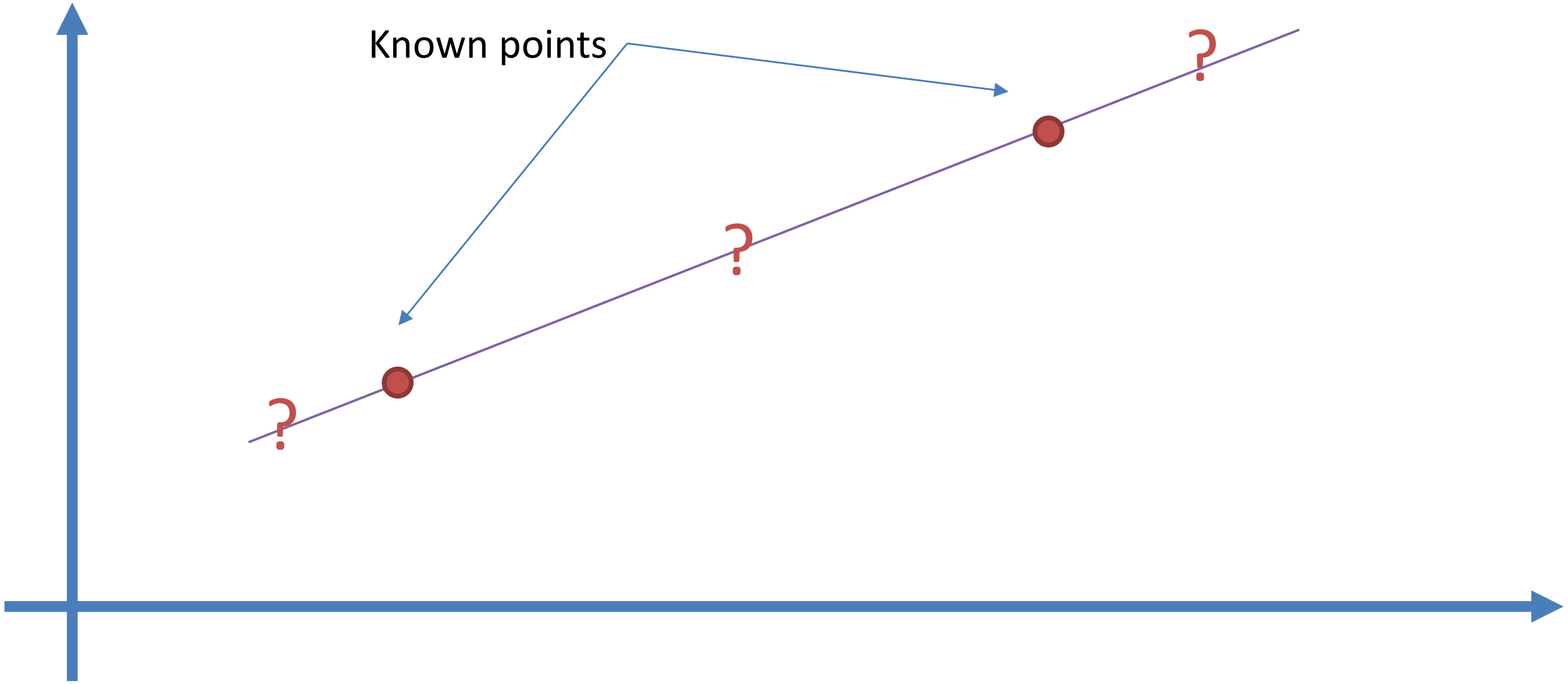
Interpolation and Curve Fitting with MATLAB

Hans-Petter Halvorsen

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
  - etc.
- Type “help interp1” in order to read more about the different options.

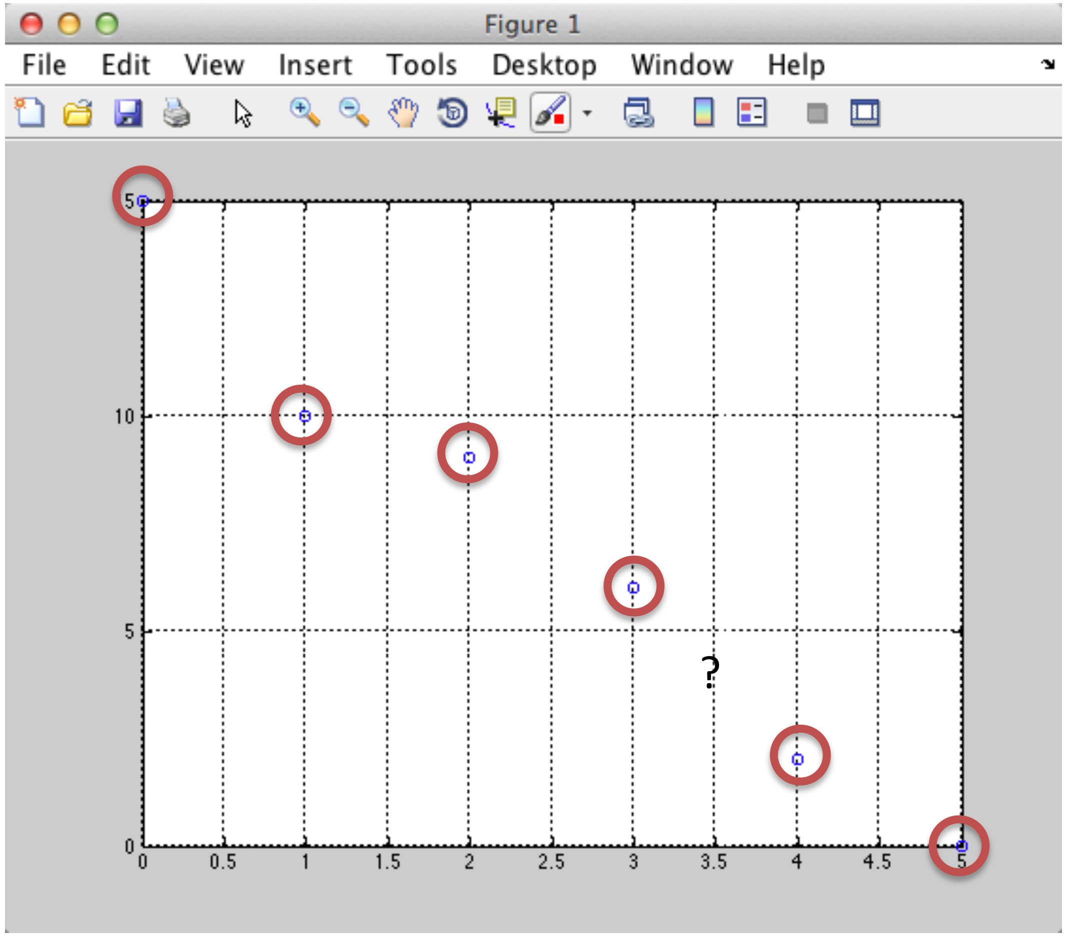
Interpolation

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., $x = 3.5$

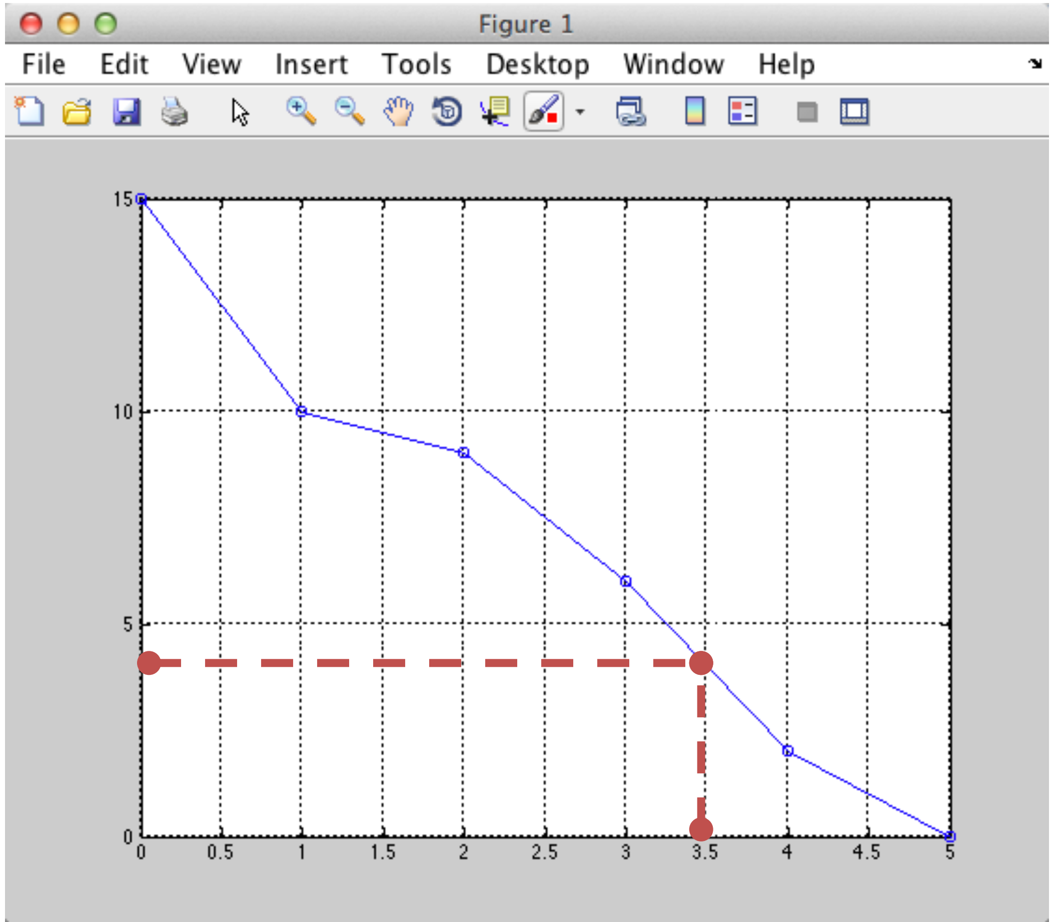
Interpolation

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
```

MATLAB gives us the answer 4.
From the plot we see this is a good guess:



Interpolation

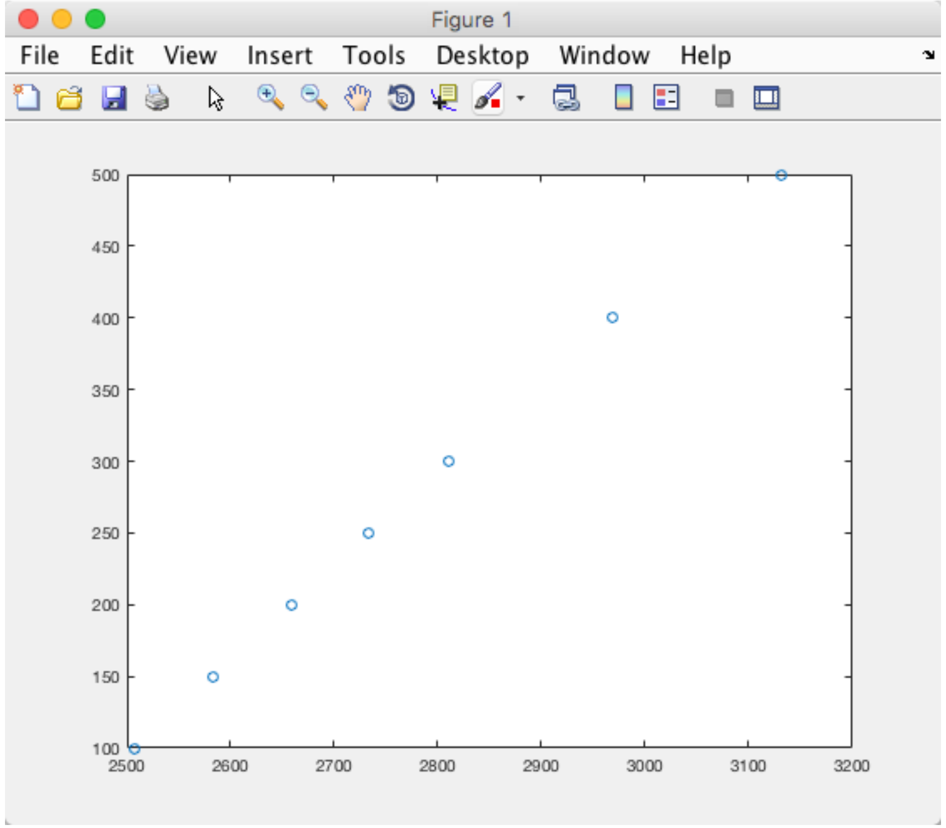
Given the following data:

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?

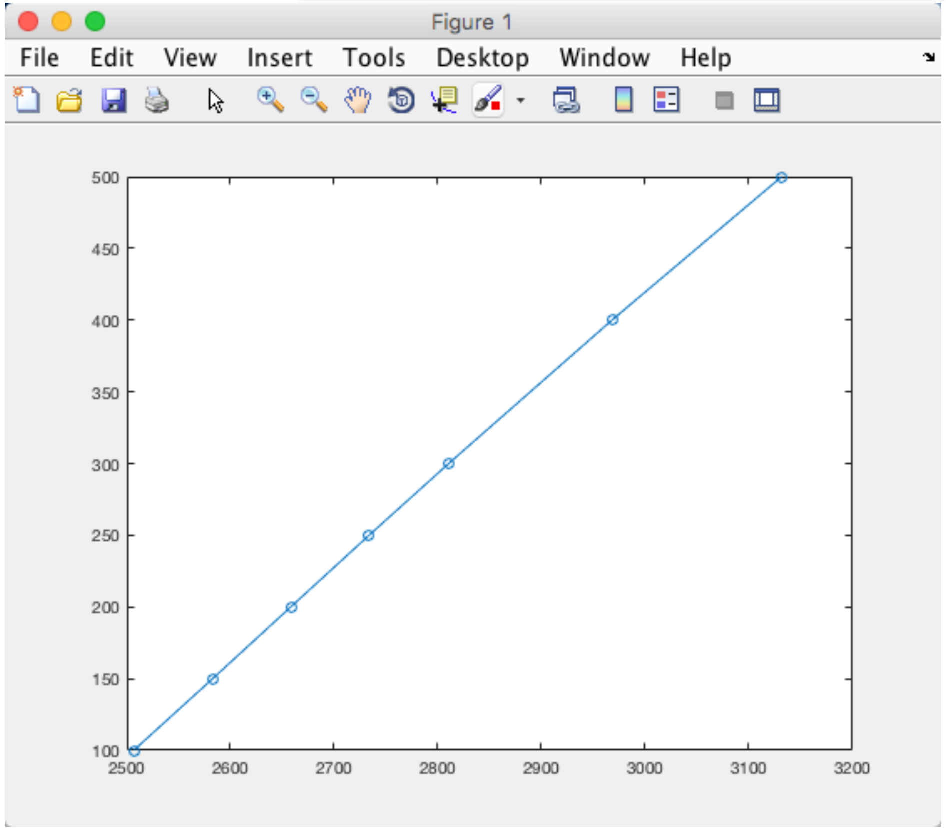
```
clear
clc

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)
```

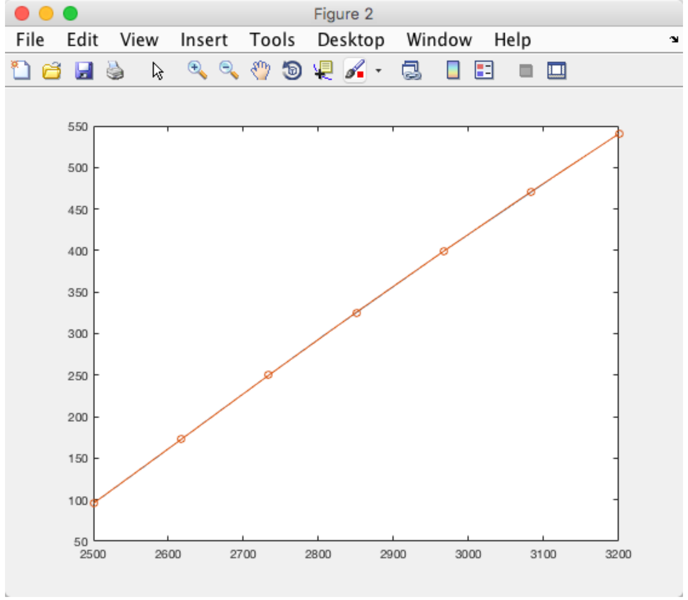
The interpolated value for u=2680.78 KJ/kg is:

```
ans =
  215.0000
```

i.e, for $u = 2680.76$ we get $T = 215$

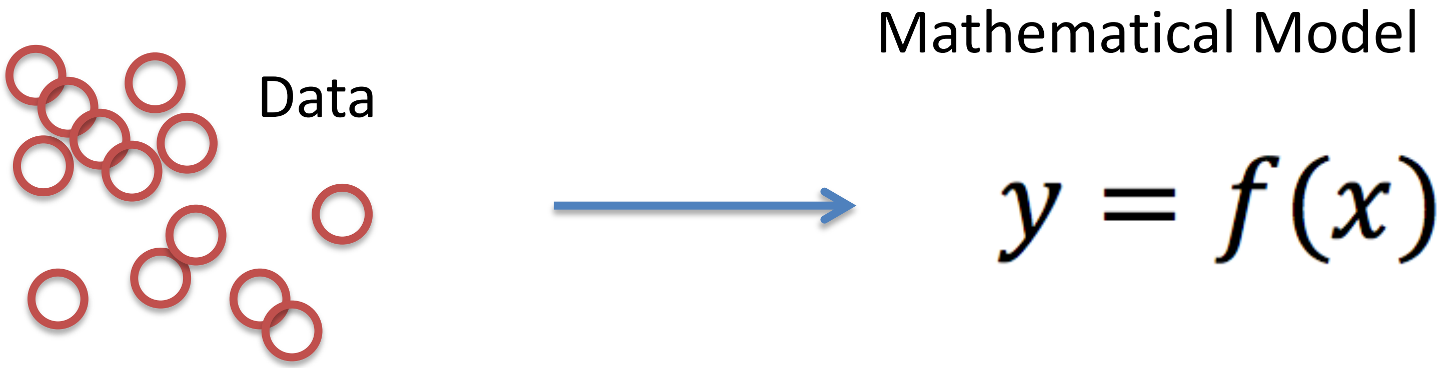
```
%Spline
new_u = linspace(2500,3200,length(u));
new_T = interp1(u, T, new_u, 'spline');
figure(2)
plot(u,T, new_u, new_T, '-o')
```

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



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 Y best in a least-squares sense.

Regression Models

Linear Regression:

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

Polynomial Regression:

$$y(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + \ a_n$$

1.order (linear):

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

2.order:

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

etc.

Linear Regression

Given the following data:

Temperature, T [^oC]	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

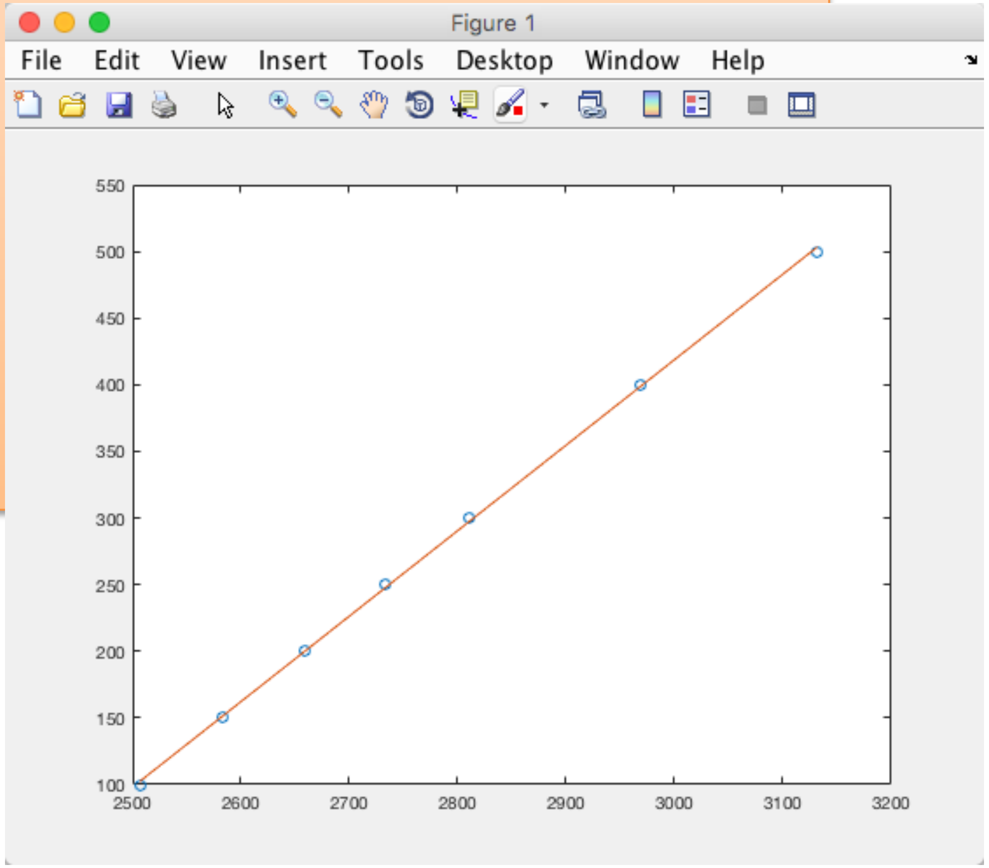
Plot u versus T.

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

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

Polynomial Regression

Given the following data:

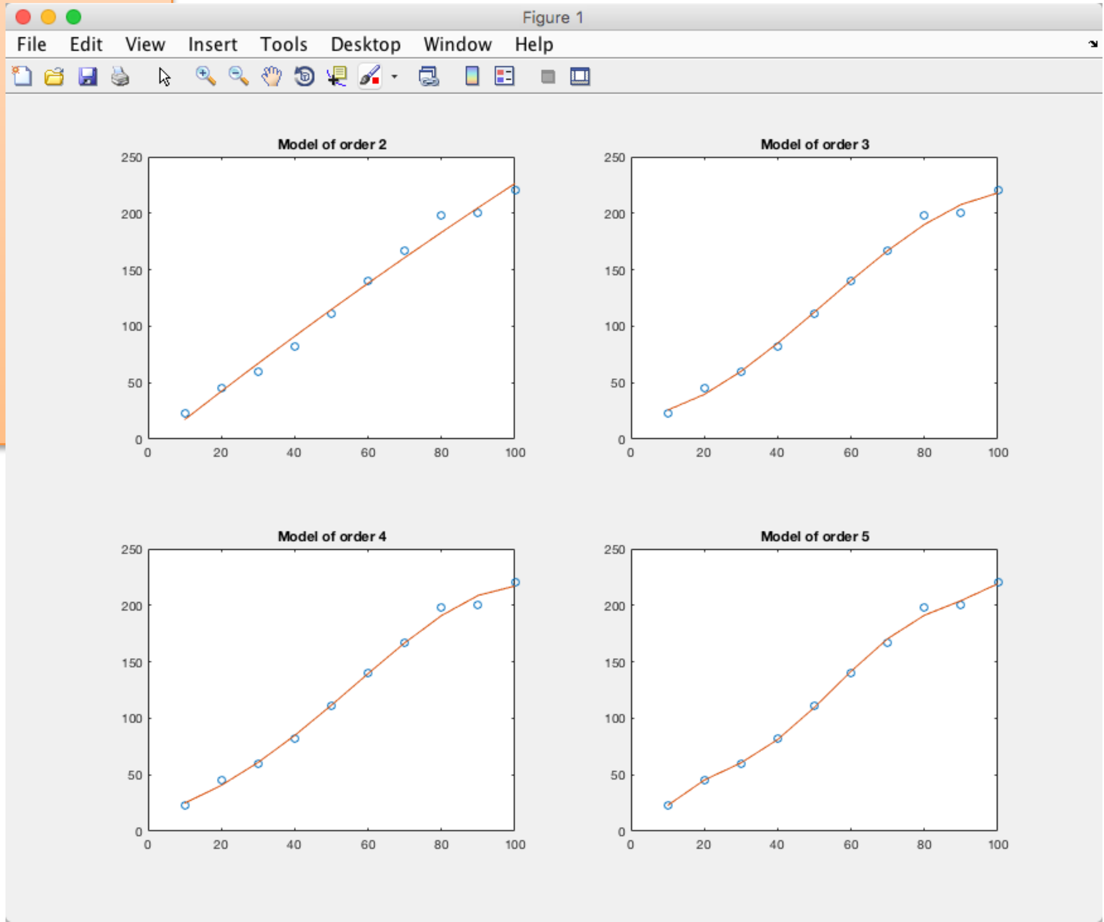
x	y
10	23
20	45
30	60
40	82
50	111
60	140
70	167
80	198
90	200
100	220

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 different line styles so the user can easily see the difference.

```
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')
legend('real data', 'linear model', 'quadratic model', 'cubic model')
```

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:

$$p_1 = a_0x + a_1 = 5.4x - 5.8$$

2. order model:

$$p_2 = a_0x^2 + a_1x + a_2 = 1.5x^2 - 2.6x + 1.1$$

3. order model:

$$p_3 = a_0x^3 + a_1x^2 + a_2x + a_3 = 0.5x^3 - 2.7x^2 + 4.9x - 0.1$$

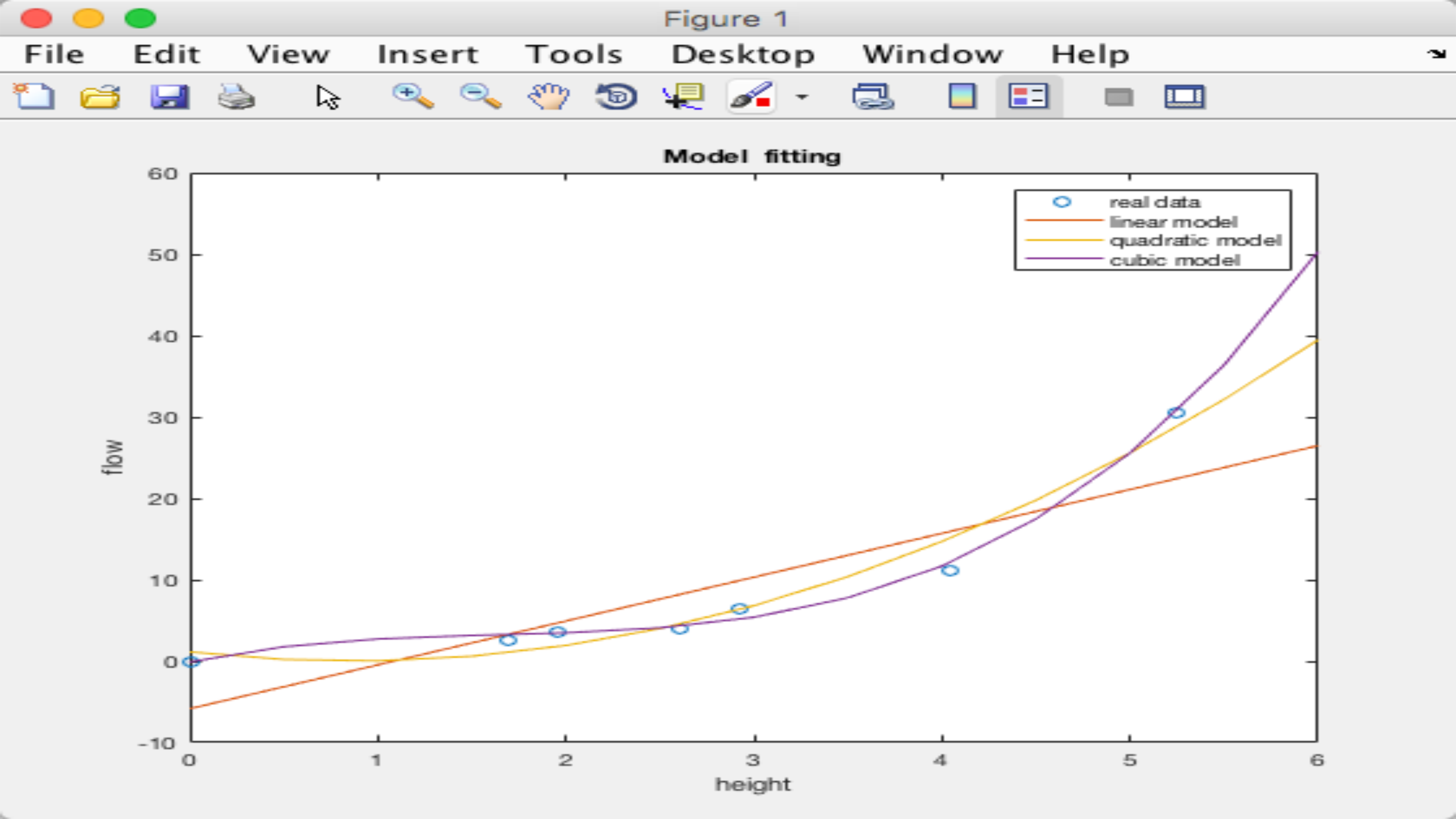


Figure 1
File Edit View Insert Tools Desktop Window Help
Model fitting
real data
linear model
quadratic model
cubic model
flow
height
60
50
40
30
20
10
0
-10
0
1
2
3
4
5
6

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog