Peter Julius M. Estacio

Grouped with:

Augusto Gabbriel M. Calilung

Rafael Luis Mari R. Lambo

ENGG 27.01 - M

February 23, 2026

Project 1: Fixed-Point Approach for Finding Roots of Polynomials

A C++ program was developed to demonstrate the use of the fixed-point approach to find the root of a polynomial. By reading a valid polynomial file and extracting its contents, it can devise a fixed-point form of the equation $x = g(x)$ for each non-constant term of the original function $f(x) = 0$. The fixed-point form is found by isolating the non-constant term and dividing the whole function in such a way that only x remains on one side of the equation. Specifically, for a function $f(x) = 0$ with non-constant term ax^n, x is isolated on one side by first transposing the term then dividing the whole function by $-ax^{n-1}$, as shown in the equations below.

$$f(x) = 0$$

$$f(x) - ax^n = -ax^n$$

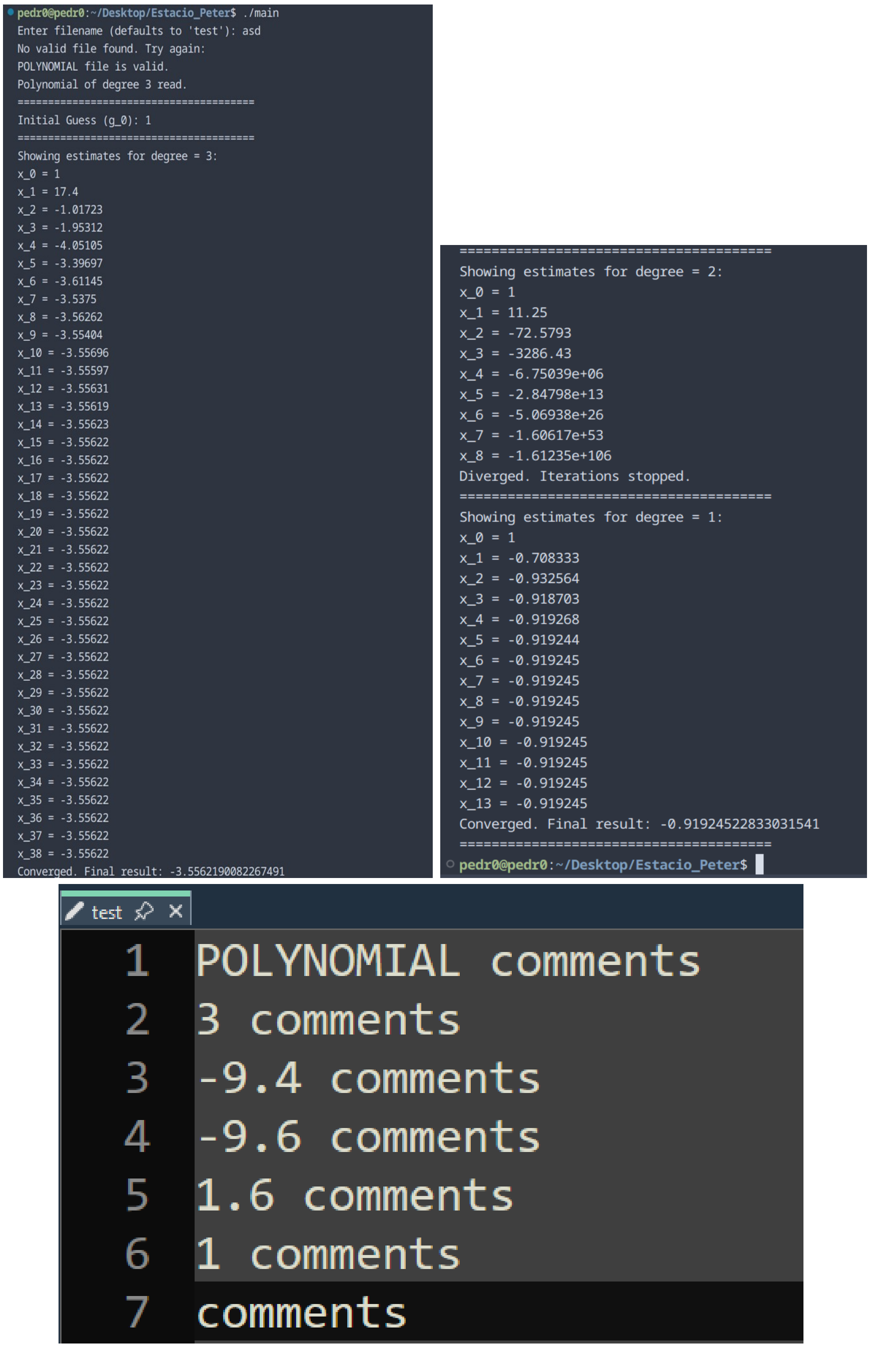
$$g(x) = \frac{f(x) - ax^n}{-ax^{n-1}} = x$$

This is implemented in the code in the screenshot below. All other terms for this iteration are first calculated for within the `for loop` then stored in the `double g_x`. Then, the non-constant term in question is isolated by dividing `g_x` by the term $-a * x^{i-1}$, where the coefficient a is stored in the `double* equation`, and the `integer i` is an iterator representing the degree of the function. This is executed along with the fixed-point calculations for all non-constant terms, producing multiple fixed-point forms for the same equation depending on the degree of the initial function. It must be noted that this produces a rational function $g(x)$ where $x \neq 0$, for which the program makes sure to validate user input.

```
for(int k = equationIndex; k > -1; k--)
{
  if(k != i) g_x += equation[k]*pow(x, k);
}
g_x /= -equation[i]*pow(x, i-1);
```

screenshot - how the application obtains the equation for the fixed-point iterations

Shown below are sample screenshots of the program being run using the basic test parameters. The program can read the polynomial file, extract its contents, and it’s shown to iterate through all non-constant terms by degree to calculate for the roots using the fixed-point approach. It is also demonstrated that the program will iterate until the estimates diverge, or the estimates converge to a value to the maximum precision stored in a double. The contents of the polynomial file “test” used during runtime are also shown below.



```
pedr0@pedr0:~/Desktop/Estacio_Peter$ ./main
Enter filename (defaults to 'test'): asd
No valid file found. Try again:
POLYNOMIAL file is valid.
Polynomial of degree 3 read.
========================================
Initial Guess (g_0): 1
========================================
Showing estimates for degree = 3:
x_0 = 1
x_1 = 17.4
x_2 = -1.01723
x_3 = -1.95312
x_4 = -4.05105
x_5 = -3.39697
x_6 = -3.61145
x_7 = -3.5375
x_8 = -3.56262
x_9 = -3.55404
x_10 = -3.55696
x_11 = -3.55597
x_12 = -3.55631
x_13 = -3.55619
x_14 = -3.55623
x_15 = -3.55622
x_16 = -3.55622
x_17 = -3.55622
x_18 = -3.55622
x_19 = -3.55622
x_20 = -3.55622
x_21 = -3.55622
x_22 = -3.55622
x_23 = -3.55622
x_24 = -3.55622
x_25 = -3.55622
x_26 = -3.55622
x_27 = -3.55622
x_28 = -3.55622
x_29 = -3.55622
x_30 = -3.55622
x_31 = -3.55622
x_32 = -3.55622
x_33 = -3.55622
x_34 = -3.55622
x_35 = -3.55622
x_36 = -3.55622
x_37 = -3.55622
x_38 = -3.55622
Converged. Final result: -3.5562190082267491
```

```
========================================
Showing estimates for degree = 2:
x_0 = 1
x_1 = 11.25
x_2 = -72.5793
x_3 = -3286.43
x_4 = -6.75039e+06
x_5 = -2.84798e+13
x_6 = -5.06938e+26
x_7 = -1.60617e+53
x_8 = -1.61235e+106
Diverged. Iterations stopped.
========================================
Showing estimates for degree = 1:
x_0 = 1
x_1 = -0.708333
x_2 = -0.932564
x_3 = -0.918703
x_4 = -0.919268
x_5 = -0.919244
x_6 = -0.919245
x_7 = -0.919245
x_8 = -0.919245
x_9 = -0.919245
x_10 = -0.919245
x_11 = -0.919245
x_12 = -0.919245
x_13 = -0.919245
Converged. Final result: -0.91924522833031541
========================================
pedr0@pedr0:~/Desktop/Estacio_Peter$
```

```
test
1 POLYNOMIAL comments
2 3 comments
3 -9.4 comments
4 -9.6 comments
5 1.6 comments
6 1 comments
7 comments
```

sample screenshots – program executable run with example results, and polynomial file used

```
 *  Executing task: C/C++: LINUX BUILD (g++)

Starting build...
/usr/bin/g++ -fdiagnostics-color=always -g *.cpp -o '/home/pedr0/Documents/Coding/ENGG 27 - 2/Project 1/main'

Build finished successfully.
 *  Terminal will be reused by tasks, press any key to close it.
```

sample screenshot – compilation on Visual Studio Code

```
pedr0@pedr0:~/Documents/Coding$ g++ --version
g++ (Debian 12.2.0-14+deb12u1) 12.2.0
Copyright (C) 2022 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

compiler version – g++ (Debian 12.2.0-14+deb12u1) 12.2.0

Operating System	Debian GNU/Linux 12 bookworm (x86-64)
Cinnamon Version	5.6.8
Linux Kernel	6.1.0-39-amd64
Processor	Intel© Core™ i5-8250U CPU @ 1.60GHz × 4
Memory	15.5 GiB
Hard Drives	628.6 GB
Graphics Card	Intel Corporation UHD Graphics 620

Upload system information

Copy to clipboard

operating system – Debian 12 bookworm, Cinnamon version 5.6.8

Visual Studio Code

Version: 1.104.0
Commit: f220831ea2d946c0dcb0f3eaa480eb435a2c1260
Date: 2025-09-10T06:46:18.035Z (4 mos ago)
Electron: 37.3.1
ElectronBuildId: 12342881
Chromium: 138.0.7204.235
Node.js: 22.18.0
V8: 13.8.258.31-electron.0
OS: Linux x64 6.1.0-39-amd64

IDE – Visual Studio Code ver 1.104.0

Project Evaluation – Fixed-Point Approach for Finding Roots of Polynomials		
Item	**Points**	**Rubrics**
early work	8/8	(all or nothing) 8: at most 10% of the code in the final implementation differs from that in early work submission
implementation and testing environment report	4/4	4 – all instructions followed correctly
user interface: input file	4/4	4 – appropriate user interface provided
polynomial file	8/8	8 – application properly parses polynomial files
documentation of fixed point iteration	32/32	32 – properly working and clearly documented implementation of fixed point iteration
convergence	16/16	16 – all instructions related to convergence are properly implemented
divergence	16/16	16 – divergence is always detected properly
polynomial file	4/4	4 – correctly prepared polynomial file
program output	4/4	4 – output produced according to instructions
self-evaluation	4/4	4 – self-evaluation accurate (or evaluating this item leads to an error)
total	100/100	