Group Members:

Ali Valiyev

Murathan Bilgen

Orkhan Ashrafov

Hüseyin Eren Demirtaş

Solutions of Exercise 1

1. In order to define such a vector for sketching the graph of the functions, we have used the following code:

xplot = linspace(-pi,pi,100)

1. We have defined “y1plot” vector, in order to sketch the graph of the , with using the following code:

y1plot = cos(xplot); % for the y = cosx

1. We have also defined “y2plot” vector, in order to sketch the graph of the , with using the following code:

y2plot = xplot % for the y = x

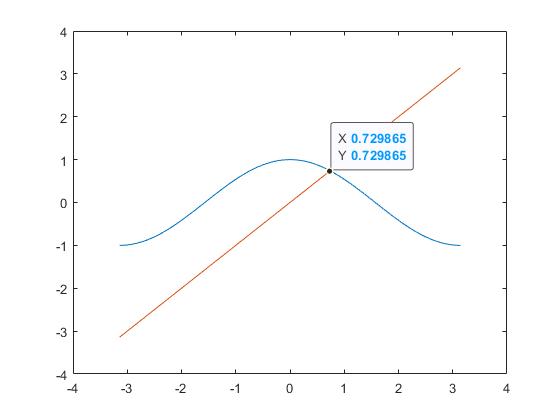
1. After that, in order to sketch the graph of the both functions at the same time, we have used the “hold on” and “hold off” command, with using the following code we have obtain the graph of the both function at the same time:

plot(xplot, y1plot)

hold on

plot(xplot, y2plot)

hold off

1. Question 5 was missing in the homework document file.
2. We have read the approximated value of the x at the intersection point from the obtained graph, and it is: x = 0.729865
3. We have saved the obtained graph as .jpg file form and attached to the homework file.

Solutions of Exercise 2

1. We have written a new m-file with named “cosmx.m”, which defines us the function; , by using the flowing script in the file:

function y = cosmx(x) % y = cosmx(x) computes the difference y=cos(x)-x

y = cos(x) - x;

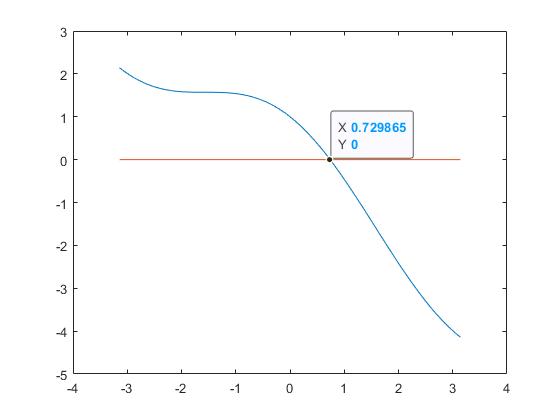
1. After this part, in order to check that whether our m-file is working or not, we have given some values for the function to test it, such as x = 0.5, and the output was 0.377582561890373, which agree with the value given in the homework document file.
2. First of all, to see the intersection of the graph with x-axis, we need to define a function on . For this purpose, we have used again same linespace with previous exercise bur this time function will be returning to 0. In order to sketch the graph of the both functions, we have used similar codes with previous exercise, which is following:

plot(xplot, cosmx(xplot))

hold on

plot(xplot, 0\*xplot)

hold off

1. Between 4 and 7, all questions were missing in the homework document file.
2. Between 4 and 7, all questions were missing in the homework document file.
3. Between 4 and 7, all questions were missing in the homework document file.
4. Between 4 and 7, all questions were missing in the homework document file.
5. We have read the approximated x value at the intersection point with the curve and x-axis, and the result is same with the value we have obtain in previous exercise, which was x = 0.729865

Solutions of Exercise 3

1. If the given function is continuous in and also , by the Intermediate Value Theorem there exists root for the function namely and the if given initial points are suitable for the bisection algorithm then we have a pretty nice formula that gives us the minimum number of iteration in order to converge the root with desired tolerance epsilon value, the formula is given as follow:
2. For instance, , since , bisection method not applicable here.

Solutions of Exercise 4

1. We replaced the part “???” with
2. It is clear that “EPSILON” is a variable. Initially, we consider that the reason why it is capitalized is that it may confuse with the its lowercase form, i.e. “epsilon”. Thus, we try to replace “EPSILON” with “epsilon”. Then, we realize that actually nothing changes. In other words, the result is the same as the before case. Finally, we consider that the reason may about “eps”. So, we change all “epsilon” with “eps”. Surprisingly, the result still remains unchanged. Therefore, we thought that since “EPSILON” is a variable, there is no specific reason why it is capitalized.
3. Yes, actually our variable and “eps” have similar sounding but they are totally different because “EPSILON” is a variable while “eps” is a real number stored in MATLAB.
4. When we apply real numbers x into “sign(x)” function, output for positive real numbers 1 whereas for negative real numbers -1. When 0 is applied to sign(x) function, the output is 0. If we apply sign(x) function to and , then we get always 1, -1 or 0. So, actually we get the same answer all the times. Thus, this does not give us any solution.
5. Done.
6. When the error function is called, the result is “bisect\_cosmx failed with too many iterations!”.
7. If a and b are close to each other, then , , will be close to each other. Thus, itCount must be fewer because solution must be closer to x.
8. We tried the command “[z, iterations] = bisect\_cosmx (0,3)”. We realized that the root z is much close to the root of the equation “cos(z) = z” or “cos(z) – z = 0”. Both of them are 0.7391, which is very close to the root found at the first question.
9. With the help formula used in 3rd exercise, we found that . On the other hand, we found minimum number of iterations as 35 in MATLAB. Thus, the number found by MATLAB is more accurate. Moreover, the result computed by hand is smaller than the result computed by MATLAB.
10. We wrote “help bisect\_cosmx” command line and we get our comment written in bisect\_cosmx.m document as like at the bottom

[x,itCount] = bisect\_cosmx( a, b) uses bisection to find a root of cosmx between a and b to tolerance of 1.0e-10

a=left end point of interval

b=right end point of interval

cosmx(a) and cosmx(b) should be of opposite signs x is the approximate root found itCount is the number of iterations required. Orkhan Ashrafov, Murathan Bilgen, Hüseyin Eren Demirtaş & Ali Valiyev and the date is 04.01.2022

1. If or then this does not give us a result. To determine this, we add

if fa \* fb >= 0

error('bisect\_cosmx failed with too many iterations!')

else

...

...

...

...

end

end

end

part in the code. Thus, we prevent returning incorrect answer in this way.

Solutions of Exercise 5

1. We changed file name by bisect and write it by   
   “ function [x, itCount] = bisect ( func, a, b ) “
2. We replaced all cosmx with func.
3. When we write “ [z, iterations] = bisect ( cosmx, 0, 3 ) “ in the command part, we get an error.
4. We created firstly a file called f0.m containing new function . Then, we firstly write “ [z, iterations] = bisect ( @f0, 0, 3 ) “ and find   
   “ [z, iterations] = [1.0000, 35] “
5. Similarly, when we write command line “ [z, iterations] = bisect ( @(x) 1-x, 0, 3 ) “ and find the same answer  
    “ [z, iterations] = [1.0000, 35] “.
6. Finally, we write

f0=@(x) 1-x;

[z, iterations] = bisect ( f0, 0, 3 )

codes in the command line. Thus, unsurprisingly, we get the same answer founded before, i.e.   
“ [z, iterations] = [1.0000, 35] “.

Solutions of Exercise 6

1. We have defined 5 different script files for each of these given functions. The definitions of them will be very similar with cosmx. After defining them, we have used bisect.m script file with bisection algorithm for each of those function with suitable interval and with the tolerance . After applying the algorithm, the obtained results are all saved to the table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Name | Formula | Interval | Approx. Root | No. Steps |
| f1 |  |  | 2.999999999956344 | 36 |
| f2 |  |  | 1.167303978290875 | 34 |
| f3 |  |  | 2.910383045673370e-11 | 35 |
| f4 |  |  | 0.282032183924457 | 36 |
| f5 |  |  | 1.000000000029104 | 35 |

Solutions of Exercise 7

1. We have changed the command line of the “bisect.m” as described in the homework file. After that we have saved this new script file as “bisect0.m”.
2. Between 4 and 7, all questions were missing in the homework document file.
3. We have tested our new script “bisect0.m” with using the following command line;  
   After running this command line, the system gives us output as follows;  
   Which means that, the script file is really working.

z =

0.999999999941792

iteration =

34

[z, iteration] = bisect0(@f0,0,3)

1. By the given definition, we have;  
   0.999999999941792) = -5.820799398037479e-11. Therefore, the residual error must be “5.820799398037479e-11”.  
   Also, since we know that the function has root at 1, so the true solution is 1, we have that, the true error must be “1 - 0.999999999941792” = “5.8207994e-11”. The number of iterations it takes is 34.
2. We have used the bisect0.m script to get the value of residual approximation for (x-1)^5 and the output of the system was;  
     
   Also we have f(1.007812500000000) = 2.910383045673370e-11. Therefore, the residual error must be “2.910383045673370e-11”.   
   Also, since we know that the function has root at 1, so the true solution is 1, we have that, the true error must be “1 - 1.007812500000000” = “0.0078125”. The number of iterations it takes is 7.

z =

1.007812500000000

iteration =

7

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Formula | Interval | Residual Error | True Error | Number of Steps |
| f0 |  |  | 5.820799398037479e-11 | 5.8207994e-11 | 34 |
| f5 |  |  | 2.910383045673370e-11 | 0.0078125 | 7 |

1. While using the result that we have obtained from the previous part of this exercise, we have filled the following table to summarize what we have;
2. As we can see from the table, the residual error must always be less than the given ESPILON value, in our case it was 1.0e-10. However, true error may or may not be less than the EPSILON, it depends on the function.
3. It is because of that since the error of the function always must be bigger than 0, as we are looking for the “size” of the true error and residual error, we do not want to consider the negative value of the function.