Assignment 2 Design

Michael Coe

September 2021

* All functions require mathlib.h to be included in their work.

e.c

Goal: Approximate the value of e and track the number of computed terms:

- `static int count = 0`
  Static integer that counts the terms used to compute e
- `public double e(void)`
  This function will approximate e using a for loop.
  Starting with outside variables: `double e = 0.0`
  The for loop starts with:
  `for (int k = 1, double term = 1; term > epsilon; k++)`
  Inside this for loop, `term` will be updated to equal `term / k` and then added to `e`. Afterwards, `count` will be updated to equal k in order to keep track of the number of terms used. The loop will break when `term` is less then the given epsilon.
- `public int e_terms()` function that simply returns the count value after e() has been called.

madhava.c

Goal: to approximate π using the Madhava Formula.
This file requires newton.c to utilize square roots.

- `static int count = 0`
  Static integer that counts the terms used to compute π
- `public double pi_madhava(void)`
  This function will approximate π using a for loop.
  Starting with outside variables: `double pi = 0.0`
  The for loop starts with:
  `for (int k = 1, double term = -3; term > epsilon or (term * -1)`

`> epsilon; k++)`
Inside this for loop, `term` will be updated to equal `term / -3`. Then `term / 2k +1` will be added to `pi`. Afterwards, `count` will be updated to equal k in order to keep track of the number of terms used. The loop will break when the absolute value of `term` is less then the given epsilon. After the loop, pi should by updated by multiplying itself with $\sqrt{12}$, implemented from the newton.c file. The function will then return the final approximate value of π.

- `public int pi_madhava_terms()` function that simply returns the count value after pi_madhava() has been called.

euler.c

Goal: to approximate π using the Euler Formula.
This file requires newton.c to utilize square roots.

- `static int count = 0`
  Static integer that counts the terms used to compute π

- `public double pi_euler(void)`
  This function will approximate π using a for loop.
  Starting with outside variables: `double pi = 0.0`
  The for loop starts with:
  `for (int k = 1, double term = 1; term > epsilon; k++)`
  Inside this for loop, `term` will be updated to equal `1 / (k * k)`. Then `term` will be added to `pi`. Afterwards, `count` will be updated to equal k in order to keep track of the number of terms used. The loop will break when `term` is less then the given epsilon. Afterwards, the `pi` value will be updated to be the square root of itself (using the sqare root implemented from newton.c) and then divided by 6. The function will then return the final approximate value of π.

- `public int pi_euler_terms()` function that simply returns the count value after pi_euler() has been called.

bbp.c

Goal: to approximate π using the Bailey-Borwein-Plouffe Formula.

- `static int count = 0`
  Static integer that counts the terms used to compute π
- `public double pi_bbp(void)`
  This function will approximate π using a for loop.
  Starting with outside variables: `double pi = 0.0`
  The for loop starts with:
  `for (int k = 1, double term = 1,double exp = 1; term > epsilon or (term * -1) > epsilon; k++)`
  Inside this for loop, `exp` will be updated to be divided by 16 and `term` will be updated to equal `exp * (4/(8*k+1)-2/(8*k+4)-1/(8*k+5)-1/(8*k+6))`. Then `term` will be added to `pi`. Afterwards, `count` will be updated to equal k in order to keep track of the number of terms used. The loop will break when `term` is less then the given epsilon. Afterwards, the function will then return the final approximate value of π.
- `public int pi_bbp_terms()` function that simply returns the count value after pi_bbp() has been called.

viete.c

Goal: to approximate π using the Viete Formula.
This file requires newton.c to utilize square roots.

- `static int count = 0`
  Static integer that counts the terms used to compute π
- `public double pi_viete(void)`
  This function will approximate π using a for loop.
  Starting with outside variables: `double pi = 0.0`

  The for loop starts with:
  `for (int k = 1, double term = 1; term > epsilon; k++)`
  Inside this for loop, `term` will be updated to equal $\sqrt{(\texttt{term} * 2) + 2}/2$ (using square root function from newton.c). Then `term` will be multiplied to the value of `pi`. Afterwards, `count` will be updated to equal k in order to keep track of the number of terms used. The loop will break when `term` is less then the given epsilon. Afterwards, the `pi` value will be updated to be 2 divided by its prior value. The function will then return the final approximate value of π.
- `public int pi_viete_terms()` function that simply returns the count value after pi_viete() has been called.

newton.c

Goal: approximate the square root of an argument.

- `static int count = 0`
  Static integer that counts the iterations used to approximate the square root.

- `public double sqrt_newton(x)` This is based on the python code provided by Professor Long in the documentation for homework 2.

  Create the following variables: `double z,y = 0.0,1.0`
  Create a while loop that stays true while `(y - z) > epsilon or (z - y) > epsilon`
  Inside the loop, set z to y and update y to equal `0.5 * (z + x / z)`.
  Afterwards return y.

- `sqrt_newton_iters()` function that simply returns the count value after sqrt_newton() has been called.

mathlib-test.c

Goal: To test out the library described in this design against the c `math.h` library

Using `getopt`, take parameters from the command line to perfrom actions:

- -a:Runs all tests.
- -e:Runs e approximation test.
- -b:Runs Bailey-Borwein-Plouffe π approximation test.
- -m:Runs Madhava π approximation test.
- -r:Runs Euler sequence π approximation test.
- -v:Runs Viète π approximation test.
- -n:Runs Newton-Raphson square root approximation tests.
- -s:Enable printing of statistics to see computed terms and factors for each tested function.
- -h:Display a help message detailing program usage.