Raw_Me

10/2/2017

CSCI-3656

Homework 2

Q1:

(a) The code I used for the three methods:

```
%bisection function

function [counter,iter,root,errors] = bisection(f,a,b,tolerance)

errors=[];

counter=[];

iter=[];

if f(a)*f(b)>0

   disp("guesses do not bracket root!")

else

   root = (a + b)/2; err = abs(f(root)); count = 0;

   while err > tolerance

   if f(a)*f(root)<0

      b = root;

   else

      a = root;

   end

   root = (a + b)/2; err = abs(f(root)); count = count + 1;

   errors = [errors,err]; counter = [counter,count]; iter = [iter,root];

   end
```

```
end

end
```

%Newton’s function

```
function [counter,iter,root,errors] = newton(f,f_prime,root,tolerance)
    errors=[]; err = abs(f(root));counter=[]; iter=[]; count = 0;
    while err > tolerance
    root = root - f(root)/f_prime(root);
    err = abs(f(root)); errors = [errors,err]; count = count +1;
    counter = [counter,count];
    iter = [iter,root];
    end
end
```

%Secant function

```
function [counter,iter,root,errors] = newton(f,f_prime,root,tolerance)
    errors=[]; err = abs(f(root));counter=[]; iter=[]; count = 0;
    while err > tolerance
    root = root - f(root)/f_prime(root);
    err = abs(f(root)); errors = [errors,err]; count = count +1;
    counter = [counter,count];
    iter = [iter,root];
    end
end
```

(b) The roots are -0.56873 , 1.3187. When using the methods(in Matlab):

>> [counter,iter,root,errors] = bisection(f,-1,0,0.0001)

counter =

1 2 3 4 5 6 7 8 9 10 11 12

iter =

-0.7500 -0.6250 -0.5625 -0.5938 -0.5781 -0.5703 -0.5664 -0.5684 -0.5693

-0.5688 -0.5686 -0.5687

root =

-0.5687

errors =

1.5000 0.4375 0.0469 0.1914 0.0713 0.0120 0.0175 0.0028 0.0046

0.0009 0.0009 0.0000

>> [counter,iter,root,errors] = bisection(f,0,2,0.0001)

counter =

1 2 3 4 5 6 7 8 9 10 11 12 13

iter =

1.5000 1.2500 1.3750 1.3125 1.3438 1.3281 1.3203 1.3164 1.3184

1.3193 1.3188 1.3186 1.3187

root =

1.3187

errors =

1.5000 0.5000 0.4375 0.0469 0.1914 0.0713 0.0120 0.0175 0.0028

0.0046 0.0009 0.0009 0.0000

>> [counter,iter,root,errors]=newton(f,f_prime,-1,0.0001)

```
counter =

     1     2     3

iter =

   -0.6364   -0.5710   -0.5687

root =

   -0.5687

errors =

    0.5289    0.0171    0.0000
```

>> [counter,iter,root,errors]=newton(f,f_prime,1,0.0001)

```
counter =

     1     2     3

iter =

    1.4000    1.3220    1.3187

root =

    1.3187

errors =

    0.6400    0.0244    0.0000
```

>> [counter,iter,root,errors]=secant(f,-1,0,0.0001)

```
counter =

     1     2     3     4     5     6     7     8     9    10    11    12

iter =
```

-0.6364 -0.5410 -0.5810 -0.5635 -0.5710 -0.5678 -0.5691 -0.5685 -0.5688

-0.5687 -0.5687 -0.5687

root =

-0.5687

errors =

0.5289 0.2064 0.0929 0.0393 0.0171 0.0073 0.0032 0.0014 0.0006

0.0003 0.0001 0.0000

>> [counter,iter,root,errors]=secant(f,0,2,0.0001)

counter =

1 2 3 4 5 6 7 8 9

iter =

1.0541 1.2405 1.2973 1.3130 1.3172 1.3183 1.3186 1.3187 1.3187

root =

1.3187

errors =

1.7180 0.5664 0.1598 0.0431 0.0115 0.0030 0.0008 0.0002 0.0001

(c) I did each function in a separate figure (less confusion):

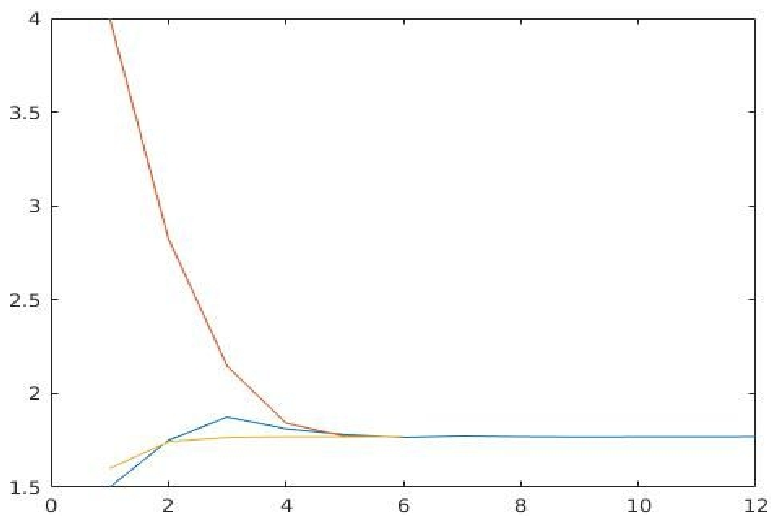
Blue: Bisection

Red: Newton's

Yellow: Secant

(i) f(x) = x^3 − 2x − 2, using this code:

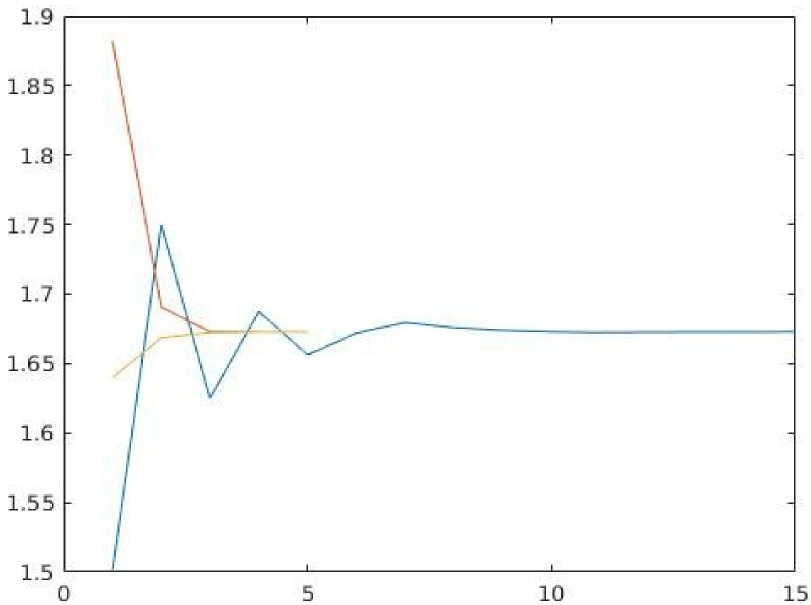
```
[counter,iter,root,errors] = bisection(f1,0,2,0.0001);

plot(counter,iter);

hold on

[counter,iter,root,errors]=newton(f1,f1_prime,1,0.0001);

plot(counter,iter);

hold on

[counter,iter,root,errors]=secant(f1,0,2,0.0001);

plot(counter,iter);

hold off
```



(ii) exp(x) + x − 7, using this code:

```
[counter,iter,root,errors] = bisection(f2,0,2,0.0001);

plot(counter,iter);

hold on
```

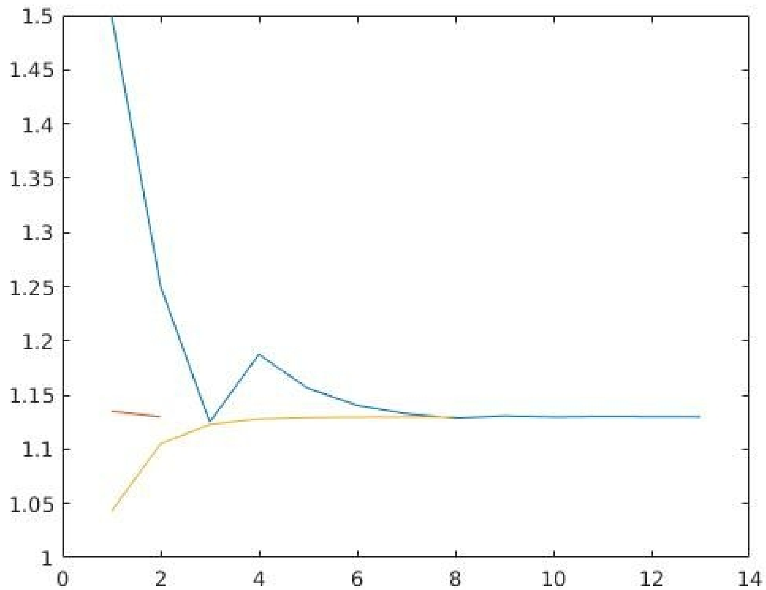
```
[counter,iter,root,errors]=newton(f2,f2_prime,1,0.0001);

plot(counter,iter);

hold on

[counter,iter,root,errors]=secant(f2,0,2,0.0001);

plot(counter,iter);

hold off
```



(iii) exp(x)+ sin(x)−4, using this code:

```
[counter,iter,root,errors] = bisection(f3,0,2,0.0001);

plot(counter,iter);

hold on

[counter,iter,root,errors]=newton(f3,f3_prime,1,0.0001);

plot(counter,iter);

hold on

[counter,iter,root,errors]=secant(f3,0,2,0.0001);
```

```
plot(counter,iter);
hold off
```



Note that based on all the three figures we can see that Newton's cost way less than the other two which makes it the best especially for small number of tolerance.

(d) Bisection converges linearly (the error is proportional to the error at the previous step).

Newton converges quadratically, if f'(x) = 0 then linear. (the error is proportional to the square of the previous error times "M.")

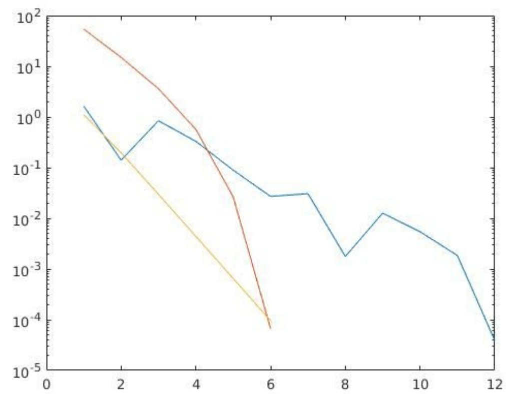
Secant converges superlinearly (the error is proportional to the previous error raised to the 1.62)

Blue: Bisection

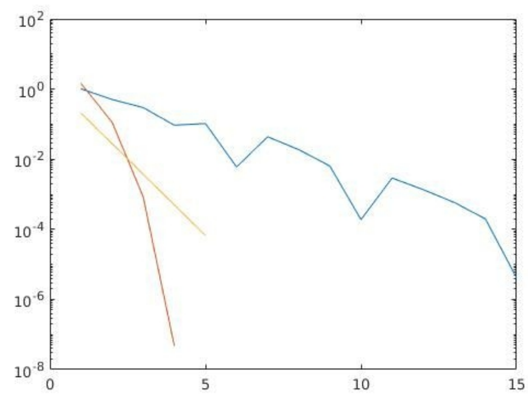
Red: Newton's

Yellow: Secant

(i) $f(x) = x^3 - 2x - 2$



(ii) $\exp(x) + x - 7$



(iii) $\exp(x) + \sin(x) - 4$