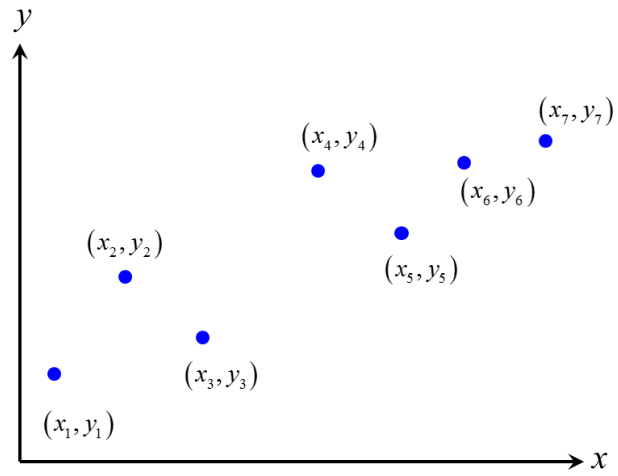
University of Washington
AMATH 301 Spring 2017
Instructor: Dr. King-Fai Li

Week 4

Consider the following given dataset:



x_{in}	y_{in}
1	6
2	9
3	7
4	12
5	11
6	13
7	15

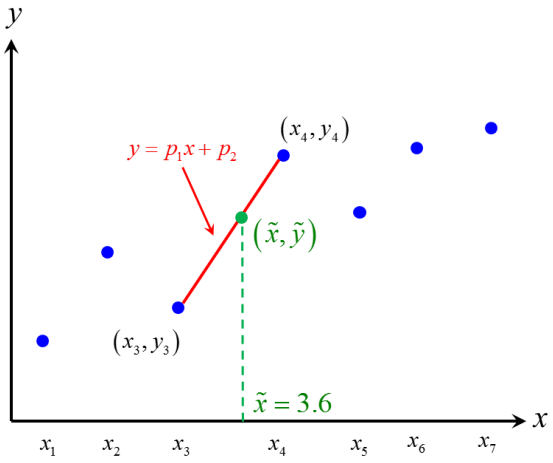
```
x_in=[1  2  3  4  5  6  7];
y_in=[6  9  7  12 11 13 15];
```

Interpolation: Estimate non-existing values based on existing information

Say we want to estimate the value of $\tilde{y}$ at $\tilde{x} = 3.6$.

Linear interpolation

Find two points (i.e. x_3 and x_4) embracing $\tilde{x}$. Draw a line between x_3 and x_4: $y = p_1 x + p_2$.



Solve for p_1 and p_2:

$$\begin{aligned} y_3 &= p_1 x_3 + p_2 \\ y_4 &= p_1 x_4 + p_2 \end{aligned} \rightarrow \begin{pmatrix} y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} x_3 & 1 \\ x_4 & 1 \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

$$\begin{pmatrix} p_1 \\ p_2 \end{pmatrix} = \begin{pmatrix} x_3 & 1 \\ x_4 & 1 \end{pmatrix} \backslash \begin{pmatrix} y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 3 & 1 \\ 4 & 1 \end{pmatrix} \backslash \begin{pmatrix} 7 \\ 12 \end{pmatrix} = \begin{pmatrix} 5 \\ -8 \end{pmatrix}$$

Then $\tilde{y} = p_1 \tilde{x} + p_2 = 5 \times 3.6 - 8 = 10$

In Matlab,

```
p = polyfit(x_in(3:4),y_in(3:4),1);
y_tilde = polyval(p,3.6);

>> p =  5.0000   -8.0000
>> y_tilde =  10.0000
```

A more convenient way using `interp1`:

```
y_tilde = interp1(x_in,y_in,3.6)
>> y_tilde =  10.0000
```

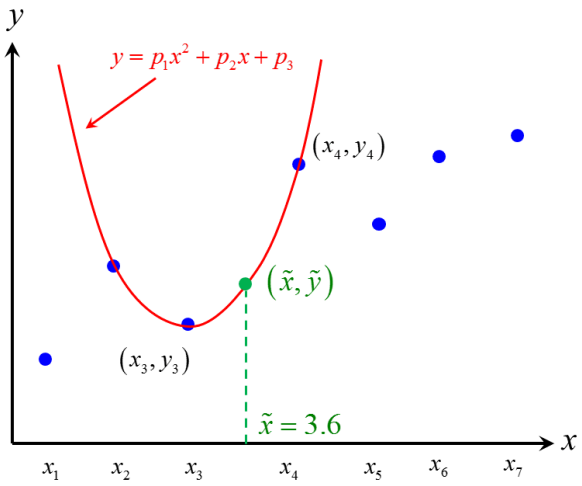
Quadratic interpolation

Find three points (either x_2, x_3 and x_4 or x_3, x_4 and x_5) embracing $\tilde{x}$. Draw a parabola connection these three points: $y = p_1x^2 + p_2x + p_3$.

Solve for p_1 and p_2 with x_2, x_3 and x_4:

$$y_2 = p_1x_2^2 + p_2x_2 + p_3$$
$$y_3 = p_1x_3^2 + p_2x_3 + p_3$$
$$y_4 = p_1x_4^2 + p_2x_4 + p_3$$

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} = \begin{pmatrix} x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{pmatrix} \backslash \begin{pmatrix} y_2 \\ y_3 \\ y_4 \end{pmatrix}$$
$$= \begin{pmatrix} 4 & 2 & 1 \\ 9 & 3 & 1 \\ 16 & 4 & 1 \end{pmatrix} \backslash \begin{pmatrix} 9 \\ 7 \\ 12 \end{pmatrix} = \begin{pmatrix} 3.5 \\ -19.5 \\ 34 \end{pmatrix}$$



Then $\tilde{y} = p_1\tilde{x}^2 + p_2\tilde{x} + p_3 = 9.16$.

In Matlab,

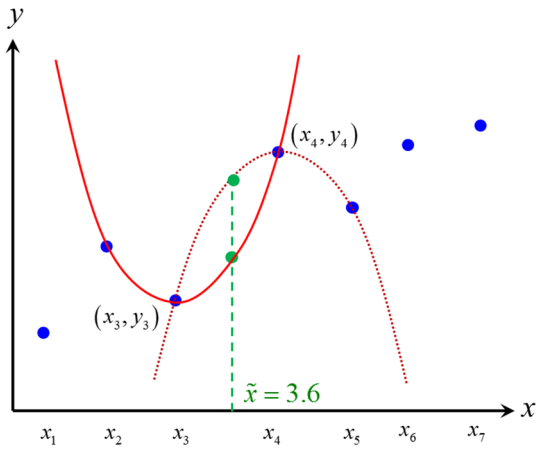
```
p = polyfit(x_in(2:4),y_in(2:4),2)
y_tilde = polyval(p,3.6)

>> p = 3.5000  -19.5000   34.0000
>> y_tilde = 9.1600
```

Note that if x_3, x_4 and x_5 are used, a different answer will be obtained.

```
p = polyfit(x_in(3:5),y_in(3:5),2)
y_tilde = polyval(p,3.6)
>> p = -3.0000   26.0000   -44.0000
>> y_tilde = 10.7200
```

Both guesses are okay because they are just estimates.



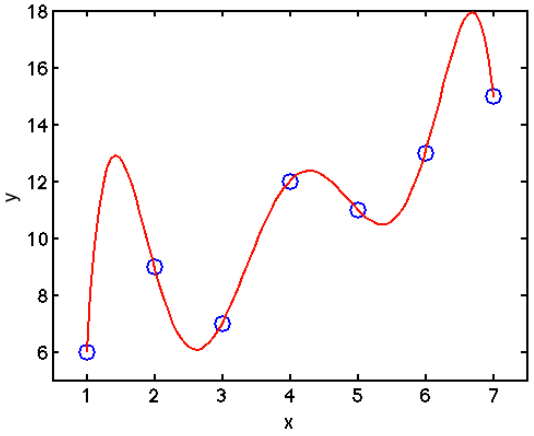
Lagrange interpolation
Use all n points, construct the unique $(n-1)$-th order polynomial passing through all points.

$$y = p_1 x^{n-1} + p_2 x^{n-2} + \cdots + p_{n-1} x + p_n$$

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \underbrace{\begin{pmatrix} x_1^{n-1} & x_1^{n-2} & \cdots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \cdots & x_2 & 1 \\ \vdots & \vdots & & & \vdots \\ x_n^{n-1} & x_n^{n-2} & \cdots & x_n & 1 \end{pmatrix}}_{\text{Vandermonde matrix } \mathbf{X}} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n+1} \end{pmatrix}$$

or $\mathbf{p} = \mathbf{X} \setminus \mathbf{y}$

```
p = polyfit(x_in,y_in,6);
xdraw = linspace(1,7,100);
ydraw = polyval(p,xdraw);
plot(x_in,y_in,'ob')
hold on
plot(xdraw,ydraw,'r')
hold off
```



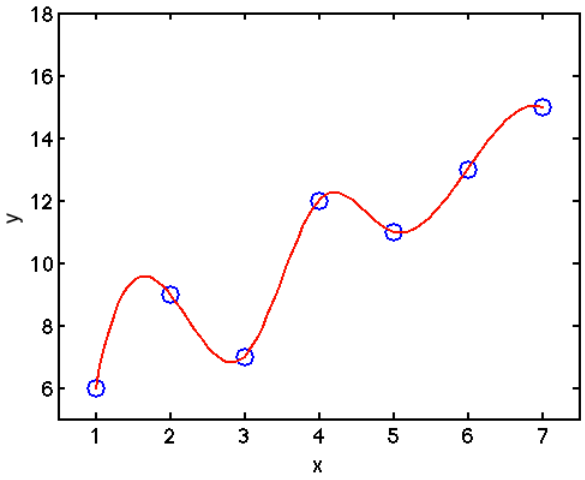
Note: `xdraw` is defined for visualization only.

Runge Phenomenon at the edge: Unrealistic values.

Solutions to the Runge Pheomenon:
See Question 4 of Homework 2.

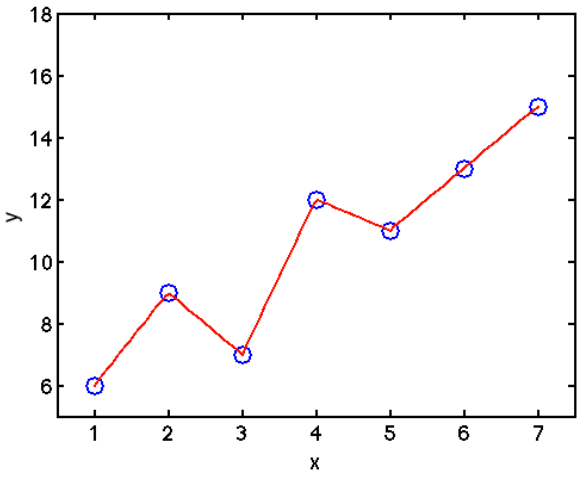
Spline interpolation

```
ydraw = spline(x_in,y_in,xdraw);
```



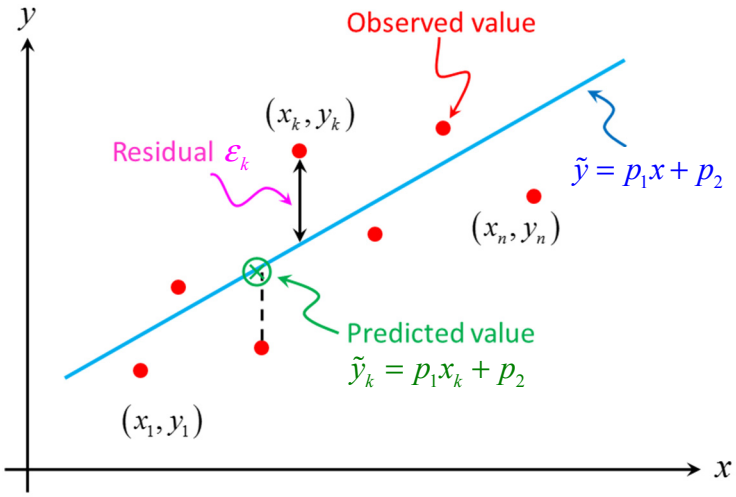
Piecewise linear interpolation

```
ydraw = interp1(x_in,y_in,xdraw);
```



Curve fitting

The spread of the data may be due to noise. Given n data points (x_1, y_1), (x_2, y_2), … , (x_n, y_n), fit the data with a straight line $\tilde{y} = p_1 x + p_2$. Since there are only two unknowns (p_1 and p_2) but n data points, the system is overdetermined and only approximate solutions for p_1 and p_2 can be found.



Input y_{in} = best-fit $\tilde{y}$ + residual

$$y_1 = p_1 x_1 + p_2 + \varepsilon_1$$
$$y_2 = p_1 x_2 + p_2 + \varepsilon_2$$
$$\vdots$$
$$y_n = p_1 x_n + p_2 + \varepsilon_n$$

or

$$\underbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}_{\mathbf{y}} = \underbrace{\begin{pmatrix} x_1 & 1 \\ x_2 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{pmatrix}}_{\mathbf{X}} \underbrace{\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}}_{\mathbf{p}} + \underbrace{\begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}}_{\boldsymbol{\varepsilon}}$$

Or in matrix form

$$\mathbf{y} = \mathbf{X}_{n\times 2}\mathbf{p} + \boldsymbol{\varepsilon}$$

$\mathbf{p} = \mathbf{X}^{-1}\mathbf{y}$ is not possible because $\mathbf{X}$ is not square.

Instead $\mathbf{p} = \underbrace{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}_{2\times 2}\mathbf{X}^T\mathbf{y}$, called the least squares solution.

```
p = polyfit(x_in,y_in,1)
y_tilde = polyval(p,x_tilde)  % for any x_tilde
```

Physical Meaning: Minimize $\sum_{k=1}^{n} \varepsilon_k^2 = \sum_{k=1}^{n} \left| y_k - p_1 x_k - p_2 \right|^2$

Higher order fit

Best fit $(m-1)$-th order polynomial $\tilde{y} = p_1 x^{m-1} + p_2 x^{m-2} + \cdots + p_{m-1} x + p_m$ for $m < n$.

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \underbrace{\begin{pmatrix} x_1^{m-1} & x_1^{m-2} & \cdots & x_1 & 1 \\ x_2^{m-1} & x_2^{m-2} & \cdots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^{m-1} & x_n^{m-2} & \cdots & x_n & 1 \end{pmatrix}}_{\mathbf{X}_{n\times m}} \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_m \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

$$\mathbf{p} = \underbrace{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}_{m\times m}\mathbf{X}^T\mathbf{y} \qquad \text{The same solution}$$

If $m = n$, this is just a polynomial interpolation with an $(n-1)$-th order polynomial and $\boldsymbol{\varepsilon} = 0$.

Matlab's Magic

$$\mathbf{p} = \mathbf{X}_{n\times m} \backslash \mathbf{y}$$

If $m<n$, $\mathbf{X}\backslash\mathbf{Y}$ is the least squares solution $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$.

$\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T$ is the pseudo-inverse of $\mathbf{X}$.

If $m=n$, $\mathbf{X}\backslash\mathbf{Y}$ is the Gaussian Elimination $\mathbf{X}^{-1}\mathbf{y}$.

In Matlab,

```
p = [x_in.^(m-1)  x_in.^(m-2) … x_in  ones(6,1)]\y_in
```

or simply

```
p = polyfit(x_in,y_in,m)
```

For fitting a straight line

```
p = polyfit(x_in,y_in,1)
```

To get the predicted value of the fitting curve:

```
y_tilde = polyval(p,x_tilde)
```

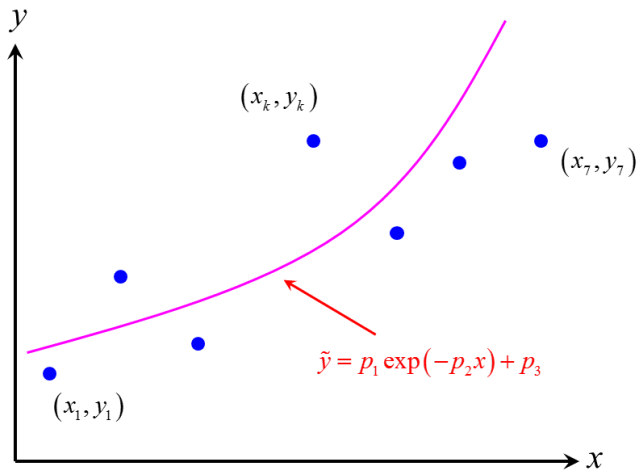
Non-Linear regression

If $\tilde{y} = f(\mathbf{p}, \mathbf{x}_{in})$ is not separable in $\mathbf{p}$ and $\mathbf{x}_{in}$, then we have to minimize the residual manually.

Algorithm:

1. Define the sum-of-square error function $Error(\mathbf{p}) = \sum_{k=1}^{n} \varepsilon_k^2 = \sum_{k=1}^{n} |y_k - f(\mathbf{p}, x_k)|^2$
2. Use `fminsearch` in Matlab to find $\mathbf{p}$.
3. Or `lsqcurvefit` in Matlab to find $\tilde{y}$ directly.

Example:




```
f = @(p,x) p(1)*exp(-p(2)*x) + p(3);
err = @(p) sum((y_in - f(p,x_in)).^2);
[p_fit,err_min]=fminsearch(err,[1 1 0])

>> p_fit = 5.9286 -0.1331 -0.0259
>> err_min = 9.5077
```

Or

```
p_fit=lsqcurvefit(f,[1 1 0],x_in,y_in);
```

Minimize $\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon} = (\mathbf{Y}-\mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta}) \quad \Rightarrow \quad \dfrac{d\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}{d\boldsymbol{\beta}} = 0$

$$\frac{d\boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}}{d\boldsymbol{\beta}} = -2\mathbf{X}^T(\mathbf{Y}-\mathbf{X}\boldsymbol{\beta})$$

Note: $\dfrac{d(\mathbf{A}\boldsymbol{\beta})}{d\boldsymbol{\beta}^T} = \mathbf{A}, \quad \dfrac{d(\boldsymbol{\beta}^T\mathbf{A})}{d\boldsymbol{\beta}} = \mathbf{A}, \quad \dfrac{d(\mathbf{A}\boldsymbol{\beta}^T)}{d\boldsymbol{\beta}} = \mathbf{A}^T$

$$\mathbf{X}^T\mathbf{Y} - (\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} = 0$$

$$\mathbf{X}^T\mathbf{X}\boldsymbol{\beta} = \mathbf{X}^T\mathbf{Y}$$

Normal equation

$$(\mathbf{X}^T\mathbf{X})^{-1}(\mathbf{X}^T\mathbf{X})\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

$$\boldsymbol{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$