Chapter Three
Polynomials

Polynomials are functions that have the form:

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

The coefficients $a_n, a_{n-1}, \ldots, a_1, a_0$ are real numbers, and *n,* which is a nonnegative integer, is the degree, or order, of the polynomial.

Examples of polynomials are:

$f(x) = 5x^5 + 6x^2 + 7x + 3$	polynomial of degree 5.
$f(x) = 2x^2 - 4x + 10$	polynomial of degree 2.
$f(x) = 11x - 5$	polynomial of degree 1.

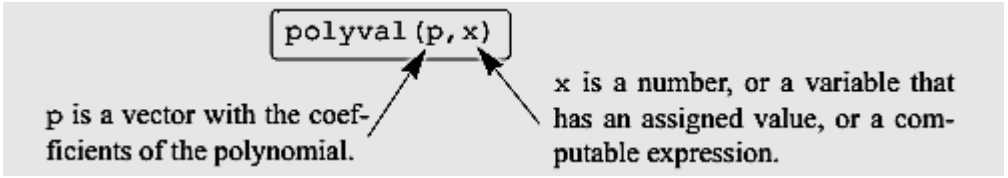
A constant (e.g.,) is a polynomial of degree 0.

n MATLAB, polynomials are represented by a row vector in which the elements are the coefficients $a_n, a_{n-1}, \ldots, a_1, a_0$. The first element is the coefficient of the *x* with the highest power. The vector has to include all the coefficients, including the ones that are equal to 0. For example:

Polynomial	MATLAB representation
$8x + 5$	p = [8 5]
$2x^2 - 4x + 10$	d = [2 –4 10]
$6x^2 - 150$, MATLAB form: $6x^2 + 0x - 150$	h = [6 0 –150]
$5x^5 + 6x^2 - 7x$, MATLAB form: $5x^5 + 0x^4 + 0x^3 + 6x^2 - 7x + 0$	c = [5 0 0 6 –7 0]

Value of a Polynomial

The value of a polynomial at a point *x* can be calculated with the function **`polyval`** which has the form:



x can also be a vector or a matrix. In such a case the polynomial is calculated for each element (element-by-element), and the answer is a vector, or a matrix, with the corresponding values of the polynomial.

Example: or the polynomial: $f(x) = x^5 - 12.1x^4 + 40.59x^3 - 17.015x^2 - 71.95x + 35.88$:
(*a*) Calculate .f(9)
(*b*) Plot the polynomial for . $-1.5 \le x \le 6.7$.

Solution

The problem is solved in the Command Window.

(a) The coefficients of the polynomials are assigned to vector p. The function **polyva**l is then used to calculate the value at $x = 9$.

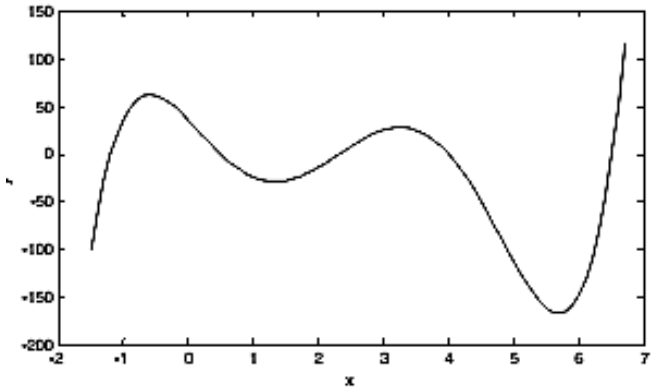
```
>> p = [1 -12.1 40.59 -17.015 -71.95 35.88];
>> polyval(p,9)
ans =
  7.2611e+003
```

(*b*) To plot the polynomial, a vector x is first defined with elements ranging from –1.5 to 6.7. Then a vector y is created with the values of the polynomial for every element of x. Finally, a plot of y vs. x is made.

```
>> x=-1.5:0.1:6.7;
>> y=polyval(p,x);
>> plot(x,y)
```

Calculating the value of the polynomial for each element of the vector x.

The plot created by MATLAB is presented below (axis labels were added with the Plot Editor).



Roots of a Polynomial

The roots of a polynomial are the values of the argument for which the value of the polynomial is equal to zero. For example, the roots of the polynomial $f(x) = x^2 - 2x - 3$ are the values of *x* for which $x^2 - 2x - 3 = 0$, which are x=-1 and x=3.

MATLAB has a function, called roots, that determines the root, or roots, of a polynomial. The form of the function is:

```
r = roots(p)
```

r is a column vector with the roots of the polynomial.

p is a row vector with the coefficients of the polynomial.

For example, the roots of the polynomial in Sample Problem $f(x) = x^5 - 12.1x^4 + 40.59x^3 - 17.015x^2 - 71.95x + 35.88$: can be determined by:

```
>> p= 1 -12.1 40.59 -17.015 -71.95 35.88];
>> r=roots(p)
r =
    6.5000
    4.0000
    2.3000
   -1.2000
    0.5000
```

When the roots are known, the polynomial can actually be written as:
$f(x) = (x+1.2)(x-0.5)(x-2.3)(x-4)(x-6.5)$

The roots command is very useful for finding the roots of a quadratic equation.

For example, to find the roots of $f(x) = 4x^2 + 10x - 8$, type:

```
>> roots([4 10 -8])
ans =
   -3.1375
    0.6375
```

When the roots of a polynomial are known, the poly command can be used for determining the coefficients of the polynomial. The form of the poly command is:

```
p = poly(r)
```

p is a row vector with the coefficients of the polynomial.

r is a vector (row or column) with the roots of the polynomial.

For example, the coefficients of the polynomial in Sample Problem $f(x) = x^5 - 12.1x^4 + 40.59x^3 - 17.015x^2 - 71.95x + 35.88$: can be obtained from the roots of the polynomial (see above) by

```
>> r=6.5 4 2.3 -1.2 0.5];
>> p=poly(r)
p =
    1.0000  -12.1000   40.5900  -17.0150  -71.9500   35.8800
```

Addition Polynomials:

Two polynomials can be added (or subtracted) by adding (subtracting) the vectors of the coefficients. If the polynomials are not of the same order (which means that the vectors of the coefficients are not of the same length), the shorter vector has to be modified to be of the same length as the longer vector by adding zeros (called padding) in front.

For example, the polynomials

$f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40$ and $f_2(x) = 3x^3 - 2x - 6$ can be added

```
>> p1=[3 15 0 -10 -3 15 -40];
>> p2=[3 0 -2 -6];
>> p=p1+[0 0 0 p2]
p =
     3    15     0    -7    -3    13   -46
```

Three 0s are added in front of p2, since the order of p1 is 6 and the order of p2 is 3.

Multiplication:

Two polynomials can be multiplied using the MATLAB built-in function **conv**, which has the form:

```
c = conv(a,b)
```

c is a vector of the coefficients of the polynomial that is the product of the multiplication.

a and b are the vectors of the coefficients of the polynomials that are being multiplied.

• The two polynomials do not have to be of the same order.
• Multiplication of three or more polynomials is done by using the **conv** function repeatedly.

For example, multiplication of the polynomials $f_1(x)$ and $f_2(x)$ above gives:

```
>> pm=conv(p1,p2)
pm =
     9    45    -6   -78   -99    65   -54   -12   -10   240
```

which means that the answer is:

$$9x^9 + 45x^8 - 6x^7 - 78x^6 - 99x^5 + 65x^4 - 54x^3 - 12x^2 - 10x + 240$$

Division:

A polynomial can be divided by another polynomial with the MATLAB built-in function **deconv**, which has the form:

```
[q,r] = deconv(u,v)
```

q is a vector with the coefficients of the quotient polynomial.
r is a vector with the coefficients of the remainder polynomial.

u is a vector with the coefficients of the numerator polynomial.
v is a vector with the coefficients of the denominator polynomial.

For example, dividing $2x^3 + 9x^2 + 7x - 6$ by $x + 3$ is done by:

```
>> u=[2 9 7 -6];
>> v=[1 3];
>> [a b]=deconv(u,v)
a =
     2     3    -2
b =
     0     0     0     0
```

The answer is: $2x^2 + 3x - 2$.

Remainder is zero.

An example of division that gives a remainder is $2x^6 - 13x^5 + 75x^3 + 2x^2 - 60$ divided by $x^2 - 5$:

```
>> w=[2 -13 0 75 2 0 -60];
>> z=[1 0 -5];
>> [g h]=deconv(w,z)
g =
     2   -13    10    10    52
h =
     0     0     0     0     0    50   200
```

The quotient is: $2x^4 - 13x^3 + 10x^2 + 10x + 52$.

The remainder is: $50x + 200$.

The answer is:

$$2x^4 - 13x^3 + 10x^2 + 10x + 52 + \frac{50x + 200}{x^2 - 5}.$$

Derivatives of Polynomials

The built-in function polyder can be used to calculate the derivative of a single polynomial, a product of two polynomials, or a quotient of two polynomials, as shown in the following three commands.

`k = polyder(p)`	Derivative of a single polynomial. p is a vector with the coefficients of the polynomial. k is a vector with the coefficients of the polynomial that is the derivative.
`k = polyder(a,b)`	Derivative of a product of two polynomials. a and b are vectors with the coefficients of the polynomials that are multiplied. k is a vector with the coefficients of the polynomial that is the derivative of the product.
`[n d] = polyder(u,v)`	Derivative of a quotient of two polynomials. u and v are vectors with the coefficients of the numerator and denominator polynomials. n and d are vectors with the coefficients of the numerator and denominator polynomials in the quotient that is the derivative.

The only difference between the last two commands is the number of output arguments. With two output arguments MATLAB calculates the derivative of the quotient of two polynomials. With one output argument the derivative is of the product.

For example, if $f_1(x) = 3x^2 - 2x + 4$, and $f_2(x) = x^2 + 5$, the derivatives of $3x^2 - 2x + 4$, $(3x^2 - 2x + 4)(x^2 + 5)$, and $\frac{3x^2 - 2x + 4}{x^2 + 5}$ can be determined by:

```
>> f1= 3 -2 4];
>> f2=[1 0 5];
>> k=polyder(f1)
k =
     6    -2
>> d=polyder(f1,f2)
d =
    12    -6    38   -10
>> [n d]=polyder(f1,f2)
n =
     2    22   -10
d =
     1     0    10     0    25
```

Creating the vectors of coefficients of f_1 and f_2.

The derivative of f_1 is: $6x-2$.

The derivative of $f_1 * f_2$ is: $12x^3-6x^2+38x-10$.

The derivative of $\frac{3x^2-2x+4}{x^2+5}$ is: $\frac{2x^2+22x-10}{x^4+10x^2+25}$.

Curve Fitting with Polynomials; The `polyfit` Function

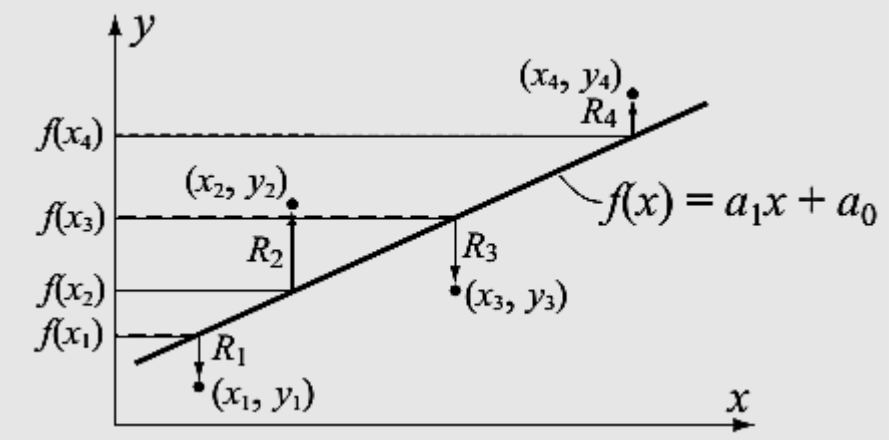
Polynomials can be used to fit data points in two ways. In one the polynomial passes through all the data points, and in the other the polynomial does not necessarily pass through any of the points, but overall gives a good approximation of the data. The two options are described below.

Polynomials that pass through all the points:

When *n* points (*xi*, *yi*) are given, it is possible to write a polynomial of degree $n-1$ that passes through all the points. For example, if two points are given it is possible to write a linear equation in the form of $y = mx + b$ that passes through the points. With three points the equation has the form of $y = ax^2 + bx + c$. With *n* points the polynomial has the form $a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots + a_1x + a_0$. The coefficients of the polynomial are determined by substituting each point in the polynomial and then solving the *n* equations for the coefficients. As will be shown later in this section, polynomials of high degree might give a large error if they are used to estimate values between data points.

Polynomials that do not necessarily pass through any of the points:

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 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 he difference between the value of the function at x_I and y_i, $R_i = f(x_i) - y_i$.



Least squares fitting of first-degree polynomial to four points.

An equation for the sum of the squares of the residuals 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 questions 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. More details on the least squares method can be found in books on numerical analysis. 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:

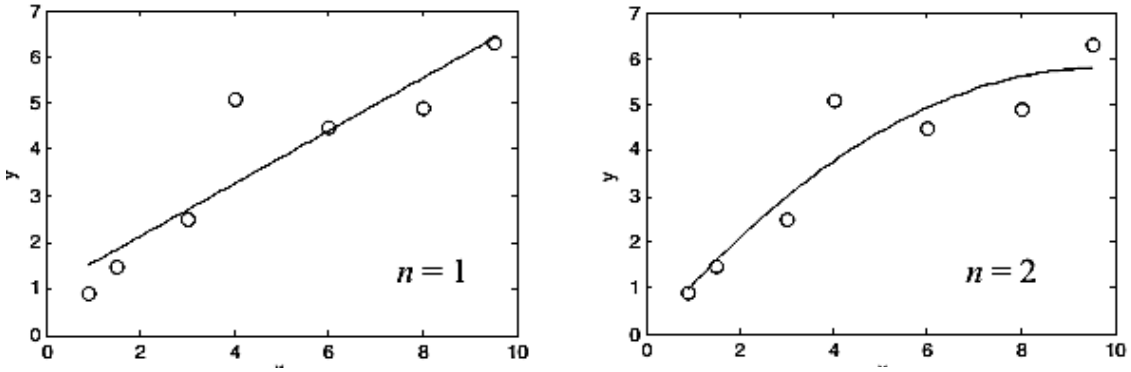
```
p = polyfit(x,y,n)
```

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

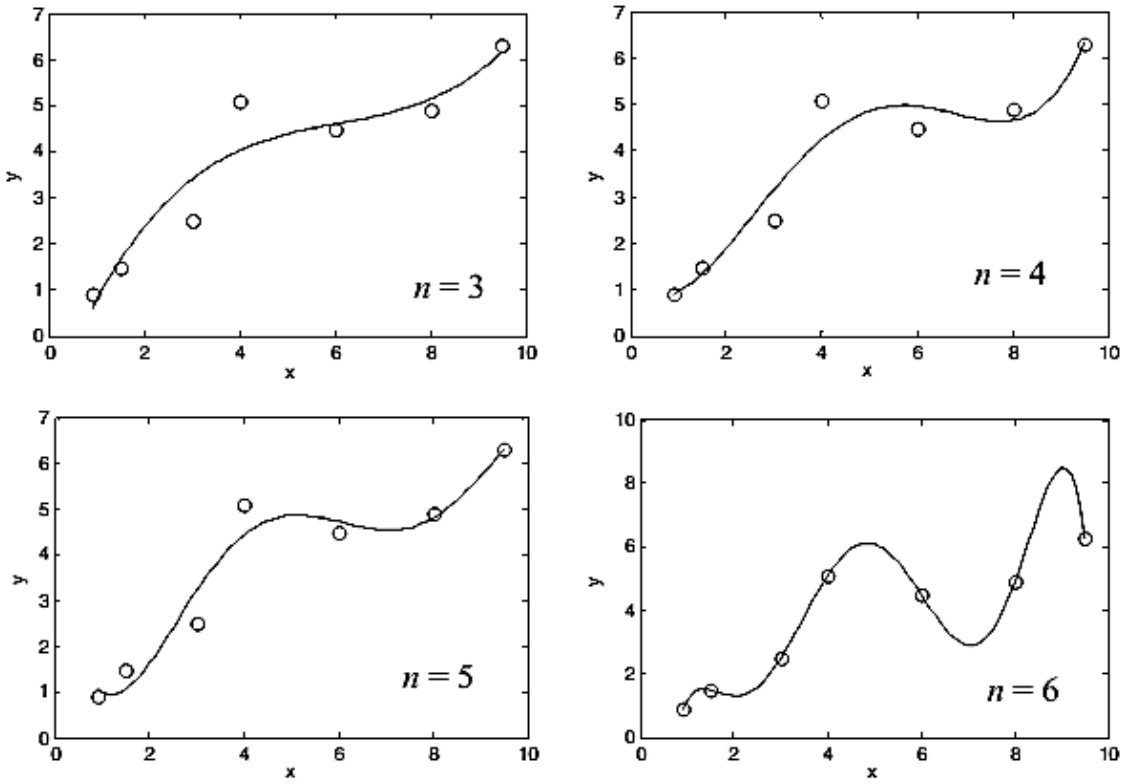
x is a vector with the horizontal coordinates of the data points (independent variable).
y is a vector with the vertical coordinates of the data points (dependent variable).
n is the degree of the polynomial.

For the same set of *m* points, the **`polyfit`** function can be used to fit polynomials of any order up to *m*-1. If $n = 1$ the polynomial is a straight line, if $n = 2$ the polynomial is a parabola, and so on. The polynomial passes through all the points if $n=m-1$(the order of the polynomial is one less than the number of points). It should be pointed out here that a polynomial that passes through all the points, or polynomials with higher order, do not necessarily give a better fit overall. High-order polynomials can deviate significantly between the data points.

Figure below shows how polynomials of different degrees fit the same set of data points. A set of seven points is given by (0.9, 0.9), (1.5, 1.5), (3, 2.5), (4, 5.1),



Fitting data with polynomials of different order.



Fitting data with polynomials of different order. (Continued)

(6, 4.5), (8, 4.9), and (9.5, 6.3). The points are fitted using the polyfit function with polynomials of degrees 1 through 6. Each plot in Figure up shows the same data points, marked with circles, and a curve-fitted line that corresponds to a polynomial of the specified degree. It can be seen that the polynomial with $n = 1$ is a straight line, and with $n = 2$ is a slightly curved line. As the degree of the polynomial increases, the line develops more bends such that it passes closer to more points. When $n = 6$, which is one less than the number of points, the line passes through all the points. However, between some of the points, the line deviates significantly from the trend of the data.

The script file used to generate one of the plots in Figure up (the polynomial with $n = 3$) is shown below. Note that in order to plot the polynomial (the line) a new vector xp with small spacing is created. This vector is then used with he function `polyval` to create a vector `yp` with the value of the polynomial for each element of `xp`.

```
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];
p=polyfit(x,y,3)
xp=0.9:0.1:9.5;
yp=polyval(p,xp)
plot(x,y,'o',xp,yp)
xlabel('x'); ylabel('y')
```

Create vectors x and y with the coordinates of the data points.

Create a vector p using the `polyfit` function.

Create a vector xp to be used for plotting the polynomial.

Create a vector yp with values of the polynomial at each xp.

A plot of the seven points and the polynomial.

When the script file is executed, the following vector p is displayed in the Command Window.

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

This means that the polynomial of the third degree has the form

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