TOPIC 5
Computer application for handling polynomial

Polynomials are used so commonly in computation that Matlab has special commands to deal with them. The polynomial x4 +2x3 −13x2 −14x+24 is represented in Matlab by the array [1,2,-13,-14,24], i.e., by the coefficients of the polynomial starting with the highest power and ending with the constant term. If any power is missing from the polynomial its coefficient must appear in the array as a zero. Here are some of the things Matlab can do with polynomials. Try each piece of code in Matlab and see what it does.

The following table lists the MATLAB polynomial functions.

Function	Description
conv	Multiply polynomials
deconv	Divide polynomials
poly	Polynomial with specified roots
polyder	Polynomial derivative
polyfit	Polynomial curve fitting
polyval	Polynomial evaluation
polyvalm	Matrix polynomial evaluation
residue	Partial-fraction expansion (residues)
roots	Find polynomial roots

Roots of a Polynomial

```
% find the roots of a polynomial
p=[1,2,-13,-14,24];
r=roots(p)
```

Find the polynomial from the roots

If you know that the roots of a polynomial are 1, 2, and 3, then you can find the polynomial in Matlab's array form this way

```
r=[1,2,3];
p=poly(r)
```

Multiply and devide Polynomials

Polynomial multiplication and division correspond to the operations convolution and deconvolution. The functions conv and deconv implement these operations.

Consider the polynomials $a(s) = s^2 + 2s + 3$ and $b(s) = 4s^2 + 5s + 6$. To compute their product,

```
a = [1 2 3]; b = [4 5 6];
c = conv(a,b)
c =
    4    13    28    27    18
```

Use deconvolution to divide $a(s)$ back out of the product:

```
[q,r] = deconv(c,a)
q =
    4     5     6
r =
    0     0     0     0     0
```

.

First Derivative

The polyder function computes the derivative of any polynomial. To obtain the derivative of the polynomial p = [1 0 -2 -5],

```
q = polyder(p)
q =
   3     0    -2
```

polyder also computes the derivative of the product or quotient of two polynomials. For example, create two polynomials a and b:

```
a = [1 3 5];
b = [2 4 6];
```

Calculate the derivative of the product a*b by calling polyder with a single output argument:

```
c = polyder(a,b)
c =
    8    30    56    38
```

Calculate the derivative of the quotient a/b by calling polyder with two output arguments:

```
[q,d] = polyder(a,b)
q =
   -2    -8    -2
d =
    4    16    40    48    36
```

q/d is the result of the operation.

Evaluating Polynomials
The polyval function evaluates a polynomial at a specified value. To evaluate p at s = 5, use polyval(p,5)

```
ans =
  110
```

It is also possible to evaluate a polynomial in a matrix sense. In this case $p(s) = x^3 - 2x - 5$ becomes $p(X) = X^3 - 2X - 5I$, where X is a square matrix and I is the identity matrix. For example, create a square matrix X and evaluate the polynomial p at X:

```
X = [2 4 5; -1 0 3; 7 1 5];
Y = polyvalm(p,X)
```

```
Y =
  377   179   439
  111    81   136
  490   253   639
```

Evaluate a Polynomial and drawing the graph
If you have an array of x-values and you want to evaluate a polynomial at each one, do this:

```
% define the polynomial
a=[1,2,-13,-14,24];
% load the x-values
x=-5:.01:5;
% evaluate the polynomial
y=polyval(a,x);
% plot it
plot(x,y)
```

Partial Fraction Expansions
residue finds the partial fraction expansion of the ratio of two polynomials. This is particularly useful for applications that represent systems in transfer function form. For polynomials b and a, if there are no multiple roots,

$$\frac{b(s)}{a(s)} = \frac{r_1}{s - p_1} + \frac{r_2}{s - p_2} + \ldots + \frac{r_n}{s - p_n} + k_s$$

where r is a column vector of residues, p is a column vector of pole locations, and k is a row vector of direct terms. Consider the transfer function

$$\frac{-4s + 8}{s^2 + 6s + 8}$$

```
b = [-4 8];
a = [1 6 8];
[r,p,k] = residue(b,a)
```

```
r =
   -12
     8

p =
    -4
    -2

k =
     [ ]
```
Given three input arguments (r, p, and k), residue converts back to polynomial form:
[b2,a2] = residue(r,p,k)

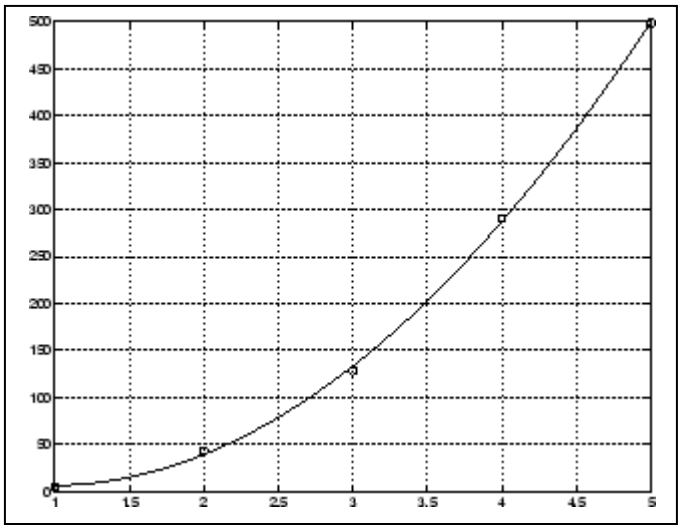
```
b2 =
    -4     8
a2 =
     1     6     8
```

Polynomial Curve Fitting

polyfit finds the coefficients of a polynomial that fits a set of data in a least-squares sense:
p = polyfit(x,y,n)
x and y are vectors containing the x and y data to be fitted, and n is the degree of the polynomial to return. For example, consider the x-y test data
x = [1 2 3 4 5]; y = [5.5 43.1 128 290.7 498.4];
A third degree polynomial that approximately fits the data is
p = polyfit(x,y,3)

```
p =
   -0.1917   31.5821  -60.3262   35.3400
```
Compute the values of the polyfit estimate over a finer range, and plot the estimate over the real data values for comparison:

```
x2 = 1:.1:5;
y2 = polyval(p,x2);
plot(x,y,'o',x2,y2)
grid on
```



Fitting Data to a Polynomial

If you have some data in the form of arrays (x,y), Matlab will do a least-squares fit of a polynomial of any order you choose to this data. In this example we will let the data be the sine function between 0 and _ and we will fit a polynomial of order 4 to it. Then we will plot the two functions on the same frame to see if the fit is any good.

```
x=linspace(0,pi,50);
% make a sine function with 1% random error on it
f=sin(x)+.01*rand(1,length(x));
% fit to the data
p=polyfit(x,f,4);
% evaluate the fit
g=polyval(p,x);
% plot fit and data together
plot(x,f,'r*',x,g,'b-')
```

Interpolating With polyfit and polyval

You can also use Matlab's polynomial commands to build an interpolating polynomial. Here is an example of how to use them to find a 5th-order polynomial fit to a crude representation of the sine function.

```
% make the crude data set with dx too big for  good accuracy
>>dx=pi/5;
>>x=0:dx:2*pi;
>>y=sin(x);
>>p=polyfit(x,y,5);     % make a 5th order polynomial fit to this data

>>xi=0:dx/20:2*pi;     % make a fine x-grid

% evaluate the fitting polynomial on the fine grid
>>yi=polyval(p,xi);
% and display the fit, the data, and the exact sine function

>>plot(x,y,'b*',xi,yi,'r-',xi,sin(xi),'c-')
>>pause
% display the difference between the polynomial fit and the exact sine function
>>plot(xi,yi-sin(xi),'b-')
```