Final Project Report

Isaac Chi

***Abstract*—This paper describes the π generating program that I have completed for my final project. First, it will introduce my program idea. Next, it will describe the program's features. Finally, It will describe some trends in the results.**

I. Idea

A. *Origin*

The idea of choosing a π program came about when I was recollecting my past. I remembered a time when our school had π day on March 14, 2015. That would be first and last time that such an event will occur. This was what made me make some a program relating to π.

B. Idea

The idea of my program is to collect some statistics on some formulas(listed below) that converge to π[1]. Specifically, it will show how much precision is gained and the total precision gained after each iteration is added to the partial sum of the sigma term of the formula. π is an irrational constant that represents the ratio of the circumference of a circle to its diameter[2]. No matter what type of formula or trick you use, there is no way one could find the end of π. However, each technique takes a different amount of time to calculate up to a certain place in π. In other words, each formula converges at a different pace to π. I am writing this program to allow people to visualize those speeds.

C. Formulas

*These formulas are all set to equal π.

- Gregory Series [3]

$$4\sum_{i=1}^{\infty}\frac{(-1)^i}{2\mathrm{i}-1}$$

- Nilakantha Series[4]

$$3+4\sum_{i=0}^{\infty}\frac{(-1)^i}{(2\mathrm{i}+2)(2\mathrm{i}+3)(2\mathrm{i}+4)}$$

- Machin's Fomula For π [5]

$$4\sum_{i=0}^{\infty}\frac{(-1)^i}{(2\mathrm{i}+1)}\left(4\left(\frac{1}{5}\right)^{2\mathrm{i}+1}-\left(\frac{1}{239}\right)^{2\mathrm{i}+1}\right)$$

- Newton's Formula For π[6]

$$24\left(\frac{\sqrt{3}}{32}-\sum_{i=0}^{\infty}\frac{(2\mathrm{i})!}{2^{4\mathrm{i}+2}(n!)^2(2\mathrm{i}-1)(2\mathrm{i}+3)}\right)$$

- Ramanujan's Formula For π [7]

$$\frac{9801}{\sqrt{8}}\left[\sum_{i=0}^{\infty}\frac{(4\mathrm{n})!(1103+26390\mathrm{i})}{(n!)^4 396^{4\mathrm{i}}}\right]^{-1}$$

- Chudnovsky Algorithm [8]

$$\frac{1}{12}\left[\sum_{i=0}^{\infty}\frac{(-1)^i(6\mathrm{i})!(13591409+545140134\mathrm{i})}{(3\mathrm{i})!(i!)^3(640320)^{n+\frac{1}{2}}}\right]^{-1}$$

II. Program

A. Input

The user begins by entering which mode they would like. There is "Iteration Mode" and "Digit Mode". In Iteration Mode, the user enters the number of iterations(i) and how precise they want their output to be(n). In Digit Mode, the user types in how many digits they would like to generate(n) and how many iterations each formula is limited to(i). Finally, the user selects which formulas they would like to use. These parameters allow the user to manipulate there input in there own way to fit their capabilities and wants.

I need to process the input for it to be useful in my calculations. First I need to process the selections. I have created several functions, 2 for each formula. One of them is gives the nth term of the sequence in the series and the other gives the approximation given the nth partial sum. I put each pair as a pair of function handles into a case block in a switch statement while looping through the user selections and adding the handles to a cell array. This way I could easily access the functions without needing to create a name for each one. Next, I had to initialize some values depending on which mode is used. If the user selected Iteration Mode, then I would need to create a cell array for reach formulas result. If the user chose Digits Mode, I would need to create an array for the precision data. Finally, I needed to consider one major thing about π that would only matter for digits mode: its irrationality. How would I know that the digits I generate are correct? I would need a template to compare my results with. But π is irrational. And MATLAB has a set precision for doubles. To fix this, I changed the precision of arithmetic to n+4 to guard from potential rounding errors. Then, I used the vpa function (Variable Precision Arithmetic) to get the value of π up to n+4 digits and converted it to a char for simplicity. By the way, there would be no other way to get the value of π up to n digits without using some sort of template. There have not been a digit extraction algorithm to get the nth digit of π(there is one in hexadecimal though.)

B. Calculations

The program will loop through the indices of the formula handles. For both programs, the approximations of π need to be stored for plotting the actual values of π. Variables that need to be updated for every formula include the iteration counter, nth partial sum, and the current value. In general, both modes calculate the nth partial sum of the sum term in the formula by calling the first function with the iteration number given as a parameter. The second function manipulates the nth partial sum so that an approximation is found. At this point, each mode does different tasks.

1. Iteration Mode

After the approximation is calculated, the loop runs again after the counter has been incremented. The program will stop once the counter is equal to i. The final approximation for the formula is stored in a cell array. This is repeated for all formulas.

2. Digits Mode

After the approximation is calculated, it is compared to the template. It will store the amount of precision gained or

lost in a vector in a cell array, as well as the current total precision. The total amount of precision is the same as the first occurrence where the template and the approximation don't match minus 2 (because of the 3 and the decimal point). The precision gained is the current precision minus the the previous precision. The counter is incremented, and the loop loops again. This is repeated for all formulas.

C. *Output*

There are 2 outputs for each mode. There is text output and graphing output. Each mode provides a slightly different output though.

1. *Text output*

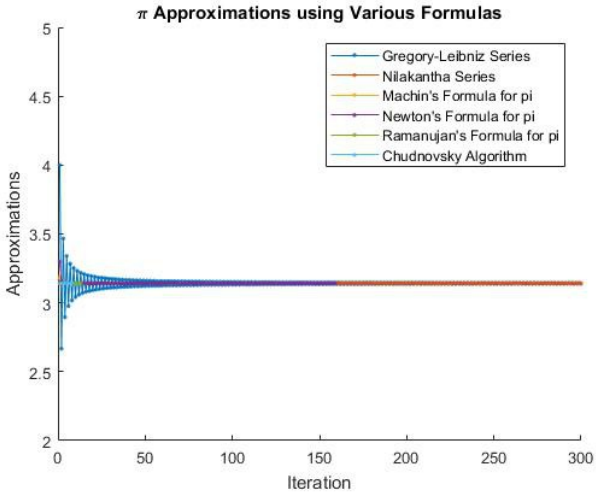
For Digits Mode, I would write all the digits of the template into a file. The file name would be the number of digits calculated plus "digitsOfPi.txt". For example, for 500 digits, the file name would be "500digitsOfPi.txt". In Iterations Mode, I would write all the approximations for each mode into the file. The file name would be the iteration number plus "iterToPi.txt"("500iterToPi.txt"). Despite the differences, I made each line of the file have no more than 70 characters.

2. *Graph output*

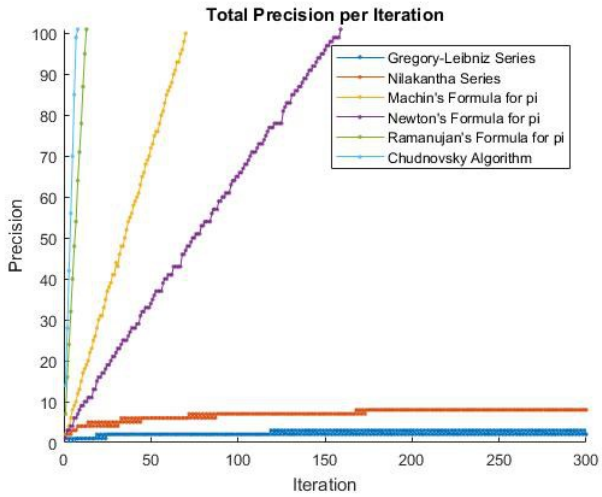
For Iteration Mode, I only have to plot the approximations against the iterations using the stored values which means that there will be a total of one plot. All I had to do was loop through the selections and obtain the corresponding π approximations for the formula. For Digit Mode, in addition to the approximations, I have to plot the precision gained and total precision against the iterations, which means that there will be 3 plots. Because I stored the precision in a same way as the approximations, plotting it was exactly the same. I had to explicitly set annotations for each graph though.

III. Analysis and Results

After executing my program a couple of times, I have noticed some trends in the data. First, the convergence each formula varies by a major amount. As seen in the pictures below, the Chudnosky Algorithm is the most rapidly converging formula and the Gregory Series is the slowest converging formula among the six.

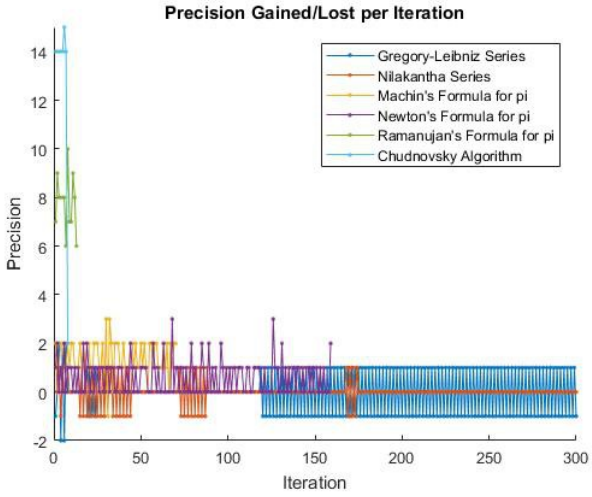


Note: It seems like the Chudnovsky and Ramanujan formulas converge at the same rate. This claim is refuted in the next image.



Note: It is now shown that the Chudnovsky Series converges quicker than the Ramanujan's Formula for π.

Next, I observed that the Nilakantha and the Gregory Series, are the only series that has lost precision. From the first to the 300th iteration, the Gregory series had lost precision about 97 times and the Nilakantha series had lost precision 27 times.



Note: Upon closer inspection, the only series where the precision is lost are the Gregory-Leibniz Series and the Nilakantha Series.

In the end, the order of formulas by convergence speed is the following: Chudnovsky Algorithm, Ramanujan's Formula for π, Machin's Formula for π, Newton's formula for π, Nilakantha Series, and the Gregory Series. There are many more formulas and one can simply create a function for the nth term and the manipulation of the nth partial sum and add a new pair into the switch case structure in the processing of the input. Overall, this project has been interesting and could be expanded on even more as further research into the constant π continues[9].

REFERENCES

[1] Weisstein, Eric W. “Pi Approximations.” From *MathWorld*—A Wolfram Web Resource.
http://mathworld.wolfram.com/PiApproximations.html

[2] Weisstein, Eric W. “Pi.” From *MathWorld*—A Wolfram Web Resource.
http://mathworld.wolfram.com/Pi.html

[3] Weisstein, Eric W. "Gregory Series." From *MathWorld*--A Wolfram Web Resource.
http://mathworld.wolfram.com/GregorySeries.html

[4] “Two Simple Equations to Compute PI,” *Computing & Technology, The Ultimate Computer Technology Blog and The Knowledgebase of Computing*, 20-May-2012. [Online]. Available: https://helloacm.com/two-simple-equations-to-compute-pi/. [Accessed: 21-May-2018].

[5] B. Gourévitch, “The world of Pi – Machin,” *The World of Pi - Home*, 20-Jun-1999. [Online]. Available: http://www.pi314.net/eng/machin.php. [Accessed: 21-May-2018].

[6] B. Gourévitch, “The world of Pi – Newton,” *The World of Pi - Home*, 20-Jun-1999. [Online]. Available: http://www.pi314.net/eng/newton.php. [Accessed: 21-May-2018].

[7] “Ramanujan's formula for pi,” *PlanetMath*, 03-Mar-2013. [Online]. Available: http://planetmath.org/RamanujansFormulaForPi. [Accessed: 21-May-2018].

[8] B. Gourévitch, “The world of Pi – Chudnovski brothers” *The World of Pi - Home*, 20-Jun-1999. [Online]. Available:http://www.pi314.net/eng/chudnovsky.php. [Accessed: 21-May-2018].

[9] Weisstein, Eric W. “Pi Digits.” From *MathWorld*—A Wolfram Web Resource.
http://mathworld.wolfram.com/PiDigits.html