26-Applications

text: Chapter 8, 10

ECEGR 101

Engineering Problem Solving with Matlab

Professor Henry Louie

Overview

- Polynomials
- Curve Fitting
- Interpolation
- Numerical Analyses

Polynomials

- Polynomials have the form

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

  - x: variable
  - a: coefficients
  - n: order of the polynomial

Polynomials

- MATLAB can represent polynomials, as long as the order is known
- Example:
  - $8x + 5$ p=[8 5]
  - $2x^2 - 4x + 10$ p=[2 -4 10]
  - $6x^2 - 150$ p=[6 0 -150]
  - x^3 p=[1 0 0 0]

Vector length = order + 1

Polynomials

- Polynomials can be evaluated using polyval
- Y = polyval(P,X) returns the value of a polynomial P evaluated at X. P is a vector of length N+1 whose elements are the coefficients of the polynomial in descending powers.

Example

- Evaluate $2x^2 - 4x + 10$ for $x = 7.25$

Example

- Evaluate $2x^2 - 4x + 10$ for $x = 7.25$

```
>> p=[2 -4 10]
p =
     2    -4    10
>> polyval(p,7.25)
ans =
   86.1250
```

Exercise

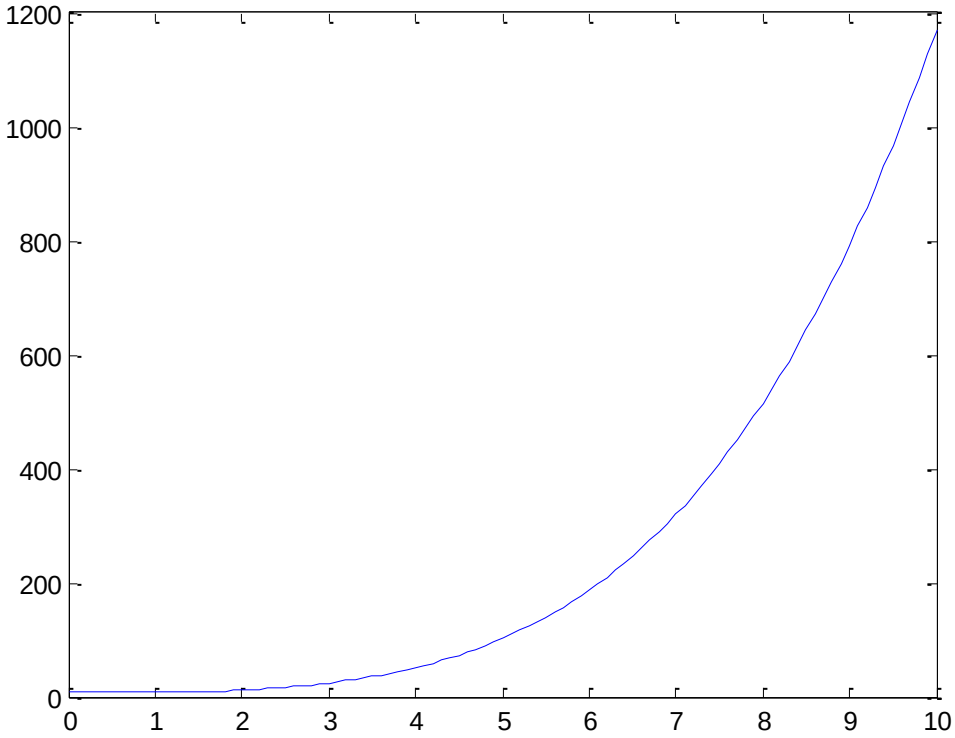
- Plot $0.10x^4 + 2x^2 - 4x + 10$ for x=0:.1:10 using polyval

Hint: use y = polyval(p,x), where x is a vector and y Is the polynomial evaluated at each x.

Exercise

- Plot $0.10x^4 + 2x^2 - 4x + 10$ for x=0:.1:10 using polyval

```
>> p=[.1 0 2 -4 10];
>> x=0:.1:10;
>> y=polyval(p,x);
>> plot(x,y)
```



Root Solving

- Roots of a polynomial can be found using roots(p) command
- Example
  - Let $f(x) = x^2 - 2x - 3$
  - Find x such that f(x) = 0 (i.e. find the roots of f(x))

```
>> p=[1 -2 -3]
p =
     1    -2    -3
>> r=roots(p)
r =
    3.0000
   -1.0000
```

Polynomial Solving

- When the roots are known, the polynomial can be found using poly
- Example:

```
>> r=[3 -1]
r =
     3    -1
>> p=poly(r)
p =
     1    -2    -3
```

$\longrightarrow$ $x^2 - 2x - 3$

Polynomial Arithmetic

- Polynomials p1 and p2 can be added, subtracted, multiplied and divided
  - Addition: p1 + p2 (may need to pad zeros if not of the same order)
  - Subtraction: p1 – p2 (may need to pad zeros if not of the same order)
  - Multiplication: conv(p1,p2) (repeated use of conv() also for multiplication of more than two polynomials
  - Division: [q,r] = deconv(p1,p2) (q is the quotient, r is the remainder)

Example

- Let $g(x) = 0.10x^4 + 2x^2 - 4x + 10$ and Let $f(x) = x^2 - 2x - 3$
- Find $h(x) = g(x) * f(x)$

```
>> g=[.1 0 2 -4 10];
>> f=[1 -2 -3];
>> h=conv(g,f)
h =
    0.1000   -0.2000    1.7000   -8.0000   12.0000   -8.0000  -30.0000
```

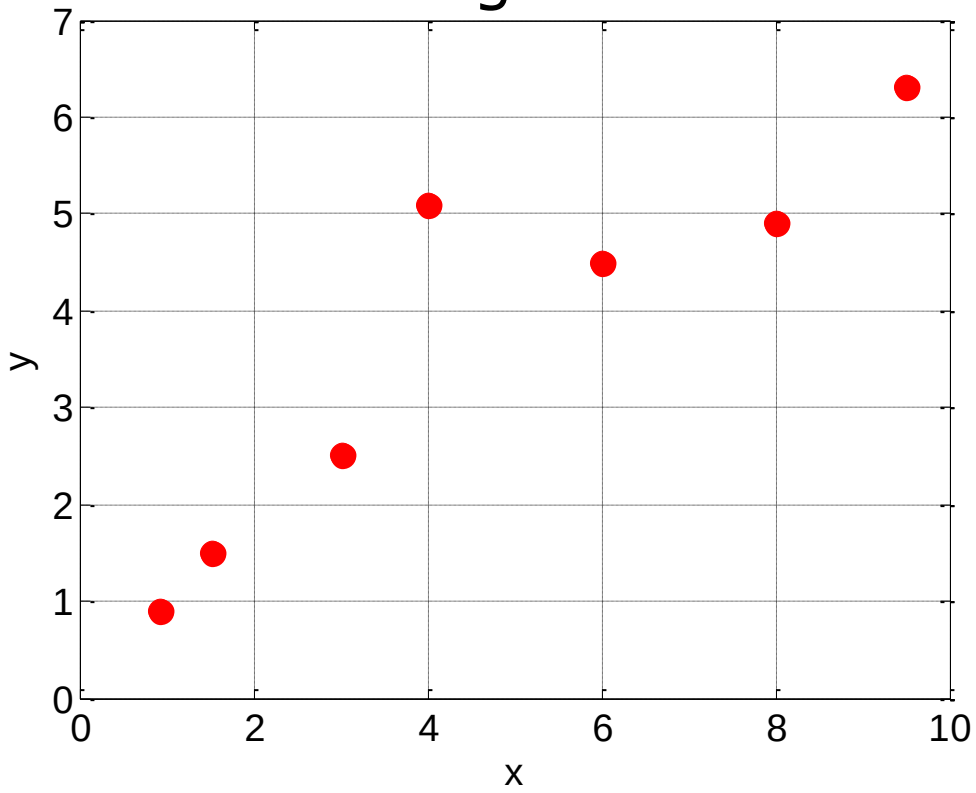
$$h(x) = 0.1x^6 - 0.2x^5 + 1.7x^4 - 8x^3 + 12x^2 - 8x - 30$$

Derivatives of Polynomials

- k = polyder(p)
  - p: vector of polynomial coefficients
  - k: a vector of polynomial derivative coefficients

Curve Fitting

- We often want to fit a polynomial to data
- Known as "Curve Fitting"



Curve Fitting

- Which order polynomial to use?
  - $y = b$
  - $y = mx + b$
  - $y = ax^2 + bx + c$
  - and so on…

If there are n data points, then
the order of the fit polynomial cannot exceed n-1

Curve Fitting

- p = polyfit(x,y,n)
  - x: vector of horizontal component of data
  - y: vector of vertical component of data
  - n: order of polynomial to fit
  - p: resulting coefficients

Example

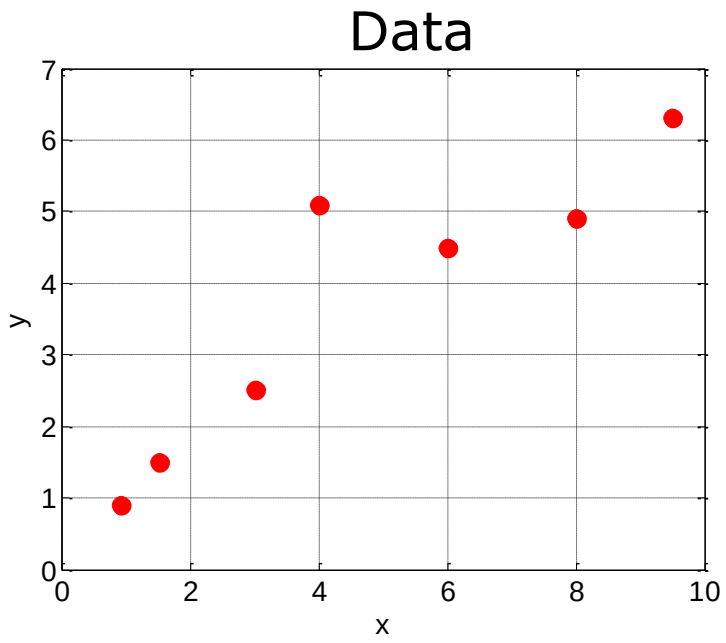
```
>> x=[.9 1.5 3 4 6 8 9.5];%%x data
>> y=[.9 1.5 2.5 5.1 4.5 4.9 6.3];%%y data
>> p=polyfit(x,y,1)

p =

    0.5688    0.9982
```

first order polynomial

y= 0.5688x + 0.9982



Example

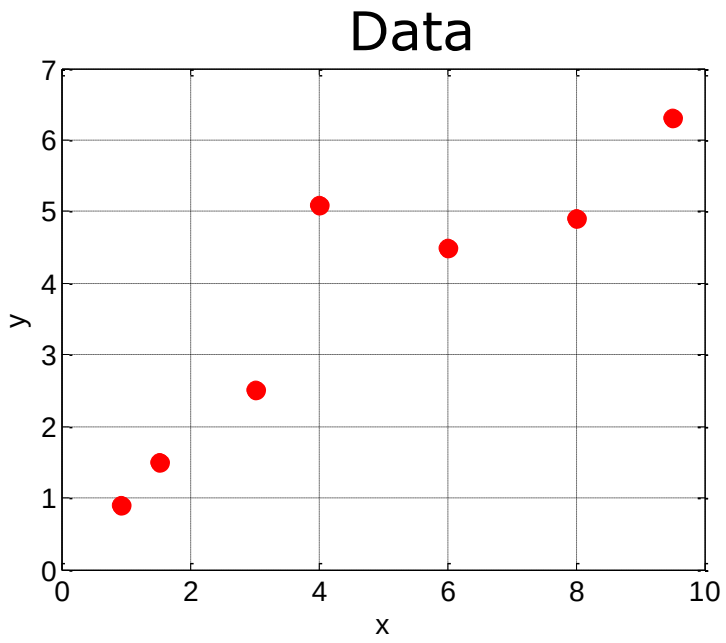
```
>> x=[.9 1.5 3 4 6 8 9.5];%%x data
>> y=[.9 1.5 2.5 5.1 4.5 4.9 6.3];%%y data
>> p=polyfit(x,y,2)

p =

   -0.0617    1.2030   -0.0580
```

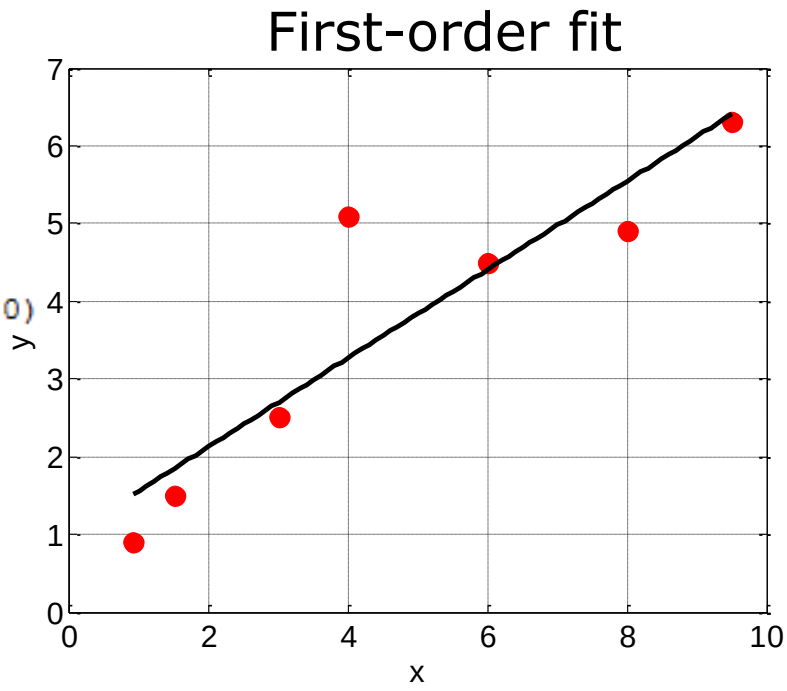
second order polynomial

$y = -0.0617x^2 + 1.2030x - 0.058$



Example

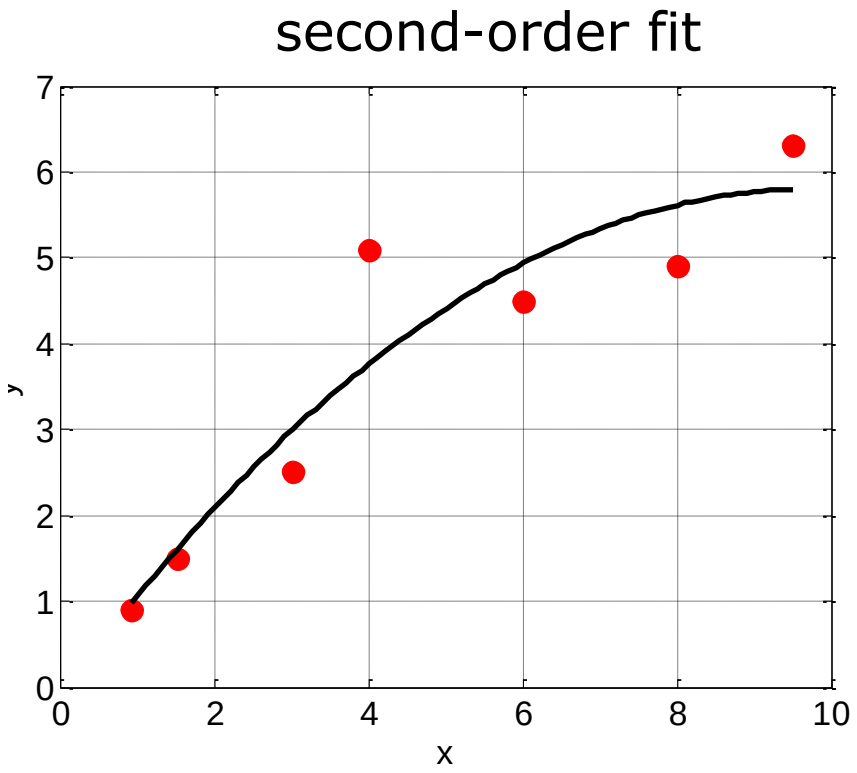
```
x=[.9 1.5 3 4 6 8 9.5];%%x data
y=[.9 1.5 2.5 5.1 4.5 4.9 6.3];%%y data
p=polyfit(x,y,1)
figure
plot(x,y,'ro','markerfacecolor','red','markersize',10)
grid on
xlabel('x','fontsize',14)
ylabel('y','fontsize',14)
set(gca,'fontsize',14)
hold on
xp=[min(x):.1:max(x)];
yp=polyval(p,xp);
plot(xp,yp,'black','linewidth',2.5)
```



Use polyval to evaluate fitted polynomial

Example

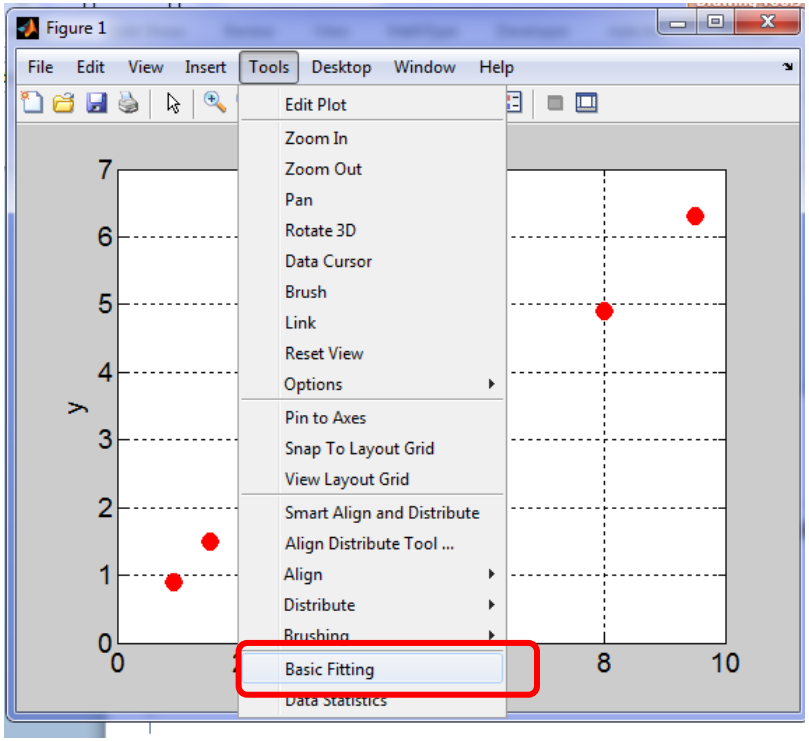
```
clear all
x=[.9 1.5 3 4 6 8 9.5];%%x data
y=[.9 1.5 2.5 5.1 4.5 4.9 6.3];%%y data
p=polyfit(x,y,2)
figure
plot(x,y,'ro','markerfacecolor','red','markersize',10)
grid on
xlabel('x','fontsize',14)
ylabel('y','fontsize',14)
set(gca,'fontsize',14)
hold on
xp=[min(x):.1:max(x)];
yp=polyval(p,xp);
plot(xp,yp,'black','linewidth',2.5)
```

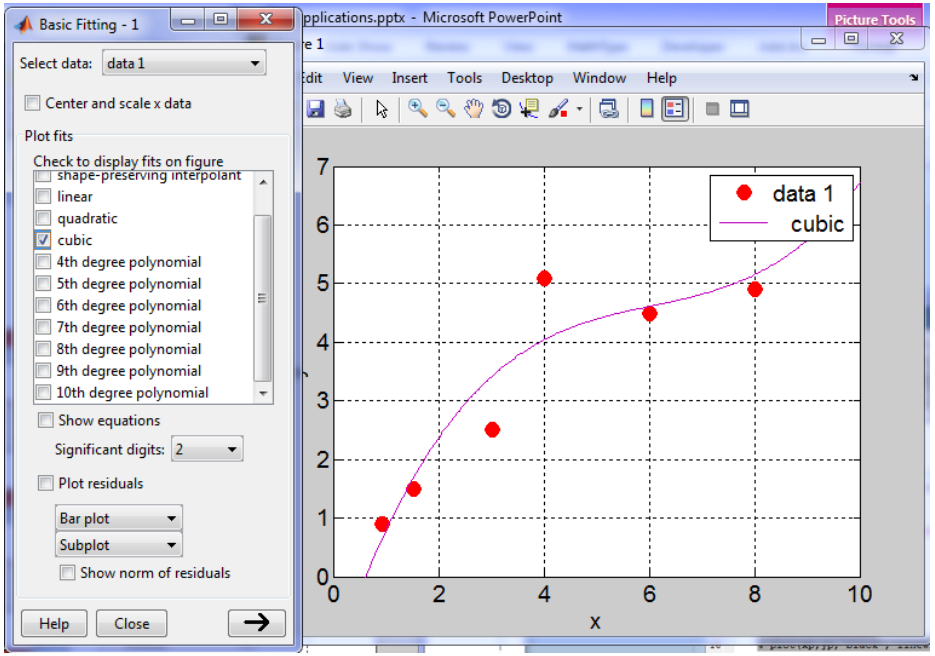


Curve Fitting Notes

- MATLAB fits the polynomials based on "Least Squares"
- Other (non-polynomial) fits are possible
  - See "fit" (requires curve-fitting toolbox)
  - See text
- GUI Basic Fit
  - Create plot of data
  - In Figure window Tools>> Basic Fitting

Curve Fitting



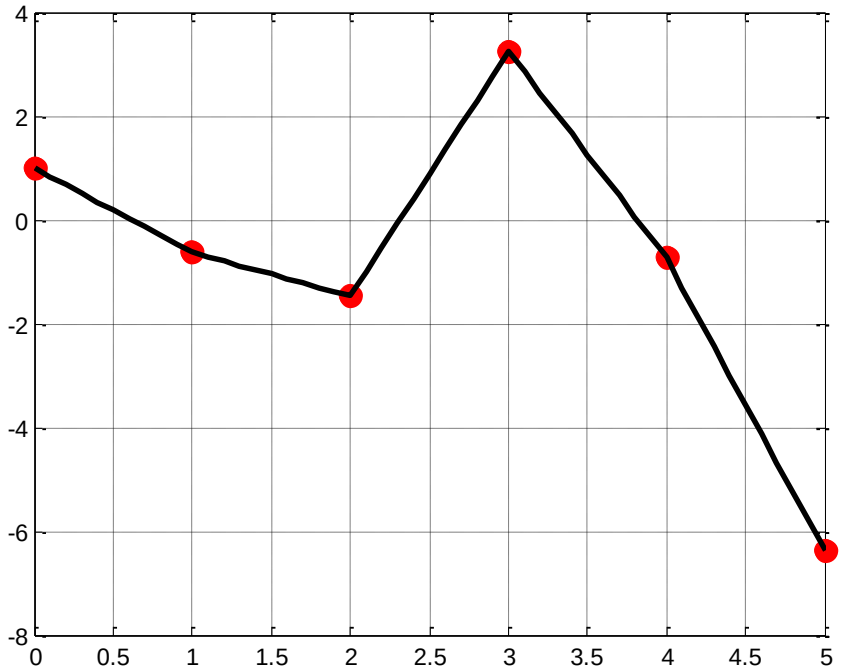


Interpolation

- Interpolation is the estimation of values between data points
- yi = interp1(x,y,xi,'method')
  - x: vector of horizontal component of data
  - y: vector of vertical component of data
  - xi: vector of points whose values are to be interpolotated
  - method: interpolation method (optional)

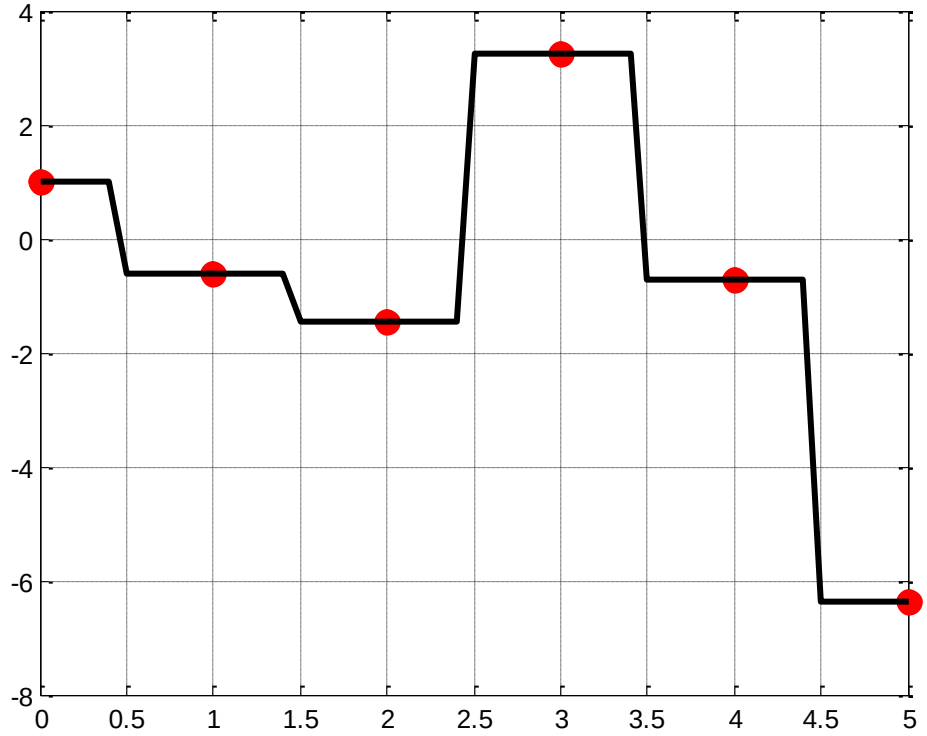
interp1 (the last character is the number 1)

Example ('linear' method)



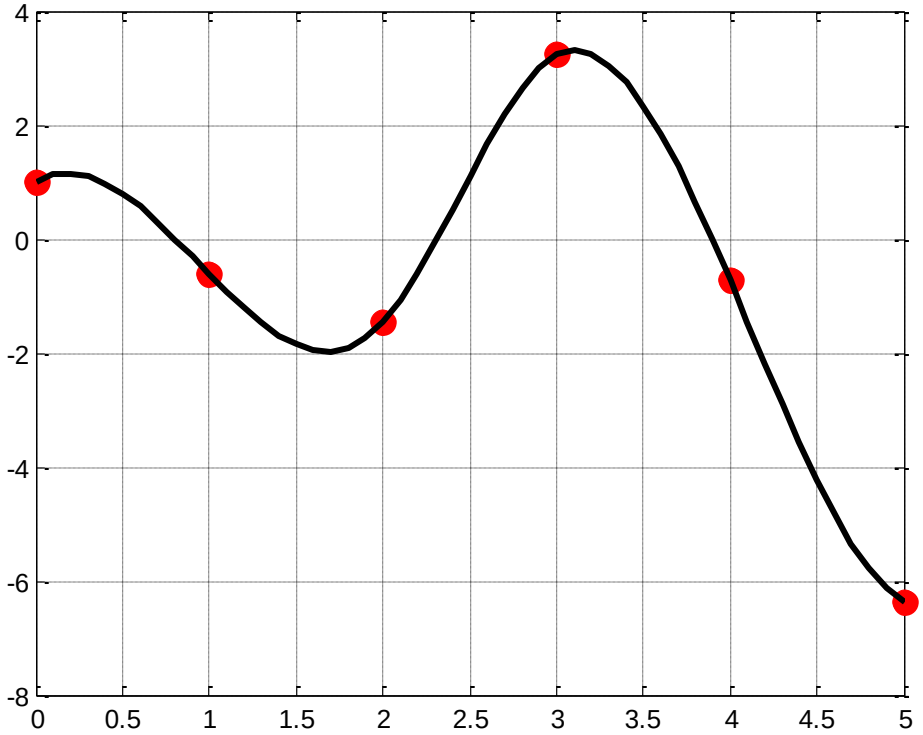

```
x=[0:1:5];%%x data (1 resolution)
y=[1 -.6242 -1.4707 3.24 -.7366 -6.3717];%%y data
xi=[0:.1:5]%%interpolated points (0.1 resolution)
yi=interp1(x,y,xi,'linear')
plot(x,y,'ro','markerfacecolor','red','markersize',10);hold on
plot(xi,yi,'black','linewidth',2.5);grid on
```

Example ('nearest' method')




```
x=[0:1:5];%%x data (1 resolution)
y=[1 -.6242 -1.4707 3.24 -.7366 -6.3717];%%y data
xi=[0:.1:5]%%interpolated points (0.1 resolution)
yi=interp1(x,y,xi,'nearest')
plot(x,y,'ro','markerfacecolor','red','markersize',10);hold on
plot(xi,yi,'black','linewidth',2.5);grid on
```

Example ('spline' method)




```
x=[0:1:5];%%x data (1 resolution)
y=[1 -.6242 -1.4707 3.24 -.7366 -6.3717];%%y data
xi=[0:.1:5]%%interpolated points (0.1 resolution)
yi=interp1(x,y,xi,'spline')
plot(x,y,'ro','markerfacecolor','red','markersize',10);hold on
plot(xi,yi,'black','linewidth',2.5);grid on
```

Numerical Analysis

- MATLAB can be used to find zero crossings, minimum and maximum, integrate functions, solve differential equations and more!
- Covered in the text, but beyond the scope of this courses (very useful in other courses!)

Useful Functions

- X=fzero(function,x0)
  - Solves the 'function' f(x) = 0 near an x value of x=0
- X=fminbnd(function,x1,x2)
  - Finds the minimum value of the function between x1 and x2
- Q= quad(function,a,b)
  - Numerically integrates the function between a and b
- Q = trapz(x,y)
  - Numerically integrates a function whose data are given as points in x and y

Useful Functions

- Differential equations can be solved using different “solvers”
  - ode45
  - ode23
  - ode113
  - And many more
- See text for more details

Final Words

- MATLAB has many more capabilities than we discussed
- Advanced features require optional "toolboxes"
- See www.mathworks.com for more help, as well as http://www.mathworks.com/matlabcentral/fileexchange/ for user generated codes