

Simulation Assignment 2 Report

ECE 204
October 29, 2023
Punit Shah and Dylan Nogueira

Initial Notes

-> In the section for testerror, our value for the second testerror (0.005%) we had the value at 0.000005 instead of 0.00005, which we remedied in the .m files.

-> In the output section, we are not sure why the output is formatted as such, but it is outputting the correct values (as far as we know)

Codes

Initialization.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Punit Shah and Dylan Nogueira %
%   Group 5 - Section 205   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

% A, B and C values to be used

A = 3.9083e-3;

B = -5.775e-7;

C = -4.183e-12;

% Input resistance

R = input("Please input resistance value: ");

% Use syms so we can utilize T universally

syms T

% Used for bisection method

fx = 100 * (1 + A * T + B * T^2 + C * (T - 100) * T^3) - R;

gx = 100 * (1 + A * T + B * T^2) - R;

% Used for Newton - Raphson (used diff to find the derivative of the
% original equations)

diffx = diff(100 * (1 + A * T + B * T^2 + C * (T - 100) * T^3) - R);

diffgx = diff(100 * (1 + A * T + B * T^2) - R);

% Used for fixed point iteration

fnew = ((R/100)- 1 -B*T^2 + 100*C*T^3 - C*T^4)/A;

gnew = ((R/100)- 1 - B*T^2)/A;

% Choose which equations to use depending on resistance entered

if(R <= 100)

left = -200;

right = 0;

% Call the bisection, onepoint and raphson method and their [return
% values]

[bitemp ,biiter ,bierror] = bisection(fx, left, right, T);

[raptemp, rapiter, raperror] = raphson(fx, diffx, -100, T);

```

[fixtemp, fixiter, fixerror] = fixed(fnew, -100, T);

else
    left = 0;
    right = 850;
    % call the bisection, onepoint and raphson method and their [return
    % values]
    [bitemp, biiter, bierror] = bisection(gx, left, right, T);
    [raptemp, rapiter, raperror] = raphson(gx, diffgx, 425, T);
    [fixtemp, fixiter, fixerror] = fixed(gnew, 300, T);
end

% display all the values
disp(["The temperature obtained by bisection is " num2str(bitemp) "C"]);
disp(["The temperature obtained by fixed point is " num2str(fixtemp) "C"]);
disp(["The temperature obtained by NR is " num2str(raptemp) "C"]);
disp(" ");
disp(["The number of required iterations for bisection is " num2str(biiter)]);
disp(["The number of required iterations for fixed point is " num2str(fixiter)]);
disp(["The number of required iterations for NR is " num2str(rapiter)]);
disp(" ");
disp(["The absolute relative approximate error % for bisection is " num2str(bierror) "%"]);
disp(["The absolute relative approximate error % fixed point is " num2str(fixerror) "%"]);
disp(["The absolute relative approximate error % NR is " num2str(raperror) "%"]);

```

bisection.m

%%

% Punit Shah and Dylan Nogueira %

% Group 5 - Section 205 %

%%

% reallocate the return values so they can be called in initialization.m

function[temp, iter, error] = bisection(F, left, right, sym)

% find initial midpoint

mid = (left + right)/2;

% using a boolean instead of a error value because it wasn't working with
% an error value while loop :)

done = false;

% iteration count number

iterations = 0;

% test error value, uncomment and comment accordingly

testerror = 0.0005;

% testerror = 0.000005

% Solve using bisection method

while done == false

% find the leftval by plugging left into the equation in initialization.m

leftval = double(subs(F, sym, left));

% find the midval by plugging left into the equation in initialization.m

midval = double(subs(F, sym, mid));

% Update midpoint so we can use the oldmiddle in the error calculation

oldmiddle = mid;

% MAKE CASE FOR IF IT GUESSES ROOT INSTANTLY?

% Check for root locations

if(leftval*midval > 0)

% change the middle value to be between the old middle value and
% the right value

mid = (right + mid)/2;

% change left value to be the old middle value

left = oldmiddle;

elseif(leftval*midval < 0)

% change the middle value to be between the old middle value and

```

    % the left value
    mid = (left + mid)/2;
    % change the right value to be the old middle value;
    right = oldmiddle;
end

% calculate error values
Error = abs((mid-oldmiddle)/mid);
if(Error < testerror)
    % make the boolean return true, process terminates
    done = true;
end

% Update the iteration counter
iterations = iterations + 1;
end

% reallocate the values to be used in initialization.m
temp = mid;
iter = iterations;
% multiply error by 100 because we are using decimal values
error = Error * 100;

end

```

raphson.m

%%

% Punit Shah and Dylan Nogueira %

% Group 5 - Section 205 %

%%

% reallocate the return values so they can be called in initialization.m

function [temp, iter, error] = raphson(F, diffF, initial,sym)

% setting our initial guess

initialroot = initial;

% using a boolean instead of a error value because it wasn't working with

% an error value while loop :)

done = false;

% iteration count number

iterations = 0;

% test error value, uncomment and comment accordingly

testerror = 0.0005;

% testerror = 0.000005

while done == false

 % plug and chug formula for newton - raphson method

 secondaryroot = initialroot - double(subs(F, sym, initialroot) / subs(diffF, sym, initialroot));

 % calculate error values

 Error = abs((secondaryroot - initialroot) / secondaryroot);

 if(Error < testerror)

 % make the boolean return true, process terminates

 done = true;

 end

 % Update previous guess if not broken out of loop

 initialroot = secondaryroot;

 % Update the iteration counter

 iterations = iterations + 1;

end

% reallocate the values to be used in initialization.m

temp = secondaryroot;

iter = iterations;

% multiply error by 100 because we are using decimal values

error = Error * 100;

end

fixed.m

%%

% Punit Shah and Dylan Nogueira %

% Group 5 - Section 205 %

%%

% reallocate the return values so they can be called in initialization.m

function [temp, iter, error] = fixed(F, initial, sym)

% iteration count number

iterations = 0;

% setting our initial guess

oldtemp = initial;

% test error value, uncomment and comment accordingly

testerror = 0.0005;

% testerror = 0.000005

% using a boolean instead of a error value because it wasn't working with

% an error value while loop :)

done = false;

while(done == false)

% plug and chug formula for fixed point iteration

tempnew = double(subs(F, sym, oldtemp));

% calculate error values

Error = abs((tempnew - oldtemp)/tempnew);

if (Error < testerror)

% make the boolean return true, process terminates

done = true;

end

% Update previous guess if not broken out of loop

oldtemp = tempnew;

% Update the iteration counter

iterations = iterations + 1;

end

% reallocate the values to be used in initialization.m

temp = tempnew;

iter = iterations;

% multiply error by 100 because we are using decimal values

error = Error * 100;

end

Screenshots of Code

Initialization.m

```
fixed.m x bisection.m x initialization.m x raphson.m x +
1      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2      % Punit Shah and Dylan Nogueira %
3      %      Group 5 - Section 205      %
4      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6      % A, B and C values to be used
7 -    A = 3.9083e-3;
8 -    B = -5.775e-7;
9 -    C = -4.183e-12;
10
11     % Input resistance
12 -    R = input("Please input resistance value: ");
13
14     % Use syms so we can utilize T universally
15 -    syms T
16     % Used for bisection method
17 -    fx = 100 * (1 + A * T + B * T^2 + C * (T - 100) * T^3) - R;
18 -    gx = 100 * (1 + A * T + B * T^2) - R;
19
20     % Used for Newton - Raphson (used diff to find the derivative of the
21     % original equations)
22 -    diffx = diff(100 * (1 + A * T + B * T^2 + C * (T - 100) * T^3) - R);
23 -    diffgx = diff(100 * (1 + A * T + B * T^2) - R);
24
25     % Used for fixed point iteration
26 -    fnew = ((R/100)- 1 -B*T^2 + 100*C*T^3 - C*T^4)/A;
27 -    gnew = ((R/100)- 1 - B*T^2)/A;
28
29     % Choose which equations to use depending on resistance entered
30 -    if(R <= 100)
31 -        left = -200;
32 -        right = 0;
33 -        % Call the bisection, onepoint and raphson method and their [return
34 -        % values]
35 -        [bitemp ,biiter ,bierror] = bisection(fx, left, right, T);
36 -        [raptemp, rapiter, raperror] = raphson(fx, diffx, -100, T);
37 -        [fixtemp, fixiter, fixerror] = fixed(fnew, -100, T);
38
```



```

else
    left = 0;
    right = 850;
    % call the bisection, onepoint and raphson method and their [return
    % values]
    [bitemp ,biiter ,bierror] = bisection(gx, left, right, T);
    [raptemp, rapiter, raperror] = raphson(gx, diffgx, 425, T);
    [fixtemp, fixiter, fixerror] = fixed(gnew, 300, T);
end

% display all the values
disp(["The temperature obtained by bisection is " num2str(bitemp) "C"]);
disp(["The temperature obtained by fixed point is " num2str(fixtemp) "C"]);
disp(["The temperature obtained by NR is " num2str(raptemp) "C"]);
disp(" ");
disp(["The number of required iterations for bisection is " num2str(biiter)]);
disp(["The number of required iterations for fixed point is " num2str(fixiter)]);
disp(["The number of required iterations for NR is " num2str(rapiter)]);
disp(" ");
disp(["The absolute relative approximate error % for bisection is " num2str(bierror) "%"]);
disp(["The absolute relative approximate error % fixed point is " num2str(fixerror) "%"]);
disp(["The absolute relative approximate error % NR is " num2str(raperror) "%"]);

```

bisection.m

```
fixed.m x bisection.m x initialization.m x raphson.m x +
1      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2      % Punit Shah and Dylan Nogueira %
3      %      Group 5 - Section 205      %
4      %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6      % reallocate the return values so they can be called in initialization.m
7      function[temp, iter, error] = bisection(F, left, right, sym)
8
9      % find initial midpoint
10     mid = (left + right)/2;
11
12     % using a boolean instead of a error value because it wasn't working with
13     % an error value while loop :)
14     done = false;
15
16     % iteration count number
17     iterations = 0;
18
19     % test error value, uncomment and comment accordingly
20     testerror = 0.0005;
21     % testerror = 0.000005
22
23     % Solve using bisection method
24     while done == false
25         % find the leftval by plugging left into the equation in initialization.m
26         leftval = double(subs(F, sym, left));
27         % find the midval by plugging left into the equation in initialization.m
28         midval = double(subs(F, sym, mid));
29
30         % Update midpoint so we can use the oldmiddle in the error calculation
31         oldmiddle = mid;
32
33         % MAKE CASE FOR IF IT GUESSES ROOT INSTANTLY?
34
```

```

% Check for root locations
if(leftval*midval > 0)
    % change the middle value to be between the old middle value and
    % the right value
    mid = (right + mid)/2;
    % change left value to be the old middle value
    left = oldmiddle;

elseif(leftval*midval < 0)
    % change the middle value to be between the old middle value and
    % the left value
    mid = (left + mid)/2;
    % change the right value to be the old middle value;
    right = oldmiddle;
end

% calculate error values
Error = abs((mid-oldmiddle)/mid);
if(Error < testerror)
    % make the boolean return true, process terminates
    done = true;
end

% Update the iteration counter
iterations = iterations + 1;
end

% reallocate the values to be used in initialization.m
temp = mid;
iter = iterations;
% multiply error by 100 because we are using decimal values
error = Error * 100;

end

```

raphson.m

```
fixed.m  x bisection.m  x initialization.m  x raphson.m  x +
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Punit Shah and Dylan Nogueira %
3  %      Group 5 - Section 205      %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6  % reallocate the return values so they can be called in initialization.m
7  function [temp, iter, error] = raphson(F, diffF, initial,sym)
8
9  % setting our initial guess
10 initialroot = initial;
11
12 % using a boolean instead of a error value because it wasn't working with
13 % an error value while loop :)
14 done = false;
15 % iteration count number
16 iterations = 0;
17
18 % test error value, uncomment and comment accordingly
19 testerror = 0.0005;
20 % testerror = 0.000005
21
22 while done == false
23     % plug and chug formula for newton - raphson method
24     secondaryroot = initialroot - double(subs(F, sym, initialroot) / subs(diffF, sym, initialroot));
25
26     % calculate error values
27     Error = abs((secondaryroot - initialroot) / secondaryroot);
28     if(Error < testerror)
29         % make the boolean return true, process terminates
30         done = true;
31     end
32
33     % Update previous guess if not broken out of loop
34     initialroot = secondaryroot;
35     % Update the iteration counter
36     iterations = iterations + 1;
37 end
38
39 % reallocate the values to be used in initialization.m
40 temp = secondaryroot;
41 iter = iterations;
42 % multiply error by 100 because we are using decimal values
43 error = Error * 100;
44
45 end
```

fixed.m

```
fixed.m x bisection.m x initialization.m x raphson.m x +
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  % Punit Shah and Dylan Nogueira %
3  %      Group 5 - Section 205      %
4  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
5
6  % reallocate the return values so they can be called in initialization.m
7  function [temp, iter, error] = fixed(F, initial, sym)
8
9  % iteration count number
10 iterations = 0;
11
12 % setting our initial guess
13 oldtemp = initial;
14
15 % test error value, uncomment and comment accordingly
16 testerror = 0.0005;
17 % testerror = 0.000005
18
19 % using a boolean instead of a error value because it wasn't working with
20 % an error value while loop :)
21 done = false;

while(done == false)

    % plug and chug formula for fixed point iteration
    tempnew = double(subs(F, sym, oldtemp));
    % calculate error values
    Error = abs((tempnew - oldtemp)/tempnew);
    if (Error < testerror)
        % make the boolean return true, process terminates
        done = true;
    end

    % Update previous guess if not broken out of loop
    oldtemp = tempnew;
    % Update the iteration counter
    iterations = iterations + 1;
end

% reallocate the values to be used in initialization.m
temp = tempnew;
iter = iterations;
% multiply error by 100 because we are using decimal values
error = Error * 100;

end
```

Command Window Outcomes

55 ohms

maxerror = 0.0005 (0.05%)

```
Command Window

>> initialization
Please input resistance value: 55
    "The temperature obtained by bi..."    "-112.9395"    "C"

    "The temperature obtained by fi..."    "-112.928"    "C"

    "The temperature obtained by NR..."    "-112.927"    "C"

    "The number of required iterations for bisection is "    "11"

    "The number of required iterations for fixed point ..."    "3"

    "The number of required iterations for NR is "    "2"

    "The absolute relative approxim..."    "0.043234"    "%"

    "The absolute relative approxim..."    "0.021069"    "%"

    "The absolute relative approxim..."    "0.035842"    "%"

>>
```

maxerror = 0.00005 (0.005%)

Please input resistance value: 55

"The temperature obtained by bi..." "-112.9242" "C"

"The temperature obtained by fi..." "-112.927" "C"

"The temperature obtained by NR..." "-112.927" "C"

"The number of required iterations for bisection is " "15"

"The number of required iterations for fixed point ..." "4"

"The number of required iterations for NR is " "3"

"The absolute relative approxim..." "0.0027025" "%"

"The absolute relative approxim..." "0.00091919" "%"

"The absolute relative approxim..." "3.697e-07" "%"

f_x >>


265 ohms

maxerror = 0.0005 (0.05%)


```
Please input resistance value: 265
    "The temperature obtained by bi..."    "452.6001"    "C"
    "The temperature obtained by fi..."    "452.4182"    "C"
    "The temperature obtained by NR..."    "452.4235"    "C"

    "The number of required iterations for bisection is "    "11"
    "The number of required iterations for fixed point ..."    "5"
    "The number of required iterations for NR is "    "2"

    "The absolute relative approxim..."    "0.045851"    "%"
    "The absolute relative approxim..."    "0.0075893"    "%"
    "The absolute relative approxim..."    "0.02809"    "%"
```

 >> |

maxerror = 0.00005 (0.005%)

 Command Window

```
>> initialization
Please input resistance value: 265
    "The temperature obtained by bi..."    "452.4315"    "C"

    "The temperature obtained by fi..."    "452.4228"    "C"

    "The temperature obtained by NR..."    "452.4235"    "C"

    "The number of required iterations for bisection is "    "15"

    "The number of required iterations for fixed point ..."    "6"

    "The number of required iterations for NR is "    "3"

    "The absolute relative approxim..."    "0.0028667"    "%"

    "The absolute relative approxim..."    "0.0010147"    "%"

    "The absolute relative approxim..."    "6.0888e-07"    "%"
```

 >>