Chapter Two

CURVE FITTING

Curve fitting is the process of finding equations to approximate straight lines and curves that best fit given sets of data. For example, for the data of Figure 2.1, we can use the equation of a straight line, that is:

$$y = mx + b$$

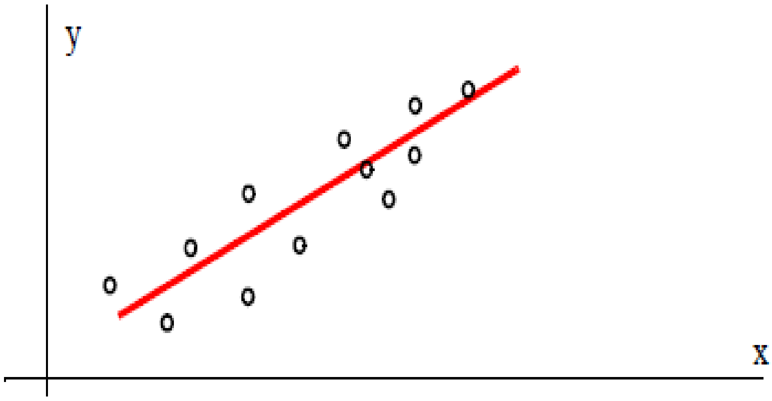


Figure 2.1: linear fitting.

For Figure 2.2, we can use the equation for the quadratic or parabolic curve of the form

$$y = ax^2 + bx + c$$

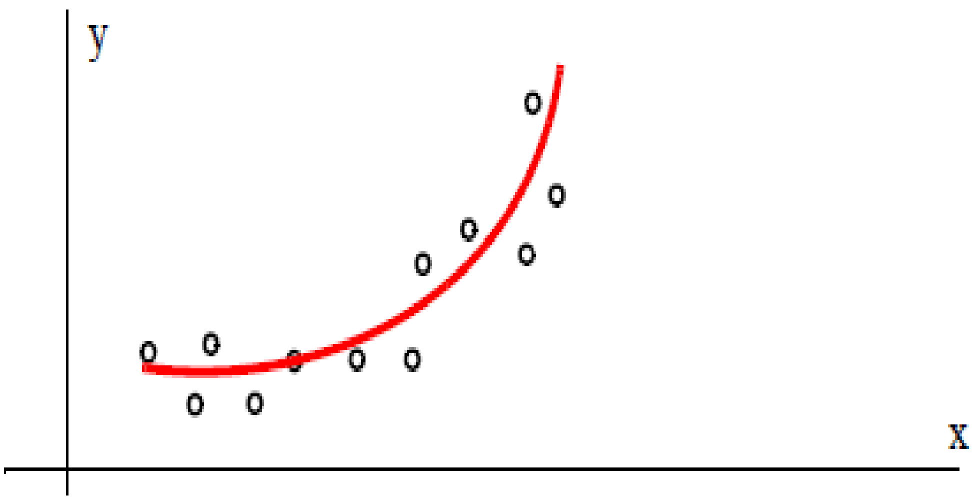


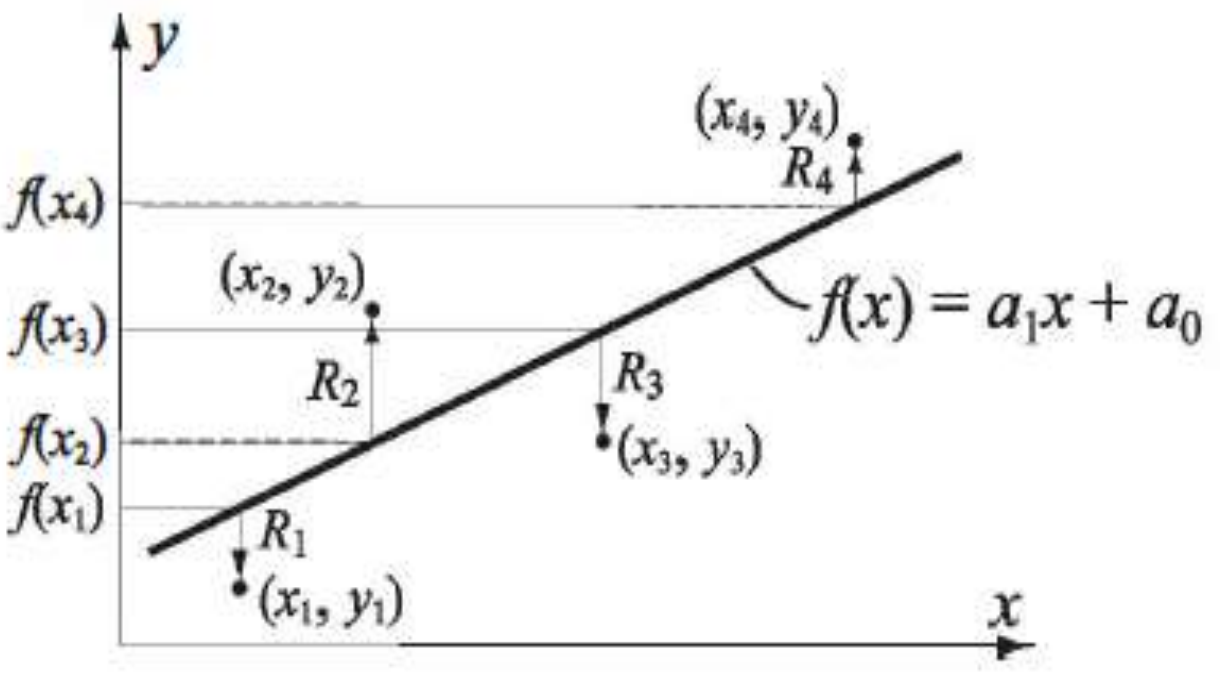
Figure 2.2: Polynomial fitting.

In finding the best line, we normally assume that the data, shown by the small circles in Figures 2.1 and 2.2, represent the independent variable, and our task is to find the dependent variable.

This process is called *regression*. Obviously, we can find more than one straight line or curve to fit a set of given data, but we interested in finding the most suitable.

Curve Fitting with Polynomials; The polyfit Function:

- When n points are given, it is possible to write a polynomial of degree less than n - 1 that does not necessarily pass through any of the points but that overall approximates the data.
- The most common method of finding the best fit to data points is the method of least squares.
- In this method, the coefficients of the polynomial are determined by minimizing the sum of the squares of the residuals at all the data points. The residual at each point is defined as the difference between the value of the polynomial and the value of the data.
- For example, consider the case of finding the equation of a straight line that best fits four data points as shown in Figure below



- The points are (x_1,y_1), (x_2, y_2), (x_3,y_3), and (x_4,y_4), and the polynomial of the first degree can be written as

$$f(x) = a_1x + a_0 .$$

- The residual, R_i, at each point is the difference between the value of the function at x_i and y_i,

$$R_i = f(x_i) - y_i .$$

- An equation for the sum of the squares of the residuals R_i of all the points is given by:

$$R = [f(x_1) - y_1]^2 + [f(x_2) - y_2]^2 + [f(x_3) - y_3]^2 + [f(x_4) - y_4]^2$$

or, after substituting the equation of the polynomial at each point, by:

$$R = [a_1x_1 + a_0 - y_1]^2 + [a_1x_2 + a_0 - y_2]^2 + [a_1x_3 + a_0 - y_3]^2 + [a_1x_4 + a_0 - y_4]^2$$

- At this stage R is a function of a_1 and a_0.
- The minimum of R can be determined by taking the partial derivative of R with respect to a_1 and a_0 (two equations) and equating them to zero:

$$\frac{\partial R}{\partial a_1} = 0 \quad \text{and} \quad \frac{\partial R}{\partial a_0} = 0$$

- This results in a system of two equations with two unknowns, a_1 and a_0.
- The solution of these equations gives the values of the coefficients of the polynomial that best fits the data. The same procedure can be followed with more points and higher-order polynomials.
- Curve fitting with polynomials is done in MATLAB with the polyfit function, which uses the least squares method. The basic form of the polyfit function is:

```
p = polyfit(x,y,n)
```

p is the vector of the coefficients of the polynomial that fits the data.

x is a vector with the horizontal coordinates of the data points (independent variable).
y is a vector with the vertical coordinates of the data points (dependent variable).
n is the degree of the polynomial.

Remarks:

- For the same set of m points, the **polyfit** function can be used to fit polynomials of any order up to m – 1.

- The polynomial passes through all the points if n = m - 1.
- It should be pointed out here that a polynomial that passes through all the points, or polynomials with higher order, does not necessarily give a better fit overall. High-order polynomials can deviate significantly between the data points.

Example1/ A set of seven points is given by (0.9, 0.9), (1.5, 1.5), (3, 2.5), (4, 5.1), (6, 4.5), (8, 4.9), and (9.5, 6.3). Find the best fit to it

`x=[0.9 1.5 3 4 6 8 9.5];` `y=[0.9 1.5 2.5 5.1 4.5 4.9 6.3];`	Create vectors x and y with the coordinates of the data points.
`p=polyfit(x,y,3)`	Create a vector p using the polyfit function.
`xp=0.9:0.1:9.5;`	Create a vector xp to be used for plotting the polynomial.
`yp=polyval(p,xp)`	Create a vector yp with values of the polynomial at each xp.
`plot(x,y,'o',xp,yp)`	A plot of the seven points and the polynomial.
`xlabel('x'); ylabel('y')`	

```
p =
    0.0220    -0.4005    2.6138    -1.4158
```

This means that the polynomial of the third degree in Figure 2-3 has the form:

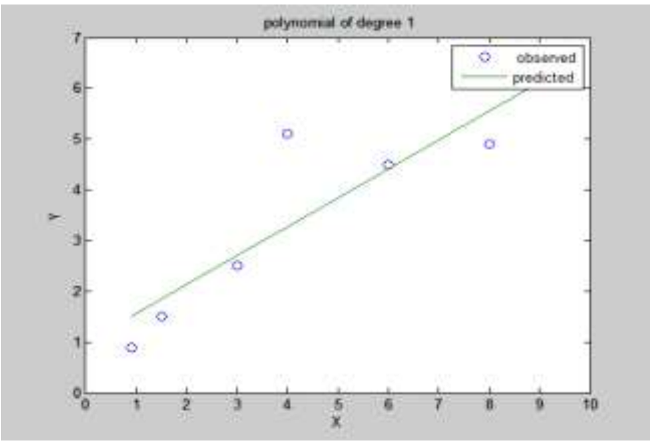
$$0.022x^3 - 0.4005x^2 + 2.6138x - 1.4148.$$

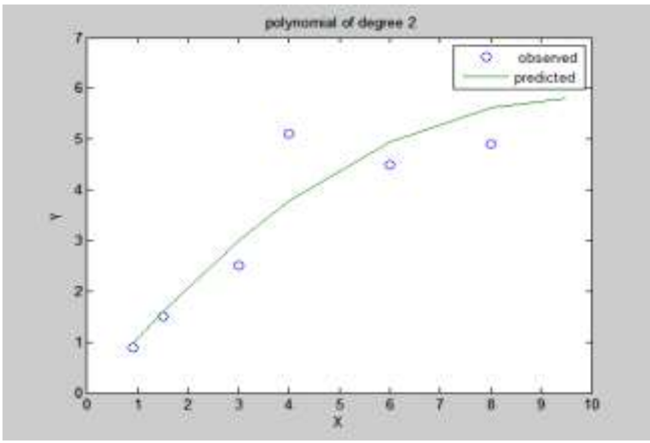
Note that the number of coefficients p equals n+1.

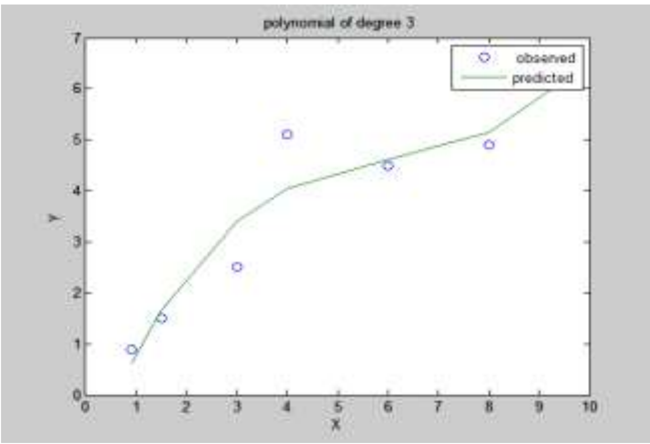
The main important statistical factor that measures the goodness of fitting is given by the root mean square error (RMSE):

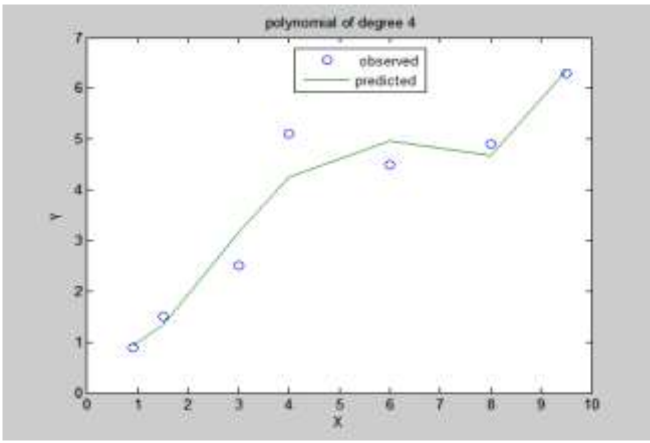
RMSE = sqrt (mean((predicted - observation)^2)

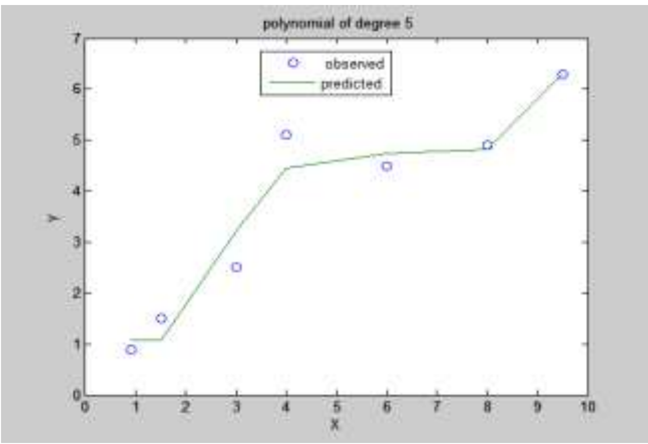
n	1	2	3	4	5	6	7	8
Coef.	0.5688 0.9982	-0.0617 -1.2030 0.0580	0.0220 -0.4005 2.6138 -1.4158	0.0101 -0.1896 1.0567 -1.0693 1.1627	-0.0026 0.0785 -0.8368 3.7762 -5.9354 3.9275	-0.0055 0.1617 -1.7906 9.3934 -23.9489 28.4158 -11.0411	-0.0014 0.0403 -0.4409 2.2598 -5.4721 5.4127 0 -0.7382	**-0.0004** **0.0124** **-0.1381** **0.7263** **1.8209** **-1.8296** **0** **0** **0.3996**
RMSE	0.7849	0.6541	0.5549	0.4569	0.4212	3.4429e-12	2.6588e-12	**9.1448e-12**

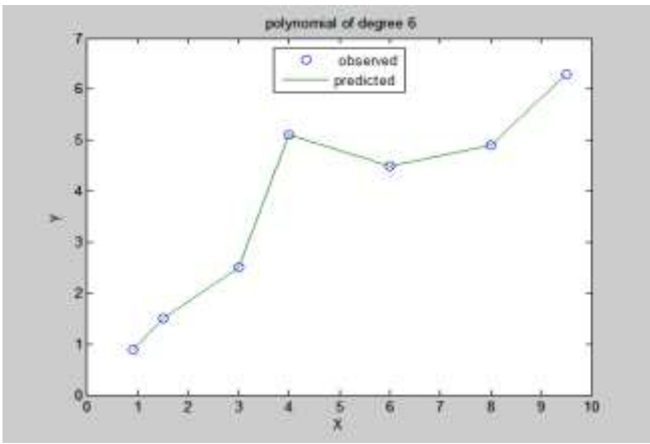


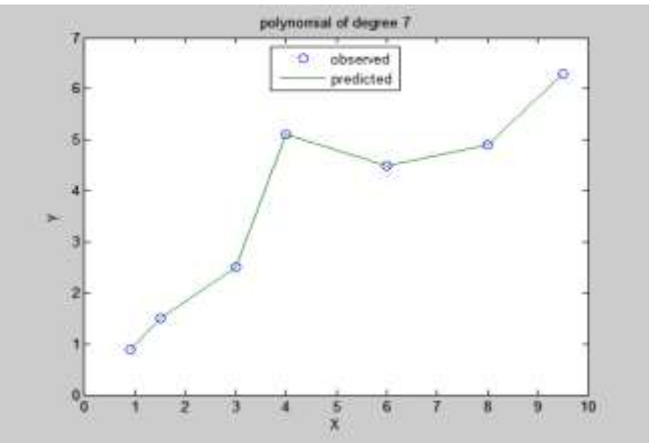


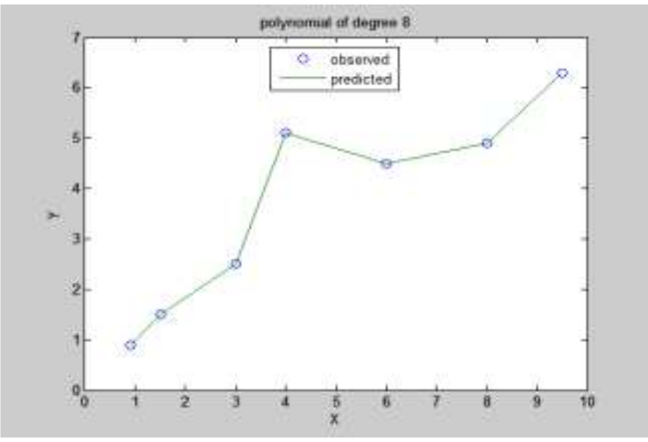












Example 2:

Fit Polynomial to Trigonometric Function

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

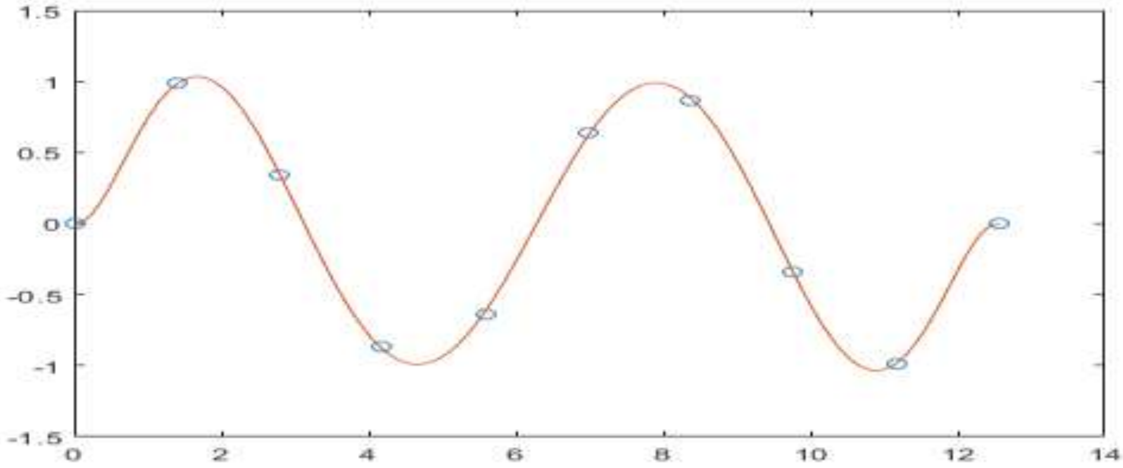
```
x = linspace(0,4*pi,10);
y = sin(x);
```

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
```



Example3

```
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')
```