Ellie Ishii
Project 1, 8/1/2023
EE160 – Summer 2023

EE160 Project 1 - Report

1. PURPOSE OF THE PROJECT

This project is intended as a solo programming project using C, where a numerical problem needs to be solved under a set of given constraints. The user is prompted to enter the coefficients of the polynomial function (a + bx + cx^2 + dx^3 + ex^4 + fx^5) as well as the starting and ending x-values, and number of intervals. The program then generates a table for x-values, corresponding y-values, and any roots that are found. It also displays a total number of sign changes located on the table and lets the user decide if they want to change the search range for the table before ending the program.

2. PROBLEM DESCRIPTION

This project is aimed at solving for the y-values and roots of the user-input based polynomial function of the form: a + bx + cx^2 + dx^3 + ex^4 + fx^5. Therefore, a total of three source codes (project1.c, secant.c, and myfunc.c) and two header files (secant.h and myfunc.h) are included. The program uses two functions, the secant method to find the roots of the polynomial function and the polynomial function to calculate corresponding y values. For the program to run successfully, three loops are required. One loop displays x-values, y-values, and root locations (if any) or a blank line (if no root is found). The second loop is for the secant method which finds the root of the polynomial when there is a sign change. The third loop is used to repeat the program and calculations if the user is not done. Additionally, two if statements are included. The first if statement is to detect a sign change and trigger the secant method. The second one is to check if the user wants to keep calculating or finish the program.

3. VERSION HISTORY- INCREMENTAL DEVELOPMENT

By adopting an incremental development approach and building the code step by step, it ensures that the foundation of the code is strong, which also helps in debugging the code effectively. The very first step for me was to print out my previous homework and consider how to implement homework 15, 16, and 17 into this project and figure out where if statements and loops were necessary. The Phase 1 helped me get my ideas together. Although Version 1 was very stressful and the most difficult part for me, looking back at Phase 1 helped me gather my thoughts. In Version 1, I built the foundation of the project. At first, I created up until displaying the x-values and y-values using loop, and ensured that the program ran successfully. Then I completed the rest of Version 1. After figuring out Version 1, the rest of the versions were relatively simple and straightforward. I used Version 1 as a foundation and kept building another step upon it to add complexity. For example, I made the user chose the coefficients in Version 2, then changed the polynomial to 5th order in Version 3. I separated this main file into three in version 4 and five in version 5. I tested every time I made a change and only moved on to the next version after debugging it. I continued this process until I reached Version 5. This incremental development approach has helped me keep things simple and ensure that my code was in a good shape each time.

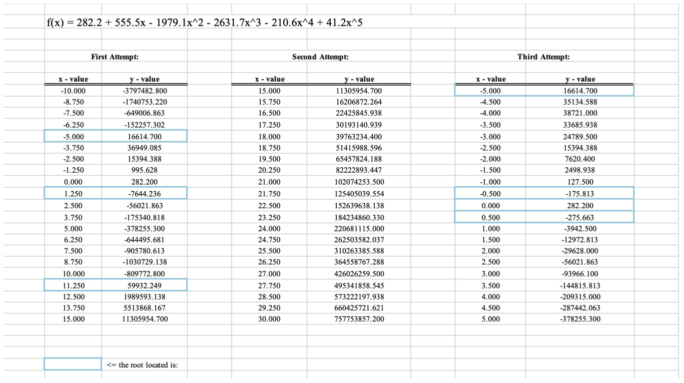
4. TESTING AND TEST-RUN #1, TEST-RUN #2 RESULTS

In Version 1, since the function was already hard coded, I validated my outputs using my previous homework and class exercises. I searched up quadratic equations with known solutions online and inputted the same values, as well as the function we have been using, to ensure my code was working properly for Version 2. From Version 3 to 5, I used the two test runs provided, looked up polynomial functions with answers online and used outside sources to check my outputs. I was able to obtain the predicted results for Test Run #1 and Test Run #2.

5. VALIDATING RESULTS OUTSIDE ‘C’

I verified my answers by using Excel for the x-values and y-values in the table and desmos for my solutions.

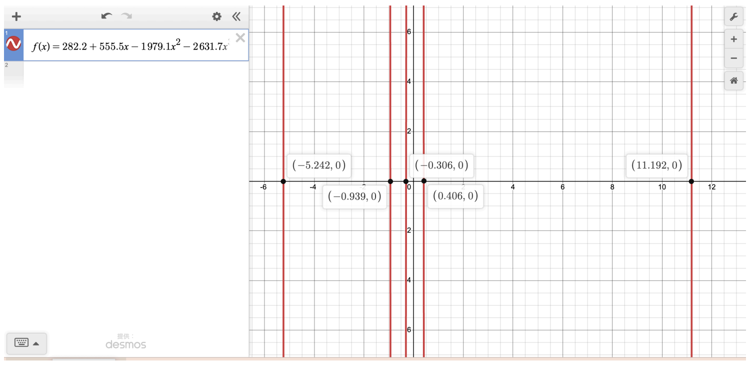
< Validating Results for Unsolved Polynomial >



f(x) = 282.2 + 555.5x - 1979.1x^2 - 2631.7x^3 - 210.6x^4 + 41.2x^5

First Attempt:		Second Attempt:		Third Attempt:	
x - value	**y - value**	**x - value**	**y - value**	**x - value**	**y - value**
-10.000	-3797482.800	15.000	11305954.700	-5.000	16614.700
-8.750	-1740753.220	15.750	16206872.264	-4.500	35134.588
-7.500	-649006.863	16.500	22425845.938	-4.000	38721.000
-6.250	-152257.302	17.250	30193140.939	-3.500	33685.938
-5.000	16614.700	18.000	39763234.400	-3.000	24789.500
-3.750	36949.085	18.750	51415988.596	-2.500	15394.388
-2.500	15394.388	19.500	65457824.188	-2.000	7620.400
-1.250	995.628	20.250	82222893.447	-1.500	2498.938
0.000	282.200	21.000	102074253.500	-1.000	127.500
1.250	-7644.236	21.750	125405039.554	-0.500	-175.813
2.500	-56021.863	22.500	152639638.138	0.000	282.200
3.750	-175340.818	23.250	184234860.330	0.500	-275.663
5.000	-378255.300	24.000	220681115.000	1.000	-3942.500
6.250	-644495.681	24.750	262503582.037	1.500	-12972.813
7.500	-905780.613	25.500	310263385.588	2.000	-29628.000
8.750	-1030729.138	26.250	364558767.288	2.500	-56021.863
10.000	-809772.800	27.000	426026259.500	3.000	-93966.100
11.250	59932.249	27.750	495341858.545	3.500	-144815.813
12.500	1989593.138	28.500	573222197.938	4.000	-209315.000
13.750	5513868.167	29.250	660425721.621	4.500	-287442.063
15.000	11305954.700	30.000	757753857.200	5.000	-378255.300

<= the root located is:



6. VULNERABILITIES IN THE CODE

When there are multiple roots that are very close to each other with a small number of intervals, some roots may not be found and displayed. This is what happened when I tried to find the roots for the unknown polynomial. I was not able to find all of the roots in the first attempt because I found out that the three of the roots were very close to each other, and each x values were not close enough. Additionally, If the variable type of user-input does not match with what is declared in the source code, then it will generate error. For instance, if the user enters a decimal number for steps, my program will not work. Another circumstance is when y0 and y1 are very close to each other, the secant method may have a division by zero which could generate errors. I was not able to implement any error handlings this time. However, if I had time, I would have used fgets and sscanf instead of scanf to perform quality control on user input. Also, I would add an if statement to check if y0 and y1 are very close to each other to prevent a division by 0.

7. UNSOLVED POLYNOMIAL

What are the real roots of the polynomial**:**

$$f(x) = 282.2 + 555.5x - 1979.1x^2 - 2631.7x^3 - 210.6x^4 + 41.2x^5 = 0$$

x = - 5.242, - 0.939, - 0.306, 0.406, 11.192

8. CONCLUSIONS

I was able to meet all the project requirements and my code works properly. Since this is my first coding class and I am still trying to get used to coding, this project was challenging for me.
However, I am glad and contented that I managed to complete the project and find the roots of the unsolved polynomial.

APPENDIX

PSEUDOCODE

project1.c

```
START
 * Declare double a, b, c, d, e, f
 * Ask user to enter the coefficients for a polynomial of the form: a+bx+cx^2+dx^3+ex^4+fx^5
 * Read and store inputs as a b c d e f
 * Print "The function is: (the function with the values)"
 * Declare and assign integer done = FALSE
 * Start while loop while user is not done
 *      Print "The range of x-values for calculations of the table?"
 *      Declare double xmin and xmax
 *      Print "Enter start end: "
 *      Read and store inputs as xmin and xmax
 *      Declare integer steps
 *      Print "Enter the number of intervals for your table: "
 *      Read and store input as steps
 *      Declare double incr
 *      Calculate incr (incr = (xmax - xmin) / steps)
 *      Print "x-value      y-value" for the table
 *      Declare double x
 *      Declare integer i
 *      Declare and initialize integer j = 0
 *      Start FOR loop (initialize i = 0, loop until i<= steps, increment i by 1)
 *        Define x = xmin + i * incr
 *        Declare and initialize double previousx =  x - incr
 *        Print the calculated x and y values
 *        Declare double x0, x1, x2, y0, y1, y2, xnew
 *        Declare and initialize double epsilon = 0.001
 *        Declare and set int signChange = FALSE
 *        If (F(previousx)<0 and F(x)>=0) or (F(previousx)>=0 and F(x)<0)
 *             Reassign the x values
 *             Set sign change = TRUE
 *             Start WHILE loop while |F(a,b,c,d,e,f,x2)|>epsilon
 *                  Call myfunc file
 *                  Set y1 = F1,
 *                  Set y2 = F2
 *                  Call secant file
 *                  Set xnew = X(x1, x2, y1, y2)
 *                  Reassign the x values
 *             x = xnew
 *             Print where the root is located
 *             Increment j by 1
 *        Else
 *             Print a blank line
 *      Print the total sign changes located on the table
 *      Declare character choice
 *      Print "Do you wish to change the search range for the table? (y/n)"
 *      Read and store input as choice
 *      If (choice = y or Y)
 *        Rerun the program
 *      Else
 *        End the program
 * Return 0
END
```

secant.c

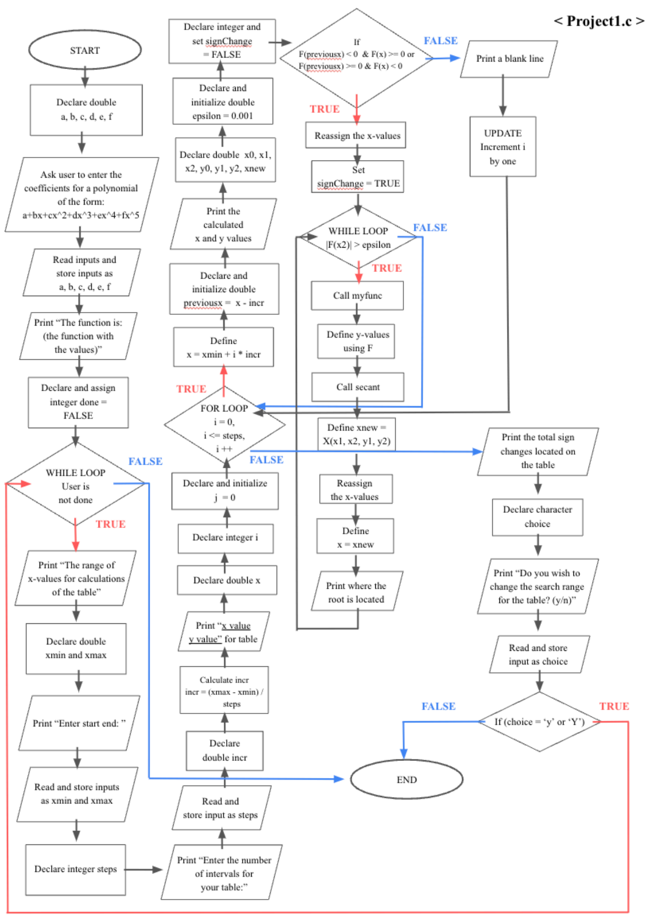
```
START
 * Receive a, b, c, d, e, f, x0, x1
 * Call myfunc file
 * Define X = x1-y1*(x1-x0)/(y1-y0)
 * Return X
END
```

myfunc.c

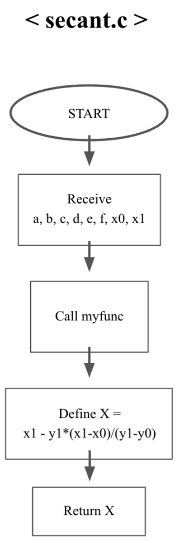
```
START
 * Receive a, b, c, d, e, f, x
 * Define F(x) = a+bx+cx^2+dx^3+ex^4+fx^5
 * Return F(x)
END
```

FLOWCHARTS

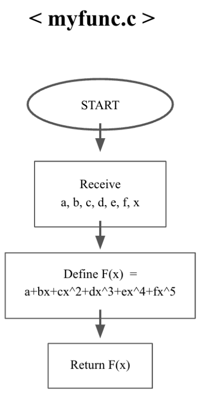
project1.c


< Project1.c >
START
Declare double a, b, c, d, e, f
Ask user to enter the coefficients for a polynomial of the form: a+bx+cx^2+dx^3+ex^4+fx^5
Read inputs and store inputs as a, b, c, d, e, f
Print "The function is: (the function with the values)"
Declare and assign integer done = FALSE
WHILE LOOP User is not done
FALSE
TRUE
Print "The range of x-values for calculations of the table"
Declare double xmin and xmax
Print "Enter start end: "
Read and store inputs as xmin and xmax
Declare integer steps
Print "Enter the number of intervals for your table:"
Read and store input as steps
Declare double incr
Calculate incr incr = (xmax - xmin) / steps
Print "x value y value" for table
Declare double x
Declare integer i
Declare and initialize j = 0
FOR LOOP i = 0, i <= steps, i ++
TRUE
FALSE
Define x = xmin + i * incr
Declare and initialize double previousx = x - incr
Print the calculated x and y values
Declare double x0, x1, x2, y0, y1, y2, xnew
Declare and initialize double epsilon = 0.001
Declare integer and set signChange = FALSE
If F(previousx) < 0 & F(x) >= 0 or F(previousx) >= 0 & F(x) < 0
FALSE
TRUE
Print a blank line
UPDATE Increment i by one
Reassign the x-values
Set signChange = TRUE
WHILE LOOP |F(x2)| > epsilon
FALSE
TRUE
Call myfunc
Define y-values using F
Call secant
Define xnew = X(x1, x2, y1, y2)
Reassign the x-values
Define x = xnew
Print where the root is located
Print the total sign changes located on the table
Declare character choice
Print "Do you wish to change the search range for the table? (y/n)"
Read and store input as choice
If (choice = 'y' or 'Y')
FALSE
TRUE
END

secant.c


< secant.c >
START
Receive a, b, c, d, e, f, x0, x1
Call myfunc
Define X = x1 - y1*(x1-x0)/(y1-y0)
Return X

myfunc.c


< myfunc.c >
START
Receive a, b, c, d, e, f, x
Define F(x) = a+bx+cx^2+dx^3+ex^4+fx^5
Return F(x)