Н. М. Гаврилова, М. С. Ларькин

Тюменский государственный университет, г. Тюмень

УДК 519.684.4

ИССЛЕДОВАНИЕ ТЕХНОЛОГИЙ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ НА ПРИМЕРЕ РЕШЕНИЯ СИСТЕМ ЛИНЕЙНЫХ УРАВНЕНИЙ С ТРЕХДИАГОНАЛЬНОЙ МАТРИЦЕЙ

Аннотация. В статье рассматривается применение технологии параллельного программирования при решении систем линейных уравнений с трехдиагональной матрицей коэффициентов при неизвестных.

Ключевые слова: Параллельное программирование, метод прогонки, встречная прогонка, блочная прогонка.

Введение

Современное развитие вычислительной техники строится на применении технологий параллельного программирования. Численное решение дифференциальных уравнений находит свое применение во многих отраслях науки. Неявная конечно-разностная аппроксимация дифференциальных уравнений приводит к необходимости решать систему линейных алгебраических уравнений(СЛАУ) вида $Ax = b$, где матрица A – трехдиагональная.

Метод прогонки

Основным методом решения систем уравнений с матрицей трехдиагонального вида является метод прогонки.

Систему уравнений $Ax = b$ можно представить в виде:

$$a_i x_{i-1} + b_i x_i + c_i x_{i+1} = f_i, \quad i = 1, \dots, N$$

$$a_0 = 0, \qquad c_0 = 0.$$

Метод прогонки является модификацией метода Гаусса, использующий специальный вид матрицы системы. Алгоритм метода разделяется на прямой и обратный ход:

1. Прямой ход – вычисляются специальные прогоночные коэффициенты;
2. Обратный ход – вычисляются неизвестные x_i.

Алгоритм метода прогонки можно организовывать разными способами. Разница в том, как происходит вычисления прогоночных коэффициентов и в какой порядке находятся неизвестные системы.

Алгоритм правой прогонки:

Прогоночные коэффициенты α и β можно найти последующим формулам:

$$\alpha_2 = \frac{-b_1}{c_1},\ \alpha_{i+1} = \frac{-b_i}{a_i\alpha_i + c_i}, \tag{1}$$

$$\beta_2 = \frac{f_1}{c_1},\ \beta_{i+1} = \frac{f_i - a_i\beta_i}{a_i\alpha_i + c_i},\quad i = 2,...,n-1. \tag{2}$$

Обратный ход:

$$x_n = \frac{f_n - a_n\beta_n}{a_n\alpha_n + c_n},\qquad x_i = \alpha_{i+1}x_{i+1} + \beta_{i+1},\quad i = n-1,...,1.$$

Алгоритм левой прогонки:

Прогоночные коэффициенты метода левой прогонки ξ и η определяются по следующим формулам:

$$\xi_n = \frac{-a_n}{c_n},\ \xi_i = \frac{-a_i}{c_i + b_i\xi_{i+1}},\quad i = n-1,...,2. \tag{3}$$

$$\eta_n = \frac{f_n}{c_n},\ \eta_i = \frac{f_i - b_i\eta_{i+1}}{c_i + b_i\xi_{i+1}},\quad i = n-1,...,2. \tag{4}$$

Обратный ход:

$$x_1 = \frac{f_1 - b_1\eta_2}{b_1\xi_2 + c_1}\qquad x_{i+1} = \xi_{i+1}x_i + \eta_{i+1},\quad i = 1,...,n-1.$$

Встречная прогонка

Метод встречной прогонки дает возможность организовать параллельный расчет в двух потоках. В первом потоке определяются прогоночные коэффициенты α_i, β_i метода правой прогонки для уравнений с номерами $1 \le i \le p$. Второй поток находит прогоночные коэффициенты метода левой прогонки ξ_i, η_i для уравнений с номерами $p \le i \le n$, $p = \left[\frac{N}{2}\right]$

При $i=p$ проводится сопряжение решений: определяется значение x_p из системы уравнений:

$$\begin{cases} x_p = \alpha_{p+1} x_{p+1} + \beta_{p+1} \\ x_{p+1} = \xi_{p+1} x_p + \eta_{p+1} \end{cases}$$

Определив значение x_p, можно найти все x_i, при $1 \le i < p$ в первом потоке, все значения x_i, при $p < i \le N$ во втором потоке [1].

Блочная прогонка

Идея метода заключается в том, что каждый поток будет обрабатывать $m = [N / p]$ строк матрицы A, где p число параллельных потоков. Рассмотрим случай системы из 12 уравнений, число потоков p =3 (рис. 1).

В каждой полосе матрицы, соответствующей потоку с номером k, организуется исключение поддиагональных элементов. Для этого производится вычитание строки с номером i, умноженной на константу a_{i+1} / c_i из строки с номером $i + 1$ с тем, чтобы коэффициент при неизвестной x_i в $(i + 1) - $й строке стал нулевым.

В результате, в каждой рассматриваемой полосе появится новый столбец коэффициентов, за исключением первой.

Число ненулевых элементов в строке останется тем же, но изменится структура уравнений.

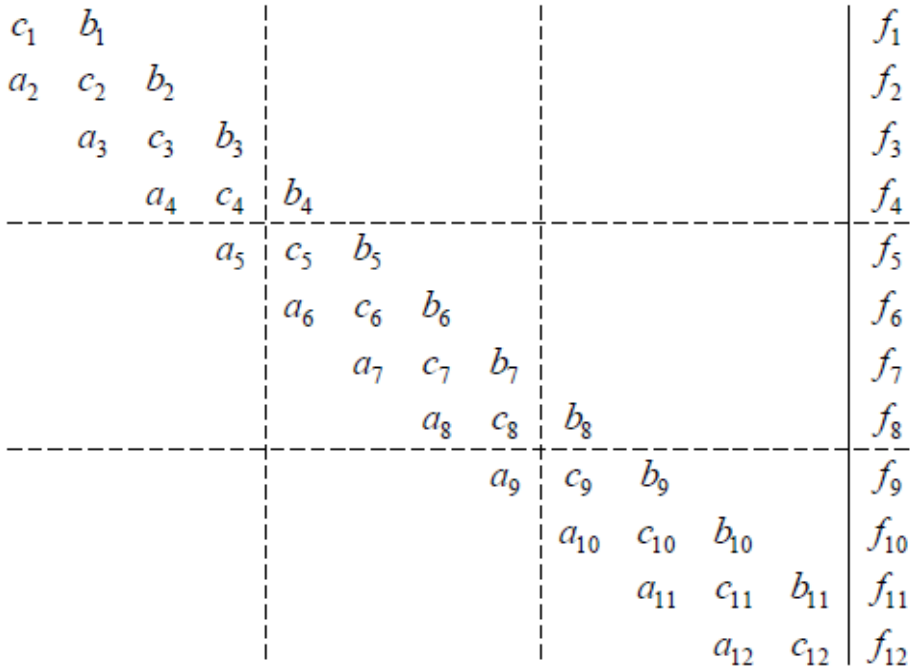

Рис. 1. Пример разделения исходной матрицы на блоки

Обратный ход алгоритма организует в каждом потоке исключение наддиагональных элементов, начиная с последнего. В итоге матрица коэффициентов исходной системы уравнений принимает блочный вид (рис 2):

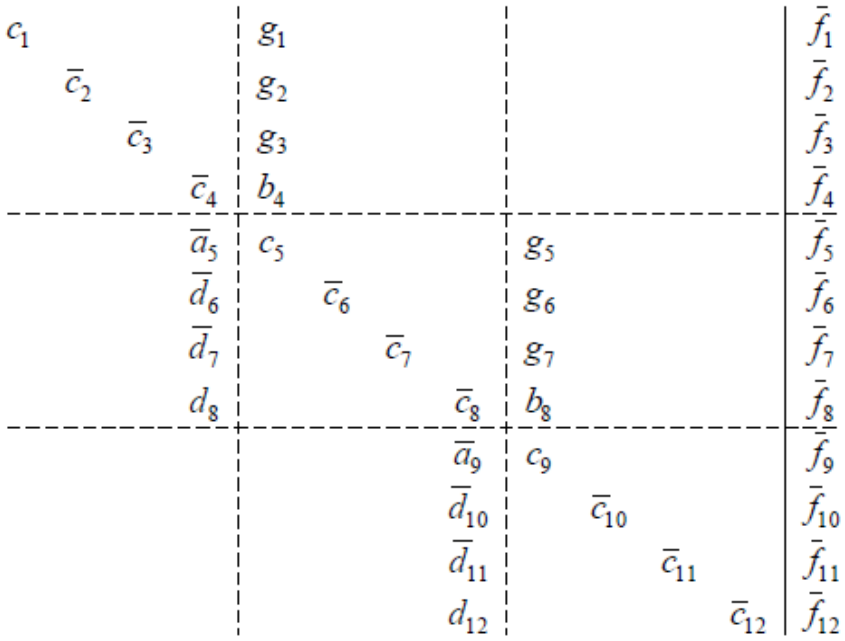

Рис. 2. Обратный ход блочной прогонки

Далее рассматривается система уравнений, содержащая первое и последнее уравнение каждой полосы. Эта система будет содержать $2p$

уравнений, и будет трехдиагональной. Для решения полученной системы уравнений применяется последовательный метод прогонки. После этого, с помощью этих граничных значений неизвестных определяются значения внутренних переменных в каждом блоке [1-3].

Применение OpenMP для реализации встречной и блочной прогонки

OpenMP – это технология написания параллельных программ для систем с общей памятью. Технология OpenMP использует встроенную библиотеку для организации распараллеливания вычислений, состоит из набора директив компилятора и библиотечных функций и дает возможность создавать многопоточные приложения на языках C/C++, Fortran. Поддерживается производителями аппаратуры (Intel, HP, IBM, и т.д.), разработчиками компиляторов (Intel, Microsoft, и т.д.).

Встречная прогонка подразумевает параллельное использование различного программного кода, т.е. в одном потоке обрабатывается одна половина матрицы по формулам (1) и (2) а вторая по формулам (3) и (4).

Для этих целей в OpenMP существует конструкция *section*, которая позволяет распределить программный код между потоками.

Блочная прогонка подразумевает использование одного и того же метода, применяемого к каждому блоку системы. Так как обработку каждого блока можно записать в один цикл и блоки независимы друг от друга, то можно применить конструкцию OpenMP *parallel for,* которая выполняет каждую итерацию цикла в отдельном потоке.

Применение пространства имен System.Threading для реализации встречной и блочной прогонки

Основной функционал для использования потоков в языке C# сосредоточен в пространстве имен System.Threading. В нем определен класс, представляющий отдельный поток – класс Thread.

При реализации встречной прогонки, используя язык программирования C# необходимо создать два потока. Первый поток обрабатывает половину СЛАУ по формулам (1) и (2), а второй поток обрабатывает вторую половину СЛАУ по формулам (3) и (4).

Для реализации блочной прогонки необходимо создать массив потоков каждый из которых обрабатывает свою часть исходной системы, при этом к каждому блоку СЛАУ применяется один и тот же алгоритм [5].

Сравнительный анализ времени работы методов прогонки с использованием технологии параллельных вычислений на языках программирования C++ и C#

1. Вычисления проводились устройстве со следующими характеристиками:
    - Процессор – Intel®Core™ $i5$-825OU.
    - Количество ядер – 4.
    - ОЗУ: 4 ГБ.
2. Сравниваемые алгоритмы:
    - правая прогонка (последовательный алгоритм);
    - встречная прогонка (распараллеливание на 2 потока);
    - блочная прогонка (распараллеливание на 4 -10 потоков).

Количество уравнений N изменялось от 10^3 до 10^6. Трехдиагональная матрица коэффициентов имела диагональное преобладание (элементы главной диагонали являлись удвоенной суммой элементов в строке).

Время расчетов T измерялось в миллисекундах. Каждый алгоритм выполнялся 10 раз, после чего рассчитывается среднее время работы алгоритма. На основании полученных результатов было рассчитано ускорение S работы параллельных алгоритмов по формуле:

$$S = \frac{T_1}{T_p}, \text{где}$$

T_1 –время работы последовательного алгоритма, T_p – время работы параллельного алгоритма.

В таблице 1 представлено сравнение расчетов с использованием правой, встречной и блочной прогонок, реализованных на языке программирования C++.

Таблица 1. Результаты времени работы алгоритмов

N/T, мс	Правая прогонка	Встречная прогонка (2 потока)	Блочная прогонка (4 потока)	Блочная прогонка (10 потоков)
10^3	4	8	14	24
10^4	124,3	143,8	167,7	227,15
10^5	580,1	600,12	745,13	948,18
10^6	1923,8	1736,72	1702,44	2315,2

На рис. 3 представлено ускорение работы параллельных алгоритмов S в зависимости от размерности исходной матрицы N.

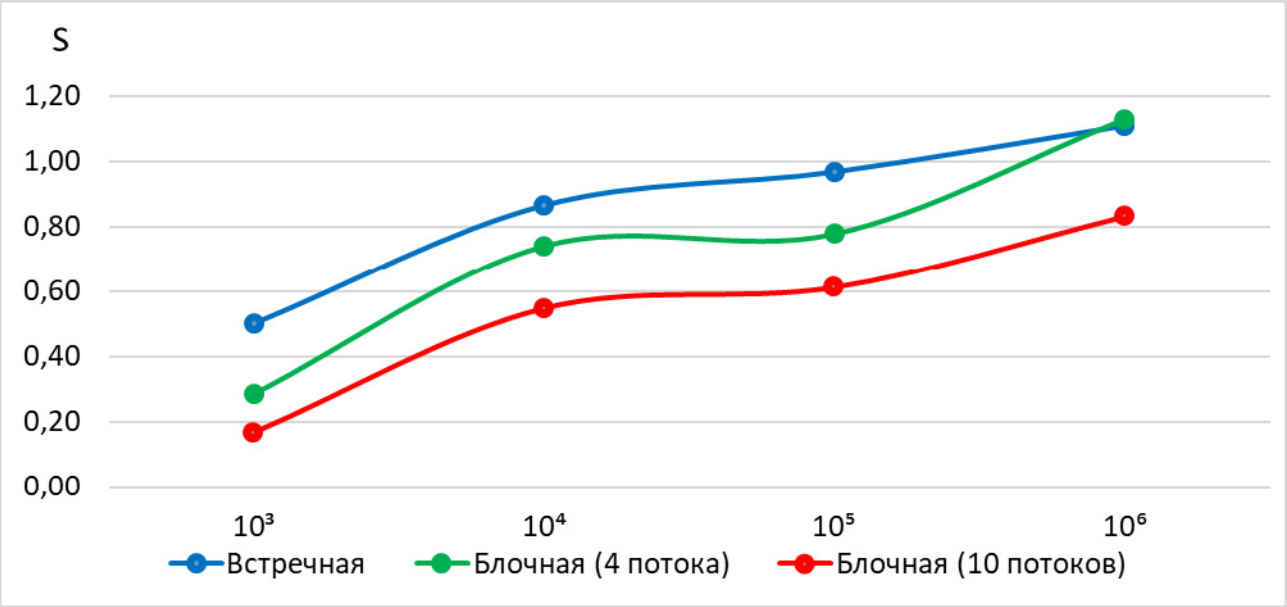

Рис. 3. Ускорение параллельных методов прогонки выполненных на C++.

В таблице 2 представлено сравнение расчетов, реализованных на языке программирования C# .

Таблица 2. Результаты времени работы алгоритмов

N/T, мс	Правая прогонка	Встречная прогонка	Блочная (4 потока) прогонка	Блочная (10 потоков) прогонка
10^3	12	24	35	86
10^4	203,3	222,74	284,43	412,15
10^5	913,2	1003,88	1203,14	1806,17
10^6	2734,3	2000,5	2056,27	3453,1

Анализ результатов

Приведенные в таблицах 1 и 2 расчетные данные позволяют сделать следующие выводы:

1. Распараллеленные алгоритмы на размерностях N до 10^6 работают медленнее, чем последовательные.

2. Блочная прогонка в отличии от встречной может быть распараллелена на произвольное число потоков.

3. Большое количество потоков не гарантирует быструю скорость работы алгоритма.

4. Реализация с использованием технологии OpenMP (C++) выполняется быстрее чем, при создании потоков с помощью класса Thread (C#).

Заключение

Метод прогонки является одним из экономичных и простых в реализации в последовательном режиме. Алгоритмы встречной и блочной прогонки позволяют распараллелить решение системы уравнений на потоки, что позволяет ускорить время решения системы при больших размерностях исходной системы. Однако если размерность невелика, то применение

технологий параллельного программирования лишь замедлит время выполнения алгоритма.

СПИСОК ЛИТЕРАТУРЫ

1. Баркалов К.А. Параллельные численные методы: Лекционные материалы. – Нижний Новгород, 2011. – 30 с.
2. Воеводин В.В., Воеводин Вл.В. Параллельные вычисления. – СПб.: БХВПетербург, 2002. – 608 с.
3. Богачев К.Ю. Основы параллельного программирования Изд.: Бином, 2003. С. 344.
4. Алабужев А.А. Основы параллельного программирования: учеб.-метод. пособие / А.А. Алабужев. – Пермь: Изд-во ПГУ, 2007. – 100 с.
5. Metanit.com Сайт о программировании [Электр. ресурс]. – Режим доступа https://metanit.com/sharp/tutorial/12.2.php (дата обращения: 05.04.2019).