

ENGR 6101: COMPUTATIONAL ENGINEERING

Solution Set 2 (due Monday in class, 9/14)

Questions:

In problems 1-5, your number of iterations may differ by one compared to the solutions here. If this is only because of the way you enumerated iterations (indices starting with 0 vs. 1), then do not cut off any points.

The concentration of pollutant bacteria c in a lake decreases according to

$$c = 75e^{-1.5t} + 20e^{-0.075t}$$

Determine the time required for the bacteria concentration to be reduced to 15.

1. (3 pts.) Write a Matlab code to solve this problem using Newton's method with an initial guess of $t = 6$ and stopping criterion of 0.5% approximate relative error (equation 3.5 in textbook). At each iteration, your code should print (i) iteration number, (ii) approximated value of t , and (iii) estimated error¹. Print your code, and the output of your code, and include them in your solution. Name your code as `newton.m` and submit soft copy as an e-mail attachment to `caner@uga.edu`.

SOLUTION. Here is the code and the output:

```
clear all;
i = 1;
x(i) = 6;
error = 1;
disp(' iteration      x(i)      error  ')
while ( error > 0.005 )
    x(i+1) = x(i) - (75*exp(-1.5*x(i))+20*exp(-0.075*x(i))-15)/(-112.5*exp(-1.5*x(i))-1.5*exp(-0.075*x(i)));
    error = abs( ( x(i+1) - x(i) ) / x(i+1) );
    disp([ i, x(i+1), error ])
    i = i+1;
end
```

```
>> hw12_newton
 iteration      x(i)      error
    1.0000      3.6934      0.6245
    2.0000      3.9819      0.0725
    3.0000      4.0016      0.0049
>>
```

2. (3 pts.) Modify the code `bisection.m` given on the course website to solve the same problem using bisection method. Take your initial interval to be $[0, 6]$. Use the current interval length for error and stopping criterion of 0.5% approximate relative error. At each iteration, your code should print (i) iteration number, (ii) approximated value of t , and (iii) the error. Print your code, and the output of your code, and include both in your solution. Name your code as `bisection.m` and submit soft copy as an e-mail attachment to `caner@uga.edu`.

SOLUTION. Here is the code and the output(Give your self full-credit whether you have used relative error or absolute error).

¹You can use `disp` command to do this.

```

clear all;
xl = 0;
xr = 6;
i = 1;
error = 6;
disp(' iteration      x(i)      error  ')
while ( error > 0.005 )
    xm = ( xr + xl ) / 2;
    if (75*exp(-1.5*xl)+20*exp(-0.075*xl)-15)*(75*exp(-1.5*xm)+20*exp(-0.075*xm)-15) < 0
        xr = xm;
    else
        xl = xm;
    end
    error = error / 2;
    disp([ i, xm, error ])
    i = i + 1;
end

```

```

>> hw12_bisection
iteration      x(i)      error
     1         3         3
     2.0000    4.5000    1.5000
     3.0000    3.7500    0.7500
     4.0000    4.1250    0.3750
     5.0000    3.9375    0.1875
     6.0000    4.0312    0.0938
     7.0000    3.9844    0.0469
     8.0000    4.0078    0.0234
     9.0000    3.9961    0.0117
    10.0000    4.0020    0.0059
    11.0000    3.9990    0.0029
>>

```

3. (3 pts.) Write a Matlab code to solve the same problem using secant method with the same stopping criterion as in 1. Start at $x_1 = 6$, $x_2 = 5$. At each iteration, your code should print (i) iteration number, (ii) approximated value of t, and (iii) estimated error. Print your code, and the output of your code, and include both in your solution. Name your code as `secant.m` and submit soft copy as an e-mail attachment to `caner@uga.edu`.

SOLUTION. Here is the code and the output:

```

clear all;
i = 1;
x(1) = 6;
x(2) = 5;
error = 1;
disp(' iteration      x(i)      error  ')
while ( error > 0.005 )

```

```

f_x_1 = 75*exp(-1.5*x(i+1))+20*exp(-0.075*x(i+1))-15;
f_x_0 = 75*exp(-1.5*x(i))+20*exp(-0.075*x(i))-15;
x(i+2) = x(i+1) - f_x_1 * ( x(i+1) - x(i) ) / (f_x_1 - f_x_0);
error = abs( ( x(i+2) - x(i+1) ) / x(i+2) );
disp([ i, x(i+2), error ])
i = i+1;
end

```

```

>> hw12_secant
iteration    x(i)      error
   1.0000    3.8174    0.3098
   2.0000    4.0296    0.0527
   3.0000    4.0026    0.0067
   4.0000    4.0016    0.0002
>>

```

4. (3 pts.) Write a Matlab code to solve the same problem using fixed point iterations with the same stopping criterion as in 1. Start at $x = 6$. At each iteration, your code should print (i) iteration number, (ii) approximated value of t , and (iii) estimated error. Print your code, and the output of your code, and include both in your solution. Name your code as `fixed_point.m` and submit soft copy as an e-mail attachment to `caner@uga.edu`.

SOLUTION. Here is the code and the output:

```

clear all;
i = 1;
x(i) = 6;
error = 1;
disp(' iteration    x(i)      error  ')
while ( error > 0.005 )
    x(i+1) = (75*exp(-1.5*x(i))+20*exp(-0.075*x(i))-15+x(i));
    error = abs( ( x(i+1) - x(i) ) / x(i+1) );
    disp([ i, x(i+1), error ])
    i = i+1;
end

```

```

>> hw12_fixed
iteration    x(i)      error
   1.0000    3.7618    0.5950
   2.0000    4.1110    0.0849
   3.0000    3.9619    0.0376
   4.0000    4.0175    0.0138
   5.0000    3.9955    0.0055
   6.0000    4.0040    0.0021
>>

```

5. (3 pts) Re-run your codes as follows:
- Use $x = 0$ as the initial value for Newton.

- Use $x = 0$ as the initial value for fixed point iteration.
- Use $x = 0, 1$ as the initial values for the secant method.
- Use $[0, 10]$ as the initial interval for bisection method.

Print the just the outputs (not the codes) and include them in your solution. How many iterations does each one take (with the same error)? Which one converges faster? Which method is easier to implement in Matlab? If you were given a choice, which method would you use?

SOLUTION. Here is the output:

```
>> hw12_newton
iteration    x(i)      error
  1.0000    0.7018    1.0000
  2.0000    1.4428    0.5136
  3.0000    2.2532    0.3597
  4.0000    3.1251    0.2790
  5.0000    3.8053    0.1788
  6.0000    3.9940    0.0472
  7.0000    4.0016    0.0019
>> hw12_fixed
iteration    x(i)      error
   1      80     1
  2.0000    65.0496    0.2298
  3.0000    50.2017    0.2958
  4.0000    35.6650    0.4076
  5.0000    22.0433    0.6180
  6.0000    10.8719    1.0276
  7.0000     4.7212    1.3028
  8.0000     3.8205    0.2358
  9.0000     4.0810    0.0639
 10.0000     3.9722    0.0274
 11.0000     4.0133    0.0102
 12.0000     3.9971    0.0040
>> hw12_secant
iteration    x(i)      error
  1.0000    1.3398    0.2536
  2.0000    1.9643    0.3179
  3.0000    2.5222    0.2212
  4.0000    3.1405    0.1969
  5.0000    3.6534    0.1404
  6.0000    3.9312    0.0707
  7.0000    3.9968    0.0164
  8.0000    4.0016    0.0012
>> hw12_bisection
iteration    x(i)      error
   1       5     5
  2.0000    2.5000    2.5000
  3.0000    3.7500    1.2500
  4.0000    4.3750    0.6250
```

5.0000	4.0625	0.3125
6.0000	3.9062	0.1562
7.0000	3.9844	0.0781
8.0000	4.0234	0.0391
9.0000	4.0039	0.0195
10.0000	3.9941	0.0098
11.0000	3.9990	0.0049

>>

Newton is the fastest, followed by Secant. Bisection and Fixed point method are slower. (The following is based on personal preferences and observations so please do not cut off points even if what you wrote is different than the following text.) Fixed point method is rather unpredictable, the error keeps increasing and decreasing from one iteration to the other, while the bisection method shows a steady decrease in error. Fixed point is very easy to implement. Newton and bisection are easy too. I would go with Newton if the function is smooth, and the derivative is available. If the derivative is not available, I would go with secant. For non-smooth functions, I would go with bisection. Fixed point method is too unpredictable to be useful.