Ralph Miller
CSE13s
10/10/2021

A Little Slice of Pi

Objectives

This program will mimic the C math library <math.h>, implementing a small number of mathematical functions e^x and $\sqrt{x}$. This will be used to approximate different functions using different mathematical methods. The Taylor series, Madhava series, Euler series, Bailey-Borwein-Plouffe formula, Viete's formula, and the Newton-Raphson method.

A test harness shall also be implemented to run all tests, together or individually, using command-line options.

The output will compare the programmed approximations with the <math.h> output, and print any differences between the two.

Taylor Series

The Taylor Series will be used to approximate e^x. Below is the sigma notation for approximating e^x

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} \quad or \quad e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ ...$$

We can write this in C by using a for loop to iterate a counter (n), using this counter variable for the power of x, for the factorial, you can take the previous term and multiply it to the counter (ie. the sum)

Ralph Miller
CSE13s
10/10/2021

Madhava Series

The Madhava Series will be used to approximate π. Below is the sigma notation for approximating π using $tan^{-1} x$.

$$\sqrt{12}\sum_{k=0}^{n}\frac{(-3)^{-k}}{2k+1} \; or \; \sqrt{12}\sum_{k=0}^{n}\frac{1}{(-3)^{k}(2k+1)}$$

This series can be written in C by breaking up the summation into 3 parts. First create a for loop that iterates k in the equation. I would then split the denominator into two parts. Calculate $(-\ 3)^k$ and $(2k\ +\ 1)$ separately, then multiply afterwards. Then complete the equation by dividing $\frac{numerator}{denominator}$. Add the result to a sum variable at the bottom of the for loop. Once the summation is finished. Finally to get the value of pi, you need to solve for pi in this equation.

$$Sum\ =\ \frac{\pi}{\sqrt{12}}\ or,\ \pi\ =\ Sum\ *\ \sqrt{12}$$

Ralph Miller
CSE13s
10/10/2021

Euler Series

The Euler Series will be used to approximate π. Below is the sigma notation for approximating π.

$$\pi = \sqrt{6 \sum_{k=1}^{\infty} \frac{1}{k^2}}$$

To program this in C you will first need to set up a for loop for the summation and iterate through k. Next you will need to compute k^2. To do this I would make a variable for the denominator that is set by the value of $k * k$. Giving you k squared. Then I would divide the $\frac{numerator}{denominator}$ and add the result to a rolling sum. When the summation is finished the last step is to multiply the sum by 6, and take the square root of the result to get π.

$$\pi = \sqrt{6 * Sum}$$

Ralph Miller
CSE13s
10/10/2021

Bailey-Borwein-Plouffe

The Bailey-Borwein-Plouffe formula will be used to approximate π. Below is the sigma notation for approximating π.

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

To program this in C, I would first set up a for loop for the summation, iterating k. Then I would break the equation into two parts, the inner equation and the outer equation. For the inner equation we can just plug in k from our formula and compute. For the outer equation, it can be rewritten as: $\frac{1}{16^k}$. Then we can have a variable compute 16^k and then after divide 1 by the result. To do this I would initialize this variable outside of the for loop storing the value 1. Then when we enter the for loop we can multiply this variable by 16 to get 16^k. ie) $power16 = power16 * 16.$ This will work for all time k > 0. When k = 0, we have a special case where we calculate $\frac{1}{0}$ and in C this will return infinity. To combat this we can set up a special case for when k = 0 using an if statement to set our power variable to 1.

The next step is to multiply the inner and outer equations. Then save the result into a rolling sum. When the summation is over the result will be an approximation of π.

Ralph Miller
CSE13s
10/10/2021

Viete's Formula

The Viete Formula will be used to approximate π

$$\frac{\pi}{2} = \frac{\sqrt{2}}{2} * \frac{\sqrt{2+\sqrt{2}}}{2} * \frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2} * \quad \ldots$$

Viete's Formula is fairly easy to program in C. First setup a while loop for our product series. Then we will have a numerator variable initialized to 0, that will take the sqrt of 2 plus our previous numerator $\sqrt{2 \ + \ prevNumerator}$. This will give us our numerator expanded each iteration, then we divide this by 2 to get our result. Then multiply this into a rolling product sum. After the product series has completed the last step is to solve for π. $ProdSum \ = \ \frac{\pi}{2} \ or, \ \pi \ = \ \frac{2}{ProdSum}$

Ralph Miller
CSE13s
10/10/2021

The Newton-Raphson Method

The Newton-Raphson Method will be used to approximate $\sqrt{x}$. By computing the inverse of x^2.

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

We can write this in C by creating a while loop that takes two variables initialized as , z = 0 & y = 1. And our input x. Swapping the two, then setting $y = \frac{1}{2}(z + \frac{x}{z})$. Until the absolute value of y - z is > EPSILON. The result will be an approximate of $\sqrt{x}$.