DESIGN

Wenyu Liang

January 30, 2023

1 Program Description

This program uses different solutions to approximate Euler's number e and π. To obtain enough precision, we will force the difference between the estimation and the relatively accurate values defined in math.lib smaller than 10^{-14}, and return the number of terms used for the approximation.

2 Files to be included in directory asgn2

1. **bbp.c**: This contains the implementation of the Bailey-Borwein-Plouffe formula to approximate π and the function to return the number of computed terms.
2. **e.c**: This contains the implementation of the Taylor series to approximate Euler's number e and the function to return the number of computed terms.
3. **euler.c**: This contains the implementation of Euler's solution used to approximate π and the function to return the number of computed terms.
4. **madhava.c**: This contains the implementation of the Madhava series to approximate π and the function to return the number of computed terms.
5. **mathlib-test.c**: This contains the main() function which tests each of your math library functions.
6. **mathlib.h**: This contains the interface for your math library.
7. **newton.c**: This contains the implementation of the square root approximation using Newton's method and the function to return the number of computed iterations.
8. **viete.c**: This contains the implementation of Viète's formula to approximate π and the function to return the number of computed factors.
9. **Makefile**: This file is provided and directs the compilation process of this program.

10. **README.md**: Describe how to use the script and Makefile.
11. **DESIGN.pdf**: Describes design for the program with pseudocode and visualization.
12. **WRITEUP.pdf**: Describes how the program works in detail, codes included.

3 Structure

Implementation of different formulas with we written into independent .c files. They are bbp.c, e.c, euler.c, madhava.c, newton.c, viete.c. In addition, we need to implement square root approximation using Newton's method, which we be written in newton.c. We're also interested in the number of computed terms for approximation, so there will be a static variable to record the computed terms, and a function to return this number, in each above .c files.
The main function is in mathlib-test.c, which keeps calling above approximating formula until the difference between the outputs and M_PI or M_E is smaller than a certain threshold.
The mathlib.h file will be an interface to coordinate all these files. Makefile will compile .c files and make it an binary file.

4 Pseudocode

1. **mathlib-test.c**

```
void main(option) {
    for (int i=0; i<argc; i++){
        if -s is provided:
            s = YES
        else
            s = NO
    }

    while ( difference > EPSILON )
        switch option:
            case _:
                 PI = call pi approximation test
                 difference = abs(M_PI - PI)
                 if (s = YES) {terms() activated}
            case a:
                 run all test
```

2. ***.c general structure**

```
static int k = 0
static current_term = 1
function(){
    formula -> current_term
    current_term + = current_term
    k + =1
}
terms(){
    return k
}
```