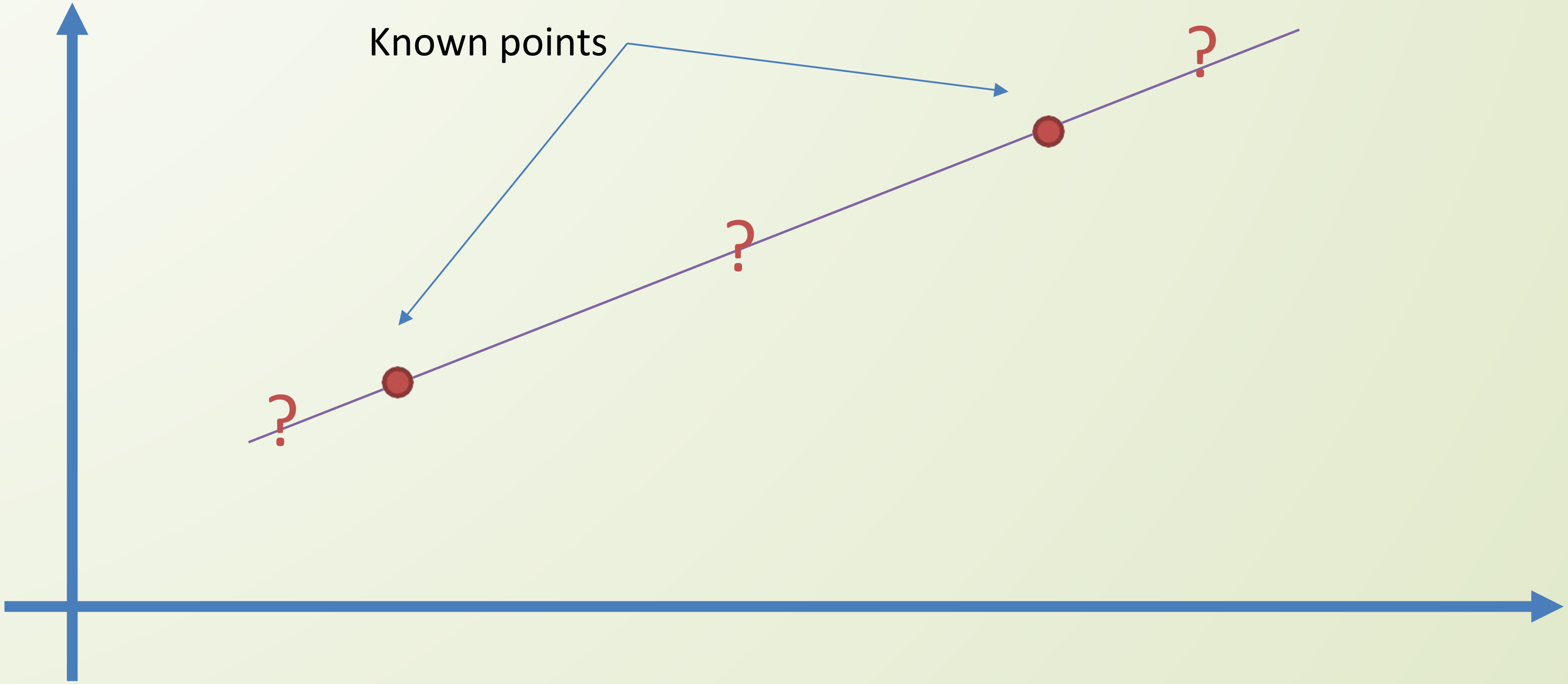
Curve fitting & Interpolation

Plotting and Curve Fitting

- Visualizing trends by plotting large sets of data from experiments or from computer calculations helps you interpret the data. There are numerous grapical tools available in MATLAB.
- Curve fitting is a powerful way to use a set of data to find a mathematical model that approximates the set of data.

Interpolation

Interpolation is used to estimate data points between two known points. The most common interpolation technique is Linear Interpolation.



Interpolation

- Interpolation is used to estimate data points between two known points. The most common interpolation technique is Linear Interpolation.
- In MATLAB we can use the *interp1()* function.
- The default is linear interpolation, but there are other types available, such as:
  - linear
  - nearest
  - spline
  - cubic

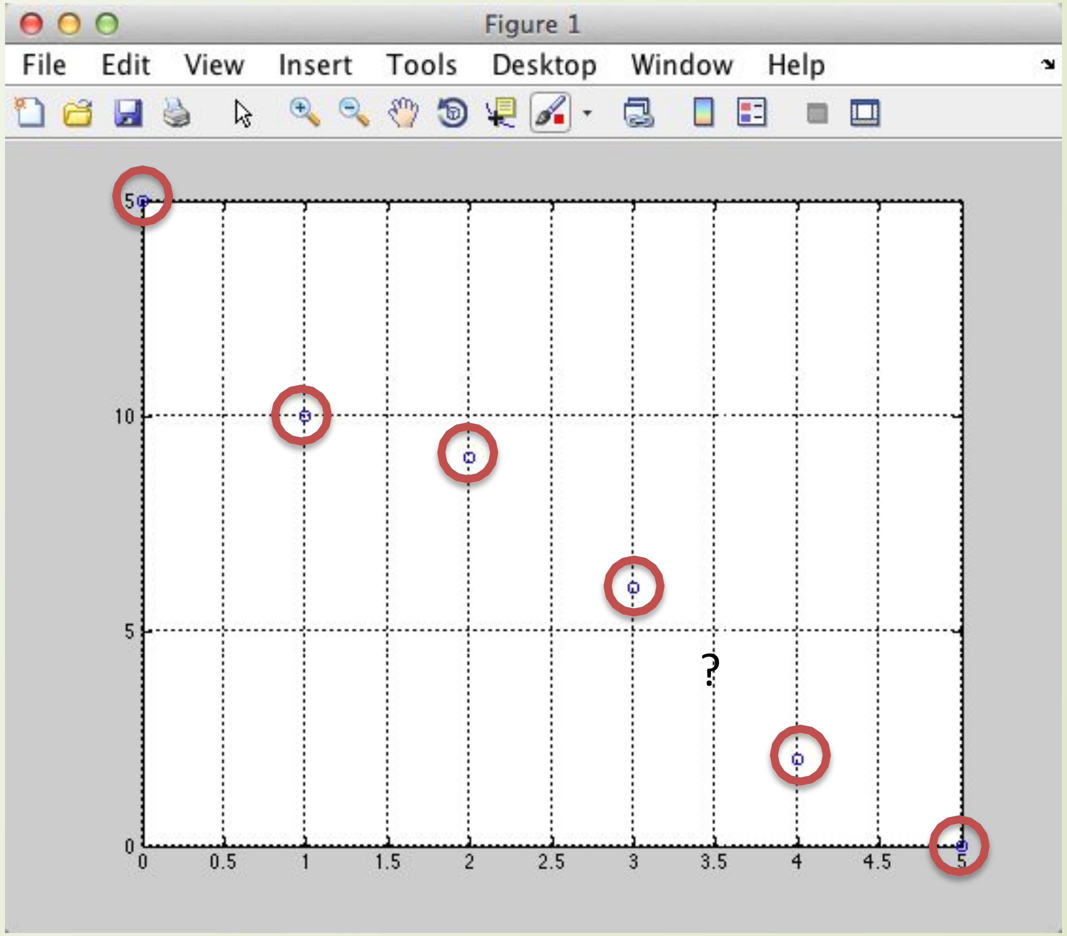
Interpolation

Given the following Data Points:

x	y
0	15
1	10
2	9
3	6
4	2
5	0

(Logged Data from a given Process)

```
x=0:5;
y=[15, 10, 9, 6, 2, 0];

plot(x,y ,'o')
grid
```



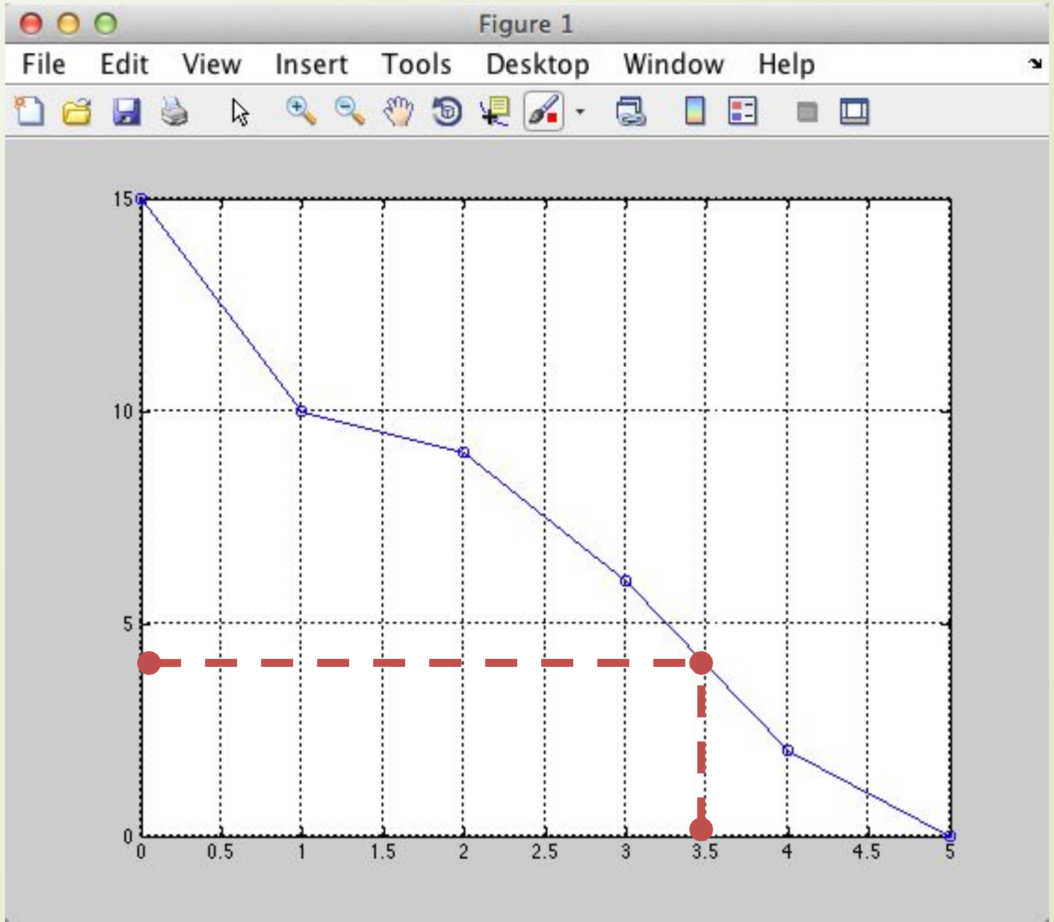
Problem: Assume we want to find the interpolated value for, e.g., $x = 3.5$

Interpolation

We can use one of the built-in Interpolation functions in MATLAB:

```
x=0:5;
y=[15, 10, 9, 6, 2, 0];

plot(x,y ,'-o')
grid on

new_x=3.5;
new_y = interp1(x,y,new_x)
```

→ **new_y =4**



Interpolation

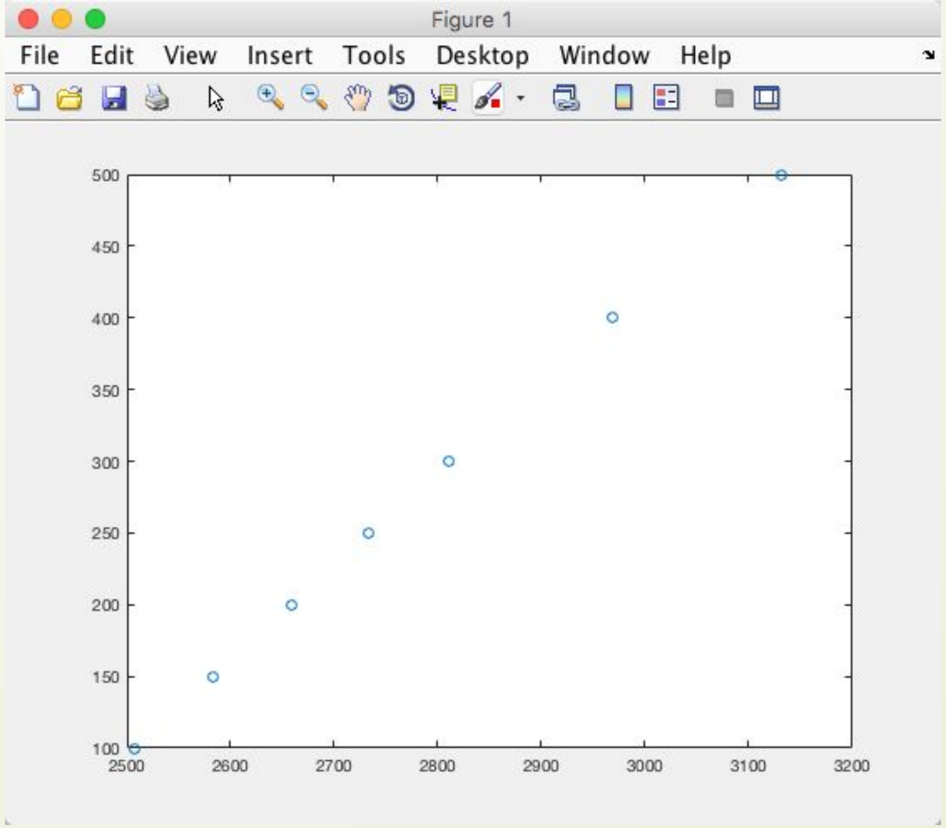
Given the following data:

Temperature, T [°C]	Energy, u [KJ/kg]
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6

- Plot u versus T.
- Find the interpolated data and plot it in the same graph.
- Test out different interpolation types (spline, cubic).
- What is the interpolated value for u=2680.78 KJ/kg?

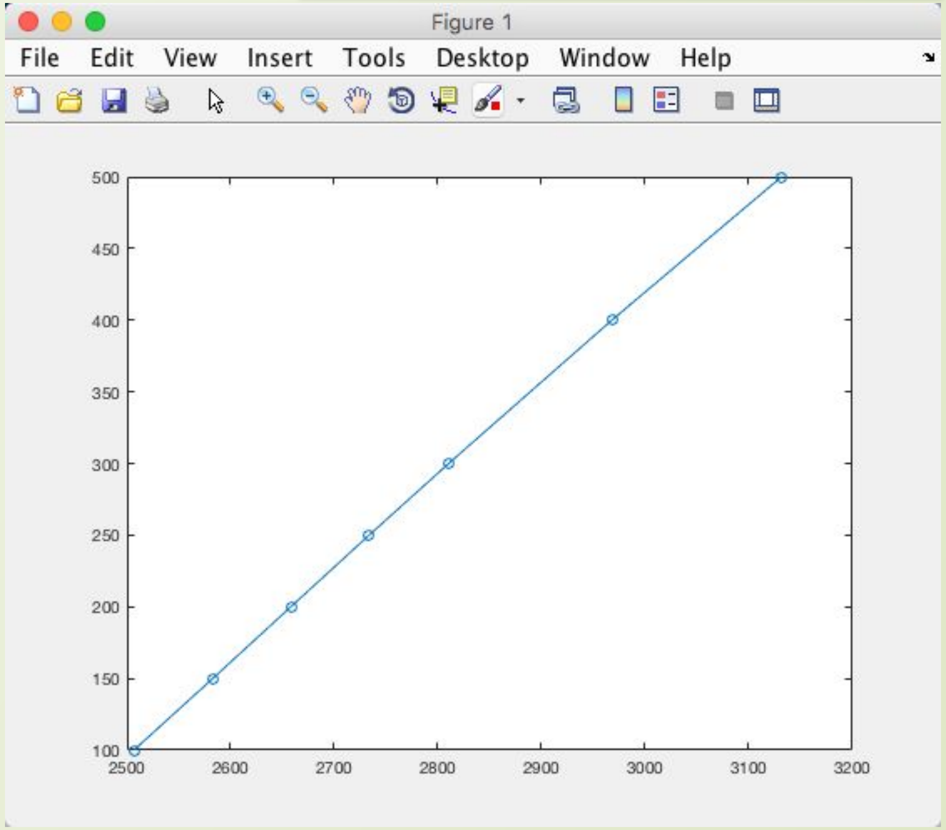
```
clear
clc

T = [100, 150, 200, 250, 300, 400, 500];
u=[2506.7, 2582.8, 2658.1, 2733.7, 2810.4, 2967.9, 3131.6];

figure(1)
plot(u,T, '-o')


% Find interpolated value for u=2680.78
new_u=2680.78;
interp1(u, T, new_u)

%Spline
new_u = linspace(2500,3200,length(u));
new_T = interp1(u, T, new_u, 'spline');
figure(2)
plot(u,T, new_u, new_T, '-o')
```

```
T = [100, 150, 200, 250, 300, 400, 500];
u=[2506.7, 2582.8, 2658.1, 2733.7, 2810.4, 2967.9, 3131.6];


figure(1)
plot(u,T, 'o')
```



or:

```
plot(u,T, '-o')
```



```
% Find interpolated value for u=2680.78
new_u=2680.78;
interp1(u, T, new_u)
```

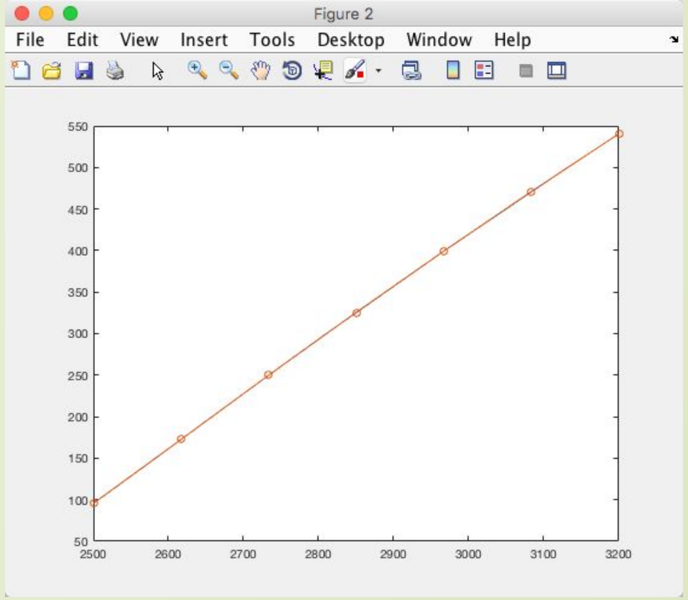
The interpolated value for u=2680.78 KJ/kg

is:

```
ans =
  215.0000
```

```
%Spline
new_u = linspace(2500,3200,length(u));
new_T = interp1(u, T, new_u, 'spline');
figure(2)
plot(u,T, new_u, new_T, '-o')
```

For ‘spline’/’cubic’ we get almost the same. This is because the points listed above are quite linear in their nature.



Define the sample points, x, and corresponding sample values, v.

x = 0:pi/4:2*pi;

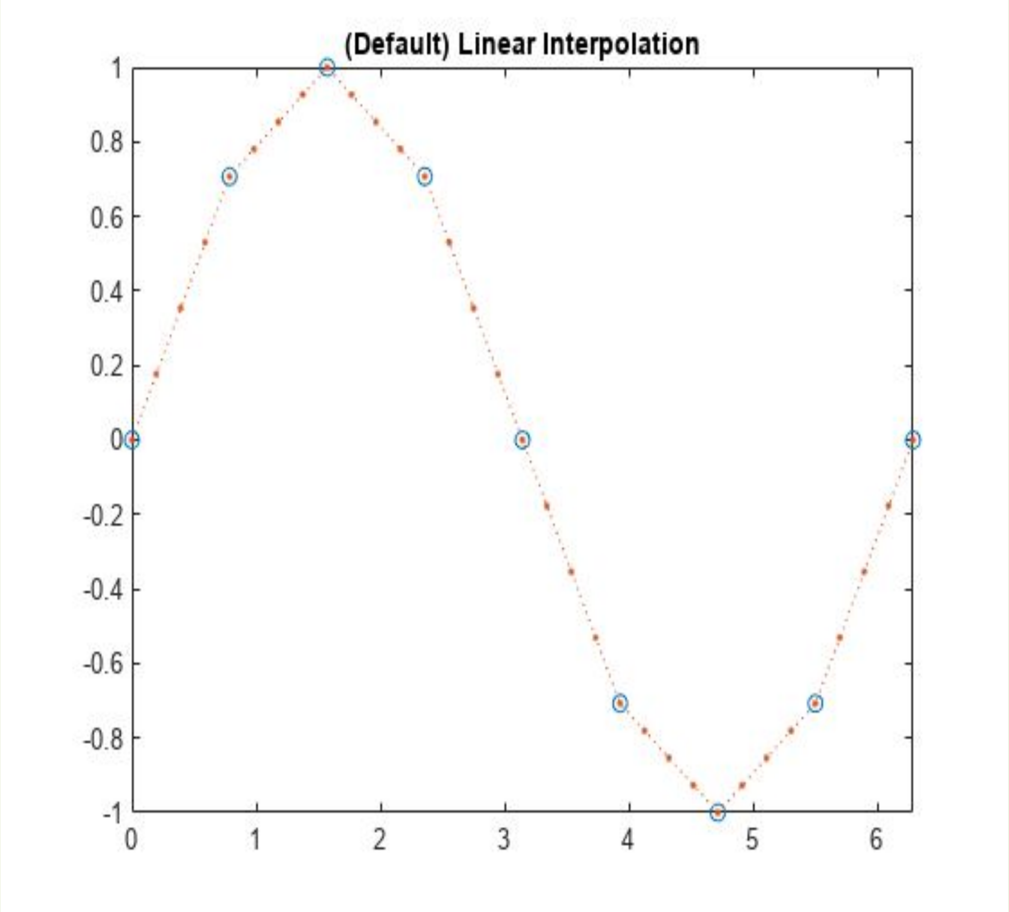
v = sin(x);

Define the query points to be a finer sampling over the range of x.

xq = 0:pi/16:2*pi;

Interpolate the function at the query points and plot the result.

```
figure
vq1 = interp1(x,v,xq);
plot(x,v,'o',xq,vq1,':.');
xlim([0 2*pi]);
title('(Default) Linear Interpolation');
```



Curve fitting

- The simplest way to fit a set of 2D data is a straight line.
- Linear regression is a method of fitting data with a straight line.
- Linear regression minimizes the squared distance between data points and the equation modeling the data points. This prevents positive and negative “errors” from canceling.

Linear approximation by hand

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

- slope ≈ $(y_2\text{-}y_1)/(x_2\text{-}x_1)$ = (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

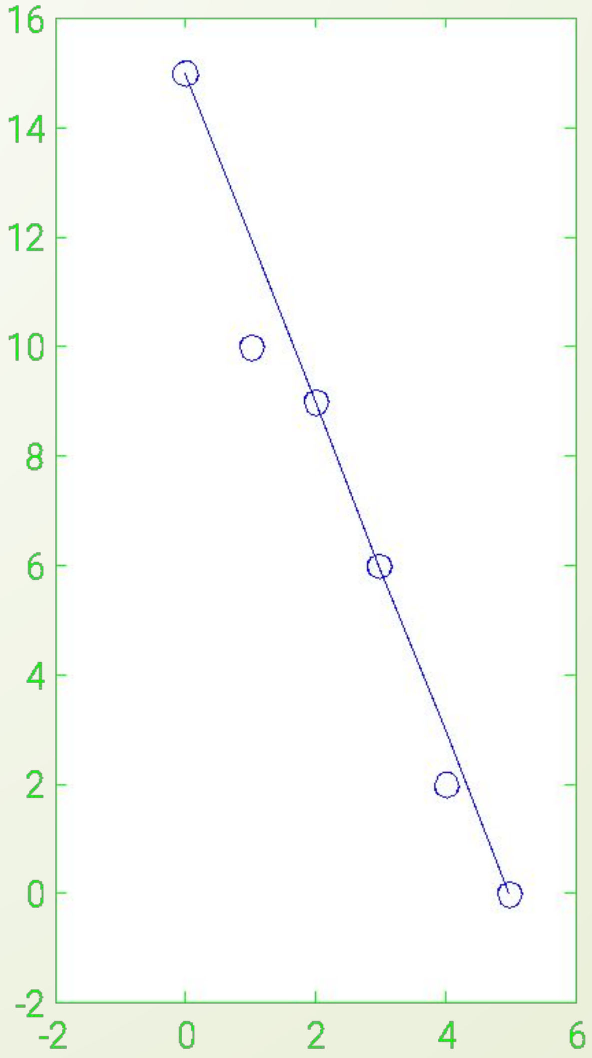
polyfit function

- 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:

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

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

Best Fit Comparison



Polynomial regression

- Polynomial regression is used to fit a set of data with a polynomial.
- The polyfit function can be used to find the best fit polynomial of a specified degree; the result is the coefficients.
- **Warning:** Increasing the degree of the best fit polynomial can create mathematical models that ay fit the data better, but care must be taken in your interpretation of the result.

polyval function

- 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.

Example using polyval

- Referring to the data from this lecture that we used from the polyfit example:

```
coef = polyfit(x,y,1)
coef = [-2.9143   14.2857]
fitted_data = polyval(coef,x);
```

- Then by plotting fitted_data vs. x and y_{LR} vs. x. Identical graphs are generated.

Generate 10 points equally spaced along a sine curve in the interval [0,4*pi].

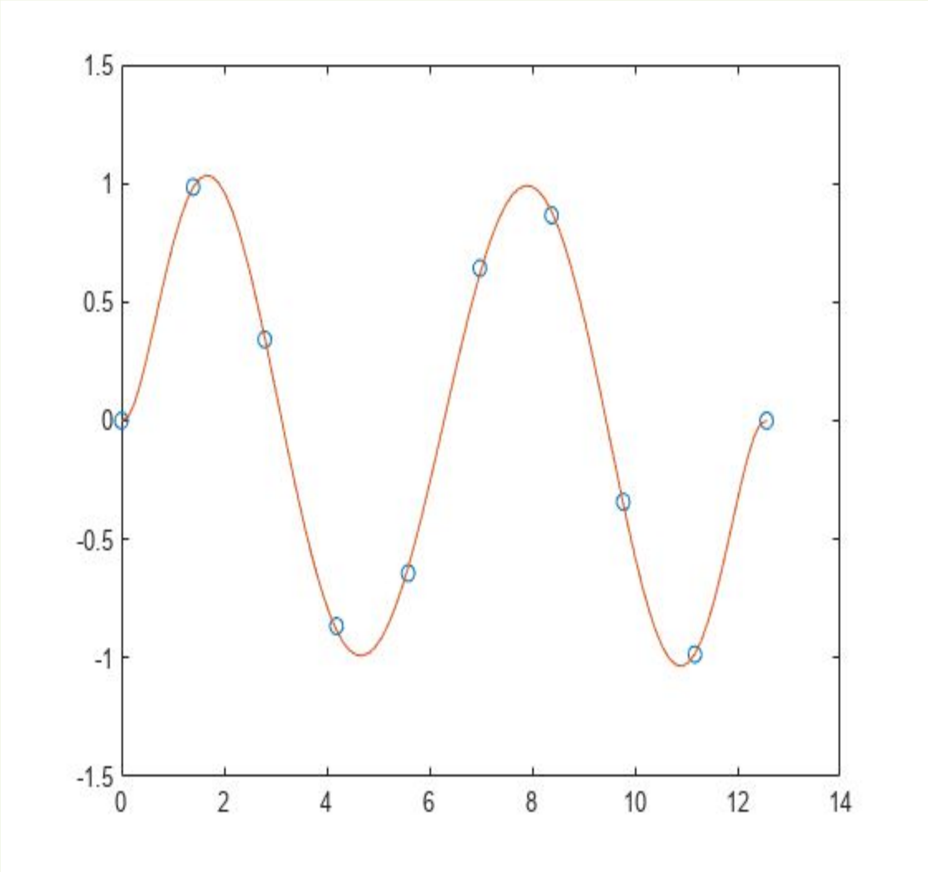
```
x = linspace(0,4*pi,10);

y = sin(x);
```

Use polyfit to fit a 7th-degree polynomial to the points.

```
p = polyfit(x,y,7);
```

Evaluate the polynomial on a finer grid and plot the results.

```
x1 = linspace(0,4*pi);

y1 = polyval(p,x1);

figure

plot(x,y,'o')

hold on

plot(x1,y1)

hold off
```



Create a vector of 5 equally spaced points in the interval [0,1], and evaluate $y(x)=(1+x)-1$ at those points.

```
x = linspace(0,1,5);
y = 1./(1+x);
```

Fit a polynomial of degree 4 to the 5 points. In general, for n points, you can fit a polynomial of degree n-1 to exactly pass through the points.

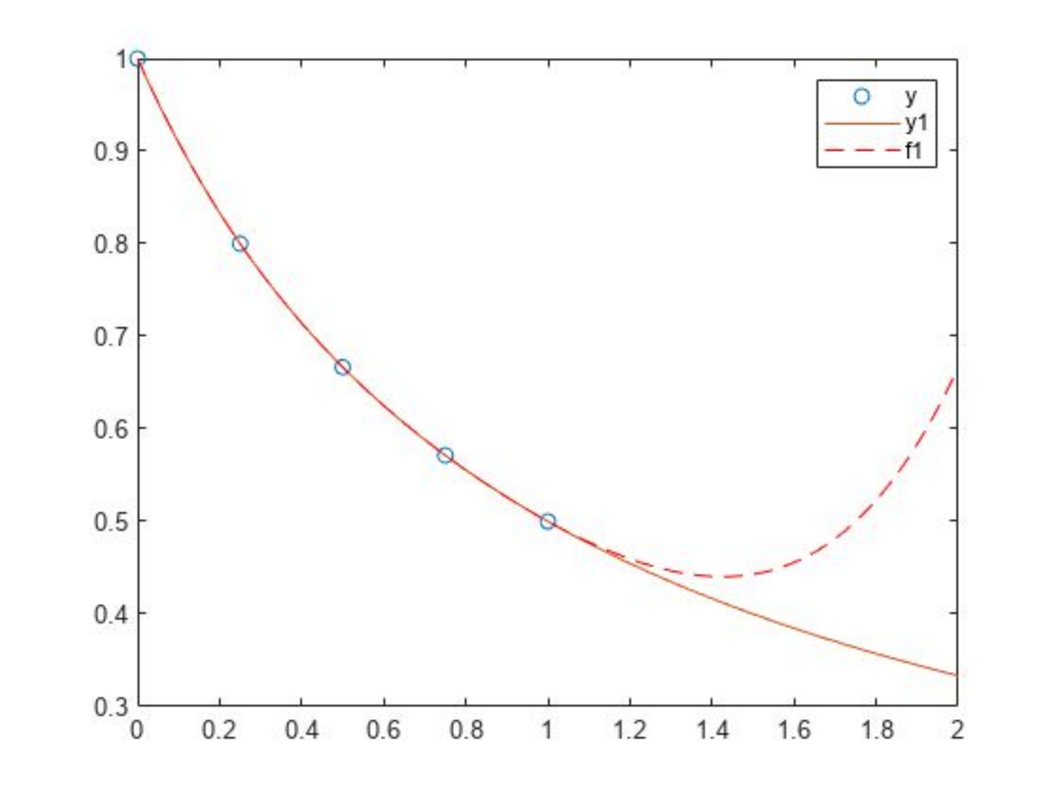
```
p = polyfit(x,y,4);
```

Evaluate the original function and the polynomial fit on a finer grid of points between 0 and 2.

```
x1 = linspace(0,2);
y1 = 1./(1+x1);
f1 = polyval(p,x1);
```

Plot the function values and the polynomial fit in the wider interval [0,2], with the points used to obtain the polynomial fit highlighted as circles. The polynomial fit is good in the original [0,1] interval, but quickly diverges from the fitted function outside of that interval.

```
figure
plot(x,y,'o')
hold on
plot(x1,y1)
plot(x1,f1,'r--')
legend('y','y1','f1')
```

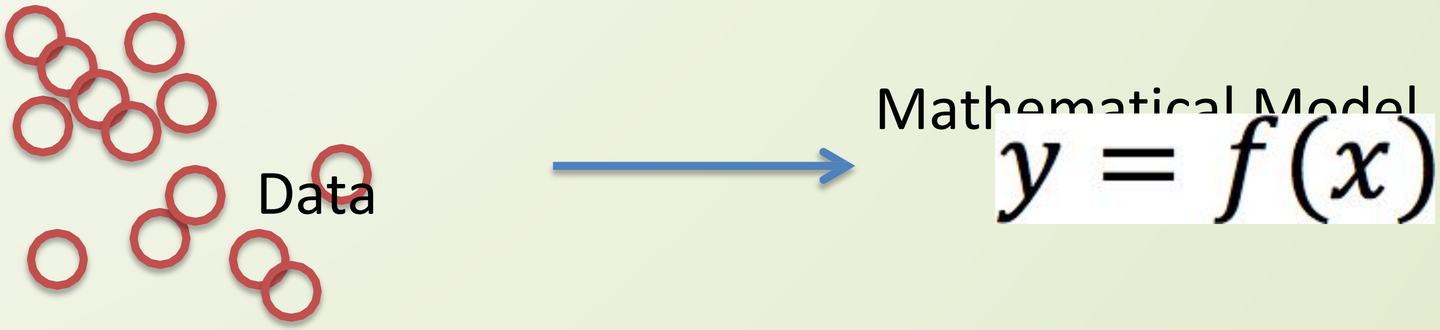


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
y
y1
f1

Curve Fitting

- In the previous section we found interpolated points, i.e., we found values between the measured points using the interpolation technique.
- It would be more convenient to model the data as a mathematical function

  $y = f(x)$.
- Then we can easily calculate any data we want based on this model.



Curve Fitting

- MATLAB has built-in curve fitting functions that allows us to create empiric data model.
- It is important to have in mind that these models are good only in the region we have collected data.
- Here are some of the functions available in MATLAB used for curve fitting:
  - *polyfit()*
  - *polyval()*
- These techniques use a polynomial of degree N that fits the data

Regression Models

Linear Regression:

$$y(x) = ax + b$$

Polynomial Regression:

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

1. order (linear):

$$y(x) = ax + b$$

2. order:

$$y(x) = ax^2 + bx + c$$

etc.

Linear Regression

Given the following data:

Temperature, T [°C]	Energy, u [KJ/kg]
100	2506.7
150	2582.8
200	2658.1
250	2733.7
300	2810.4
400	2967.9
500	3131.6

Plot u versus T.

Find the linear regression model from the data

$$y = ax + b$$

Plot it in the same graph.

```
T = [100, 150, 200, 250, 300, 400, 500];
u=[2506.7, 2582.8, 2658.1, 2733.7, 2810.4, 2967.9,
3131.6];
n=1; % 1.order polynomial(linear
regression) p=polyfit(u,T,n);

a=p(1)
b=p(2)

x=u;
ymodel=a*x+b;

plot(u,T,'o',u,ymodel)
```

a =

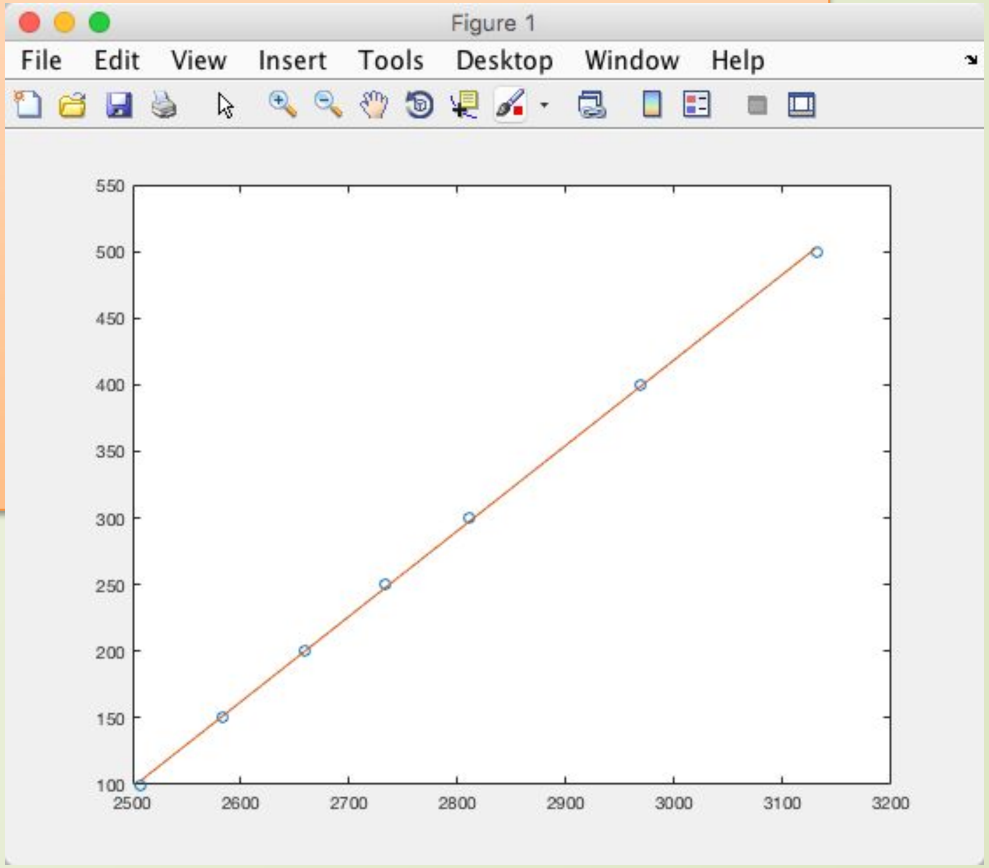
0.6415

b =

-1.5057e+003

$$y \approx 0.64x - 1.5 / 10^3$$

i.e, we get a polynomial $p = [0.6, -1.5 + 10^3]$



Polynomial Regression

Given the following data:

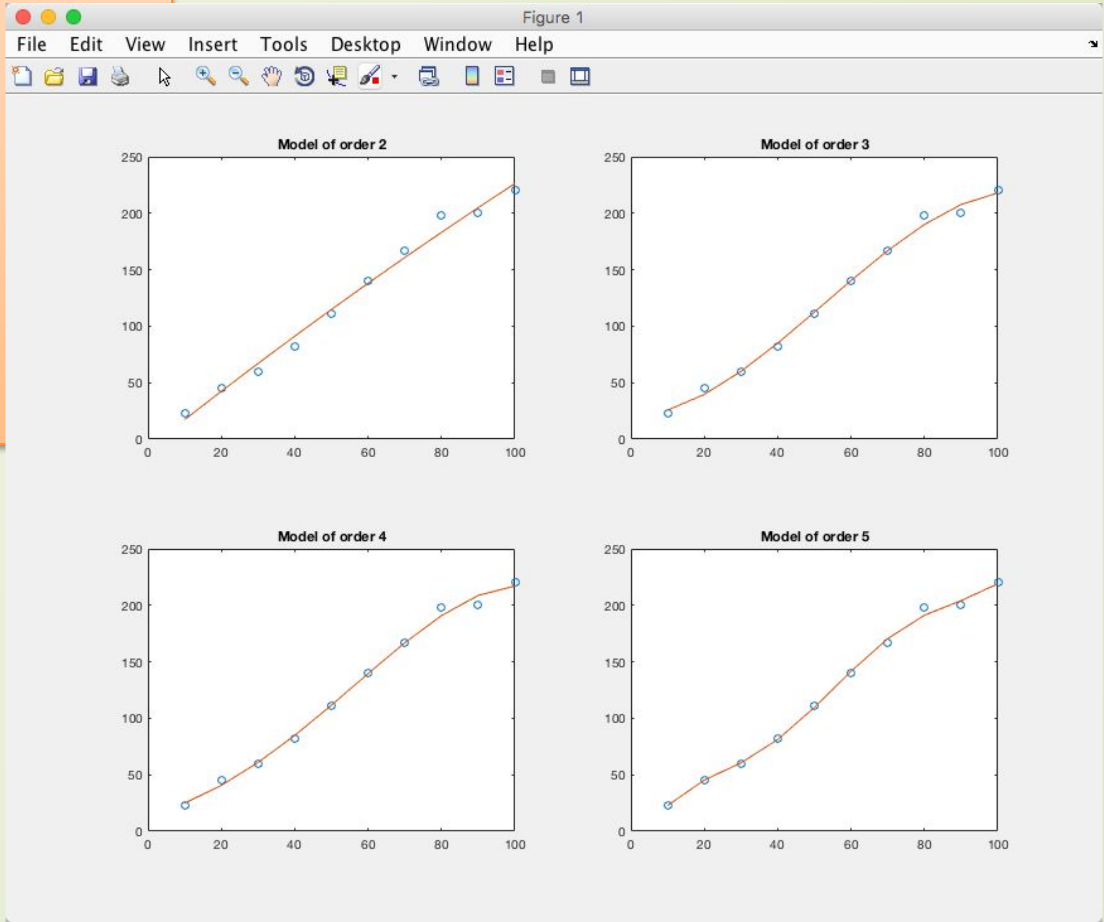
x	y
10	23
20	45
30	60
40	82
50	111
60	140
70	167
80	198
90	200
100	220

In polynomial regression we will find the following model:

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

- We will use the **polyfit** and **polyval** functions in MATLAB and compare the models using different orders of the polynomial.
- We will use subplots then add titles, etc.

```
clear, clc

x=[10, 20, 30, 40, 50, 60, 70, 80, 90, 100];
y=[23, 45, 60, 82, 111, 140, 167, 198, 200, 220];

for n=2:5
    p=polyfit(x,y,n);

    ymodel=polyval(p,x);

    subplot(2,2,n-1)
    plot(x,y,'o',x,ymodel)
    title(sprintf('Model of order %d', n));
end
```



Model Fitting

Given the following data:

Height, h[ft]	Flow, f[ft^3/s]
0	0
1.7	2.6
1.95	3.6
2.60	4.03
2.92	6.45
4.04	11.22
5.24	30.61

- We will create a 1. (linear), 2. (quadratic) and 3.order (cubic) model.
- Which gives the best model? We will plot the result in the same plot and compare them.
- We will add xlabel, ylabel, title and a legend to the plot and use

```
clear, clc
% Real Data
height = [0, 1.7, 1.95, 2.60, 2.92, 4.04, 5.24];
flow = [0, 2.6, 3.6, 4.03, 6.45, 11.22, 30.61];

new_height = 0:0.5:6; % generating new height values used to test the model

%linear--------------------------------------
polyorder = 1; %linear
p1 = polyfit(height, flow, polyorder) % 1.order model
new_flow1 = polyval(p1,new_height); % We use the model to find new flow values

%quadratic--------------------------------------
polyorder = 2; %quadratic
p2 = polyfit(height, flow, polyorder) % 2.order model
new_flow2 = polyval(p2,new_height); % We use the model to find new flow values
--------------------------------------
%cubic
polyorder = 3; %cubic
p3 = polyfit(height, flow, polyorder) % 3.order model
new_flow3 = polyval(p3,new_height); % We use the model to find new flow values

%Plotting
%We plot the original data together with the model found for comparison
plot(height, flow, 'o', new_height, new_flow1, new_height, new_flow2, new_height,
new_flow3) title('Model   fitting')
xlabel('height'
)
ylabel('flow')
```

The result becomes:

```
p1 =
    5.3862  -5.8380
p2 =
    1.4982  -2.5990     1.1350
p3 =
    0.5378  -2.6501  4.9412-0.1001
```

Where p1 is the linear model (1.order), p2 is the quadratic model (2.order) and p3 is the cubic model (3.order).

This gives:

1. order model:

$$p_1 = a_0 x + a_1 = 5.4x - 5.8$$

2. order model:

$$p_2 = a_0 x^2 + a_1 x + a_2 = 1.5x^2 - 2.6x + 1.1$$

3. order model:

$$p_3 = a_0 x^3 + a_1 x^2 + a_2 x + a_3 = 0.5x^3 - 2.7x^2 + 4.9x - 0.1$$

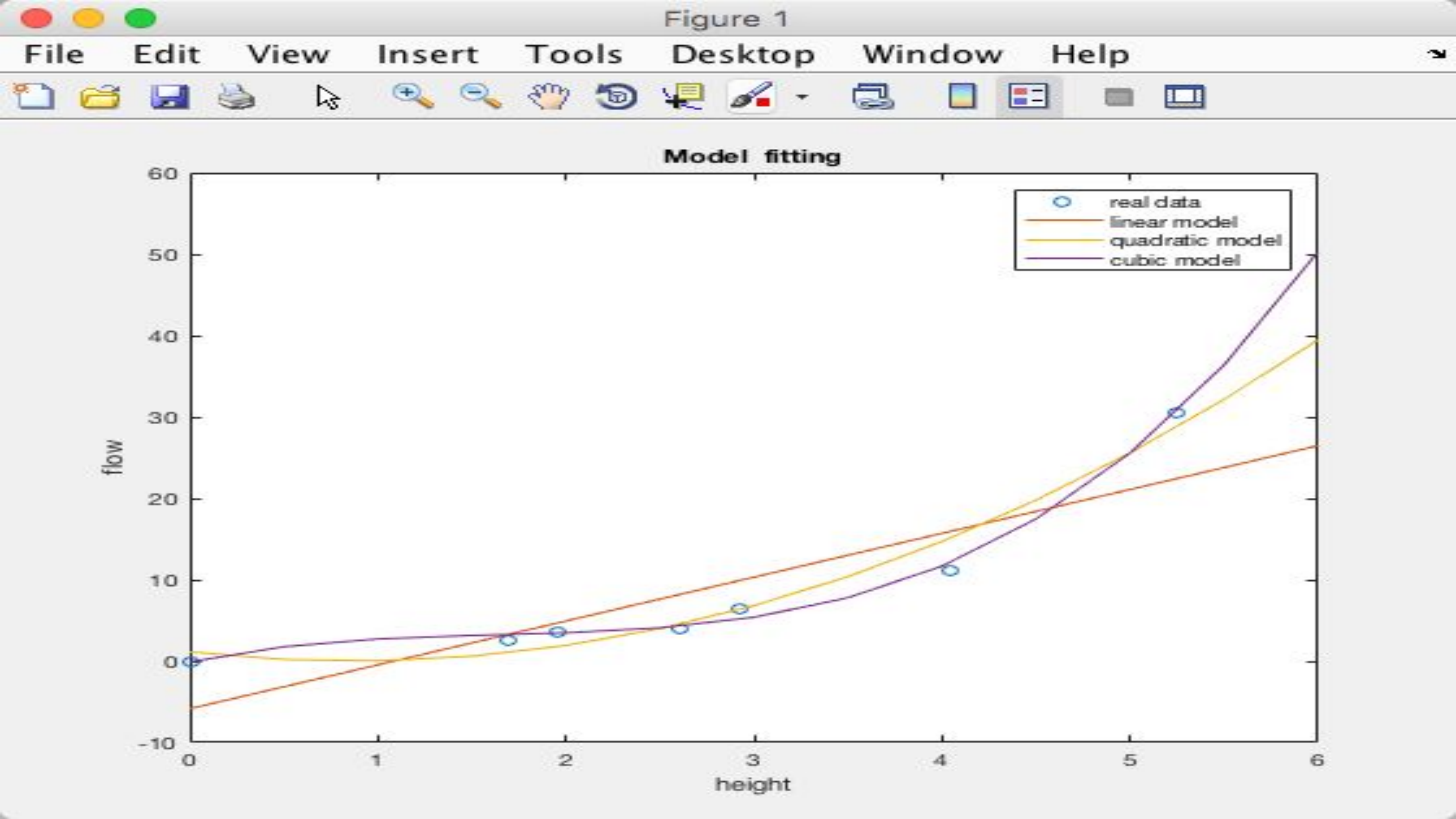


Figure 1
File Edit View Insert Tools Desktop Window Help
Model fitting
real data
linear model
quadratic model
cubic model
flow
height
60
50
40
30
20
10
0
-10
0
1
2
3
4
5
6