**Ministry of Education and Research of the Republic of Moldova**

**Technical University of Moldova**

**Faculty of Computers, Informatics and Microelectronics**

**REPORT**

Laboratory work no. 6

*Empirical analysis of algorithms*

*that determine a N decimal digit of PI*

Did:

St. gr. FAF-211 Andrei Ceban

Checked:

asist. univ. Cristofor Fistic

Chişinău - 2023

**Laboratory work no. 6**

**Objective:**

1. Implement at least 2 algorithms that determine the Nth decimal digit of Pi in a programming language. (For ten you need to implement 3 algorithms)
2. Choose metrics for comparing algorithms
3. Perform empirical analysis of the proposed algorithms
4. Make a graphical presentation of the data obtained
5. Make a conclusion on the work done.

**INTRODUCTION**

The fascination of the number π by mathematicians is ancient, and numerous computations of its digits have been performed in the history. Later, computers have been used to increase the number of computed digits. The largest computation as of today is impressive : more than 1241 trillions (1.241×1012) digits of π have been recently computed on a super computer by Yasumasa Kanada and his team. Home computers are far from being able to reach these sizes ; for example, the data of the latest computation could fill around one thousand of CD-roms. To over-pass the memory limitation on home computers, pifast extensively makes use of disk memory, but even this possibility does not permit to reach the feat of super computers. For thousands of years, mathematicians have attempted to extend their understanding of π, sometimes by computing its value to a high degree of accuracy. Ancient civilizations, including the Egyptians and Babylonians, required fairly accurate approximations of π for practical computations. Around 250 BC, the Greek mathematician Archimedes created an algorithm to approximate π with arbitrary accuracy. The algorithm maintains a set of visited vertices and a set of unvisited vertices, and at each iteration, it selects the unvisited vertex with the shortest path from the source and adds it to the visited set. Then it updates the distances of the adjacent vertices to the newly visited vertex, if a shorter path is found. The algorithm terminates when all vertices have been visited, and the final distances are the shortest path from the source vertex to all other vertices in the graph. The analysis of these algorithms based on empirical data is vital in order to comprehend their efficiency, accuracy, and convergence characteristics. By examining the performance of these algorithms, we can obtain a deeper understanding of their advantages, disadvantages, and computational needs. This analysis aids in the selection of the most appropriate algorithm for a specific level of precision, optimizing computational resources, and identifying potential areas where algorithms can be enhanced. Our objective is to offer valuable insights into the performance and behavior of these algorithms by conducting extensive computational experiments and numerical simulations. This will enable researchers and practitioners to make informed decisions. Ultimately, this study serves as a thorough exploration of the theoretical foundations, algorithmic advancements, and empirical analysis required to determine N decimal digits of pi. By integrating these aspects, our aim is to provide a better understanding and calculation of this iconic mathematical constant, thereby contributing to the ongoing pursuit of knowledge in the fascinating realm of pi.

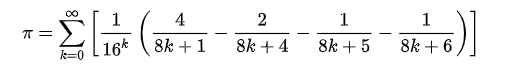
**IMPLEMENTATION**

**The Bailey-Borwein-Plouffe**

The Bailey–Borwein–Plouffe formula is a formula for [*π*](https://en.wikipedia.org/wiki/Pi). It was discovered in 1995 by [Simon Plouffe](https://en.wikipedia.org/wiki/Simon_Plouffe) and is named after the authors of the article in which it was published, [David H. Bailey](https://en.wikipedia.org/wiki/David_H._Bailey_(mathematician)), [Peter Borwein](https://en.wikipedia.org/wiki/Peter_Borwein), and Plouffe. Before that, it had been published by Plouffe on his own site.

The formula depends on the hexadecimal system (base-16) and utilizes powers of 16 in each term. This characteristic enables it to directly compute the hexadecimal digits of pi. The BBP formula offers a distinct and efficient method for calculating specific hexadecimal digits of pi, which proves valuable in diverse applications and computations related to pi.

The formula is



**Algorithm explanation :**

1. Initialize the variables: Set the variables S, n, and k to 0. S will be used to accumulate the value of the sum, n represents the current term, and k is used as a power of 16.
2. Enter a loop: Start a loop that continues indefinitely or until the desired number of digits of pi has been computed.
3. Compute the term: Calculate the current term using the following formula:

term = 1 / (16^n) \* ((4 / (8n + 1)) - (2 / (8n + 4)) - (1 / (8n + 5)) - (1 / (8n + 6)))

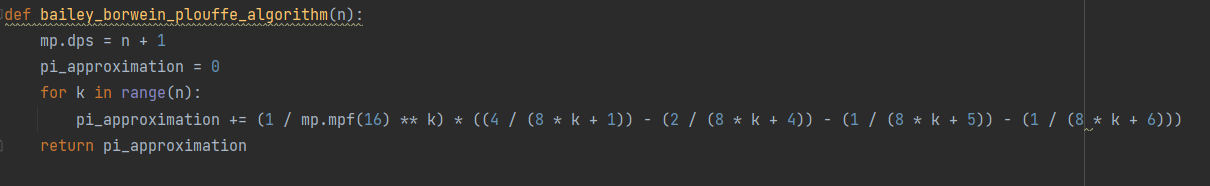
This formula generates the successive terms in the BBP series.

1. Accumulate the sum: Add the computed term to the sum variable S. S = S + term
2. Update the variables: Increment the values of n and k by 1. n = n + 1 k = k \* 16
3. Calculate the hexadecimal digit: Multiply the sum S by 16^k to shift the decimal point and extract the hexadecimal digit at the integer part.

digit = integer\_part(S \* (16^k))

The integer part of the result gives us the hexadecimal digit.

1. Output the digit: Display or store the computed hexadecimal digit.
2. Check termination condition: If the desired number of digits has been reached, exit the loop. Otherwise, return to step 3 and continue the computation.



**The Gauss–Legendre algorithm**

The Gauss–Legendre algorithm is an algorithm to compute the digits of π. It is notable for being rapidly convergent, with only 25 iterations producing 45 million correct digits of π. The algorithm has quadratic convergence, which essentially means that the number of correct digits doubles with each iteration of the algorithm.

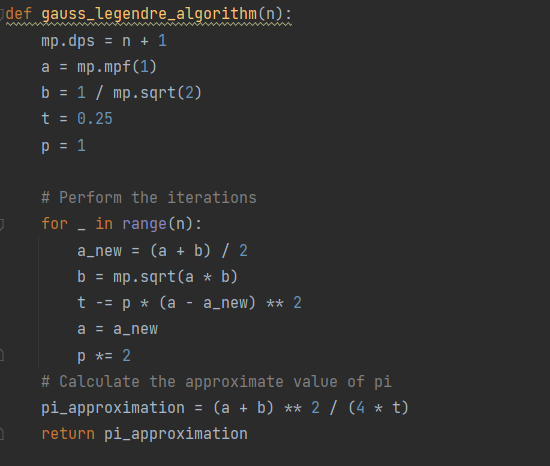
The algorithm starts with an initial guess for pi and iteratively refines the approximation by updating two sequences of numbers, known as the "a" and "b" sequences. These sequences converge to the desired value of pi as more iterations are performed.

The formula is



**Algorithm explanation :**

1. Start with initial values:
   * Set a₀ = 1
   * Set b₀ = 1/√2
   * Set t₀ = 1/4
   * Set p₀ = 1
2. Perform the following iterations until the desired level of precision is achieved:
   * Update aᵢ₊₁ = (aᵢ + bᵢ)/2
   * Update bᵢ₊₁ = √(aᵢ \* bᵢ)
   * Update tᵢ₊₁ = tᵢ - pᵢ \* (aᵢ - aᵢ₊₁)²
   * Update pᵢ₊₁ = 2pᵢ
3. Once the desired level of precision is reached, calculate pi as:
   * pi ≈ (aₙ + bₙ)² / (4tₙ)

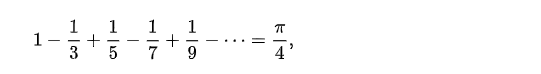


**The Leibniz formula for π**

The Leibniz formula is an infinite series method of calculating Pi. The formula is a very simple way of calculating Pi, however, it takes a large amount of iterations to produce a low precision value of Pi.

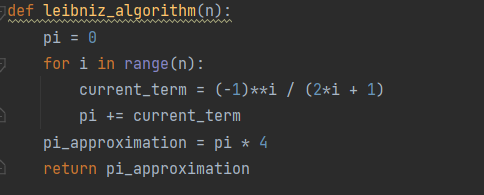
The Leibniz formula for π is a simple and intuitive way to approximate π and has historical significance in the development of mathematical series and the study of π. However, for practical purposes and higher precision, more advanced algorithms and formulas are used.

The formula is

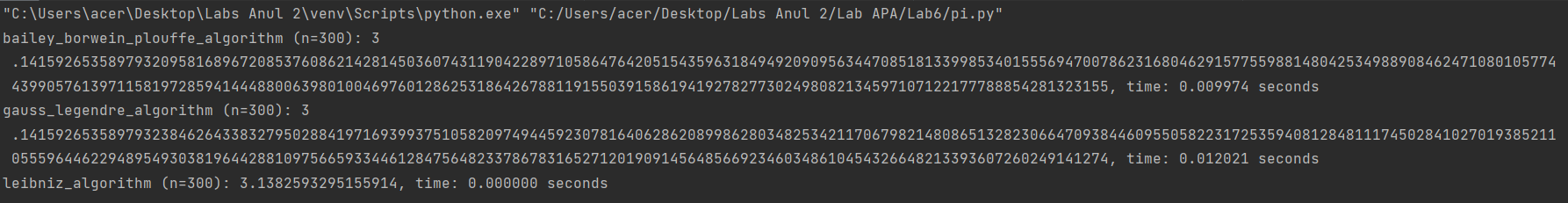


**Algorithm explanation :**

1. Start with an initial value of zero for the sum, let's call it "totalSum."
2. Set up a loop to iterate through the terms of the series. Each term is represented by an index variable, let's call it "n," which starts from zero.
3. Inside the loop, calculate the value of the current term by using the formula (-1)^n / (2n + 1). The exponent (-1)^n alternates between positive and negative for each term, and the denominator (2n + 1) increases with each term.
4. Add the value of the current term to the "totalSum." If it is an odd-numbered term (n is odd), subtract the term from the sum. If it is an even-numbered term (n is even), add the term to the sum.
5. Continue the loop, increasing the value of "n" by 1, and repeat steps 3 and 4 for the desired number of iterations or until the desired level of precision is reached.
6. After the loop finishes, multiply the "totalSum" by 4 to obtain an approximation of π. This is because the sum of the series represents one-fourth of the value of π.



**RESULTS**

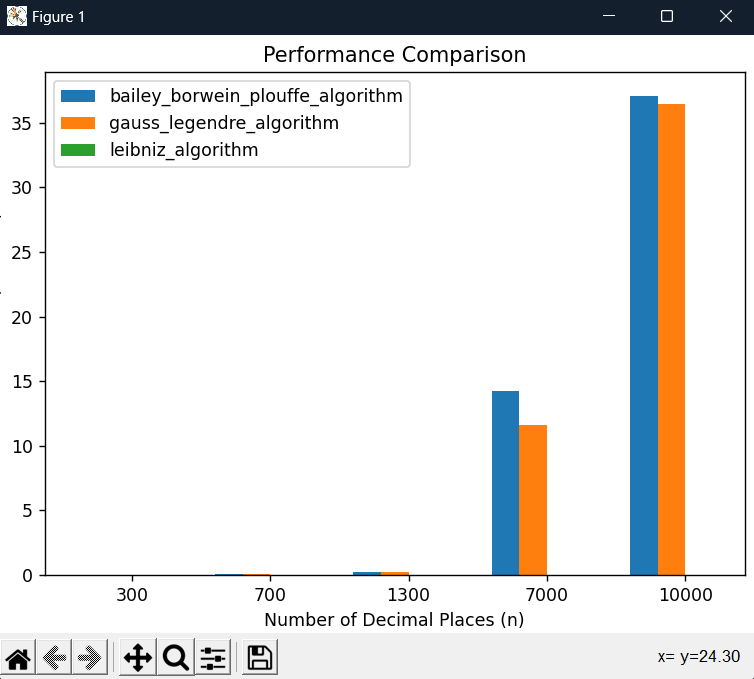


The Leibniz formula, Gauss-Legendre algorithm, and Bailey-Borwein-Plouffe (BBP) algorithm are all methods for approximating the value of π (pi), but they differ in terms of their approaches and computational efficiency.

In terms of time comparison, the Gauss-Legendre algorithm and BBP algorithm generally outperform the Leibniz formula. The Gauss-Legendre algorithm converges rapidly, providing accurate results with fewer iterations, while the BBP algorithm offers a direct calculation of specific digits of π without the need for extensive iterations.

It's important to note that the choice of algorithm depends on the specific requirements of the application. If a high level of precision is needed, more advanced algorithms like the Gauss-Legendre algorithm or BBP algorithm are preferred due to their faster convergence and efficiency.

**Time Comparison for all three algorithms**



**CONCLUSION**

During this lab experiment, I dedicatedly examined, implemented, and scrutinized three distinct algorithms used to calculate the mathematical constant Pi (π) with a specific decimal precision. These algorithms include the Leibniz formula, the Bailey-Borwein-Plouffe (BBP) formula, and the Gauss-Legendre algorithm. Furthermore, I conducted an analysis of the time complexity associated with each algorithm, offering valuable insights into their respective performance characteristics.

After comparing the three pi calculation algorithms, Leibniz formula, Gauss-Legendre, and Bailey-Borwein-Plouffe (BBP), we can draw the following conclusions:

Leibniz Formula: The Leibniz formula for π is a simple and intuitive algorithm. However, it has a slow convergence rate, meaning that it requires a large number of iterations to achieve high accuracy. Consequently, it is not the most efficient algorithm for computing π, especially when compared to more advanced algorithms like Gauss-Legendre and BBP.

Gauss-Legendre Algorithm: The Gauss-Legendre algorithm is an iterative method that converges rapidly and provides highly accurate approximations of π. It is known for its efficiency and precision. The algorithm is based on a series of iterations that refine the approximation by updating two sequences of numbers. It is commonly used in numerical computations and serves as a benchmark for evaluating the performance of other pi approximation algorithms.

Bailey-Borwein-Plouffe (BBP) Algorithm: The BBP algorithm provides a unique and efficient approach for calculating individual hexadecimal digits of π. It relies on the hexadecimal system and involves powers of 16 in its formula. The algorithm is particularly useful when direct computation of hexadecimal digits of π is required. Compared to the Leibniz formula, BBP offers faster convergence and higher precision.

To summarize, when it comes to calculating π, the choice of algorithm depends on the precision requirements and the specific use-case. The Leibniz formula and the BBP formula offer simplicity and are suitable for lower precision needs or when computing individual digits of π is the primary goal. However, if the task requires a high number of decimal places for π, the Gauss-Legendre algorithm is the optimal choice. It is important to consider the nature of the task and the available resources before selecting the most suitable algorithm.

Git Repo : https://github.com/andeiceban0352/Labs-Anul2/tree/main/Lab%20APA/Lab6