**Отчет по лабораторной 1: Логаш Полина Александровна 3 курс, 2 группа**

**Постановка задачи:** Рассчитать время, необходимое для вычисление корней квадратного уравнения в одном или нескольких потоках

**Конечный результат:**

Время выполнения для одного потока - 932ms

Для 4 - 737ms

Для 8 - 497ms

**Коэффициенты эффективности:**

p = T1 / Tp   -  коэффициент ускорения

4 = T1 / T4 = 932 / 737 = 1.26

8 = T1 / T4 = 932 / 497 = 1.88

p = T1 /(p\*Tp)   -  коэффициент эффективности

4 = T1 /(p\*Tp)   = 932 / (737 \* 4) = 0.316

8 = T1 /(p\*Tp)  = 932 / (497 \* 8) = 0.234

**Отчет по лабораторной 2: Логаш Полина Александровна 3 курс, 2 группа**

**Постановка задачи:** рассчитать оптимальное число “входов” на стадион для заполнения стадиона 50\_000 людей, при пропускной способности прохода 100 человек за tn = 60 + 10n секунд, где n - номер входа, начиная с нуля

**Конечный результат:**

Для расчета оптимального числа входов на стадион, мы можем использовать следующий алгоритм:

Рассчитаем время, которое потребуется на заполнение стадиона при использовании одного входа. Для этого мы будем использовать формулу:

время = количество людей / (пропускная способность входа \* время прохода)

* количество людей - это 50 000
* пропускная способность входа - 100 человек за tn = 60 + 10n секунд
* время прохода - это время, которое займет человеку, чтобы пройти через вход.

Первый вход (n = 0) займет 60 секунд (tn = 60 + 0),

второй вход (n = 1) займет 70 секунд (tn = 60 + 10),

третий вход (n = 2) займет 80 секунд (tn = 60 + 20), и т.д.

**Конечный результат: 9 входов**

**Отчет по лабораторной 3: Логаш Полина Александровна 3 курс, 2 группа**

**Постановка задачи:** программа вычисляет интеграл ∫ sin x² dx используя формулу трапеций и формулу разложения в ряд

**Конечный результат:**

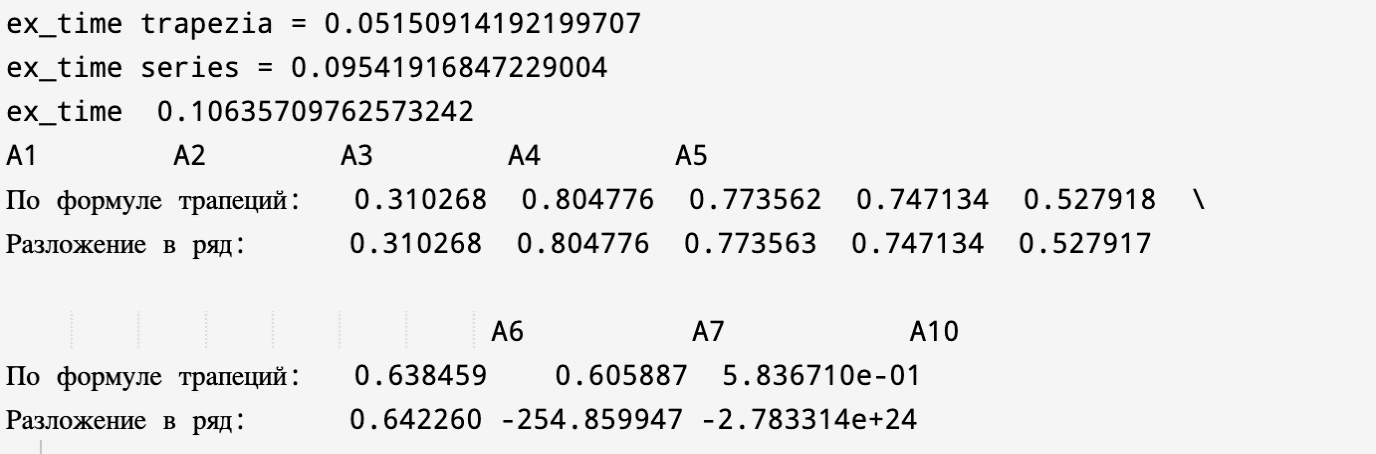


Таблица сравнения точности вычисления через вольфрам:

n |       Формула трапеций | Разложение в ряд

1 | 0.4644577527786463 | 0.4638961834499646

2 | 0.4644603712804756 | 0.4644631487079905

3 | 0.4644603688093074 | 0.4644603692353347

4 | 0.4634603688089657 | 0.4644603688090005

5 | 0.4644688097786463 | 0.4638961834499626

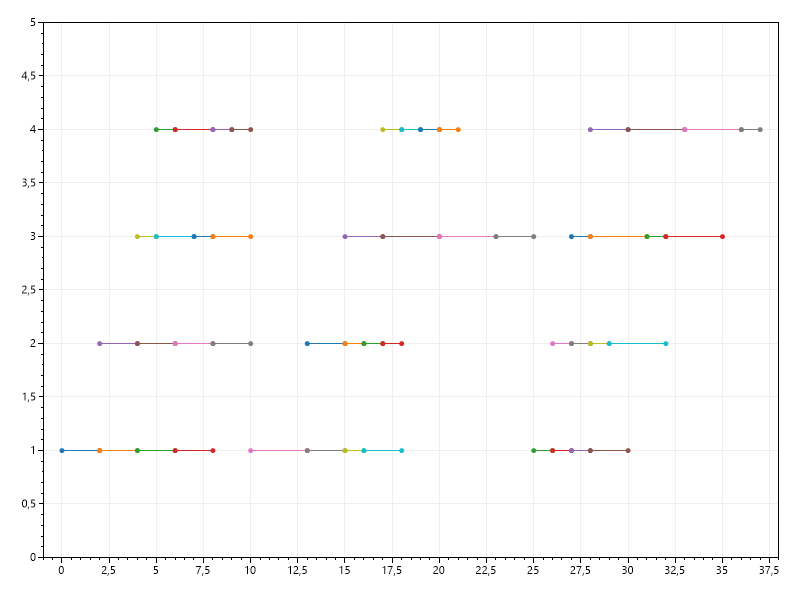
6 | 0.4644603718688056 | 0.4644631487079905

7 | 0.4645603688446074 | 0.4644603694870799

10| 0.464460371289657 | 0.4644603663148005

**Отчет по лабораторной 4: Логаш Полина Александровна 3 курс, 2 группа**

Программа находит величину выполнения заданного процесса в многопоточном режиме и строит соответствующую диаграмму.



На графике представлено время выполнения процессов (нижняя ось, в миллисекундах) и потоки, на которых эти процессы выполняются (ось слева, номер потока от 1 до 4).

**Program.cs**

const int matrixHeight = 4;

const int matrixWidth = 12;

const int matrixMinValue = 1;

const int matrixMaxValue = 4;

Solve();

static void Solve()

{

int[] processesIndexes = {1, 2, 3, 4};

var plot = new Plot();

var matrix = new int[matrixWidth, matrixHeight];

CreateMatrix(matrixWidth, matrixHeight, matrix);

Draw(matrixWidth, matrixHeight, plot, processesIndexes, matrix);

plot.SetAxisLimitsY(0, processesIndexes.Length + 1);

plot.SaveFig("../../../../Diagrams/FirstSynchronousMethod.png");

}

static void Draw(int width, int height, Plot plot, int[] processesIndexes, int[,] matrix)

{

var count = 0;

var begin = 0;

var newBegin = 0;

var max = 0;

for (var i = 0; i < width; i++)

{

if (count % processesIndexes.Length == 0)

{

count = 0;

newBegin = begin;

}

if (count == 0)

{

newBegin = max;

}

else

{

newBegin += matrix[i - 1, 0];

}

begin = newBegin;

for (var j = 0; j < height; j++)

{

begin = AddSegment(begin, begin + matrix[i, j], plot, processesIndexes[count], begin, matrix[i, j]);

if (max < begin)

{

max = begin;

}

}

count++;

}

}

static int AddSegment(int a, int b, Plot plot, double process, int begin, int c)

{

var x = new double[] {a, b};

var y = new[] {process, process};

plot.AddScatter(x, y);

begin += c;

return begin;

}

static void CreateMatrix(int width, int height, int[,] matrix)

{

var random = new Random();

for (var i = 0; i < height; i++)

{

for (var j = 0; j < width; j++)

{

matrix[j, i] = random.Next(matrixMinValue, matrixMaxValue); // рандомим по очереди каждый элемент матрицы

}

}

}

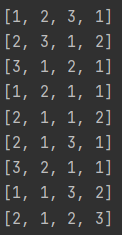
**Отчет по лабораторной 5: Логаш Полина Александровна 3 курс, 2 группа**

Программа находит критический путь выполнения процессов в матрице, используя формулы из учебника.

**Алгоритм:**

1. Задать p, n, s
2. Подставить их значение в формулы
3. Рассчитать критическое время

Для следующий матрицы



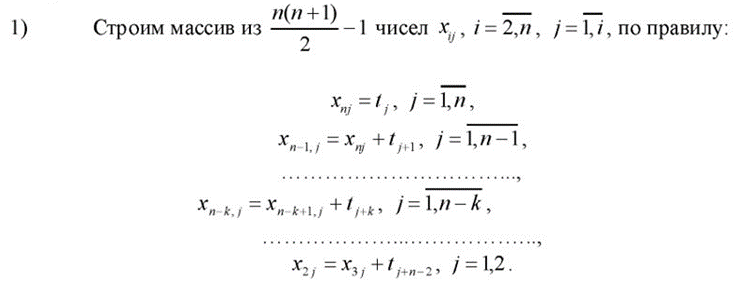
состоящей из n = 9 строк и s = 4 столбцов, разделенную на p = 3 матрицы, критическое время составило 27.

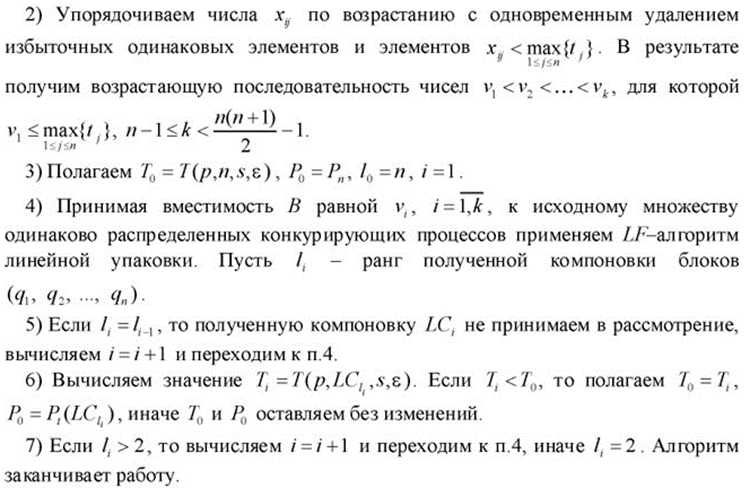
Для решения задачи была создана программа на языке C#, которая находит критическое время для матрицы процессов.

**Отчет по лабораторной 6: Логаш Полина Александровна 3 курс, 2 группа**

Программа выполняет построение оптимальной системы одинаковых процессов, улучшает время выполнения за счет оптимальной компоновки для системы одинаково распределенных процессов.

**Алгоритм:**

****

****

Для Р₉ = (4, 1, 9, 2, 4, 4, 1, 4, 2), i = [1, 9], p = 3, s = 4, ε = 1 неоптимальное время выполнения составило **121**. После выполнения оптимизации время уменьшилось до **108**, то есть прирост в производительности составил **12,04%**.

Программа, выполненная на языке C#, оптимизирует построение оптимальной системы одинаковых процессов.

**Отчет по лабораторной 7: Логаш Полина Александровна 3 курс, 2 группа**

**Постановка задачи:** программа вычисляет интеграл ∫ sin x² dx используя формулу трапеций и формулу разложения в ряд

**Конечный результат:**

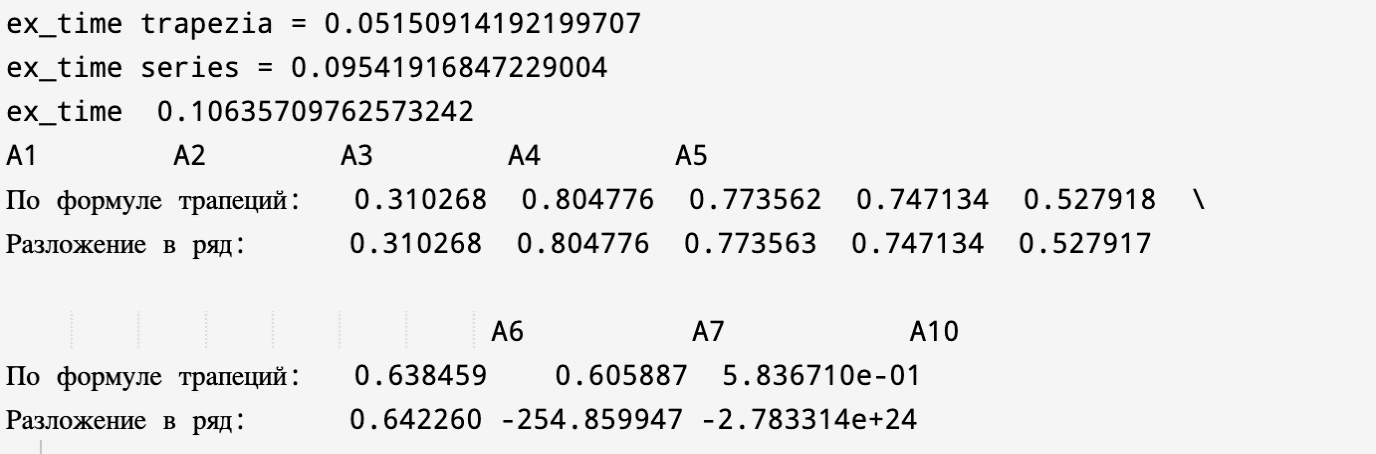


Таблица сравнения точности вычисления через вольфрам:

n |       Формула трапеций | Разложение в ряд

1 | 0.4644577527786463 | 0.4638961834499646

2 | 0.4644603712804756 | 0.4644631487079905

3 | 0.4644603688093074 | 0.4644603692353347

4 | 0.4634603688089657 | 0.4644603688090005

5 | 0.4644688097786463 | 0.4638961834499626

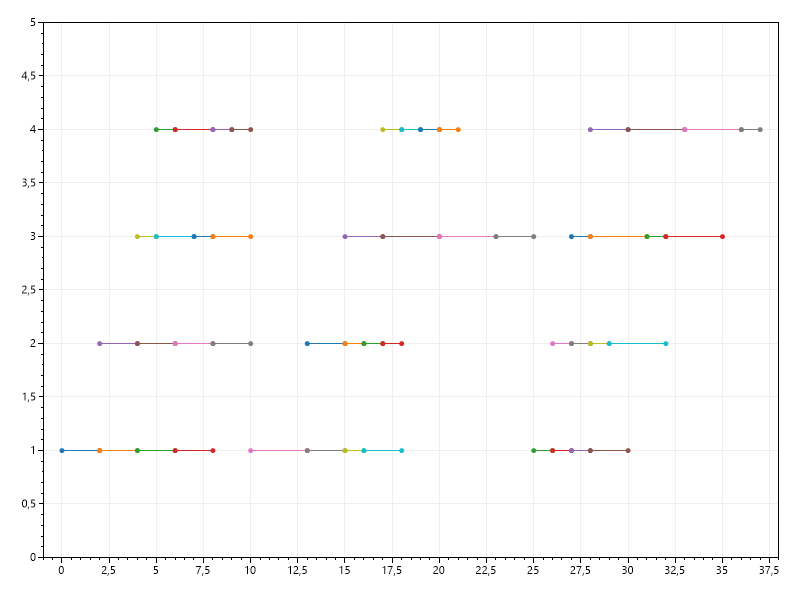
6 | 0.4644603718688056 | 0.4644631487079905

7 | 0.4645603688446074 | 0.4644603694870799

10| 0.464460371289657 | 0.4644603663148005

**Отчет по лабораторной 8: Логаш Полина Александровна 3 курс, 2 группа**

Программа находит величину выполнения заданного процесса в многопоточном режиме и строит соответствующую диаграмму.



На графике представлено время выполнения процессов (нижняя ось, в миллисекундах) и потоки, на которых эти процессы выполняются (ось слева, номер потока от 1 до 4).

**Program.cs**

const int matrixHeight = 4;

const int matrixWidth = 12;

const int matrixMinValue = 1;

const int matrixMaxValue = 4;

Solve();

static void Solve()

{

int[] processesIndexes = {1, 2, 3, 4};

var plot = new Plot();

var matrix = new int[matrixWidth, matrixHeight];

CreateMatrix(matrixWidth, matrixHeight, matrix);

Draw(matrixWidth, matrixHeight, plot, processesIndexes, matrix);

plot.SetAxisLimitsY(0, processesIndexes.Length + 1);

plot.SaveFig("../../../../Diagrams/FirstSynchronousMethod.png");

}