Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Самарский национальный исследовательский университет
имени академика С.П. Королева»

Институт информатики и кибернетики

Кафедра технической кибернетики

Отчет по лабораторной работе № 1
Вычисления “GAXPY”»

Курс: «Параллельные алгоритмы»

Студент: Грязнов Илья Евгеньевич
(фамилия, имя отчество)
Группы 6131-010402D

Самара 2022

Содержание

Задание

В целях ознакомления с операцией gaxpy самостоятельно реализовать метод вычисления данной операции на любом языке программирования. Записать входные и выходные данные, произвести аналогичные вычисления с теми же данными при помощи уже реализованной библиотеки BLAS.

Теоретическая часть

Умножение матриц является одной из базовых операций вычислительной линейной алгебры. К ней неизменно сводятся блочные алгоритмы матричных разложений, которые в свою очередь широко применяются при решении систем линейных алгебраических уравнений и алгебраической проблемы собственных значений – двух основных задач упомянутой предметной области.

Операцию gaxpy (general A x plus y.) принято записывать в виде

$$z=Ax+y$$

Через нее удобно выражать умножение матриц, матричные разложения и итерационные методы решения СЛАУ.

Положим, что вектора и матрица из gaxpy для удобства составления параллельного алгоритма разбиты на блоки: $z_i, y_i \in \mathbb{R}^{\alpha \times 1}$, где $1 \leq i, j \leq p$, $\alpha = n/p$, $\beta = m/p$, а p – количество задач искомого алгоритма. Построение параллельного алгоритма начинается с распределения блоков матрицы A между задачами; Выбранный способ распределения служит для дальнейшей классификации алгоритмов.

Ход выполнения работы

Первым делом был реализован код программы на C#, отвечающий за создание квадратной матрицы заданного размера и двух векторов, длинной соответственно подходящей для выполнения операции gaxpy.

Далее был реализована часть выполняющая непосредственно вычислительные операции для данных объектов с числовыми значениями типа double (число с плавающей точкой). Сама реализация программы представлена ниже в приложении (после списка используемой литературы).

Так как в языке C# хоть и присутствует библиотека blas, переписанная с языка Fortran. Все же интересующий нас метод (gaxpy) не очень хорошо там реализован. Поэтому вторую часть работы (вычисления средствами стандартной библиотеки blas и сравнения полученных результатов) будем осуществлять при помощи GNU Octave (GUI). Для этого полученные в консольном приложении .NET данные записываем в txt фаил и передаем в Octave (Код реализуемый в Octave так же представлен в приложении ниже). Где мы открываем саму программу считываем данные и производим вычисления, результаты функции norm(A) представлены ниже.

Для наглядности, взяв небольшую матрицу 10x10, выведем результат первой итерации:

Матица 10x10:

Выходные данные после операции gaxpy в Octave:

```
>> outOctave
outOctave =

   3.0592
   2.0310
   2.8681
   1.8403
   2.6136
   3.5180
   3.0708
   3.1788
   2.2106
   3.3219
```

Выходные данные после операции gaxpy в .NET:

```
>> output
output =

   3.0592
   2.0310
   2.8681
   1.8403
   2.6136
   3.5180
   3.0708
   3.1788
   2.2106
   3.3219
```

Результат нормировки:

```
>> result = norm(output - outOctave);
>> result
result = 8.8846e-15
```

Далее не будет выводить полностью вектор, ввиду своего размера, а будут представлены только результаты нормировки.

Матица 100x100:

```
>> result = norm(output - outOctave);
>> result
result = 3.3653e-13
>>
```

Матица 500x500:

```
>> result = norm(output - outOctave);
>> result
result = 6.8527e-12
>>
```

Матица 700x700:

```
>> result
result = 8.5511e-12
>>
```

Размерность матрицы	Погрешность вычислений
Матрица 10x10	8.8846e-15
Матица 100x100	3.3653e-13
Матица 500x500	6.8527e-12
Матица 700x700	8.5511e-12

Таблица 1. Результаты работы операции gaxpy

Вывод

В рамках первой лабораторной работы по изучению параллельных алгоритмов была реализована операция gaxpy на языке C# в консольном приложении .NET с дальнейшим ее применением и сравнением выходных результатов с работой стандартной библиотеки blas в приложении GNU Octave (GUI). Результаты оказались вполне приемлемы, метод нормировки выходных данных показал, что разница в вычислениях между вышеуказанными методами ничтожно мала. И с уменьшением размера входных данных стремиться к нулю.

Список использованных источников

1. Головашкин Д.Л. Векторные алгоритмы вычислительной линейной алгебры: учебной пособие – Самара: Изд-во Самарского университета, 2019. – 76 с.
2. Головашкин Д.Л. Параллельные алгоритмы вычислительной линейной алгебры: учебной пособие – Самара: Изд-во Самарского университета, 2019. – 88 с.
3. Головашкин Д.Л. Модели в теории параллельных вычислений: учебной пособие – Самара: Изд-во Самарского университета, 2019. – 96 с.
4. Голуб Дж., Ван Лоун Ч. Матричные вычисления. Пер. с англ. / Под ред. Воеводина В.В. –М.: Мир, 1999. - 548 с.

Листинг программы

Часть, выполненная на C#

```
using System;
using System.IO;


namespace gaxpy
{
    internal class Program
    {
        static void Main(string[] args)
        {
            //Формула gaxpy z = A*x + y
            //Заем размерность матрицы
            Random rnd = new Random();
            int n = 10;
            int m = 10;
            string text = "";

            Console.WriteLine($"Матрица размерностью: n={n} и m={m} \n");
            //Созаем матрицу с заданной размерностью
            double[,] matrix = new double[n, m];


            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    matrix[i, j] = rnd.NextDouble();
                    Console.Write("{0} ", matrix[i, j]);
                    text = text + matrix[i, j] + "\n";
                }
                Console.WriteLine();
                //text = text + "\n";
            }
            write("matrix", text.Replace(",", "."));
            Console.WriteLine("---------");


            // Вектор
            text = "";
            Console.WriteLine($"Вектор x: \n");
            double[] vector = new double[m];
            for (int i = 0; i < m; i++)
            {
                vector[i] = rnd.NextDouble();
                Console.WriteLine(vector[i]);
                text = text + vector[i] + "\n";
            }
            write("vector", text.Replace(",", "."));
            Console.WriteLine("---------");


            // второй вектор
            text = "";
            Console.WriteLine($"Вектор y: \n");
            double[] vectorNext = new double[n];
            for (int i = 0; i < n; i++)
            {
                vectorNext[i] = rnd.NextDouble();
                Console.WriteLine(vectorNext[i]);
```

```
                text = text = text + vectorNext[i] + "\n";
            }
            write("vectorNext", text.Replace(",", "."));
            Console.WriteLine("---------");



            //Выходная матрица (уже вектор)
            double[] outputMatrixV = new double[n];

            for (int i = 0; i < n; i++)
            {
                for (int j = 0; j < m; j++)
                {
                    outputMatrixV[i] += matrix[i, j] * vector[j];
                }
            }

            //итоговый вектор
            text = "";
            double[] output = new double[n];

            for (int i = 0; i < n; i++)
            {
                output[i] = outputMatrixV[i] + vectorNext[i];
            }




            Console.WriteLine($"Выходной итоговый вектор после gaxpy: \n");
            for (int i = 0; i < n; i++)
            {
                Console.WriteLine($"{output[i]}");
                text = text + output[i] + "\n";
            }
            write("output", text.Replace(",", "."));

            Console.ReadKey();
        }

        private static void write(string nameFile, string text)
        {
            string path = $@"C:\Users\Dark_Monk\Desktop\gaxpy\{nameFile}.txt";
            // полная перезапись файла
            using (StreamWriter writer = new StreamWriter(path, false))
            {
                writer.WriteLine(text);
            }
        }
    }
}
```

Часть, выполненная в Octave

```
clear all; close all;
load matrix.txt;

n = 500;
A = zeros(n, n);

for j = 1:n
  b = (j-1) * n + 1:j * n;
  A(j,:) = matrix(b);
end

load vector.txt;
load vectorNext.txt;
load output.txt;

outOctave = A * vector + vectorNext;

outOctave
output

result = norm(output - outOctave);
result
```