УДК 004.942
doi:10.21685/2227-8486-2021-2-9

ВЫЧИСЛЕНИЕ ЧИСЛА ПИ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ OpenMP

О. А. Бакаева

Национальный исследовательский Мордовский государственный университет
имени Н. П. Огарева, Саранск, Россия
helga_rm@rambler.ru

Аннотация. *Актуальность и цели.* Цель исследования – провести анализ различных методов вычисления числа Пи с использованием языка программирования C++ и сравнить сходимость, точность и скорость вычисления для этих методов с использованием технологии параллельного программирования и без нее. *Материалы и методы.* На основе известных математических выражений, которые являются качественной аппроксимацией числа Пи: ряды Грегори–Лейбница, Мадхавы, Нилаканта, формулы Эйлера и Валлиса – вычисляется приближенное значение числа Пи с точностью до 10^{-10} и количеством слагаемых до 10^6. Расчеты производятся в среде Visual Studio на языке программирования C++ стандартным способом и с использованием технологии параллельных вычислений OpenMP. *Результаты.* Для каждой из расчетных формул найдены абсолютная и относительная ошибки вычисления числа Пи. Проведен сравнительный анализ результатов, полученных в среде Free Pascal, C++ и C++ с использованием технологии OpenMP. *Выводы*. Выявлен самый эффективный способ вычисления числа Пи, учитывая сходимость ряда, точность и время вычислений.

Ключевые слова: число Пи, технологии параллельного программирования, OpenMP, Visual Studio, язык программирования C++, ряд Грегори–Лейбница, ряд Мадхавы, ряд Нилаканта, формула Эйлера, формула Валлиса

Для цитирования: Бакаева О. А. Вычисление числа Пи с использованием технологии параллельного программирования OpenMP // Модели, системы, сети в экономике, технике, природе и обществе. 2021. № 2. С. 130–143. doi:10.21685/2227-8486-2021-2-9

CALCULATING THE NUMBER PI USING DATA PARALLEL PROGRAMMING OpenMP

O.A. Bakaeva

National Research Ogarev Mordovia State University, Saransk, Russia
helga_rm@rambler.ru

Abstract. *Background.* To analyze various methods for calculating the number Pi using the C ++ programming language and compare the convergence, accuracy and computation speed for these methods with and without data parallel programming. *Materials and methods.* Based on the well-known mathematical expressions that are a qualitative approximation of the number Pi: the Gregory–Leibniz, Madhava, Nilakant series, Euler and Wallis formulas, the approximate value of Pi is calculated with an accuracy of 10^{-10} and the number of terms up to 10^6. The calculations are performed in the Visual Studio environment on the C ++ programming language in a standard way and using the OpenMP parallel compu-

ting technology. *Results.* For each of the calculation formulas, the absolute and relative error in calculating the number Pi is found. A comparative analysis of the results obtained in Free Pascal, C ++ and C ++ using OpenMP technology is carried out. *Conclusions.* The most efficient way of calculating the number Pi was revealed, taking into account the convergence of the series, the accuracy and time of calculations.

Keywords: number Pi, data parallel programming, OpenMP, Visual Studio, C ++ programming language, Gregory–Leibniz series, Madhava series, Nilakant series, Euler's formula, Wallis's formula

For citation: Bakaeva O.A. Calculating the number of Pi using data parallel programming OpenMP. *Modeli, sistemy, seti v ekonomike, tekhnike, prirode i obshchestve = Models, systems, networks in economics, technology, nature and society*. 2021;2:130–143. (In Russ.). doi:10.21685/2227-8486-2021-2-9

Введение

Вопрос о величине, природе, способах вычисления и точности числа Пи интересовал ученых в различные времена. Эта тематика остается актуальной и в настоящее время. О том, как далеко продвинулась вычислительная наука, можно судить по количеству знаков в дробной части числа Пи, которое уже удалось определить на текущий момент времени.

Число Пи – фундаментальная математическая константа, равная отношению длины окружности к ее диаметру. Значение 3,14 является достаточно приблизительным и используется для обучения и простоты расчетов [1].

В настоящее время число Пи применяется в прикладных вычислениях во многих областях: электротехника, электроника, физика, космос, теория вероятностей, строительство, навигация и пр.

Число Пи настолько многогранно, что существует много подходов к его вычислению. Наиболее часто встречается геометрический подход к исследованию числа Пи. Так, в работах С. П. Коростелева описывается коррекция значения числа Пи на основании абсолютно точных решений задач квадратуры круга и удвоения куба [2, 3]. А коллектив авторов В. И. Чепасов, М. А. Токарева, О. В. Буреш вычисляет число Пи методом касательных в длинной арифметике [4].

В работе В. С. Сенашова, И. Л. Савостьянова [5] отражен геометрический и вероятностный смысл числа Пи, в [6] представлен информационный подход и АСК-анализ, а В. А. Андросенко исследует иррациональность выражения $\frac{\pi}{\sqrt{3}}$ с помощью новой интегральной конструкции [7]. Достаточно часто число Пи пересекается с физикой [8].

В зарубежных работах тоже отмечается важность использования числа Пи в инженерии и науке и приводятся классические способы его вычисления. Например, в [9] представлен исторический обзор методов вычисления данной константы: от Архимеда, использовавшего правильные многоугольники $\left(\pi \approx \frac{22}{7}\right)$, китайского математика Цзу Чунчжи (вычислил 6 знаков в дробной части), малоизвестных рядов Грегори–Лейбница и Нилаканта до компьютерных вычислений с точностью до 13 300 000 000 000 знаков после запятой.

Число Пи нашло свое применение в архитектуре, искусстве и предметах культуры древних народов [10]. По размерам древних украшений в виде

дисков, спиралей и пр. можно найти значение Пи и получить представление о знаниях геометрии в доисторические времена.

Начиная с середины XX в. для вычисления числа Пи использовались компьютеры и их вычислительные мощности. Огромный вклад в нахождение максимального количества знаков в дробной части числа Пи внесли Дж. фон Нейман (1949 г.), Ф. Женюи (1959 г.), Дж. Гийу и М. Буйе (1973 г.), Братья Чудновские (1989 г.), Я. Канада (2002 г.), А. Йи и С. Кондо [11]. Перспективной стала формула Бэйли–Боруэйна–Плаффа, открытая С. Плаффом [12].

Самый полный и детальный хронологический обзор методов нахождения числа Пи на английском языке представлен в работе R. P. Agarwal, H. Agarwal, S. K. Sen [13].

Причем с прогрессом вычислительной техники и увеличением мощностей современных компьютеров открываются новые возможности изучения самой константы, способов ее вычисления и ее практического применения [14].

Материалы и методы

С развитием техники и внедрением персональных компьютеров математическую природу числа Пи можно исследовать, используя мощные и быстрые вычислительные средства [15, 16].

В современном мире начали быстро развиваться и уже достигли определенного уровня технологии параллельного программирования. Благодаря присущим им преимуществам были решены некоторые классы задач, которые до внедрения данных методов оставались нерешенными или имели приближенное решение, или, что случалось чаще всего, требовали больших временных и вычислительных затрат. С использованием технологии параллельного программирования стало возможным решение задач с большими объемами данных и многочисленными операциями.

В настоящее время одной из самых используемых технологий параллельного программирования является OpenMP [17].

OpenMP (Open Multi-Processing) – это набор директив компилятора, библиотечных процедур и переменных окружения, которые предназначены для программирования многопоточных приложений на многопроцессорных системах с общей памятью (SMP-системах).

При использовании технологии OpenMP разработчик не создает новую параллельную программу, а просто добавляет в текст последовательной программы OpenMP-директивы. При этом система программирования OpenMP предоставляет разработчику большие возможности по контролю над поведением параллельного приложения. Вся программа разбивается на последовательные и параллельные области. Все последовательные области выполняет главная нить, возникающая при запуске программы, а при входе в параллельную область главная нить порождает дополнительные нити [18].

Данная технология может быть эффективно реализована с помощью среды Visual Studio и языка программирования C++. Для обеспечения работоспособности OpenMP необходимо включить поддержку OpenMP в свойстве проекта и подключить одноименную библиотеку через *#include <omp.h>*.

Вычисление числа Пи можно проводить с помощью различных вычислительных инструментов [19]. Для этого подходят языки программирования Pascal, C++, математические пакеты Mathcad, MATLAB и др. Даже широко

используемый табличный процессор MS Excel позволяет вычислить данную константу с вполне приемлемой точностью. Но данные методы и инструменты стандартны и известны [20].

В данной статье для вычисления числа Пи через сумму ряда предлагается использовать технологию параллельного программирования OpenMP. Благодаря распараллеливанию действий в циклах появляется возможность увеличивать скорость вычислений. А увеличивая количество слагаемых и число итераций в программе, можно получить любое количество знаков в дробной части числа и тем самым необходимую точность.

Существует достаточно много последовательностей и рядов, которые сходятся к числу Пи или приближенно равны данной константе. Это ряд Грегори–Лейбница, ряд Мадхавы, ряд Нилаканта, формулы Эйлера и Валлиса и др. [19, 20].

1. Ряд Грегори–Лейбница имеет вид

$$\pi = 4 - \frac{4}{4} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} \pm \ldots \quad (1)$$

Его недостаток состоит в том, что он сходится очень медленно, что приводит к объемным рутинным вычислениям.

2. Ряд Мадхавы определяется выражением

$$\pi = \sqrt{12} \cdot \left(1 - \frac{1}{3 \cdot 3} + \frac{1}{5 \cdot 3^2} - \frac{1}{7 \cdot 3^3} + \ldots\right). \quad (2)$$

3. Ряд Нилаканта имеет вид

$$\pi = 3 + \frac{4}{2 \cdot 3 \cdot 4} - \frac{4}{4 \cdot 5 \cdot 6} + \frac{4}{6 \cdot 7 \cdot 8} - \frac{4}{8 \cdot 9 \cdot 10} + \frac{4}{10 \cdot 11 \cdot 12} + \ldots \quad (3)$$

4. Формула Эйлера:

$$\pi^2 = 6 \cdot \left(\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots\right). \quad (4)$$

5. Формула Валлиса:

$$\pi = 2 \cdot \frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \frac{6}{7} \cdot \frac{8}{7} \cdot \frac{8}{9} \cdot \ldots \quad (5)$$

Необходимо выяснить, какой из указанных рядов быстрее будет сходиться к числу Пи и сколько слагаемых (множителей для формулы Валлиса) необходимо вычислить, чтобы точность была больше 10^{-10}, а также определить и сравнить время вычислений при обычном программировании и использовании технологии OpenMP.

Для решения данной задачи проведем расчеты в среде Visual Studio с помощью языка программирования C++ и технологии параллельного программирования OpenMP. Далее сравним полученные результаты по нескольким характеристикам. Во-первых, определим, получились ли одинаковые результаты вычисления числа Пи при решении в данном случае и в работе, когда для расчетов использовалась среда Free Pascal [20]. Во-вторых, если

значения суммы ряда различны, сделаем вывод об изменении скорости сходимости. И, в-третьих, сравним результаты вычислений и время работы программы для вычисления числа Пи в C++ без использования технологии параллельного программирования и с применением ее (в частности использованием технологии OpenMP).

Результаты

После разработки алгоритмов вычисления числа Пи по формулам (1)–(5) был написан код, позволяющий реализовать технологию OpenMP.

Листинг кода вычисления числа Пи с помощью технологии OpenMP (метод Грегори–Лейбница)

```
#include "stdafx.h"
#include <iostream>
#include <omp.h>
void task_1(int n)

{
	double pi = 0;
	double start, end;
	start = omp_get_wtime();

#pragma omp parallel reduction(+:pi)
	{
	  #pragma omp for
		for (int i = 0; i < n; i++)
		{
			pi = pi + 4.0 * pow(-1, i) / (2.0 * i + 1.0);
		}
	}
	end = omp_get_wtime();
	printf("Значение pi для первых %d слагаемых по формуле Грегори-
		Лейбница: %.10f\n",
	n, pi);
	printf("Время по формуле Грегори-Лейбница: %.6f \n", end-
start);
}

int main()
{
	setlocale(LC_ALL, "Russian");
	for (int i = 10; i <= 1000000; i = i * 10)
	{
	task_1(i);
	}
    return 0;
}
```

1. Проведем расчеты и вычислим число Пи в C++ с точностью до 10^{-10} методом Грегори–Лейбница, найдем и сравним абсолютную и относительную ошибки (табл. 1).

Таблица 1

Вычисление числа Пи методом Грегори–Лейбница в C++

Число слагаемых	Полученное значение в Free Pascal*	Полученное значение в C++	Δ для (C++)	ε для (C++), %
$n = 10$	3,2323158094	3,0418396189	0,0997530347	3,1752
$n = 100$	3,1514934011	3,1315929036	0,0099997500	0,3183
$n = 1000$	3,1425916543	3,1405926538	0,0009999998	0,0318
$n = 10\,000$	3,1416926436	3,1414926536	0,0001000000	0,0032
$n = 100\,000$	3,1416026535	3,1415826536	0,0000100000	0,0003
$n = 1\,000\,000$	3,1415946536	3,1415906536	0,0000020000	0,0001

П р и м е ч а н и е. * – здесь и далее результаты вычислений взяты из работы [20].

Как видно, метод Грегори–Лейбница даже при $n = 10^6$ слагаемых не дает результат, верный с точностью 10^{-10}. Абсолютная ошибка $>10^{-7}$, относительная ≈ 0,0001 %.

Значения числа Пи, полученные по данной формуле, без использования технологии параллельного программирования и с ее использованием получились одинаковыми. Такое будет наблюдаться и при вычислениях по другим формулам (2)–(5).

Значение времени при обычных вычислениях, начиная со значения $n = 100$, непрерывно растет, что объясняется увеличением числа слагаемых (табл. 2).

Таблица 2

Сравнение времени вычисления числа Пи методом Грегори–Лейбница

Число слагаемых	Полученное значение C++	t_{C++}	Полученное значение $C++_{OpenMp}$	$t_{C++\ OpenMp}$
$n = 10$	3,0418396189	0,000017	3,0418396189	0,000283
$n = 100$	3,1315929036	0,000009	3,1315929036	0,000034
$n = 1000$	3,1405926538	0,000084	3,1405926538	0,000109
$n = 10\,000$	3,1414926536	0,000842	3,1414926536	0,000883
$n = 100\,000$	3,1415826536	0,008451	3,1415826536	0,010765
$n = 1\,000\,000$	3,1415916536	0,091316	3,1415916536	0,066932

При параллельных вычислениях наблюдается та же тенденция. Так как значение числа Пи не достигло требуемой точности, в дальнейшем анализировать время, затраченное на вычисления, нет смысла: результат не удовлетворяет заявленной точности.

Можно только заметить, что вычисления с использованием технологии параллельного программирования до $n = 10^6$ занимают больше времени, чем обычные методы. А видимый эффект достигается при $n = 10^6$ и составляет ≈ 0,03 с.

2. Рассмотрим результаты вычислений, полученные с использованием ряда Мадхавы (табл. 3).

Фрагмент листинга кода, реализующий вычисление числа Пи, через ряд Мадхавы

```
void task_2(int n)
{
	double pi = 0;
	double start, end;
	start = omp_get_wtime();

#pragma omp parallel reduction(+:pi)
	{
#pragma omp for
		for (int i = 0; i < n; i++)
		{
			pi = pi + sqrt(12) * pow(-1, i) / ((2.0 * i + 1) * 
pow(3, i));
		}
	}

	end = omp_get_wtime();
	printf("Значение pi для первых %d слагаемых,используя ряд Мад-
хавы:%.10f\n",n, pi);
	printf("Время: %.6f \n", end - start);
}
```

Таблица 3

Вычисление числа Пи с помощью ряда Мадхавы в C++

Число слагаемых	Полученное значение Free Pascal	Полученное значение C++	Δ C++	ε, %
n = 10	3,2323158094	3,1415905109	0,0000021427	0,0001
n = 100	3,1514934011	**3,1415926536**	0,0000000000	0,0000
n = 1000	3,1425916543	**3,1415926536**	0,0000000000	0,0000
n = 10 000	3,1416926436	**3,1415926536**	0,0000000000	0,0000
n = 100 000	3,1416026535	**3,1415926536**	0,0000000000	0,0000
n = 1 000 000	3,1415946536	**3,1415926536**	0,0000000000	0,0000

Реализация алгоритма Мадхавы в C++ приводит к более точным результатам, чем предыдущие расчеты и за меньшее количество действий (табл. 4).

Таблица 4

Сравнение времени вычисления числа Пи с помощью ряда Мадхавы

Число слагаемых	Полученное значение C++	t_{C++}	Полученное значение C++ $_{OpenMp}$	$t_{C++\ OpenMp}$
n = 10	3,1415905109	0,000029	3,1415905109	0,039706
n = 100	**3,1415926536**	**0,000023**	**3,1415926536**	**0,000021**
n = 1000	**3,1415926536**	**0,000281**	**3,1415926536**	**0,000206**
n = 10 000	**3,1415926536**	**0,003524**	**3,1415926536**	**0,002405**
n = 100 000	**3,1415926536**	**0,036854**	**3,1415926536**	**0,021092**
n = 1 000 000	**3,1415926536**	**0,355709**	**3,1415926536**	**0,206818**

Так как вычисляемое значение числа Пи достигает точности 10^{-10} при $n = 100$, можно сделать вывод о быстрой сходимости ряда. Время на вычисления увеличивается пропорционально числу слагаемых, но при параллельных вычислениях данный показатель значительно меньше. Получается, что точное значение (абсолютная ошибка $< 10^{-10}$) можно вычислить за $t_{C++} = 0{,}000023$ с и $t_{C++\ OpenMp} = 0{,}000021$ с.

И в целом при использовании ряда Мадхавы, за исключением $n = 10$, времени на расчеты при использовании технологии параллельного программирования тратится меньше, чем при классических методах.

3. Рассмотрим результаты вычислений, полученные с помощью ряда Нилаканта (табл. 5).

Фрагмент листинга кода, реализующий вычисление числа Пи через ряд Нилаканта

```
void task_3(int n)
{
	double pi = 3.0;
	int znak = 1;
	int i;
	int z = 1;
	double start, end;
	start = omp_get_wtime();
#pragma omp parallel reduction(+:pi)
	{
#pragma omp for
		for (i = 2; i <= n; i += 2)
		{
			pi += (4*z) / (i * (i + 1.0) * (i + 2.0));
			z = -z;
		}
	}
	end = omp_get_wtime();
	printf("Значение pi для первых %d слагаемых, используя ряд
		Нилаканта:%.10f\n",n, pi);
	printf("Время: %.6f \n", end - start);
	}
```

Таблица 5

Вычисление числа Пи с помощью ряда Нилаканта в C++

Число слагаемых	Полученное значение Free Pascal	Полученное значение C++	Δ C++	ε, %
$n = 10$	3,2323158094	3,1427128427	0,0011201891	0,0357
$n = 100$	3,1514934011	3,1415907698	0,0000018838	0,0001
$n = 1000$	3,1425916543	3,1415926516	0,0000000020	0,0000
$n = 10\ 000$	3,1416926436	**3,1415926536**	0,0000000000	0,0000
$n = 100\ 000$	3,1416026535	**3,1415926536**	0,0000000000	0,0000
$n = 1\ 000\ 000$	3,1415946536	**3,1415926536**	0,0000000000	0,0000

Как и в предыдущем случае, реализация данного алгоритма более эффективна в среде Visual Studio с помощью языка C++ (табл. 6).

Таблица 6

Сравнение времени вычисления числа Пи с помощью ряда Нилаканта

Число слагаемых	Полученное значение C++	t_{C++}	Полученное значение C++ OpenMp	$t_{C++\ OpenMp}$
$n = 10$	3,1427128427	0,000000	3,1427128427	0,002104
$n = 100$	3,1415907698	0,000001	3,1415907698	0,000022
$n = 1000$	3,1415926516	0,000010	3,1415926516	0,000016
$n = 10\ 000$	**3,1415926536**	***0,000100***	**3,1415926536**	***0,000057***
$n = 100\ 000$	**3,1415926536**	**0,001007**	**3,1415926536**	**0,000617**
$n = 1\ 000\ 000$	**3,1415926536**	**0,011177**	**3,1415926536**	**0,006506**

Относительно времени вычисления ряд Нилаканта – достаточно быстрый. Затрачивается 0,000057 с, но при работе с 10 000 слагаемыми. При этом в ряде Мадхавы суммируется только 100 слагаемых и на эти вычисления тратится 0,000021 с. При вычислении по формуле Нилаканта технологии параллельного программирования позволяют сэкономить время при $n \geq 10\ 000$ слагаемых.

4. Рассмотрим результаты вычислений, полученные с помощью формулы Эйлера (табл. 7).

Фрагмент листинга кода, реализующий вычисление числа Пи через ряд Эйлера

```
void task_4(int n)
{
	double pi = 0;
	double start, end;
	start = omp_get_wtime();
#pragma omp parallel reduction(+:pi)
	{
#pragma omp for
		for (int i = 0; i < n; i++)
		{
			pi = pi + 1 / pow(i + 1, 2);
		}
	}
	pi = sqrt(6 * pi);
	end = omp_get_wtime();
	printf("Значение pi для первых %d слагаемых по формуле Эйлера:
%.10f \n", n, pi);
	printf("Время: %.6f \n", end - start);
}
```

Таблица 7

Вычисление числа Пи с помощью формулы Эйлера в C++

Число слагаемых	Полученное значение Free Pascal	Полученное значение C++	Δ C++	ε, %
$n = 10$	3,2323158094	3,0493616360	0,0922310176	2,9358
$n = 100$	3,1514934011	3,1320765318	0,0095161218	0,3029
$n = 1000$	3,1425916543	3,1406380562	0,0009545974	0,0304
$n = 10\ 000$	3,1416926436	3,1414971639	0,0000954897	0,0030
$n = 100\ 000$	3,1416026535	3,1415831043	0,0000095493	0,0003
$n = 1\ 000\ 000$	3,1415946536	3,1415916987	0,0000009549	0,0000

Используя формулу Эйлера, даже при больших n не удалось вычислить число Пи с точностью 10^{-10}. Так как требуемая точность не достигнута, проводить детальный анализ времени вычисления бессмысленно. Но в целом видна следующая тенденция: время при использовании технологии OpenMP меньше, чем в расчетах обычным способом.

5. Рассмотрим результаты вычислений, полученные с помощью формулы Валлиса (табл. 8).

Фрагмент листинга кода, реализующий вычисление числа Пи через формулу Валлиса

```
void task_5(int n)
{
	double pi = 1;
	double start, end;
	start = omp_get_wtime();
#pragma omp parallel reduction(*:pi)
	{
#pragma omp for
		for (int i = 1; i <= n; i++)
		{
			pi = pi*pow(2 * i, 2) / ((2.0*i - 1)*(2.0*i + 1));
		}
	}
	pi*= 2;
	end = omp_get_wtime();
	printf("Значение pi для первых %d слагаемых по формуле Валлиса:
%.10f \n", n, pi);
	printf("Время: %.6f \n", end - start);
}
```

Таблица 8

Вычисление числа Пи с помощью формулы Валлиса в C++

Число слагаемых	Полученное значение Free Pascal	Полученное значение C++	Δ C++	ε, %
n = 10	3,2323158094	3,0677038066	0,0738888470	2,3520
n = 100	3,1514934011	3,1337874906	0,0078051630	0,2484
n = 1000	3,1425916543	3,1408077460	0,0007849076	0,0250
n = 10 000	3,1416926436	3,1415141187	0,0000785349	0,0025
n = 100 000	3,1416026535	3,1415847997	0,0000078539	0,0002
n = 1 000 000	3,1415946536	3,1415918682	0,0000007854	0,0000

Как и при использовании формулы Эйлера, даже при больших n не удалось вычислить число Пи с точностью 10^{-10}. Как видно, явного преимущества технологии параллельного программирования в скорости вычисления методом Валлиса не дают. Это может быть связано с тем, что считается не сумма ряда, а произведение множителей.

Обсуждение

В ходе исследования были проведены практические вычисления числа Пи разными методами: Грегори–Лейбница, Мадхавы, Нилаканта, Эйлера, Валлиса – с использованием методов классического программирования и с применением технологии параллельного программирования OpenMP.

Анализ инструментов вычисления показывает, что среда Visual Studio, и в частности язык C++, дают одинаковую точность вычисления, независимо от использования методов параллельного программирования или проведения расчетов только классическими методами.

Требуемая точность 10^{-10} была достигнута только при использовании методов Мадхавы (n = 100) и Нилаканта (n = 10 000). Остальные способы даже при количестве слагаемых $n = 10^6$ не удовлетворяют заявленной точности.

Результаты вычислений при использовании технологии параллельного программирования и, только стандартных методов оказались абсолютно одинаковыми. А время вычисления в среднем при технологии OpenMP значительно меньше, особенно это заметно при увеличении числа слагаемых. При $n \geq 100$ метод Нилаканта и технология OpenMP дают самые быстрые расчеты (рис. 1).

Минимальное время вычисления числа Пи (t = 0,000021 с) получено по формуле Мадхавы при n = 100 и использовании технологии параллельного программирования OpenMP.

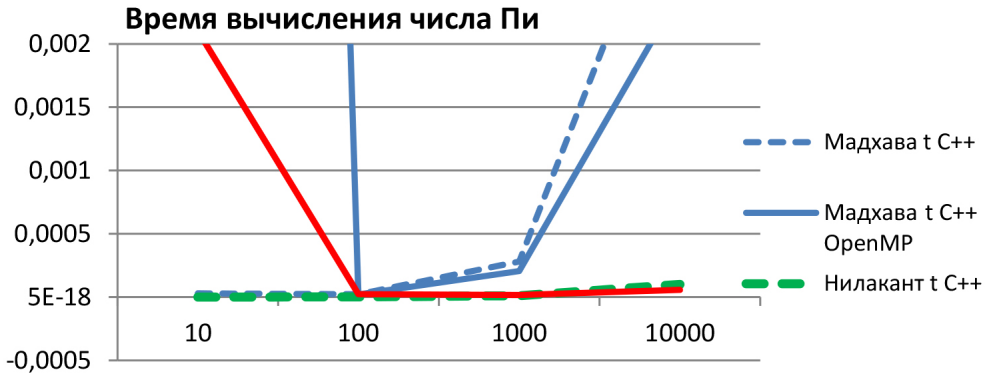

Рис. 1. Время вычисления числа Пи

Заключение

В работе продемонстрировано использование технологии параллельного программирования OpenMP в вычислительной среде Visual Studio с помощью языка C++ при вычислении математической константы Пи с десятью знаками в дробной части. Показаны преимущества данной технологии: высокая точность вычислений по сравнению с другими инструментами, синхронизация полученных результатов в расчетах с использованием технологии параллельных вычислений и классических методов, увеличение скорости расчетов за счет экономии времени при распараллеливании. Все эти преимущества свидетельствуют о необходимости внедрения и использования технологий параллельных вычислений при обработке больших объемов данных и решении задач вычислительного характера.

Список литературы

1. Зудин В. П. Развитие креативного мышления обучающихся с помощью нестандартных методов вычисления числа Пи // Информатика и образование. 2017. № 10. С. 26–37.
2. Коростелев С. П. Существенная коррекция значения числа Пи на основании абсолютно точных решений задач квадратуры круга и удвоения куба // Вестник науки и образования. 2019. № 15 (69). С. 6–16.
3. Коростелев С. П. Существенная коррекция значения числа Пи на основании абсолютно точных решений задач квадратуры круга и удвоения куба с прибавлением математического обоснования необходимости в такой коррекции // Вестник науки и образования. 2019. № 16. С. 5–21.
4. Чепасов В. И., Токарева М. А., Буреш О. В. Вычисление числа Пи методом касательных в длинной арифметике. Оренбург : Изд-во ОГУ, 2011. 119 с.
5. Сенашов В. С., Савостьянова И. Л. Вычисление числа π с помощью геометрической вероятности // Актуальные проблемы авиации и космонавтики. 2018. Т. 2, № 4 (14). С. 362–364.
6. Луценко Е. В. Исследование символьных и цифровых рядов методами теории информации и АСК-анализа (на примере числа Пи с одним миллионом знаков после запятой) // Политематический сетевой электронный научный журнал Кубанского государственного аграрного университета. 2014. № 99. С. 73–100.
7. Андросенко В. А. Мера иррациональности числа $\frac{\pi}{\sqrt{3}}$ // Известия Российской академии наук. Серия математическая. 2015. Т. 79, № 1. С. 3–20.
8. Selenskih V. N. Physical method of determining the exact Pi number // Eastern European Scientific Journal. 2013. № 6. P. 100–104.
9. Lewis H. Calculating Pi (π) // The Institute of Mathematics and its Applications. URL: https://www.mathscareers.org.uk/calculating-pi/
10. Sparavigna A. C. Number Pi from the decorations of ancient artifacts // Archaeoastronomy and Ancient Technologies. 2013. Vol. 1, № 2. C. 40–47.
11. Arndt J., Haenel C. Pi. Algorithmen, Computer, Arithmetik. Springer, 2000, 264 p.
12. Bailey D., Borwein J. The next generation a sourcebook on the recent history of Pi and its computation. Springer, 2016. 507 p.
13. Agarwal R. P., Agarwal H., Sen S. K. Birth, growth and computation of Pi to ten trillion digits // Advances in Difference Equations. 2013. № 1. P. 100. URL: https://advancesindifferenceequations.springeropen.com/track/pdf/10.1186/1687-1847-2013-100.pdf
14. Графова А. А. Число Пи: краткий обзор свойств и нерешенные проблемы // Проблемы современных интеграционных процессов и поиск инновационных решений : сб. ст. Междунар. науч.-практ. конф. (Стерлитамак, 2020). Уфа : Агентство международных исследований, 2020. С. 15–18.
15. Кормилицына Т. В. Обучение программированию в языках сверхвысокого уровня на примере входных языков систем компьютерной математики // Учебный эксперимент в образовании. 2017. № 1 (81). С. 41–45.
16. Бакаева О. А. Анализ процессов компьютерного моделирования вычисления числа Пи методом Монте-Карло // Вестник Чувашского университета. 2018. № 3. С. 151–162.
17. Окань С. В. Изучение влияния распараллеливания вычислений с использованием OPENMP // Совершенствование методологии и организации научных исследований в целях развития общества : сб. ст. по итогам Междунар. науч.-практ. конф. Стерлитамак : Агентство международных исследований, 2020. С. 173–176.
18. Жалнин Р. В., Панюшкина Е. Н., Пескова Е. Е., Шаманаев П. А. Основы параллельного программирования с использованием технологий MPI и OpenMP : учеб. пособие. Саранск : Изд-во СВМО, 2013. 76 с.

19. Как вычислить значение Пи. URL: https://ru.wikihow.com/ %D0 %B2 %D1 %8B %D1 %87 %D0 %B8 %D1 %81 %D0 %BB %D0 %B8 %D1 %82 %D1 %8C- %D0 %B7 %D0 %BD %D0 %B0 %D1 %87 %D0 %B5 %D0 %BD %D0 %B8 %D0 %B5- %D0 %9F %D0 %B8
20. Бакаева О. А. Сравнительный анализ методов вычисления числа Пи стандартными средствами // Программные продукты и системы. 2018. № 2. С. 409–413.

References

1. Zudin V.P. Development of students' creative thinking with the help of non-standard methods for calculating the number Pi. *Informatika i obrazovanie = Informatics and Education*. 2017;(10):26–37. (In Russ.)
2. Korostelev S.P. Substantial correction of the value of pi on the basis of absolutely exact solutions to the problems of squaring the circle and doubling the cube. *Vestnik nauki i obrazovaniya = Bulletin of Science and Education*. 2019;(15):6–16. (In Russ.)
3. Korostelev S.P. Substantial correction of the value of pi on the basis of absolutely exact solutions of the problems of squaring the circle and doubling the cube with the addition of a mathematical justification for the need for such a correction. *Vestnik nauki i obrazovaniya = Bulletin of Science and Education*. 2019;(16):5–21. (In Russ.)
4. Chepasov V.I., Tokareva M.A., Buresh O.V. *Vychislenie chisla Pi metodom kasatel'nykh v dlinnoy arifmetike = Calculation of the number Pi by the tangent method in long arithmetic*. Orenburg: Izd-vo OGU, 2011:119. (In Russ.)
5. Senashov V.S., Savost'yanova I.L. Calculation of the number π using geometric probability. *Aktual'nye problemy aviatsii i kosmonavtiki = Actual problems of aviation and cosmonautics*. 2018;2(4):362–364. (In Russ.)
6. Lutsenko E.V. Research of symbolic and digital series by methods of information theory and ASK-analysis (on the example of the number Pi with one million decimal places). *Politematicheskiy setevoy elektronnyy nauchnyy zhurnal Kubanskogo gosudarstvennogo agrarnogo universiteta = Polythematic network electronic scientific journal of the Kuban State Agrarian University*. 2014;(99):73–100. (In Russ.)
7. Androsenko V.A. The measure of irrationality of the number $\frac{\pi}{\sqrt{3}}$. *Izvestiya Rossiyskoy akademii nauk. Seriya matematicheskaya = Bulletin of the Russian Academy of Sciences. Mathematical series*. 2015;79(1):3–20. (In Russ.)
8. Selenskih V.N. Physical method of determining the exact Pi number. *Eastern European Scientific Journal*. 2013;(6):100–104.
9. Lewis H. Calculating Pi (π). *The Institute of Mathematics and its Applications*. Available at: https://www.mathscareers.org.uk/calculating-pi/
10. Sparavigna A.C. Number Pi from the decorations of ancient artifacts. *Archaeoastronomy and Ancient Technologies*. 2013;1(2):40–47.
11. Arndt J., Haenel C. Pi. *Algorithmen, Computer, Arithmetik*. Springer, 2000:264.
12. Bailey D., Borwein J. *The next generation a sourcebook on the recent history of Pi and its computation*. Springer, 2016:507.
13. Agarwal R.P., Agarwal H., Sen S.K. Birth, growth and computation of Pi to ten trillion digits. *Advances in Difference Equations*. 2013;(1):100. Available at: https://advancesindifferenceequations.springeropen.com/track/pdf/10.1186/1687-1847- 2013-100.pdf
14. Grafova A.A. Number Pi: a brief overview of properties and unsolved problems. *Problemy sovremennykh integratsionnykh protsessov i poisk innovatsionnykh resheniy: sb. st. Mezhdunar. nauch.-prakt. konf. (Sterlitamak, 2020) = Problems of modern integration processes and the search for innovative solutions: collection of articles. Art. Int. scientific-practical conf. (Sterlitamak, 2020).* Ufa: Agentstvo mezhdunarodnykh issledovaniy, 2020:15–18. (In Russ.)

15. Kormilitsyna T.V. Teaching programming in ultrahigh-level languages on the example of input languages of computer mathematics systems. *Uchebnyy eksperiment v obrazovanii = Educational experiment in education*. 2017;(1):41–45. (In Russ.)
16. Bakaeva O.A. Analysis of the processes of computer modeling of calculating the number Pi by the Monte Carlo method. *Vestnik Chuvashskogo universiteta = Bulletin of the Chuvash University*. 2018;(3):151–162. (In Russ.)
17. Okan' S.V. Studying the influence of parallel computing with the use of OPENMP. *Sovershenstvovanie metodologii i organizatsii nauchnykh issledovaniy v tselyakh razvitiya obshchestva: sb. st. po itogam Mezhdunar. nauch.-prakt. konf. = Improving the methodology and organization of scientific research for the development of society: collection of articles. Art. following the results of Intern. scientific-practical conf.* Sterlitamak: Agentstvo mezhdunarodnykh issledovaniy, 2020:173–176. (In Russ.)
18. Zhalnin R.V., Panyushkina E.N., Peskova E.E., Shamanaev P.A. *Osnovy parallel'nogo programmirovaniya s ispol'zovaniem tekhnologiy MPI i OpenMP: ucheb. posobie = Basics of parallel programming using MPI and OpenMP technologies: textbook.* Saransk: Izd-vo SVMO, 2013:76. (In Russ.)
19. *Kak vychislit' znachenie Pi = How to calculate the Pi value.* (In Russ.). Available at: https://ru.wikihow.com/ %D0 %B2 %D1 %8B %D1 %87 %D0 %B8 %D1 %81 %D0 %BB %D0 %B8 %D1 %82 %D1 %8C- %D0 %B7 %D0 %BD %D0 %B0 %D1 %87 %D0 %B5 %D0 %BD %D0 %B8 %D0 %B5- %D0 %9F %D0 %B8
20. Bakaeva O.A. Comparative analysis of methods for calculating the number Pi by standard means. *Programmnye produkty i sistemy = Software products and systems*. 2018;(2):409–413. (In Russ.)

Информация об авторах / Information about the authors

Ольга Александровна Бакаева
кандидат технических наук, доцент,
доцент кафедры систем
автоматизированного проектирования,
Национальный исследовательский
Мордовский государственный
университет имени Н. П. Огарева
(Россия, Республика Мордовия,
г. Саранск, ул. Большевистская, 68)
E-mail: helga_rm@rambler.ru

Olga A. Bakaeva
Candidate of technical sciences,
associate professor,
associate professor of the sub-department
of computer-aided design,
National Research Ogarev Mordovia
State University
(68 Bolshevistskaya street, Saransk,
Republic of Mordovia, Russia)