МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Нижегородский государственный университет им. Н.И. Лобачевского»

(ННГУ)

Отчёт по лабораторной работе

Решение систем линейных уравнений методом сопряженных градиентов

Выполнил:
студент гр. 381506-3
Куликова С.А.

Проверил:
Доцент кафедры МОСТ
Сысоев. А.В.

Нижний Новгород

2018

Оглавление

Введение 3
Постановка задачи 4
Метод решения 5
Схема распараллеливания 6
Программная реализация 7
Руководство пользователя 8
Подтверждение корректности 9
Результаты экспериментов по оценке масштабируемости 10
Заключение 12

Введение

Система линейных алгебраических уравнений (линейная система, также употребляются аббревиатуры СЛАУ, СЛУ) — система уравнений, каждое уравнение в которой является линейным — алгебраическим уравнением первой степени.

В классическом варианте коэффициенты при переменных, свободные члены и неизвестные считаются вещественными числами, но все методы и результаты сохраняются (либо естественным образом обобщаются) на случай любых полей, например, комплексных чисел.

Решение систем линейных алгебраических уравнений — одна из классических задач линейной алгебры, во многом определившая её объекты и методы. Кроме того, линейные алгебраические уравнения и методы их решения играют важную роль во многих прикладных направлениях, в том числе в линейном программировании, эконометрике.

Общий вид системы линейных алгебраических уравнений:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \\ \cdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n = b_m \end{cases}$$

Метод сопряженных градиентов — численный метод решения систем линейных алгебраических уравнений, является итерационным методом Крыловского типа.

Постановка задачи

Требуется решить систему линейных алгебраических уравнений вида: $Ax = b$ методом сопряженных градиентом. Результат решения СЛАУ будет представлен в виде вектора x, при котором будут выполняться все уравнения системы.

Задача формулируется следующим образом: требуется разработать программу, которая находит вектор решение систем линейных алгебраических уравнений методом сопряженных градиентов. Для этого разработать последовательный и параллельный алгоритмы. Затем сравнить время выполнения на различном числе процессов и определить ускорение.

Программа должна использовать интерфейс OpenMP и TBB для эффективной работы на системах с распределенной памятью.

Выполнение задачи включает:

1. Освоение темы (постановка задачи).
2. Изучение метода решения.
3. Разработку схемы параллельных вычислений.
4. Реализацию программы.
5. Проведение численных экспериментов с анализом масштабируемости.
6. Подготовку отчета с анализом результатов экспериментов.

В программе необходимо:

1. Выполнить проверку совпадения результатов последовательной и параллельной реализаций.
2. Продемонстрировать корректность работы алгоритмов на задачах малой размерности.
3. Обеспечить генерацию данных (или чтение из файла) для задач произвольной размерности. Параметры задачи должен задавать пользователь.
4. Вывести время работы последовательного и параллельного алгоритмов. Ускорение параллельного алгоритма.

Метод решения

Пусть дана система линейных уравнений: $Ax = b$. Причём матрица системы — это действительная матрица, обладающая следующими свойствами: $A = A^T > 0$, то есть это симметричная положительно определённая матрица.

- Матрица называется симметричной, если она совпадает со своей транспонированной матрицей $A = A^T$
- Матрица называется положительно-определенной если $x^T Ax > 0$

После выполнения n итераций метода сопряженных градиентов(n есть порядок решаемой системы линейных уравнений), очередное приближение X_n совпадает с точным решением.

Итерация метода сопряженных градиентов состоит в вычислении очередного приближения к точному решению

$$x^k = x^{k-1} + s^k d^k$$

x^k - очередное приближение

x^{k-1} - приближение, построенное на предыдущем шаге

s^k - скалярный шаг

d^k - вектор направления

Перед выполнением первой итерации будем считать

$$x^0 = 0, d^0 = 0, g^0 = -b$$

Алгоритм:

1. Вычисление градиента

$$g^k = A \cdot x^{k-1} - b$$

2. Вычисление вектора направления

$$d^k = -g^k + \frac{((g^k)^T, g^k)}{((g^{k-1})^T, g^{k-1}} d^{k-1}$$

3. Вычисление величины смещения по заданному направлению

$$s^k = \frac{(d^k, g^k)}{(d^k)^T \cdot A \cdot d^k}$$

4. Вычисление нового приближения

$$x^k = x^{k-1} - s^k d^k$$

Схема распараллеливания

Выполнение итераций метода осуществляется последовательно, следовательно, наиболее целесообразный подход состоит в распараллеливании вычислений, реализуемых в ходе выполнения отдельных итераций. Основные вычисления, выполняемые в соответствии с методом, состоят в перемножении матрицы A на вектора x и d. Дополнительные вычисления, имеющие меньший порядок сложности представляют собой различные операции обработки векторов (скалярное произведение, сложение и вычитание, умножение на скаляр).

В последовательном алгоритме всего 2 умножения матрицы на вектор на шагах 1 и 3, именно здесь и задействуется параллельная схема. Рассмотрим параллельный вариант умножения матрицы на вектор. Распределение данных – разбиение матрицы на строки

Базовая подзадача для выполнения вычисления должна содержать строку матрицы A и копию вектора b. После завершения вычислений каждая базовая подзадача будет содержать один из элементов вектора результата c. Для объединения результатов расчетов и получения полного вектора c на каждом из процессоров вычислительной системы необходимо выполнить операцию обобщенного сбора данных.

Вычислительная сложность алгоритма $T_1 = 2n^3 + 13n^2$

Программная реализация

Программа реализована с использование библиотек omp.h и tbb.h.

1. GetMatr(…) возвращает матрицу без i-ой строи и j-ого столбца
2. Determinant(…) рекурсивно вычисляет детерминант матрицы
3. dimension(…) проверяет размерность системы, она должна быть положительной
4. symmetry(…) проверяет симметричность матрицы
5. PositiveDefinite(…) проверяет положительно-определенность матрицы
6. main(…) читает из бинарного файла СЛАУ, отправляет её решать в один из выбранных пользователем способов, и записывает результат в другой бинарный файл.
7. NonlinearConjugateGradient(…) решает СЛАУ последовательно.
8. NonlinearConjugateGradient_OMP(…) решает СЛАУ параллельно с использованием технологии OpenMP.
9. NonlinearConjugateGradient_TBB(…) решает СЛАУ параллельно с использованием технологии TBB.
10. vec(…) вычисляет скалярное произведение векторов в последовательной версии.
11. vec_OMP(…) вычисляет скалярное произведение векторов паралельно с использование технологии OpenMP.
12. vec_TBB(…) вычисляет скалярное произведение векторов паралельно с использование технологии TBB.

Руководство пользователя

Программа реализована с использованием нескольких подпрограмм.

1. Пользователь должен либо ввести данные СЛАУ вручную в текстовый файл.
    a. Записать целочисленный размер системы и через пробел записать рациональные числа в количестве n^2 штук.
    b. Запустить программу typer [input] [output] – где input имя входного файла, в который пользователь записал СЛАУ, output имя выходного файла.
2. Либо сгенерировать СЛАУ в программе generator [number] [output] – где number является числом в диапазоне [0, 24] и определяет номер размера матрицы СЛАУ, output имя выходного файла.
    a. Программа выведет типы генерируемых матриц и попросит выбрать один из них. Требуется ввести целочисленное значение в диапазоне [1, 7].
3. Затем запускается программа before_code [number] [input] [output] – где number количество потоков, input имя входного файла, сгенерированного или typer, или generator, output имя выходного файла.
    a. Программа попросит выбрать тип паралельной версии. Требуется ввести целочисленное значение в диапазоне [1, 3].
4. После чего запускается программа эталон gauss [input] [output] – где input имя входного файла, сгенерированного или typer, или generator, output имя выходного файла.
5. И последним шагом запускается программа checker [input1] [input2] – где input1 имя файла содержащего результат программы before_code, а input2 имя файла содержащего результат программы gauss.

Подтверждение корректности

Для подтверждения корректности работы программы требуется выполнить пункты 4 и 5 из руководства пользователя. Программа checker будем сравнивать результат из программы before_code, решающей СЛАУ методом сопряженных градиентов, и результат программы gauss, которая решает СЛАУ методом Гаусса. Ошибка вычисляется как квадрат нормы разности векторов решений. В случае совпадения значений выводится «Совпадает», иначе «Не совпадает». При сравнении надо учитывать, что допустима некоторая погрешность, из-за которой результат выполнения программ могут отличаться на некоторый ε.

Результаты экспериментов по оценке масштабируемости

Ускорение, получаемое при использовании параллельного алгоритма для p процессоров, по сравнению с последовательным вариантом выполнения вычислений, определяется величиной:

$$S_p(n) = \frac{T_1(n)}{T_p(n)}$$

где n – количество входных данных задачи.

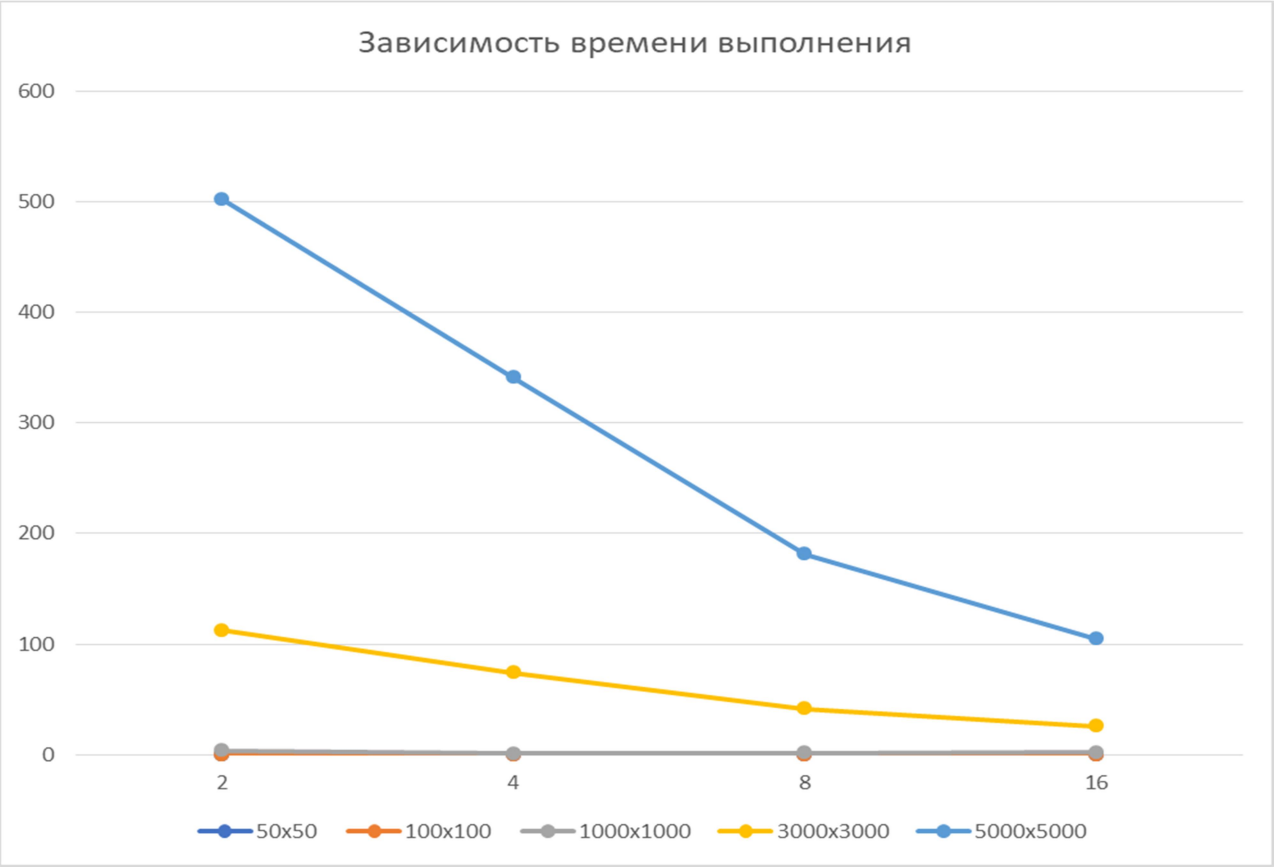
T_p – время работы алгоритма на p процессах (измеряется в секундах).

Тестовая инфаструктора:

Процессор	Intel® Core™ i5-3450 CPU @ 3.10GHz 4 ядра
Память	8 Gb
Операционная система	Windows 10 Домашняя x64
Среда разработки	Visual Studio 2017
Компилятор, профилировщик, отладчик	Visual Studio 2017
Библиотеки	Intel® Threading Building Blocks tbb2018_20180312oss, OpenMP 2.0

Результаты экспериментов:

Размер матрицы		50x50	100x100	1000x1000	3000x3000	5000x5000
Последовательная	1	0.00024	0.00228	5.50584	157.048	749.749
OpenMP	2	0.00096	0.00274	3.01302	112.27	502.368
	3	0.00169	0.00314	2.39370	74.079	341.028
	4	0.04307	0.08760	2.03233	41.578	180.992
TBB	2	0.00755	0.00325	3.14960	109.42	511.011
	3	0.01148	0.00390	2.38212	79.341	339.829
	4	0.01631	0.09223	1.99622	39.989	185.720



Ускорение:

Ускорение для матрицы 1000x1000	OpenMP	TBB
2 процесса	1,82734930	1,74810769
3 процесса	2,30013786	2,31131932
4 процессов	2,70912696	2,75813287

Заключение

В результате выполнения лабораторной работы была реализована программа с использованием технологий распараллеливания OpenMP и TBB, которая решает систему линейных алгебраических уравнения методом сопряженных градиентов.

Был взят за основу последовательный алгоритм данного метода и реализована его параллельная версия.

Были проведены тесты алгоритма на различных входных данных, а так же на разном количестве процессов. Была доказана корректность полученного решения, путем оценки погрешности.

Анализируя результаты, видно, что в версии OpenMP и TBB, ускорение улучшается при увеличении, как числа процессов, так и числа потоков.