Introduction to Computer for Engineers

Lecture 12
Curve Fitting

Dr. Vo Tan Phuoc
School of Electrical Engineer – International University

Review Math

Polynomial

1st order : $y = a_1 x + a_0$

2nd order : $y = a_2 x^2 + a_1 x + a_0$

nth order:

$$y = a_n x^n + a_{n-2} x^{n-1} + \ldots + a_1 x + a_0$$

Review Math

Polynomial

1st order : $y = a_1 x + a_0$

2nd order : $y = a_2 x^2 + a_1 x + a_0$

nth order:

$$y = a_n x^n + a_{n-2} x^{n-1} + \ldots + a_1 x + a_0$$

→ N-th polynomial has n+1 coefficients

Review Math

Find the coef. of the first order polynomial given 2 points A and B

$$y = a_1 x + a_0 \quad \begin{cases} A(x_A, y_A) \\ B(x_B, y_B) \end{cases}$$

2 equations with two unknowns a_0, a_1

→ formulate this problem in the matrix form

→ apply Gaussian Elimination to obtain a_0, a_1

$$\begin{cases} y_A = a_1 x_A + a_0 \\ y_B = a_1 x_B + a_0 \end{cases} \quad \rightarrow \quad \begin{bmatrix} x_A & 1 \\ x_B & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \end{bmatrix}$$

Review Math

How many points are required for finding coefs of 2nd order polynomial ?

$$y = a_2 x^2 + a_1 x + a_0$$

Review Math

How many points are required for finding coefs. of 2nd order polynomial ?

$$y \quad = \quad a_2 x^2 \quad + \quad a_1 x \quad + \quad a_0$$

For the case of 2nd order polynomial, we need 3 points

$$\begin{cases} y_A = a_2 x_A^2 + a_1 x_A + a_0 \\ y_B = a_2 x_B^2 + a_1 x_B + a_0 \\ y_C = a_2 x_C^2 + a_1 x_C + a_0 \end{cases} \quad \rightarrow \quad \begin{bmatrix} x_A^2 & x_A & 1 \\ x_B^2 & x_B & 1 \\ x_C^2 & x_C & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \\ y_C \end{bmatrix}$$

Review Math

Linear form

1st order :

$$\begin{cases} y_A = a_1 x_A + a_0 \\ y_B = a_1 x_B + a_0 \end{cases} \quad \rightarrow \quad \begin{bmatrix} x_A & 1 \\ x_B & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \end{bmatrix} \quad \rightarrow \quad AX = B$$

Review Math

Linear relation

1st order :

$$\begin{cases} y_A = a_1 x_A + a_0 \\ y_B = a_1 x_B + a_0 \end{cases} \rightarrow \begin{bmatrix} x_A & 1 \\ x_B & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \end{bmatrix} \rightarrow AX = B$$

2nd order – we still have linear form

$$\begin{cases} y_A = a_2 x_A^2 + a_1 x_A + a_0 \\ y_B = a_2 x_B^2 + a_1 x_B + a_0 \\ y_C = a_2 x_C^2 + a_1 x_C + a_0 \end{cases} \rightarrow \begin{bmatrix} x_A^2 & x_A & 1 \\ x_B^2 & x_B & 1 \\ x_C^2 & x_C & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \\ y_C \end{bmatrix} \rightarrow AX = B$$

Review Math

Linear relation

1st order :

$$\begin{cases} y_A = a_1 x_A + a_0 \\ y_B = a_1 x_B + a_0 \end{cases} \rightarrow \begin{bmatrix} x_A & 1 \\ x_B & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \end{bmatrix} \rightarrow AX = B$$

2nd order – we still have linear form

$$\begin{cases} y_A = a_2 x_A^2 + a_1 x_A + a_0 \\ y_B = a_2 x_B^2 + a_1 x_B + a_0 \\ y_C = a_2 x_C^2 + a_1 x_C + a_0 \end{cases} \rightarrow \begin{bmatrix} x_A^2 & x_A & 1 \\ x_B^2 & x_B & 1 \\ x_C^2 & x_C & 1 \end{bmatrix} \begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} = \begin{bmatrix} y_A \\ y_B \\ y_C \end{bmatrix} \rightarrow AX = B$$

Coefs are recovered exactly with Gaussian Elimination

→ Algebraic Curve Fitting

Linear Least Square Curve Fitting

In practice (lab experiment, hardware project), measurement error always exist. Thus

$$y = a_1 x + a_0 \quad \begin{cases} A(x_A, y_A) \\ B(x_B, y_B) \end{cases}$$

becomes

$$\begin{cases} A(x_A + \Delta x_A, y_A + \Delta y_A) \\ B(x_B + \Delta x_B, y_B + \Delta y_B) \end{cases} \quad \rightarrow \quad y = a_1 x + a_0 + \Delta$$

A and B with 2 measurements (x_A,y_A),$(x_B,y_B)$are called, in general, measurement data accompanying with their error quantities Δx_A, Δx_B, Δy_A, Δy_B → "noisy", "discrete-time "

Linear Least Square Curve Fitting

In practice (lab experiment, hardware project), measurement error always exist. Thus

$$y = a_1 x + a_0 \quad \begin{cases} A(x_A, y_A) \\ B(x_B, y_B) \end{cases}$$

Become

$$\begin{cases} A(x_A + \Delta x_A, y_A + \Delta y_A) \\ B(x_B + \Delta x_B, y_B + \Delta y_B) \end{cases} \quad \rightarrow \quad y = a_1 x + a_0 + \Delta$$

With 2 measurements (x_A, y_A),(x_B, y_B), we can not determine the coef. of 1st order polynomial.

Linear Least Square Curve Fitting

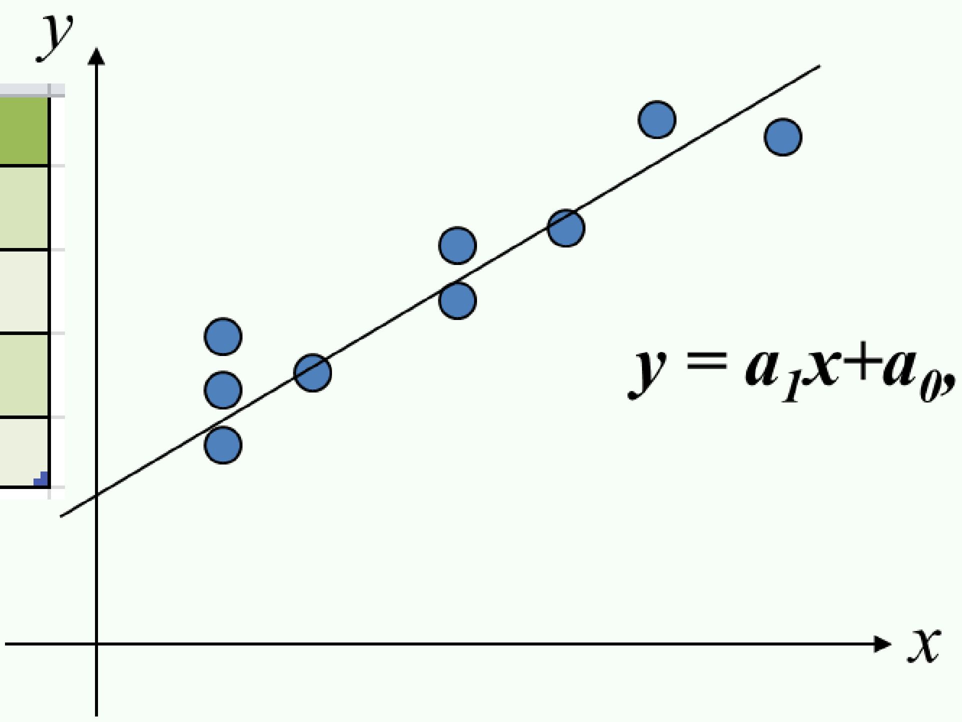
How many measurements are required for determining the coef ?

$$\begin{cases} A(x_A + \Delta x_A, y_A + \Delta y_A) \\ B(x_B + \Delta x_B, y_B + \Delta y_B) \\ \\ \quad ? \end{cases} \quad \rightarrow \quad y = a_1 x + a_0 + \Delta$$

Linear Least Square Curve Fitting

Measurements are taken as many as possible together with a "MODEL" of 1st order polynomial

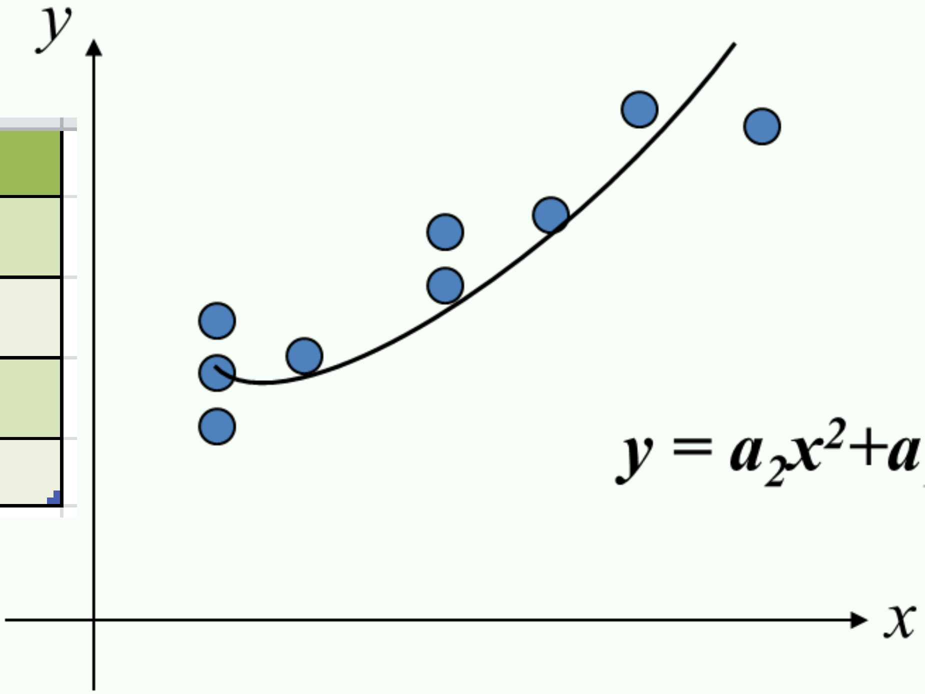
#Measurement	x	y
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
...	...	...



Linear Least Square Curve Fitting

Measurements are taken as many as possible together with a "MODEL" of 2nd order polynomial

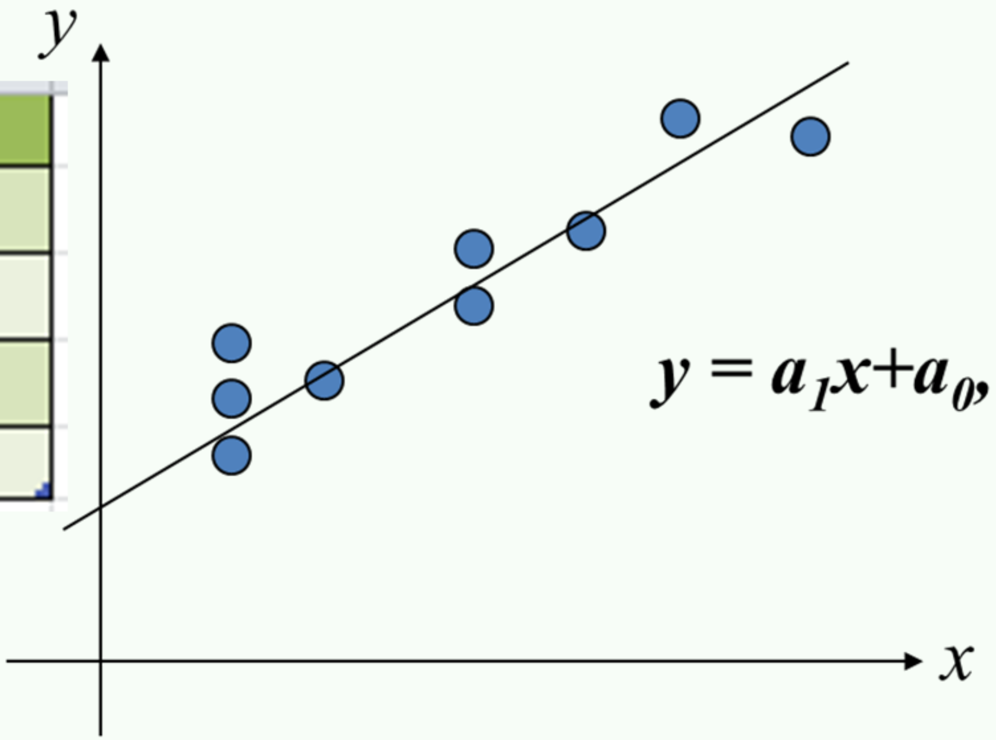
#Measurement	x	y
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
…	…	…



$$y = a_2x^2+a_1x+a_0$$

Linear Least Square Curve Fitting

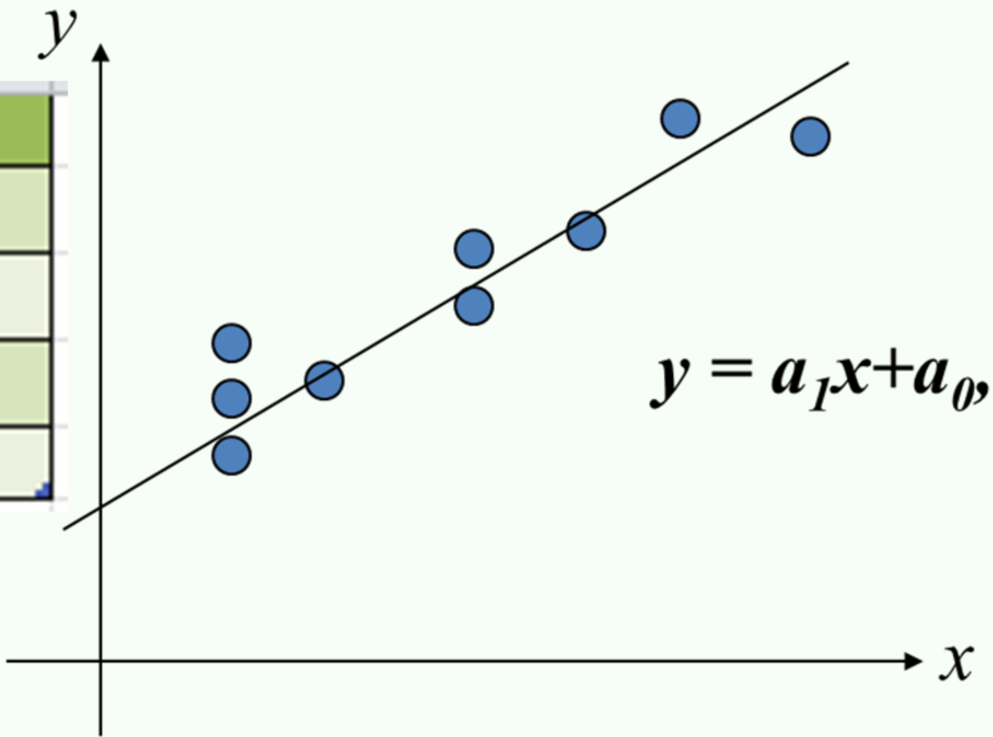
#Measurement	x	y
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
...	...	...



Data points (x_i, y_i) **can be approximated by a function** $y = f(x)$ **such that the function passes "*close*" to the data points but does not necessarily pass through them.**

Linear Least Square Curve Fitting

#Measurement	x	y
1	x_1	y_1
2	x_2	y_2
3	x_3	y_3
…	…	…



Data fitting is necessary to model data with fluctuations such as experimental measurements.

$$\begin{cases} A(x_A + \Delta x_A, y_A + \Delta y_A) \\ B(x_B + \Delta x_B, y_B + \Delta y_B) \end{cases} \rightarrow \quad y = a_1 x + a_0 + \Delta$$

Linear Least Square Curve Fitting

Principle

- **Data points** $(x_i, y_i) \quad i = 1 \ldots n$
- **Choice of fitting function (*model*)** $y = f(x) = a_1 x + a_0$
- **Errors between function and data points**

$$e_i = y_i - (a_1 x_i + a_0)$$

- **Sum of the squares of the errors**

$$z = e_1^2 + e_2^2 + \ldots + e_n^2$$

- **In compact notation**

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

Linear Least Square Curve Fitting

- Our goal is to determine the values of a_1 and a_0 that will minimize z, the sum of the squares of the errors.

To find the minimum value for z, Matlab uses the same technique that we would use analytically (i.e., setting the *derivative* of z to zero and solving a_1 and a_0)

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

Linear Least Square Curve Fitting

Find the minimum value for *z* – case 1st order polynomial

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

$$\frac{\delta z}{\delta a_1} = -2 \sum_{i=1}^{n} x_i \left(y_i - a_1 x_i - a_0 \right) = 0$$

$$\frac{\delta z}{\delta a_0} = -2 \sum_{i=1}^{n} \left(y_i - a_1 x_i - a_0 \right) = 0$$

Linear Least Square Curve Fitting

Find the minimum value for *z* – case 1st order polynomial

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

$$\rightarrow \quad \begin{aligned} a_1 \sum_{i=1}^{n} x_i^2 + a_0 \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i \\ a_1 \sum_{i=1}^{n} x_i + n a_0 &= \sum_{i=1}^{n} y_i \end{aligned}$$

Linear Least Square Curve Fitting

Find the minimum value for *z* – case 1st order polynomial

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

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^{n} x_i \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \end{bmatrix}$$

Linear Least Square Curve Fitting

Find the minimum value for *z* – case 2nd order polynomial

$$z = \sum_{i=1}^{n} \left[y_i - \left(a_2 x_i^2 + a_1 x_i + a_0 \right) \right]^2$$

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 \\ \sum_{i=1}^n x_i^2 & \sum_{i=1}^n x_i^3 & \sum_{i=1}^n x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \\ \sum_{i=1}^n x_i^2 y_i \end{bmatrix}$$

Linear Least Square Curve Fitting

Find the minimum value for *z* – case 2nd order polynomial

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

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix}$$

Case nth order polynomial ?

Linear Least Square Curve Fitting

Roadmap to linear least square

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix}$$

Input

→ Data point (measurements) (x_i,y_i) i=1,N

→ Fitting function (model – polynomial of order n)

Output

→ Polynomial coefficient columns vector of dim. (n+1) by 1

Linear Least Square Curve Fitting

Roadmap to linear least square – Case 2nd order polynomial

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix}$$

Input

→ **Data point (measurements) (x_i,y_i) i=1,N**

→ **Fitting function (n=2)**

→ **7 sum to be computed, Gaussian Elimination (n+1) x (n+2)**

Output

→ **Polynomial coefficient columns vector of dim. (2+1) by 1**

Linear Least Square Curve Fitting

Roadmap to linear least square – Case 2[nd] order polynomial

$$\rightarrow \begin{bmatrix} n & \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 \\ \sum_{i=1}^{n} x_i & \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 \\ \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i^3 & \sum_{i=1}^{n} x_i^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i^2 y_i \end{bmatrix}$$

Input

→ Data point (measurements) (x_i, y_i) i=1,N

→ Fitting function (n=2)

→ 7 sum to be computed, Gaussian Elimination (n+1) x (n+2)

Could you estimate the complexity in the case n=2 ?

Linear Least Square Curve Fitting

Implementation

- **Your own script using road map OR**
- **MATLAB bulit-in function polyfit**

Linear Least Square Curve Fitting

Implementation – using polyfit

Polyfit is a built in Matlab function for fitting data to a n^th degree polynomial

Assume a polynomial of the following form:

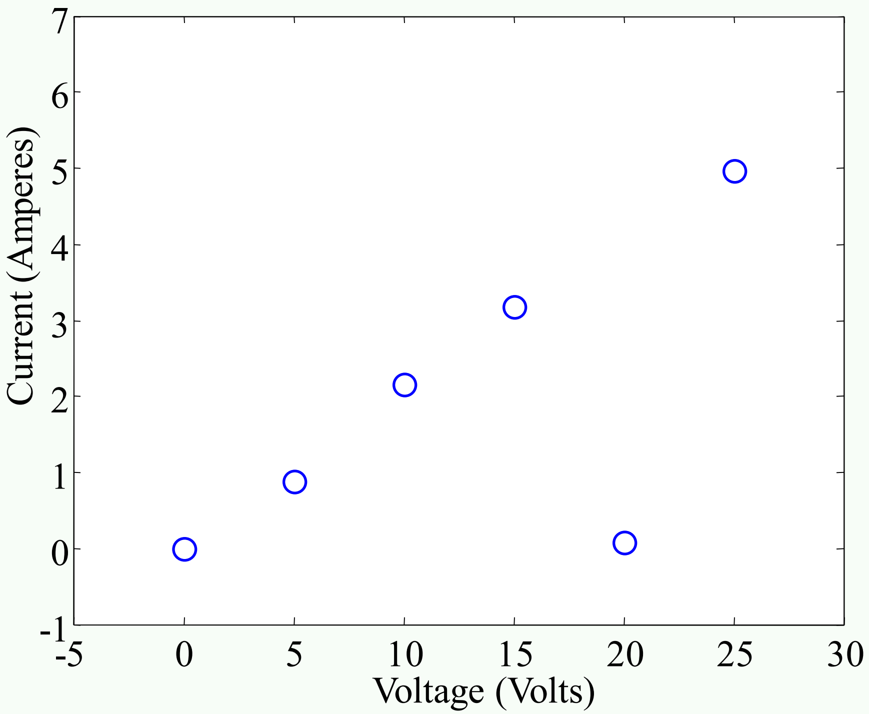
$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_1 \cdot x + a_0$$

Assume we have also obtained a series of data (x_i, y_i) $i=1,N$.

The command a=polyfit(x,y,n) would find the coefficients of the polynomial above $a=[a_0, a_1, \ldots, a_{n-1}, a_n]$ that would best fit the measured data in the "least squares" sense.

Linear Least Square Curve Fitting

Example – Ohm Law



- This graph shows the amount of current through an unknown resistor, plotted against the voltage applied to the resistor.
- Knowing Ohm's Law (*I = V/R*), find the resistance, and plot an approximating function.
- Ohm's Law shows a linear dependence between current and voltage, so we will use a first degree fit.

Linear Least Square Curve Fitting

- Rewrite $I = V/R$ as $I = (1/R)*V$, which is a linear relationship between V and I, with coefficient $1/R$.

$$y = f(x) = ax + b; I = \frac{1}{R}V + 0$$

```
xdata = [0 5 10 15 20 25];
ydata = [0.001 0.881 2.1637 3.1827 0 4.961];
degree = 1;                    % Linear relationship
coef = polyfit(xdata, ydata, degree);
coef
coef = [0.1324    0.2095] % This is a and b
resistance = 1 / coef(1) % Output the answer
```

Linear Least Square Curve Fitting

Polyval is a built in Matlab function for evaluating a polynomial curve fit that was calculated using polyfit (i.e. first you use polyfit then you check how well it worked using polyval)

Assume a polynomial of the following form:

$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_1 \cdot x + a_0$$

and that we have determined the coefficient vector p using the command a=polyfit(x,y,n)

Linear Least Square Curve Fitting

Assume a polynomial of the following form:

$$y = a_n \cdot x^n + a_{n-1} \cdot x^{n-1} + a_{n-2} \cdot x^{n-2} + \cdots + a_1 \cdot x + a_0$$

and that we have determined the coefficient vector p using the command a=polyfit(x,y,n)

We can now determine the value of y for any value of x using the polynomial found by polyfit using the command polyval

ynew=polyval(a,xnew)

Linear Least Square Curve Fitting

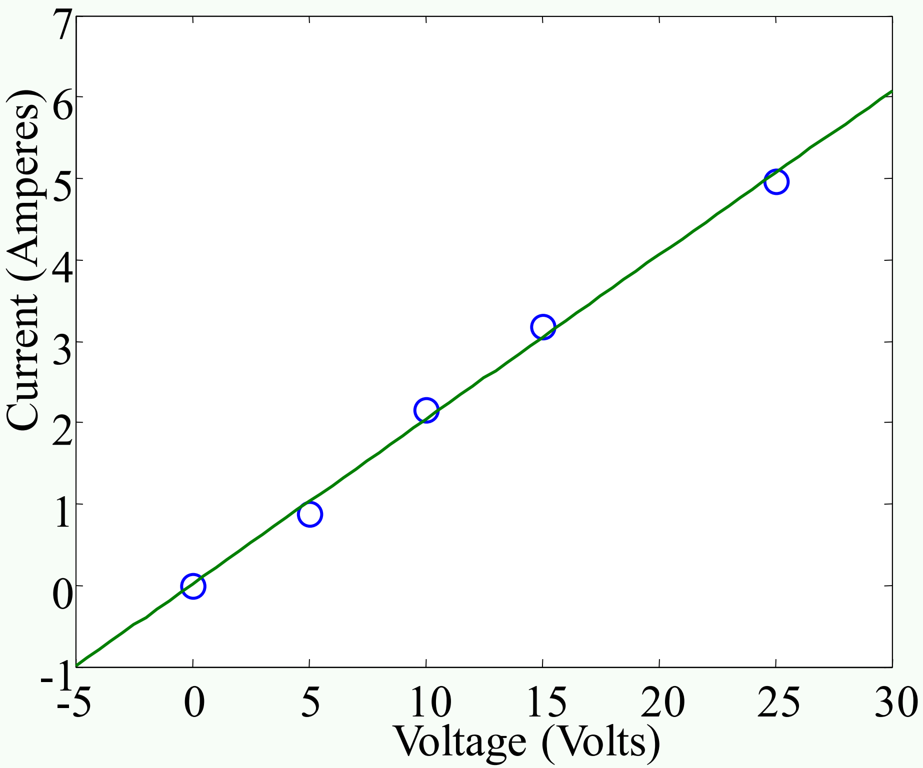
In MATLAB:

```
xdata = [0 5 10 15 20 25];
ydata = [0.001 0.881 2.1637 3.1827 0 4.961];
degree = 1;                  % Linear relationship
coef = polyfit(xdata, ydata, degree);
xx = [-5 : 0.5 : 30];        % Range for plotting
yy = polyval(coef, xx);
plot(xdata, ydata, 'o', xx, yy);
resistance = 1 / coef(1) % Output the answer
```

Linear Least Square Curve Fitting

Plotting & Result

```
plot(xdata, ydata, 'o', xx, yy);
resistance = 1 / coef(1) % Output the answer
```



- **The output from Matlab is**

  resistance =

  4.9515

- **Thus, the resistance is 4.95 Ω.**

Linear Least Square Curve Fitting

Choosing the Right Polynomial

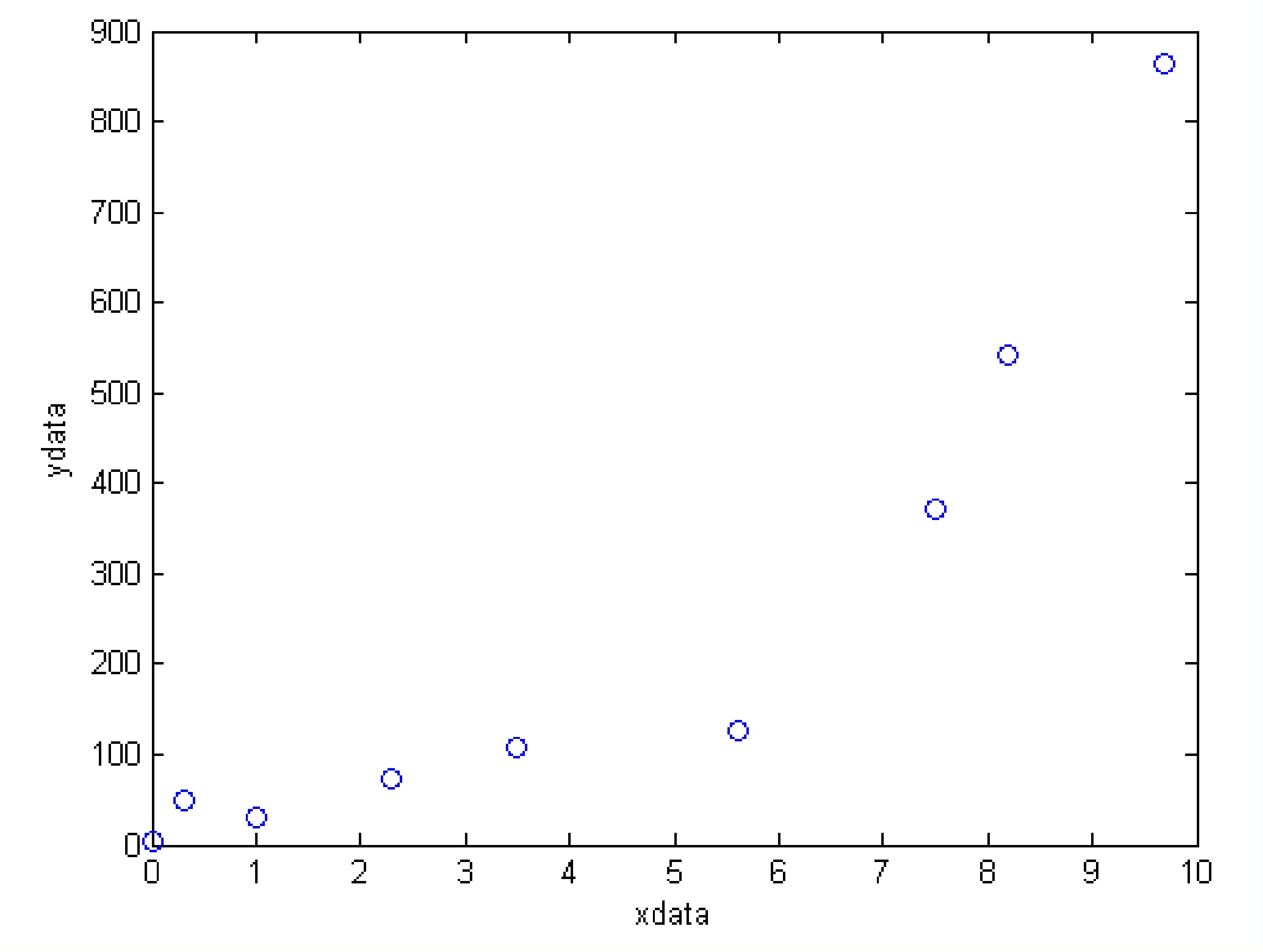
The degree of the correct approximating function depends on the type of data being analyzed.

When a certain behavior is expected, we know what type of function to use, and simply have to solve for its coefficients.

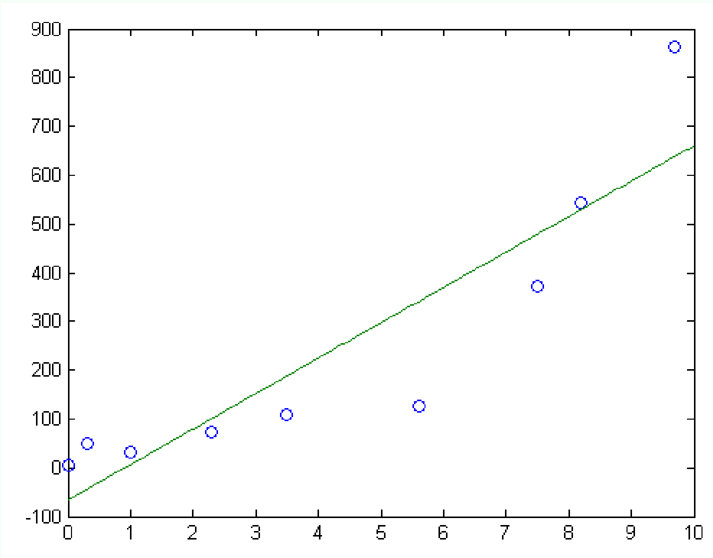
When we don’t know what sort of response to expect, ensure your data sample size is large enough to clearly distinguish which degree is the best fit.

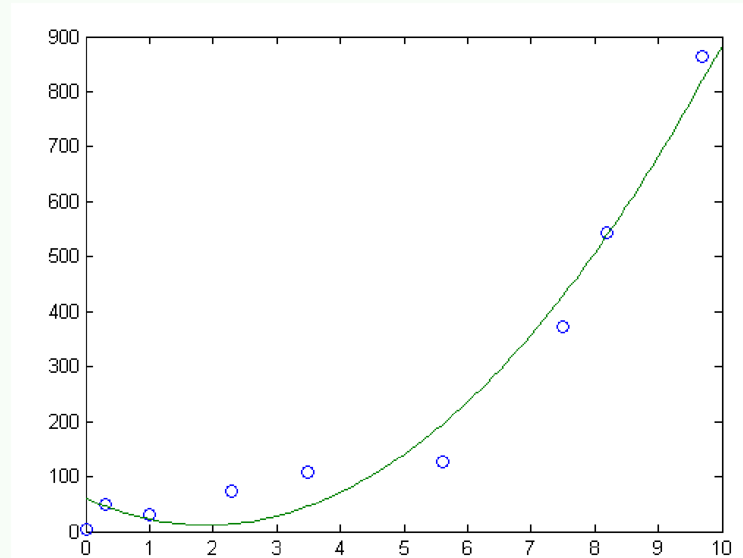
Linear Least Square Curve Fitting

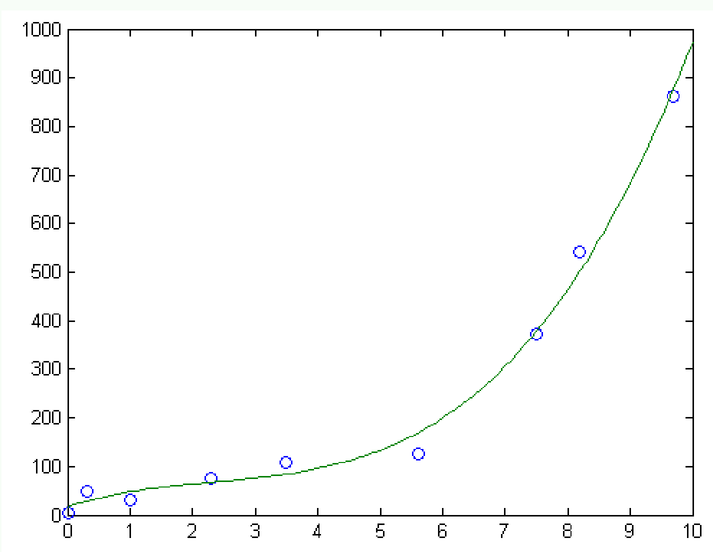
Choosing the Right Polynomial:Example

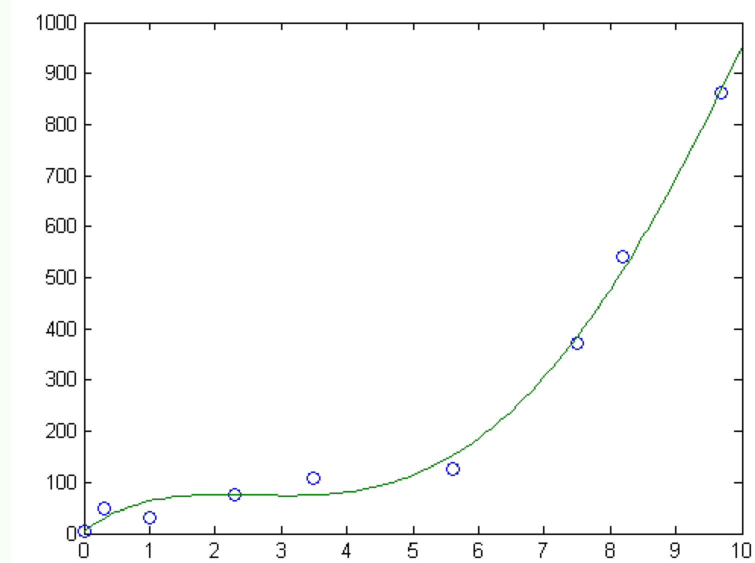


Linear Least Square Curve Fitting









Linear Least Square Curve Fitting

- ***To determine the "best" fit quantitatively we need to calculate the best sum of squares between the original data and the polynomial fit.***

```
xdata = [0 0.3000 1.0000 2.3000 3.5000 5.6000 7.5000 8.2000 9.7000];
ydata = [4.9838   49.0727   31.0883   74.4117  107.8540  127.0157
371.8902  542.5732  863.1195];
for degree=1:6
coef{degree} = polyfit(xdata, ydata, degree);
xx = 0 : 0.1 : 10;      % Range for plotting
yy = polyval(coef{degree}, xx);
figure(degree)
plot(xdata, ydata, 'o', xx, yy);
drawnow;
yfit=polyval(coef{degree},xdata);
Error_fit(degree)=sum((ydata-yfit).^2);
end
```

Linear Least Square Curve Fitting

```
xdata = [0 0.3000 1.0000 2.3000 3.5000 5.6000 7.5000 8.2000 9.7000];
ydata = [4.9838   49.0727   31.0883   74.4117  107.8540  127.0157
371.8902  542.5732  863.1195];
for degree=1:6
coef{degree} = polyfit(xdata, ydata, degree);
xx = 0 : 0.1 : 10;       % Range for plotting
yy = polyval(coef{degree}, xx);
figure(degree)
plot(xdata, ydata, 'o', xx, yy);
drawnow;
yfit=polyval(coef{degree},xdata);
Error_fit(degree)=sum((ydata-yfit).^2);
end
```

$$z = \sum_{i=1}^{n} \left[y_i - (ax_i + b) \right]^2 = \sum_{i=1}^{n} \left[y_{data} - y_{calculated} \right]^2$$

Error_fit =1.0e+005* [1.292 0.20095 0.05123 0.0400 0.01543 0.00800]

Linear Least Square Curve Fitting

- **Not all experimental data can be approximated with polynomial functions.**
- **Exponential data can be fit using the least squares method by first converting the data to a linear form.**

Linear Least Square Curve Fitting

Relationship between an Exponential & Linear Form

- An exponential function,

$$y = ae^{bx}$$

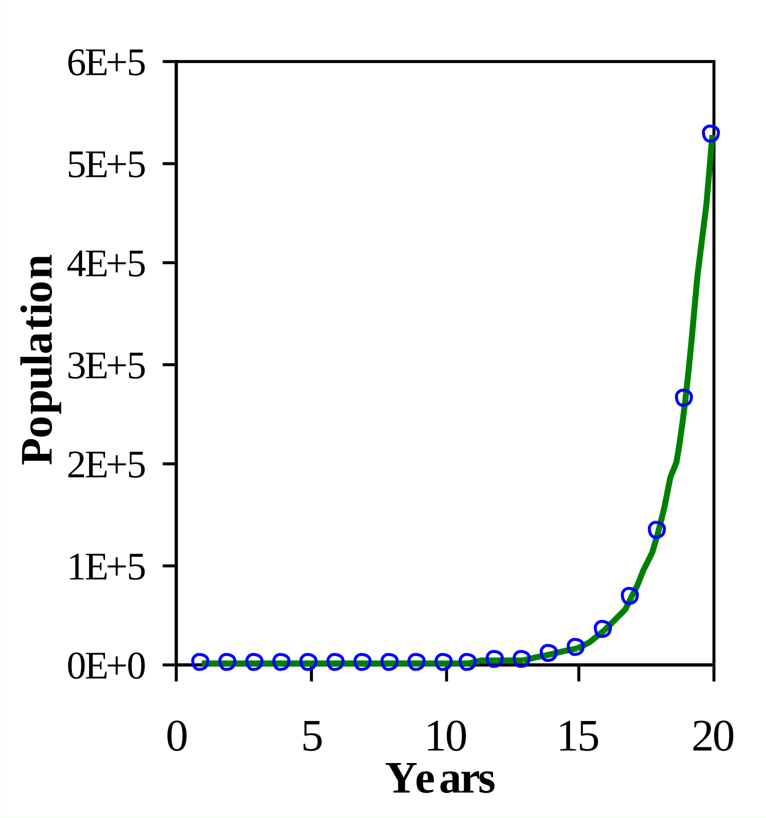
can be rewritten as a linear polynomial by taking the natural logarithm of each side:

$$\ln y = \ln a + bx$$

- By finding $\ln y_i$ for each point in a data set, we can solve for a and b using the least squares method.

Linear Least Square Curve Fitting

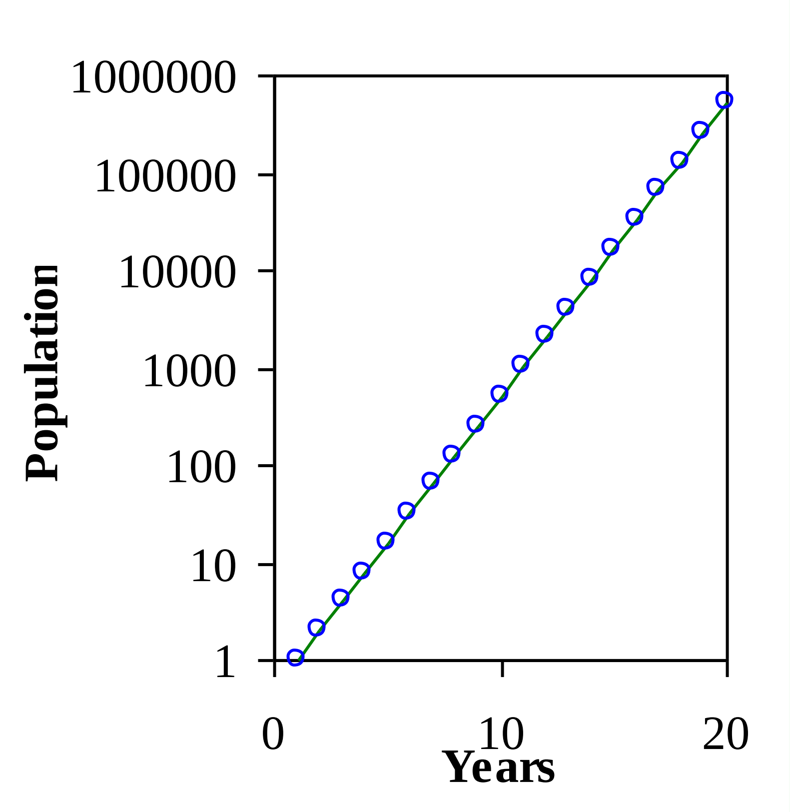
Relationship between an Exponential & Linear Form



This graph shows a population that doubles every year. Notice that for the first fifteen years, growth is almost imperceptible at this scale. *It would be difficult to approximate this raw data.*

Linear Least Square Curve Fitting

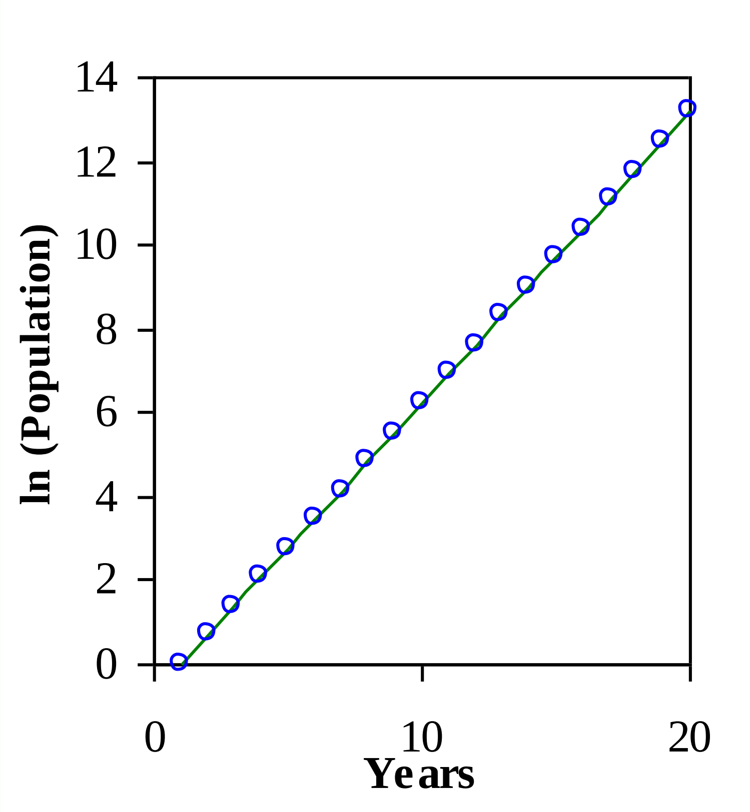
Relationship between an Exponential & Linear Form



- What if we plot the same data with the *y*-axis against a logarithmic scale?
- Notice that the change in population is not simply perceptible, but is clearly linear.
- If we can transform the numeric exponential data, it would be simple to approximate with the least-squares method.

Linear Least Square Curve Fitting

Relationship between an Exponential & Linear Form



- This graph shows what happens if we take the natural logarithm of each *y* value.

- Notice that the shape of the graph did not change from the last example, only the scaling of the *y*-axis.

Nonlinear Least Square Curve Fitting

Relationship between an Exponential & Linear Form

- **Case exponential function of the form**

$$y = a_1e^{b1x+}a_1e^{b2x} + \ldots + a_ne^{bnx}$$

$a_1,a_2,..,a_n$ still have linear relation wrt. the value x,y
$b_1,b_2, \ldots,b_n$ are non-linear wrt the value x,y

Matlab optimization built-in function such as fmincon is used

End of Lecture 12