ENGG 27.01 - Advanced Engineering Math with Numerical Methods, Lecture
ENGG 27.01 M 2025-2
Luisito L. Agustin

20260123

Project 1
Fixed-Point Approach for Finding Roots of Polynomials

Luisito L. Agustin

Project 1
Fixed-Point Approach for Finding Roots of Polynomials

Due: start of class time, February 24, 2026

Progress Report 1	Jan. 30
Progress Report 2	Feb. 6
Progress Report 3	Feb. 13
Early Work Submission	Feb. 20

Numerical Methods Project

Fixed-Point Approach for Finding Roots of Polynomials

Luisito L. Agustin
2026 01 23

General Specifications

Develop an application in C++ that demonstrates use of a fixed-point approach for finding roots of polynomials.

Detailed Specifications

Develop an application in C++ that demonstrates use of a fixed-point approach for finding roots of polynomials.

The application obtains a polynomial *f(x)* by extracting its coefficients from a polynomial file compliant with the Polynomial File Format specified in a separate document. Provide a means and appropriate guidance for the user to specify the polynomial file. Do not modify any filename provided by the user. Do not assume, nor require, nor add a ".txt" extension. Use whatever filename is provided.

Using each non-constant term of the polynomial, the application attempts to find a root of *f(x)* by solving for *x* from the non-constant term, and manipulating the equation

$$f(x) = 0$$

into the form

$$x = g(x)$$

and applying fixed point iterations. Decide on appropriate initial guesses.

The application monitors the fixed point iterations for convergence or divergence. If divergence is detected, the iterations are stopped. If the estimates for the root appear to converge, the application continues the iterations until the most precise value of the root is obtained.

The application displays each estimate generated for the root on a separate line. Announce divergence when it is detected. If the iteration converges, display the final value of the estimate using the maximum precission available for a *double* and display the value of the polynomial when evaluated at this estimated root. If the estimated root is accurate, the value of the polynomial should be 0, or very close to 0.

Revision History - Fixed-Point Approach for Finding Roots of Polynomials

2023 01 23 first version

Variable Specifications

Specifications and instructions from this point onwards may vary from one use of the document to the next. These are not tracked as revisions to the project specifications.

Group Work

Work on the C++ code may be done by groups. Documentation and testing must be done individually.

It is expected that C++ code submitted by groupmates may be similar or exactly the same. Points earned for the entire project will be subject to penalties for unauthorized collaboration if documentation submitted by at least two students contain at least one distinctive phrase, image, or item in common.

Notes

Solution of Equations by Iteration

For each of the remaining sections of this chapter, we select basic kinds of problems and discuss numeric methods on how to solve them. The reader will learn about a variety of important problems and become familiar with ways of thinking in numerical analysis.

Perhaps the easiest conceptual problem is to find solutions of a single equation

$$f(x) = 0, \tag{1}$$

where f is a given function. A **solution** of (1) is a number $x = s$ such that $f(s) = 0$. Here, s suggests "solution," but we shall also use other letters.

It is interesting to note that the task of solving (1) is a question made for numeric algorithms, as in general there are no direct formulas, except in a few simple cases.

Examples of single equations are $x^3 + x = 1$, $\sin x = 0.5x$, $\tan x = x$, $\cosh x = \sec x$, $\cosh x \cos x = -1$, which can all be written in the form of (1). The first of the five equations is an **algebraic equation** because the corresponding f is a polynomial. In this case the solutions are called **roots** of the equation and the solution process is called *finding roots*. The other equations are **transcendental equations** because they involve transcendental functions.

Erwin Kreyszig: *Advanced Engineering Mathematics*, 10th ed, 2011
Sec 19.2

Fixed-Point Iteration for Solving Equations $f(x) = 0$

Note: Our present use of the word "fixed point" has absolutely nothing to do with that in the last section.

By some *algebraic steps* we transform (1) into the form

(2) $$x = g(x).$$

Then we choose an x_0 and compute $x_1 = g(x_0)$, $x_2 = g(x_1)$, and in general

(3) $$x_{n+1} = g(x_n) \qquad (n = 0, 1, \cdots).$$

A solution of (2) is called a **fixed point** of g, motivating the name of the method. This is a solution of (1), since from $x = g(x)$ we can return to the original form $f(x) = 0$. From (1) we may get several different forms of (2). The behavior of corresponding iterative sequences $x_0, x_1, \cdots$ may differ, in particular, with respect to their speed of convergence. Indeed, some of them may not converge at all. Let us illustrate these facts with a simple example.

Erwin Kreyszig: *Advanced Engineering Mathematics*, 10th ed, 2011
Sec 19.2

Submission

Implementation and Testing Environment Report

Provide details of at least one platform used in implementing and testing the project.

* name and version of compiler
* name and version of integrated development environment (IDE)
* operating system name/distribution and version

screenshot of compilation

The screenshot must show the files involved and show feedback from the IDE reporting successful compilation and linking; if the project is compiled on the command line, the screenshot must show the commands issued; if a makefile was used, include a copy of the makefile in addition to the screenshot.

screenshot of program run on the console

Copy the executable to a location whose path contains a recognizable version of your name. If working with a group, make sure that the path does not contain any version of the names of any groupmates. Create a new folder named after yourself as needed. Run the executable by typing its name at the command prompt, and take a screenshot as the executable starts. Do not run the application from within an IDE.

Documentation

Obtain a screenshot of the application when used to solve for roots of the cubic

x^3 +1.6 x^2 -9.6 x -9.4 ($x^3+1.6x^2-9.6x-9.4$).

Be sure to prepare the polynomial file properly according to the specifications. Submit the polynomial file, and include the contents of the polynomial file in your documentation.

Provide documentation explaining how the application obtains the equation of the form

$x = g(x)$

from the polynomial in preparation for fixed point iterations.

Do a self-evaluation using the project evaluation table and evaluation procedures provided. Fill in your scores and add your points correctly.

Place all required documentation and the self-evaluation table together in a single document and export this document to pdf. Label all screenshots accurately.

Submit a single zip file containing
* C++ implementation files and header files for the project
* required documentation

Do NOT include executables in your submission.

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 instuctions 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	

Evaluation Procedures

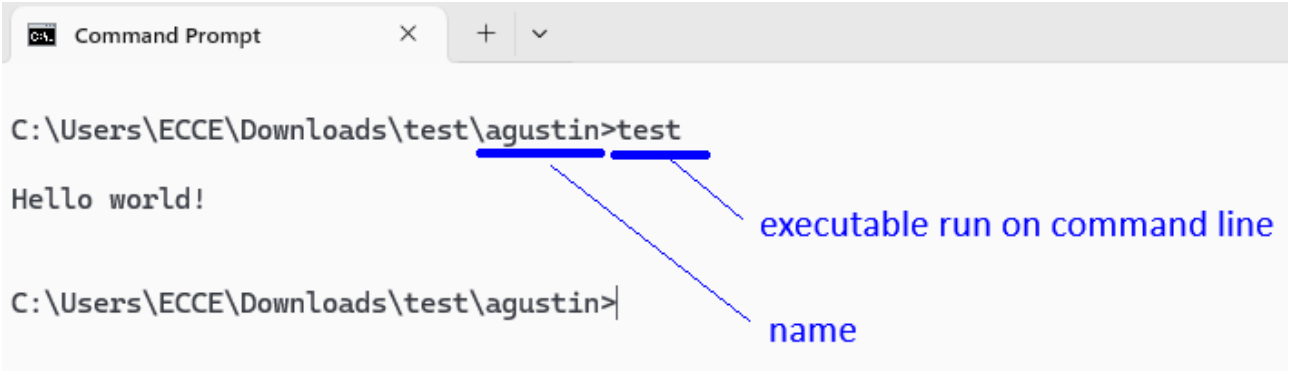
Special Cases

Score for the entire project is 0 if
* no source files are submitted
* the source files, as submitted, will not compile as a C++ project
* the implementation uses resources that are not allowed

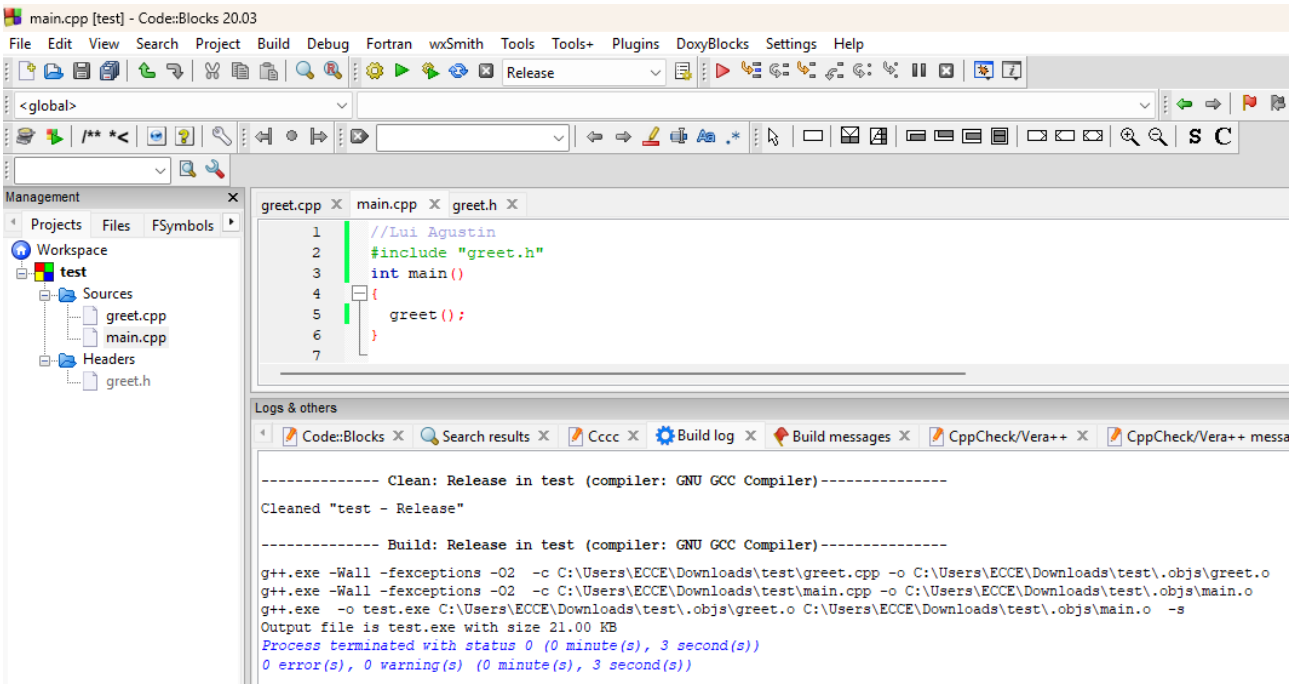
early work

Early work credits are based on the final code submitted, assuming the final and early work code use the same style. The total number of lines in the final code that are not in the early work version, or that were modified from the early work version, other than formatting changes that do not modify compiled code, must not be more than 10% of the number of lines in the final code, excluding blank lines.

Implementation and Testing Environment (4 pts)
1 - all details of implementation platform are provided
1 - screenshot of compilation: all instructions are followed correctly
1 - screenshot of program run on the console: all instructions are followed correctly
1 - all filenames appearing in the screenshots are consistent with files in the project submission



sample screenshot - executable test.exe run by typing its name at the prompt from a path with name of author

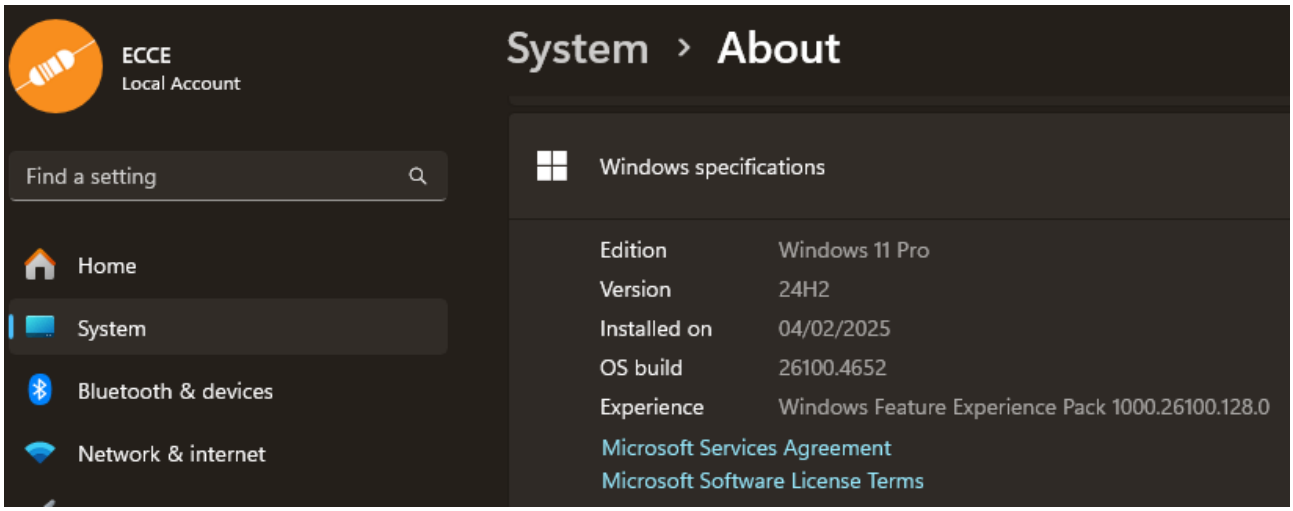


sample screenshot - compilation on Code::Blocks

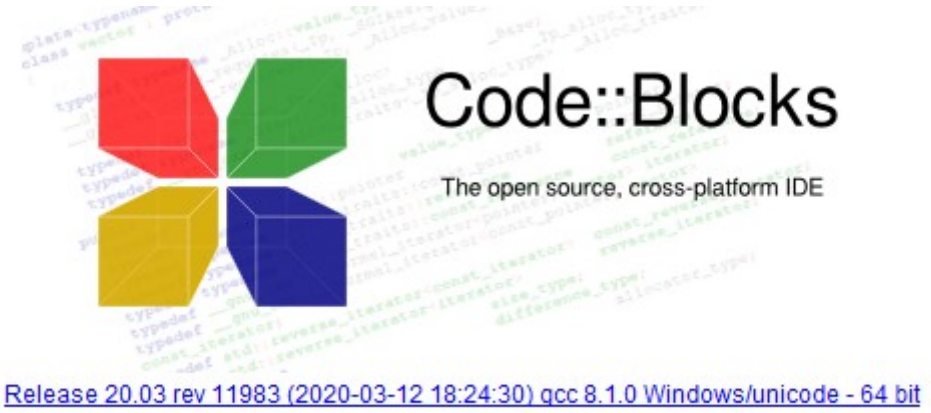
```
C:\Users\ECCE\codeblocks\MinGW\bin>g++ -v
Using built-in specs.
COLLECT_GCC=g++
COLLECT_LTO_WRAPPER=C:/Users/ECCE/codeblocks/MinGW/bin/../libexec/gcc/x86_64-w64-mingw32/8.1.0/lto-wrapper.exe
Target: x86_64-w64-mingw32
Configured with: ../../../src/gcc-8.1.0/configure --host=x86_64-w64-mingw32 --build=x86_64-w64-mingw32 --target=x86_64-w64-mingw32 --prefix=/mingw64
 --with-sysroot=/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64 --enable-shared --enable-static --disable-multilib --enable-languages=c,c++,fort
ran,lto --enable-libstdcxx-time=yes --enable-threads=posix --enable-libgomp --enable-libatomic --enable-lto --enable-graphite --enable-checking=rele
ase --enable-fully-dynamic-string --enable-version-specific-runtime-libs --disable-libstdcxx-pch --disable-libstdcxx-debug --enable-bootstrap --disa
ble-rpath --disable-win32-registry --disable-nls --disable-werror --disable-symvers --with-gnu-as --with-gnu-ld --with-arch=nocona --with-tune=core2
 --with-libiconv --with-system-zlib --with-gmp=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-mpfr=/c/mingw810/prerequisites/x86_64-w64-
mingw32-static --with-mpc=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-isl=/c/mingw810/prerequisites/x86_64-w64-mingw32-static --with-
pkgversion='x86_64-posix-seh-rev0, Built by MinGW-W64 project' --with-bugurl=https://sourceforge.net/projects/mingw-w64 CFLAGS='-O2 -pipe -fno-ident
 -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisite
s/x86_64-w64-mingw32-static/include' CXXFLAGS='-O2 -pipe -fno-ident -I/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/
prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/include' CPPFLAGS=' -I/c/mingw810/x86_64-810-posix-se
h-rt_v6-rev0/mingw64/opt/include -I/c/mingw810/prerequisites/x86_64-zlib-static/include -I/c/mingw810/prerequisites/x86_64-w64-mingw32-static/includ
e' LDFLAGS='-pipe -fno-ident -L/c/mingw810/x86_64-810-posix-seh-rt_v6-rev0/mingw64/opt/lib -L/c/mingw810/prerequisites/x86_64-zlib-static/lib -L/c/m
ingw810/prerequisites/x86_64-w64-mingw32-static/lib '
Thread model: posix
gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)

C:\Users\ECCE\codeblocks\MinGW\bin>
```

compiler - gcc version 8.1.0 (x86_64-posix-seh-rev0, Built by MinGW-W64 project)

ENGG 27.01 - Advanced Engineering Math with Numerical Methods, Lecture
ENGG 27.01 M 2025-2
Luisito L. Agustin



operating system - Windows 11 Pro, 24H2, build 26100.4652



IDE -Code::Blocks 20.03
rev 11983 (2020-03-12 18:24:30) gcc 8.1.0 Windows/unicode - 64 bit

self-evaluation
0 - not done
2 - self-evaluation score differs from project score by more than 10 points or scores not tallied properly
4 - self-evaluation accurate (or evaluating this item leads to an error)

Appendix: Polynomial File Format

See next page.

Polynomial File Format

Luisito L. Agustin
2024 10 09

This document specifies a text file format for representing polynomials with real-valued coefficients. The polynomials being represented are of the form

p(x) = a_0 + a_1 x + a_2 x^2 + . . . + a_{n-1} x^{n-1} + a_n x^n

$$p(x)=a_0+a_1x+a_2x^2+\ldots+a_{n-1}x^{n-1}+a_nx^n$$

of degree *n* in a single variable *x*. The variable name itself is not explicitly part of the specification. The file name extension is not part of the specification. Routines that process polynomial files must check the contents of the file regardless of the file name extension.

A polynomial is specified thru the first tokens extracted from consecutive lines of a text file, starting with the first line. These tokens may be preceded by any amount of white space and may be followed on the same line by additional text that may be treated as comments or otherwise ignored as far as specifying the polynomial is concerned.

POLYNOMIAL *optional comments* n *optional comments* a_0 *optional comments* a_1 *optional comments* . . . a_{n-1} *optional comments* a_n *optional comments* *optional comments*

The first token on the first line must be the string “POLYNOMIAL” in uppercase. The first token on the second line must be a valid integer specifying the degree of the polynomial. Routines that process polynomials files must refer to this value in processing the rest of the file. The first tokens on subsequent lines must be valid floating point numbers representing the coefficients of the polynomial. The constant coefficient is specified first on line 3 followed by coefficients of increasing powers of the unspecified variable on subsequent lines.

To be considered compliant with the specifications, routines that process polynomial files must correctly identify polynomials from files that strictly comply with the specifications.

Compliant routines may accept polynomial specifications from files where the required string on the first line is not entirely in uppercase, or where it is absent. Compliant routines may also accept polynomial specifications where the required integer specifying the degree of the polynomial is absent, in which case the routine may instead determine the degree from the actual number of valid floating point coefficients extracted. The routine remains compliant provided that it also correctly identifies polynomials from files that strictly comply with the specifications.

Revision History - Polynomial File Format

2024 10 09	first version

Notes are not tracked as revisions to the specifications.

Notes

In C++, the extraction operator >> will skip white space in identifying tokens and will stop at white space.

The zero polynomial may be specified with a degree of 0 and a constant coefficient of 0 or 0.0, giving the zero polynomial the same status as other constant polynomials. However, there are references that consider the degree of the zero polynomial to be undefined or to have a value of -1. The specifications do not explicitly give any special status to the zero polynomial and hence, may not be adequate in applications where giving special status to the zero polynomial is critical.