Linh Tang

CEE/MAE M20

UID: 205542275

Nov 7, 2020

HOMEWORK 5

1. Finding roots of nonlinear equations

1.1 Bisection method

1.1.1 Introduction

The goal of this part is to find the root of the nonlinear equation f(x) = 1 + 0.5sin(x) – x = 0 in the range x ∈ (1, 2) by using bisection method. In the command window, the program displays the root and the number of iterations needed to find that root.

1.1.2 Model and methods

Firstly, a function handle is used to define the given equation. The function takes x as an input and returns the value of f(x) as an output.

```
function fx = fun1(x)
%Inputs:
%x: a point at which to evaluate the value of the function
% Outputs:
%fx: the numerical value of the function

fx = 1+ 0.5*sin(x) - x;

end
```

After that, a function for the bisection method is written, which takes function f(x), the a and b of interval [a,b] and the tolerance as inputs. The bisection function returns the root of the function f(x) and the number of iterations after which the root is found.

The idea of the bisection method is there is a root x ∈ (a, b), when f(x) is continuous on [a,b] and f(a).f(b) <0. When fa and fb have opposite signs, it shows one is above the x-axis, and one I below the x-axis; therefore, the plot of fx will cut the x- axis at the value of root x.

The maximum number of iterations is determined by the following equation where $\epsilon 0 = b - a$.

$$nMax = \log_2\left(\frac{\epsilon_0}{tol}\right) = \frac{\log(\epsilon_0) - \log(tol)}{\log(2)}$$

```
function [xRoot, numIter] = biSection(f, a, b, tol)

   eps = b - a;

   nMax = (log(eps) - log(tol))/log(2);
   numIter = 1;

   while
     …
   end

 end
```

The while loop is used to iterate the bisection method until the program gets a midpoint which is close enough to be the root.

The xRoot is first defined as the mean between a and b. If the value of f(xRoot) is equal to 0 or (b-a)/2 is very small, then it breaks the while loop and outputs the value of xRoot and number of iterations. Otherwise, the loop will continue. If the sign of f(xRoot) is the same with the sign of f(a) or the sign of f(b), the new values are set for a or b.

```
while (numIter <= nMax)

       xRoot = (a+b)/2;

       if( f(xRoot) == 0 || eps/2 < tol)

           break;

       end

       numIter = numIter + 1;

       if(sign(f(xRoot)) == sign(f(a)))
           a = xRoot;
       else
           b =xRoot;
       end

 end
```

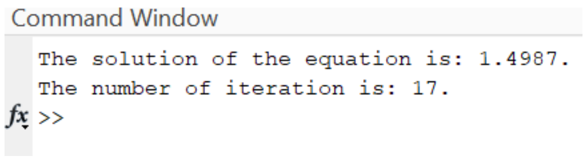
To run the program, constants are defined, and bisection function is called at the beginning of the script.

```
a = 1;
b = 2;
tol = 1e-5;
[root, numIter] = biSection(@fun1, a, b, tol);

fprintf('The solution of the equation is: %.4f. \n',
root);
fprintf('The number of iteration is: %d. \n',
numIter);
```

1.1.3 Result

The root of function 1+0.5sin(x) – x = 0 is 1.4987 with 17 iterations.



```
The solution of the equation is: 1.4987.
The number of iteration is: 17.
>>
```

1.1.4 Discussion

This method gives out a correct solution for the nonlinear equation 1 + sin(x) – x =0. The bisection method is a simple and is guaranteed for the convergence in finding real roots of the equation, but this method requires 1 +

sin(x) – x continuous in the interval (1,2). In this script, all functions are put in separate files, but they might be put in the main script, at the bottom. In the future, the script can be improved by asking users to input tolerance and the interval of the problem.

1.2 Fixed point iteration method

1.2.1 Introduction

This part is finding the root of nonlinear equations by using fixed point iteration method this method converts f(x) = 0 into the form x = g(x), now x is the fixed point that the program looks for. This script is used to solve two equations 1 + 0.5 sin(x) – x =0 and 3 + 2 sin(x) -x = 0, and check if the fixed point and the root are converged

1.2.2 Model and methods

Before start getting into the iteration, the function is converted into the form x* = g(x*). Now, the program solves for the fixed-point x* in the g(x) function instead of solving x in the function 1 + 0.5 sin(x) – x =0. The function handle for g(x) takes a variable x as an input and outputs the value of g(x) at that input value.

```
function gx = fun2(x)
% Inputs:
% x: a point at which to evaluate the value of the function
% Outputs:
% gx: the numerical value of the function

    gx = 1 + 0.5*sin(x);

end
```

A function named fixed-point takes four inputs: function g(x), the initial guess of the fixed point, the tolerance, and the maximum number of iterations. This function returns the fixed point xStar and the root xRoot.

The problem is solved by using the fixed-point iteration method and Newton's iteration; this process requires an initial guess for the fixed point.

```
        xStar = x0;

        iter = 1;
```

The program uses a while loop to find xStar when it is convergent. There is an if condition to break up the while loop and gives out the output. At the end of the loop, iter get updated before getting into the next iteration.

If xStar is convergent, xRoot is equal to xStar.

```
while(abs(fun2(xStar) - xStar) > tol && iter < maxIter)
      xStar = fun2(xStar);
      if (abs(fun2(xStar) - xStar) <= tol)
            break;
      end
      iter = iter + 1;
end
xRoot = xStar;
```

Outside the loop, there is an if condition for the case that xStar does not converge when the iteration exceeds the maximum number of iterations; the program prints out a statement.

```
if(abs(xStar - xStar) > tol && iter >= maxIter)
        fprintf('* xStar did not converge with tolerance =
%.1e and %d iterations. \n The following xRoot is not
correct.\n', tol, maxIter);

end
```

To run the program, constants are defined at the top of the script. After that, the function fixPoint is called to solve the function g(x).

Here is the code for solving the equation x = 1 + 0.5sin(x); fun2 is passed by reference in the fixPoint function.

```
x0= 0;
tol = 1e-5;
maxIter = 100;

[xStar1, xRoot1] = fixPoint(@fun2, x0, tol, maxIter);
fprintf('Part ii: Fixed point method for equation: 1 +
0.5sin(x) - x =0 \n');
fprintf('Initial guess = 0.\n');
fprintf ('- xRoot is: %.5f. \n', xRoot1);
fprintf ('- xStar is: %.5f. \n\n', xStar1);
```

To solve the equation x = 3 + 2sin(x), the logic is similar, but the function is solved in two initial guessing: 0 and 3. A handle function is created for this function; fun2 can not be used since it is a completely different function.

```
guess = 3;

hx = @(x) 3 + 2*sin(x);

fprintf('Part iii: Fixed point method for equation: 3 +
2sin(x) - x =0 \n');
fprintf('1) Initial guess = 3.\n');
[xStar2, xRoot2] = fixPoint(hx, guess, tol, maxIter);

fprintf ('- xRoot is: %.5f. \n', xRoot2);
fprintf ('- xStar is: %.5f. \n\n', xStar2);

fprintf('2) Initial guess = 0.\n');
[xStar3, xRoot3] = fixPoint(hx, x0, tol, maxIter);

fprintf ('xRoot is: %.5f. \n', xRoot3);
fprintf ('xStar is: %.5f. \n', xStar3);
```

To visualize the problem and intersections, the program is designed to graph 3 function: y =x, g(x) = 1 + 0.5sin(x), and h(x) = 3 + 2 sin(x).

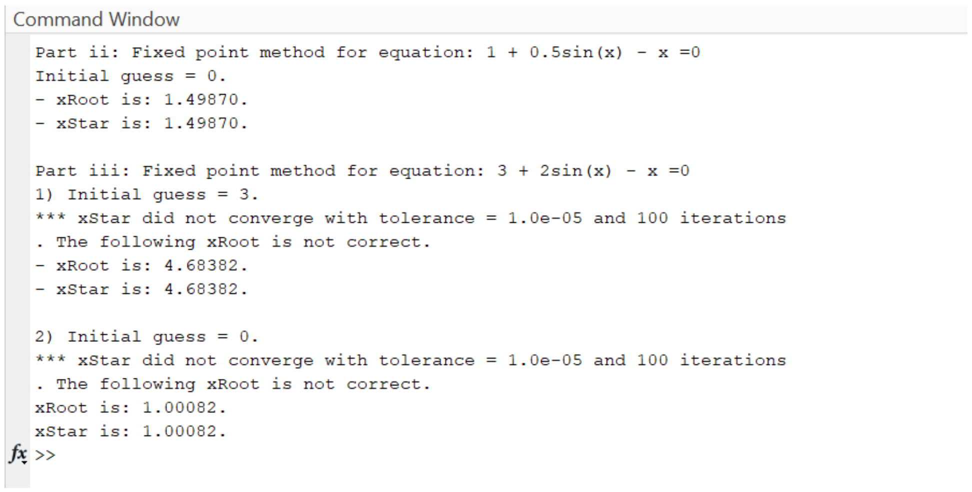
```
figure(1)
p = plot(x,y,'b',x,g,'r',x,h,'g');
set(p, 'LineWidth', 2);
set(gcf, 'Position', [100 40 1000 600]);
set(gca, 'LineWidth', 2, 'FontSize', 15);
```

```
xticks(0:1:10);
yticks(0:1:10);
legend('y = x','g(x) = 1 + 0.5sin(x)','h(x) = 3 +
2sin(x)');
xlabel ('Domain');
ylabel ('Range');
box on
grid on
```

1.2.3 Result

This method works well when solving for the fixed point and root for the equation 1 + 0.5sin(x) – x = 0, giving the root is 1.49870 which is also its fixed point.

However, the simulation for the equation 3 + 2sin(x) – x = 0 is incorrect since the function is divergent.



```
Part ii: Fixed point method for equation: 1 + 0.5sin(x) - x =0
Initial guess = 0.
- xRoot is: 1.49870.
- xStar is: 1.49870.

Part iii: Fixed point method for equation: 3 + 2sin(x) - x =0
1) Initial guess = 3.
*** xStar did not converge with tolerance = 1.0e-05 and 100 iterations
. The following xRoot is not correct.
- xRoot is: 4.68382.
- xStar is: 4.68382.

2) Initial guess = 0.
*** xStar did not converge with tolerance = 1.0e-05 and 100 iterations
. The following xRoot is not correct.
xRoot is: 1.00082.
xStar is: 1.00082.
>>
```

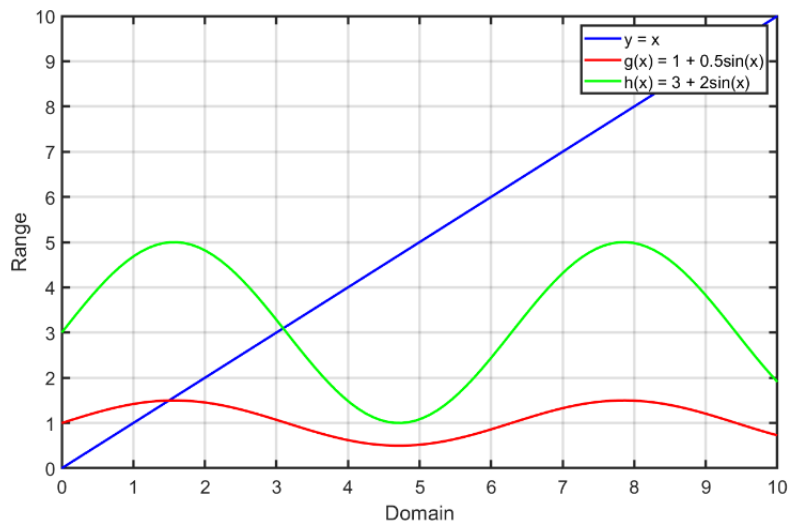


Figure1. The graph of functions g(x), h(x) and y =x.

According to the graph, the intersection of g(x) and y=x is at around 1.5 which matches with the calculation. The intersection between h(x) and y=x is around 3.1 which is different from what is obtained from the program.

1.2.4 Discussion

The fixed-point iteration is another method to solve nonlinear equation; however, this iteration method may or may not converge. When the function f(x) = 0 is converted to the fixed-point form x = g(x), g(x) must be continuous on the interval [a, b]. To see if the fixed-point iteration whether converge or not, g(x) and g'(x) are continuous and |g'(x)| < 1.

For g(x) = 1+ 0.5sin(x), the range of g(x) is from 0.5 to 1.5 for every x values, the xStar will be in the interval [0.5,1.5]. By taking the derivative of g(x), g'(x) = 0.5cos (x) is obtained, thus |g'(x)| < 1 as a result. With this calculation, it shows the fixed-point iteration for g(x) = 1 + 0.5sin(x) is converge; the xStar is also the xRoot of function f(x).

Similarly, the range of h(x) = 3 + 2sin(x) is from 1 to 5 for every x values; therefore, the solution will be in the interval [1,5]. When x = 3 which is in the interval, the fixed-point iteration is divergence because |h'(3)| = |2cos(3)| = 1.98; this number is greater than 1. The fixed- point iteration for h(x) = 3 + 2sin(x) is not converge for both initial guessing x0 = 0 and x0 = 3. According to the outputs, xStar is equal to 1.00082 with the initial guess is 0; xStar is equal to 4.68382 when the initial guess is 3. Since the function is not converge, xRoot cannot be set as equal to xStar. With any values for initial guess, the fixed point reaches the point that the value jumps back and forth between two values, 1.00082 and 4.68382.

In this part, the problem I encountered was I could not find the way to calculate xRoot based on xStar when it was divergent. The only thing I could do was printing a statement saying that it is divergence. I was thinking about calling the bisection function in this part to get the xRoot when the xRoot cannot be obtained, but I thought it was not allowed for this fixedPoint function.