Лабораторная работа 2 "Линейная алгебра".

Задание 3 (решение СЛАУ методом Гаусса).

Задание выполнил: Лисунов Никита, группа 20ПИ-2

Выполнение лабораторной работы происходило в команде.

Участники команды:

- Корнев Егор, группа 20ПИ-1
- Самсонова Карина, группа 20ПИ-1
- Лисунов Никита, группа 20ПИ-2

Отчет.

Формат ввода/вывода.

На ввод прорамме ничего не подается. Вывод программы содержит 4 строки: время работы последовательной программы, решение СЛАУ последовательной программы, время работы параллельной программы и решение СЛАУ параллельной программы.

Исходные данные.

Матрица чисел (исходная СЛАУ) генерируется случайным образом в зависимости от значения параметра matrixDimSize, который обозначает кол-во строк матрицы. В алгоритме используется расширенная матрица размером matrixDimSize x (matrixDimSize + 1).

Тестирование.

Тестирование производительности программы происходило с использованием сервиса Google Colab.

Алгоритм решения с использованием CUDA.

Для решения СЛАУ методом Гаусса необходимо выполнить следующие шаги:

1. Привести заданную матрицу к верхнетреугольному виду.
2. Вычислить неизвестные исходной системы.

Для первого шага используется функция transformToUpperTriangularLinearSystem. Она работает с matrixDimSize кол-вом SM и (matrixDimSize + 1) кол-вом SP. Алгоритм запускается (matrixDimSize - 1) кол-во раз для прохождения всех стадий преобразования матрицы к верхнетреугольному виду (на каждой стадии выбирается опорная строка, от которой преобразуются нижележащие строки). Во время работы каждой стадии за каждую строку матрицы отвечает отдельный блок, а за каждый элемент строки отвечает отдельный поток внутри этого блока. Каждая итерация сначала вычисляет

элементы, лежащие в верхней части матрицы, но ниже опорной строки (элементы, которые не нужно приводить к 0), а затем зануляет элементы столбца, который должен быть приведен к такому виду на данной стадии. Для синхронизации потоков внутри каждого блока (в рамках каждой стадии) используется __syncthreads() (это необходимо для того, чтобы обнуление нужного элемента не произошло до того, как все остальные потоки произведут свои расчеты с использованием этого самого элемента), а синхронизация блоков между стадиями происходит автоматически, т.к. пока не завершится предыдущий kernel, новый не начнет свою работу (это нужно для того, чтобы значения одной стадии не использовались в другой, пока не будут пересчитаны).

После того, как матрица была приведена к верхнетреугольному виду и все потоки (и блоки) синхронизировались, вызывается функция calculateLinearSystemResults. Она работает с 1 SM и с matrixDimSize кол-вом SP. На данном шаге алгоритма мы проходим по матрице снизу вверх, справа налево. Таким образом мы всегда можем прийти к неизвестному, которое можно вычислить. Здесь потоки отвечают за элементы своих строк матрицы, а столбцы перебираются последовательно в цикле. После того, как вычисляется одно из неизвестных, все остальные потоки вычисляют промежуточное значение исходя из полученного значения одной из неизвестных в данном столбце. Далее мы переходим к следующему столбцу. В каждом столбце будет 1 неизвестная, которую можно вычислить. И так мы проходим, вычисляя все результаты. Для синхронизации потоков внутри каждого блока используется __syncthreads(). Это нужно для того, чтобы потоки дожидались вычисления неизвестного значения, которое им необходимо использовать в дальнейшем.