PROGRAMMING WITH MATLAB

WEEK 7

CURVE FITTING

CURVE FITTING

- Linear, power and exponential functions to describe the data.
- In the linear function $y = mx + b$, m gives the slope of the line, and b gives the intersection point with the vertical axis.
- This function gives a straight line when plotted on rectilinear axes.
- The power function $y = bx^m$ gives a straight line when plotted on log-log axes.
- The exponential function $y = b10^{mx}$ and its equivalent form $y = be^{mx}$ give a straight line when plotted on a semilog plot.

CURVE FITTING

- Many functions can be fairly well defined by high-order polynomials.
- MATLAB specifies a polynomial with the vector of coefficients
- If the p vector defines a polynomial:

$a_3x^3 + a_2x^2 + a_1x^1 + a_0$

$p(1) = a_3, p(2) = a_2, p(3) = a_1, p(4) = a_0$

$p = [1\ 0\ -3\ 5]$, represents $x^3 - 3x + 5$ polynomial

CURVE FITTING

- To find the roots of a polynomial:

Syntax: k = roots(p)

- It is possible to construct a polynomial from its roots.

Syntax : p = poly(k)

```
>> p=[1 0 3 5]
p =
     1     0     3     5
>> k = roots(p)
k =
   0.5771 + 1.9998i
   0.5771 - 1.9998i
  -1.1542 + 0.0000i
>> p = poly(k)
p =
    1.0000    0.0000    3.0000    5.0000
```

CURVE FITTING

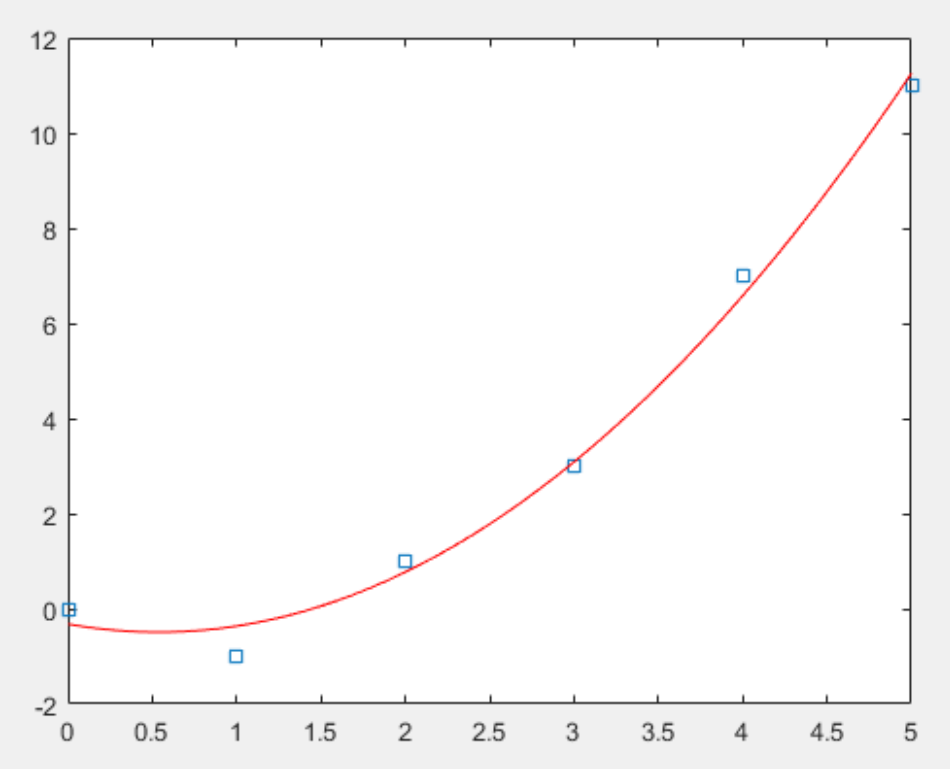
- MATLAB makes polynomial fitting to data very easy:

Syntax: p = polyfit(x,y, n)

```
>> x = [0 1 2 3 4 5];
>> y = [0 -1 1 3 7 11];
>> p = polyfit(x,y,2)  % finds a polynomial of the second order that
%best fits to the points (0,0), (1,-1), (2,1), (3,3), (4, 7), (5, 11)
 p =
    0.5893   -0.6321   -0.3214
```

- To calculate the polynomial at many points:

```
>> y = polyval(p, [1 2 3])
 y =
    9.0000   19.0000   41.0000
```



CURVE FITTING

To calculate $y = x^2 + 3$ for x = -3: 0.1: 3

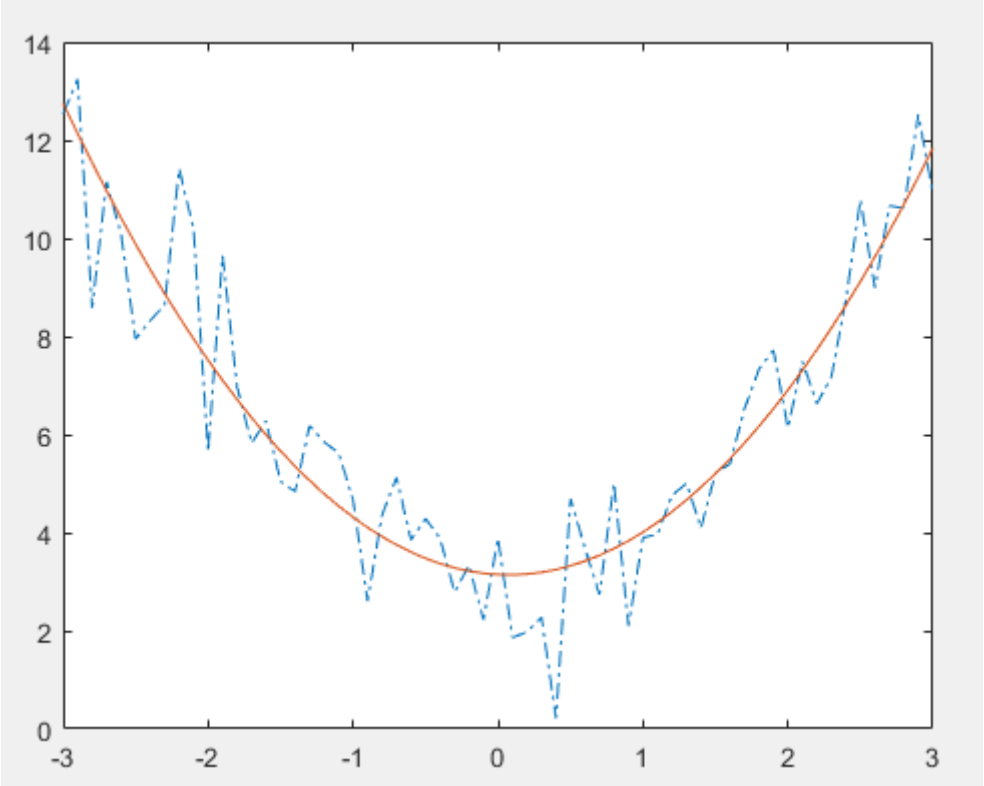
```
>> x = -3: 0.1: 3;
>> y = x.^2+3;
```

Add random noise to these samples (use randn)

```
>> y = y+randn(size(y));
```

Find a second-degree polynomial that fits the noisy data

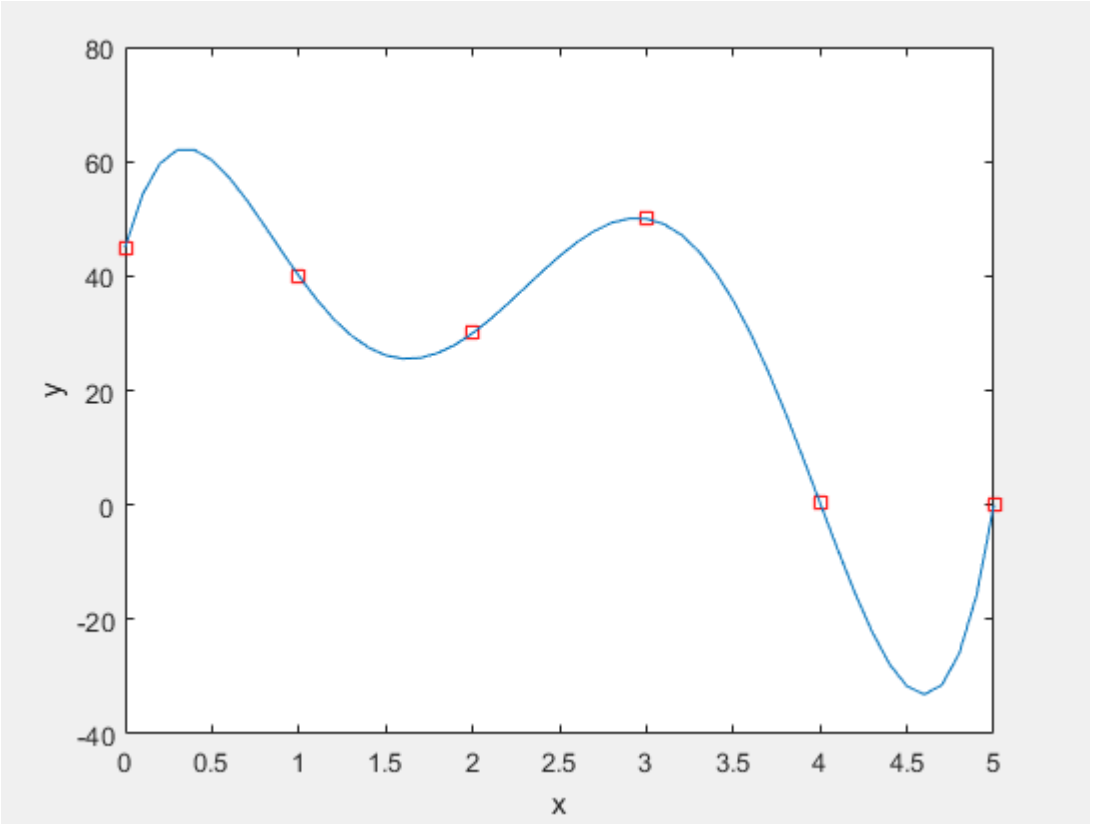
```
>> p = polyfit(x,y,2)
p =
    1.0166   -0.1520    3.1457
>> plot(x,y,'-.');
>> hold on
>> plot(x,polyval(p,x,'c'))
```



CURVE FITTING

Be careful when using high degree polynomials

```
>> x = [0 1 2 3 4 5];
>> y = [45 40 30 50 0.3 0];
>> p = polyfit(x,y,5);
>> plot(x,y,'sr');
>> hold on
>> plot((0:0.1:5),polyval(p,(0:0.1:5),'c'))
>> xlabel('x');ylabel('y')
```



CURVE FITTING

Goodness-of-Fit:

$$s = \sum_{i=1}^{N} (y - \bar{y})^2$$

$$R^2 = 1 - \frac{u}{s}$$

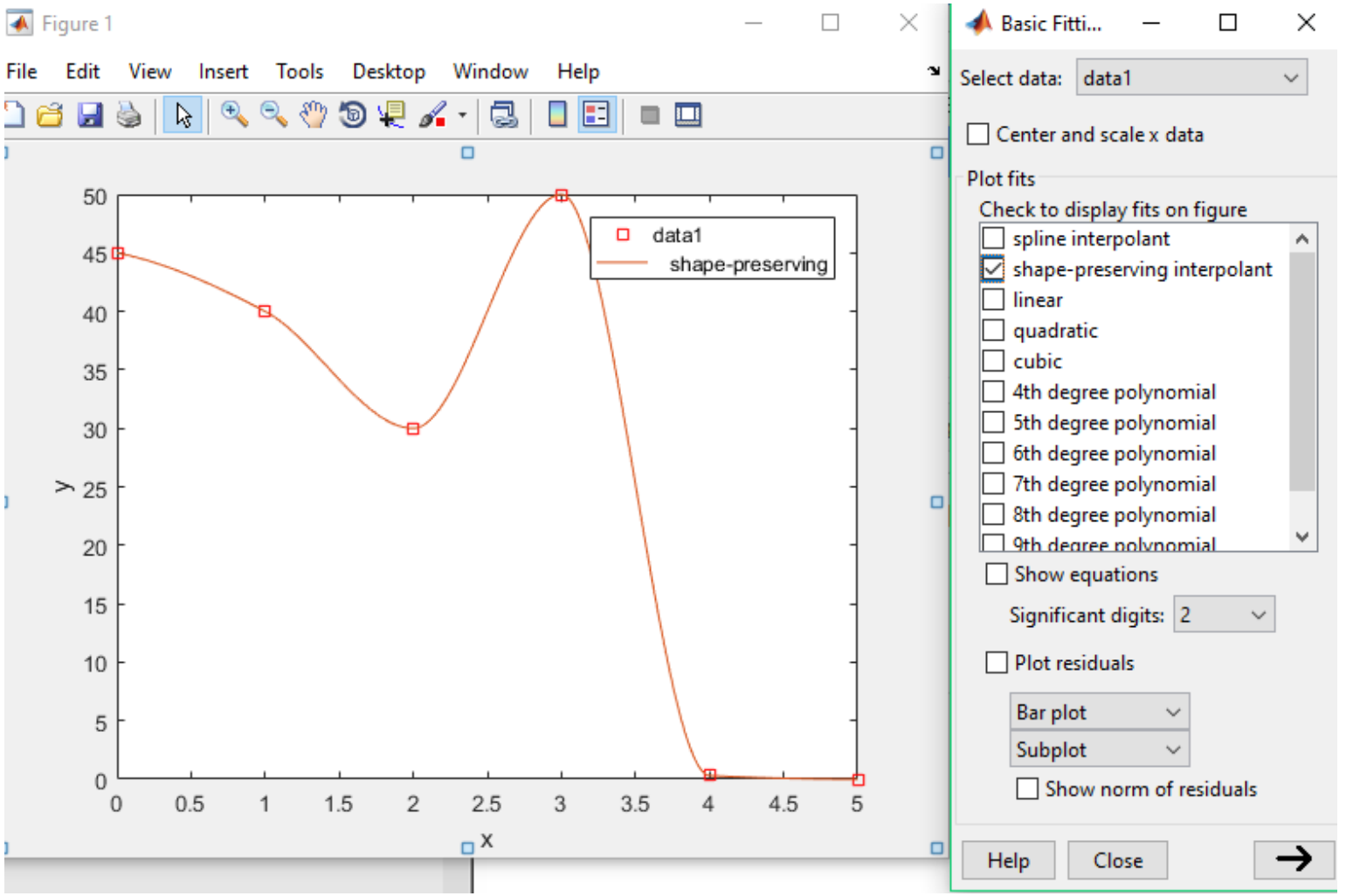
R-squared is a statistical measure of how close the data are to the fitted regression line. It is also known as the coefficient of determination, or the coefficient of multiple determination for multiple regression.

In general, the higher the R-squared, the better the model fits your data.

CURVE FITTING

Fitting interface:

MATLAB supports curve fitting through the Basic Fitting interface



ROOT FINDING

- Many real world problems require us to solve an equation such as f (x) = 0
- fzero : tries to find a point x where f(x) = 0

Syntax: x = fzero(fun,x0)

```
>> fun = @cos; % function
x0 = 1; % initial point
x = fzero(fun,x0)
x =
    1.5708
```

ROOT FINDING

- fminbnd: minimize a function in a bounded interval.

Syntax: x = fminbnd(fun,x1,x2)

```
>> fun = @cos;
x1 = 0;
x2 = 2*pi;
x = fminbnd(fun,x1,x2)
x =
    3.1416
```

ROOT FINDING

Syntax: x = fminsearch(fun,x0). starts at the point x0 and attempts to find a local minimum x of the function described in fun

```
>> fun = @(x)5*(x(2) - x(1)^1.5)^2 + (x(1) + 1)^2;

x0 = [-0.5,1];

x = fminsearch(fun,x0)

x =

   -0.6812    0.1000
```

OPTIMIZATION TOOLBOX

- If you are familiar with the optimization methods, you can use this toolbox.
- It is useful for larger scale and more structural optimization problems

linprog : Linear programming solver

Syntax: x = linprog(f,A,b) solves min f'*x such that A*x ≤ b.

quadprog : Quadratic programming.

Syntax: x = quadprog(H,f) returns a vector x that minimizes 1/2*x'*H*x + f'*x

OPTIMIZATION TOOLBOX

fminunc : Find minimum of unconstrained multivariable function.

Syntax: x = fminunc(fun,x0) starts at the point x0 and attempts to find a local minimum x of the function described in fun.

fmincon: Find minimum of constrained nonlinear multivariable function.

Syntax: x = fmincon(fun,x0,A,b) starts at x0 and attempts to find a minimizer x of the function described in fun subject to the linear inequalities $A*x \leq b$.