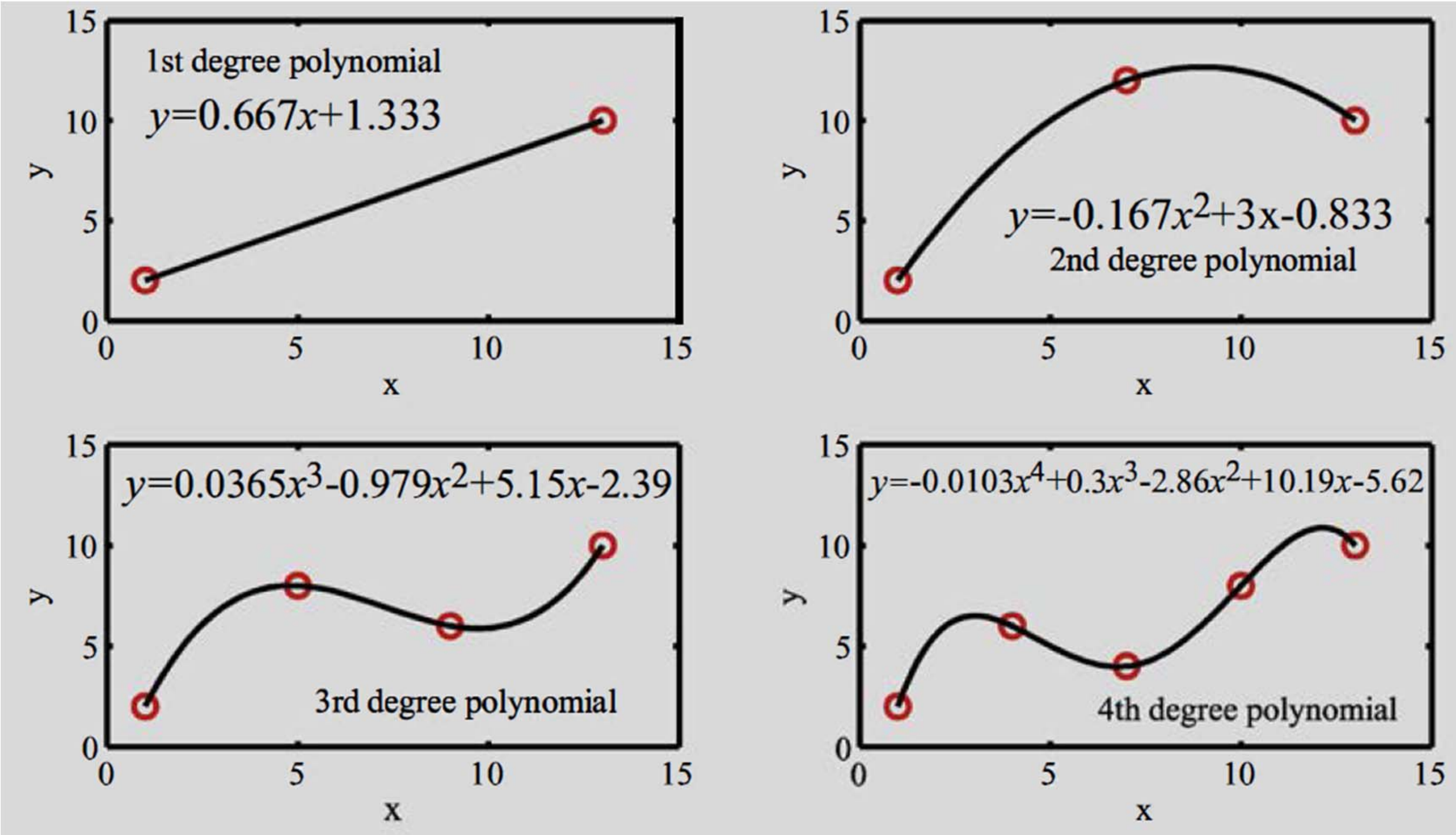
## Curve fitting and interpolation

- Curve fitting with nonlinear equation



## Interpolation

- Uses a formula which gives the exact values at all the data points and an estimated value between the points.
- For $n$ data points, there is a polynomial of order on $n-1$ that passes through all of the points.



### Different forms of polynomials

- Standard form

$$f(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x + a_0$$

- Lagrange form

$$f(x) = y = a_1(x - x_2) + a_2(x - x_1) \qquad f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2$$

$$f(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)} y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)} y_2 + \ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)} y_i + \ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})} y_n$$

- Newton's form

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

# 6.5 Interpolation Using a Single Polynomial

## ❖ Interpolation with standard form

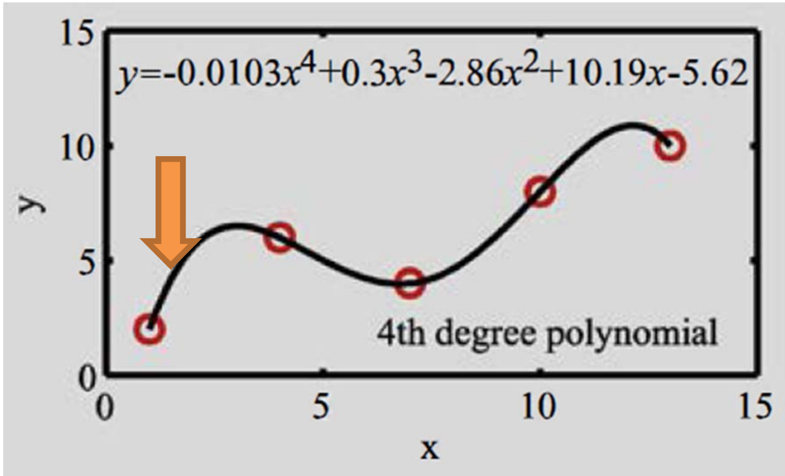
$$f(x) = a_m x^m + a_{m-1}x^{m-1} + \dots + a_1 x + a_0$$

$$\begin{aligned} a_4 1^4 + a_3 1^3 + a_2 1^2 + a_1 1 + a_0 &= 2 \\ a_4 4^4 + a_3 4^3 + a_2 4^2 + a_1 4 + a_0 &= 6 \\ a_4 7^4 + a_3 7^3 + a_2 7^2 + a_1 7 + a_0 &= 4 \\ a_4 10^4 + a_3 10^3 + a_2 10^2 + a_1 10 + a_0 &= 8 \\ a_4 13^4 + a_3 13^3 + a_2 13^2 + a_1 13 + a_0 &= 10 \end{aligned}$$

- Ex) 4th order polynomial with 5 data points
  - $(1,2), (4,6), (7,4), (10,8), and\ (13,10)$

$$y = -0.0103x^4 + 0.3x^3 - 2.86x^2 + 10.19x - 5.62$$

  - $x = 2$ ?

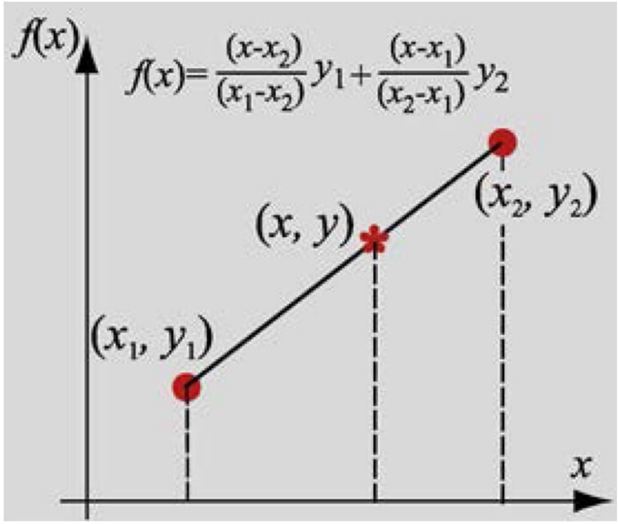



- $m + 1$ linear equations
  - For high order polynomials $\Rightarrow$ not efficient
  - Frequently, the matrix of the coefficients is **ill conditioned**!

## Lagrange interpolating polynomials

- First-order Lagrange polynomial

$$f(x) = y = a_1(x - x_2) + a_2(x - x_1)$$




$$y_1 = a_1(x_1 - x_2) + a_2(x_1 - x_1) \quad \text{or} \quad a_1 = \frac{y_1}{(x_1 - x_2)}$$

$$y_2 = a_1(x_2 - x_2) + a_2(x_2 - x_1) \quad \text{or} \quad a_2 = \frac{y_2}{(x_2 - x_1)}$$

$$\boxed{f(x) = \frac{(x - x_2)}{(x_1 - x_2)} y_1 + \frac{(x - x_1)}{(x_2 - x_1)} y_2}$$

- $x = x_1 \Rightarrow y_1 \qquad x = x_2 \Rightarrow y_2$

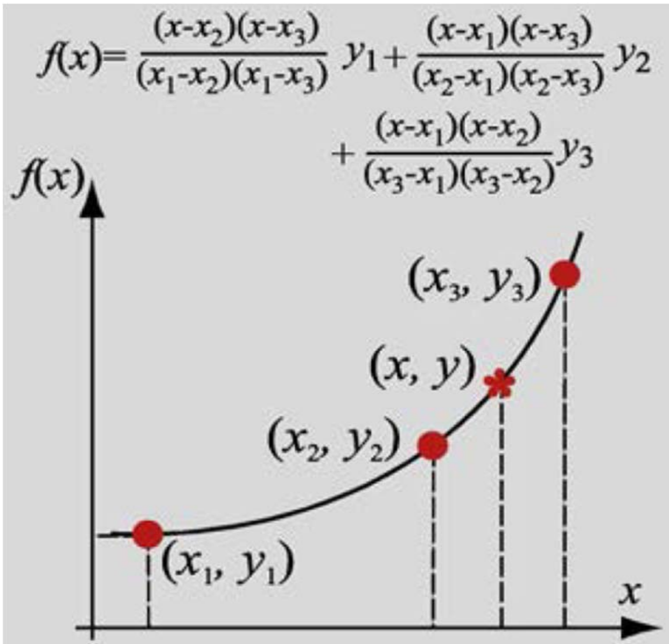
$$f(x) = \frac{(y_2 - y_1)}{(x_2 - x_1)} x + \frac{x_2 y_1 - x_1 y_2}{(x_2 - x_1)}$$

## ❖ Lagrange interpolating polynomials

- Second-order Lagrange polynomial

$$f(x) = y = a_1(x-x_2)(x-x_3) + a_2(x-x_1)(x-x_3) + a_3(x-x_1)(x-x_2)$$

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



## Lagrange interpolating polynomials

- $n-1$ order Lagrange polynomial

$$(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$$

$$f(x) = \frac{(x-x_2)(x-x_3)\ldots(x-x_n)}{(x_1-x_2)(x_1-x_3)\ldots(x_1-x_n)}y_1 + \frac{(x-x_1)(x-x_3)\ldots(x-x_n)}{(x_2-x_1)(x_2-x_3)\ldots(x_2-x_n)}y_2 +$$
$$\ldots + \frac{(x-x_1)(x-x_2)\ldots(x-x_{i-1})(x-x_{i+1})\ldots(x-x_n)}{(x_i-x_1)(x_i-x_2)\ldots(x_i-x_{i-1})(x_i-x_{i+1})\ldots(x_i-x_n)}y_i + \ldots +$$
$$\frac{(x-x_1)(x-x_2)\ldots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\ldots(x_n-x_{n-1})}y_n$$

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}$$ : Lagrange functions

## Lagrange interpolating polynomials

- Additional notes
  - For a given set of points, the whole expression of the interpolation polynomial has to be calculated for every value of $x$.
  - In other words, the interpolation calculations for each value of $x$ are independent of others.
  - This is different from other forms where once the coefficients of the polynomial are determined, they can be used for calculating different values of $x$.

## ❖ Program: interpolation using a Lagrange polynomial

```
function Yint = LagrangeINT(x,y,Xint)
% LagrangeINT fits a Lagrange polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x   A vector with the x coordinates of the given points.
% y   A vector with the y coordinates of the given points.
% Xint   The x coordinate of the point to be interpolated.
% Output variable:
% Yint   The interpolated value of Xint.

n = length(x);
for i = 1:n
    L(i) = 1;
    for j = 1:n
        if j ~= i
            L(i)= L(i)*(Xint-x(j))/(x(i)-x(j));
        end
    end
end
Yint = sum(y.*L);
```

$$\boxed{f(x) = \sum_{i=1}^{n} y_i L_i(x) = \sum_{i=1}^{n} y_i \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{(x-x_j)}{(x_i-x_j)}}$$

## Newton's interpolating polynomials

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

- Determination of the coefficients does not require a solution of a system of $n$ equations.
- Desirable features
  - Data points do not have to be in descending or ascending order
  - Additional data set can be treated easily.

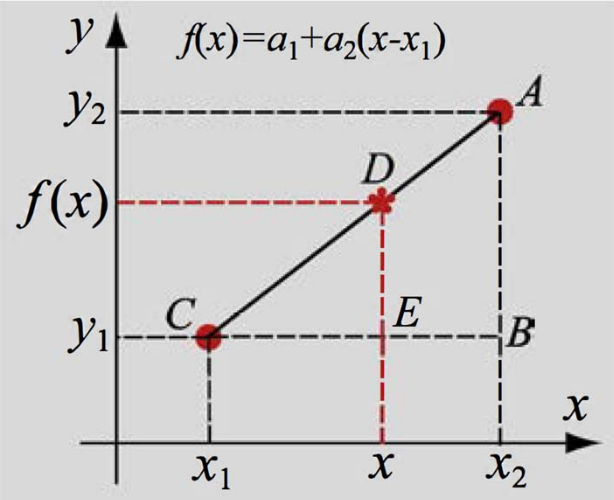
- First-order Newton's polynomial

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

$$\frac{DE}{CE} = \frac{AB}{CB}, \quad \text{or} \quad \frac{f(x) - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

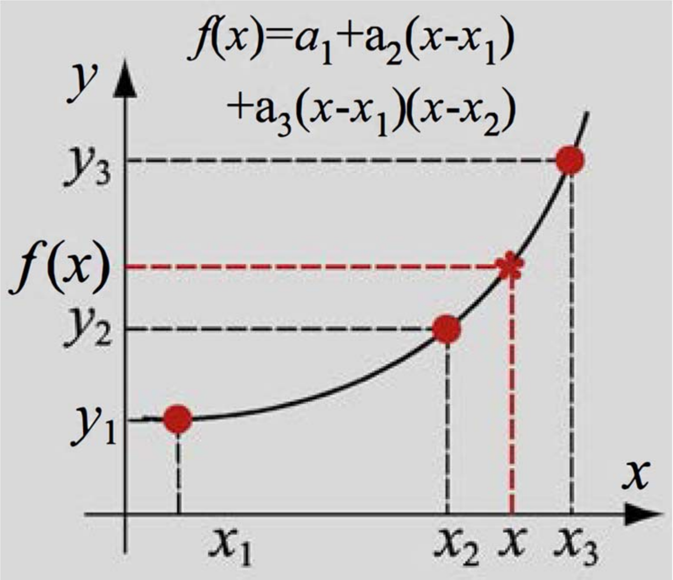
$$f(x) = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1)$$

$$\boxed{a_1 = y_1}, \quad \text{and} \quad \boxed{a_2 = \frac{y_2 - y_1}{x_2 - x_1}}$$



## Newton's interpolating polynomials

- Second-order Newton's polynomial




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

- $x = x_1 \Rightarrow \boxed{a_1 = y_1}$

$$x = x_2 \Rightarrow y_2 = a_1 + a_2(x_2 - x_1) \Rightarrow \boxed{a_2 = \frac{y_2 - y_1}{x_2 - x_1}}$$

- $x = x_3 \Rightarrow y_3 = a_1 + a_2(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2)$

$$\Rightarrow y_3 = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2)$$

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

$$a_1 = y_1$$

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

## Newton's interpolating polynomials

- Third-order Newton's polynomial

$$f(x) = y = a_1 + a_2(x-x_1) + a_3(x-x_1)(x-x_2) + a_4(x-x_1)(x-x_2)(x-x_3)$$

$$a_4 = \frac{\dfrac{\left(\dfrac{y_4-y_3}{x_4-x_3} - \dfrac{y_3-y_2}{x_3-x_2}\right)}{(x_4-x_2)} - \dfrac{\left(\dfrac{y_3-y_2}{x_3-x_2} - \dfrac{y_2-y_1}{x_2-x_1}\right)}{(x_3-x_1)}}{(x_4-x_1)}$$

$$f[x_4, x_3, x_2, x_1] = \frac{f[x_4, x_3, x_2] - f[x_3, x_2, x_1]}{x_4 - x_1}$$

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

$$f[x_3, x_2, x_1] = \frac{f[x_3, x_2] - f[x_2, x_1]}{x_3 - x_1} = \frac{\dfrac{y_3-y_2}{x_3-x_2} - \dfrac{y_2-y_1}{x_2-x_1}}{(x_3-x_1)}$$

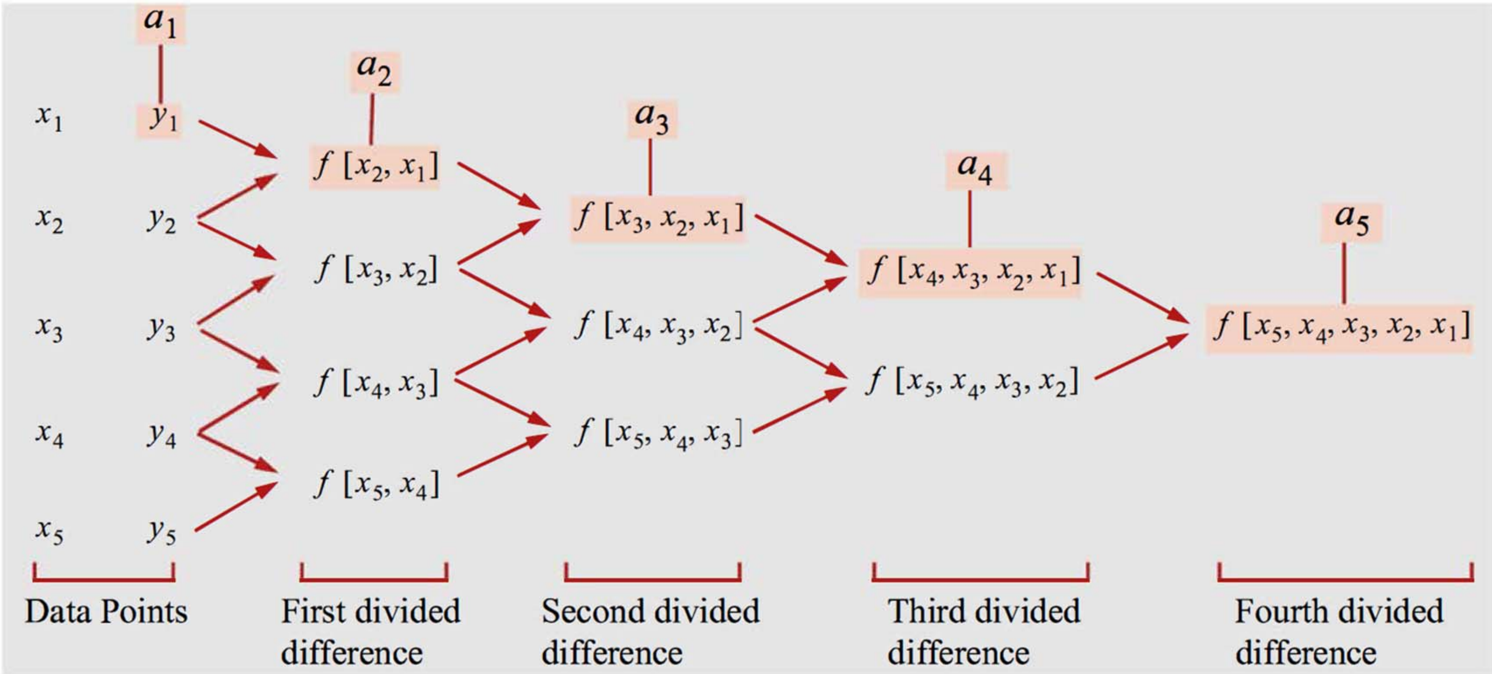
$$a_2 = \frac{y_2-y_1}{x_2-x_1}$$

$$f[x_2, x_1] = \frac{y_2-y_1}{x_2-x_1} = a_2$$

# 6.5 Interpolation Using a Single Polynomial

## Newton's interpolating polynomials

- General form of Newton's polynomial




$$f\,[x_k, x_{k-1}, \dots, x_2, x_1] = \frac{f\,[x_k, x_{k-1}, \dots, x_3, x_2] - f\,[x_{k-1}, x_{k-2}, \dots, x_2, x_1]}{x_k - x_1}$$

$$f(x) = y = \underbrace{y_1}_{a_1} + \underbrace{f\,[x_2, x_1]}_{a_2}(x - x_1) + \underbrace{f\,[x_3, x_2, x_1]}_{a_3}(x - x_1)(x - x_2) + \dots + \underbrace{f\,[x_n, x_{n-1}, \dots, x_2, x_1]}_{a_n}(x - x_1)(x - x_2)\dots(x - x_{n-1})$$

## Newton's interpolating polynomials

- Example 6-5

**Example 6-5: Newton's interpolating polynomial.**

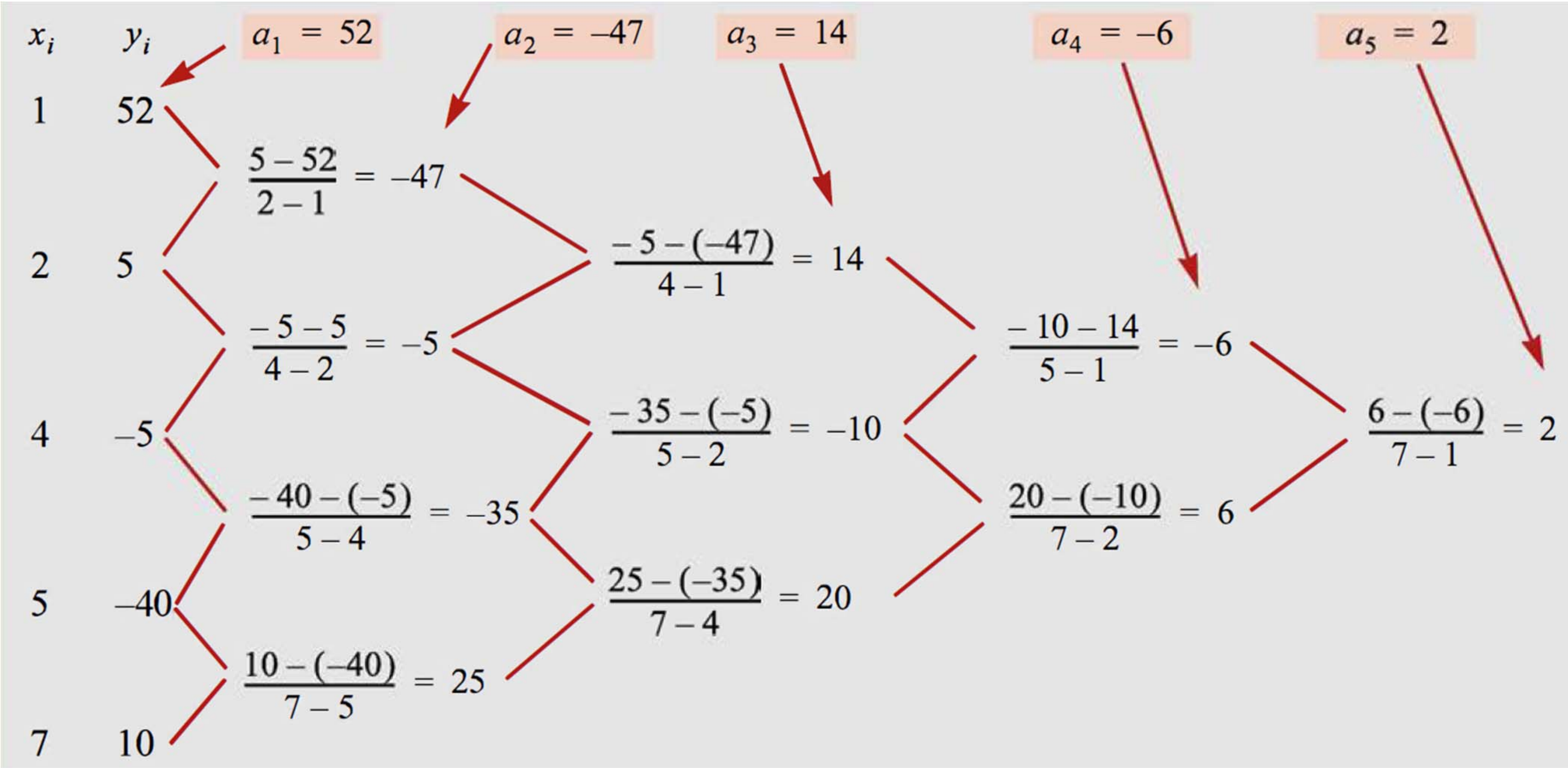
The set of the following five data points is given:

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 1 | 2 | 4 | 5 | 7 |
| $y$ | 52 | 5 | –5 | –40 | 10 |

(*a*) Determine the fourth-order polynomial in Newton's form that passes through the points. Calculate the coefficients by using a divided difference table.
(*b*) Use the polynomial obtained in part (*a*) to determine the interpolated value for $x = 3$.
(*c*) Write a MATLAB user-defined function that interpolates using Newton's polynomial. The input to the function should be the coordinates of the given data points and the $x$ coordinate of the point at which $y$ is to be interpolated. The output from the function is the $y$ value of the interpolated point.

- **Newton's interpolating polynomials**
  - Example 6-5

$$f(x) = y = 52 - 47(x-1) + 14(x-1)(x-2) - 6(x-1)(x-2)(x-4) + 2(x-1)(x-2)(x-4)(x-5)$$



## Newton's interpolating polynomials

- Example 6-5

```
function Yint = NewtonsINT(x,y,Xint)
n = length(x);
a(1) = y(1);
for i = 1:n-1
    divDIF(i,1) = (y(i+1)-y(i))/(x(i+1)-x(i));
end
for j = 2:n-1
    for i = 1:n-j
        divDIF(i,j) = (divDIF(i+1,j-1) - divDIF(i,j-1))/(x(j+i) - x(i));
    end
end
for j=2:n
    a(j)=divDIF(1,j-1);
end
Yint=a(1);
xn=1;
for k=2:n
    xn=xn*(Xint-x(k-1));
    Yint=Yint+a(k)*xn;
end
```

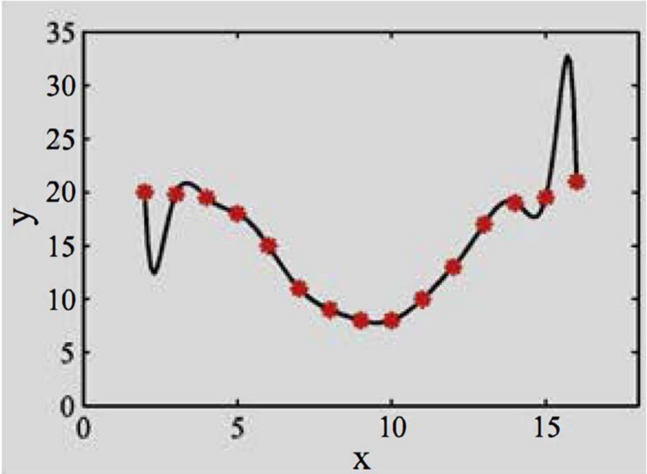
```
% NewtonsINT fits a Newtons polynomial to a set of given points and
% uses the polynomial to determines the interpolated value of a point.
% Input variables:
% x   A vector with the x coordinates of the given points.
% y   A vector with the y coordinates of the given points.
% Xint   The x coordinate of the point to be interpolated.
% Output variable:
% Yint   The interpolated value of Xint.
```

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

## 6.6 Piecewise (spline) Interpolation

- **Problem of a single polynomial interpolation**
  - In high-order polynomial

- **Spline interpolation**
  - Many low-order polynomials
  - Each low-order polynomial is valid in one interval
  - First-order polynomials
    - Data points connected with straight lines
  - Second-order (quadratic) polynomials
  - Third-order (cubic) polynomials

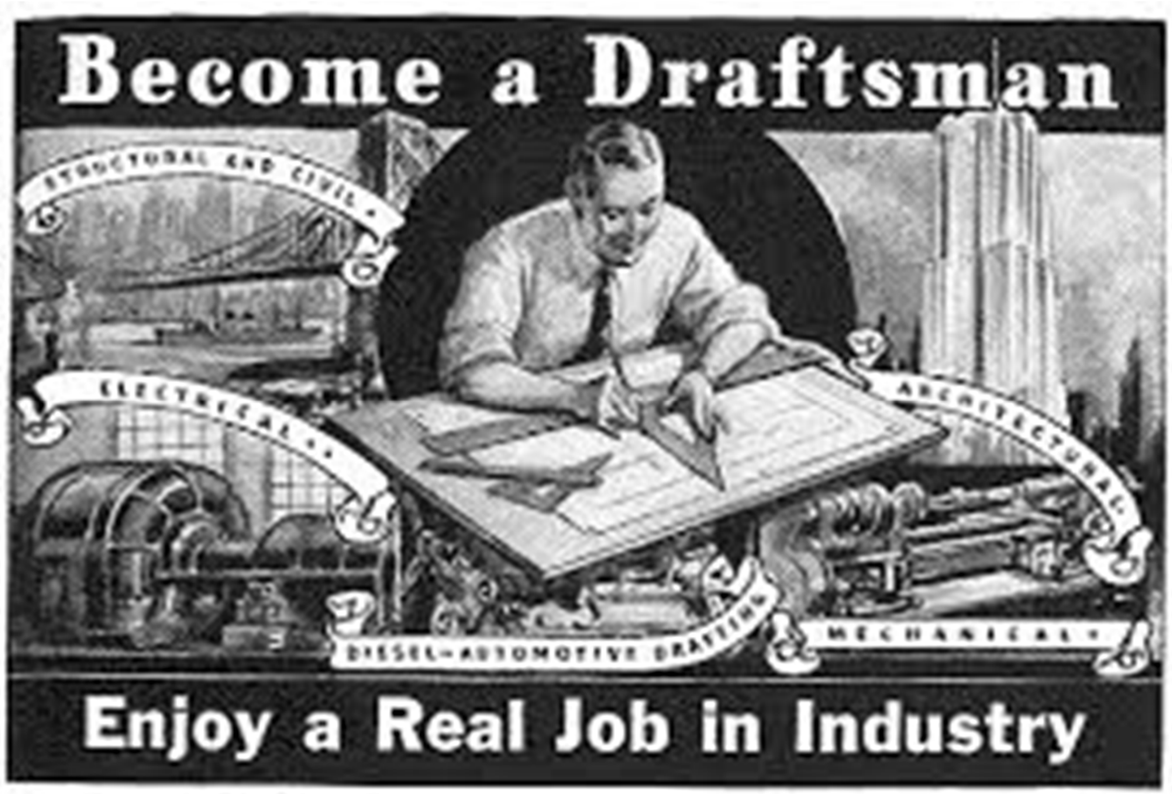


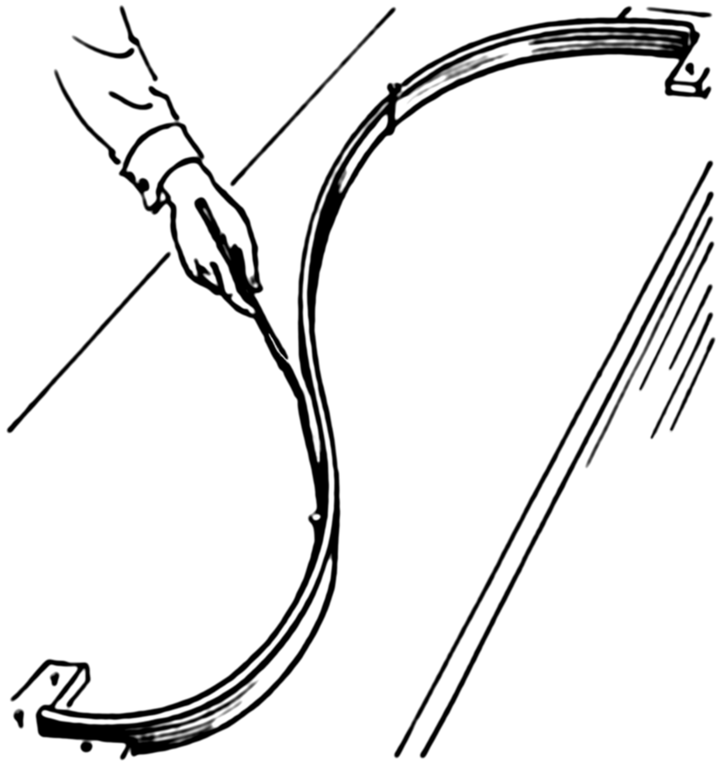

<u>Piecewise or spline interpolation</u>

  - Knots
    - The data points where the polynomials from two adjacent intervals meet.

## Spline ?

- The name "spline" comes from a draftsman's spline, which is a thin flexible rod used to physically interpolate over discrete points marked by pegs.



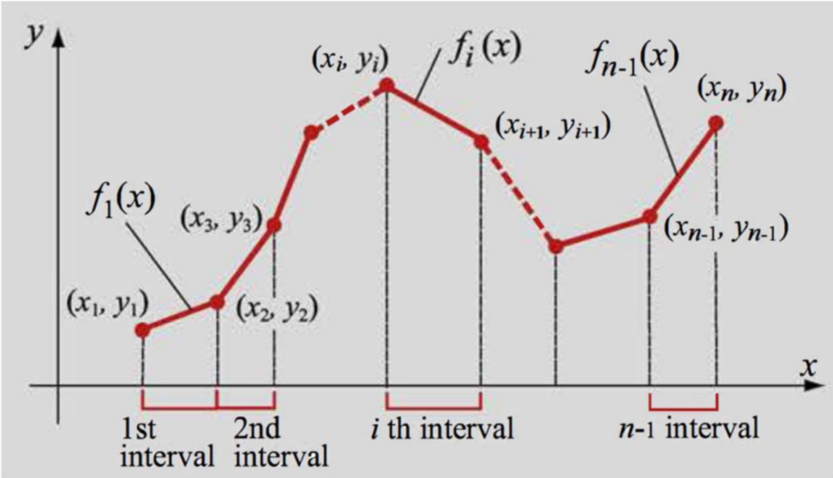
## Linear splines

- Straight line in Lagrange form
  - For $n$ given points, there are $(n-1)$ intervals.

$$f_1(x) = \frac{(x-x_2)}{(x_1-x_2)}y_1 + \frac{(x-x_1)}{(x_2-x_1)}y_2$$

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

$$\text{for} \quad i = 1, 2, \ldots, n-1$$




- Continuous interpolation since the two adjacent polynomials have the same value at a common knot

## Linear splines

- Example

**Example 6-6: Linear splines.**

The set of the following four data points is given:

| | | | | |
|---|---|---|---|---|
| $x$ | 8 | 11 | 15 | 18 |
| $y$ | 5 | 9 | 10 | 8 |

(*a*) Determine the linear splines that fit the data.
(*b*) Determine the interpolated value for $x = 12.7$.
(*c*) Write a MATLAB user-defined function for interpolation with linear splines. The inputs to the function are the coordinates of the given data points and the $x$ coordinate of the point at which $y$ is to be interpolated. The output from the function is the interpolated $y$ value at the given point. Use the function for determining the interpolated value of $y$ for $x = 12.7$.

$$f_1(x) = \frac{(x-x_2)}{(x_1-x_2)}y_1 + \frac{(x-x_1)}{(x_2-x_1)}y_2 = \frac{(x-11)}{(8-11)}5 + \frac{(x-8)}{(11-8)}9 = \frac{5}{-3}(x-11) + \frac{9}{2}(x-8) \quad \text{for} \quad 8 \le x \le 11$$

$$f_2(x) = \frac{(x-x_3)}{(x_2-x_3)}y_2 + \frac{(x-x_2)}{(x_3-x_2)}y_3 = \frac{(x-15)}{(11-15)}9 + \frac{(x-11)}{(15-11)}10 = \frac{9}{-4}(x-15) + \frac{10}{4}(x-11) \quad \text{for} \quad 11 \le x \le 15$$

$$f_3(x) = \frac{(x-x_4)}{(x_3-x_4)}y_3 + \frac{(x-x_3)}{(x_4-x_3)}y_4 = \frac{(x-18)}{(15-18)}10 + \frac{(x-15)}{(18-15)}8 = \frac{10}{-3}(x-18) + \frac{8}{3}(x-15) \quad \text{for} \quad 15 \le x \le 18$$

$$f_2(12.7) = \frac{9}{-4}(12.7-15) + \frac{10}{4}(12.7-11) = 9.425$$

## Linear splines

- Example

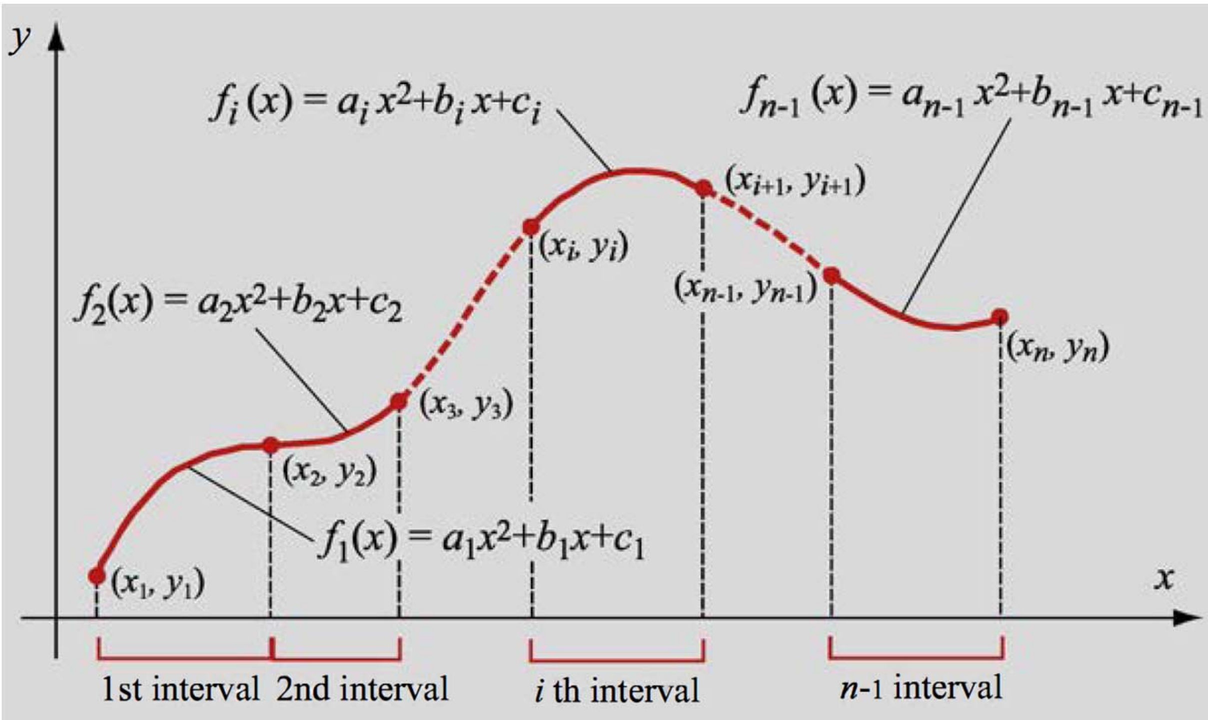
```
function Yint = LinearSpline(x, y, Xint)
% LinearSpline calculates interpolation using linear splines.
% Input variables:
% x    A vector with the coordinates x of the data points.
% y    A vector with the coordinates y of the data points.
% Xint The x coordinate of the interpolated point.
% Output variable:
% Yint The y value of the interpolated point.

n = length(x);
for i = 2:n
    if Xint < x(i)
        break
    end
end
Yint = (Xint - x(i))*y(i-1)/(x(i-1)-x(i)) + (Xint - x(i-1))*y(i)/(x(i)-x(i-1));
```

## Quadratic splines

- For $n$ given points, there are $(n-1)$ intervals.

$$f_i(x) = a_i x^2 + b_i x + c_i \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

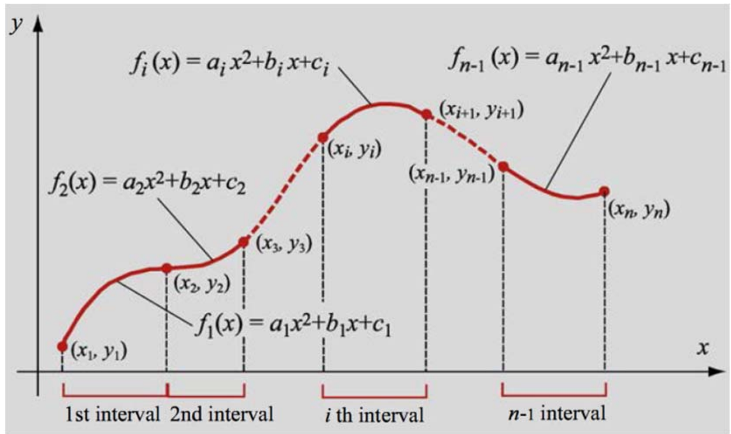
- Number of coefficients: $3(n-1)$



# 6.6 Piecewise (spline) Interpolation

## Quadratic splines

- Each polynomial must pass through the endpoints of the interval
  - $(x_i, y_i)$ $and$ $(x_{i+1}, y_{i+1})$
  - $f_i(x_i) = y_i$ $and$ $f_i(x_{i+1}) = y_{i+1}$




$$a_i x_i^2 + b_i x_i + c_i = y_i \quad \text{for} \quad i = 1, 2, \ldots, n-1$$
$$a_i x_{i+1}^2 + b_i x_{i+1} + c_i = y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

  - $2(n-1)$ equations

- At the interior knots
  - Slopes (the first derivative) of the polynomials from adjacent intervals are equal.
  - **The slope is continuous!**

$$f'(x) = \frac{df}{dx} = 2a_i x + b_i \qquad 2a_{i-1}x_i + b_{i-1} = 2a_i x_i + b_i \quad \text{for} \quad i = 2, 3, \ldots, n-1$$

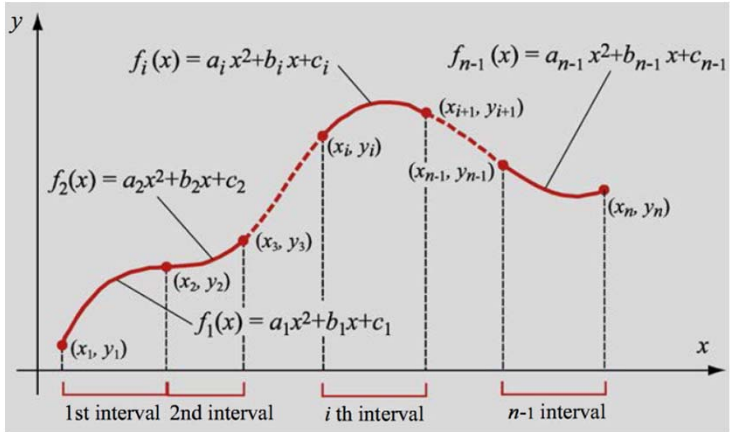
  - $(n-2)$ equations $\Rightarrow$ one more equation is necessary!

## Quadratic splines

- At the first point (or the last points)
  - The second derivative is zero.

$$f_1(x) = a_1x^2 + b_1x + c_1 \Rightarrow a_1 = 0$$

  - A straight line connects the first two points!



## Quadratic splines

- Example 6-7

**Example 6-7: Quadratic splines.**

The set of the following five data points is given:

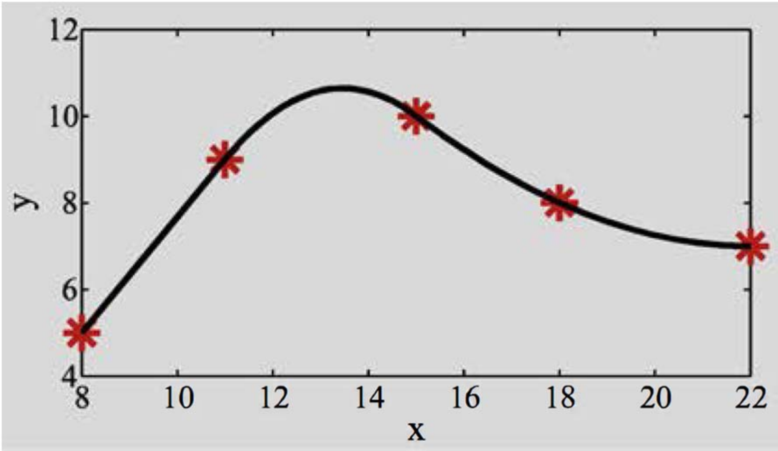
| | | | | | |
|---|---|---|---|---|---|
| $x$ | 8 | 11 | 15 | 18 | 22 |
| $y$ | 5 | 9 | 10 | 8 | 7 |

(*a*) Determine the quadratic splines that fit the data.
(*b*) Determine the interpolated value of $y$ for $x = 12.7$ .
(*c*) Make a plot of the data points and the interpolating polynomials.

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

4 splines: 8 equations from endpoints of intervals
+ 3 equations from the internal knots
+ 1 equations from the first point

$$\begin{bmatrix} 8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 11^2 & 11 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 15^2 & 15 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 18^2 & 18 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 22^2 & 22 & 1 \\ 1 & 0 & -22 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 30 & 1 & 0 & -30 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 36 & 1 & 0 & -36 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ c_1 \\ a_2 \\ b_2 \\ c_2 \\ a_3 \\ b_3 \\ c_3 \\ a_4 \\ b_4 \\ c_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 9 \\ 10 \\ 10 \\ 8 \\ 8 \\ 7 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$



## Cubic splines

- For $n$ given points, there are $(n-1)$ intervals.

- Standard form

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$

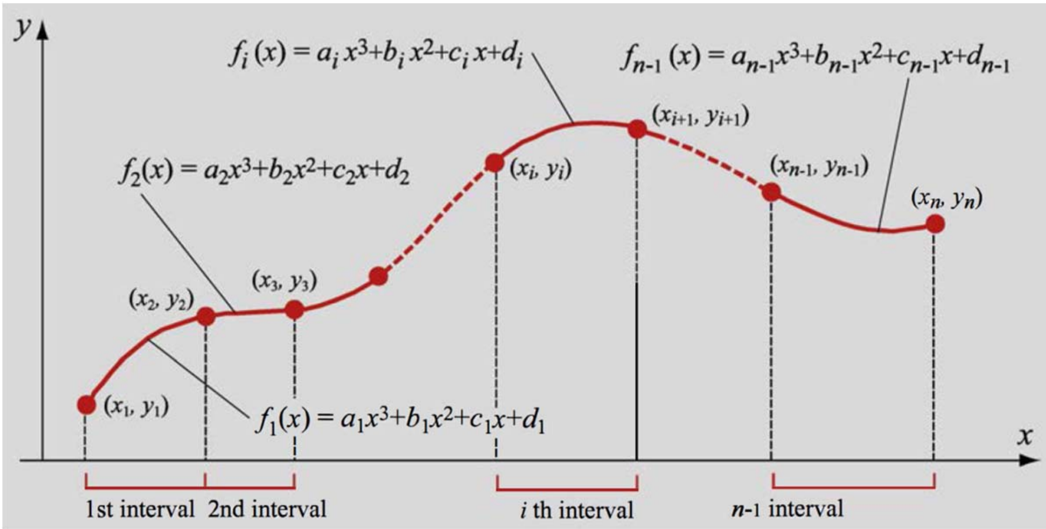
  - $4(n-1)$ linear equations

- Lagrange form

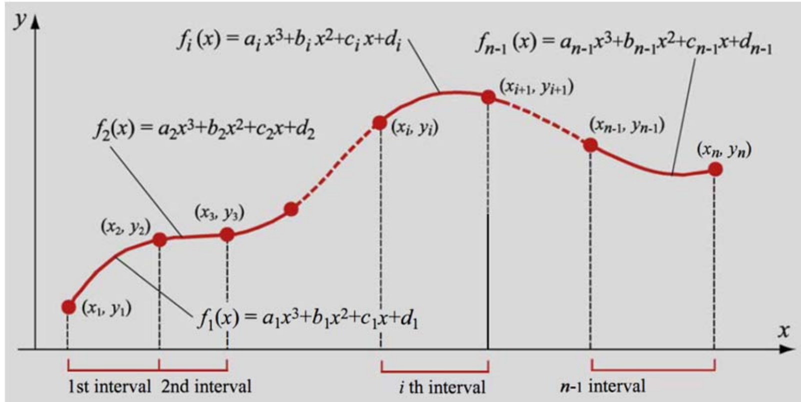
$$f(x) = \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} y_1$$

$$\cdots + \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} y_4$$

  - $n-2$ linear equations



## Cubic splines with standard form polynomials




$$f_i(x) = a_ix^3 + b_ix^2 + c_ix + d_i$$

- For $n$ given points, there are $(n-1)$ intervals.
- $4(n-1)$ cofficients

- From endpoints of the interval: $2(n-1)$ equations
  - $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$

$$a_ix_i^3 + b_ix_i^2 + c_ix_i + d_i = y_i \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

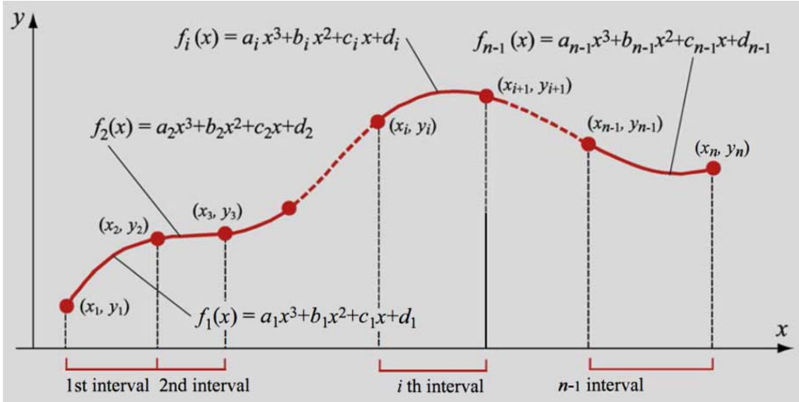
$$a_ix_{i+1}^3 + b_ix_{i+1}^2 + c_ix_{i+1} + d_i = y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

- At the interior knots: $(n-2)$ equations
  - First derivative of the *i*-th polynomial

$$f_i'(x) = \frac{df_i}{dx} = 3a_ix^2 + 2b_ix + c_i$$

$$3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_{i-1} = 3a_ix_i^2 + 2b_ix_i + c_i$$

## Cubic splines with standard form polynomials




- At the interior knots: $(n-2)$ equations
  - Second derivative of the *i*-th polynomial

$$f_i''(x) = \frac{d^2 f_i}{dx^2} = 6a_i x + 2b_i$$

$$6a_{i-1}x_i + 2b_{i-1} = 6a_i x_i + 2b_i \quad \text{for} \quad i = 2, 3, \ldots, n-1$$

- Two more equations at the first and last points
  - Second derivative is zero.

$$6a_1x_1 + 2b_1 = 0 \text{ and } 6a_{n-1}x_n + 2b_{n-1} = 0$$

  - **Natural cubic splines**

## Cubic splines based on Lagrange form polynomials

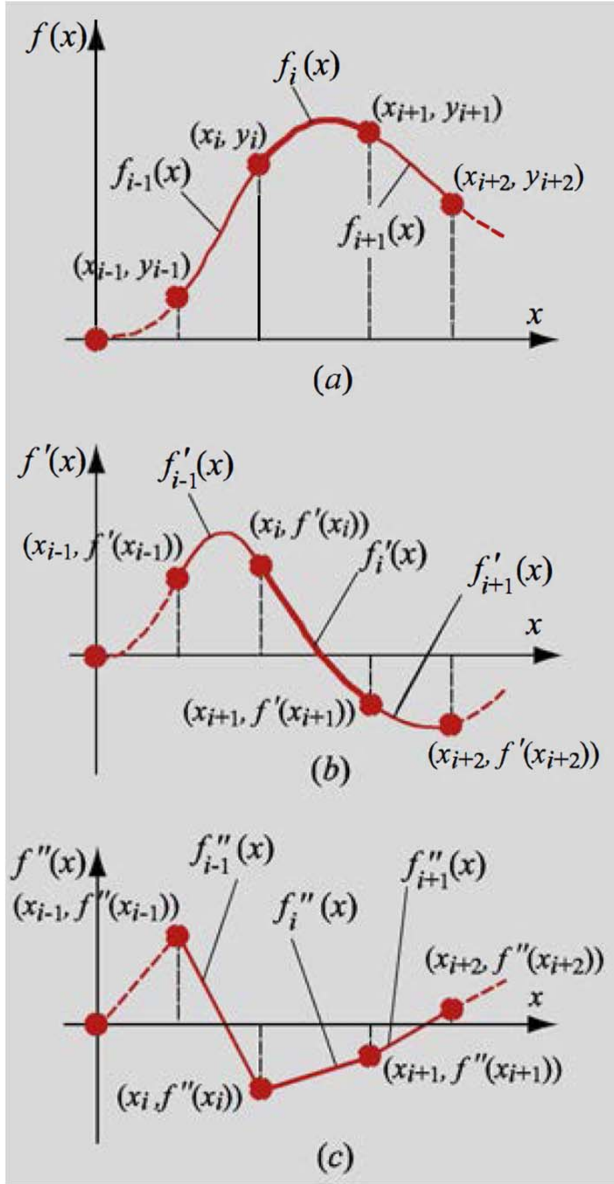
- Second derivative
  - Linear function of $x$

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

- Integrating twice : two constants
  - Determined with the values at the knots

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

$$\text{for } x_i \le x \le x_{i+1} \qquad \text{and } i = 1, 2, \ldots, n - 1$$



## Cubic splines based on Lagrange form polynomials

- First derivative at the interior knots

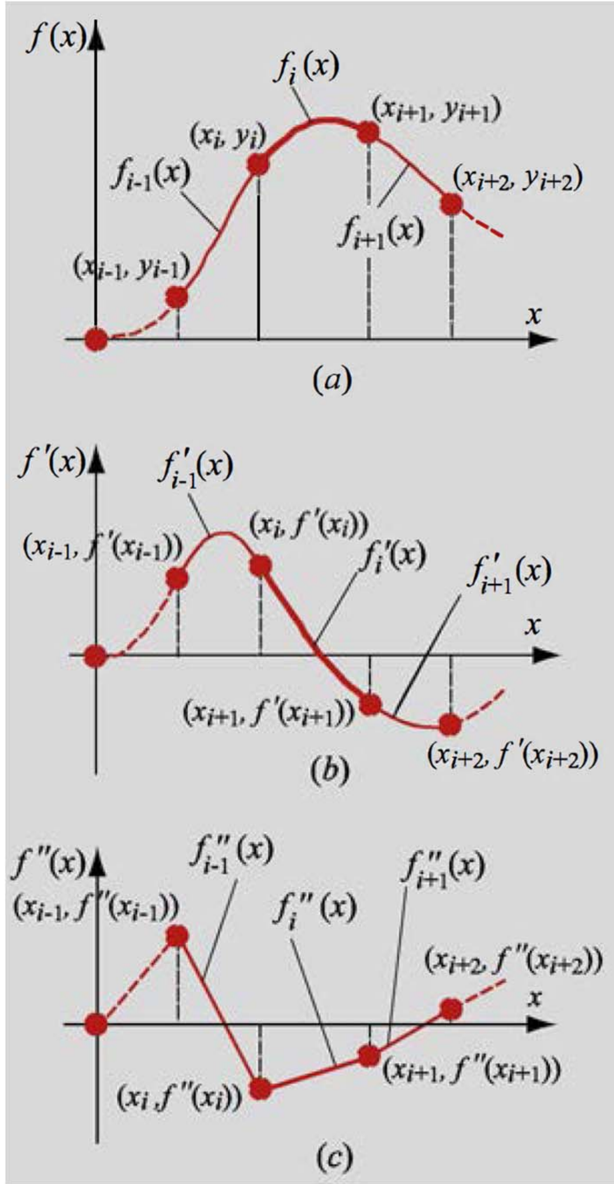
$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \quad \text{for} \quad i = 1, 2, \ldots, n-2$$

$$(x_{i+1}-x_i)f''(x_i) + 2(x_{i+2}-x_i)f''(x_{i+1}) + (x_{i+2}-x_{i+1})f''(x_{i+2}) = 6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} - \frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right] \quad \text{for} \quad i = 1, 2, \ldots, n-2$$

  - System of $(n-2)$ linear equations with $n$ unknowns

- Second derivative at the endpoints of the data is set to zero.
  - Natural cubic splines

$$f''(x_1) = 0 \quad \text{and} \quad f''(x_n) = 0$$



## Cubic splines based on Lagrange form polynomials

- Original form of polynomials

$$f_i(x) = \frac{f_i''(x_i)}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1}-x_i)}(x-x_i)^3$$
$$+\left[\frac{y_i}{x_{i+1}-x_i} - \frac{f_i''(x_i)(x_{i+1}-x_i)}{6}\right](x_{i+1}-x)$$
$$+\left[\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_i''(x_{i+1})(x_{i+1}-x_i)}{6}\right](x-x_i)$$
$$\text{for } x_i \le x \le x_{i+1} \quad \text{and } i = 1, 2, \ldots, n-1$$

$$h_i = x_{i+1} - x_i$$

$$a_i = f''(x_i)$$

- Simplified form of polynomials

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3$$
$$+\left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1} h_i}{6}\right](x-x_i)$$

$$\text{for } x_i \le x \le x_{i+1} \quad \text{and} \quad i = 1, 2, \ldots, n-1$$

## ❖ Cubic splines based on Lagrange form polynomials

- Original form of the system of linear equations

$$(x_{i+1}-x_i)f''(x_i)+2(x_{i+2}-x_i)f''(x_{i+1})+(x_{i+2}-x_{i+1})f''(x_{i+2})$$
$$=6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}}-\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right]$$
$$\text{for}\quad i=1,2,\ldots,n-2$$

- Simplified form of the system of linear equations

$$h_i a_i+2(h_i+h_{i+1})a_{i+1}+h_{i+1}a_{i+2}=6\left[\frac{y_{i+2}-y_{i+1}}{h_{i+1}}-\frac{y_{i+1}-y_i}{h_i}\right]$$
$$\text{for}\quad i=1,2,\ldots,n-2$$

  - Natural cubic spline: $a_1$ and $a_n=0$
  - $(n-2)$ equations for $(n-2)$ unknowns

## Cubic splines based on Lagrange form polynomials

- Example

**Example 6-8: Cubic splines.**

The set of the following five data points is given:

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 8 | 11 | 15 | 18 | 22 |
| $y$ | 5 | 9 | 10 | 8 | 7 |

(*a*) Determine the natural cubic splines that fit the data.
(*b*) Determine the interpolated value of $y$ for $x = 12.7$.
(*c*) Plot of the data points and the interpolating polynomials.

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3 + \left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1}h_i}{6}\right](x-x_i) \quad \text{for} \quad i = 1, \ldots 4$$

# 6.6 Piecewise (spline) Interpolation

## ❖ Cubic splines based on Lagrange form polynomials

- Example

$$i = 1 \quad h_1 a_1 + 2(h_1 + h_2)a_2 + h_2 a_3 = 6\left[\frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1}\right]$$

$$3 \cdot 0 + 2(3+4)a_2 + 4a_3 = 6\left[\frac{10-9}{4} - \frac{9-5}{3}\right] \longrightarrow 14a_2 + 4a_3 = -6.5$$

$$i = 2 \quad h_2 a_2 + 2(h_2 + h_3)a_3 + h_3 a_4 = 6\left[\frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2}\right]$$

$$4a_2 + 2(4+3)a_3 + 3a_4 = 6\left[\frac{8-10}{3} - \frac{10-9}{4}\right] \longrightarrow 4a_2 + 14a_3 + 3a_4 = -5.5$$

$$i = 3 \quad h_3 a_3 + 2(h_3 + h_4)a_4 + h_4 a_5 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right]$$

$$3a_3 + 2(3+4)a_4 + 4 \cdot 0 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right] \longrightarrow 3a_3 + 14a_4 = 2.5$$

$$\begin{bmatrix} 14 & 4 & 0 \\ 4 & 14 & 3 \\ 0 & 3 & 14 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -6.5 \\ -5.5 \\ 2.5 \end{bmatrix}$$

```
-0.3665 -0.3421 0.2519
```

$a_2 \qquad a_3 \qquad a_4$

## Cubic splines based on Lagrange form polynomials

- Example

$$f_1(x) = \frac{0}{6\cdot 3}(11-x)^3 + \frac{-0.3665}{6\cdot 3}(x-8)^3 + \left[\frac{5}{3} - \frac{0\cdot 3}{6}\right](11-x) + \left[\frac{9}{3} - \frac{-0.3665\cdot 3}{6}\right](x-8)$$

$$f_1(x) = (-0.02036)(x-8)^3 + 1.667(11-x) + 3.183(x-8) \quad \text{for } 8 \le x \le 11$$

$$f_2(x) = \frac{-0.3665}{6\cdot 4}(15-x)^3 + \frac{-0.3421}{6\cdot 4}(x-11)^3 + \left[\frac{9}{4} - \frac{-0.3665\cdot 4}{6}\right](15-x) + \left[\frac{10}{4} - \frac{-0.3421\cdot 4}{6}\right](x-11)$$

$$f_2(x) = (-0.01527)(15-x)^3 + (-0.01427)(x-11)^3 + 2.494(15-x) + 2.728(x-11) \quad \text{for } 11 \le x \le 15$$

$$f_3(x) = \frac{-0.3421}{6\cdot 3}(18-x)^3 + \frac{0.2519}{6\cdot 3}(x-15)^3 + \left[\frac{10}{3} - \frac{-0.3421\cdot 3}{6}\right](18-x) + \left[\frac{8}{3} - \frac{0.2519\cdot 3}{6}\right](x-15)$$

$$f_3(x) = (-0.019)(18-x)^3 + 0.014(x-15)^3 + 3.504(18-x) + 2.5407(x-15) \quad \text{for } 15 \le x \le 18$$

$$f_4(x) = \frac{0.2519}{6\cdot 4}(22-x)^3 + \frac{0}{6\cdot 4}(x-18)^3 + \left[\frac{8}{4} - \frac{0.2519\cdot 4}{6}\right](22-x) + \left[\frac{7}{4} - \frac{0\cdot 4}{6}\right](x-18)$$

$$f_4(x) = 0.0105(22-x)^3 + 1.832(22-x) + 1.75(x-18) \quad \text{for } 18 \le x \le 22$$

## Cubic splines based on Lagrange form polynomials

- Example

$f_1(x) = \frac{0}{6 \cdot 3}$ … $-8)$

$f_1(x) = (-0.$ …

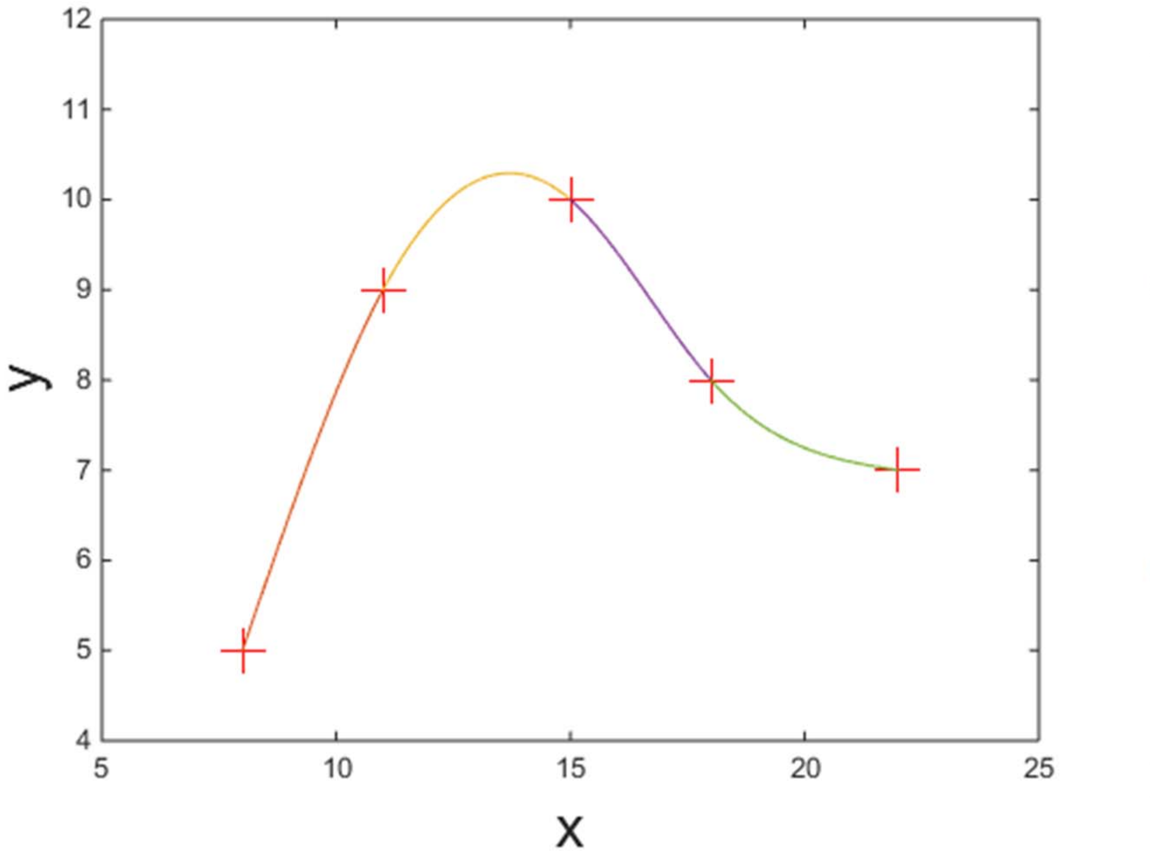
$f_2(x) = \frac{-0.3}{6 \cdot}$ … $\frac{.3421 \cdot 4}{6}\Big](x-11)$

$f_2(x) = (-0.$ … for $11 \le x \le 15$

$f_3(x) = \frac{-0.34}{6 \cdot}$ … $\frac{19 \cdot 3}{6}\Big](x-15)$

$f_3(x) = (-0.$ … $15 \le x \le 18$

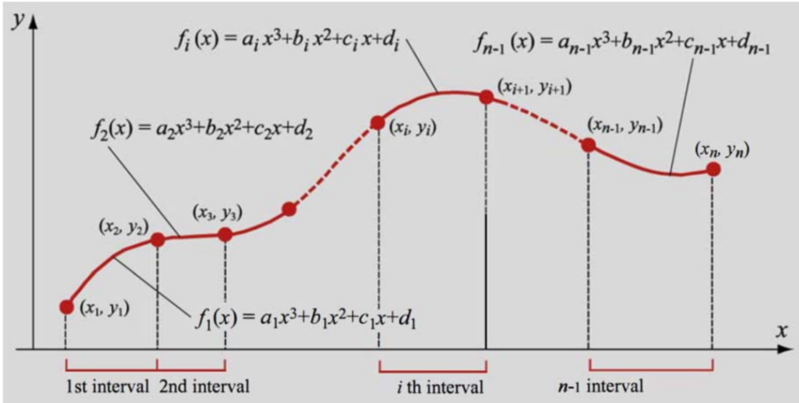
$f_4(x) = \frac{0.251}{6 \cdot 4}$ … $18)$




$f_4(x) = 0.0105(22-x)^3 + 1.832(22-x) + 1.75(x-18)$ for $18 \le x \le 22$

## ❖ Cubic splines with standard form polynomials

$$f_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i$$




- For $n$ given points, there are $(n-1)$ intervals.
- $4(n-1)$ cofficients

- From endpoints of the interval: $2(n-1)$ equations
  - $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$

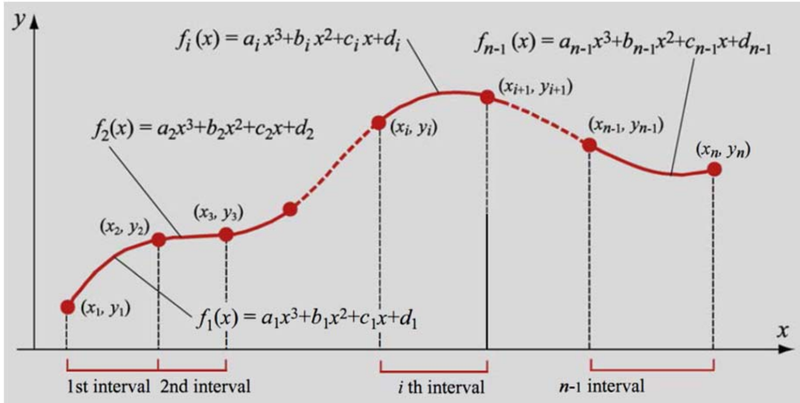
$$a_i x_i^3 + b_i x_i^2 + c_i x_i + d_i = y_i \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

$$a_i x_{i+1}^3 + b_i x_{i+1}^2 + c_i x_{i+1} + d_i = y_{i+1} \quad \text{for} \quad i = 1, 2, \ldots, n-1$$

- At the interior knots: $(n-2)$ equations
  - First derivative of the *i*-th polynomial

$$f_i'(x) = \frac{df_i}{dx} = 3a_i x^2 + 2b_i x + c_i \qquad 3a_{i-1}x_i^2 + 2b_{i-1}x_i + c_{i-1} = 3a_i x_i^2 + 2b_i x_i + c_i$$

## Cubic splines with standard form polynomials




- At the interior knots: $(n-2)$ equations
  - Second derivative of the *i*-th polynomial

$$f_i''(x) = \frac{d^2 f_i}{dx^2} = 6a_i x + 2b_i$$

$$6a_{i-1}x_i + 2b_{i-1} = 6a_i x_i + 2b_i \text{ for } i = 2, 3, \ldots, n-1$$

- Two more equations at the first and last points
  - Second derivative is zero.

$$6a_1 x_1 + 2b_1 = 0 \text{ and } 6a_{n-1}x_n + 2b_{n-1} = 0$$

  - **Natural cubic splines**

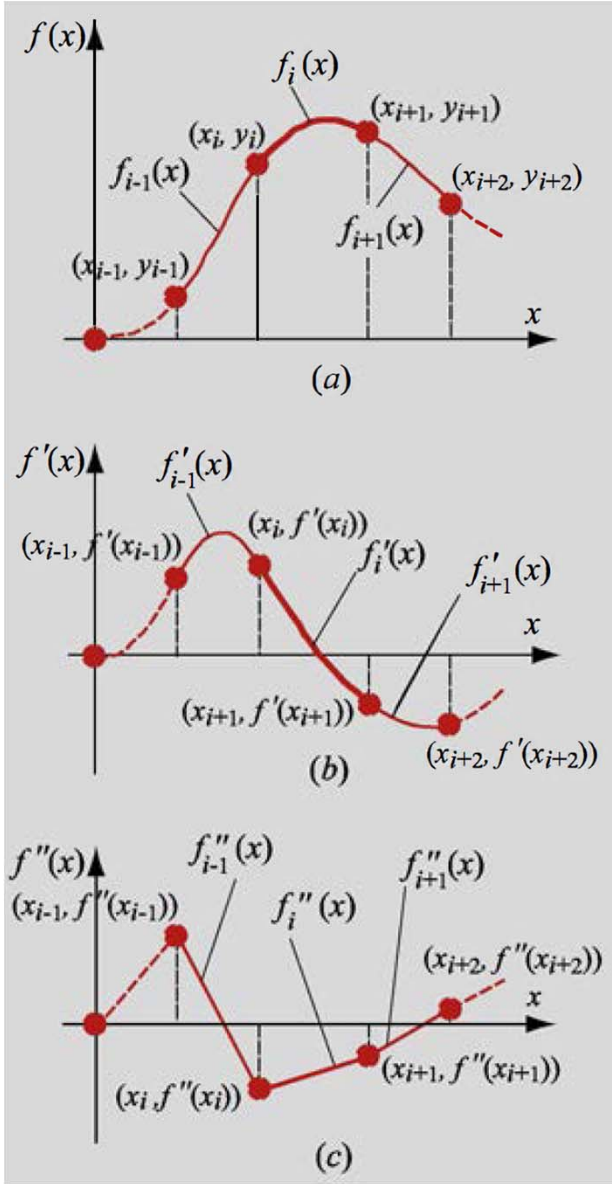
## ❖ Cubic splines based on Lagrange form polynomials

- Second derivative
  - Linear function of $x$

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

- Integrating twice : two constants
  - Determined with the values at the knots

$$\begin{aligned} f_i(x) = {} & \frac{f_i''(x_i)}{6(x_{i+1} - x_i)}(x_{i+1} - x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1} - x_i)}(x - x_i)^3 \\ & + \left[\frac{y_i}{x_{i+1} - x_i} - \frac{f_i''(x_i)(x_{i+1} - x_i)}{6}\right](x_{i+1} - x) \\ & + \left[\frac{y_{i+1}}{x_{i+1} - x_i} - \frac{f_i''(x_{i+1})(x_{i+1} - x_i)}{6}\right](x - x_i) \\ & \text{for } x_i \le x \le x_{i+1} \qquad \text{and } i = 1, 2, \ldots, n-1 \end{aligned}$$



## Cubic splines based on Lagrange form polynomials

- First derivative at the interior knots

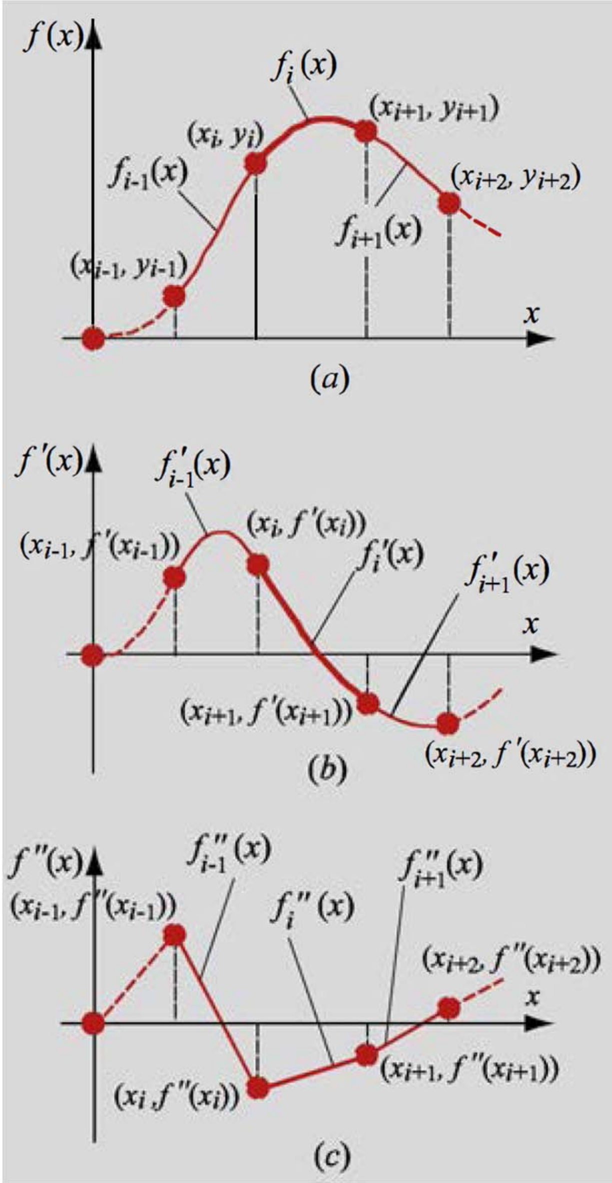
$$f_i'(x_{i+1}) = f_{i+1}'(x_{i+1}) \quad \text{for} \quad i = 1, 2, \ldots, n-2$$

$$(x_{i+1}-x_i)f''(x_i) + 2(x_{i+2}-x_i)f''(x_{i+1}) + (x_{i+2}-x_{i+1})f''(x_{i+2}) = 6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}} - \frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right] \quad \text{for} \quad i = 1, 2, \ldots, n-2$$

  - System of $(n-2)$ linear equations with $n$ unknowns

- Second derivative at the endpoints of the data is set to zero.
  - Natural cubic splines

$$f''(x_1) = 0 \quad \text{and} \quad f''(x_n) = 0$$



## Cubic splines based on Lagrange form polynomials

- Original form of polynomials

$$f_i(x) = \frac{f_i''(x_i)}{6(x_{i+1}-x_i)}(x_{i+1}-x)^3 + \frac{f_i''(x_{i+1})}{6(x_{i+1}-x_i)}(x-x_i)^3$$
$$+\left[\frac{y_i}{x_{i+1}-x_i} - \frac{f_i''(x_i)(x_{i+1}-x_i)}{6}\right](x_{i+1}-x)$$
$$+\left[\frac{y_{i+1}}{x_{i+1}-x_i} - \frac{f_i''(x_{i+1})(x_{i+1}-x_i)}{6}\right](x-x_i)$$
$$\text{for } x_i \le x \le x_{i+1} \quad \text{and } i = 1, 2, \ldots, n-1$$

$$h_i = x_{i+1} - x_i$$

$$a_i = f''(x_i)$$

- Simplified form of polynomials

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3$$
$$+\left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1}h_i}{6}\right](x-x_i)$$

$$\text{for } x_i \le x \le x_{i+1} \quad \text{and} \quad i = 1, 2, \ldots, n-1$$

## Cubic splines based on Lagrange form polynomials

- Original form of the system of linear equations

$$(x_{i+1}-x_i)f''(x_i)+2(x_{i+2}-x_i)f''(x_{i+1})+(x_{i+2}-x_{i+1})f''(x_{i+2})$$
$$=6\left[\frac{y_{i+2}-y_{i+1}}{x_{i+2}-x_{i+1}}-\frac{y_{i+1}-y_i}{x_{i+1}-x_i}\right]$$
$$\text{for} \quad i=1,2,\ldots,n-2$$

- Simplified form of the system of linear equations

$$h_i a_i+2(h_i+h_{i+1})a_{i+1}+h_{i+1}a_{i+2}=6\left[\frac{y_{i+2}-y_{i+1}}{h_{i+1}}-\frac{y_{i+1}-y_i}{h_i}\right]$$
$$\text{for} \quad i=1,2,\ldots,n-2$$

  - Natural cubic spline: $a_1$ and $a_n=0$
  - $(n-2)$ equations for $(n-2)$ unknowns

## Cubic splines based on Lagrange form polynomials

- Example

**Example 6-8: Cubic splines.**

The set of the following five data points is given:

| | | | | | |
|---|---|---|---|---|---|
| $x$ | 8 | 11 | 15 | 18 | 22 |
| $y$ | 5 | 9 | 10 | 8 | 7 |

(*a*) Determine the natural cubic splines that fit the data.
(*b*) Determine the interpolated value of $y$ for $x = 12.7$.
(*c*) Plot of the data points and the interpolating polynomials.

$$f_i(x) = \frac{a_i}{6h_i}(x_{i+1}-x)^3 + \frac{a_{i+1}}{6h_i}(x-x_i)^3 + \left[\frac{y_i}{h_i} - \frac{a_i h_i}{6}\right](x_{i+1}-x) + \left[\frac{y_{i+1}}{h_i} - \frac{a_{i+1}h_i}{6}\right](x-x_i) \quad \text{for} \quad i = 1, \ldots 4$$

# 6.6 Piecewise (spline) Interpolation

## ❖ Cubic splines based on Lagrange form polynomials

- Example

$$i = 1 \quad h_1 a_1 + 2(h_1 + h_2)a_2 + h_2 a_3 = 6\left[\frac{y_3 - y_2}{h_2} - \frac{y_2 - y_1}{h_1}\right]$$

$$3 \cdot 0 + 2(3 + 4)a_2 + 4a_3 = 6\left[\frac{10 - 9}{4} - \frac{9 - 5}{3}\right] \longrightarrow 14a_2 + 4a_3 = -6.5$$

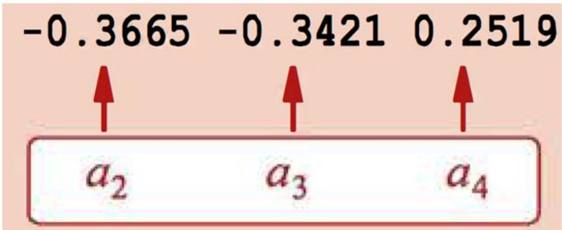
$$i = 2 \quad h_2 a_2 + 2(h_2 + h_3)a_3 + h_3 a_4 = 6\left[\frac{y_4 - y_3}{h_3} - \frac{y_3 - y_2}{h_2}\right]$$

$$4a_2 + 2(4 + 3)a_3 + 3a_4 = 6\left[\frac{8 - 10}{3} - \frac{10 - 9}{4}\right] \longrightarrow 4a_2 + 14a_3 + 3a_4 = -5.5$$

$$i = 3 \quad h_3 a_3 + 2(h_3 + h_4)a_4 + h_4 a_5 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right]$$

$$3a_3 + 2(3 + 4)a_4 + 4 \cdot 0 = 6\left[\frac{y_5 - y_4}{h_4} - \frac{y_4 - y_3}{h_3}\right] \longrightarrow 3a_3 + 14a_4 = 2.5$$

$$\begin{bmatrix} 14 & 4 & 0 \\ 4 & 14 & 3 \\ 0 & 3 & 14 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} -6.5 \\ -5.5 \\ 2.5 \end{bmatrix}$$



## Cubic splines based on Lagrange form polynomials

- Example

$$f_1(x) = \frac{0}{6 \cdot 3}(11-x)^3 + \frac{-0.3665}{6 \cdot 3}(x-8)^3 + \left[\frac{5}{3} - \frac{0 \cdot 3}{6}\right](11-x) + \left[\frac{9}{3} - \frac{-0.3665 \cdot 3}{6}\right](x-8)$$
$$f_1(x) = (-0.02036)(x-8)^3 + 1.667(11-x) + 3.183(x-8) \quad \text{for } 8 \le x \le 11$$

$$f_2(x) = \frac{-0.3665}{6 \cdot 4}(15-x)^3 + \frac{-0.3421}{6 \cdot 4}(x-11)^3 + \left[\frac{9}{4} - \frac{-0.3665 \cdot 4}{6}\right](15-x) + \left[\frac{10}{4} - \frac{-0.3421 \cdot 4}{6}\right](x-11)$$
$$f_2(x) = (-0.01527)(15-x)^3 + (-0.01427)(x-11)^3 + 2.494(15-x) + 2.728(x-11) \quad \text{for } 11 \le x \le 15$$

$$f_3(x) = \frac{-0.3421}{6 \cdot 3}(18-x)^3 + \frac{0.2519}{6 \cdot 3}(x-15)^3 + \left[\frac{10}{3} - \frac{-0.3421 \cdot 3}{6}\right](18-x) + \left[\frac{8}{3} - \frac{0.2519 \cdot 3}{6}\right](x-15)$$
$$f_3(x) = (-0.019)(18-x)^3 + 0.014(x-15)^3 + 3.504(18-x) + 2.5407(x-15) \quad \text{for } 15 \le x \le 18$$

$$f_4(x) = \frac{0.2519}{6 \cdot 4}(22-x)^3 + \frac{0}{6 \cdot 4}(x-18)^3 + \left[\frac{8}{4} - \frac{0.2519 \cdot 4}{6}\right](22-x) + \left[\frac{7}{4} - \frac{0 \cdot 4}{6}\right](x-18)$$
$$f_4(x) = 0.0105(22-x)^3 + 1.832(22-x) + 1.75(x-18) \quad \text{for } 18 \le x \le 22$$

## Cubic splines based on Lagrange form polynomials

- Example

$f_1(x) = \frac{0}{6 \cdot 3}$ … $-8)$

$f_1(x) = (-0.$ …

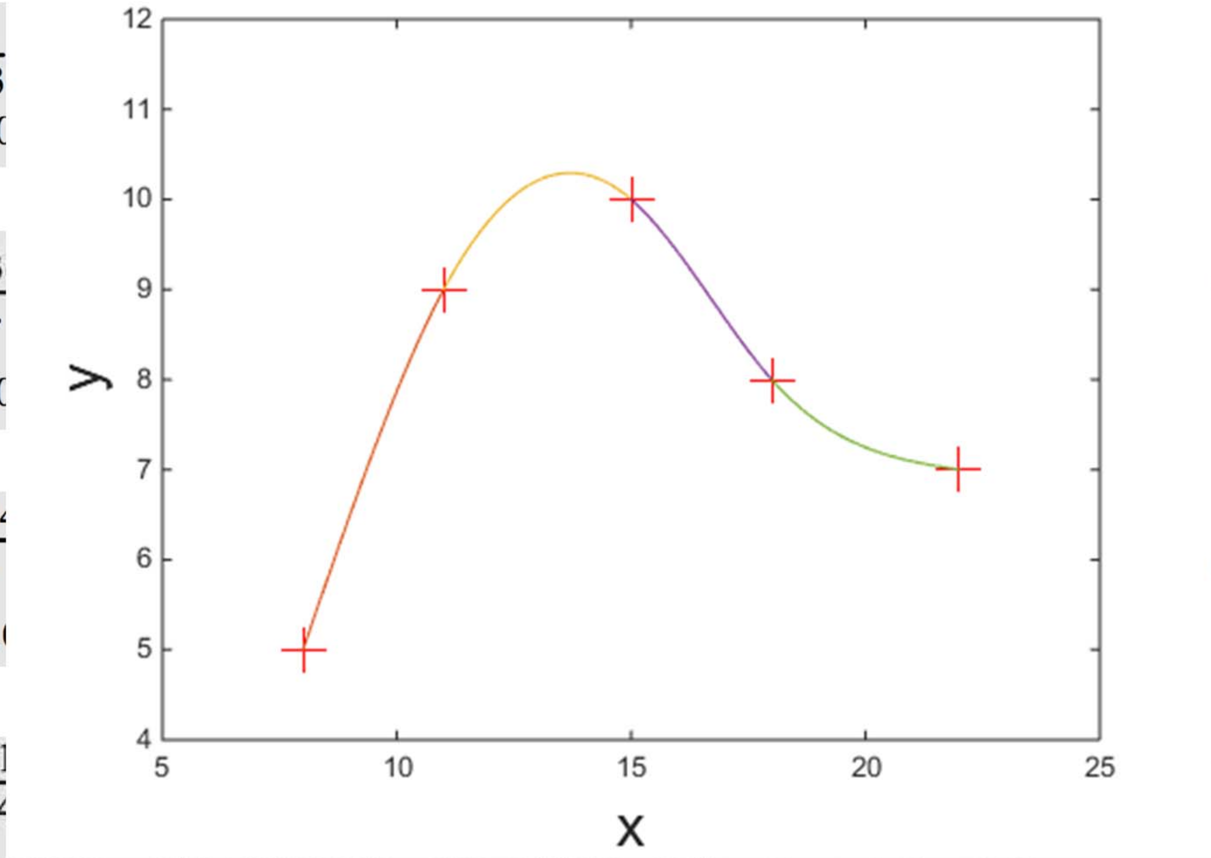
$f_2(x) = \frac{-0.3}{6 \cdot}$ … $\frac{.3421 \cdot 4}{6}\Big](x-11)$

$f_2(x) = (-0.$ … for $11 \le x \le 15$

$f_3(x) = \frac{-0.34}{6 \cdot}$ … $\frac{19 \cdot 3}{6}\Big](x-15)$

$f_3(x) = (-0.$ … $15 \le x \le 18$

$f_4(x) = \frac{0.251}{6 \cdot 4}$ … $18)$




$f_4(x) = 0.0105(22-x)^3 + 1.832(22-x) + 1.75(x-18)$ for $18 \le x \le 22$

# 6.7 Use of MATLAB Built-in Functions

## MATLAB built-in functions

- The `polyfit` command

```
p = polyfit(x,y,m)
```

`p` is a vector of the coefficients of the polynomial that best fits the data.

`x` and `y` are vectors with the horizontal and vertical coordinates of the data points, respectively.

`m` is the degree of the polynomial.

- The `interp1` command

```
yi = interp1(x,y,xi,'method')
```

`yi` is the interpolated value (dependent variable).

`x` and `y` are vectors with the independent variable and dependent variable of the data points, respectively.
`xi` is the value of $x$ at which $y$ is to be interpolated.

Method of interpolation, typed as a string (optional).

## MATLAB built-in functions

- The `interp1` command (method, check MATLBA help)

| method string | Description | Continuity | Comments |
|---|---|---|---|
| 'linear' | Linear interpolation. The interpolated value at a query point is based on linear interpolation of the values at neighboring grid points in each respective dimension. This is the default interpolation method. | $C^0$ | • Requires at least 2 points. • Requires more memory and computation time than nearest neighbor. |
| 'nearest' | Nearest neighbor interpolation. The interpolated value at a query point is the value at the nearest sample grid point. | Discontinuous | • Requires at least 2 points. • Modest memory requirements • Fastest computation time |
| 'next' | Next neighbor interpolation. The interpolated value at a query point is the value at the next sample grid point. | Discontinuous | • Requires at least 2 points. • Same memory requirements and computation time as 'nearest'. |
| 'previous' | Previous neighbor interpolation. The interpolated value at a query point is the value at the previous sample grid point. | Discontinuous | • Requires at least 2 points. • Same memory requirements and computation time as 'nearest'. |
| 'pchip' | Shape-preserving piecewise cubic interpolation. The interpolated value at a query point is based on a shape-preserving piecewise cubic interpolation of the values at neighboring grid points. | $C^1$ | • Requires at least 4 points. • Requires more memory and computation time than linear. |
| 'cubic' | Same as 'pchip'. | $C^1$ | This method currently returns the same result as 'pchip'. In a future release, this method will perform cubic convolution. |
| 'v5cubic' | Cubic convolution used in MATLAB® 5. | $C^1$ | Points must be uniformly spaced. 'cubic' will replace 'v5cubic' in a future release. |
| 'spline' | Spline interpolation using not-a-knot end conditions. The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension. | $C^2$ | • Requires at least 4 points. • Requires more memory and computation time than 'p |

## MATLAB built-in functions

- The `polyfit` command

```
>> x = 0:0.4:6;
>> y = [0 3 4.5 5.8 5.9 5.8 6.2 7.4 9.6 15.6 20.7 26.7 31.1
35.6 39.3 41.5];
>> p = polyfit(x,y,4)
p =
   -0.2644    3.1185 -10.1927   12.8780   -0.2746
```

The polynomial that corresponds to these coefficients is:
$f(x) = (-0.2644)x^4 + 3.1185x^3 - 10.1927x^2 + 12.878x - 0.2746.$

## MATLAB built-in functions

- The `interp1` command

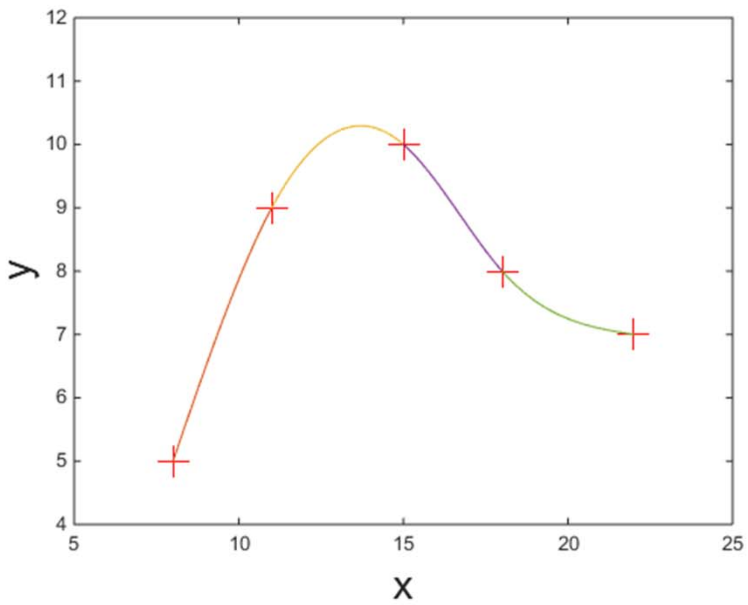
```
>> x = [8 11 15 18 22];
>> y = [5 9 10 8 7];
>> xint=8:0.1:22;
>> yint=interp1(x,y,xint,'pchip');
>> plot(x,y,'*',xint,yint)
```

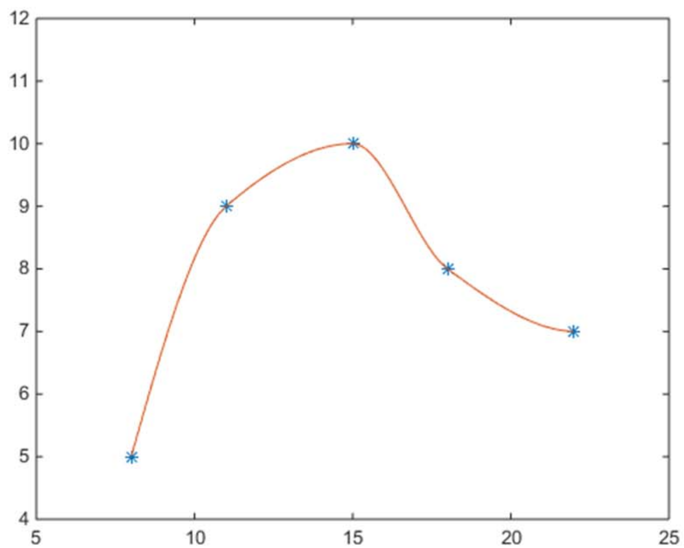
Assign the data points to x and y.

Vector with points for interpolation.

Calculate the interpolated values.

Create a plot with the data points and interpolated values.



# 6.8 Curve Fitting with a Linear Combination of Non-linear Functions

## Linear combination of non-linear functions

$$F(x) = C_1 f_1(x) + C_2 f_2(x) + C_3 f_3(x) + \ldots + C_m f_m(x) = \sum_{j=1}^{m} C_j f_j(x_i)$$

- $f_1, \cdots, f_m$ : prescribed functions
- $C_1, \cdots, C_m$ : unknown coefficients
- Given data points: $n$ points $(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)$

- Sum of the squares of the residuals

$$E = \sum_{i=1}^{n}\left[y_i - \sum_{j=1}^{m} C_j f_j(x_i)\right]^2$$

- Least square method

$$\frac{\partial E}{\partial C_k} = 0 \quad \text{for } k = 1, 2, \ldots, m$$

$$\frac{\partial E}{\partial C_k} = \sum_{i=1}^{n} 2\left[y_i - \sum_{j=1}^{m} C_j f_j(x_i)\right]\left[-\frac{\partial}{\partial C_k}\left(\sum_{j=1}^{m} C_j f_j(x_i)\right)\right] = 0$$

# 6.8 Curve Fitting with a Linear Combination of Non-linear Functions

## ❖ Linear combination of non-linear functions

- Least square method
  - The coefficients are Independent of each other

$$\frac{\partial E}{\partial C_k} = \sum_{i=1}^{n} 2\left[y_i - \sum_{j=1}^{m} C_j f_j(x_i)\right]\left[-\frac{\partial}{\partial C_k}\left(\sum_{j=1}^{m} C_j f_j(x_i)\right)\right] = 0$$

$$\frac{\partial}{\partial C_k}\left(\sum_{j=1}^{m} C_j f_j(x_i)\right) = f_k(x_i)$$

$$\frac{\partial E}{\partial C_k} = -\sum_{i=1}^{n} 2\left[y_i - \sum_{j=1}^{m} C_j f_j(x_i)\right] f_k(x_i) = 0$$

$$\boxed{\sum_{i=1}^{n}\sum_{j=1}^{m} C_j f_j(x_i)\, f_k(x_i) = \sum_{i=1}^{n} y_i\, f_k(x_i)}$$

$f_k(x_i)$ : chosen based on a guiding theory

# 6.8 Curve Fitting with a Linear Combination of Non-linear Functions

## Linear combination of non-linear functions

- Example 6-9

**Example 6-9: Curve fitting with linear combination of nonlinear functions.**

The following data is obtained from wind-tunnel tests, for the variation of the ratio of the tangential velocity of a vortex to the free stream flow velocity $y = V_\theta / V_\infty$ versus the ratio of the distance from the vortex core to the chord of an aircraft wing, $x = R/C$:

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $x$ | 0.6 | 0.8 | 0.85 | 0.95 | 1.0 | 1.1 | 1.2 | 1.3 | 1.45 | 1.6 | 1.8 |
| $y$ | 0.08 | 0.06 | 0.07 | 0.07 | 0.07 | 0.06 | 0.06 | 0.06 | 0.05 | 0.05 | 0.04 |

Theory predicts that the relationship between $x$ and $y$ should be of the form $y = \frac{A}{x} + \frac{Be^{-2x^2}}{x}$. Find the values of $A$ and $B$ using the least-squares method to fit the above data.

$$F(x) = C_1 f_1(x) + C_2 f_2(x)$$

$$C_1 = A, \ C_2 = B, \ f_1(x) = \frac{1}{x}, \text{ and } f_2(x) = \frac{e^{-2x^2}}{x}$$

$$\sum_{i=1}^{11} A \frac{1}{x_i} \frac{1}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i} \frac{1}{x_i} = \sum_{i=1}^{11} y_i \frac{1}{x_i} \quad \text{for } k = 1$$

$$\sum_{i=1}^{11} A \frac{1}{x_i} \frac{e^{-2x_i^2}}{x_i} + \sum_{i=1}^{11} B \frac{e^{-2x_i^2}}{x_i} \frac{e^{-2x_i^2}}{x_i} = \sum_{i=1}^{11} y_i \frac{e^{-2x_i^2}}{x_i} \quad \text{for } k = 2$$

# 6.8 Curve Fitting with a Linear Combination of Non-linear Functions

## ❖ Linear combination of non-linear functions

- Example 6-9

$$\sum_{i=1}^{11} A\frac{1}{x_i}\frac{1}{x_i} + \sum_{i=1}^{11} B\frac{e^{-2x_i^2}}{x_i}\frac{1}{x_i} = \sum_{i=1}^{11} y_i\frac{1}{x_i} \quad \text{for } k = 1$$

$$\sum_{i=1}^{11} A\frac{1}{x_i}\frac{e^{-2x_i^2}}{x_i} + \sum_{i=1}^{11} B\frac{e^{-2x_i^2}}{x_i}\frac{e^{-2x_i^2}}{x_i} = \sum_{i=1}^{11} y_i\frac{e^{-2x_i^2}}{x_i} \quad \text{for } k = 2$$

$$A\sum_{i=1}^{11}\frac{1}{x_i^2} + B\sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2} = \sum_{i=1}^{11} y_i\frac{1}{x_i}$$

$$A\sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2} + B\sum_{i=1}^{11}\frac{e^{-4x_i^2}}{x_i^2} = \sum_{i=1}^{11} y_i\frac{e^{-2x_i^2}}{x_i}$$

$$\begin{bmatrix} \sum_{i=1}^{11}\frac{1}{x_i^2} & \sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2} \\ \sum_{i=1}^{11}\frac{e^{-2x_i^2}}{x_i^2} & \sum_{i=1}^{11}\frac{e^{-4x_i^2}}{x_i^2} \end{bmatrix} \begin{bmatrix} A \\ B \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{11} y_i\frac{1}{x_i} \\ \sum_{i=1}^{11} y_i\frac{e^{-2x_i^2}}{x_i} \end{bmatrix}$$

# 6.8 Curve Fitting with a Linear Combination of Non-linear Functions

## Linear combination of non-linear functions

- Example 6-9

```
x = [0.6 0.8 0.85 0.95 1.0 1.1 1.2 1.3 1.45 1.6 1.8];
y = [0.08 0.06 0.07 0.07 0.07 0.06 0.06 0.06 0.05 0.05 0.04];
a(1,1) = sum(1./x.^2);
a(1,2) = sum(exp(-2*x.^2)./x.^2);
a(2,1) = a(1,2);
a(2,2) = sum(exp(-4*x.^2)./x.^2);
b(1,1) = sum(y./x);
b(2,1) = sum((y.*exp(-2*x.^2))./x);
AB = a\b
xfit = 0.6:0.02:1.8;
yfit = AB(1)./xfit + AB(2)*exp(-2*xfit.^2)./xfit;
plot(x,y,'o',xfit,yfit)
```