DESIGN.pdf ASGN2

Nathan Ong

October 8, 2021

1 Description

This collection of files contains a functional math library that has various methods that are used to approximate π and one method to approximate e. This library is also usable with the following command line options:

- -a: Runs all functions
- -e: Runs the e approximation function
- -b: Runs the Bailey-Borwein-Plouffe function
- -m Runs the Madhava series function
- -r: Runs the Euler solution function
- -v: Runs the Viete function
- -n: Runs the Newton-Raphson square root function
- -s: Enables printing to statistics to see the computed terms and factors for each function
- -h: Displays a help message detailing the usage of the program

1.1 Formulas

1. Bailey-Borwein-Plouffe Formula: $p(n) \sum_{k=0}^{n} 16^{-k}(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k-6})$
2. Euler's Number (e): $\sum_{k=0}^{\infty} \frac{1}{k!}$
3. The Madhava Series: $\sum_{k=0}^{\infty} \frac{-3^{-k}}{2k+1} = \frac{\pi}{12}$
4. Newton-Raphson Method ($\sqrt{x}$): $\mathrm{x}_1 = 1.0, x_{k+1} = \mathrm{x}_k - \frac{f(x_k)}{f'(x_k)}$
5. Viete's Formula: $\frac{2}{\pi} = \prod_{k=1}^{\infty} \frac{a_k}{2}, a_1 = \sqrt{2}, a_k = \sqrt{2 + a_{k-1}}$
6. Euler's Solution to the Basel Problem: $p(n) = \sqrt{6 \sum_{k=1}^{n} \frac{1}{k^2}}$

2 Files Included in the Directory

1. mathlib-test.c
   (a) This file contains the main() function that tests the math library functions.
2. bbp.c
   (a) This file contains the code for the Bailey-Borwein-Plouffe formula to calculate an approximation of π along with an additional function that returns the number of computed terms.
3. e.c

(a) This file contains the code for the Euler's formula to calculate an approximation of e, or Euler's number, along with an additional function that returns the number of computed terms.

4. euler.c

    (a) This file contains the code for Euler's formula used to calculate an approximation of π along with an additional function that returns the number of computed terms.

5. madhava.c

    (a) This file contains the code for the Madhava series used to calculate an approximation of π along with an additional function that returns the number of computed terms.

6. newton.c

    (a) This file contains the code for Newton's method used to calculate an approximation of $\sqrt{x}$ along with an additional function that returns the number of computed terms.

7. viete.c

    (a) This file contains the code for Viete's formula used to calculate an approximation of π along with an additional function that returns the number of computed terms.

8. mathlib.h

    (a) This file contains the interface for the math library contained in mathlib-test.c.

3 Pseudocode and Structure

3.1 mathlib-test.c

initialize opt variable
    while opt isn't -1
        switch statement for all library command line functions

3.2 bbp.c

set static counter variable
    loop until computed term is over $x * 10^{-14}$
        calculate 16^{-k}
        calculate $\frac{(k(120k+151)+47)}{k(k(k(512k+1024)+712)+194)+15}$
        multiply both and add above to total term
        increment term counter by 1
    return total term

3.3 e.c

set static counter variable
    loop until computed term is over $x * 10^{-14}$
        calculate 1/n where n = (k-1)! * k
        add above to total term
        increment term counter by 1
    return total term

3.4 euler.c

set static counter variable
 loop until computed term is over $x * 10^{-14}$
  calculate $1/k^2$
  add to total term
  increment total terms by 1
  multiply 6 and above
  get square root of above (save separately)
 return above

3.5 madhava.c

set static counter variable
 loop until computed term is over $x * 10^{-14}$
  calculate $\frac{(-3)^{-k}}{2k+1}$
  add to total term
  increment term counter by 1
  calculate $\sqrt{12}$
  multiply above and total term together (save separately)
 return total term

3.6 newton.c

set static counter variable
 define and set two variables to 1.0 and 0.0 respectively (y and z)
  loop until the absolute value of y - z is greater than x * 10^{-14}
   equate y to z
   y = 0.5 * ($\frac{z+x}{z}$)
   increment total terms by 1
 return y

3.7 viete.c

set static counter variable
 loop until computed term is over $x * 10^{-14}$
  calculate $\sqrt{2}$
  calculate $\frac{a_k}{2}$ where $a_1 = \sqrt{2}, a_k = \sqrt{2 + a_{k-1}}$
  multiply above and total term together
  multiply above and 2 together (save seperately)
 return total term

4 Additional Credits

- I attended Sloan's in-person 1:15 PM section on October 5^{th}, and he provided the general structure of the Makefile including the shortening of implementations with various Linux commands.
- I derived the newton.c formula from the Python function given in the asgn2.pdf document.
- I derived the basis of mathlib-test.c from the supplied code given in the asgn2.pdf document.