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 | bisection.m | initialization.m | raphson.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

fixed.m | bisection.m | initialization.m | 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] = 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

fixed.m | bisection.m | initialization.m | 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

fixed.m | bisection.m | initialization.m | 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] = 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
```

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"    "%"

fx >>
```

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"    "%"

fx >>
```

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"    "%"

fx >>
```