EAFIT University - Department of Information and Computer Sciences

Mauricio Escudero Restrepo
César Esteban Peñuela Cubides
Diego Mesa Ospina

August 2024

1 Objetivo

Review the progress of the final project and the work methodologies of the workgroups

2 Course

Numerical Analisis

3 Responsible Faculty Member

Edwar Samir Posada Murillo

4 Current Report Delivery Date

October 1st 2024

5 Numerical Methods

The following numerical methods will be presented in this document in the following manner, first a pseudocode version of the method algorith will be presented, after the code for the implementation of the method in both languages selected will be presented, finally proof for the execution and results of the methods will be provided.

5.1 Incremental Search

Incremental Search Method is a numerical method used to find the roots of a function (i.e., where the function equals zero). This method works by iteratively narrowing down an interval where a root lies. The goal is to find an approximation of the root with increasing precision.

5.1.1 Pseudo-code

5.1.2 Method Implementation

Python

Rust

5.1.3 Method Tests

The test parameters are as follows: - f= math.log((math.sin(x)**2) + 1) - 1/2 - x0 = -3 - step = 0.5 - Tol = 1x10-7 - N = 100

Result: Interval: [-2.5, -2.0], Iterations: 2

i	x_i	f_xi	e
1	-2.5	-0.193863	0.5
2	-2.0	0.102578	0.5

5.2 Incremental Search

5.2.1 Pseudo-code

5.2.2 Method Implementation

Python

Rust

5.2.3 Method Test

5.3 Bisection

The bisection method is based on the Intermediate Value Theorem, which states that if a continuous function changes sign over an interval, there is at least one root in that interval. The method systematically reduces the interval in which the root lies by repeatedly bisecting it.

5.3.1 Pseudo-code

5.3.2 Method Implementation

Python

Rust

5.3.3 Method Test

The test parameters are as follows: - f= math.log((math.sin(x)**2) + 1) - 1/2 - a = 0 - b = 1 - Tol = 1x10-7 - N = 100

Result: - Root: 0.9364047050476074 - Iterations: 21 - Converged: True

i	x_i	f_x_i	e
0	0.5	-0.2931087267313766	0.5
1	0.75	-0.11839639385347844	0.25
2	0.875	-0.036817690757380506	0.125
3	0.9375	0.0006339161592386899	0.0625
4	0.90625	-0.017772289226861138	0.03125
5	0.921875	-0.008486582211768012	0.015625
6	0.9296875	-0.0039053586270640928	0.0078125
7	0.93359375	-0.0016304381170096915	0.00390625
8	0.935546875	-0.0004969353153196909	0.001953125
9	0.9365234375	6.882244496264622e-05	0.0009765625
10	0.93603515625	-0.00021397350516405567	0.00048828125
11	0.936279296875	-7.255478812057126e-05	0.000244140625
12	0.9364013671875	-1.860984900181606e-06	0.0001220703125
13	0.93646240234375	3.348202684883006e-05	6.103515625e-05
14	0.936431884765625	1.581084516011355e-05	3.0517578125e-05
15	0.9364166259765625	6.975011174192858e-06	1.52587890625e-05
16	0.9364089965820312	2.5570333977986692e-06	7.62939453125e-06
17	0.9364051818847656	3.4802931392352576e-07	3.814697265625e-06
18	0.9364032745361328	-7.564765268641693e-07	1.9073486328125e-06
19	0.9364042282104492	-2.042232898902263e-07	9.5367431640625e-07

5.4 False Rule

The false position method (also known as the regula falsi method) is a root-finding technique used in numerical analysis. It is similar to the bisection method in that it iteratively narrows down an interval where a root of a function exists. However, instead of using the midpoint of the interval as in the bisection method, the false position method uses a more refined estimate by linearly interpolating the function between the endpoints.

5.4.1 Pseudo-code

5.4.2 Method Implementation

Python

Rust

5.4.3 Method Tests

Approximate solution: x = 0.9364045808798893 Iterations: 5 Error: 2.2097967899981086e-10

i	x_i	f_xi	e
1	0.9365060516656253	5.875600835791861e-05	0.0025656709474095596
2	0.9364047307426415	8.67825411532408e-08	0.00010132092298376083
3	0.936404581100869	1.2815393191090152e-10	1.4964177252885236e-07
4	0.9364045808798893	1.894040480010517e-13	2.2097967899981086e-10

5.5 Fixed Point

The fixed-point method is an iterative numerical technique used to solve equations of the form x=g(x)x = g(x)x=g(x). In this method, the goal is to find a value xxx such that g(x)=xg(x) = xg(x)=x, which is known as a fixed point of the function g(x)g(x)g(x).

5.5.1 Pseudo-code

5.5.2 Method Implementation

Python

Rust

5.6 Newton

5.6.1 Pseudo-code

5.6.2 Method Implementation

Python

Rust

5.6.3 Method Test

5.7 Seccant

5.7.1 Pseudo-code

5.7.2 Method Implementation

Python

Rust