Raul Rojas
Math 2335 Test 2 part 2
October 28, 2017

Part 1

The function f(x) = 4cosx – e^x has a zero on interval [0.5, 1] because knowing that the function is continuous on that interval, we can evaluate the function at the interval endpoints, 0.5 and 1, and the signs of the result will be opposite. This needs to be true so that we are sure there is a zero on the initial interval before we can apply the bisection method to approximating the root.

```
%Root finding with Bisection Method%
function root = bisRoot(f,a,b,max,t)
ya = f(a);
yb = f(b);
if sign(ya) == sign(yb), error('function has same sign at end points'),end
disp('          step                 a                  b                   m
ym                bound')
for k = 1:max
    m = (a+b)/2;
    ym = f(m);
    i = k;
    bound = (b-a)/2;
    out = [i, a, b, m, ym, bound ]; disp(out)
    if abs(ym)<t, disp('bisection has converged');
        root = m;
        break;
    end
    if sign(ym) ~= sign(ya)
        b = m;
        yb = ym;
    else
        a = m;
        ya = ym;
    end
    if (i >= max), disp('zero not found to desired tolerance'), end
end
```

Results

```
>> f=inline('4*cos(x)-exp(x)')

f =

     Inline function:
     f(x) = 4*cos(x)-exp(x)

>> a=0.5;
>> b=1;
>> max=20;
>> tol=0.00001;
>> root=bisRoot(f,a,b,max,tol)
          step                  a                   b                   m                  ym                 bound
   1.000000000000000   0.500000000000000   1.000000000000000   0.750000000000000   0.809755458882609   0.250000000000000

   2.000000000000000   0.750000000000000   1.000000000000000   0.875000000000000   0.165112138686203   0.125000000000000

   3.000000000000000   0.875000000000000   1.000000000000000   0.937500000000000  -0.186369157693017   0.062500000000000

   4.000000000000000   0.875000000000000   0.937500000000000   0.906250000000000  -0.008215505247377   0.031250000000000

   5.000000000000000   0.875000000000000   0.906250000000000   0.890625000000000   0.079052852292025   0.015625000000000

   6.000000000000000   0.890625000000000   0.906250000000000   0.898437500000000   0.035569646893613   0.007812500000000

   7.000000000000000   0.898437500000000   0.906250000000000   0.902343750000000   0.013714794089033   0.003906250000000

   8.000000000000000   0.902343750000000   0.906250000000000   0.904296875000000   0.002759072727380   0.001953125000000

   9.000000000000000   0.904296875000000   0.906250000000000   0.905273437500000  -0.002725859497108   0.000976562500000

  10.000000000000000   0.904296875000000   0.905273437500000   0.904785156250000   0.000017195845077   0.000488281250000

  11.000000000000000   0.904785156250000   0.905273437500000   0.905029296875000  -0.001354184523432   0.000244140625000

  12.000000000000000   0.904785156250000   0.905029296875000   0.904907226562500  -0.000668457512919   0.000122070312500

  13.000000000000000   0.904785156250000   0.904907226562500   0.904846191406250  -0.000325621627280   0.000061035156250

  14.000000000000000   0.904785156250000   0.904846191406250   0.904815673828125  -0.000154210589431   0.000030517578125

  15.000000000000000   0.904785156250000   0.904815673828125   0.904800415039063  -0.000068506796758   0.000015258789063

  16.000000000000000   0.904785156250000   0.904800415039063   0.904792785644531  -0.000025655331986   0.000007629394531

  17.000000000000000   0.904785156250000   0.904792785644531   0.904788970947266  -0.000004229707491   0.000003814697266

bisection has converged

root =

   0.904788970947266

>>
```

Part 2

Determining the zero locations:

```
>> x=0:.5:10

x =

  Columns 1 through 6

                   0   0.500000000000000   1.000000000000000   1.500000000000000   2.000000000000000   2.500000000000000

  Columns 7 through 12

   3.000000000000000   3.500000000000000   4.000000000000000   4.500000000000000   5.000000000000000   5.500000000000000

  Columns 13 through 18

   6.000000000000000   6.500000000000000   7.000000000000000   7.500000000000000   8.000000000000000   8.500000000000000

  Columns 19 through 21

   9.000000000000000   9.500000000000000  10.000000000000000

>> f=x.*sin(x)+cos(x)

f =

  Columns 1 through 6

   1.000000000000000   1.117295331192474   1.381773290676036   1.566979681573784   1.402448017104221   0.695036744712957

  Columns 7 through 12

  -0.566632472420844  -2.164197984204466  -3.680853602095325  -4.609681328923716  -4.510959187852466  -3.171802016345896

  Columns 13 through 18

  -0.716322702543189   2.374867548298824   5.352808445374828   7.381635143645568   7.769365939178440   6.185128554614844

  Columns 19 through 21

   2.797936105291132  -1.711107800583567  -6.279282637970150

>> plot(x,f)
>>
```

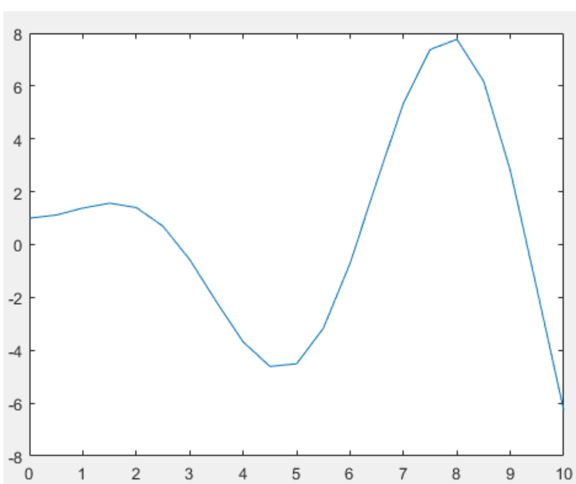

Figure 1: Part 2 Graph.

```
function [x, y, z] = Newton(fun, funpr, x1, tol, kmax)
%Newton Method to finding roots%
x(1) = x1;
y(1) = feval(fun, x(1));
ypr(1) = feval(funpr, x(1));
for k = 2: kmax
    x(k) = x(k-1)-y(k-1)/ypr(k-1);
    y(k) = feval(fun, x(k));
    z(k) = x(k) - x(k-1);
    if abs(z(k)) < tol
        disp('Newton method has converged'); break;
    end
    ypr(k) = feval(funpr, x(k));
    iter = k;
end
if(iter>=kmax)
    disp('zero not found to desired tolerance');
end
disp('           k                   x(k)                     y(k)                x(k)-x(k-1)')
n = length(x);
out=[1, x(1), y(1)]; disp(out)
for k=2:n
    out=[k,x(k), y(k), z(k)];disp(out)
end
```

Results

Root 1

```
>> f=inline('x.*sin(x)+cos(x)');
>> a=2.5;
>> b=3;
>> m=(a+b)/2;
>> df=inline('x.*cos(x)');
>> max=10;
>> tol=0.00001;
>> [x,y,z]=Newton(f,df,m,tol,max)
Newton method has converged
           k                 x(k)                y(k)            x(k)-x(k-1)
   1.000000000000000   2.750000000000000   0.125265349511449

   2.000000000000000   2.799281530848569  -0.002360524109045   0.049281530848569

   3.000000000000000   2.798386331978174  -0.000000754175099  -0.000895198870395

   4.000000000000000   2.798386045783917  -0.000000000000077  -0.000000286194258


x =

   2.750000000000000   2.799281530848569   2.798386331978174   2.798386045783917


y =

   0.125265349511449  -0.002360524109045  -0.000000754175099  -0.000000000000077


z =

                   0   0.049281530848569  -0.000895198870395  -0.000000286194258

>>
```

Root 2

```
>> a=6;
>> b=6.5;
>> m=(a+b)/2;
>> [x,y,z]=Newton(f,df,m,tol,max)
Newton method has converged
           k                 x(k)                y(k)            x(k)-x(k-1)
   1.000000000000000   6.250000000000000   0.792079314802269

   2.000000000000000   6.123197494483011   0.011766053566626  -0.126802505516989

   3.000000000000000   6.121251083480993   0.000003724880946  -0.001946411002018

   4.000000000000000   6.121250466898131   0.000000000000377  -0.000000616582862


x =

   6.250000000000000   6.123197494483011   6.121251083480993   6.121250466898131


y =

   0.792079314802269   0.011766053566626   0.000003724880946   0.000000000000377


z =

                   0  -0.126802505516989  -0.001946411002018  -0.000000616582862

>>
```

Root 3

```
>> a=9;
>> b=9.5;
>> m=(a+b)/2;
>> [x,y,z]=Newton(f,df,m,tol,max)
Newton method has converged
          k                 x(k)               y(k)          x(k)-x(k-1)
   1.000000000000000   9.250000000000000   0.623712566301687

   2.000000000000000   9.318471537543022  -0.005606187978040   0.068471537543022

   3.000000000000000   9.317866501005730  -0.000000363310739  -0.000605036537292

   4.000000000000000   9.317866461791066  -0.000000000000008  -0.000000039214664


x =

   9.250000000000000   9.318471537543022   9.317866501005730   9.317866461791066


y =

   0.623712566301687  -0.005606187978040  -0.000000363310739  -0.000000000000008


z =

                   0   0.068471537543022  -0.000605036537292  -0.000000039214664

>>
```

Conclusion:

Before applying the Newton method script to find the roots I plotted the graph to determine initial points 'm'. Once those points were found, the newton method script resulted in the following zero approximations (approximated with a tolerance less than 10^{-5}): 2.798386045783917, 6.121250466898131, 9.317866461791066.