Assignment 2 Design Document

Alex Yeh

January 27, 2023

1 Description of Program

The program "mathlib-test.c" is a program that contains the main test harness for my implemented math library. My math library contains functions to compute the constants e and pi.

2 Files to be included in directory "asgn2":

1. bbp.c
   - This C file contains the implementation of the Bailey-Borwein-Plouffe formula to approximate pi and the function to return the number of computed terms.
2. e.c
   - This C file 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 C file contains the implementation of Euler's solution to approximate pi and the function to return the number of computed terms.
4. madhava.c
   - This C file contains the implementation of the Madhava series to approximate pi and the function to return the number of computed terms.
5. mathlib-test.c
   - This C file contain the main function which tests each of my math library functions.
6. mathlib.h
   - This header file contains the interface for my math library.

7. newton.c
   - This C file 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 C file contains the implementation of Viete's formula to approximate pi and the function to return the number of computed factors.
9. Makefile
   - This file directs the compilation process of mathlib-test.c
10. README.md
    - This file is in Markdown format and describes how to use my program and Makefile. It also lists and explains the different command-line options that my program accepts.
11. DESIGN.pdf
    - This file is a PDF version of this design document for assignment 2. It describes my design and design process for my program with pseudocode and images.
12. WRITEUP.pdf
    - This file is a PDF version of my writeup for assignment 2. It contains graphs displaying the difference between the values reported by my implemented functions and that of the math library. It also contains analysis and explanations for any discrepancies and findings that I gleaned from my testing.

3 Pseudocode

bbp.c

Create a static variable called "computed_terms" to keep track of computed terms

$$p(n) = \sum_{k=0}^{n} 16^{-k}\left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6}\right).$$

The formula above is the Bailey-Borwein-Plouffe (BBP) formula that I used from the assignment 2 pdf to help me write my function below.

double pi_bbp(void)
Set computed_terms equal to 0
Create double variable called “sum” and set equal to 0.0
Create double variable called “term” and set equal to 1.0
Create double variable called “multiplier” and set equal to 1.0
For loop from double k = 0, while the absolute value of term is greater than EPSILON, incrementing k by 1
Set term equal to (4.0/(8.0*k+1.0))-(2.0/(8.0*k+4.0))-(1.0/(8.0*k+5.0))-(1.0/(8.0*k+6.0))
Set term equal to term * multiplier
Set multiplier equal to multiplier / 16
Add term to the sum
Increment computed terms variable by 1
Return the sum

int pi_bbp_terms(void)
Return variable “computed_terms”

e.c

Create a static variable called “computed_terms” to keep track of computed terms

$$\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!} \times \frac{x}{k}.$$

The formula above is the formula for Euler’s number e that I used from the assignment 2 pdf to help me write my function below.

double e(void)
Declare double variable called “sum” and set to 1
Declare double variable called “term” and set to 1
For loop starting with k=1 while the absolute value of the term is greater than epsilon, incrementing up by 1
Multiply term variable by 1/k and assign value to term variable
Add the term to the sum
Increment computed terms variable by 1
Return the sum

int e_terms(void)
Return variable “computed_terms”

euler.c

Create a static variable called “computed_terms” to keep track of computed terms

$$\frac{x^k}{k!} = \frac{x^{k-1}}{(k-1)!} \times \frac{x}{k}.$$

The formula above is the formula for Euler's solution that I used from the assignment 2 pdf to help me write my function below.

double pi_euler(void)

Declare double variable called "sum" and set to 0

Declare double variable called "term" and set to 1

For loop starting with k=1 while the absolute value of the term is greater than epsilon, incrementing up by 1

Set term variable equal to 1/(k*k)

Add the term to the sum

Increment computed terms variable by 1

Multiply the sum by 6

Take the square root of the sum

Return the sum

int pi_euler_terms(void)

Return variable "computed_terms"

madhava.c

Create a static variable called "computed_terms" to keep track of computed terms

$$p(n) = \sqrt{12}\sum_{k=0}^{n} \frac{(-3)^{-k}}{2k+1} = \sqrt{12}\left[\frac{1}{2}3^{-n-1}\left((-1)^n \Phi\left(-\frac{1}{3}, 1, n+\frac{3}{2}\right) + \pi 3^{n+\frac{1}{2}}\right)\right]$$

The formula above is the formula for the Madhava series that I used from the assignment 2 pdf to help me write my function below.

double pi_madhava(void)

Declare double variable called "sum" and set to 0

Declare double variable called "term" and set to 1

Declare double variable called "numerator" and set to 1

Declare double variable called "denominator" and set to 1

For loop starting with k=0 while the absolute value of the term is greater than epsilon, incrementing up by 1

Add term to the sum

Multiply numerator by -1/3

Add 2 to the denominator

Set term to numerator/denominator
Increment computed terms variable by 1
Multiply the sum by the square root of 12
Return the sum

int pi_madhava_terms(void)
Return variable "computed_terms"

newton.c

Create a static variable called "computed_terms" to keep track of computed terms

```
def sqrt(x):
    z = 0.0
    y = 1.0
    while abs(y - z) > epsilon:
        z = y
        y = 0.5 * (z + x / z)
    return y
```

The picture above is python code for the square root function from assignment 2's pdf. I used this as reference to write my sqrt_newton function.

double sqrt_newton(double x)
Create a double variable called "z" and set equal to 0
Create a double variable called "y" and set equal to 1
Set computed terms variable to 0
While loop while the absolute value of z-y is greater than epsilon
Set z equal to y
Set y equal to 0.5 *(z + x/z)
Increment computed terms variable by 1
Return y

int sqrt_newton_terms(void)
Return variable "computed_terms"

viete.c

Create a static variable called "computed_terms" to keep track of computed terms

Viète's formula can be written as follows:

$$\frac{2}{\pi} = \frac{\sqrt{2}}{2} \times \frac{\sqrt{2+\sqrt{2}}}{2} \times \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} \cdots$$

Or more simply,

$$\frac{2}{\pi} = \prod_{k=1}^{\infty} \frac{a_k}{2}$$

The formula above is Viete's formula that I used from the assignment 2 pdf to help me write my function below.

double pi_viete(void)

Create a double variable called "previous" and set equal to 0
Create a double variable called "current" and set equal to the sqaure root of 2
Create a double variable called "sum" and set equal to current/2
Set computed terms variable to 0
While loop while the absolute value of current-previous is greater than epsilon
    Set previous equal to current
    Set current equal to the square root of 2 + previous
    Multiply the sum by the current term divided by 2
    Increment computed terms variable by 1
Return 2 divided by the sum
Return the sum

int pi_viete_terms(void)

Return variable "computed_terms"

mathlib-test.c

define OPTIONS "aebmrvnsh :"

Create static void function called "program_usage" to print out help message

int main(int argc, char **argv)

Create int variable called "opt" and set equal to 0
Create boolean variable called "e_value" and set equal to false
Create boolean variable called "bbp" and set equal to false
Create boolean variable called "madhava" and set equal to false
Create boolean variable called "euler" and set equal to false
Create boolean variable called "viete" and set equal to false
Create boolean variable called "newton" and set equal to false

Create boolean variable called “stats” and set equal to false
While loop while opt is not equal to -1
    Switch statement for opt
        Case ’a’:
            Set all boolean variables above except stats to true
            break
        Case ’e’:
            Set e_value to true
            break
        Case ’b’:
            Set bbp to true
            break
        Case ’m’:
            Set madhava to true
            break
        Case ’r’:
            Set euler to true
            break
        Case ’v’:
            Set viete to true
            break
        Case ’n’:
            Set newton to true
            break
        Case ’s’:
            Set stats to true
            break
        Case ’h’:
            Call program_usage function
            Return 0
        Default:
            Call program_usage function
            Return 1

If statement for if e_value is true
    Print my e value, print the math library’s e value, and print the difference between the two values
    If statement for if stats is true
        Print variable that keeps track of e terms

If statement for if bbp is true
    Print my pi bbp value, print the math library’s pi value, and print the difference between

the two values
    If statement for if stats is true
        Print variable that keeps track of pi_bbp terms

If statement for if madhava is true
    Print my pi madhava value, print the math library's pi value, and print the difference between the two values
    If statement for if stats is true
        Print variable that keeps track of pi_madhava terms

If statement for if euler is true
    Print my pi euler value, print the math library's pi value, and print the difference between the two values
    If statement for if stats is true
        Print variable that keeps track of pi_euler terms

If statement for if viete is true
    Print my pi viete value, print the math library's pi value, and print the difference between the two values
    If statement for if stats is true
        Print variable that keeps track of pi_viete terms

If statement for if bbp is true
    For loop starting from double i = 0, while i is less than 10, incrementing up by 0.1
        Print my sqrt_newton value, print the math library's square root value, and print the difference between the two values
        If statement for if stats is true
            Print variable that keeps track of sqrt_newton terms

If statement for if e_value is false or bbp is false or madhava is false or euler is false or viete is false or newton is false
    Call program_usage function
    Return 1

Return 0 to signify success