Introduction to Computing for Engineers 050IU

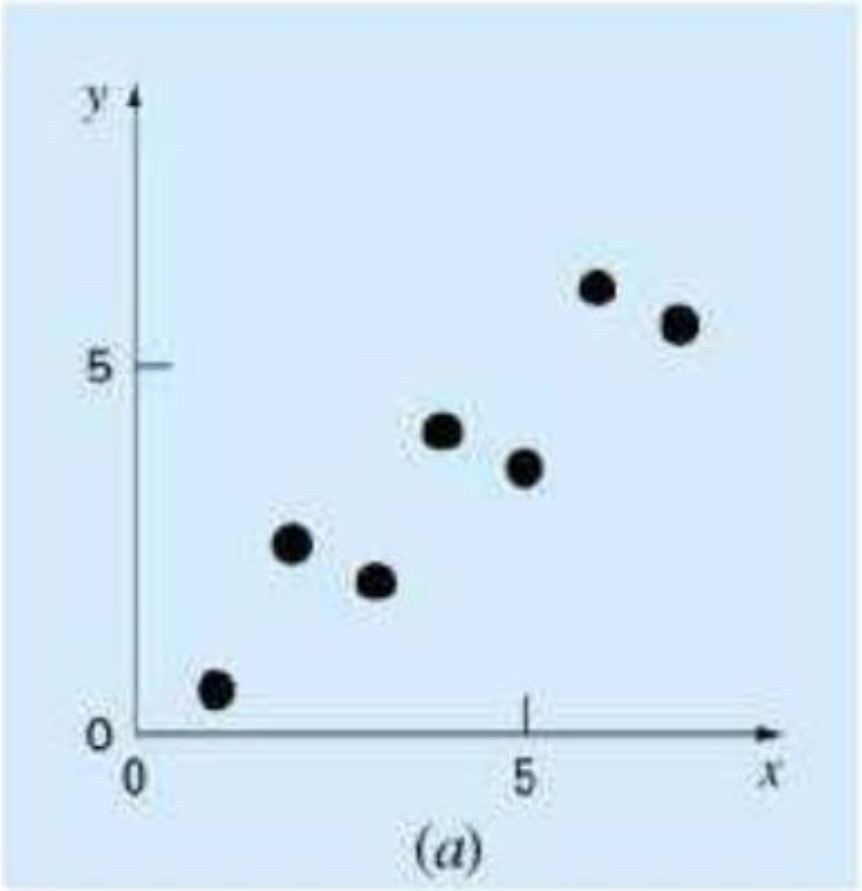
Curve Fitting

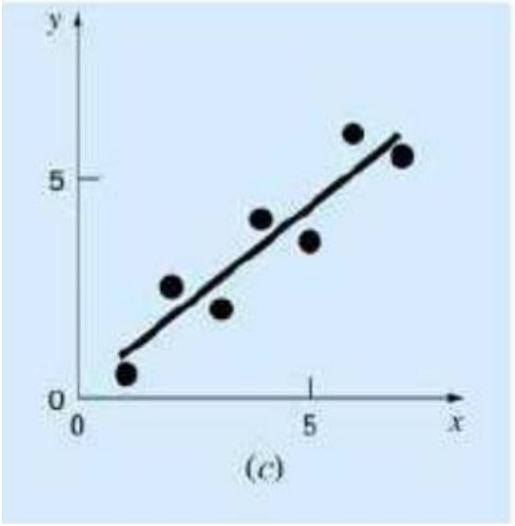
Dr. Nguyen Ngoc Truong Minh

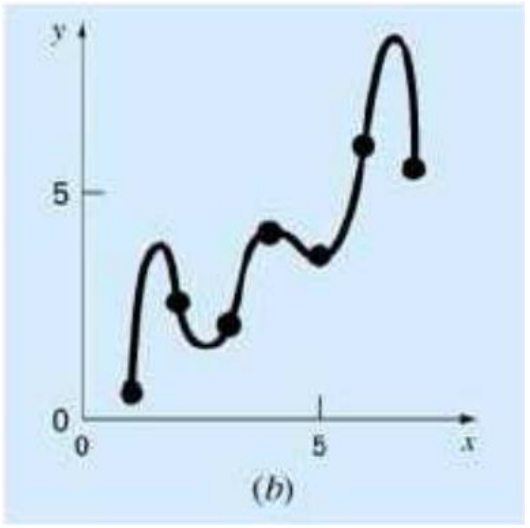
Fundamental to Engineering Practice and Research

- **Data analysis** is a process of inspecting, cleansing, transforming and modeling data with the goal of discovering useful information, informing conclusions and supporting decision-making

Curve fitting



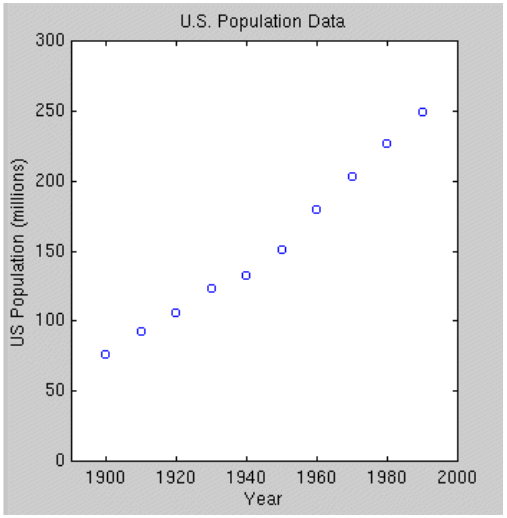


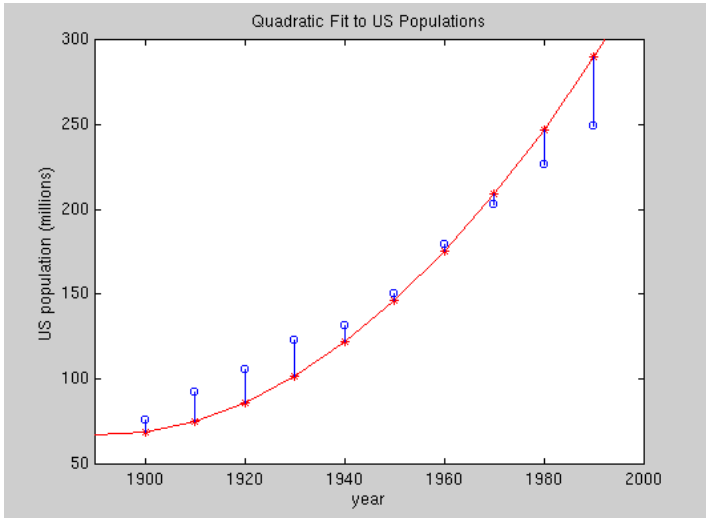


Fitting Data to a Curve
Prediction of Future Values

U.S. Populations
[Census data]

Year	Population (in millions)
1900	75.996
1910	91.972
1920	105.711
1930	122.775
1940	131.669
1950	150.697
1960	179.323
1970	203.185
1980	226.546
1990	248.710





We can predict future US populations by fitting historical data to a curve.

Perhaps a 2nd order polynomial (quadratic) would work.

$$Population = a \cdot t^2 + b \cdot t + c$$

Goal is to find a, b and c so that the curve best fits the data.

Polynomial

A polynomial is a function of a single variable that can be express in the general form

$$\mathrm{P}(s) = p_1 s^N + p_2 s^{N-1} + p_3 s^{N-2} + \cdots + p_N s + p_{N+1}$$

The polynomial is of degree.

Degree 3: P(s)= $s^3 + 4s^2 - 7s - 10$.

Linear Least Squares Algorithm

- Data points $(x_i, y_i) \quad i = 1 \ldots n$
- Choice of fitting function (*linear*) $y = f(x) = ax + b$
- Errors between function and data points $e_i = y_i - (ax_i + b)$
- Sum of the squares of the errors $z = e_1^2 + e_2^2 + \ldots + e_n^2$
- In compact notation $z = \sum_{i=1}^{n} [y_i - (ax_i + b)]^2$

Linear Least Squares Algorithm-cont.

- Our goal is to determine the values of *a* and *b* 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 for *a* and *b*)

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

Curve Fitting with Matlab's 'polyfit'

Polyfit is a built in MATLAB function for fitting data to a n^{th} degree polynomial

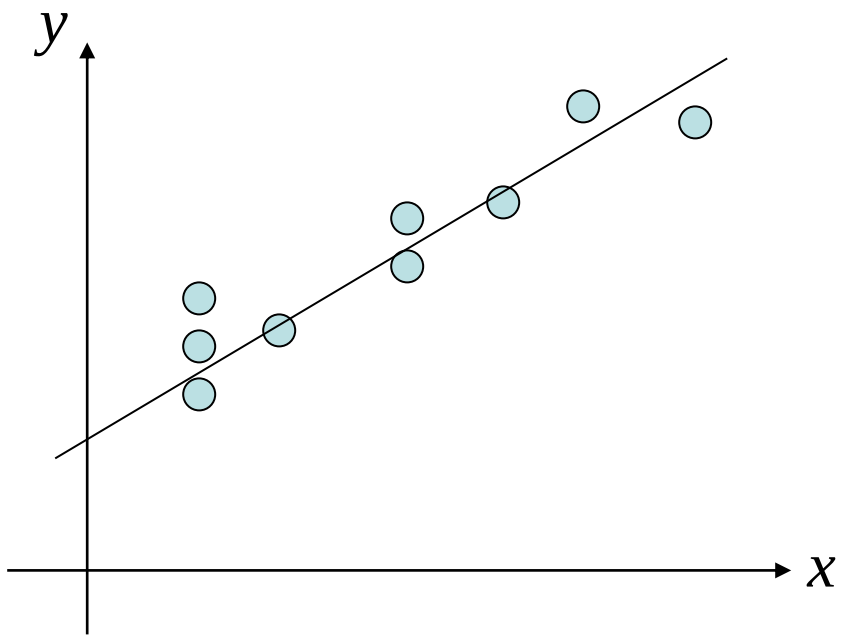
Assume a polynomial of the following form:

$$y = p_1 \cdot x^N + p_2 \cdot x^{N-1} + p_3 \cdot x^{N-2} + \cdots + p_N \cdot x + p_{N+1}$$

Assume we have also obtained a series of data (x_i, y_i)

The command p=polyfit(x,y,N) would find the coefficients of the polynomial above $p=[p_1, p_2, \ldots, p_N, p_{N+1}]$ that would best fit the measured data in the "least squares" sense.

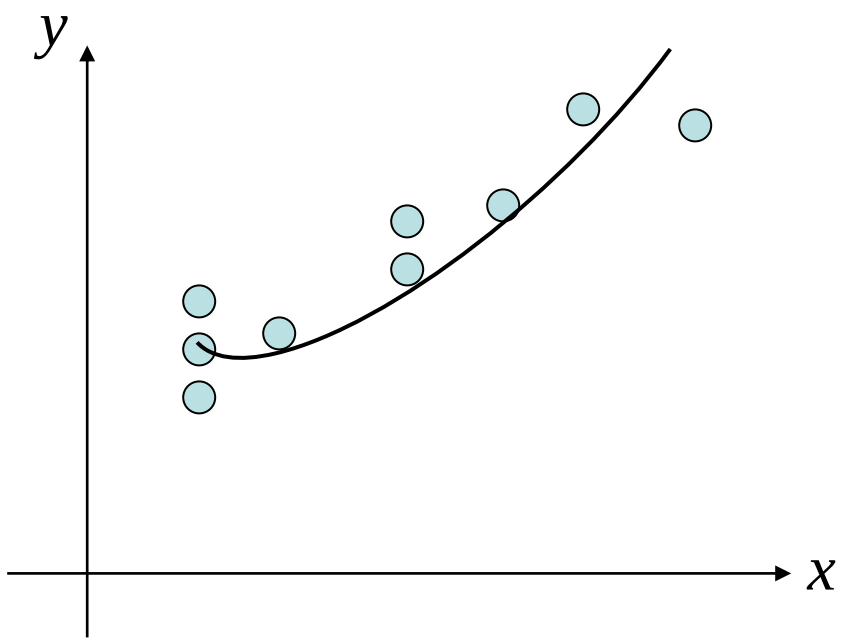
Fitting Data With a Linear Function



$y = ax+b$,
Determine *a* and *b*.

Using the *least squares* algorithm, ensure that all the data points fall close to the straight line/function.

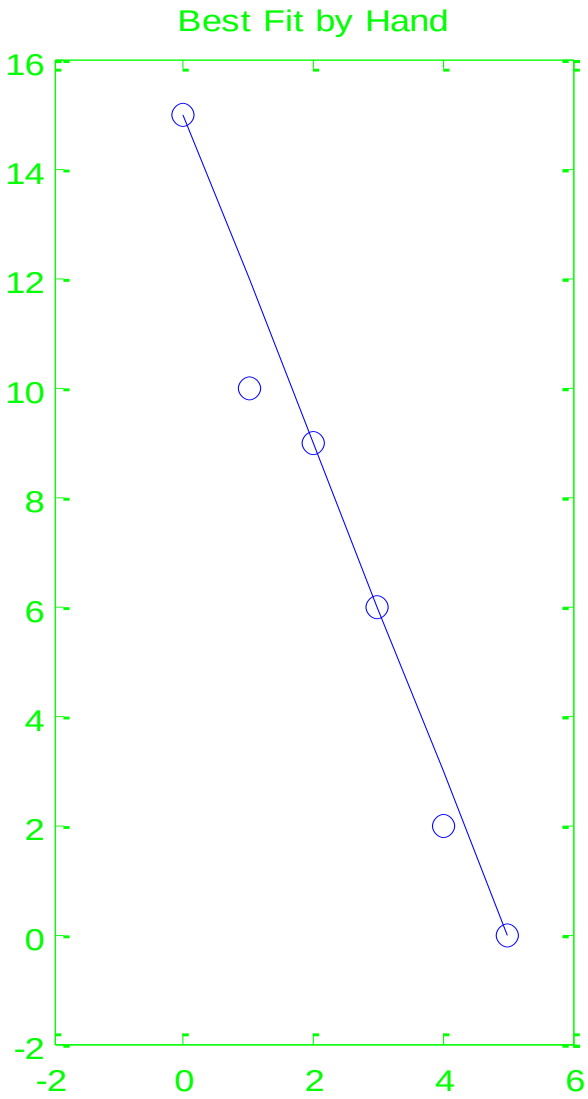
Fitting Data With a non-linear Function

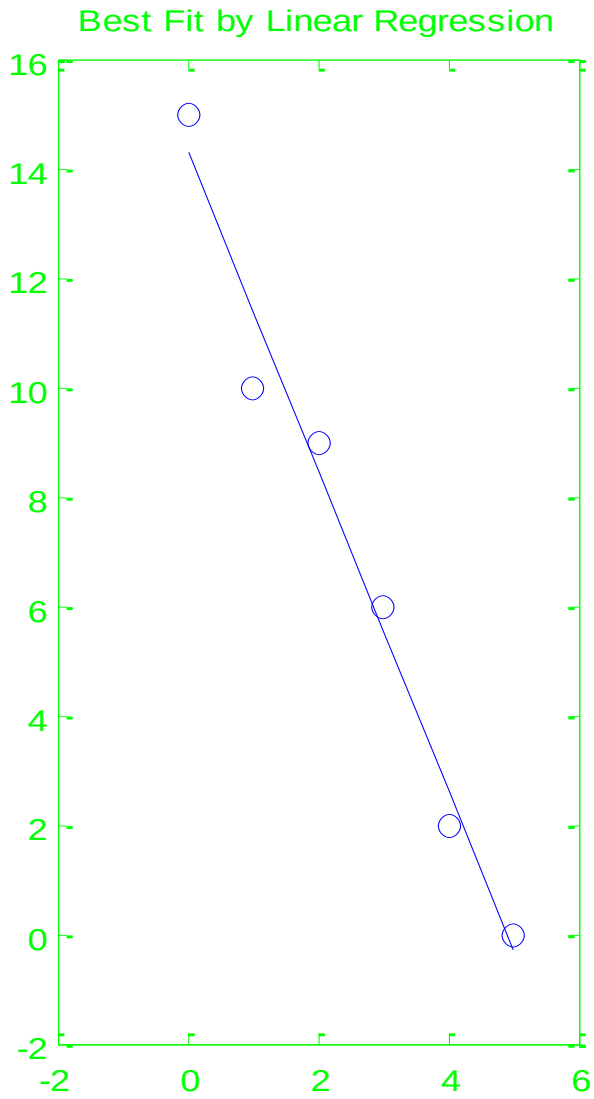


$y = ax^2+bx+c$,
Determine *a*, *b*, and *c*.

Using the *least squares* algorithm, ensure that all the data points fall close to the straight line/function.

Best Fit Comparison





Curve Fitting with Matlab's 'polyfit'

```
>>x = [0 .1 .2 .3 .4 .5 .6 .7 .8 .9 1];
>>y = [-.447 1.978 3.28 6.16 7.08 7.34 7.66 9.56 9.48 9.30 11.2];
>>n = 2;
>>p = polyfit( x, y, n ) %Find the best quadratic fit to the data

p =  -9.81083916083917        20.1292937062937
-0.0316713286713306
```

or p(x) = -9.8108x^2 + 20.1293x – 0.317

Evaluating the Curve fit with MATLAB's 'polyval'

Polyval is a built in MATLAB function for evaluating a polynomial curve fit that was calculated using polyfit.

p=polyfit(x,y,N)

ynew=polyval(p,xnew)

Where xnew is any value or vector of x, p is the coefficient vector returned by polyfit and ynew is the output

```
>>xi = linspace(0,1,100);

>>yi = polyval(p, xi);

>>plot(x,y,'-o', xi, yi, '-')

>>xlabel('x'), ylabel('y = f(x)')

>>title('Second Order Curve Fitting Example')
```

IN MATLAB

- Ohm Law $I = V/R$ 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]; % V
ydata = [0.001 0.881 2.1637 3.1827 0 4.961]; % I
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
```

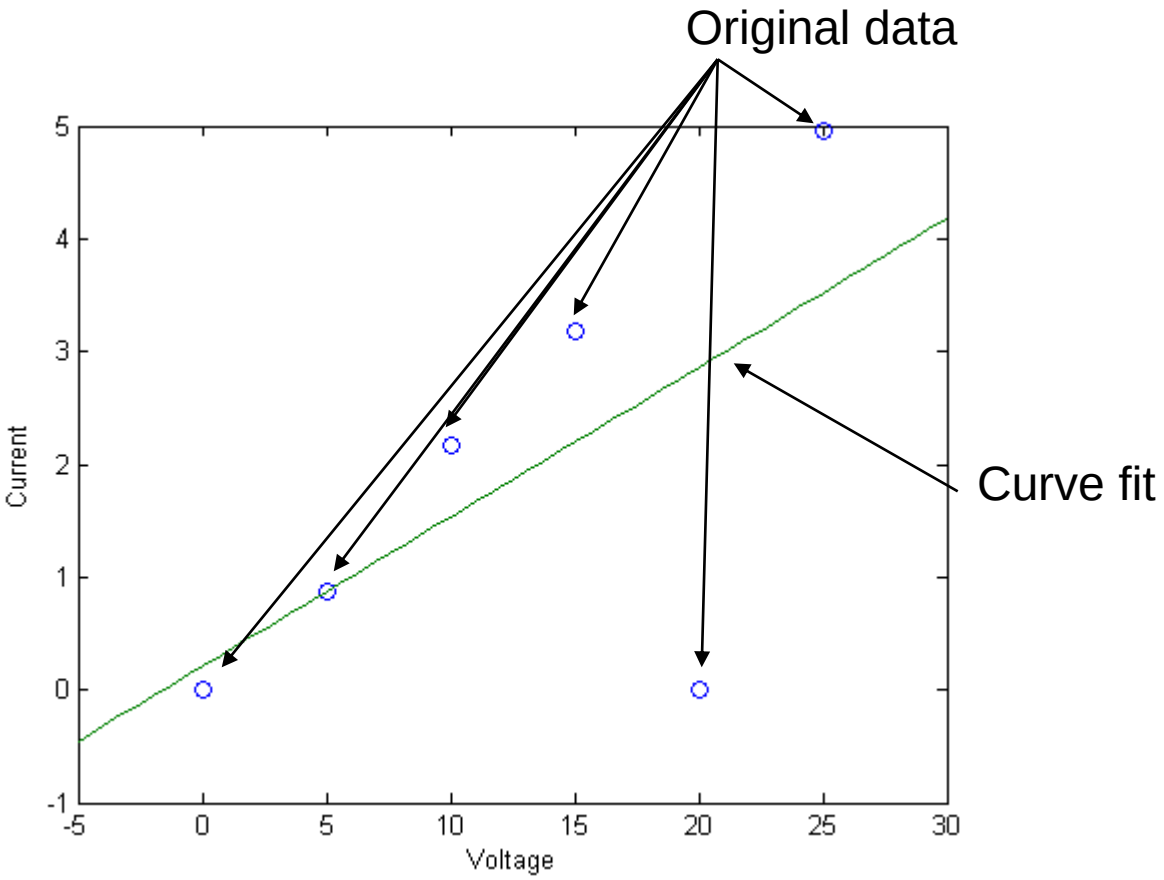
IN MATLAB: Example

- **The output from Matlab is**

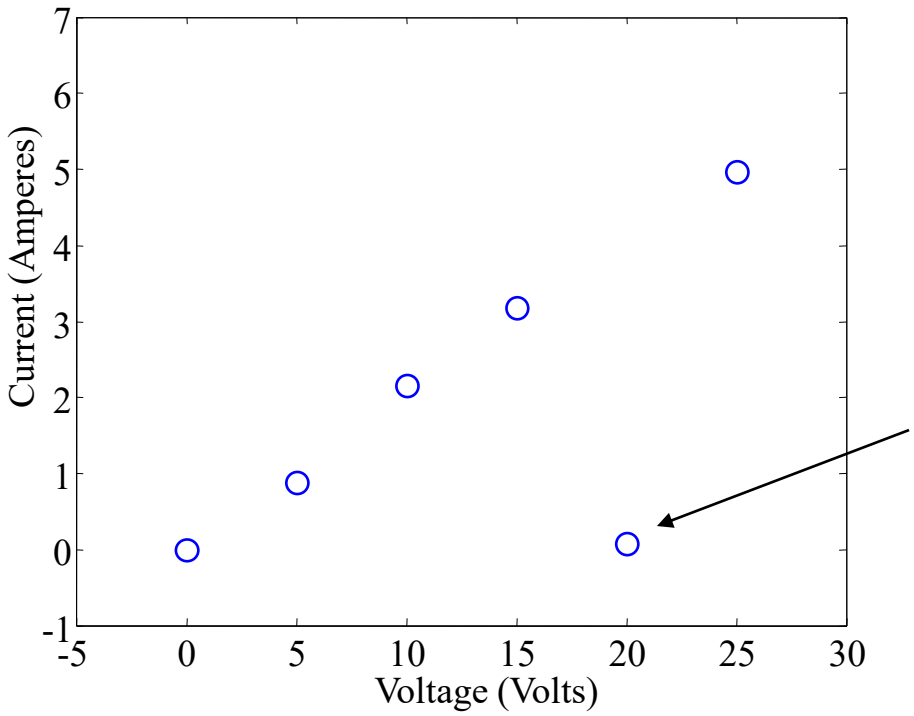
  resistance =

  7.5509

- **Thus, the resistance is 7.55 Ω.**



Example-cont.



- Data collection error

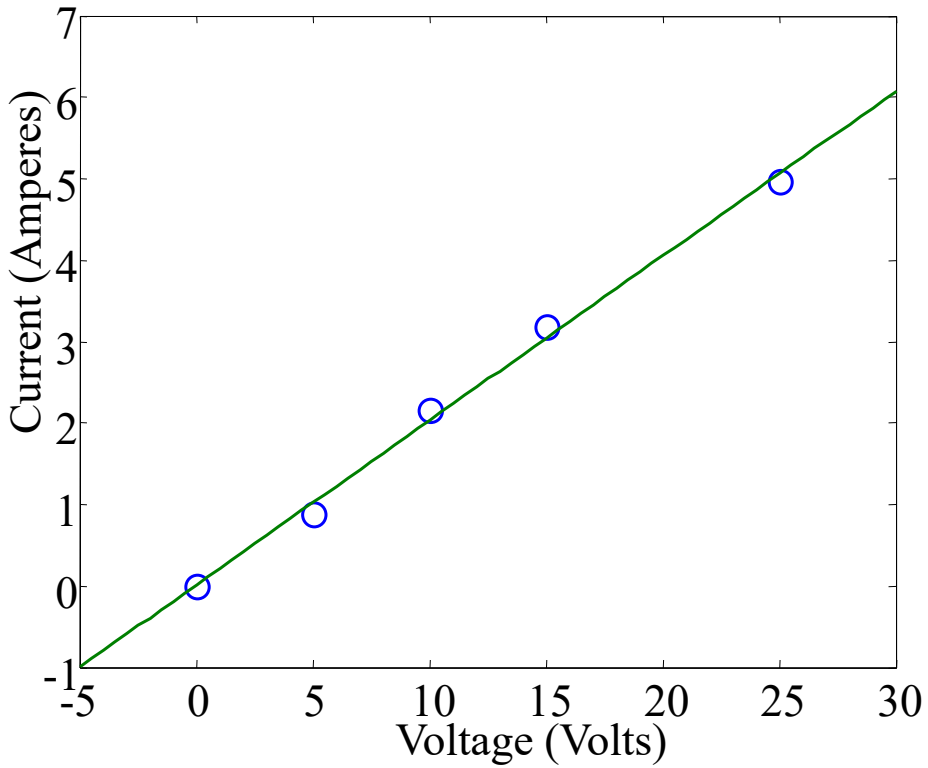
Example

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

```
% Do not include data point V=20
xdata = [0 5 10 15 25];
ydata = [0.001 0.881 2.1637 3.1827 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
```

IN MATLAB-The Plot

```
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.**

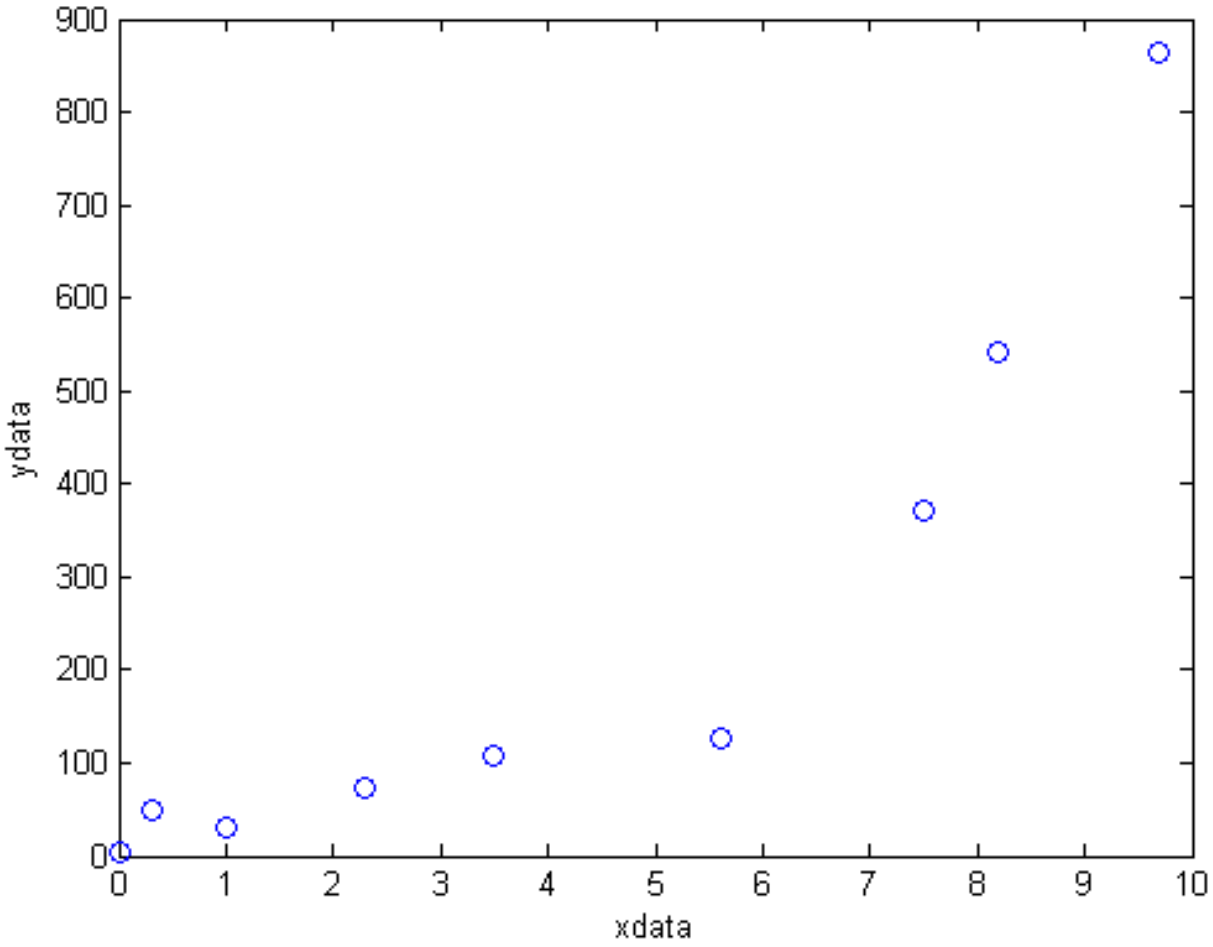
Choosing the Right Polynomial

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.

Choosing the Right Polynomial:Example



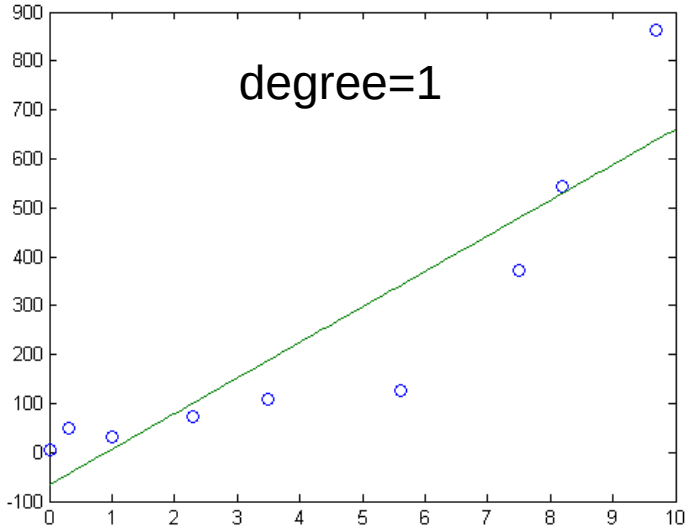

```
xdata = [0 0.3000 1.0000 2.3000 3.5000 5.6000 7.5000 8.2000 9.7000];
ydata = [4.9838   49.0727   31.0883   74.4117  107.8540  127.0157
371.8902  542.5732  863.1195];
```

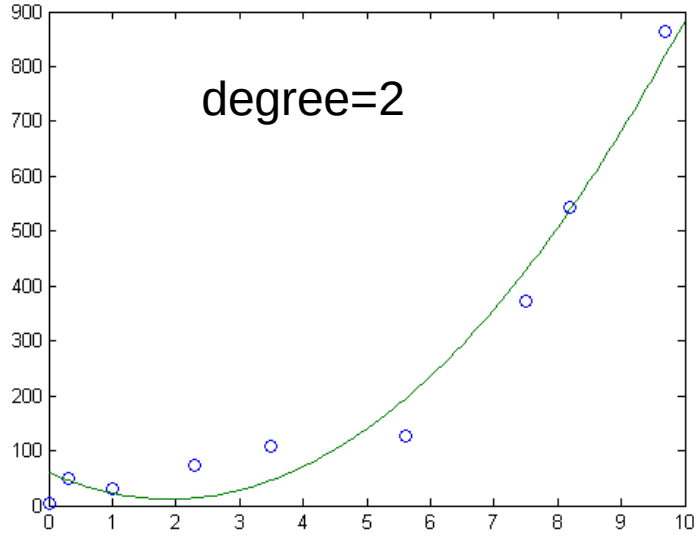
IN MATLAB

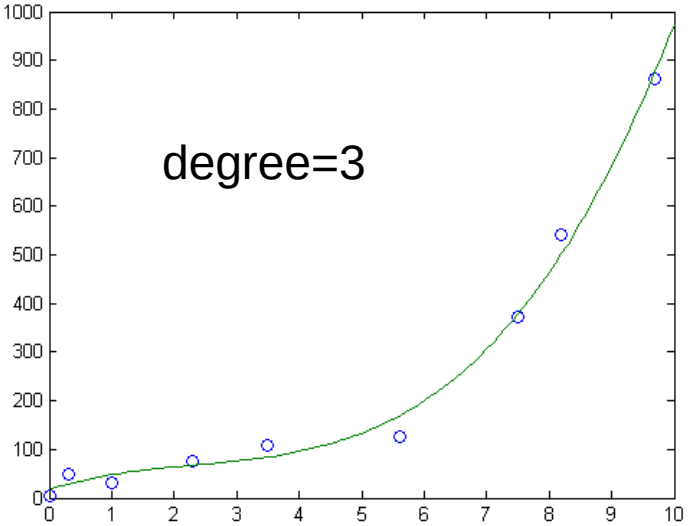
- *Write a matlab code that computes and fits polynomials from degree 1 (straight line) to degree 4.*

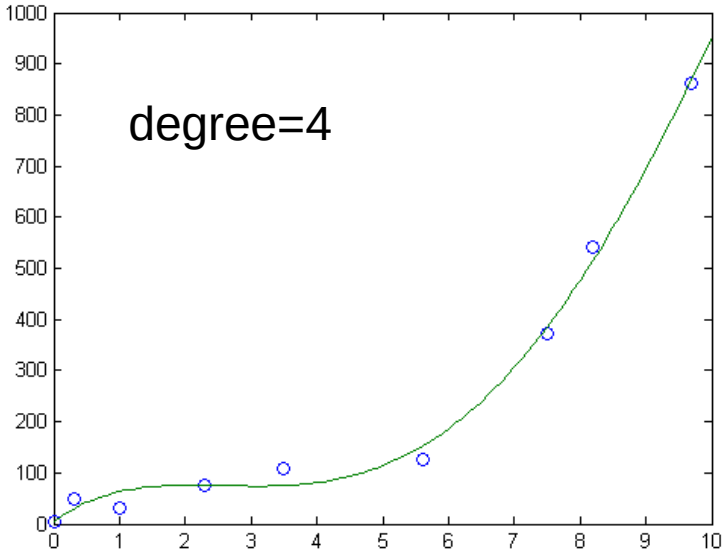
```
xdata = [0 0.3000 1.0000 2.3000 3.5000 5.6000 7.5000 8.2000 9.7000];
ydata = [4.9838   49.0727   31.0883   74.4117  107.8540  127.0157
371.8902  542.5732  863.1195];

for degree=1:4
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;
end
```

IN MATLAB









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
```

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

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

Summaries

- polyfit returns the coefficients of a polynomial that best fits the data.
- To evaluate the polynomial at any value of x, use the polyval function.
- polyval requires two inputs: the array of coefficients and the array of x-values at the locations the polynomial is to be evaluated.

Exercises

Use **`polyfit`** to find a 3rd degree polynomial to fit the following set of data:

x = -5:1:4

y = [-506.6, -262.88, -99.43, -36.78, 6.2, 7.11, 16.6, 51, 183, 427.97].

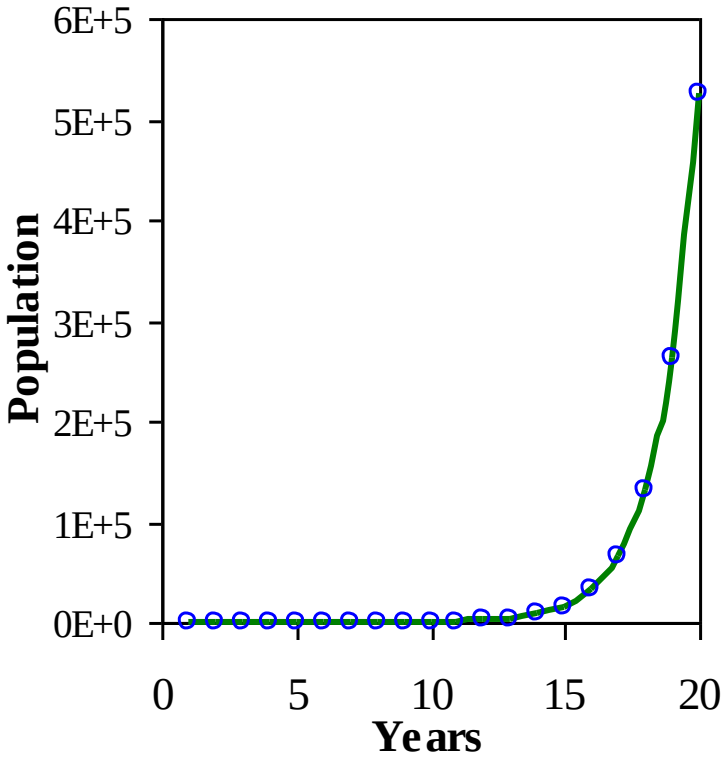
- Plot the best fit curve and the data points on the same figure.

- Thank You!!!!!!

Exponential Fit

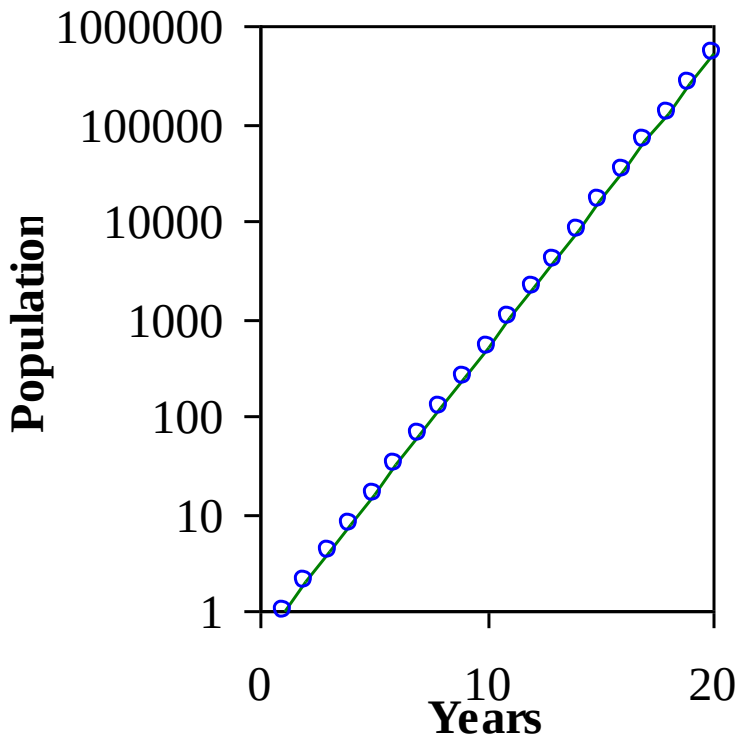
- Not all experimental data can be approximated with polynomial functions.
- Exponential data can be fit using the least squares method by first converting the data to a linear form.

Relationship between an Exponential & Linear Form



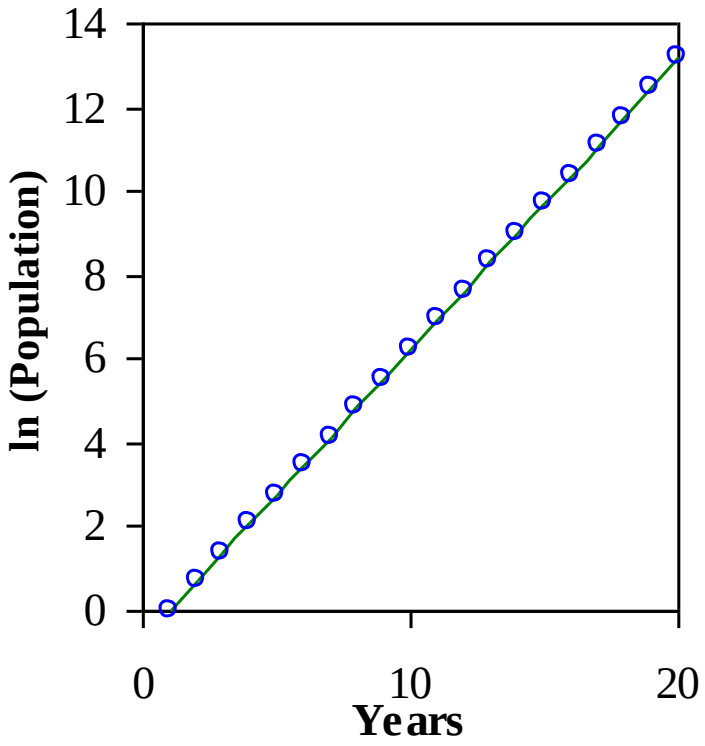
This graph shows a population that doubles every year. Notice that for the first fifteen years, growth is almost imperceptible at this scale. *It would be difficult to approximate this raw data.*

Population Example-cont.



- 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.

Population Example-cont.



- 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.

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 approximation by hand

```
x = [0,1,2,3,4,5]
y = [15,10,9,6,2,0]
```

- slope ≈ $(y_2-y_1)/(x_2-x_1)$ = (0-15) / (5-0) = -3
- Crosses y axis at 15 (note the point (0,15) in our data)
- y_{hand} = -3x + 15
- sum_of_squares = sum((y - y_{hand}).^2) = 5

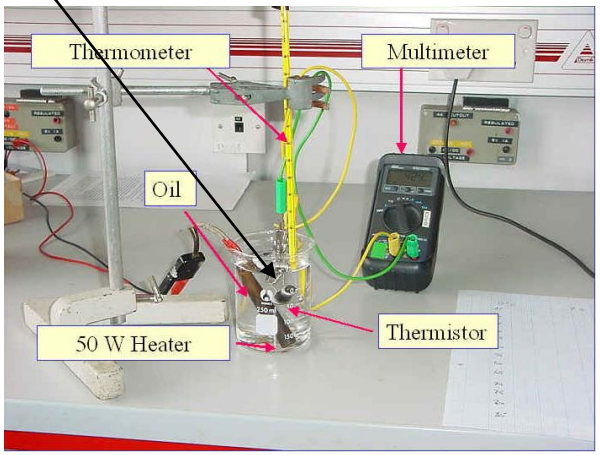
polyfit function

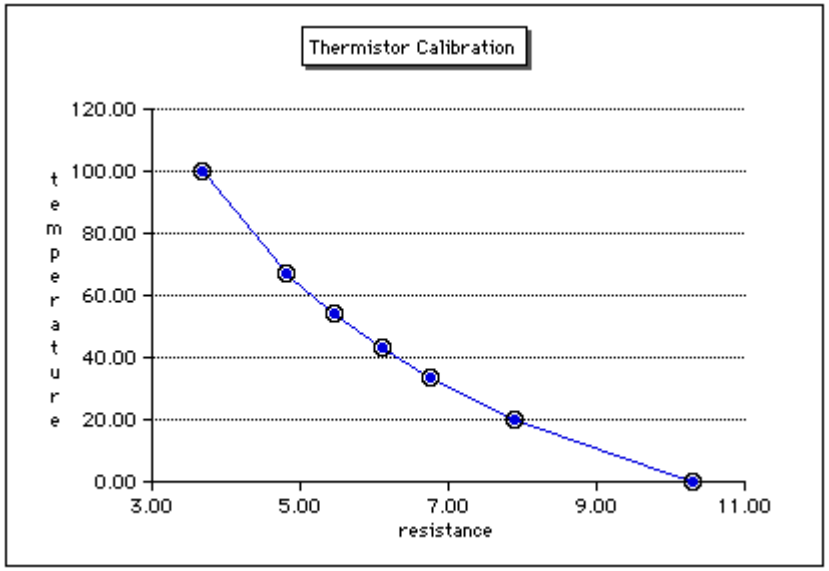
- The polyfit function takes (x, y) data, and the degree n of a polynomial as input. It returns the coefficients of the polynomial of degree n that best fits the data.
- Using our data:

```
polyfit(x,y,1)   ans = [-2.9143 14.2857]
```

- So, $y_{LR} = -2.9143x + 14.2857$
- sum_of_squares2 = sum((y_{LR} - y).^2) = 3.3714

Fitting Data to a Curve: Example: Calibration of a Sensor





A thermistor :Ectrical resistance changes a function of temperature.

"Steinhart - Hart equation"

$$\frac{1}{T} = A + B \cdot \ln(R) + C \cdot \ln(R)^3$$

Goal is to find A, B and C so that the curve best fits the measured R verses T data

Curve Fitting with MATLAB's 'polyfit'