MATH 3340 - Scientific Computing Assignment 2

Libao Jin

October 2, 2020

Please note that the deadline will be enforced as per the first homework. Remember that you are allowed to work in teams of two on this assignment. You are encouraged to prepare your work in LaTeX; a template will be provided to help you put it all together. In case you work in a team, both members of a team should submit, through WyoCourses, the PDF file containing their work and showing both their names.

Instruction

1. Go to `https://www.overleaf.com` and sign in (required).
2. Open template, click *Menu* (up left corner), then *Copy Project*.
3. Go to `LaTeX/meta.tex` (the file `meta.tex` under the folder `LaTeX`) to change the section and your name, e.g.,
   - change title to `\title{MATH 3340-01 Scientific Computing Homework 2}`
   - change author to `\author{Albert Einstein \& Carl F. Gauss}`
4. For Problem 1, 2, 3, 4, you need to write function/script files and store results to output files. Here are suggested names for function files, script files, and output files:

Problem	Function File	Script File	Output File
1	`bisection.m`	`hw2_p1.m`	`hw2_p1.txt`
2	`newton.m`	`hw2_p2.m`	`hw2_p2.txt`
3 (a)	`bisectionImproved.m`	`hw2_p3_a.m`	`hw2_p3_a.txt`
3 (b)	`newton.m`	`hw2_p3_b.m`	`hw2_p3_b.txt`
4	`bisectionRecursive.m`	`hw2_p4.m`	`hw2_p4.txt`

   Once finished, you need to upload these files to the folder `src` on Overleaf. If you have different filenames, please update the filenames in `\lstinputlisting{../src/your_script_name.m}` accordingly. You can use the MATLAB script `src/save_output.m` to generate the output files automatically (the script filenames should be exactly same as listed above). You can code in the provided files in hw2.zip. You can type `save_output` in the Command Window to call scripts above and save the output to corresponding files (Make sure the `save_output.m` is in the current working directory).
5. For Problem 3(a), 3(b), 5, you can either print the recompiled PDF out and write your solution/explanation there or directly type the answer in `body.tex` on Overleaf and print it out at the end.
6. Recompile, download, and print out the generated PDF.
7. You may find LaTeX.Mathematical.Symbols.pdf and the second part of Lab 01 Slides and Lab 02 Slides helpful.

1 Problem 1

Write your own MATLAB function that uses the bisection method to find the root of $f(x)$. The function header for the bisection method could read:

```
function [r, iters] = bisection(f, xL, xR)
```

where the inputs are:

```
f  = the function f(x) for which you want to find the root
xL = left limit of the interval
xR = right limit of the interval
```

and the outputs are:

```
r     = the root
iters = number of iterations performed
```

This is a *minimal* list of inputs and outputs required for your function. Additional inputs or outputs may be necessary and they are left to your discretion. Use this code to find the root of the piecewise function:

$$f(x) = \begin{cases} x^3 + 3x + 1 & \text{if } x \leq 0, \\ 1 + \sin(x) & \text{if } x > 0, \end{cases}$$

on the interval $x_L = -2$, $x_R = 0.1$. For this problem submit

- Your function file defining the bisection method
- The script file that calls this function
- A text file that contains the following output:
  - the value of the root x^*
  - the value of $f(x^*)$
  - the number of iterations performed

  The output must clearly identify each one of these results.

Note: Your script file should define the inputs of your function, call your function, and contain any additional formatting needed for pretty-printing the output.

Solution.

- Function file `bisection.m`

```
1  function [r, iters] = bisection(f, xL, xR)
2
3  %BISECTION: Bisection Method
4  % Syntax: [r, iters] = bisection(f, xL, xR)
5  % Inputs:
6  %   f  = the function f(x) for which you want to find the root
7  %   xL = left limit of the interval
8  %   xR = right limit of the interval
9  % Outputs:
10 %   r     = the root
11 %   iters = number of iterations performed
12 %
13 % Author: Libao Jin
14 % Date: 02/13/2020
```

```
%

iters = 0;
maxIters = 20000;
tol = 1e-5;
xM = (xL + xR) / 2;
while abs(f(xM)) > tol && iters < maxIters
    if f(xL) * f(xM) < 0
        xR = xM;
    else
        xL = xM;
    end
    xM = (xL + xR) / 2;
    iters = iters + 1;
end
r = xM;

end
```

- Script file `hw2_p1.m`

```
% MATH 3340, Spring 2020
% Homework 2, Problem 1
% Author: Libao Jin
% Date: 02/13/2020

clear; clc;
f = @(x) (x.^3 + 3 * x + 1) .* (x <= 0) + (1 + sin(x)) .* (x > 0);
[r, iters] = bisection(f, -2, 0.1);
fprintf('%10s %10s %10s\n', 'x*', 'f(x*)', 'iters')
fprintf('%10f %10f %10d\n', r, f(r), iters)
```

- Output file `hw2_p1.txt`

```
        x*      f(x*)      iters
 -0.322185   0.000003         18
```

□

2 Problem 2

Using the code in Problem 1 as a template (thus possibly with similar input and output arguments), write a different MATLAB function which now implements Newton's Method to find the root of a function. Use this new code to find the root of

$$f(x) = x^3 + 3x + 1$$

with an initial guess $x_0 = -2$ and an accuracy $\alpha = 10^{-5}$. For this problem submit the equivalent files similar to those for Problem 1: a function file, script file, and output file. Your output must also meet the same requirements: it must list the value of the root, function value at the root, and number of iterations performed.

Solution.

- Function file `newton.m`

```
function [r, iters] = newton(f, df, x0)

%NEWTON: Newton Method
% Syntax: [r, iters] = newton(f, df, x0)
% Inputs:
%   f  = the function f(x) for which you want to find the root
%   df = the first derivative of function f(x)
%   x0 = the initial guess
% Outputs:
%   r     = the root
%   iters = number of iterations performed
%
% Author: Libao Jin
% Date: 02/13/2020
%

r = x0;
iters = 0;
tol = 1e-5;
maxIter = 20000;
while abs(f(r)) > tol && iters < maxIter
    r = x0 - f(x0) / df(x0);
    x0 = r;
    iters = iters + 1;
end

end
```

- Script file `hw2_p2.m`

```
% MATH 3340, Spring 2020
% Homework 2, Problem 2
% Author: Libao Jin
% Date: 02/13/2020

clear; clc;
f = @(x) x.^3 + 3 * x + 1;
df = @(x) 3 * x.^2 + 3;
x0 = -2;
[r, iters] = newton(f, df, x0);
fprintf('%10s %10s %10s\n', 'x*', 'f(x*)', 'iters')
fprintf('%10f %10f %10d\n', r, f(r), iters)
```

- Output file `hw2_p2.txt`

```
        x*      f(x*)      iters
 -0.322185  -0.000000          5
```

□

3 Problem 3

Now use both codes you developed above (Newton and bisection) to find the roots of the equation:

$$f(x) = \frac{1}{2} + \frac{x^2}{4} - x\sin(x) - \frac{\cos(2x)}{2}.$$

(a) **Bisection method:** Use the initial bracket $x_L = 0$, $x_R = \pi$. Do you encounter any problems? Is there a way your code can be improved to behave appropriately for this problem? Explain any changes you choose to make.

(b) **Newton's method:** Use all the following initial guesses:

- $x_0 = \pi/2$,
- $x_0 = 5\pi$,
- $x_0 = 10\pi$.

Use an accuracy of $\alpha = 10^{-5}$, and a maximum number of 20000 iterations for all computations. Try to explain the different behavior for the three starting values in Newton's method case.

For each part, again submit the same kind of files indicated in Problem 1: a function file, script file, and output file. Your output must again align to the same requirements.

Solution.

(a)

- Problem encountered:

- Explanation for changes made:

- Function file `bisectionImproved.m`

```
function [r, iters] = bisectionImproved(f, xL, xR)

%BISECTIONIMPROVED: Improved Bisection Method
% Syntax: [r, iters] = bisectionImproved(f, xL, xR)
% Inputs:
%   f  = the function f(x) for which you want to find the root
%   xL = left limit of the interval
%   xR = right limit of the interval
% Outputs:
%   r     = the root
%   iters = number of iterations performed
%
% Author: Libao Jin
% Date: 02/13/2020
%

r = 0;
iters = 0;
tol = 1e-5;

```

```
if f(xL) < tol
    r = xL;
    return
elseif f(xR) < tol
    r = xR;
    return
elseif f(xL) * f(xR) > 0
    disp('The prerequisite of using bisection method is not satisfied');
end
xM = (xL + xR) / 2;
while abs(f(xM)) > tol
    if f(xL) * f(xM) < 0
        xR = xM;
    else
        xL = xM;
    end
    xM = (xL + xR) / 2;
    iters = iters + 1;
end
r = xM;

end
```

- Script file `hw2_p3_a.m`

```
% MATH 3340, Spring 2020
% Homework 2, Problem 3 (a)
% Author: Libao Jin
% Date: 02/13/2020

clear; clc;
f = @(x) 1/2 + x.^2 / 4 - x .* sin(x) - cos(2 * x) / 2;
fprintf('%10s %10s %10s\n', 'x*', 'f(x*)', 'iters')
[r, iters] = bisection(f, 0, pi);
fprintf('%10f %10f %10d\n', r, f(r), iters)
[r, iters] = bisectionImproved(f, 0, pi);
fprintf('%10f %10f %10d\n', r, f(r), iters)
fplot(f, [0, pi]);
```

- Output file `hw2_p3_a.txt`

```
        x*      f(x*)      iters
  3.141593   2.467401      20000
  0.000000   0.000000          0
```

(b)

- Explanation for different behavior for the three initial guesses:

- Function file `newton.m`

```
function [r, iters] = newton(f, df, x0)

%NEWTON: Newton Method
% Syntax: [r, iters] = newton(f, df, x0)
% Inputs:
```

```
%   f  = the function f(x) for which you want to find the root
%   df = the first derivative of function f(x)
%   x0 = the initial guess
% Outputs:
%   r     = the root
%   iters = number of iterations performed
%
% Author: Libao Jin
% Date: 02/13/2020
%

r = x0;
iters = 0;
tol = 1e-5;
maxIter = 20000;
while abs(f(r)) > tol && iters < maxIter
    r = x0 - f(x0) / df(x0);
    x0 = r;
    iters = iters + 1;
end

end
```

- Script file `hw2_p3_b.m`

```
% MATH 3340, Spring 2020
% Homework 2, Problem 3 (b)
% Author: Libao Jin
% Date: 02/13/2020

clear; clc;
f = @(x) 1/2 + x.^2 / 4 - x .* sin(x) - cos(2 * x) / 2;
df = @(x) x / 2 - sin(x) - x .* cos(x) + sin(2 * x);
x0 = [pi/2, 5 * pi, 10 * pi];
fprintf('%10s %10s %10s\n', 'x*', 'f(x*)', 'iters')
for i = 1:length(x0)
    [r, iters] = newton(f, df, x0(i));
    fprintf('%10f %10f %10d\n', r, f(r), iters)
end
```

- Output file `hw2_p3_b.txt`

```
        x*      f(x*)      iters
  1.892490   0.000006          6
  1.892790   0.000005         10
 -1.897893   0.000004      13044
```

□

4 Problem 4

Write a recursive MATLAB function that implements the bisection method to find the root of a given function $f(x)$. The function header for the bisection method should be similar to the one in the first problem, using the same inputs as a minimum and producing the same outputs. You may

find it necessary to use some extra variable or variables as input arguments. Run this code on the function:

$$f(x) = x^2 - 7$$

with the initial bounds $x_L = -1$, $x_R = 9$. Again submit the type of files corresponding to those in Problem 1.

Solution.

- Function file `bisectionRecursive.m`

```
function [r, iters] = bisectionRecursive(f, xL, xR)

%BISECTIONRECURSIVE: Recursive Bisection Method
% Syntax: [r, iters] = bisectionRecursive(f, xL, xR)
% Inputs:
%   f  = the function f(x) for which you want to find the root
%   xL = left limit of the interval
%   xR = right limit of the interval
% Outputs:
%   r     = the root
%   iters = number of iterations performed
%
% Author: Libao Jin
% Date: 02/13/2020
%

tol = 1e-5;
r = 0;
iters = 0;
xM = (xL + xR) / 2;

if abs(f(xL)) < tol
    r = xL;
    return
elseif abs(f(xR)) < tol
    r = xR;
    return
elseif f(xL) * f(xR) > 0
    disp('The prerequisite of using bisection method is not satisified')
    return
elseif abs(f(xM)) < tol
    r = xM;
    iters = 0;
    return
end

if f(xL) * f(xM) < 0
    [r, iters] = bisectionRecursive(f, xL, xM);
else
    [r, iters] = bisectionRecursive(f, xM, xR);
end
iters = iters + 1;

end
```

- Script file `hw2_p4.m`

```
% MATH 3340, Spring 2020
% Homework 2, Problem 4
% Author: Libao Jin
% Date: 02/13/2020

clear; clc;
f = @(x) x.^2 - 7;
[r, iters] = bisectionRecursive(f, -1, 9);
fprintf('%10s %10s %10s\n', 'x*', 'f(x*)', 'iters')
fprintf('%10f %10f %10d\n', r, f(r), iters)
```

- Output file `hw2_p4.txt`

```
        x*      f(x*)      iters
  2.645751   0.000001         21
```

□

5 Problem 5

This is a problem to be solved only analytically. Think about Newton's method and its disadvantage: it requires the knowledge for the derivative of the function. Develop a method that circumvents this problem by using an approximation to the derivative obtained using a secant to the graph of the function: suppose you start with two points x_0 and x_1. Your approximation to the root, x_2, will be the intersection of the secant through $(x_1, f(x_1))$ and $(x_0, f(x_0))$ with the x-axis. Express x_2 as a function of x_0 and x_1, then show how this can be generalized from $\{x_0, x_1, x_2\}$ to $\{x_{k-1}, x_k, x_{k+1}\}$.

Solution.

□