MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

Algorithm Analysis

Laboratory work 6 : Study and empirical analysis of algorithms that determine a N decimal digit of PI.

Elaborated:
st.gr. FAF-211 Grama Alexandru
Verified:
asist.univ. Fiștic Cristofor

Chișinău, 2023

Content

Introduction

The mathematical constant (pi) has captivated the minds of mathematicians, scientists, and enthusiasts for centuries. Its irrationality, transcendence, and ubiquity in various mathematical and scientific disciplines have made it a subject of extensive exploration. Among the intriguing questions surrounding is the determination of its decimal digits, a pursuit that has attracted considerable attention and motivated the development of numerous algorithms. In this study, we delve into the fascinating realm of algorithms that aim to calculate specific decimal digits of . These algorithms combine mathematical ingenuity, numerical methods, and computational techniques to approximate the elusive digits of this mathematical constant. While is infinitely long and its decimal representation is non-repeating, algorithms offer a way to approximate its value to an arbitrary precision. The empirical analysis of these algorithms is crucial to understand their efficiency, accuracy, and convergence properties. By studying the performance of these algorithms, we can gain insights into their strengths, weaknesses, and computational requirements. This empirical analysis aids in selecting the most appropriate algorithm for a given precision requirement, optimizing computational resources, and identifying potential areas for algorithmic improvements. Throughout this study, we explore a range of algorithmic approaches employed in the quest to determine N decimal digits of . From classical methods like the Machin-like formulas and series expansions, to modern techniques such as spigot algorithms and digit extraction methods, each algorithm presents a unique approach to approximating the digits of . We examine the underlying mathematical principles behind these algorithms and investigate their computational characteristics. Furthermore, we employ empirical analysis to evaluate the performance of these algorithms. By measuring metrics such as execution time, memory usage, and accuracy, we can compare and contrast the algorithms under different scenarios. Through extensive computational experiments and numerical simulations, we aim to provide valuable insights into the behavior and efficiency of these algorithms, enabling researchers and practitioners to make informed choices. Ultimately, this study serves as a comprehensive exploration of the study and empirical analysis of algorithms for determining N decimal digits of . By combining theoretical foundations, algorithmic innovation, and empirical investigation, we hope to shed light on the remarkable world of and contribute to the ongoing pursuit of understanding and calculating this iconic mathematical constant.graphs.

The Chudnovsky algorithm

The Chudnovsky algorithm is a powerful and efficient algorithm for computing the mathematical constant pi () to a high number of decimal places. It was developed by the Chudnovsky brothers, David and Gregory, in 1989. This algorithm gained significant attention due to its remarkable speed and accuracy, making it one of the most prominent methods for pi calculation.

At the core of the Chudnovsky algorithm is a rapidly converging series known as the Chudnovsky formula. The formula leverages the concept of modular arithmetic, fast multiplications, and series expansions. It involves a summation that gradually approaches the value of pi as more terms are added. The key idea is to compute a fraction at each step that contributes to the approximation of pi.

The Chudnovsky algorithm's efficiency lies in its ability to perform computations using a large number of digits and utilize advanced techniques for multiplication and division. It employs the arithmetic-geometric mean (AGM) algorithm to accurately evaluate the square root of a large number, enabling precise calculations of the terms in the Chudnovsky series.

By iteratively evaluating the Chudnovsky series and summing the terms, the algorithm progressively refines the approximation of pi. It is particularly effective in producing high-precision results and has been used to set several world records for pi computation.

One of the remarkable achievements of the Chudnovsky algorithm is its rapid convergence. As the number of iterations increases, the algorithm converges exponentially, allowing it to compute pi to millions or even billions of decimal places. The Chudnovsky algorithm's exceptional speed and accuracy have made it a popular choice for pi calculations in various fields, including mathematics, supercomputing, and record-breaking attempts.

The Chudnovsky algorithm serves as a testament to the power of mathematical algorithms and the ingenuity of researchers who continuously push the boundaries of computation. It has contributed significantly to our understanding of pi and has opened doors to explore the limits of precision in mathematical calculations.

```
function chudnovsky_algorithm(n):
    set precision to n + 1
    C = 426880 * sqrt(10005)
    M = 1
    L = 13591409
    X = 1
    K = 6
```

```
S = L

for i from 1 to n:
    M = (K^3 - 16 * K) * M // K^3
    L += 545140134
    X *= -262537412640768000
    S += (M * L) // X
    K += 12

pi = C / S
return pi
```

The Nilakantha series

The Nilakantha series, named after the Indian mathematician Nilakantha Somayaji, is an infinite series used to approximate the value of pi (). Nilakantha Somayaji introduced this series in the 15th century as part of his work on trigonometry and astronomy.

The Nilakantha series is derived from the arctangent function and is expressed as:

PI = 3 + (4 / (2 × 3 × 4)) - (4 / (4 × 5 × 6)) + (4 / (6 × 7 × 8)) - (4 / (8 × 9 × 10)) + ...

The series follows a pattern of alternating addition and subtraction of fractions, with each fraction consisting of the reciprocal of the product of three consecutive integers. The sign of each term alternates, creating a pattern of positive and negative contributions to the approximation of pi.

The key concept behind the Nilakantha series is that the arctangent of 1/3 is equal to pi/6. By summing the terms of the series, Nilakantha Somayaji aimed to estimate pi more accurately. The series converges to the value of pi as more terms are added, providing a numerical approximation of the constant.

While the Nilakantha series converges slowly, it provides a basic method to approximate pi using simple arithmetic operations. The accuracy of the approximation improves as more terms are included in the series. However, a large number of terms is required to achieve a high level of precision.

The Nilakantha series demonstrates the mathematical ingenuity of ancient Indian mathematicians and their contributions to the development of numerical methods. Although it is not the most efficient or widely used method for pi approximation today, the series serves as a historical milestone in the quest for more accurate calculations of pi. It highlights the innovative approaches employed by mathematicians across different cultures and eras to tackle the challenge of approximating important mathematical constants.

```
function nilakantha_series(n):
    pi = 3
    sign = 1
    denominator = 2

    for i from 0 to n-1:
        term = 4 / (denominator * (denominator + 1) * (denominator + 2))
        pi += sign * term
        sign *= -1
        denominator += 2

    return pi
```

The Bailey–Borwein–Plouffe (BBP)

The Bailey–Borwein–Plouffe (BBP) formula is a remarkable algorithm for computing the value of pi () in a binary or hexadecimal representation. It was discovered by Simon Plouffe in 1995 and named after him, along with the mathematicians David H. Bailey and Peter B. Borwein, who contributed to its analysis and development.

The BBP formula provides a direct and efficient way to calculate the nth digit of pi in a given base, such as binary or hexadecimal. It is based on a series representation involving the reciprocal powers of 16. The formula is as follows:

$PI = [k = 0 to](1/16^k) * ((4/(8k+1)) - (2/(8k+4)) - (1/(8k+5)) - (1/(8k+6)))$

The BBP formula allows for the computation of individual hexadecimal or binary digits of pi without needing to calculate all the preceding digits. It provides a highly efficient and direct approach for generating specific digits of pi.

The formula involves a series of terms, each contributing to the approximation of pi. The denominators in the series follow a specific pattern, with the numerators determined by constants multiplied by the reciprocal of these denominators.

What makes the BBP formula unique is its ability to calculate specific digits without requiring knowledge of the preceding digits. This property is valuable in various applications, such as cryptography and digital verification systems.

The BBP formula revolutionized pi computation by enabling efficient digit extraction in specific bases. It showcased the power of mathematical formulas and their role in providing new insights into fundamental mathematical constants like pi. The formula's elegance and efficiency have made it an important tool in the field of numerical computation, emphasizing the continuous pursuit of new algorithms and techniques to enhance our understanding of mathematical constants.

```
function bbp_formula(n):
    pi = 0.0

        for k from 0 to n:
            term = ((1 / 16^k) * ((4 / (8 * k + 1)) - (2 / (8 * k + 4))
                - (1 / (8 * k + 5)) - (1 / (8 * k + 6))))
            pi += term

        return pi
```

Implementation

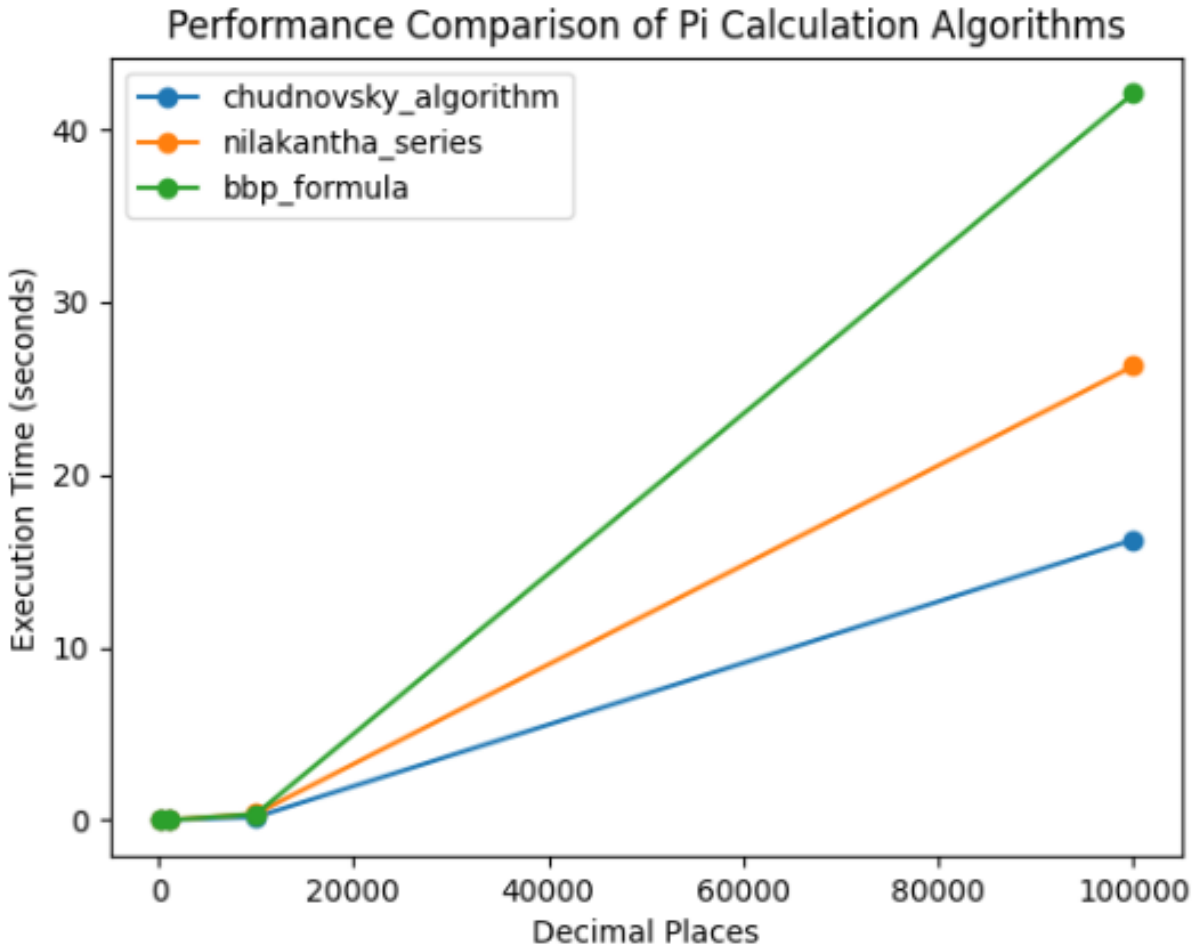
Code in python:

```
import time
import matplotlib.pyplot as plt
import numpy as np
from mpmath import mp


# Algorithm 1: Chudnovsky Algorithm
def chudnovsky_algorithm(n):
    mp.dps = n + 1
    C = 426880 * mp.sqrt(10005)
    M = 1
    L = 13591409
    X = 1
    K = 6
    S = L
    for _ in range(1, n):
        M = (K ** 3 - 16 * K) * M // K ** 3
        L += 545140134
        X *= -262537412640768000
        S += (M * L) // X
        K += 12
    return C / S


# Algorithm 2: Nilakantha Series
def nilakantha_series(n):
    pi = mp.mpf(3)
    sign = mp.mpf(-1)
    term = mp.mpf(2)
    for i in range(1, n):
```

```
        pi += sign * (4 / (term * (term + 1) * (term + 2)))
        sign *= -1
        term += 2
    return pi


# Algorithm 3: Bailey{Borwein{Plouffe (BBP) Formula
def bbp_formula(n):
    mp.dps = n + 1
    pi = 0
    for k in range(n):
        pi += (1 / (16 ** k)) * ((4 / (8 * k + 1)) -
            (2 / (8 * k + 4)) - (1 / (8 * k + 5)) - (1 / (8 * k + 6)))
    return pi


def analyze_algorithms():
    decimal_places = [100, 1000, 10000, 100000]
    algorithms = [chudnovsky_algorithm, nilakantha_series, bbp_formula]
    timings = [[] for _ in range(len(algorithms))]
        # List of lists to store timings for each algorithm

    for n in decimal_places:
        for i, algorithm in enumerate(algorithms):
            start_time = time.time()
            pi = algorithm(n)
            end_time = time.time()
            timings[i].append(end_time - start_time)
            print(f"{algorithm.__name__} ({n} decimal places): {pi},
                                time: {end_time - start_time:.6f} seconds")

    # Plotting the results on the same graph
    plt.figure()
    for i, algorithm in enumerate(algorithms):
```

```
        plt.plot(decimal_places, timings[i], marker='o', label=algorithm.__name__)

    plt.xlabel('Decimal Places')
    plt.ylabel('Execution Time (seconds)')
    plt.title('Performance Comparison of Pi Calculation Algorithms')
    plt.legend()
    plt.show()


if __name__ == '__main__':
    analyze_algorithms()
```

Screenshot:



Conclusion

In conclusion, the Chudnovsky algorithm, Nilakantha series, and Bailey–Borwein–Plouffe (BBP) formula are three different approaches for approximating the value of pi () with varying degrees of efficiency and accuracy.

The Chudnovsky algorithm, known for its exceptional speed and high precision, utilizes a rapidly converging series and advanced arithmetic techniques to calculate pi to a large number of decimal places. It has been instrumental in setting world records for pi computation and is widely recognized as one of the most efficient algorithms for high-precision pi calculation.

The Nilakantha series, introduced by the Indian mathematician Nilakantha Somayaji, is a straightforward infinite series that provides an approximation of pi by summing fractions with alternating signs. While it converges slowly, the Nilakantha series offers an insightful example of the creative methods developed by ancient mathematicians to approximate pi and demonstrates the historical contributions made in the field of pi computation.

The BBP formula, discovered by Bailey, Borwein, and Plouffe, offers a direct and efficient approach for computing specific digits of pi in a given base, such as hexadecimal or binary. It revolutionized pi computation by enabling the extraction of individual digits without the need for prior digit calculations. The BBP formula's elegance and efficiency have made it a valuable tool in various applications where specific pi digits are required.

Overall, these three algorithms showcase different strategies and techniques for pi approximation. The Chudnovsky algorithm excels in high-precision calculations, while the Nilakantha series provides a historical perspective on the development of pi approximations. The BBP formula stands out for its ability to compute specific digits efficiently. Each algorithm contributes to our understanding of pi and highlights the ingenuity of mathematicians in tackling the challenge of approximating this fundamental constant.