Параллельный метод для решения уравнения Пуассона*

А.А. Игнатьев, М.А. Затевахин

Представлен параллельный метод для решения уравнения Пуассона, основанный на методе бисопряженных градиентов. В методе применяется декомпозиция на равновеликие прямоугольные подобласти. Метод прост в реализации и не использует разреженное представление матрицы коэффициентов. Тестовые расчеты показали работоспособность и эффективность метода.

1. Введение

Численное решение уравнения Пуассона является важным элементом многих задач вычислительной физики. Так, например, при решении уравнений гидродинамики несжимаемой жидкости решение уравнения Пуассона требует значительных вычислительных усилий, особенно для расчетов с большим числом ячеек (от сотен тысяч и выше). В связи с этим большое значение приобретает задача параллелизации вычислительного алгоритма для его решения. Часто в таких случаях используют уже готовые параллельные пакеты, такие как, например, PETSc [1] и другие подобные, которые требуют явного формирования разреженной матрицы коэффициентов линейной системы. В данной работе мы представляем простой параллельный алгоритм численного решения уравнения Пуассона, основанный на итерационном стабилизированном методе бисопряженных градиентов с использованием библиотеки MPI, который не требует явного формирования матрицы коэффициентов и, соответственно, не использует операции умножения матрица – вектор. Численные эксперименты, проведенные на суперкомпьютерных комплексах СКИФ МГУ "Чебышев" и "Скиф К1000М", показали работоспособность и эффективность алгоритма.

2. Параллельный алгоритм

Разностная аппроксимация уравнения Пуассона

$$\Delta f = \rho \qquad (1)$$

приводит, как известно, к системе линейных уравнений, которую можно записать в матричном виде как

$$Ax = b, \qquad (2)$$

где x - вектор столбец неизвестных (дискретный набор искомой сеточной функции) размерности n, b - вектор столбец правых частей и A - матрица коэффициентов размерности $n \times n$. На практике для решения системы (2), особенно в случае большого числа неизвестных, используют обычно какую-либо модификацию метода сопряженных градиентов, считающегося наиболее эффективным.

В нашем случае мы будем рассматривать стабилизированную версию метода бисопряженных градиентов [2], как наиболее универсального алгоритма, позволяющего решать системы, как с симметричными, так и с несимметричными матрицами. Псевдокод этого метода представлен на рис 1.

* Работа выполнена в рамках научно-технической программы Союзного государства СКИФ-ГРИД, проект 407.

1) $x = x_0$

2) $r = b - Ax$,

3) $r_0 = r$,

4) $p = r$,

5) $\rho = (r_0, r)$, если $\rho = 0$ метод не работает.

Итерационный цикл . . .

6) $v = Ap$,

7) $\alpha = \dfrac{\rho}{(r_0, v)}$,

8) $s = r - \alpha v$,

9) $t = As$,

10) $\omega = \dfrac{(t, s)}{(t, t)}$,

11) $x = x + \alpha p + \omega s$,

12) $r = s - \omega t$,

13) Проверка сходимости: если $\|r\| \le \varepsilon$ стоп

14) $\beta = \rho$,

15) $\rho = (r_0, r)$, если $\rho = 0$ метод не работает.

16) $\beta = \dfrac{\rho\,\alpha}{\beta\,\omega}$,

17) $p = r + \beta(p - \omega v)$

Конец цикла

Рис. 1. Алгоритм BCGSTAB

Здесь x, x_0 - вектор решения и его начальное приближение, r, r_0 - вектор невязки и его начальное значение, t, v, s, p - рабочие вектора, $\rho, \alpha, \beta, \omega$ - скаляры. Круглыми скобками $(\cdot, \cdot)$ обозначено скалярное произведение векторов. Для описания параллельного алгоритма рассмотрим для простоты двумерную прямоугольную область, покрытую ортогональной сеткой. Рассмотрим также простейшую декомпозицию на две равновеликие подобласти, соответствующие процессорам 1 и 2 (Рис 2.)

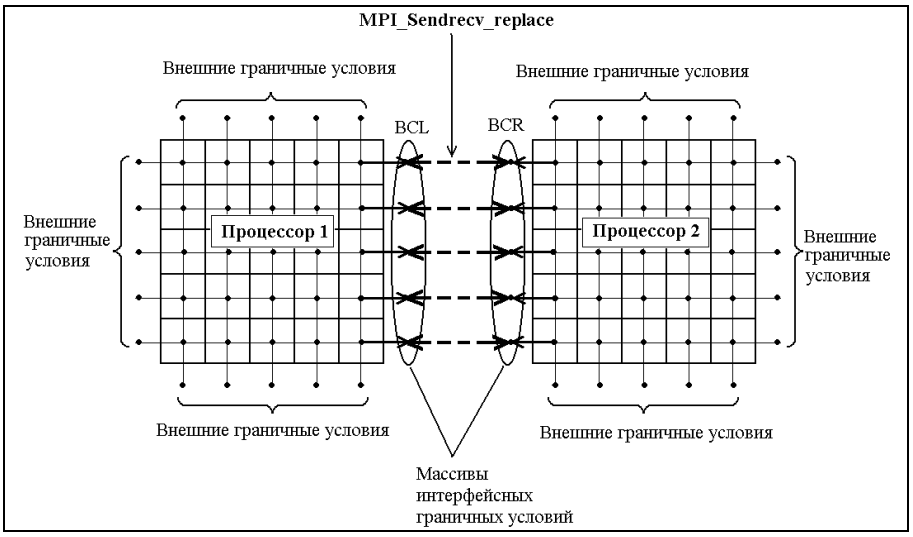

Рис. 2. Декомпозиция расчетной области и обмен данными

Здесь вектор x (искомая сеточная функция), как и все другие вектора, делится на два подвектора, имеющих размерность $n/2$ и хранящихся в памяти соответствующих процессоров. Заметим, что алгоритм BCGSTAB не требует формирования матрицы коэффициентов A в явном виде. Единственное, что требуется, это процедура вычисления невязки $r(z) = b - Az$ в зависимости от некоторого вектора z. Произведения Ap и As на шагах алгоритма 6 и 9 могут быть легко вычислены как $b - r(p)$ и $b - r(s)$, соответственно. При этом вектор правых частей b вычисляется один раз перед началом итерационного цикла как $b = r(0)$.

Пусть в нашем примере мы имеем ортогональную двумерную сетку с индексами *i,j* и шагами $\Delta x, \Delta y$. Тогда невязка для уравнения (1) в ячейке *i,j* может быть вычислена на 5-ти точечном шаблоне стандартным образом:

$$r_{i,j}(z) = \rho_{i,j} - \left(z_{i+1,j} - 2z_{i,j} + z_{i-1,j}\right)/\Delta x^2 - \left(z_{i,j+1} - 2z_{i,j} + z_{i,j-1}\right)/\Delta y^2 \qquad (3)$$

Перед вычислением всех компонент вектора невязки в пределах одной подобласти (процессора) необходимо задать граничные условия, которые мы здесь разделяем на внешние (на внешней границе расчетной области), задание которых не представляет сложности, и интерфейсные (на границе стыка с соседней подобластью), задание которых требует обмена информацией с соседом. Это может быть сделано следующим образом.

Каждый процессор имеет массив для хранения интерфейсных граничных условий. В нашем примере это массивы BCL (для процессора 1) и массив BCR для процессора 2 (рис. 2). Заполнение этих массивов проходит в два этапа:

1) каждый процессор копирует свои приграничные значения вектора z в массив (на рис. 2 этот процесс обозначен сплошными стрелками)

2) каждый процессор вызывает функцию

`MPI_Sendrecv_replace` `(buf, count, datatype, dest, sendtag, source, recvtag, comm, status)`, где в качестве буфера (`buf`) каждый процессор использует имя своего массива BCL или BCR. При этом `dest` равен `source` и равен номеру процессора, с которым происходит обмен. Остальные параметры имеют стандартный смысл. При вызове этой функции происходит обмен данными между массивами BCL и BCR (на рис.2 этот процесс обозначен пунктирными стрелками).

Если число подобластей (процессоров) больше, чем 2, то обмен данными между подобластями можно организовать параллельным образом. Покажем это на примере одномерной задачи (двухмерная и трехмерные задачи естественным образом обобщаются). Пусть мы имеем цепочку подобластей, обменивающихся данными с соседями. Рассмотрим следующий двухэтапный процесс обмена: на первом этапе *четные* подобласти обмениваются *правыми* приграничными данными с *левыми* нечетных подобластей. И на втором этапе, наоборот, *нечетные* подобласти обмениваются *правыми* приграничными данными с *левыми* четных. Так как на каждом этапе обменивающиеся пары независимы, то такие обмены будут идти параллельно. Схематично этот процесс показан на рисунке 3.

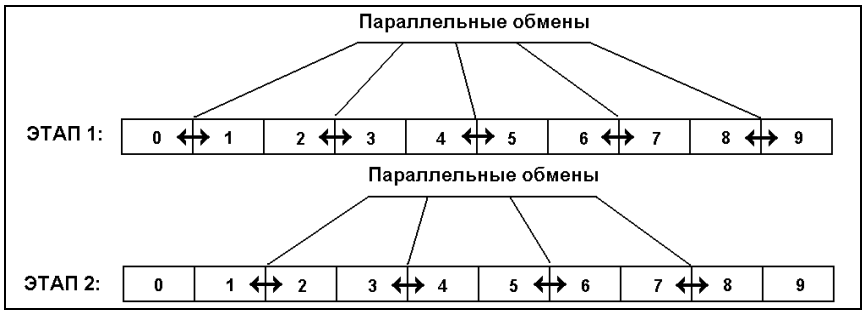

Рис. 3. Схема двухэтапных параллельных обменов между подобластями

Все шаги алгоритма BCGSTAB (Рис.1) идут параллельно на каждом процессоре. При этом каждый процессор обрабатывает свои подвектора, фигурирующие в алгоритме. Однако, вычис-

ление скалярных произведений векторов требует дополнительного обмена данными. Так как скалярное произведение векторов представляет собой сумму попарных произведений соответствующих компонент вектора, то вычисленные таким образом скалярные произведения на каждом процессоре будут представлять лишь часть общей суммы. Поэтому, для получения полного скалярного произведения, необходимо просуммировать результаты каждого процессора. Это можно сделать с помощью функции
`MPI_Allgather` `( sendbuf, sendcount, sendtype, recvbuf, recvcount, recvtype, comm )`, которая вызывается каждым процессором после вычисления своей части скалярного произведения. Здесь `sendbuf` – указатель на переменную, содержащую результат вычисления текущего процессора, `sendcount = recvcount = 1` и `recvbuf` – массив размерности общего числа процессоров. Остальные параметры имеют стандартный смысл. В результате работы этой функции в распоряжении каждого процессора оказывается массив `recvbuf`, содержащий все части общего скалярного произведения. После этого каждый процессор суммирует элементы этого массива и получает, таким образом, полное скалярное произведение, одинаковое для всех процессоров.

3. Результаты численных экспериментов

Для проверки работоспособности алгоритма и оценки его эффективности были проведены тестовые расчеты трехмерной задачи Пуассона (1) в кубической области на равномерной регулярной сетке, содержащей $256^3 = 16{,}777{,}216$ ячеек с граничными условиями Неймана $\partial f / \partial n = 0$ на всех границах. Правая часть уравнения (1) задавалась в виде $\rho_{i,j,k} = \xi_{i,j,k} - \overline{\xi}$, где i,j,k – индексы трехмерной сетки, $\xi_{i,j,k}$ число от -0.5 до 0.5, полученное с помощью генератора псевдослучайных чисел и $\overline{\xi}$ - среднее от $\xi_{i,j,k}$ по всем ячейкам. Такая постановка задачи близка к ситуации, возникающей при решении уравнений гидродинамики несжимаемой жидкости.

В таблице 1 приведены результаты тестирования, полученные на суперкомпьютерных комплексах СКИФ МГУ "Чебышев" и "Скиф К1000М" с нормой невязки $\varepsilon = 10^{-3}$, которая достигалась за 100 итераций. Для определения времени выполнения использовалась функция **`MPI_Wtime()`**, вызываемая непосредственно перед итерационным циклом и после него. На рисунке 4 показан график достигнутого ускорения счета.

Таблица 1. Результаты тестирования.

Число процессоров	**Время выполнения, с**	
	Скиф К1000М	**СКИФ МГУ**
1	1270	1200
2	501	604
4	230	386
8	133	405
16	69	123
32	31	55
64	14	18
128	-	7.3
512	-	0.36

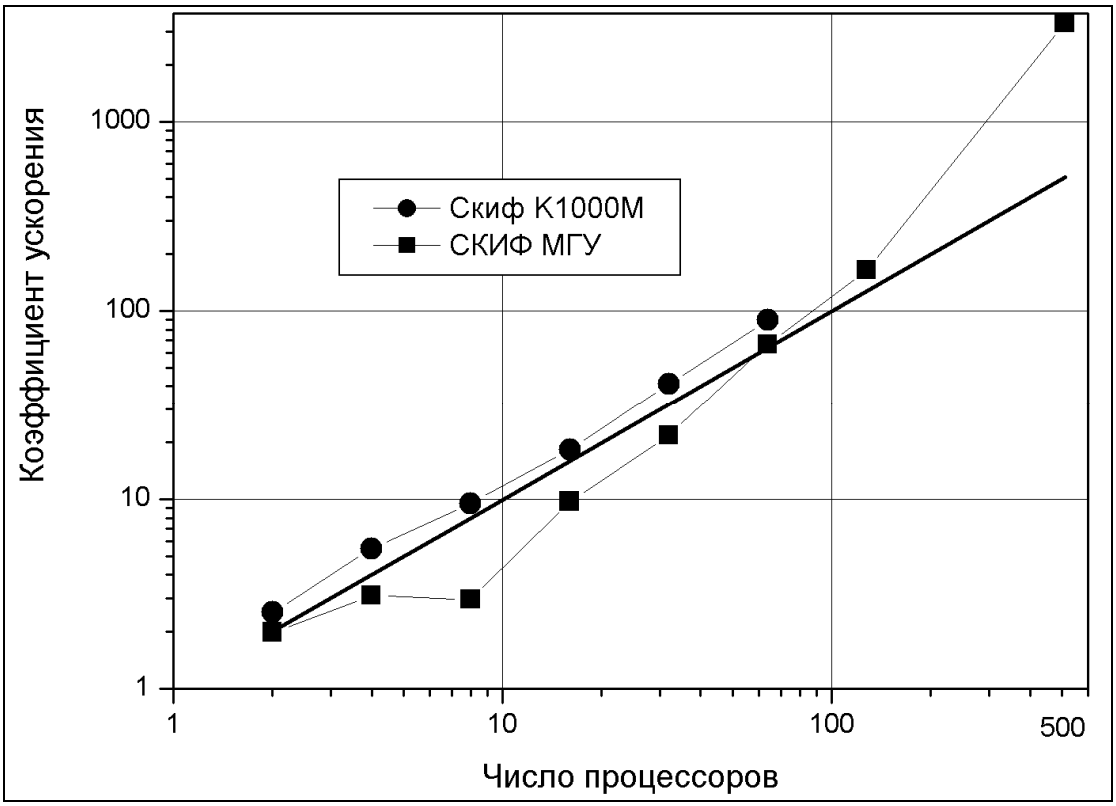

Рис. 4. Коэффициент ускорения на кластерах СКИФ МГУ "Чебышев" и "Скиф К1000М"

4. Заключение

Проведенные тесты подтвердили работоспособность и эффективность предложенного параллельного алгоритма. Зависимость ускорения счета близка к линейной. Отклонение от линейности объясняется, по всей вероятности, особенностями работы кластеров.

В будущем представляет некоторый интерес модернизация алгоритма с помощью введения в него предобуславливателя. Хотя в процессе работы авторами были сделаны попытки ввести простейший предобуславливатель Якоби для ускорения сходимости, однако заметного эффекта не было обнаружено.

Методология предложенного параллельного алгоритма имеет общий характер и может быть, очевидно, применена не только к уравнению Пуассона, но и к другим уравнениям математической физики.

В настоящий момент продолжается работа по интегрированию данного алгоритма в гидродинамический код для расчета динамики микрочастиц в турбулентном потоке в каналах. Данный код предназначен для моделирования последствий чрезвычайных ситуаций и прогнозирование техногенных и природных катастроф при запроектных авариях на АЭС.

Литература

1. [http://www.mcs.anl.gov/petsc/petsc-as/]
2. Y. Saad. Iterative Methods for Sparse Linear Systems. // SIAM, 2d edition, Philadelphia, PA, 2003.