ROOTS OF EQUATIONS

% Many engineering problems require the finding of the x's for a function f(x) such that **f(x)=0.** For example; if we want to find the solution to **x =sin(x)** , we transform this situation to **f(x) = x-sin(x)** which now reads **x-sin(x)** =0 and then use various techniques to solve for x to obtain **f(x)=0.** This concept is also used to find the roots of polynomials i.e. what are the values of x that make the polynomial

$3x^5 - 6x^3 + 12x^2 - 5x + 89 = 0$. You are already familiar with solving the quadratic equation. **$ax^2 + bx + c = 0$**.

NOTE:

There are a number of efficient numerical techniques used in programming to solve f(x) = 0. Simple techniques called Bisection or False Position, as well as, more sophisticated Newton-Raphson and Secant methods are techniques you may encounter in your careers.

ROOTS OF POLYNOMIAL EQUATIONS

% One set of problems of finding x for f(x)=0 is when f(x) is a polynomial
MATLAB has several functions to handle this case, we will consider, **roots()**, and **poly()** which act on **vectors covered here (also matrices, later)** defined in the following FUNCTION definitions:

» help roots

```
 ROOTS       Find polynomial roots. ROOTS(C) computes the roots of the
             polynomial whose coefficients are the elements of the
             vector C. If C has N+1 components, the polynomial is
             C(1)*X^N + ... + C(N)*X + C(N+1). See ROOTS1 and POL
```

% FOR EXAMPLE for $y=f(x) = x^3 - 5x^2 + 2x + 8 = 0$

We define a vector containing the coeficients of the polynomial, C, as

» C =[1 -5 2 8];

% and obtain the roots with roots() or roots1() functions
as follows

```
» roots(C)
ans =    4.0000
          2.0000
         -1.0000
```

% It is convenient to store the answer in a vector for use with our next function poy() if we wanted to check out if these are really the roots, as in

```
» V = roots(C)
V =    4.0000
        2.0000
       -1.0000
```

% Once roots have been determined by the above functions for polynomials we can **check out the resulting roots to see that they are the solutions we seek** by using the function **poly()** which regenerates the polynomial.

```
» help poly
 POLY Characteristic polynomial.
             ...If V is a vector, POLY(V) is a vector whose elements are
             the coefficients of the polynomial whose roots are the
             elements of V . For vectors, ROOTS and POLY are inverse
             functions of each other, up to ordering, scaling, and
             roundoff error.
```

Of course, the alternative way would be to substitute the values of each x into the original f(x) and see if the answer is zero!
% Checking the answer, we just did, with 'poly' becomes easy with the vector answer to roots(),namely, V above.

» poly(V)
ans = 1.0000 -5.0000 2.0000 8.0000

%Thus we obtained our original polynomial and the corresponding coefficients giving us confidence in our answer **$f(x) = x^3 - 5x^2 + 2x + 8 = 0$**
I.E. we used the roots obtained with roots() to obtain the original coefficients with poly()

Alternatively we could just let x = one of roots and evaluate y as in..
```
>> x=4;
>> y=x^3 -5*x^2 +2*x +8
y =     0
```

An another example for the roots()-poly() game plan.
Let us solve a fifth order polynomial $f(x)=x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24 = 0$

» C =[1 -3 -11 +27 +10 -24]

% The root solution is easily generated with

```
» z = roots(C)
z =    4.0000
      -3.0000
       2.0000
       1.0000
      -1.0000
```

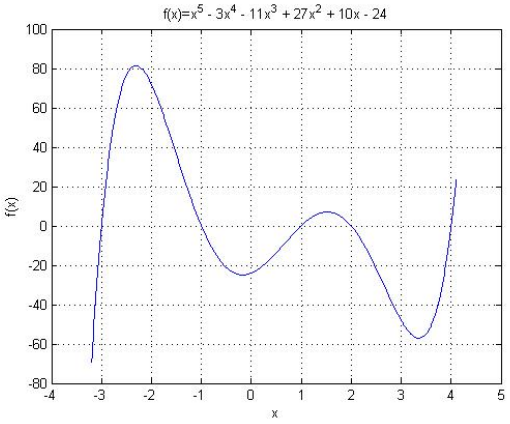
% TO CHECK IF THIS IS THE RIGHT ANSWER FOR OUR POLYNOMIAL WHICH AT THIS POINT IS REDUNDANT AND JUST BEING DONE TO MAKE A POINT HOW ACCURATE THESE FUNCTIONS ARE.
```
» p = poly(z)
 p =    1.0000   -3.0000  -11.0000   27.0000   10.0000  -24.0000
```

% SHOWING 100% AGREEMENT WITH OUR ORIGINAL COEFFICIENTS FOR THE POLYNOMIAL

% TO OBSERVE A FUNCTIONS BEHAVIOR AND VISUALLY SEE THE ROOT AREAS WE CAN ALWAYS SET UP A FUNCTION FILE
% Given the polynomial function **$f(x)=x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$**
We set up function file for it which makes it easier to use in other programs.

```
» type p5ex1
function y = p5ex1(x)
y = x.^5 -3*x.^4 -11* x.^3 +27*x.^2 +10*x -24;
```

We can observe the roots by studying the graph, as in
```
» x=-3.2:.01:4.1;
» y2=p5ex1(x);
>>plot(x,p5ex1(x))
>> grid
```
% with a few added embellishments we get and “see” the five zero’s easily



% ADDITIONAL POLYNOMIAL EXAMPLES

% MULTIPLE IDENTICAL ROOT EXAMPLE
F(x) **$f(x)=x^5 - 4x^4 - 9x^3 + 32x^2 + 28x - 48$**
» C = [1 -4 -9 +32 +28 -48]
C = 1 -4 -9 32 28 -48

» r = roots(C)
r = 4.0000
3.0000
-2.0000
-2.0000
1.0000

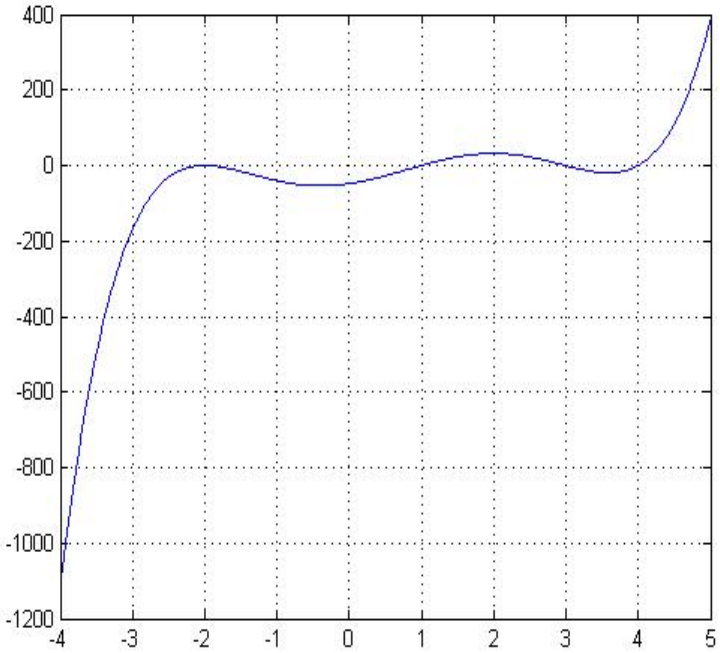
% We note the vector of the roots has two equal values

» %Though we could check our roots with **poly()** which shows us

» A =poly(r)
A = 1.0000 -4.0000 -9.0000 32.0000 28.0000 -48.0000

poly() is useful if we have roots we want and are looking for an equation to cover those roots.

Plots are always useful to visualize and help us understand our answers as before:
Check out.
>> x=-4:.01:5;
>> y=x.^5 - 4*x.^4 - 9*x.^3 + 32*x.^2 + 28*x - 48;
>> plot(x,y)
>> grid



We note the double root at -2 is uniquely an extrema point (local maximum) slope = 0 just at the root value f(x) =0 and all the roots are real.

% A MORE COMPLEX EXAMPLE **$f(x)=x^5+3x^4-4x^3-26x^2-40x-24$**
» C = [1 3 -4 -26 - 40 -24]

» r =roots(C)
3.0000 + 0.0000i
-2.0000 + 0.0000i
-2.0000 + 0.0000i
-1.0000 + 1.0000i
-1.0000 - 1.0000i
% clearly all roots look imaginary with the last two clear what they are, to probe further the first three complex numbers we use format long.

>> format long
>> r
r =
3.000000000000000 + 0.000000000000000i
-2.000000084084592 + 0.000000000000000i
-1.999999915915403 + 0.000000000000000i
-1.000000000000002 + 1.000000000000001i
-1.000000000000002 - 1.000000000000001i

Clearly the first three have ambiguous imaginary parts of the complex number. And can be considered real to the accuracy of the routine used behind the scenes. To be sure we plot the function below.
Here it is also prudent to check the results with poly(), to be sure the mathematical technique being used by function is behaving properly.

% THE CHECKING COMES WITH using the **poly()** function which in format long yields
>> A=poly(r)

A = Columns 1 through 2
1.000000000000000 2.999999999999997
Columns 3 through 4
-4.000000000000004 -25.999999999999975

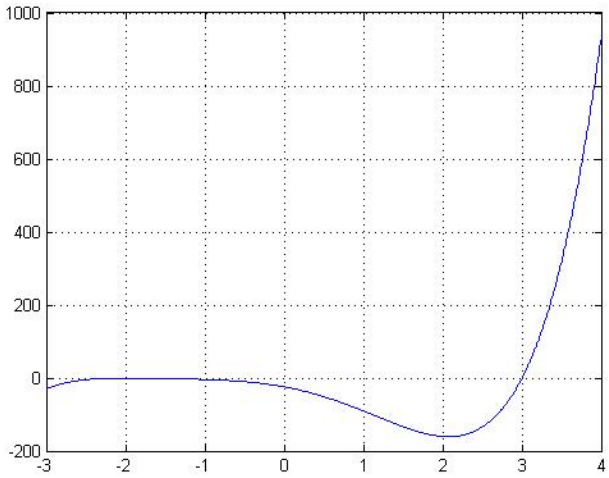
Columns 5 through 6
-39.999999999999979 -23.999999999999957

Close numbers but not exactly the same coefficients if we look at this numbers in format short we see to **that accuracy** we recover the coefficients of the original polynomial.

```
>> format short
>> A
A =  Columns 1 through 5
   1.0000    3.0000   -4.0000  -26.0000  -40.0000
  Column 6
  -24.0000
```

Here its prudent to plot the function to get a sense of what is happening.

```
>> x=-3:.01:4;
>>y5=x.^5 +3*x.^4 - 4*x.^3 -26*x.^2 -40*x – 24;
>>plot(x,y5)
>>grid
```



The root near x=3 is clearly real and the two near -2 are for both short and long accuracy above also real since the imaginary part is clearly zero and there are two additional imaginary roots.
The problem for this techniques is clearly seen on the graph in the negative x axis.

Just because roots can be complex does not mean we would have the same messy answers as above. Consider the following

```
» C=[1 -9 35  -65 64 -26]
C =     1    -9    35   -65    64   -26

» r =roots(C)
r =  3.0000 +  2.0000i
     3.0000  -  2.0000i
     1.0000 +  1.0000i
     1.0000  -  1.0000i
     1.0000
```

% THE CHECK

```
>> A= poly(r)
A =    1.0000   -9.0000   35.0000  -65.0000   64.0000  -26.0000
```

.%the graph would show a root at 1,(almost asymptotically approach from both sides and now other crossing of the x= 0 axis. The latter is why 4 complex roots can be found.

% SIMPLE TECHNIQUES OF 'BISECTION' OR 'FALSE- POSITION' OR MORE SOPHISTICATED NEWTON- RAPHSON METHOD OR SECANT METHODS YIELD US GOOD ANSWERS. WE CAN,ALSO, FIND ROOTS ONE AT A TIME WITH MATLABS GENERAL FIND A ROOT FUNCTION CALLED **fzero()** BUT WE WILL USE THIS FUNCTION TO FIND ROOTS OF NON=POLYNOMIAL EQUATIONS below.

USING THE FUNCTION TO CHECK THE ROOTS

% AS YOU SAW ABOVE A VERY GOOD CHECK ON THE VALIDITY OF THE ROOT(S) YOU FIND IS TO SUBSTITUTE A ROOT INTO THE FUNCTION (POLYNOMIAL OR ANY OTHER) TO SEE IF THE VALUE OF F(root) IS NEAR ZERO TO WITHIN THE TOLERANCE OF THE FUNCTION YOU ARE USING (FOR MOST PURPOSES STANDARD VALUES SHOULD BE OK)
IT IS EASIEST TO SET UP A FUNCTION M-FILE TO UNDERTAKE THIS CHECK WHICH IS ILLUSTRATED BY THE EXAMPLES TO FOLLOW IN SEVERAL WAYS.

%CONSIDER This specific 4^{th} order POLYNOMIAL DEFINED BY THE FOLLOWING FUNCTION M-FILE

```
» type poly458
 function y =poly458(x)
 a0=0;
 a1=1;
 a2=-2.125;
 a3=-25.0;
 a4=53.125;
 y =  a0*x.^4 + a1* x.^3 +a2* x.^2+ a3*x +a4
```

% WE NOTE THE **a** COEFFICIENTS ARE LOCAL VARIABLES TO THE FUNCTION WHICH MEANS THEY ARE NOT KNOWN OUTSIDE THE FUNCTION.AND THAT MEANS WE HAVE TO REDEFINE A Coefficient VECTOR, C, FOR poly458() AS

```
» C =[0 1 -2.125 -25 53.125]
C =         0    1.0000   -2.1250  -25.0000   53.1250
» root=roots(C)
root = -5.0000
          5.0000
          2.1250
```

% NOW AS AN ADDITIONAL CHECK TO USING **poly(),** WHEN THE FUNCITION IS A POLYNOMIAL, WE CHECK **f(root)=0** WHICH CAN BE USED EVEN WHEN WE ARE STUDYING THE ROOTS OF NON-POLYNOMIAL EQUATIONS (see below)

```
» poly458(5)
ans =     0
» poly458(-5)
ans =     0

» poly458(2.1251)
ans=   -0.0020                %ILLUSTRATED VALUE IS JUST A LITTLE OFF ROOT

» poly458(2.1250)
ans =     0
```

% WE CONCLUDE THESE ROOTS ARE VERY GOOD ANSWERS!

% A GOOD PROGRAMING STRATEGY FOR USE IN OTHER PROGRAMS FOR POLYNOMIALS is to set up a **general function** to handle a whole family of functions, AS ILLUSTRATED IN THE NEXT FUNCTION M-FILE. Where we pass all the coefficients and the value of x to the function

```
» type poly4
 function y =poly4(a0,a1,a2,a3,a4,x)
 y =  a0*x.^4 + a1* x.^3 +a2* x.^2+ a3*x +a4;
```

% Remember this is for all 4^{th} order polynomial and lower if a0 =0 we get a third order or a0=0 and a1=0 we get a quadratic etc.
% The “a” variables are the coefficients and x is the vector range to pass to the function

% THAT IS WE PASS THE COEFFICIENTS AS ARGUMENTS (function variables are local)
%AN INTERACTIVE APPROACH AFTER WE HAVE DEFINED THE ABOVE FUNCTION M-FILE IS
» a0= 3
» a1= -12.4
» a2= -26.290
» a3= 29.766
» a4= 0

» C=[a0 a1 a2 a3 a4]
C = 3.0000 -12.4000 -26.2900 29.7660 0
Format long will be needed here for the best answers
» root=roots(C)
Root= 0
5.413551255911684
-2.137622725179635
0.857404802601285

» >> poly4(a0,a1,a2,a3,a4, 5.413551255911684)
The value should be?

Since the quadratic equation comes up often we can use poly4() as noted before by setting the first two coefficients to zero.

% USING THE COEFICIENTS THAT MIGHT BE USED IN THE ABOVE GENERAL M-FILE FOR A QUADRATIC POLYNOMIAL FOLLOWS
» a0=0;a1=0;a2=1;a3=2;a4=10;
» C=[a0 a1 a2 a3 a4]
C = 0 0 1 2 10
» root=roots(C)
root = -1.0000 + 3.0000i
-1.0000 - 3.0000i

% ILLUSTRATING A THIRD ORDER POLYNOMIAL WITH THE GENERAL COEFFICIENTS
» a0=0;a1=1;a2=-2.125;a3=-25;a4=53.125;
» C=[a0 a1 a2 a3 a4]
C = 0 1.0000 -2.1250 -25.0000 53.1250
» root=roots(C)
root = 5.0000
-5.0000
2.1250

% AN ADDITIONAL QUADRATIC EQUATION
» a0=0;a1=0;a2=1;a3=14;a4=3;
» C=[a0 a1 a2 a3 a4]
C = 0 0 1 14 3
» root=roots(C)
root = -13.7823
-0.2177

% ANOTHER QUADRATIC
» a0=0;a1=0;a2=1.234;a3=-1.2;a4=10.44;
» C=[a0 a1 a2 a3 a4]
C = 0 0 1.2340 -1.2000 10.4400
» root=roots(C)
root = 0.4862 + 2.8677i
0.4862 - 2.8677i

% A LINEAR CASE
» a0=0;a1=0;a2=0;a3=3;a4=-2.5;
» C=[a0 a1 a2 a3 a4]
C = 0 0 0 3.0000 -2.5000
» root=roots(C)
root = 0.8333

NON-POLYNOMIAL FUNCTIONS AND THE USE OF fzero()

% MATLABS GENERAL FUNCTION TO FIND A ROOT IS BASED ON ONE FUNCTION CALLED fzero(). fzero() can be used for all functions including polynomials but it *only finds one root* at a time after you guess at a near solution. So for polynomials the above techniques are in general superior.

```
>> help fzero
   fzero   As a Single-variable nonlinear zero finding function.
   X = fzero(FUN,X0) tries to find a zero of the function FUN near X0,
   if X0 is a scalar.  It first finds an interval containing X0 where the
   function values of the interval endpoints differ in sign, then searches
   that interval for a zero.  FUN is a function handle.  FUN accepts real
   scalar input X and returns a real scalar function value FUN, evaluated
   at X. The value X returned by fzero is near a point where FUN changes
   sign (if FUN is continuous), or NaN if the search fails.
```

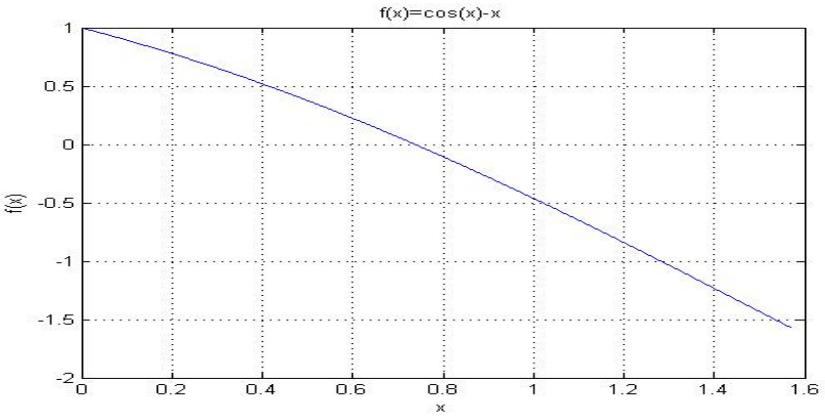
The key here for your understanding in the techniques is that fzero looks for an interval containing a sign change for FUN and containing X0. X0 is a guess at the value.

%EAMPLE in the range 0<x<pi/2 solve x=cos(x) ?
% Thus we explore f(x)=0=cos(x)-x=0

% SINCE HERE WE MUST SUPPLY A GUESS 'X' NEAR THE ROOT, IT IS BEST TO **PLOT THE FUNCTION FIRST** AND FIND THEVALUES FROM THE GRAPH THAT ARE NEAR THE ROOT. In other words for this example we are looking for values of x that make f(x) change sign! If we see none then we know there is no real root in the equation we have set up.

% Setting up the graph with very fine interval and a few embellishments to look for close values to a root.

```
>> x=0:.001:pi/2;
>> y=cos(x)-x;
>> plot(x,y)
>> grid
```



We see from the graph that a root lies between 0.6 and 0.8
So to use fzero we set up a function m file for our function.

%The last curve suggest a root near 0.6 which we will use as our Guess in fzero
% WE CONSTRUCT the M-FILE called fta.m

```
» type fta
function y =fta(x)
y = cos(x)-x;

>> root=fzero('fta',0.6)
root =    0.7391
```

The solution in this case is very robust and you have a wide range of values to pick from to converge to the root. For example

```
>> root=fzero('fta',0.8)
root =    0.7391

>> root=fzero('fta',1.4)
root =    0.7391
```

Is the function really zero at this root value

```
>> z=fta(root)
z =     0
```

actually in memory is a more precise answer which was used to evaluate the function at the root

```
>> format long
>> root=fzero('fta',1.4)
root =   0.739085133215161
>> z=fta(root)
z =     0
```

% LABORATORY TASKS

38(5PTS) To SOLVE $Y = X^5 + 3.1 X^4 - 4 X^3 - 25 X^2 - 39 X - 25.234$

1. SET UP A GENERAL FUNCTION FOR A 5TH ORDER POLYNOMIAL (M-FILE) as above
2. SET UP VECTOR WITH PROPER COEFFICIENTS FOR THIS POLYNOMIAL (like above)
3. USE FUNCTION **roots()**
4. Check answers with **poly()** TO GET BACK THE COEFFICIENTS USED IN ROOTS and 'values of function' at the various roots(ARE THEY GOOD) BY CALLING ON YOUR GENERAL 5^{TH} ORDER M FILE!
5. Plot the function as stated in above, over an appropriate interval that covers all roots. USE FINE DIVISON OF INTERVAL. Mark the roots on the graph.

39(3 PTS). use the function m file you setup in 38 and find the roots with checks using the m file function for the Quadratic $y= 3x^2 + 7x -4$ graph the solution over an appropriate interval that covers the roots and mark the roots

40(5 PTS). SOLVE $Y= SIN(X) * COS(X/2)$ FOR $-7 < X < 7$ USE FINE INTERVAL

1. SET UP FUNCTION(M-FILE)
2. PLOT THE FUNCTION OVER THE RANGE WITH A GRID TO HELP FIND GUESS VALUES Mark them on the graph.
3. USE **fzero()** AND FIND ALL ROOTS,**ONE AT A TIME**, OVER THE RANGE that you can.
4. CHECK EACH ROOT (HOW GOOD ARE THE ROOTS -COMMENT ON THEM see below)
5. Is X = pi a root?...prove it. Does 'fzero' give you an answer.
   Why or Why NOT! EXPLAIN!