BLG202E: Numerical Methods Homework #2

Uğur Ali Kaplan - 150170042

March 28, 2019

1. In order to save us the hassle of calculating the derivative, I have used the secant method as my go to method. Code files are *findroot.m*, *secant_method.m* and *f_1.m*. In my main program, I call findroot function and also there is a call to secant_method function inside the findroot function.

   (a) First of all, I have declared an anonymous function in the console. `f = @(x) (2*cosh(x/4))-x;` Since the example in our textbook uses 10^{-8} for tolerance, I have used the same value. But in my experiments, I also saw that $eps * 10$ can be used for tolerance. After that, I tried using nprobe = 10 as it was instructed. This has failed as expected.

   ```
   findroot(f, 0, 10, 10, 10^-8)
   ans = []
   ```

   After that, I tried 9 and again program was unable to find a root. But when given 8 as nprobe value:

   ```
   >> findroot(f, 0, 10, 8, 10^-8)
   ans =
  2.357551068067551   8.507199570713036
   ```

   So, two root values of this function are $x_1^* = 2.357551068067551$ and $x_2^* = 8.507199570713036$ within tolerance 10^{-8}.

   (b) For the second part of the question, I have created a function file *f_1.m*. This file also can be found within the zip file. Then, in the console:

   ```
   >> f = @f_1;
   >> findroot(f,-10,10,1,10^-7)
ans =
  -9.424777960769379  -6.283185482025146  -3.141592653589792   3.141592741012573
   6.283185307179602   9.424777030944824
   ```

   When nprobe=1, roots of the given function are $x_1^* = -9.424777960769379$, $x_2^* = -6.283185482025146$, $x_3^* = -3.141592653589792$, $x_4^* = 3.141592741012573$, $x_5^* = 6.283185307179602$ and $x_6^* = 9.424777030944824$ within tolerance 10^{-7}.

2. (a) In this function, all of the methods can be used but since bisection method will be slow we are left with secant and Newton methods. Since this function

$$f(x) = x - 1 \tag{1}$$

has a nice derivative of

$$f'(x) = 1 \tag{2}$$

using Newton's methods will be easy and fast. So, Newton's Method is a good idea for this one.

(b) For this function, when we review the graph it is obvious that we are not dealing with a smooth function so any method based on derivatives will cause problems. To solve this one, most convenient method is the bisection method since it does not rely on derivatives or Taylor series but rather on Mean Value Theorem.

Another reason to use bisection method is that this function is likely to have one or at most two roots as it can be seen on the graph.

(c) Even if this function is differentiable 5 times, since it is hard to pinpoint the first derivative we will not be able to use Newton's method so easily.

But, we know that this function is differentiable and we want to use that to our advantage. So, we will use a method based on derivatives but without the need of calculating the derivative. This description is basically what Secant method is, therefore we can use the secant method.

3. For this question, we are going to use Tridiagonal Matrix Algorithm also known as Thomas Algorithm. Thomas Algorithm is used for solutions to the tridiagonal systems of n equations when there is no need for stability i.e no need for pivoting strategies. It is also a faster version of Gaussian Elimination. Gaussian Elimination requires $O(n^3)$ flops whereas Thomas Algorithm requires only $O(n)$.

   Code can be found as *thomas.m* within the zip folder. For the given parameters, here is the code I have written on the console and the results I got:

```
>> main = [];
>> below = [];
>> above = [];
>> for i=1:10
main=[main, 3*i];
end
>> for i=1:9
above=[above,-i];
below=[below,-i];
end
>> b = randperm(100,10) % 10 random numbers between 1-100

b =

    91    94    79    24    99    31    77    65    25    54

>> thomas(below, main, above, b)

ans =

   39.8444   28.5331   18.6770   10.6756   12.0190    7.7165    7.9671    6.2872    3.7653
       2.9296
```

Listing 1: Thomas Algorithm in work

When solving Ax=b when A is defined as in the question and b as below:

$$b = \begin{bmatrix} 91 \\ 94 \\ 79 \\ 24 \\ 99 \\ 31 \\ 77 \\ 65 \\ 25 \\ 54 \end{bmatrix} \quad thenx = \begin{bmatrix} 39.8444 \\ 28.5331 \\ 18.6770 \\ 10.6756 \\ 12.0190 \\ 7.7165 \\ 7.9671 \\ 6.2872 \\ 3.7653 \\ 2.9296 \end{bmatrix}$$

4. (a) When A is examined, it is obvious that the last and the third row has to exchange places so we do not have any zeroes in the diagonal.

Exchange of the third and the fourth rows can be realized with the usage of permutation matrix P given as:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

So,

$$PA = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

When we look at the PA, we see that it is an upper triangular matrix. Then, there is no need of any elimination and lower triangular matrix L will simply be an identity matrix.(In identity matrix, all the entries above the diagonal are 0 after all.)

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$U = \begin{bmatrix} 5 & 6 & 7 & 8 \\ 0 & 4 & 3 & 2 \\ 0 & 0 & -1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(b) $A = P^{-1}LU$ then $Ax = b$ is same as $LUx = Pb$. If we say $Ux = y$ and solve $Ly = Pb$ for y, we can deduce the x. Since L is an identity matrix, $y = Pb$ is also known.

$$Pb = y = \begin{bmatrix} 26 \\ 9 \\ -3 \\ 1 \end{bmatrix}$$

Now, solving for $Ux = y$ using Backward Substitution, we have $x_4 = 1/1 = 1$, $x_3 = (-3+2)/-1 = 1$, $x_2 = (9-3-2)/4 = 1$ and $x_1 = (26-6-7-8)/5 = 1$. So,

$$x = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$