

Experiment 3: Introduction to Matlab III

1. Polynomial Roots and Characteristic Polynomial

If p is a row vector containing the coefficients of a polynomial, **roots(p)** returns a column vector whose elements are the roots of the polynomial. If r is a column vector containing the roots of a polynomial, **poly(r)** returns a row vector whose elements are the coefficients of the polynomial.

$$p(x) = x^3 - 2x - 5$$

To enter this polynomial into MATLAB, use

```
>> p = [1 0 -2 -5];
```

1.1 Polynomial Roots

The roots function calculates the roots of a polynomial.

```
>> r = roots(p)
r =
    2.0946
   -1.0473 + 1.1359i
   -1.0473 - 1.1359i
```

By convention, MATLAB stores roots in column vectors. The function poly returns to the polynomial coefficients.

```
>> p2 = poly(r)
p2 =
    1 8.8818e-16 -2 -5
```

poly and **roots** are inverse functions.

Find the roots of following polynomial

$$S^6 + 9S^5 + 31.25S^4 + 61.25S^3 + 67.75S^2 + 14.75S + 15$$

The polynomial coefficients are entered in a row vector in descending powers.
The roots are found using **roots**.

```
p = [ 1 9 31.25 61.25 67.75 14.75 15 ]
r = roots(p)
```

The polynomial roots are obtained in column vector

```
r =  
-4.0000  
-3.0000  
-1.0000 + 2.0000i  
-1.0000 - 2.0000i  
0.0000 + 0.5000i  
0.0000 - 0.5000i
```

If we want to find the coefficient of polynomial that has the roots -1, -2, -3 + j4, -3 - j4
We write this

```
r = [-1 -2 -3+4i -3-4i ]  
p = poly(r)
```

The coefficients of the polynomial equation are obtained in a row vector.

```
p =  
1 9 45 87 50
```

Therefore, the polynomial equation is

$$S^4 + 9S^3 + 45S^2 + 87S + 50 = 0$$

1.2 Polynomial Evaluation

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 $p(x) = x^3 - 2x - 5$ 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
```

2. Solving Equations

2.1 Solving Algebraic Equations

If S is a symbolic expression, `solve(S)` attempts to find values of the symbolic variable in S (as determined by `find sym`) for which S is zero. For example,

```
>>syms a b c x
S = a*x^2 + b*x + c;
solve(S)
ans =
-(b + (b^2 - 4*a*c)^(1/2))/(2*a)
-(b - (b^2 - 4*a*c)^(1/2))/(2*a)
```

This is a symbolic vector whose elements are the two solutions.

Note that these examples assume equations of the form $f(x) = 0$. If you need to solve equations of the form $f(x)=q(x)$ you must use quoted strings. In particular, the command

```
s = solve('cos(2*x)+sin(x)=1')
s =
0
pi/6
(5*pi)/6
```

2.2 Several Algebraic Equations

Now let's look at systems of equations. Suppose we have the system

$$\begin{aligned}x^2 y^2 &= 0 \\ x - \frac{y}{2} &= \alpha\end{aligned}$$

and we want to solve for x and y . First create the necessary symbolic objects. There are several ways to address the output of `solve`. One is to use a two-output call

```
>>syms x y alpha
>> [x,y] = solve(x^2*y^2, x-y/2-alpha)
x =
alpha
0
y =
0
(-2)*alpha
```

3. Single Differential Equation

The function *dsolve* computes symbolic solutions to ordinary differential equations. The equations are specified by symbolic expressions containing the letter D to denote differentiation. The symbols D2, D3, ... DN, correspond to the second, third, ..., Nth

Derivative, respectively. Thus, D2y is the Symbolic Math of $\frac{dy^2}{dt^2}$

The dependent variables are those preceded by D and the default independent variable is *t*. Note that names of symbolic variables should not contain D. The independent variable can be changed from *t* to some other symbolic variable by including that variable as the last input argument.

Initial conditions can be specified by additional equations. If initial conditions are not specified, the solutions contain constants of integration, C1, C2, etc. The output from *dsolve* parallels the output from *solve*. That is, you can call *Dsolve* with the number of output variables equal to the number of dependent variables or place the output in a structure whose fields contain the solutions of the differential equations.

Example 1

The following call to *dsolve*

`dsolve('Dy=1+y^2')`

uses *y* as the dependent variable and *t* as the default independent variable.

The output of this command is

```
>>dsolve('Dy=1+y^2')
ans =
     i
    -i
tan(C4 + t)
```

To specify an initial condition, use

```
y = dsolve('Dy=1+y^2','y(0)=1')
y =
tan(t+1/4*pi)
```

Notice that *y* is in the MATLAB workspace, but the independent variable *t* is not. Thus, the command *diff(y,t)* returns an error. To place *t* in the workspace, type *syms t*.

4. Partial Fraction Expansion

You can find the Partial Fraction Expansion (PFE) using the MATLAB command ***residue***.

`[r,p,k]=residue(num,den);`

Where r is the vector of PFE coefficients, p the vector of roots of den1, and k=0 if the degree of num1 < the degree of den1.

Consider the following transfer function:

$$\frac{2S^3 + 5S^2 + 3S + 6}{S^3 + 6S^2 + 11S + 6}$$

we write

```
num=[2 5 3 6]
```

```
den=[1 6 11 6]
```

```
[r,p,k]=residue(num,den)
```

r

-6

-4

3

p

-3

-2

-1

k

2

Which mean

$$\frac{-6}{S+3} + \frac{-4}{S+2} + \frac{3}{S+1} + 2$$