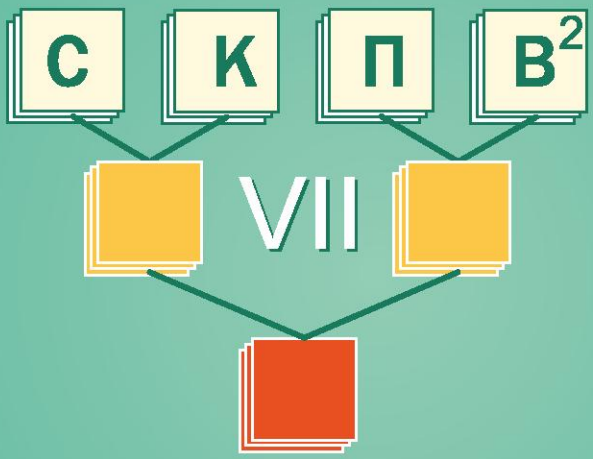
VII СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ



МАТЕРИАЛЫ КОНФЕРЕНЦИИ

РАБОТА ВЫПОЛНЕНА ПРИ ПОДДЕРЖКЕ РФФИ
РФФИ

intel

Томск-2014

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

СЕДЬМАЯ СИБИРСКАЯ КОНФЕРЕНЦИЯ ПО ПАРАЛЛЕЛЬНЫМ И ВЫСОКОПРОИЗВОДИТЕЛЬНЫМ ВЫЧИСЛЕНИЯМ

Томск, 12–14 ноября 2013 года

ИЗДАТЕЛЬСТВО ТОМСКОГО УНИВЕРСИТЕТА
2014

УДК 519.6
ББК 22.19
С 51

С 51 **Седьмая** Сибирская конференция по параллельным и высокопроизводительным вычислениям / Под ред. проф. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2014. – 146 с.
ISBN 978–5–7511–2260–7

Включены материалы Седьмой Сибирской конференции по параллельным и высокопроизводительным вычислениям, проходившей 12–14 ноября 2013 г. в Томском государственном университете при поддержке Министерства образования и науки РФ, Суперкомпьютерного консорциума России, Российского фонда фундаментальных исследований и ЗАО Интел.

Рассмотрены актуальные проблемы организации параллельных вычислений на многопроцессорных системах, современное состояние и перспективы развития методов параллельных вычислений.

Для студентов, аспирантов, преподавателей, научных работников, желающих изучить и практически использовать в научной работе высокопроизводительные вычислительные ресурсы.

УДК 519.6
ББК 22.19

ISBN 978–5–7511–2260–7

Содержание

Параллельная реализация модифицированного метода траекторий для уравнения неразрывности*

А.В. Вяткин [1], А.А. Ефремов [1], Е.Д. Карепова [1,2], В.В. Шайдуров [1]

[1] Институт вычислительного моделирования СО РАН,
[2] Институт математики и фундаментальной информатики СФУ,
Красноярск

Обсуждается параллельная реализация численного решения двумерного уравнения переноса полулагранжевым методом для высокопроизводительных вычислительных систем гибридной архитектуры. Проведено исследование производительности последовательной и нескольких параллельных реализаций алгоритма, выполненных с помощью технологий OpenMP и CUDA для языка Си.

Введение

В течение двух последних десятилетий активно развивались численные алгоритмы [1–4] решения гиперболических уравнений, основанные на смешанном полулагранжевом (semi-Lagrangian) подходе. Преимуществом подхода является выполнение условия Куранта–Фридрихса–Леви без ограничения на шаг по времени [5,6], что делает его удобным инструментом для решения задач с большими значениями скоростей. Полулагранжевый подход также используется при построении численных методов в случаях, когда на соседних слоях по времени необходимо использовать разные пространственные дискретизации вычислительной области [4]. Еще одним достоинством подхода является выполнение закона сохранения в консервативных версиях метода. Консервативность метода основана на выполнении локального закона сохранения, который справедлив для внутренних узлов сетки [1–4].

В работах [7–9] в рамках полулагранжевого подхода обоснован метод численного решения начально-краевой задачи для гиперболического уравнения, записанного в дивергентной форме. Метод основан на точном тождестве для пространственных интегралов,

*Работа выполнялась в рамках гранта РФФИ № 11-01-00224-а, интеграционного проекта СО РАН № 130, проекта Президиума РАН № 18.2.

области интегрирования которых лежат на соседних слоях по времени. Порядок сходимости метода зависит от асимптотической точности вычисления интегралов. В [7–9] представлены теоремы, позволяющие точно учитывать краевые условия и обосновывающие сходимость метода с первым порядком точности.

Основным недостатком полулагранжевого подхода следует считать большую вычислительную емкость и сложность реализации получаемых на его основе численных методов. В связи с этим актуальным является вопрос параллельной реализации этих методов.

В настоящей работе обсуждаются основные проблемы, связанные с распараллеливанием метода, описанного в [9], для высокопроизводительных систем гибридной архитектуры с помощью технологий OpenMP и CUDA.

Постановка дифференциальной задачи и последовательный алгоритм

Пусть $\overline{D} = [0,1] \times [0,1]$ – единичный квадрат. В области $[0, T] \times \overline{D}$ рассмотрим уравнение переноса:

$$\frac{\partial \rho}{\partial t} + \frac{\partial (u\rho)}{\partial x} + \frac{\partial (v\rho)}{\partial y} = 0, \tag{1}$$

где $u(t,x,y)$ и $v(t,x,y)$ – достаточно гладкие, известные в $[0, T] \times \overline{D}$ функции. Пусть $\forall\ t \in [0,T]$ на верхней и нижней границах $\overline{D}$ выполнены условия прилипания:

$$\begin{aligned} u(t,x,y)\big|_{y=0} &= u(t,x,y)\big|_{y=1} = 0, \\ v(t,x,y)\big|_{y=0} &= v(t,x,y)\big|_{y=0} = 0. \end{aligned} \tag{2}$$

Пусть также при $x = 1$ верно:

$$u(t,x,y)\big|_{x=1} \geq 0\ \forall\ t \in [0, T]. \tag{3}$$

Зададимся также начальными и граничными условиями:

$$\forall\ (x,y) \in \overline{D} \quad \rho(0,x,y) = \rho_{\text{init}}(x,y), \tag{4}$$

$$\forall\ t \in [0, T] \quad \rho(t,0,y) = \rho_{\text{lb}}(t,y). \tag{5}$$

Полулагранжевый подход к построению численного метода решения задачи (1)–(5) в [9] основан на точном интегральном равенстве

$$\int\limits_{\Omega_{i,j}} \rho\left(t_k, x, y\right) d\Omega = \int\limits_{Q_{i,j}^{k-1}} \rho\left(t_{k-1}, x, y\right) dQ + \int\limits_{I_{i,j}^{k-1}} \left(\rho_{lb} u\right)\left(t, 0, y\right) dI, \tag{6}$$

где слева интеграл берется по ячейке сетки Ω_{ij} на текущем слое по времени, а справа – по соответствующему Ω_{ij} криволинейному многоугольнику Q_{ij} на предыдущем слое с учетом возможного дополнительного слагаемого для левой границы расчетной области (рис. 1, слева). В численном методе [9] криволинейный многоугольник Q_{ij} и интегралы в (6) вычисляются с необходимой для достижения нужной сходимости метода точностью.

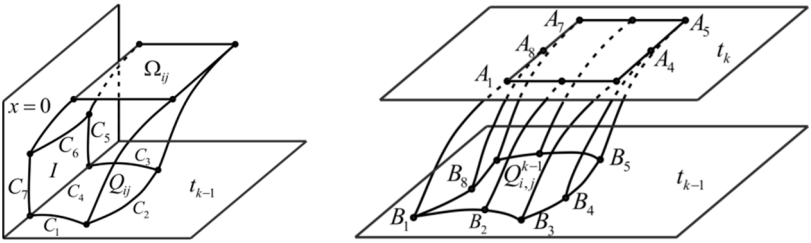


Рис. 1. Основная идея используемого полулагранжевого подхода

Опишем последовательный алгоритм [9].

1. Положим $\rho^h\left(t_0, x_i, y_j\right) = \rho_{\text{init}}\left(x_i, y_j\right)$, $i, j = 0, ..., N$, используя начальные данные (4).
2. *Цикл по времени*: для каждого шага по времени $k = 1, ..., K$ выполнить:
   Цикл по пространству: для каждой ячейки сетки Ω_{ij}^k, $i, j = 0, ..., N$ выполнить:
   2.1. Для каждой точки $A_n\left(x_n, y_n\right)$, $n = 1, .., 8$ (рис. 1, справа) решить систему обыкновенных дифференциальных уравнений для определения соответствующих траекторий $A_n B_n^h$ и координат точек $B_n^h\left(\tilde{x}_n\left(t_{k-1}\right), \tilde{y}_n\left(t_{k-1}\right)\right)$ криволинейного восьмиугольника

P_{ij}^{k-1}.

2.2. Для тех траекторий $A_n B_n^h$, которые пересекают граничную плоскость $P_0 = \{(0, y, t)\}$, определить координаты пересечения траекторий $A_n B_n^h$ с этой плоскостью.

2.3. Вычислить $J_{ij}^{k-1} = \int\limits_{P_{ij}^{k-1}} \rho_h^l \left(t_{k-1}, x, y\right) dP + \int\limits_{L_{i,j}^{k-1}} \left(\rho_{lb} u\right)^l \left(t, 0, y\right) dL$, где в качестве подынтегральных функций используются соответствующие билинейные интерполяционные многочлены, а интегралы берутся по каждому непустому пересечению $P_{i,j}^{k-1} \cap \left\{\left[x_p, x_{p+1}\right) \times \left[y_q, y_{q+1}\right)\right\}$ и $L_{i,j}^{k-1} \cap P_0$ в отдельности.

2.4. Вычислить $\rho^h \left(t_k, x_i, y_j\right) = \dfrac{1}{mes\left(\Omega_{ij}\right)} J_{ij}^{k-1}$, где через $mes\left(\Omega_{ij}\right)$ обозначена площадь ячейки Ω_{ij}.

Конец цикла по пространству.

2.5. При необходимости вычислить нормы решения и ошибки, другую статистическую информацию для текущего шага по времени.

Конец цикла по времени.

Отметим, что наиболее вычислительно трудоемким в алгоритме является анализ расположения криволинейного многоугольника P_{ij}^{k-1} относительно узлов сетки $(k-1)$-го слоя по времени. А именно, для точного, до погрешности компьютерной арифметики, вычисления интегралов на шаге 2.3 применяется процедура разбиения P_{ij}^{k-1} на составные части таким образом, что каждая часть лежит только в одной ячейке Ω_{pq} сетки $(k-1)$-го слоя по времени.

Параллельные реализации алгоритма

Для OpenMP-версии алгоритма, основанной на распараллеливании плотно вложенных циклов, исследовано влияние на производительность технологии HyperThreading (HT), а также проанализирована зависимость ускорения от размерности задачи и количества используемых потоков. Анализ результатов показал следующее.

1. Накладные расходы, связанные с синхронизацией в OpenMP-версии, невелики.

2. Для получения максимального ускорения OpenMP-версии программы выгодно использовать технологию HT с загрузкой максимально возможного количества логических ядер. При имеющихся в нашем распоряжении 12 физических ядрах программе было доступно максимально 24 логических ядра при включенной HT и 12 – при выключенной. Эксперименты показали, что время выполнения программы на 24 нитях при включенной HT в среднем на 14 % меньше времени выполнения этой же программы на 12 нитях без технологии HT (рис. 2).

3. Использование оптимизации компилятора существенно (более чем в 2 раза) уменьшает время работы последовательной программы (табл. 1). Поскольку компилятор никак не оптимизирует код CUDA, то время выполнения CUDA-версий программы не зависело от опций компиляции (табл. 1).

В табл. 1 и на рис. 3 приведены данные по ускорению нескольких параллельных OpenMP- и CUDA-версий программ.

Таблица 1. Время выполнения различных версий программы

Версия	Размерность сетки (количество шагов по времени)				
	80*80 (400)	160*160 (800)	320*320 (1600)	640*640 (3200)	1280*1280 (6400)
Последовательная, без опт.	20,16	159,40	1268,46	10107,80	*
Последовательная, опт. O2	9,99	78,97	626,72	4980,61	39598,90
Последовательная, опт. O3	9,87	78,08	619,80	4936,25	39202,91
OpenMP(12)**, без опт., HT-	1,98	12,52	103,45	819,06	6519,87
OpenMP(12)**, опт. O2, HT-	0,92	7,13	56,10	444,15	3535,53
OpenMP(24)**, опт. O2, HT+	0,91	6,17	49,93	388,20	3080,91
CUDA версия 1	2,55	13,06	74,02	***	***
CUDA версия 2	3,20	8,27	42,60	308,82	**
* – результат за разумное время не получен.					
** – приведены результаты на 12 нитях при отключенной технологии HT и на 24 – при включенной HT.					
** – не удалось запустить ядро из-за нехватки количества регистров.					

Отметим, что имеется ряд существенных ограничений алгоритма для его успешной адаптации для систем гибридной архитектуры с использованием технологии CUDA.

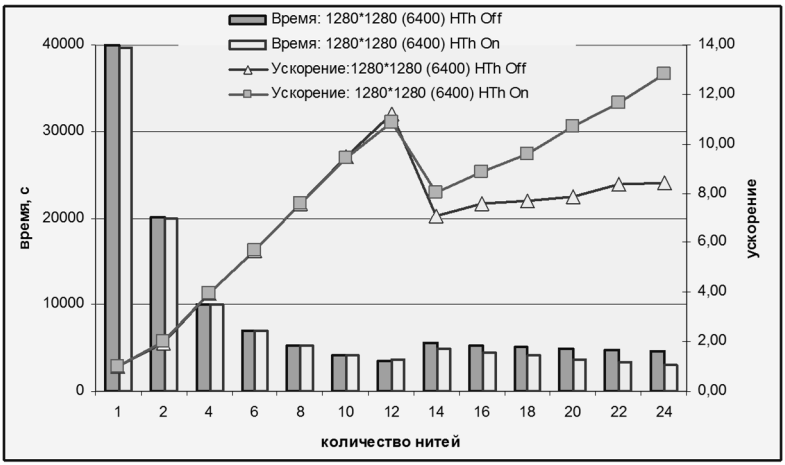


Рис. 2. Время выполнения OpenMP-версии программы (основная ось) и достигнутое ускорение относительно лучшей последовательной версии (дополнительная ось). Сравнение расчетов при включенной и выключенной поддержке технологии HyperThreading

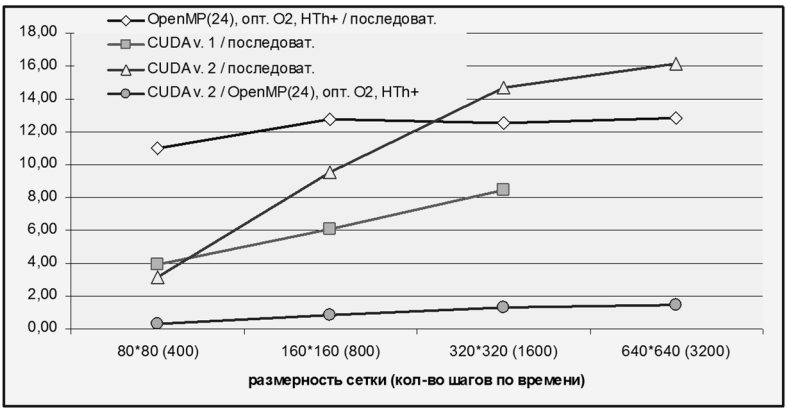


Рис. 3. Ускорение параллельных версий программы

Во-первых, существует традиционное для технологии CUDA ограничение по памяти. В нашем случае оно выражается в ограничении на размерность сетки по пространству, если использовать один поток CUDA для выполнения шагов 2.1–2.4 последовательного алгоритма (ядро CUDA – тело вложенного цикла). Для снятия этого ограничения полезным оказывается подход, основанный на «переиспользовании потоков» для обработки нескольких итераций цикла по пространству.

Другая возможность – использование нескольких графических устройств одновременно. Однако поскольку в полулагранжевом методе разностный шаблон подстраивается под решение и не является равномерным, то реализация обоих подходов имеет ряд дополнительных сложностей.

Во-вторых, анализ расположения ячейки интегрирования относительно разностной сетки на предыдущем шаге по времени содержит много ветвлений и вложенных вызовов функций, что делает актуальным второе традиционно узкое место графических процессоров общего назначения – аппаратное ограничение количества регистров на массиве потоковых мультипроцессоров (Streaming Multiprocessor, SM). В результате для повышения эффективности распараллеливания необходимо существенно менять последовательный алгоритм.

В численных экспериментах на вычислительной системе с видеокартой NVIDIA TESLA C2050 (CC 2.0) достигалось 16-кратное ускорение параллельной версии программы (рис. 3).

Обсуждение результатов

В работе метод, основанный на полулагранжевом подходе, реализован для двумерного гиперболического уравнения в дивергентной форме и распараллелен для вычислительных систем с общей памятью и гибридной архитектурой. Проведены исследования эффективности и производительности параллельных версий.

На первый взгляд сравнение технологий OpenMP и CUDA применительно к нашему алгоритму демонстрирует преимущества OpenMP. Однако хочется заметить, что некорректно сравнивать количество ядер GPU (в нашем случае 448 ядер на видеокарту) и количество ядер CPU (у нас 12 физических или 24 логических). В архитектуре ядра GPU можно найти только один аналог составляющих ядра CPU – арифметико-логическое устройство. Правильнее сравнить ядро CPU с потоковым мультипроцессором (SM), который, кроме ядер GPU, как и ядра CPU, содержит разнообразные устройства управления. Следует также учитывать, что архитектура SM на GPU заточена под работу с графикой, тогда как архитектура CPU оптимизирована для работы с большим числом приложений. Кроме того, потенциал оптимизации CUDA-версий программы пока не исчерпан.

Таким образом, в нашем исследовании проводилось сравнение технологии OpenMP на 12 ядрах CPU и технологии NVIDIA CUDA на 14 потоковых мультипроцессорах. В таком ракурсе наши результаты с

учетом дальнейшего развития CUDA-версий следует интерпретировать как перспективные для графических процессоров.

Литература

1. *Priestley A.* A quasi-conservative version of the semi-Lagrangian advection scheme // Mon. Weather Rev. 1993. Vol. 121. P.621–629.
2. *Scroggs J.S.*, *Semazzi F.H.M.* A conservative semi-Lagrangian method for multidimensional fluid dynamics applications // Numer. Meth. Part. Diff. Eq. 1995. Vol. 11. P.445–452.
3. *Phillips T.N., Williams A. J.* Conservative semi-Lagrangian finite volume schemes // Numer. Meth. Part. Diff. Eq. 2001. Vol. 17. P.403–425.
4. *Iske A.* Conservative semi-Lagrangian advection on adaptive unstructured meshes // Numer. Meth. Part. Diff. Eq. 2004. Vol. 20. P.388–411.
5. *Годунов С.К.* Уравнения математической физики. М.: Наука, 1979. 392 с.
6. *Численное* решение многомерных задач газовой динамики / под ред. С.К. Годунова. М.: Наука, 1976. 400 с.
7. *Xin Wen, Vyatkin A.V., Shaidurov V.V.* Characteristics-like approach for solving hyperbolic equation of first order // Молодой учёный. 2013. № 3(50). С. 5–12.
8. *Xin Wen, Vyatkin A.V., Shaidurov V.V.* Semi-Lagrangian Scheme for soling hyperbolic equation of first order // Молодой учёный. 2013. № 9(56). С. 6–13.
9. *Вяткин А.В., Ефремов А.А., Карепова Е.Д., Шайдуров В.В.* Использование гибридных вычислительных систем для решения уравнения переноса модифицированным методом траекторий //Пятая Международная конференция «Системный анализ и информационные технологии»: Труды конференции. В 2 т. Красноярск: ИВМ СО РАН, 2013. Т. 1. С. 45–55.

Дискретный подход к задаче формосохраняющей интерполяции

Б. И. Квасов

Институт вычислительных технологий СО РАН,
Новосибирск

Рассматриваемый подход основан на формулировке задачи как дифференциальной многоточечной краевой задачи с ее последующей конечно-разностной аппроксимацией. Конструируются алгоритмы интерполяции точечных данных дискретными весовыми кубическими сплайнами, которые сохраняют монотонность и выпуклость данных. Проведенный анализ позволяет разработать два таких алгоритма с автоматическим выбором параметров формы (веса): один для сохранения монотонности данных и второй для сохранения выпуклости данных. Приведены результаты численных расчетов.

Постановка задачи

Пусть имеются данные

$$(x_i, f_i),\ i = 0, ..., N+1, \tag{1}$$

где $a = x_0 < x_1 < ... < x_{N+1} = b$. Положим

$$f[x_i, x_{i+1}] = (f_{i+1} - f_i)/h_i, \quad h_i = x_{i+1} - x_i, \quad i = 0, ..., N.$$

Данные (1) будем называть монотонными, если

$$f[x_i, x_{i+1}] \geq 0, \quad i = 0, ..., N,$$

и выпуклыми, если

$$\delta_i f = f[x_i, x_{i+1}] - f[x_{i-1}, x_i] \geq 0, \quad i = 1, ..., N.$$

Задача формосохраняющей интерполяции состоит в построении достаточно гладкой функции S такой, что $S(x_i) = f_i$, $i = 0, ..., N+1$ и S монотонна (выпукла) на участках монотонности (выпуклости) исходных данных.

Очевидно, что решение задачи формосохраняющей интерполяции неединственно. Будем искать его в виде весового кубического сплайна. Функцию $w(x)$ такую, что $0 < m \leq w(x) \leq M$, будем называть весовой функцией.

Определение 1. Весовым кубическим сплайном S называется решение дифференциальной многоточечной краевой задачи (сокращенно ДМКЗ):

$$\frac{d^2}{dx^2}(w(x)\frac{d^2S}{dx^2}) = 0, \quad x \in (x_i, x_{i+1}), \quad i = 0,...,N; \tag{2}$$

$$S \in C^k[a,b], \quad k \geq 1; \tag{3}$$

$$w(x_i^-)S''(x_i^-) = w(x_i^+)S''(x_i^+), \quad i = 1,...,N. \tag{4}$$

Будем считать, что кубический сплайн S удовлетворяет условиям интерполяции:

$$S(x_i) = f_i, \quad i = 0,...,N+1.$$

Для однозначного определения сплайна нам также потребуются краевые условия. Наиболее часто используемыми являются ограничения следующих типов:

- задание граничных значений первой производной: $S'(a) = f_0'$, $S'(b) = f_{N+1}'$;

- задание граничных значений второй производной: $S''(a) = f_0''$, $S''(b) = f_{N+1}''$;

Конечно-разностная аппроксимация

Рассмотрим теперь дискретизацию сформулированной ДМКЗ (2)–(4). Пусть задано $n_i \in \mathrm{IN}$, $i = 0,...,N$. Будем искать сеточную функцию

$$\{u_{ij},\ j = -1,...,n_i+1,\ i = 0,...,N\},$$

удовлетворяющую разностным уравнениям:

$$\Lambda_i(w(x_{ij})\Lambda_i u_{ij}) = 0, \quad j = 1,...,n_i-1, \quad i = 0,...,N, \tag{5}$$

где

$$\Lambda_i u_{ij} = \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{\tau_i^2}, \quad \tau_i = \frac{h_i}{n_i},$$

$$x_{ij} = x_i + j\tau_i, \quad j = 0,...,n_i.$$

Аппроксимация условий гладкости (3) и (4) дает соотношения

$$u_{i-1,n_{i-1}} = u_{i0},$$

$$\frac{u_{i-1,n_{i-1}+1} - u_{i-1,n_{i-1}-1}}{2\tau_{i-1}} = \frac{u_{i,1} - u_{i,-1}}{2\tau_i}, \quad i = 1,...,N, \tag{6}$$

$w(x_i^-)\Lambda_{i-1}u_{i-1,n_{i-1}} = w(x_i^+)\Lambda_i u_{i,0}$.

Условия интерполяции и краевые условия преобразуются к виду

$$u_{i,0} = f_i, \quad i = 0,...,N, \quad u_{N,n_N} = f_{N+1}, \tag{7}$$

$$\Lambda_0 u_{0,0} = f_0'', \quad \Lambda_N u_{N,n_N} = f_{N+1}''. \tag{8}$$

Соотношения (6)–(8) позволяют исключить "лишние" неизвестные $u_{i,-1}$ и u_{i,n_i+1}, $i = 0,...,N$. Дискретное сеточное решение будет определено как

$$\left\{u_{ij}, \quad j = 0,...,n_i, \; i = 0,...,N\right\}. \tag{9}$$

Вместо системы линейных уравнений с пятидиагональной матрицей (5)–(8) можно рассмотреть цепочку трехдиагональных систем. Введем обозначение:

$$M_{ij} = w(x_{ij})\Lambda_i u_{ij}, \quad j = 0,...,n_i, \quad i = 0,...,N. \tag{10}$$

Тогда на отрезке $[x_i, x_{i+1}]$ разностные уравнения (5) принимают вид:

$M_{i0} = M_i$,

$$\frac{M_{i,j-1} - 2M_{ij} + M_{i,j+1}}{\tau_i^2} = 0, \quad j = 1,...,n_i - 1, \tag{11}$$

$M_{i,n_i} = M_{i+1}$,

где M_i и M_{i+1} – заданные числа.

В силу соотношений (10) и с учетом условий интерполяции (7) имеем

$u_{i0} = f_i$,

$$\frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{\tau_i^2} = \frac{M_{ij}}{w(x_{ij})}, \quad j = 0,...,n_i, \tag{12}$$

$u_{i,n_i} = f_{i+1}$.

Матрицы систем (11) и (12) имеют диагональное преобладание. Это позволяет устойчиво решить эти системы методом трехточечной прогонки. В общем случае к системам (11), (12) надо добавить линейную систему для нахождения величин M_i, $i = 0,...,N+1$.

Пусть далее w – кусочно-постоянная функция такая, что $w(x) = w_i$ для $x \in [x_i, x_{i+1})$, $i = 0, ..., N$. Тогда продолжение сеточного решения системы (12) будет кубическим многочленом. На всем отрезке $[a, b]$ получаем дискретный весовой кубический сплайн, у которого в силу (6) будут непрерывны разделенные разности, но не производные.

Дискретный весовой кубический сплайн

Будем использовать обозначения:

$$M_i = w_{i-1}\Lambda_i S(x_i^-) = w_i \Lambda_i S(x_i^+), \quad i = 1, ..., N,$$

$$M_0 = w_0 \Lambda_0 S(x_0^+), \quad M_{N+1} = w_N \Lambda_N S(x_{N+1}^-).$$

Для $x \in [x_i, x_{i+1}]$ имеем

$$S(x) = f_i(1-t) + f_{i+1}t + M_i[\Phi_i(x) - \Phi_i(x_i)(1-t)] + \\ + M_{i+1}[\Psi_i(x) - \Psi_i(x_{i+1})t], \tag{13}$$

где $t = (x - x_i)/h_i$ и

$$\Phi_i(x) = \frac{1}{6w_i h_i}(x_{i+1} - x - \tau_i)(x_{i+1} - x)(x_{i+1} - x + \tau_i),$$

$$\Psi_i(x) = \frac{1}{6w_i h_i}(x - x_i + \tau_i)(x - x_i)(x - x_i - \tau_i). \tag{14}$$

Используя (13), (14) и второе из соотношений (6), получаем

$$A_i M_{i-1} + B_i M_i + C_i M_{i+1} = D_i, \quad i = 1, ..., N, \tag{15}$$

где

$$A_i = \mu_i(1 - \varepsilon_{i-1}^2), \quad B_i = 2 + \mu_i \varepsilon_{i-1}^2 + \lambda_i \varepsilon_i^2, \quad C_i = \lambda_i(1 - \varepsilon_i^2),$$

$$D_i = 6(w_{i-1}\mu_i + w_i \lambda_i) f[x_{i-1}, x_i, x_{i+1}], \quad \varepsilon_i = \tau_i / h_i \le 1/2,$$

$$\lambda_i = \frac{w_{i-1}h_i}{w_{i-1}h_i + w_i h_{i-1}}, \quad \mu_i = 1 - \lambda_i. \tag{16}$$

Предположим, что система (15) замыкается краевыми условиями типа II (очевидно, что могут быть использованы краевые условия и других типов). Тогда соотношения (15) дают линейную систему с диагональным преобладанием. Решение этой системы существует, единственно и может быть найдено методом трехточечной прогонки.

Используя обозначение $m_i = S[x_i - \tau_i, x_i + \tau_i]$, $i = 0, ..., N+1$ и третье из соотношений (6), имеем

$$a_i m_{i-1} + b_i m_i + c_i m_{i+1} = d_i, \quad i = 1,\ldots,N, \tag{17}$$

где

$$a_i = \lambda_i \frac{1-\varepsilon_{i-1}^2}{1+2\varepsilon_{i-1}^2}, \quad b_i = \lambda_i \frac{2+\varepsilon_{i-1}^3}{1+2\varepsilon_{i-1}^2} + \mu_i \frac{2+\varepsilon_i^2}{1+2\varepsilon_i^2}, \quad c_i = \mu_i \frac{1-\varepsilon_i^2}{1+2\varepsilon_i^2},$$

$$d_i = \frac{3\lambda_i}{1+2\varepsilon_{i-1}^2} f[x_{i-1}, x_i] + \frac{3\mu_i}{1+2\varepsilon_i^2} f[x_i, x_{i+1}].$$

Для замыкания системы (17) можно использовать краевые условия типа I. Соответствующая система линейных уравнений имеет диагональное преобладание, обеспечивающее существование и единственность дискретного весового кубического сплайна.

Условия монотонности и выпуклости

Беря линейные комбинации уравнений (15) и (17), замыкаемых с помощью краевых условий типов I и II, матрицы этих систем можно привести к матрицам монотонного типа [1. С. 152]. Это позволяет получить достаточные условия монотонности и выпуклости дискретного весового кубического сплайна.

Теорема 1. *Пусть дискретный весовой кубический сплайн S с краевыми условиями $S'(x_0) = f_0'$ и $S'(x_{N+1}) = f_{N+1}'$ интерполирует монотонные данные $\{f_i\}$, $i = 0,\ldots,N+1$. Если выполняются условия*

$$0 \le f_0' \le 3f[x_0, x_1]/(1-\varepsilon_0^2), \quad 0 \le f_{N+1}' \le 3f[x_N, x_{N+1}]/(1-\varepsilon_N^2), \tag{18}$$
$$\lambda_i(1-\varepsilon_i^2) f[x_{i-1}, x_i] \le [1+\lambda_i+(1+\mu_i)\varepsilon_{i-1}^2] f[x_i, x_{i+1}],$$

$$\mu_i(1-\varepsilon_{i-1}^2) f[x_i, x_{i+1}] \le [1+\mu_i+(1+\lambda_i)\varepsilon_i^2]) f[x_{i-1}, x_i], \quad i = 1,\ldots,N, \tag{19}$$

то $S[x-\tau_i, x+\tau_i] \ge 0$ *для всех* $x \in [x_i, x_{i+1}]$, $i = 0,\ldots,N$.

Рассмотрим теперь условия выпуклости весового кубического сплайна. Уравнения (15) с краевыми условиями типа II могут быть переписаны в виде

$$A_1 M_0 = D_0,$$
$$A_i M_{i-1} + B_i M_i + C_i M_{i+1} = D_i, \quad i = 1,\ldots,N, \tag{20}$$
$$C_N M_{N+1} = D_{N+1},$$

где правые части имеют вид

$$D_0 = A_1 w_0 f_0', \quad D_{N+1} = C_N w_N f_{N+1}', \quad D_i = \frac{6w_{i-1}w_i}{w_i h_{i-1} + w_{i-1} h_i} \delta_i f. \tag{21}$$

Теорема 2. *Пусть весовой кубический сплайн* S *с краевыми условиями*

$S''(x_0) = f_0''$ *и* $S''(x_{N+1}) = f_{N+1}''$ *интерполирует выпуклые данные* $\{f_i\}$, $i = 0,...,N+1$. *Если правые части системы* (20) *удовлетворяют неравенствам*

$$D_0 \geq 0, \quad D_{N+1} \geq 0, \quad D_i - \frac{A_i}{B_{i-1}} D_{i-1} - \frac{C_i}{B_{i+1}} D_{i+1} \geq 0, \quad i = 1,...,N, \tag{22}$$

то $\Lambda_i S(x) \geq 0$ *для всех* $x \in [x_i, x_{i+1}]$, $i = 0,...,N$.

Адаптивный выбор параметров формы

Предположим для определенности, что данные являются монотонно возрастающими: $f[x_i, x_{i+1}] \geq 0$, $i = 0,...,N$. Принимая во внимание формулы (16), неравенства (19) можно переписать в виде

$$\frac{w_{i-1}}{w_i} \frac{h_i}{h_{i-1}} \geq \frac{f[x_i, x_{i+1}]}{f[x_{i-1}, x_i]} - 2, \quad \frac{w_i}{w_{i-1}} \frac{h_{i-1}}{h_i} \geq \frac{f[x_{i-1}, x_i]}{f[x_i, x_{i+1}]} - 2, \quad i = 1,...,N. \tag{23}$$

Отсюда следует, что при выборе достаточно больших значений отношения w_{i-1} / w_i можно существенно уменьшить ограничения на данные, достаточные для получения монотонности весового кубического сплайна. Для обычного дискретного кубического сплайна имеем $w_{i-1} / w_i = 1$.

Отметим, что для каждого i одно из неравенств (23) выполняется. Пусть, например, выполняется неравенство $f[x_i, x_{i+1}] > f[x_{i-1}, x_i]$. Тогда второе неравенство в (23) выполнено. Первое неравенство может быть выполнено за счет выбора w_i непосредственно из этого неравенства. Предлагается следующий рекуррентный алгоритм, использующий формулу для начального задания весов (см. [4]):

$$w_i = [1 + C_i f[x_i, x_{i+1}]^2]^{-\beta_i}, \quad C_i \geq 1, \quad \beta_i \geq 0, \quad i = 0,...,N. \tag{24}$$

Алгоритм 1. Пусть на отрезке монотонности данных первый параметр w_{i-1} известен (обычно по формуле (24)). Если неравенства (23) выполняются для $w_i = w_{i-1}$, то полагаем $w_i = w_{i-1}$. В противном случае вычисляем w_i из того неравенства (23), которое не выполняется при $w_i = w_{i-1}$, путем замены знака неравенства на знак равенства. Если на

некотором шаге $w_i < \varepsilon$ ($w_i > \varepsilon^{-1}$), то полагаем $w_i = \varepsilon$ ($w_i = \varepsilon^{-1}$), где ε – достаточно малое положительное число. Алгоритм начинаем с задания w_0 и легко находим все параметры $\{w_i\}$, обеспечивающие монотонность дискретного весового кубического сплайна для любых монотонных данных.

Рассмотрим теперь алгоритм выбора параметров веса w_i для сохранения выпуклости данных. Будем предполагать, что данные являются выпуклыми, т. е. $\delta_i f > 0$, $i = 1,...,N$. Ограничения (22) будут выполнены, если удовлетворяются следующие неравенства:

$$D_0 \geq 0, \quad D_{N+1} \geq 0, \quad D_i - \frac{\mu_i}{2} D_{i-1} - \frac{\lambda_i}{2} D_{i+1}, \quad i = 1,...,N.$$

Последние неравенства можно усилить, расщепив их на две части. Имеем

$$D_i - \frac{\mu_i}{2} D_{i-1} - \frac{\lambda_i}{2} D_{i+1} = \mu_i (D_i - \frac{1}{2} D_{i-1}) + \lambda_i (D_i - \frac{1}{2} D_{i+1}), \quad i = 1,...,N.$$

Поэтому неравенства (22) выполняются, если выполнены следующие условия:

$$\begin{aligned} &0 \leq D_0 / 2 \leq D_1, \quad 0 \leq D_{N+1} \leq 2D_N, \\ &D_{i-1} / 2 \leq D_i \leq 2D_{i-1}, \quad i = 2,...,N. \end{aligned} \tag{25}$$

Принимая во внимание (21), нетрудно показать, что неравенства (25) будут выполнены, если выполняются следующие ограничения на краевые условия:

$0 < f_0'' < 6\delta_1 f / h_0, \quad 0 < f''_{N+1} < 6\delta_N f / h_N$

и на весовые параметры:

$$\begin{aligned} &\frac{w_0}{w_1} \frac{h_1}{h_0} \leq \frac{6\delta_1 f}{h_0 f_0''} - 1, \\ &\frac{1}{2\lambda_{i-1}} \frac{\delta_i f}{\delta_{i-1} f} \leq \frac{w_{i-1}}{w_i} \frac{h_i}{h_{i-1}} + 1 \leq \frac{2}{\lambda_{i-1}} \frac{\delta_i f}{\delta_{i-1} f}, \quad i = 2,...,N, \\ &\frac{w_N}{w_{N-1}} \frac{h_{N-1}}{h_N} \leq \frac{6\delta_N f}{h_N f''_{N+1}} - 1. \end{aligned} \tag{26}$$

Если положить $w_i = 1$ для всех i, то неравенства (26) принимают вид

$$\frac{1}{2} f[x_{i-1}, x_i, x_{i+1}] \leq f[x_{i-2}, x_{i-1}, x_i] \leq 2 f[x_{i-1}, x_i, x_{i+1}], \quad i = 2,...,N.$$

Дискретный кубический сплайн будет выпуклым, если выполняются эти неравенства. Однако это возможно только для слабо меняющихся данных.

Нетрудно видеть, что если $\delta_{i-1} f \delta_i f \le 0$, то неравенства (26) не могут быть выполнены ни для каких весов $w_i > 0$. По этой причине отделим участки выпуклости и вогнутости данных, где может быть использован алгоритм, аналогичный рассмотренному для монотонности (или аналогичный алгоритмам из [2, 5]). Предлагается следующий алгоритм для определения параметров веса w_i.

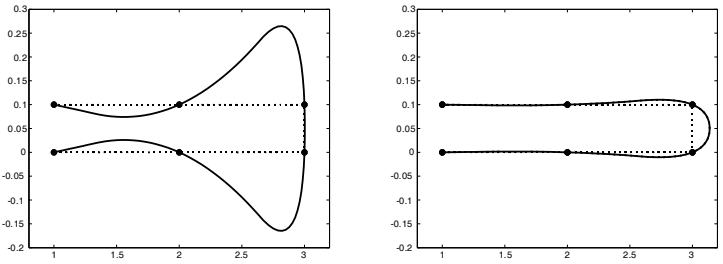
Алгоритм 2. Пусть на отрезке выпуклости данных первые два параметра w_{i-2} и w_{i-1} известны. Обычно они задаются по формуле (24). Если неравенства (26) выполняются для $w_i = w_{i-1}$, то полагаем $w_i = w_{i-1}$. В противном случае w_i находим из невыполненного неравенства (26) путем замены в нем знака неравенства на знак равенства. Для $w_i < \varepsilon$ ($w_i > \varepsilon^{-1}$) полагаем $w_i = \varepsilon$ ($w_i = \varepsilon^{-1}$), где ε – достаточно малое положительное число. Алгоритм начинаем с w_0 и w_1 и легко находим все параметры $\{w_i\}$, обеспечивающие выпуклость дискретного весового кубического сплайна для любых выпуклых данных.

Графические примеры

Цель первого примера состоит в рассмотрении эффектов параметров формы w_i при применении различных видов параметризации. Сплошная линия на рис. 1 соответствует графику дискретного кубического сплайн-интерполянта при использовании параметризаций по длине хорд (a) и единичной (b) с граничными условиями $(S_x''(t_i), S_y''(t_i)) = (0,0)$, $i = 1,6$ и исходными данными из табл. 1. Штриховые кривые на рис. 1 дают графики весовых кубических сплайн-интерполянтов для тех же данных при выборе весов по алгоритму 1 монотонной интерполяции и $\varepsilon = 0.0001$. Рис. 1 иллюстрирует тот факт, что форма интерполяционной кривой при использовании параметризации по длине хорд может не соответствовать замыслу пользователя. Весовой сплайн дает ожидаемые результаты.

Таблица 1. Данные для рис. 1

x_i	1	2	3	3	2	1
f_i	0	0	0	0,1	0,1	0,1



(a) (b)

Рис. 1. Параметрический дискретный кубический сплайн (сплошная кривая) и дискретный весовой сплайн (штриховая кривая) для одних и тех же данных: a – параметризация по суммарной длине хорд; b – единичная параметризация

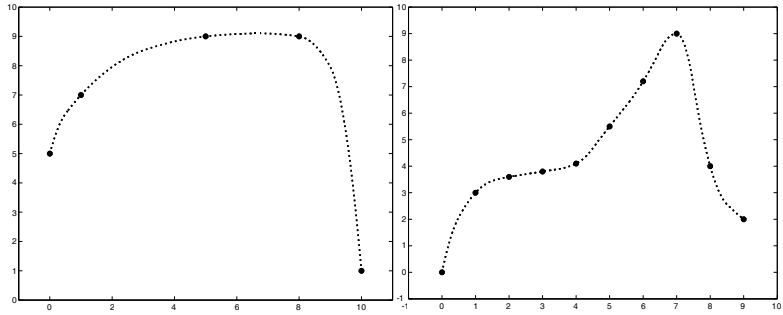
Алгоритм 2 автоматического выбора параметров веса для сохранения выпуклости данных был протестирован на двух примерах с данными в табл. 2 и 3 (см. [3]). Интерполяционные кривые показаны на рис. 2. Приведенные результаты позволяют сделать вывод, что предложенный метод выпуклой интерполяции дает визуально вполне удовлетворительные кривые. Метод прост в реализации и экономичен в вычислениях.

Таблица 2. Данные для рис. 2(a)

x_i	0	1	5	8	10
f_i	5	7	9	9	1

Таблица 3. Данные для рис. 2(b)

x_i	0	1	2	3	4	5	6	7	8	9
f_i	0	3	3,6	3,8	4,1	5,5	7,2	9	4	2



(a) (b)

Рис. 2. Интерполяция выпуклых данных по алгоритму 2 с автоматическим выбором параметров контроля формы: a – кривая по строго выпуклым данным; b – кривая по нестрого выпуклым данным

Автор выражает благодарность Российскому фонду фундаментальных исследований (код проекта 12-01-00160) за финансовую поддержку.

Литература

1. *Квасов Б.И.* Методы изогеометрической аппроксимации сплайнами. М.: Физматлит, 2006.

2. *Costantini P.* On monotone and convex spline interpolation // Math. Comp. 1986. V. 46. P. 203–214.

3. *Han X.* Convexity-preserving piecewise rational quartic interpolation // SIAM J. Numer. Anal. 2008. V. 46, N 2. P. 920–929.

4. *Salkauskas K.* C^1 splines for interpolation of rapidly varying data // Rocky Mountain Journal of Mathematics. 1984. V. 14, N 1. P. 239–250.

5. *Schmidt J.W., Hess W.* Schwach verkoppelte Ungleichungssysteme und konvexe Spline-Interpolation // Elem. Math. 1984. V. 39. P. 85–96.

Численный прогноз локальных метеорологических условий с использованием суперкомпьютера[1]

А.В. Старченко

Томский государственный университет, Томск
Институт мониторинга климатических и экологических систем СО РАН, Томск

Представлена мезомасштабная метеорологическая модель высокого разрешения для прогноза и исследования погодных явлений и качества приземного воздуха над ограниченной урбанизированной территорией, крупным промышленным или транспортным узлом. Для решения уравнений мезомасштабной модели разработан эффективный явно-неявный разностный метод второго порядка аппроксимации, ориентированный на суперкомпьютерную технику.

Физическая постановка задачи

Основные уравнения модели атмосферного пограничного слоя получаются для осредненных по Рейнольдсу дифференциальных уравнений гидротермодинамики в системе координат, связанной с поверхностью Земли, при следующих допущениях:

1. Мезомасштабные вариации плотности учитываются лишь при представлении силы плавучести в уравнении для вертикальной компоненты скорости (приближение Буссинеска).
2. Молекулярная диффузия полагается пренебрежимо малой по отношению к турбулентному обмену.
3. Учитываются фазовые переходы влаги в атмосферном пограничном слое, коротковолновый и длинноволновый радиационный теплообмен с явным представлением в атмосфере.

Математическая постановка задачи

В этом случае система уравнений мезомасштабной метеорологической модели примет вид [1]:

[1] Работа выполнена при поддержке РФФИ, грант № 12-01-00433а, и по заданию Министерства образования и науки РФ № 2014/233 (код проекта 2382).

Уравнение неразрывности

$$\frac{\partial(\rho u)}{\partial x}+\frac{\partial(\rho v)}{\partial y}+\frac{\partial(\rho w)}{\partial z}=0. \tag{1}$$

Уравнения движения

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)=-\frac{\partial p}{\partial x}+\rho f v+ \\ +\frac{\partial}{\partial x}\left(K_H\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_H\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_Z^m\frac{\partial u}{\partial z}\right); \tag{2}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)=-\frac{\partial p}{\partial y}-\rho f u+ \\ +\frac{\partial}{\partial x}\left(K_H\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_H\frac{\partial v}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_Z^m\frac{\partial v}{\partial z}\right); \tag{3}$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)=-\frac{\partial p}{\partial z}-\rho g+ \\ +\frac{\partial}{\partial x}\left(K_H\frac{\partial w}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_H\frac{\partial w}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_Z^m\frac{\partial w}{\partial z}\right). \tag{4}$$

Здесь t – время, u, v, w – продольная, поперечная и вертикальная компоненты вектора осредненной скорости ветра в направлении декартовых координат x, y, z соответственно, ρ – плотность, f – параметр Кориолиса, K_H – коэффициент горизонтальной диффузии, K_z^m – коэффициент вертикальной диффузии количества движения, g – ускорение свободного падения, p – давление.

Уравнение баланса энергии

$$\rho\left(\frac{\partial\theta}{\partial t}+u\frac{\partial\theta}{\partial x}+v\frac{\partial\theta}{\partial y}+w\frac{\partial\theta}{\partial z}\right)=\frac{\partial}{\partial x}\left(K_H\frac{\partial\theta}{\partial x}\right)+ \\ +\frac{\partial}{\partial y}\left(K_H\frac{\partial\theta}{\partial y}\right)+\frac{\partial}{\partial z}\left(K_Z^h\frac{\partial\theta}{\partial z}\right)+\frac{\theta}{c_pT}\left(Q_{rad}-\rho L_w\Phi_v\right). \tag{5}$$

Здесь T – абсолютная температура, θ – потенциальная температура, $\theta = T(p_0 / p)^{R/c_p}$, c_p – теплоемкость воздуха при постоянном давлении, p_0=101300 Н/м^2, R – газовая постоянная, Q_{rad} – нагрев (охлаждение) атмосферы за счет лучистых потоков тепла, распространяющихся в атмосфере, $\rho L_w \Phi_v$ – изменение температуры за счет фазовых переходов воды в атмосфере, K_z^h – коэффициент вертикальной диффузии тепла и влаги, L_w=2,501 МДж/кг – теплота парообразования.

Уравнения изменения удельной влажности воздуха, облачной и дождевой влаги

Предполагается, что в атмосфере влага может присутствовать только в виде водяного пара (q_v), облачной (q_c) и дождевой (q_r) воды. Процессы образования льда не учитываются, то есть используется микрофизика теплого дождя [2], которая включает объемные параметризации для перехода водяного пара в облачную влагу и обратно, захват облачной влаги дождевой, испарения дождевой влаги, а также выражение для скорости осаждения дождевых капель (рис. 1).

$$\frac{\partial(\rho q)}{\partial t} + \frac{\partial(\rho u q)}{\partial x} + \frac{\partial(\rho v q)}{\partial y} + \frac{\partial(\rho w q)}{\partial z} = \rho\Phi_v + \rho\Phi_c + \\ + \frac{\partial}{\partial x}\left(K_H \frac{\partial q}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_H \frac{\partial q}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z^h \frac{\partial q}{\partial z}\right); \quad (6)$$

$$\frac{\partial(\rho q_r)}{\partial t} + \frac{\partial(\rho u q_r)}{\partial x} + \frac{\partial(\rho v q_r)}{\partial y} + \frac{\partial(\rho w q_r)}{\partial z} = \rho\Phi_r - \frac{\partial(\rho V_r q_r)}{\partial z} + \\ + \frac{\partial}{\partial x}\left(K_H \frac{\partial q_r}{\partial x}\right) + \frac{\partial}{\partial y}\left(K_H \frac{\partial q_r}{\partial y}\right) + \frac{\partial}{\partial z}\left(K_z^h \frac{\partial q_r}{\partial z}\right). \quad (7)$$

Здесь суммарная удельная влажность $q = q_v + q_c$, где q_v – парообразная влага, q_c – облачная влага, q_r – дождевая влага; Φ_v, Φ_c и Φ_r – массовые скорости образования водяного пара, облачной и дождевой влаги соответственно; V_r – скорость осаждения дождевой воды. Для массовых скоростей образования различных видов влаги

справедливо соотношение баланса количества влаги в атмосфере: $\Phi_v + \Phi_c + \Phi_r = 0$.

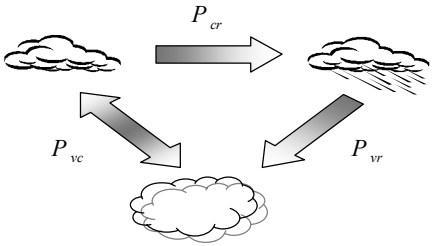


Рис.1. Схема взаимопереходов одних видов влаги в другие

Суммарная удельная влажность $q = q_v + q_c$ может быть разделена на составляющие следующим образом: $q_v = \min\left(q_{sat}, q\right)$, $q_c = \max\left(q - q_{sat}, 0\right)$, где q_{sat} – концентрация водяного пара в воздухе в насыщенном состоянии, выражающаяся как $q_{sat} = \dfrac{0,622 e_s}{p - 0,378 e_s}$, где $e_s = 610 \exp\left(L_w / R_v \left(1/273.15 + 1/T\right)\right)$, $R_v = 461,5$ *Дж/кг·К*.

Источниковые члены уравнений баланса влаги (6) – (7) – Φ_v, Φ_c и Φ_r, представляющие собой массовые скорости образования водяного пара, облачной и дождевой влаги соответственно, можно выразить следующим образом: $\Phi_v = -P_{vc} - P_{vr}$, $\Phi_c = P_{vc} - P_{cr}$, $\Phi_r = P_{cr} + P_{vr}$, где P_{vc} – массовая скорость перехода водяного пара в облачную влагу (конденсация) и обратно (испарение), P_{vr} $\left(\le 0\right)$ – массовая скорость испарения дождевых капель, P_{cr} $\left(\ge 0\right)$ – массовая скорость слипания облачных капель с образованием дождевых (автоконверсия) и захвата облачной влаги дождевой (аккреция) (рис. 1) [2].

$$P_{vc} = \frac{q_v - q_{sat}}{\Delta t} \Bigg/ \left(1 + \frac{L_w}{c_p} \frac{dq_{sat}}{dT}\right),$$

$$P_{cr} = \frac{0,057g}{\mu}\max\left(0;\, q_c - 0,0004\right)\left(\frac{\rho^4 q_c^7}{N_c \rho_{water}}\right)^{1/3} + 0,884 q_c q_r \left(\frac{g L_r \rho}{\rho_{water}}\right)^{1/2},$$

$$P_{vr} = \min\left(0, \frac{q_v}{q_{sat}} - 1\right)\frac{q_r L_r^2}{\rho_{water}}\left[\frac{0,5 + \frac{0,349}{\mu^{1/2}}\left(\frac{\rho_{water} g \rho}{L_r^3}\right)^{1/4}}{\frac{L_w^2}{\lambda_v R_v T^2} + \frac{R_v T}{e_s D_v}}\right],$$

здесь N_c – средняя концентрация облачных капель, значение которой принимается равным $3 \cdot 10^8\ м^{-3}$, ρ_{water} – плотность воды, μ – динамическая вязкость воздуха, L_r – параметр в функции распределения дождевых капель по размерам Маршала–Пальмера, D_v – температуропроводность водяного пара, λ_v – теплопроводность водяного пара. Скорость осаждения дождевой влаги определяется зависимостью $V_r = -2,13\left(\frac{g\rho_{water}}{L_r \rho}\right)^{1/2}$, а скорость осаждения дождя на поверхность определяется как $V_{precipitation} = -\frac{\rho_{surf}}{\rho_{water}} V_r q_{r(surf)}$.

Уравнение состояния

$$p = \rho R T, \quad R = R_0\left[\frac{1-q}{M_{air}} + \frac{q}{M_{H_2O}}\right]; \quad \theta = T(p_0 / p)^{R/c_p}. \qquad (8)$$

Для замыкания системы уравнений (1)–(8) используется двухпараметрическая модель турбулентности, состоящая из уравнений для кинетической энергии и масштаба турбулентности [3], а также алгебраических соотношений для определения коэффициентов турбулентной диффузии. Коэффициенты горизонтальной диффузии оцениваются по формуле Смагоринского.

Начальные условия для уравнений (1)–(8) задаются по результатам интерполирования метеорологических параметров, рассчитанных с использованием модели глобального масштаба.

Граничные условия на верхней границе расчетной области для уравнений (1) – (8) имеют следующий вид:

$$z = H:\ \frac{\partial u}{\partial z} = \frac{\partial v}{\partial z} = 0,\ \frac{\partial q}{\partial z} = \frac{\partial q_r}{\partial z} = 0,\ w = 0,\ \frac{\partial \theta}{\partial z} = \gamma\ .$$

На боковых границах используются вычислительные граничные условия «радиационного типа» [4], учитывающие тенденции в изменении мезомасштабных параметров по тенденциям параметров «ведущей» модели прогноза погоды глобального масштаба.

На подстилающей поверхности ставятся условия, соответствующие основным соотношениям теории подобия Монина–Обухова [5]. Согласно этой теории значения потоков динамических, термодинамических и турбулентных параметров определяются в приземном слое атмосферы с использованием масштаба длины Обухова: $L = -\dfrac{\theta V_*^3}{g\kappa\left(\overline{\theta' w'}\right)_w}$. При определении температуры поверхности Земли θ_w решается одномерное нестационарное уравнение теплопроводности и используется поток тепла через поверхность, обусловленный радиационным нагревом (выхолаживанием) и турбулентными потоками тепла и влаги на поверхности; также учитывается температура нижних слоев почвы, изменениями во времени которой при мезомасштабном моделировании можно пренебречь (глубина ≈ 2 м, период моделирования несколько суток).

Численный метод

Для системы уравнений (1)–(8) перед ее численным решением методом сеток применялось преобразование координат вида:

$$\begin{cases} \xi = x \\ \eta = y \\ \sigma = H\dfrac{z - h(x,y)}{H - h(x,y)}. \end{cases} \qquad (9)$$

Здесь H – высота области исследования, $h(x,y)$ – высота рельефа подстилающей поверхности над уровнем моря. Преобразование (9) позволяет отобразить трехмерную область с криволинейной границей на параллелепипед.

Аппроксимация построенной дифференциальной задачи с выполненным преобразованием (9) осуществляется на основе метода

конечного объема и явно-неявных разностных схем. Основная идея этого подхода заключается в разбиении расчетной области на непересекающиеся, граничащие друг с другом конечные объемы так, чтобы один узел расчетной сетки (*i,j,k*) содержался только в своем конечном объеме. Для дискретизации используется равномерная по горизонтальным направлениям сетка, допускающая сгущение сеточных плоскостей при приближении к поверхности Земли.

Разбив таким образом расчетную область, интегрируем каждое дифференциальное уравнение математической модели по каждому конечному объему. Значения компонент скорости определяются на гранях конечных объёмов, а скалярные характеристики – в их центре. При вычислении интегралов используются кусочно-полиномиальные приближения для зависимых величин. Аппроксимация конвективных членов уравнений переноса выполняется с использованием противопотоковой схемы MLU Ван Лира [6]. При дискретизации по времени используется явно-неявная разностная схема, в которой только диффузионные слагаемые по вертикальной координате $\frac{\partial}{\partial z}\left(\Gamma_z \frac{\partial \Phi}{\partial z}\right)$ аппроксимируются по схеме Кранка–Николсон, а остальные члены уравнений – по схеме Адамса–Бэшфорда.

В результате такого приближенного интегрирования получается дискретный аналог системы дифференциальных уравнений – система линейных алгебраических уравнений вида

$$
\begin{aligned}
&-c^{m+1}_{i,j,k}\Phi^{m+1}_{i,j,k-1} + e^{m+1}_{i,j,k}\Phi^{m+1}_{i,j,k} - d^{m+1}_{i,j,k}\Phi^{m+1}_{i,j,k+1} = ap^{0}_{i,j,k}\Phi^{m}_{i,j,k} + \\
&+\frac{3}{2}(ap^{m}_{i,j,k}\Phi^{m}_{i,j,k} + ae^{m}_{i,j,k}\Phi^{m}_{i+1,j,k} + an^{m}_{i,j,k}\Phi^{m}_{i,j+1,k} + at^{m}_{i,j,k}\Phi^{m}_{i,j,k+1} + \\
&\quad + aw^{m}_{i,j,k}\Phi^{m}_{i-1,j,k} + as^{m}_{i,j,k}\Phi^{m}_{i,j-1,k} + ab^{m}_{i,j,k}\Phi^{m}_{i,j,k-1} + b^{m}_{i,j,k}) - \\
&-\frac{1}{2}(ap^{m-1}_{i,j,k}\Phi^{m-1}_{i,j,k} + ae^{m-1}_{i,j,k}\Phi^{m-1}_{i+1,j,k} + an^{m-1}_{i,j,k}\Phi^{m-1}_{i,j+1,k} + at^{m-1}_{i,j,k}\Phi^{m-1}_{i,j,k+1} + \\
&\quad + aw^{m-1}_{i,j,k}\Phi^{m-1}_{i-1,j,k} + as^{m-1}_{i,j,k}\Phi^{m-1}_{i,j-1,k} + ab^{m-1}_{i,j,k}\Phi^{m-1}_{i,j,k-1} + b^{m-1}_{i,j,k}) + \\
&\quad + c^{m}_{i,j,k}\left(\Phi^{m}_{i,j,k-1} - \Phi^{m}_{i,j,k}\right) + d^{m}_{i,j,k}\left(\Phi^{m}_{i,j,k+1} - \Phi^{m}_{i,j,k}\right).
\end{aligned}
\qquad (10)
$$

Здесь $\left\{\Phi^{m+1}_{i,j,k}\right\}$ – неизвестная скалярная сеточная функция, *m* – номер временного слоя, со строчных букв начинаются обозначения для коэффициентов разностной схемы. Системы такого вида являются

условно устойчивыми с ограничением на шаг по времени и решаются экономичным методом прогонки вдоль вертикальных сеточных линий.

Для согласования векторного поля скорости и давления на каждом шаге используется схема «предиктор-корректор», в которой сначала предсказываются с использованием разностных схем вида (10) значения компонент вектора скорости при давлении на m-м слое по времени p_h^m, затем после решения разностного уравнения эллиптического типа для $p'_h = p_h^{m+1} - p_h^m$ производится коррекция промежуточных полей скорости с требованием, чтобы скорректированные значения компонент скорости удовлетворяли бы точно разностному уравнению неразрывности.

Параллельный алгоритм

В качестве основного подхода распараллеливания выбрана геометрическая декомпозиция сеточной области на подобласти: каждому процессорному элементу вместе с выделенной сеточной подобластью распределяются все значения сеточной функции $\Phi_{i,j,k}^m$, принадлежащие этой подобласти [7].

После этапа декомпозиции, когда производится разделение данных на блоки для построения параллельного алгоритма, переходим к этапу установления связей между блоками, вычисления в которых будут выполняться параллельно, – планированию коммуникаций. Из-за используемого шаблона явно-неявной разностной схемы для вычисления очередного приближения в приграничных узлах каждой подобласти необходимо знать значения сеточной функции с соседнего граничащего процессорного элемента. Для этого на каждом вычислительном узле создаются фиктивные ячейки для хранения данных с соседнего вычислительного узла и организуются пересылки этих граничных значений, необходимых для обеспечения однородности вычислений [7]. Для передачи данных другим процессорным элементам и получения необходимых для продолжения вычислений данных от них в нашей работе используется стандарт передачи сообщений MPI (Message Passing Interface).

В данной работе для решения разностного уравнения (10) используется метод прогонки, для разностного уравнения для поправки давления p'_h применяется полинейный метод Зейделя с красно-черным упорядочиванием узлов вычислительной сетки [7, 8], параллельная реализация которого при проведении расчетов показывает независимость скорости сходимости итерационного процесса от

количества применяемых процессорных элементов. Важно, что такая реализация алгоритма на многопроцессорной вычислительной системе целиком сохраняет свойство последовательного алгоритма и очень хорошо масштабируется на любое разумное количество вычислительных узлов.

На рис. 2 представлены графики ускорения и эффективности параллельной программы при проведении расчетов по мезомасштабной метеорологической модели для внутренней области на сетке 50x50x30 узлов в течение двух суток моделирования.

Расчеты проводились на вычислительном кластере ТГУ СКИФ Cyberia. Из рис. 2 видно, что разработанный параллельный алгоритм для умеренного количества процессорных элементов (до 25) показывает хорошую масштабируемость и высокую эффективность. Полученные результаты ускорения параллельной программы обеспечивают выполнение требования проведения прогностических расчетов при исследовании и заблаговременном предсказании погоды: получить прогноз на ближайшие сутки за 1 час времени работы программы на суперкомпьютере.

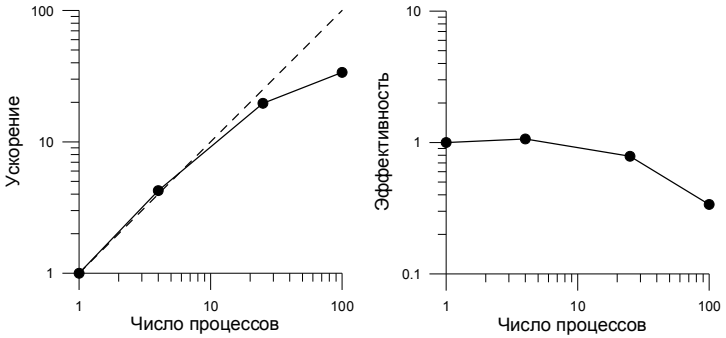


Рис. 2. Ускорение и эффективность работы параллельной программы по расчету мезомасштабных метеорологических процессов над аэропортом

Некоторые результаты расчетов

Разработанная мезомасштабная модель и измерительный комплекс были применены к исследованию метеорологических условий над городом Томск (85,0°E 56,5°N, центр области рис.3) и его аэропортом Богашево (85,21°E 56,38°N).

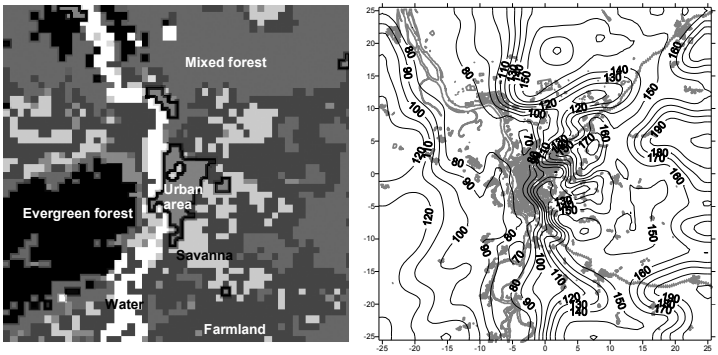


Рис. 3. Область моделирования: слева – распределение категорий землепользования, справа – рельеф подстилающей поверхности

В метеорологической модели использовались горизонтальное разрешение 1 км и 30 уровней по вертикальной координате. При задании свойств подстилающей поверхности рассматривалось семь категорий землепользования: водная поверхность, поверхность с незначительной растительностью, сельскохозяйственные угодья, лиственный, смешанный и хвойный лес, городская застройка. Эти категории отличались по следующим параметрам: высота шероховатости, альбедо, теплофизические свойства почвы, параметр испарения, степень черноты, температура на глубине 2 м (для почвы).

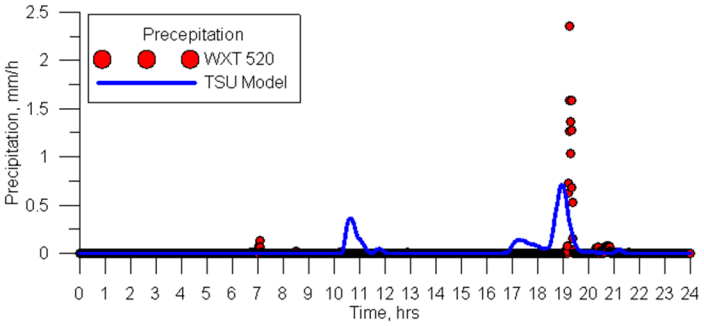


Рис. 4. Интенсивность дождевых осадков, выпавших во время грозы 27.08.2012 в районе аэропорта Богашево

На рис. 4 представлены графики интенсивности дождевых осадков, выпавших во время грозы 27 августа 2012 г. в районе аэропорта

Богашево. Измерения интенсивности осадков (мм/ч) проводились с помощью автоматической метеостанции WXT520. В соответствии с информацией, представленной на сайте http://meteo.infospace.ru, 27 августа 2012 г. в течение всего дня наблюдалась облачность 10 баллов, штилевые условия, температура воздуха в течение суток изменялась от 11 до 16 градусов Цельсия. В 18 часов местного времени в городе Томск была зафиксирована гроза, которую предсказала развиваемая мезомасштабная модель (рис.4).

Литература

1. *Старченко А.В.* Численное исследование локальных атмосферных процессов // Вычислительные технологии, 2005. Т.10. С. 81–89.
2. *Hurley P.J.* The Air Pollution Model (TAPM) Version 2 / CSIRO Atmospheric Research Technical Paper №55, 2002.
3. *Старченко А.В.* Моделирование переноса примеси в однородном атмосферном пограничном слое // Труды Международной конференции ENVIROMIS-2000. Томск: Изд-во Том. ЦНТИ, 2000. С. 77–82.
4. *Klemp J.*, *Wilhelmson R.* The simulation of three-dimensional convective storm dynamics // Journal of Atmospheric Sciences. 1978. V.35. P. 1070–1096.
5. *Монин А.С., Обухов А.М.* Основные закономерности турбулентного перемешивания в приземном слое атмосферы // Труды Геофизического института АН СССР. 1954. №24. С. 163–187.
6. *Van Leer B.* Towards the ultimate conservative difference scheme. II. Monotonicity and conservation combined in a second order scheme // J. of Computational Physics. 1974. Vol. 14. P. 361–370.
7. *Старченко А.В., Берцун В.Н.* Методы параллельных вычислений. Томск: Изд-во Том. ун-та, 2013. 234 с.
8. *Старченко А.В., Деги Д.В.* Численное моделирование локальных атмосферных процессов с использованием многопроцессорных вычислительных систем // Научный сервис в сети Интернет: поиск новых решений: Труды Международной суперкомпьютерной конференции (17-22 сентября 2012 г., г. Новороссийск). М.: Изд-во МГУ, 2012. С. 536–541.

Построение таблицы переходов клеточного автомата, моделирующего волновой процесс

В.П. Маркова

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

Рассматривается класс клеточных автоматов HPPrp, эволюция которых моделирует волновой процесс. Каждая клетка автомата содержит частицы двух типов: движущиеся частицы и частицы покоя. Взаимодействие частиц на этапе столкновения определяет функционирование недетерминированного конечного автомата, который реализует клетки HPPrp автомата. Предложена процедура построения правил перехода. Она сформулирована в терминах задачи линейного программирования. В качестве условия, которое гарантирует существование равновесного состояния, используется условие обобщенного детального баланса.

Введение

Lattice-Gas Cellular Automata [1–3] («решеточные газы», далее просто клеточные автоматы (КА)) описывают природные явления множеством гипотетических частиц. Они движутся в решеточном дискретном пространстве по некоторым правилам. Эти правила представляют моделируемый процесс на мезоуровне, исходя из общих законов физики. В [2] доказано, что они соответствуют волновому уравнению. Интерес к КА-моделированию объясняется рядом достоинств. Во-первых, отсутствие ошибок округления и способность моделировать нелинейные и разрывные процессы. Во-вторых, простота задания граничных условий. И, наконец, неограниченные возможности параллельной реализации задач на современных суперкомпьютерах.

Первым решеточным автоматом был так называемый HPP-автомат [2]. Он определен на квадратной решетке. Каждый узел решетки (клетка) имеет четырех соседей. В данный момент времени в клетке может находиться не более одной движущейся частицы с единичной массой и единичной скоростью, направленной в сторону соседнего узла. Состояние клетки однозначно определяется булевым вектором. Эволюция HPP-автомата состоит из двух последовательных фаз: столкновение и сдвиг. Столкновение реализуется одним правилом (лобовое столкновение): две частицы прилетают в клетку из противоположных направлений, сталкиваются и меняют направление

вектора скорости на 90 градусов. На фазе сдвига все частицы сдвигаются в сторону ближайшего соседа. Все клетки HPP-автомата вычисляют новое состояние синхронно и параллельно, в результате чего происходит изменение глобального состояния автомата. Итеративная смена глобальных состояний HPP-автомата описывает динамику волнового процесса. Каждой клетке HPP-автомата ставится в соответствие конечный автомат, состояние которого определяется находящимися в нем частицами. Таблица переходов автомата описывает его работу на этапе столкновения.

В [2] предложен клеточный автомат HPPrp (rest particles) для моделирования волновых процессов в неоднородных средах. Этот автомат, кроме движущихся частиц, содержит частицы покоя. Они имеют различную массу и нулевую скорость. В результате в HPPrp-клетке к правилу лобового столкновения движущихся частиц добавились правила создания (разрушения) частиц покоя с некоторой вероятностью. Формальная процедура построения недетерминированных правил перехода конечного автомата для разработки новых HPPrp-автоматов с произвольным количеством частиц покоя отсутствует. Более того, динамика частиц в существующих HPPrp-автоматах не всегда удовлетворяет необходимому условию полудетального баланса. Поэтому встает задача поиска более общего условия, которое гарантирует существование равновесного состояния клеточного автомата по сравнению с полудетальным балансом, и разработки на его основе процедуры построения таблицы переходов конечного автомата, реализующего этап столкновения в HPPrp-автоматах.

HPPrp модель волновых процессов

Формально HPPrp клеточный автомат определяется множеством $N = \langle M, S, \Theta \rangle$, где M – дискретное пространство, S – множество состояний клеток, Θ – множество функций переходов. Дискретное пространство представляет собой $2D$ решетку: $M = \{(i, j) : i = 0,1,...,I, j = 0,1,...,J\}$, где пара $(i, j) = r -$ *имя* клетки. Каждая клетка имеет четыре соседа и может содержать не более четырех *движущихся* частиц и несколько частиц *покоя*. Движущиеся частицы имеют единичную массу и единичную скорость, частицы покоя имеют разную массу (2, 4, 8, …) и нулевую скорость.

Множество состояний – множество булевых векторов $\boldsymbol{s} = (s_{b+4},...,s_4,s_3,s_2,s_1)$, где b – количество частиц покоя. Вектор $\boldsymbol{s}$ содержит информацию о частицах, находящихся в данный момент времени в данной клетке. Значения первых четырех компонент вектора $\boldsymbol{s}$ характеризуют движущиеся частицы: $s_k = 0$ (отсутствие), $s_k = 1$ (наличие) движущейся частицы, направленной в сторону *k*-го соседа. Значения остальных компонент вектора $\boldsymbol{s}$ характеризуют частицы покоя: $s_k = 0$ (отсутствие), $s_k = 1$ (присутствие) частицы покоя с массой 2^{k-4}, $k = 5,6,...,b+4$. Например, если клетка имеет состояние (11011), это означает, что она содержит три движущиеся частицы, направленные в сторону первого, второго и четвертого соседей, и одну частицу покоя массой 4. Пара (s,r) называется *клеткой*. Сумма масс всех частиц в клетке с именем r называется *модельной плотностью*. Множество клеток, в котором все клетки имеют уникальные имена, образует *клеточный массив* $\Omega = \{(s,r)\}$. Множество состояний всех клеток массива в момент времени *t* называется *глобальным* состоянием $\Omega(t)$ клеточного массива.

Множество функций переходов $\Theta = \{\theta_p\}$, $p = 1,2,...,g$ определяется правилами функционирования клеточного автомата. Автомат работает синхронно. Итерационный шаг состоит из двух фаз: *столкновения* и *сдвига*. Это означает, что функция перехода состоит из композиции функций столкновения и сдвига.

Столкновение выполняется локально для каждой клетки массива. Фаза столкновения состоит из следующих правил: лобовое столкновение и правила создания (разрушения) частиц покоя с некоторой вероятностью. Сдвиг выполняется перемещением движущихся частиц на одну клетку в направлении ее соседей. Правила сдвига и столкновения строятся таким образом, чтобы при каждом сдвиге и столкновении частиц суммарные масса и импульс всех частиц клеточного массива оставались неизменными.

На рис. 1 показан итерационный шаг HPPrp КА с двумя частицами покоя массой 2 и 4. В результате столкновения клетка из исходного состояния $\boldsymbol{s} = (010101)\,(21)$ (21 – десятичное представление состояния (010101)) меняется на состояния $\boldsymbol{s}'_1 = (011010)\,(26)$, $\boldsymbol{s}'_2 = (001111)(15)$, $\boldsymbol{s}'_3 = (100000)(32)$ и $\boldsymbol{s}'_4 = \boldsymbol{s}$ в общем случае с разными вероятностями, но при соблюдении равенства $\sum_1^4 p_i = 1$ (условия нормализации).

Состояния 26, 21, 15, 32 образуют *класс эквивалентности* Ψ_4, нижний индекс указывает модельную плотность клеток класса.

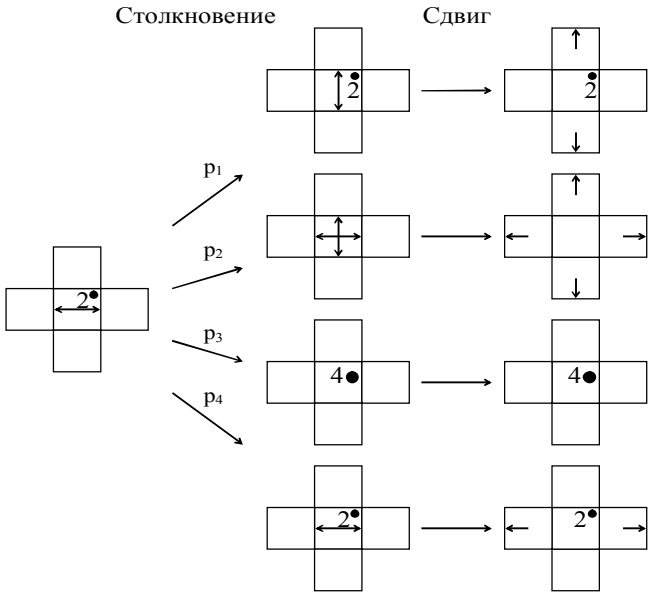


Рис.1. Итерационный шаг HPPrp с частицами покоя массой 2 и 4

В результате выполнения итерационного шага HPPrp автомат из одного глобального состояния переходит в другое глобальное состояние. Итеративная смена глобальных состояний клеточного автомата (*эволюция* клеточного автомата) описывает динамику волнового процесса.

Пример 1. Моделирование HPPrp-волнового процесса [4]. В эксперименте волновой процесс представляется эволюцией HPPrp КА с частицами покоя массой 2 и 4. Размер массива составляет 600x300 клеток (рис. 2). Источник генерации внешнего возмущения расположен в центре и представляет собой массив размером 20x300 клеток. Исходное состояние клеток массива (модельная плотность) задается генератором случайных чисел в течение одного такта. Модельная плотность источника возмущения равна 4: генератор с вероятностью 1 генерирует движущиеся частицы. Модельная плотность клеток остальной части массива равна 2: генератор с вероятностью 0,25

генерирует движущиеся частицы, с вероятностями 0,5 (0,25) генерирует частицы покоя с массой 2 (4). Граничные условия – периодические. В результате эволюции HPPгр КА в массиве образуются две плотные области, которые движутся от источника в противоположные стороны с конечной скоростью, имитируя распространение одиночной продольной волны.

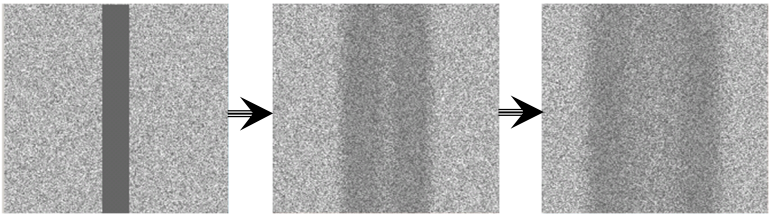


Рис.2. Эволюция клеточного автомата на разных итерациях

Для того чтобы наблюдать волновой процесс в привычном представлении физического явления, глобальные состояния КА в определенные моменты времени усредняются. На рис. 3 показаны профили одиночной КА-волны, полученные осреднением трех глобальных состояний клеточного массива на рис. 2. (Здесь в качестве области осреднения используется столбец клеточного массива.) Из рис. 3 видно, что профиль КА-волны со временем размывается, при этом суммарные масса и импульс всех частиц клеточного массива сохраняются. Такое явление указывает на то, что правила взаимодействия частиц, кроме процесса распространения волны, моделируют диффузионный процесс.

Построение таблицы переходов конечного автомата

Каждой клетке массива поставлен в соответствие конечный автомат: входами автомата являются четыре соседние клетки, выходом – текущее состояние. Новое состояние конечного автомата определяется по функции перехода. Таблица переходов конечного автомата представляет собой матрицу $\boldsymbol{P}$ размером $2^{4+b} \times 2^{4+b}$. Элементом таблицы является вероятность перехода $p_{ij} = p(\boldsymbol{s} \to \boldsymbol{s}')$ клетки из состояния $\boldsymbol{s} = (s_{b+4}, ..., s_4, s_3, s_2, s_1)$ в состояние $\boldsymbol{s}' = (s'_{b+4}, ..., s'_4, s'_3, s'_2, s'_1)$ (i и j десятичное представление двоичных векторов $\boldsymbol{s}$ и $\boldsymbol{s}'$).

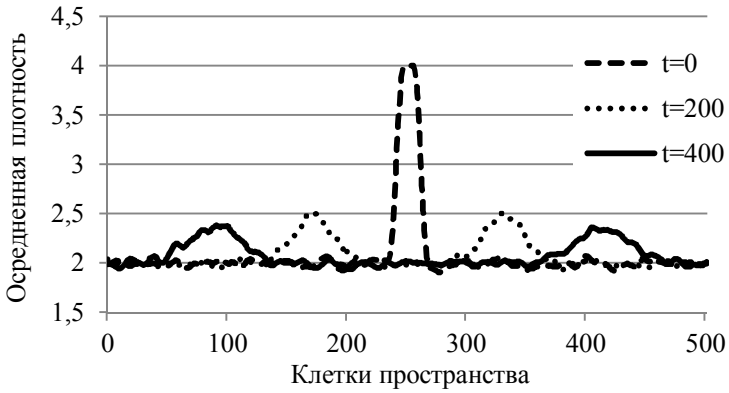


Рис.3. Изменение профиля КА-волны на разных итерациях

Элементы таблицы переходов должны удовлетворять следующим свойствам:

- закону сохранения массы и импульса,
- условию нормализации: $\forall \boldsymbol{s} \sum_{\boldsymbol{s}'} p(\boldsymbol{s} \to \boldsymbol{s}') = 1$ **,**
- детальному балансу: $\forall \boldsymbol{s}, \boldsymbol{s}'\ p(\boldsymbol{s} \to \boldsymbol{s}') = p(\boldsymbol{s}' \to \boldsymbol{s})$,
- полудетальному балансу: $\forall \boldsymbol{s}, \boldsymbol{s}' \sum_{\boldsymbol{s}} p(\boldsymbol{s} \to \boldsymbol{s}') = 1$ **.**

Для моделей с частицами покоя, где правила переходов являются недетерминированными, полудетальный баланс удовлетворяется не всегда. В [4] предложены условия, которые гарантируют существование равновесного состояния. Здесь будем использовать условие обобщенного детального баланса:

$$\forall \boldsymbol{s}, \boldsymbol{s}' \quad r(s)\, p(\boldsymbol{s} \to \boldsymbol{s}') = r(\boldsymbol{s}')\, p(\mathbf{s}' \to \mathbf{s}),$$

$$\forall \boldsymbol{s} \quad r(\boldsymbol{s}) = \prod_{k=1}^{b+4} F_k^{-s_k},$$

где F_k может быть любой положительной ограниченной функцией, которая не меняется после столкновения. Пусть $F_k = e^{-a_k}$, где a_k – произвольные числа, характеризующие движущиеся частицы и частицы покоя, и пусть $a_1 = a_2 = a_3 = a_4 = a$.

Основная идея процедуры формирования таблицы переходов сводится к составлению и решению задачи линейного

программирования. Переменными для задачи являются элементы таблицы переходов, ограничениями – свойства, которым должны удовлетворять элементы таблицы и набор чисел $a_1, a_{2,}, a_3, a_4, a_5, \ ..., a_{4+b}$.

Выбор целевой функции в рамках клеточных автоматов – задача довольно сложная, но будет разумным минимизировать (максимизировать) вероятность клетки остаться в прежнем состоянии после столкновения.

Пример 2. Построение таблицы переходов для HPPrp с двумя частицами покоя (2, 4). Поскольку классы эквивалентности состояний КА не пересекаются, то построение таблицы переходов конечного автомата можно свести к построению таблиц переходов для каждого класса эквивалентности. Сформулируем задачу линейного программирования для состояний из класса (15, 21, 26, 32).

1. $f = p(15 \to 15) + p(21 \to 21) + p(26 \to 26) + p(32 \to 32) \to \min$
2. Набор чисел: $a = 0.25, a_5 = 0.9, a_6 = 0.4.$
3. Нормализация:

$$p(15 \to 15) + p(15 \to 21) + p(15 \to 26) + p(15 \to 32) = 1,$$
$$p(21 \to 15) + p(21 \to 21) + p(21 \to 26) + p(21 \to 32) = 1,$$
$$p(26 \to 15) + p(26 \to 21) + p(26 \to 26) + p(26 \to 32) = 1,$$
$$p(32 \to 15) + p(32 \to 21) + p(32 \to 26) + p(32 \to 32) = 1.$$

4. Обобщенный детальный баланс:

$$p(15 \to 21) - e^{a_5 - 2a} p(21 \to 15) = 0,$$
$$p(15 \to 26) - e^{a_5 - 2a} p(26 \to 15) = 0,$$
$$p(15 \to 32) - e^{a_5 - 4a} p(32 \to 15) = 0,$$
$$p(21 \to 26) - p(26 \to 21) = 0,$$
$$p(21 \to 32) - e^{a_6 - a_5 - 2a} p(32 \to 21) = 0,$$
$$p(26 \to 32) - e^{a_6 - a_5 - 2a} p(32 \to 26) = 0.$$

В результате решения задачи линейного программирования таблица переходов для рассматриваемого класса эквивалентности состояний имеет следующий вид:

Таблица переходов для класса эквивалентности (15, 21, 26, 32)

	15	**21**	**26**	**32**
15	0	0,25	0	0,75
21	0,11	0	0	0,89
26	0	0	0	1
32	0,15	0,4	0,45	0

Заключение

В работе предложена формальная процедура построения правил переходов конечного автомата, реализующего HPPrp клеточного автомата. Процедура сформулирована в терминах задачи линейного программирования. В качестве условия, которое гарантирует существование равновесного состояния, используется условие обобщенного детального баланса. В дальнейшем предполагается установить соответствие между параметрами функции $F_k = e^{-a_k}$ и характеристиками волнового процесса.

Литература

1. *Dieter A.* Wolf-Gladrow. Lattice-Gas Cellular Automata and Lattice Boltzmann Models – An Introduction. Springer, 2005. 308 p.

2. *Hardy J., Pomean Y., de Pazzis O.* Time evolution of a two-dimensional model system // Journal of Math. Physics. 1973. Vol.14. P.1746–1759.

3. *Zhang M., Cule D., Shafai L., Bridges G. and Simons N.* Computing Electromagnetic Fields in Inhomogeneous Media Using Lattice Gas Automata // Proceedings of 1998 Symposium on Antenna Technology and Applied Electromagnetics, Aug.14–16, Ottawa, Canada.

4. *Chen H.* H-Theorem and Generalized Semi-Detailed Balance Condition for Lattice Gas Systems // J. of Statistical Physics. 1995. V.81. P. 347–357.

Эффективность параллельной реализации метода конечных элементов для задачи распространения поверхностных волн*

Е.В. Дементьева[1], Е. Д. Карепова[1,2]

[1] Институт вычислительного моделирования СО РАН,
[2] Институт математики и фундаментальной информатики СФУ,
Красноярск

Проведено исследование эффективности нескольких параллельных реализаций метода конечных элементов для алгоритма численного решения начально-краевой задачи для уравнений мелкой воды, выполненных с помощью технологий MPI, OpenMP и MPI+OpenMP.

Введение

Численное моделирование поверхностных волн в больших акваториях проводилось с учетом сферичности Земли и ускорения Кориолиса на основе уравнений мелкой воды [1]. В [2] для этой задачи построен метод конечных элементов (МКЭ), и дифференциальная задача сведена к векторно-матричной форме. Полученная система линейных алгебраических уравнений решается итерационным методом Якоби, который обладает хорошим параллелизмом, а диагональное преобладание для его сходимости обеспечивается выбором шага по времени [3].

Использование МКЭ при пространственной дискретизации задачи дает ряд преимуществ, основное из которых – возможность использовать неструктурированные, неравномерные сетки для вычислительных областей сложной формы. В то же время МКЭ, безусловно, имеет большую вычислительную сложность [4] по сравнению с методом конечных разностей, что делает актуальным использование высокопроизводительных вычислительных систем для его реализации [5–11].

В настоящей работе для начально-краевой задачи для уравнений мелкой воды проводится исследование эффективности нескольких

* Работа выполнялась в рамках гранта РФФИ № 11-01-00224-а, интеграционного проекта СО РАН № 130, проекта Президиума РАН № 18.2.

параллельных реализаций метода конечных элементов, выполненных с помощью технологий MPI, OpenMP и MPI+OpenMP.

Параллельная реализация на SMP-узловом кластере

Для численного моделирования распространения длинных волн в большой акватории использовался метод конечных элементов с линейными треугольными конечными элементами [2].

Для исследования ускорения нескольких параллельных реализаций алгоритма рассматривалась модельная задача для «квадрата» на сфере с твердыми границами [12, 13]. В расчетной области использовалась равномерная квадратная сетка с согласованной триангуляцией. В обсуждаемых вычислительных экспериментах использовалась сетка 1600×1600 шагов по пространству, и делалось 200 шагов по времени. В нашем случае таких параметров сетки было достаточно для хорошей масштабируемости задачи при использовании до ста вычислительных устройств.

Самой вычислительно трудоемкой операцией в МКЭ является сборка невязки на основе локальных матриц жесткости элементов. Существует, по крайней мере, два способа обхода триангуляции при сборке невязки:

1) по элементам (традиционный способ, реализующий наиболее выгодное распределение памяти при хранении информации о триангуляции);

2) по точкам сетки (требует размещения в памяти дополнительных, в общем случае, нерегулярных структур, отвечающих за хранение информации о триангуляции).

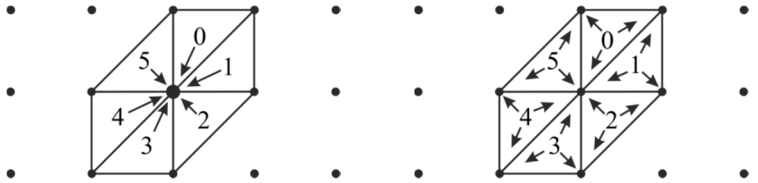


Рис. 1. Схема сборки невязки по точкам сетки (слева) и по элементам (справа)

Целью работы было исследование эффективности нескольких возможных параллельных реализаций МКЭ для решения задачи на высокопроизводительных кластерах с узлами на общей памяти (SMP-узловых кластерах). Расчеты проводились на высокопроизводительных комплексах СФУ и ССКЦ СО РАН.

Отметим, что для компиляции последовательных версий использовался набор оптимизирующих ключей компилятора, который давал наименьшее время выполнения программы. Именно это время использовалось для вычисления ускорения параллельных версий. Перечислим основные выводы, которые можно сделать на основе теоретического и численного анализа (рис. 2, 3).

1. Последовательные реализации. Численные эксперименты показали, что время выполнения последовательной программы при сборке невязки по элементам в 1,5 раза меньше, чем при сборке по точкам сетки, что можно объяснить более выгодным распределением памяти в первом случае.

2. Реализации для вычислительных систем (ВС) с общей памятью на основе технологии OpenMP. Поскольку вклад в невязку в точке дают несколько треугольников (рис. 1), то при параллельной реализации поэлементной сборки на общей памяти будут существовать точки сетки, которые обрабатываются разными нитями, причем, возможно, одновременно. Следовательно, в этом случае необходимы дополнительные затраты на синхронизацию нитей, которая занимает до 40% времени выполнения основного цикла, эффективность распараллеливания около 25% (рис. 2). При сборке невязки по точкам сетки дополнительной синхронизации нитей не требуется, что дает явные преимущества этого подхода. Эффективность распараллеливания составляет около 90% при использовании до 30 нитей и около 80% при использовании более 30 нитей (одна нить на ядро).

3. Реализация для ВС с распределенной памятью на основе технологии MPI. Поскольку используется подход, связанный с декомпозицией вычислительной области без теневых граней (перекрытий), то при обоих способах сборки невязки на каждой итерации неизбежно возникают следующие накладные расходы [12, 13]:

1) время, затрачиваемое на обмены типа точка-точка между соседними процессами для сборки полной невязки на разрезах вычислительной области;

2) время на коллективную операцию глобальной редукции для расчета критерия останова итерационного процесса.

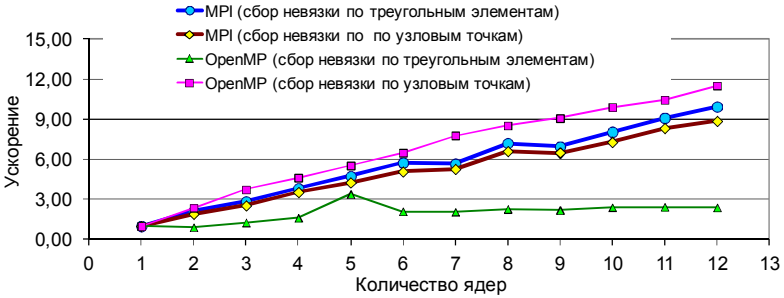


Рис. 2. График зависимости ускорения вычислений от количества используемых ядер для MPI- и OpenMP-версий программ

Численные эксперименты показали преимущество сборки невязки по элементам с эффективностью параллельной реализации около 80%. Рис. 3(в) демонстрирует, что эта версия программы выполняется быстрее, чем MPI-версия при сборке невязки по узловым точкам, что объясняется более выгодным распределением памяти в первом случае. Особо следует отметить, что эта версия MPI-программы по времени выполнения выигрывает и по сравнению с самой быстрой OpenMP-версией.

4. Совмещение технологий MPI и OpenMP для SMP-узлового кластера (рис.3). В численных экспериментах запускались два MPI-процесса на узел, по пять OpenMP-нитей на каждый MPI-процесс. Из рис. 3 видно, что при сборке невязки по узловым точкам ускорение вычислений MPI+OpenMP-версии программы практически совпадает с линейным, а эффективность составляет около 100%. В случае сборки невязки по элементам эффективность параллельной реализации составляет всего 40%, что ожидаемо ввиду затрат на синхронизацию нитей в OpenMP.

Результаты исследований показали, что наиболее эффективной из рассмотренных параллельных реализаций алгоритма оказалась MPI+OpenMP-версия программы при сборке невязки по точкам сетки. Следует отметить, что она является в то же время наиболее сложной в реализации (требует создания, хранения и обработки дополнительных структур). При этом наименьшее время выполнения показывает MPI-версия со сборкой невязки по треугольным элементам (рис. 3).

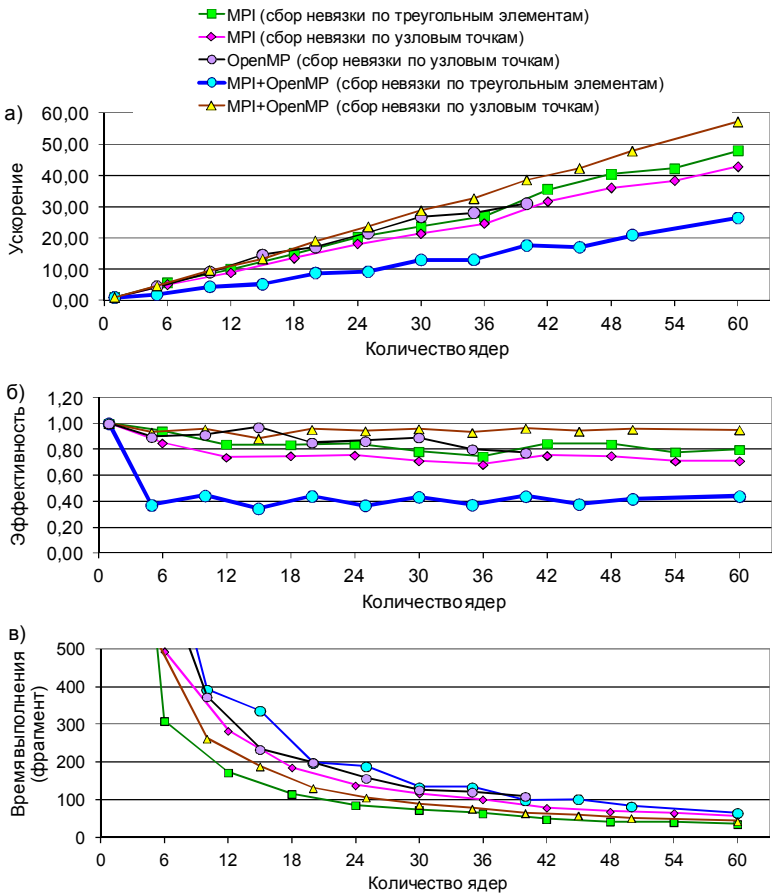


Рис. 3. Графики зависимости ускорения вычислений (а), эффективности (б) и времени выполнения (в) от количества используемых ядер для MPI-, OpenMP- и MPI+OpenMP-версий программ

Литература

1. *Agoshkov V.I.* Inverse problems of the mathematical theory of tides: boundary-function problem // Russ. J. Numer. Anal. Math. Modelling. 2005. Vol. 20, №1. P. 1–18.

2. *Kamenshchikov L.P., Karepova E.D., Shaidurov V.V.* Simulation of surface waves in basins by the finite element method // Russian J. Numer. Anal. Math. Modelling. 2006. Vol. 21(4). P. 305–320.

3. *Карепова Е.Д., Шайдуров В.В.* Параллельная реализация МКЭ для начально-краевой задачи мелкой воды // Вычислительные технологии. 2009. Т.14, № 6. С. 45–57.

4. *Ильин В.П.* Методы и технологии конечных элементов. Новосибирск: Изд-во ИВМиМГ СО РАН, 2007.

5. *Smith I. M., Griffits D. V.* Programming the finite element method. Wiley, Chichester, 2004.

6. *Vollaire C., Nicolas L., Nicolas A.* Parallel computing for the finite element method // The European Physical Journal Applied Physics. 1998. V. 1(3). P. 305–314.

7. *Jimack P. K., Touheed N.* Developing Parallel Finite Element Software Using MPI // High Performance Computing for Computational Mechanics. 2000. P.15–38.

8. *Pantale O.* Parallelization of an object-oriented FEM dynamics code: influence of the strategies on the Speedup // Advances in Engineering Software. 2005. V. 36(6). P. 361–373.

9. *Mahinthakumar G., Saied F.* A Hybrid MPI-OpenMP Implementation of an Implicit Finite-Element Code on Parallel Architectures // International Journal of High Performance Computing Applications. 2002. V. 16(4). P. 371–393,

10. *Vargas-Felix M., Botello-Rionda S.* Solution of Finite Element Problems Using Hybrid Parallelization with MPI and OpenMP // Acta Universitaria. 2012. V. 22(7). P.14–24.

11. *Ильин В. П.* Параллельные алгоритмы для больших прикладных задач: проблемы и технологии // Автометрия. 2007. Т. 43, № 2. С. 3– 21.

12. *Karepova E., Shaidurov V., Dementyeva E.* The numerical solution of data assimilation problem for shallow water equations // Int. J. of Num. Analysis and Modeling, Series B. 2011. V.2, № 2–3. P.167–182.

13. *Дементьева Е.В., Карепова Е.Д., Малышев А.В.* Эффективность численного моделирования на кластерных системах распространения поверхностных волн // Вестник НГУ. Серия: Информационные технологии. 2011. Т. 9, № 1. С. 11–20.

Параллельный алгоритм усвоения спутниковых данных измерений

Н.Н. Богословский

Томский государственный университет, Томск

Рассматривается параллельный алгоритм усвоения спутниковых данных измерения влажности почвы. Данный алгоритм основан на методах оптимальной интерполяции и позволяет вычислить начальные значения объемного влагосодержания поверхностного слоя почвы для глобальной модели численного прогноза погоды ПЛАВ. Описана параллельная реализация алгоритма.

Введение

Современная метеорология характеризуется бурным развитием математических моделей атмосферы и их применением для численных расчетов прогнозов погоды. Одним из механизмов повышения качества прогнозов является уточнение начальных данных модели. В системах усвоения данных наблюдений наряду со станционными измерениями все большее значение приобретают спутниковые данные измерений. Особенно это актуально для территории России, где густота наблюдательной сети недостаточная.

Влагосодержание почвы может оказать существенное влияние на приземную температуру и влажность, низкую облачность и осадки, так как значительно влияет на обмен тепла и влаги между поверхностью земли и атмосферой. Влагосодержание почвы может значительно меняться от одной точки поверхности до другой, даже для точек, расположенных относительно близко друг к другу. Это связано как с сильной пространственной изменчивостью почвенных характеристик, так и с влиянием внешних факторов на влагосодержание почвы, например неоднородным распределением осадков. В связи с этим прямые измерения, сделанные в одной точке, не так информативны о характеристиках почвы в других точках. Это одна из причин того, что в настоящее время не существует обширной глобальной наблюдательной сети для прямого измерения влажности почвы. Поэтому актуальной является задача оценки влагосодержания почвы с привлечением спутниковых измерений.

В работе рассматривается усвоение измерений влагосодержания поверхностного слоя почвы, полученных на основе спутниковых данных ASCAT (Advanced Scatterometer) [2, 3].

Метод усвоения спутниковых данных измерений влажности почвы

Для того, чтобы проверить целесообразность использования спутниковых данных измерений для инициализации почвенных переменных и оценить эффект от использования этих данных в системе усвоения, была использована упрощенная схема усвоения, основанная на оптимальной интерполяции. Усвоение для каждой точки расчетной сетки производится независимо от других.

Пусть ошибки наблюдений и ошибки поля первого приближения не коррелируют между собой. Тогда можно записать следующие уравнение для нахождения поля анализа влагосодержания поверхностного слоя почвы [4]:

$$\theta_a = \theta_b + K(\theta_{scat} - \theta_b), \qquad (1)$$

где θ_a – искомое влагосодержание поверхностного слоя почвы, которое будет использоваться в качестве начального условия в модели; θ_{scat} – влагосодержание поверхностного слоя почвы по данным спутниковых измерений в точке, наиболее близкой к точке сетки, в которой проводится расчет; θ_b – значение первого приближения для влагосодержания поверхностного слоя почвы в точке сетки, в которой производится расчет (как правило, в качестве первого приближения берется прогноз по модели на срок 6 часов, который был сделан 6 часов назад); K – константа, которая не зависит от пространства и времени.

Для определения оптимального значения константы K, которое позволяет минимизировать ошибку поля анализа, воспользуемся уравнением (1), переписав его следующим образом:

$$\theta_a = (1-K)\theta_b + K\theta_{scat}.$$

Пусть дисперсия ошибки поля первого приближения равна σ_b, а дисперсия ошибки спутниковых измерений равна σ_o. Как отмечалось ранее, предполагаем, что ошибки поля первого приближения и наблюдений не коррелируют между собой, тогда дисперсия ошибки поля анализа равна

$$\sigma_a^2 = (1-K)^2\sigma_b^2 + K^2\sigma_o^2. \qquad (2)$$

Как видно из данного уравнения, для минимизации дисперсии ошибки поля анализа необходимо, чтобы константа K вычислялась по следующей формуле:

$$K = \frac{\sigma_b^2}{\sigma_b^2 + \sigma_o^2}.$$

К сожалению, точно вычислить ошибки поля первого приближения и наблюдений, а соответственно и их дисперсию, нельзя, поэтому предположим, что $\sigma_b \simeq \sigma_o$, тогда согласно вышеприведенной формуле $K = 0,5$.

Используя уравнение (1) и рассчитанное значение константы K, приведём окончательное выражение для расчета поля анализа влагосодержания слоя почвы.

$$\theta_{a,l} = \begin{cases} \theta_{b,1} + 0,5 * F_1 * F_2(\theta_{scat} - \theta_b), & l = 1 \\ \theta_{b,2} & , l = 2, \end{cases} \quad (3)$$

где l принимает значение 1 или 2 для поверхностного и глубинного слоя почвы соответственно; F_1 – функция, принимающая значение, равное 0, если в точке расчета имеется снежный покров, и 1, если отсутствует; F_2 – функция, принимающая значение 0, если температура поверхностного слоя почвы ниже 275,15К, и 1, если выше 275,15К.

Исследование эффективности и точности задания начальных данных для влагосодержания поверхностного и глубинного слоя почвы по разработанному методу

Для проверки эффективности и точности задания начальных значений влагосодержания поверхностного и глубинного слоя почвы проводилось сравнение с реальными данными измерений влагосодержания на наземных научных станциях сети AmeriFlux. Для этого проводилось усвоение спутниковых данных измерений для июня 2012 года. Усвоение выполнялось каждые 6 часов. Для выбранных станций проводились сравнения станционных данных и значение влагосодержания поверхностного и глубинного слоя почвы полей объективного анализа. На рис. 1 приведено сравнение влагосодержания для июля 2012 года по станции Willow Creek (45,8060 С.Ш.; 90,0798 В.Д.). Точками на графике обозначены данные со станции, пунктирной линией – данные влагосодержания поверхностного слоя почвы без усвоения спутниковых данных. Сплошной линией отмечено влагосодержание при усвоении спутниковых данных измерений. Как видно из графика, усвоение спутниковых данных позволило уменьшить ошибку инициализации влагосодержания поверхностного слоя почвы. Среднее отклонение для станции Willow Creek уменьшилось на 0,1 $м^3/м^3$, а среднеквадратичная ошибка уменьшилась на 0,013 $м^3/м^3$.

Аналогичные сравнения были проведены для еще 6 станций (GLEES, OR – Metolius-intermediate aged ponderosa pine, WA – Wind

River Crane Site, WI – Park Falls/WLEF, Freeman_Ranch_Woodland, AZ – Santa Rita Mesquite). Уменьшение средней ошибки составило 0,07 $м^3/м^3$, а уменьшение среднеквадратичной ошибки – 0,01 $м^3/м^3$.

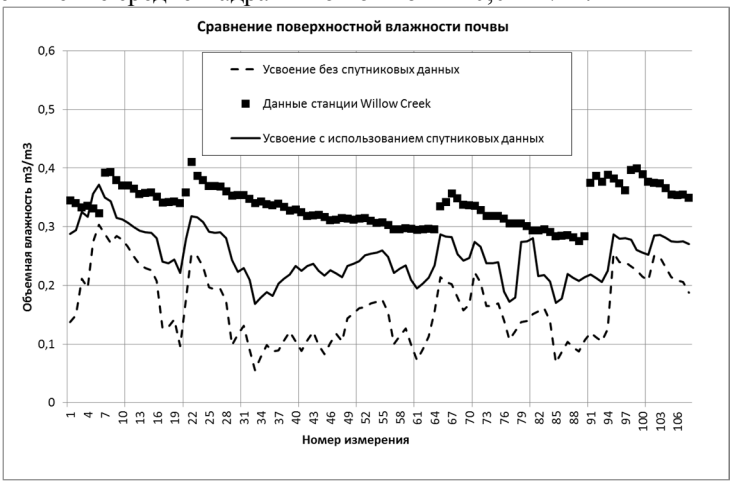


Рис. 1. Сравнение влагосодержания поверхностного слоя почвы

Так как усвоение спутниковых данных проводится только для верхнего слоя почвы, то влияние на влагосодержание глубинного слоя почвы незначительное.

Разработка параллельного алгоритма инициализации почвенных переменных

Одной из сложностей в усвоении данных является вычислительная сторона алгоритма. Большая размерность задач не позволяет их решать на персональных компьютерах. Еще одно из ограничений, накладываемых в задаче усвоения данных наблюдений, – это временное ограничение. Это связано с тем, что при усвоении данных наблюдений в задачах численного прогноза погоды в оперативном режиме необходимо получить анализ в определенный временной интервал. Соответственно, эффективность реализации численного алгоритма является важной составляющей. Поэтому для данных алгоритмов в обязательном порядке требуется проводить распараллеливание.

Увеличение производительности современных вычислительных систем производится не только за счет увеличения числа процессоров, но и за счет увеличения количества ядер в каждом процессоре. Следовательно, необходимо проводить распараллеливание не просто с

использованием какой-либо определенной параллельной технологии, но и с учетом архитектурных особенностей. В связи с этим распараллеливание необходимо проводить как с помощью OpenMP (технология программирования для систем с общей памятью), так и с помощью MPI (технология программирования для систем с распределенной памятью).

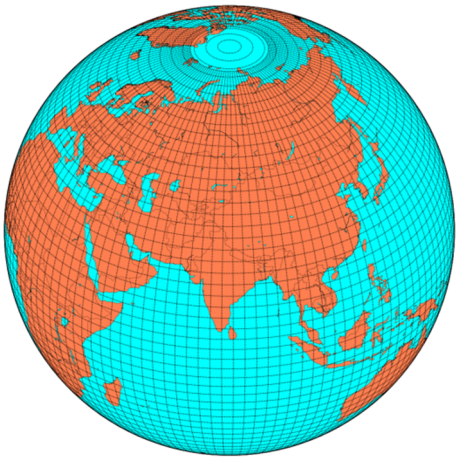


Рис. 2. Один вертикальный уровень широтно-долготной сетки модели ПЛАВ

На рис. 2 представлен один вертикальный уровень расчетной широтно-долготной сетки модели ПЛАВ [1]. Усвоение спутниковых данных для инициализации почвенных переменных проводится для каждой точки сетки. Расчеты в каждой из точек производятся независимо от расчетов в других точек. Так как нет зависимости, то расчеты в разных точках могут проводиться параллельно. Для параллельной реализации расчетов была выбрана двумерная декомпозиция по данным.

Вся расчетная сетка разбивается на подобласти в соответствии с имеющимся количеством MPI-процессов. Внутри каждой такой подобласти расчеты распараллеливаются с использованием технологии OpenMP. Использование технологии распараллеливания OpenMP позволяет эффективно проводить расчеты внутри одного расчетного узла, так как все ядра имеют доступ к общей памяти.

Тестирование и оптимизация параллельного алгоритма проводились для вычислительного кластера Томского государственного университета СКИФ Cyberia.

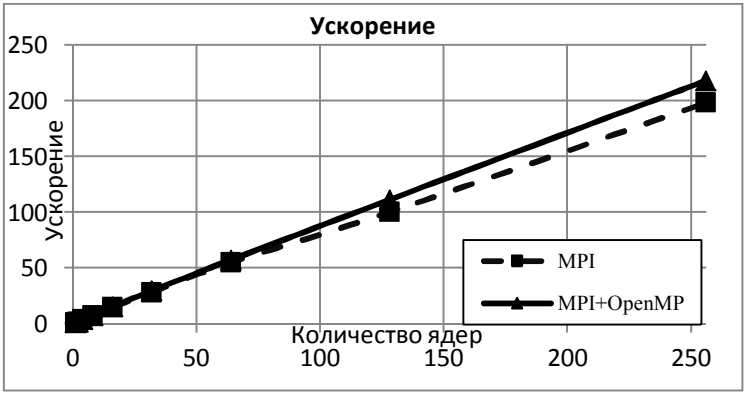


Рис. 3. Ускорение программы усвоения спутниковых данных

Было проведено два вычислительных эксперимента. В первом вычислительном эксперименте для распараллеливания использовалась только технология MPI. Таким образом, не была учтена архитектура вычислительно кластера, в котором на общей памяти располагается 12 вычислительных ядер. Во втором вычислительном эксперименте для распараллеливания использовалась как технология MPI, так и технология OpenMP. В данном эксперименте на одном вычислительном узле запускался 1 процесс MPI и внутри него создавалось 12 нитей OpenMP. На рис. 3 приведен график ускорения. Пунктирной линией на графике показано ускорение при распараллеливании с использованием только технологии MPI, сплошной линией представлен график ускорения при использовании MPI+OpenMP.

Как видно из графика, благодаря применению гибридной технологии распараллеливания удалось увеличить ускорение. Так, при расчетах с использованием 256 ядер ускорение программы при распараллеливании только с использованием MPI составило около 198, ускорение же при использовании гибридной технологии MPI+OpenMP составило 220, что на 10% больше при использовании только MPI.

Литература

1. *Толстых М.А., Богословский Н.Н., Шляева А.В., Мизяк В.Г.* Оперативная технология расчета глобальных прогнозов с помощью полулагранжевой модели атмосферы ПЛАВ // Труды

Гидрометеорологического научно-исследовательского центра Российской Федерации. 2011. № 346. С. 145–154.

2. *Bartalis Z., Wagner W., Naeimi V., Hasenauer S., Scipal K., Bonekamp K., Figa J., Anderson C.* Initial soil moisture retrievals from the METOP-A Advanced Scatterometer (ASCAT) // Geophysical Research Letters. 207. V. 34. L20, 401, doi:10.1029/2007GL031088.

3. *Bartalis Z., Naeimi V., Hasenauer S., Wagner W.* ASCAT Soil Moisture Product Handbook // ASCAT Soil Moisture Report Series, 2008, N. 15, Institute of Photogrammetry and Remote Sensing, Vienna University of Technology, Austria.

4. *Scipal K., Drusch M., Wagner W.* Assimilation of a ERS scatterometer derived soil moisture index in the ECMWF numerical weather prediction system // Advances in water resources. 2008. V.31 (8). P.1101–1112.

Разработка программного комплекса для конструирования программ обработки данных на высокопроизводительных вычислительных системах

А.Б. Купчишин, В.Г. Сарычев

Новосибирский государственный технический университет, Новосибирск

Представлены проект и прототип программного комплекса, предназначенного для визуального автоматизированного конструирования программ обработки сейсмических данных, параллельная реализация алгоритма когерентного суммирования для поиска эпицентра микросейсмической активности, полученные результаты эффективности распараллеливания.

Введение

Комплекс Madagaskar[1] представляет собой коллекцию процедур для обработки геофизических данных и язык сценариев для конструирования схем вычислений из таких процедур. Коллекция процедур расширяется за счет усилий различных авторов, которым требуются те или иные операции с данными, и качество реализации таких процедур различно. В некотором смысле Madagaskar – это набор спецификаций и реализаций-примеров тех процедур, которые часто требуются для решения практических задач, и актуальной работой является повышения качества реализаций наиболее востребованных процедур для современных вычислительных систем. С повышением количества процедур и с необходимостью делать схемы обработки данных относительно большими возрастает сложность конструирования таких схем посредством языка сценариев. Необходимо создание инструмента для высокоуровневого конструирования программ обработки данных из процедур, позволяющего сделать конструирование более наглядным и автоматизировать выбор реализаций процедур под особенности конкретной вычислительной системы.

В настоящей работе представлен проект и прототип программного комплекса, предназначенного для решения задач обработки сейсмических данных, параллельная реализация алгоритма

[1] Madagaskar, http://www.ahay.org/wiki/Main_Page

когерентного суммирования для поиска эпицентра микросейсмической активности

Архитектура программного комплекса

Программный комплекс, призванный облегчить написание управляющей программы (сценария), задача которой сводится к запуску библиотечных подпрограмм в нужном порядке, включает в себя:

- визуальный конструктор программ,
- расширяемую библиотеку процедур обработки данных, характерных для геофизических задач,
- интерпретатор сценариев.

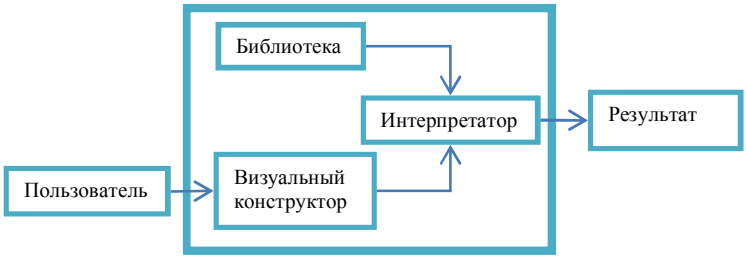


Рис. 1. Схема программного комплекса

Визуальный конструктор включает в себя:

1) инструменты визуального конструирования схем обработки данных;
2) модуль проверки корректности связей, который выполняет проверки на соответствие связываемых объектов, типов входных и выходных данных;
3) модуль кодогенерации, создающий для визуального сконструированного сценария его текстовое представление (скрипт-файл) в формате, понятном для интерпретатора.

Интерпретатор – это модуль, который исполняет сконструированную программу, используя при этом предоставленные библиотекой операции и процедуры.

Визуальный конструктор программ

Пользователь при конструировании программы оперирует тремя объектами: переменными, операциями, стрелками (см. пример на

рис. 2). Переменная – это ячейка памяти, характеризуется типом данных и содержит некоторые значения, может выступать в качестве параметра для операций. Операция – процедура, которая характеризуется набором входных и выходных данных и выполняет преобразование одного в другое. Стрелки устанавливают одностороннюю связь между объектами (операция-операция, переменная-операция), обозначающую направление передачи данных.

Рис. 2. Пример сконструированной схемы программы

Переменные представлены прямоугольниками var, операции – прямоугольниками operation, стрелки отражают направление потоков данных и аттрибутированы типом данных.

Схема на рис. 2 представляет собой цепочку из трех операций, последовательно передающих свой результат на вход последующей. Результат последней операции посылается выходной переменной, а значения входных переменных передаются первой операции.

Результаты отработавшей операции могут передаваться сразу нескольким соответствующим объектам, создавая потенциал для одновременного исполнения множества операций. Типы данных определяются предметной областью, и возможные для использования типы поставляются библиотекой процедур (в геофизике, например, приняты определенные форматы для представления данных, собранных с датчиков, которыми и могут характеризоваться уже конкретные типы данных).

Параллельная реализация метода когерентного суммирования

При добыче углеводородов методом гидроразрыва пласта требуется находить координаты разрывов среды. В момент порождения разрывы являются источниками (эпицентрами) микросейсмической активности. Для их локализации используется метод когерентного суммирования [1].

Модель задачи

Чтобы определить координаты эпицентра микросейсмической активности, на поверхности исследуемого объема земли раскладывают

датчики, улавливающие распространяющееся возмущение. Время, за которое волна достигает различных датчиков, неодинаково. Датчики делают замеры через промежутки времени. Каждый такой замер являет собой картину состояний датчиков на определенный момент времени. За промежуток времени набирается набор состояний для различных моментов времени, который будем называть разверткой по времени.

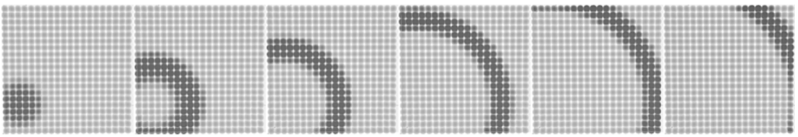


Рис. 3. Регистрация датчиками распространения волны в идеальных условиях без шумов

Датчики, помимо возмущения от разрыва, улавливают еще и шум, который сопоставим по силе с самим возмущением. Поэтому наблюдаемая картина соответствует рис. 4.

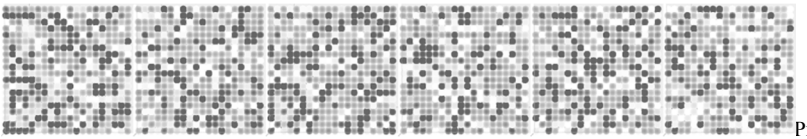


Рис. 4. Состояния датчиков с учетом шума

Когерентное суммирование

На рис. 5 представлен срез куба земли, треугольниками обозначены датчики, звездочкой – эпицентр, справа – развертка по времени. Каждая из вертикальных линий отображает временную развертку состояния датчика. Направление времени указано стрелкой. Пики на линиях отмечают время прихода сигнала до соответствующего датчика. Метод когерентного суммирования предполагает следующую процедуру:

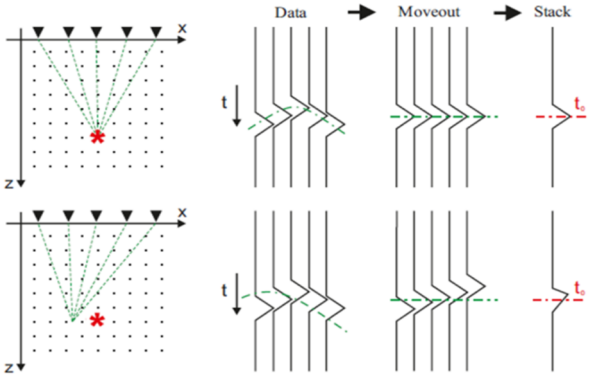


Рис. 5. Обнаружение эпицентров сейсмической активности с помощью метода когерентного суммирования

1) в кубе среды строится сетка;
2) для каждого узла:
   a) рассчитывается время распространения сигнала из этого узла до датчиков. Таким образом, получается некоторый профиль волны, которая могла бы распространяться из этого узла, если бы эпицентр был в нем;
   b) выбирается датчик, до которого время прихода волны минимально, берется значение состояния датчика в начальный момент времени (первое состояние с начала замеров), состояния других датчиков выбираются в моменты времени, соответствующие приходу волны до этих датчиков. Все выбранные состояния суммируются. Далее фронт сдвигается на следующий шаг по времени и повторяется процедура суммирования. Так продолжается до конца таблицы измерений. Из всех сумм выбирается сумма с максимальным значением;
3) из всех максимальных для каждого узла сумм выбирается максимальная. Узел, соответствующий этой сумме, считается наиболее близким к эпицентру.

Реализация

Реализован параллельный алгоритм когерентного суммирования, сочетающий в себе следующие особенности:

- параллелизм в распределенной памяти – исходные данные являются разверткой по времени, которую мы можем поделить на некоторое

количество отрезков, назначив каждому узлу свой отрезок, который он будет обрабатывать;

- параллелизм в общей памяти – внутри вычислительного узла потоки параллельно вычисляют суммы для разных узлов пространственной сетки;
- двойная буферизация – подгрузка данных в память на фоне обработки ранее загруженных данных;
- алгоритм уточнения исследуемой области – позволяет сократить количество обрабатываемых точек за счет поэтапного сужения исследуемой области локализации эпицентра.

На рис. 6 представлена схема распараллеливания в распределённой памяти, таблица состояний датчиков делится поровну на отрезки времени между процессами, и каждый поток ищет эпицентр на ограниченном отрезке времени, такой эпицентр будем называть локальным.

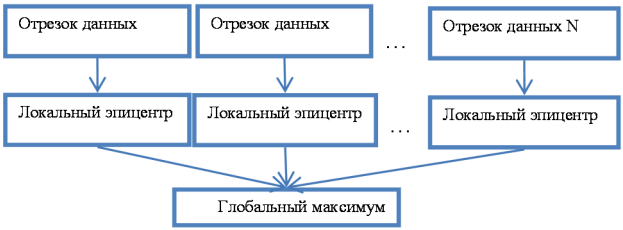


Рис. 6. Схема распараллеливания в распределенной памяти

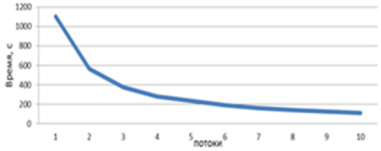


Рис. 7. Зависимость времени от количества потоков

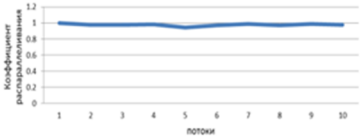


Рис. 8. Эффективность распараллеливания

При таком распределении нагрузки потоки работают с минимальными накладными расходами и независимо друг от друга, чем и обусловлены близкие к идеальным показатели эффективности распараллеливания (рис. 7, 8).

На рис. 9 представлена схема распараллеливания в общей памяти, потоки распределяют между собой узлы сетки, для которых проводится суммирование, т.е. делят исследуемое пространство.

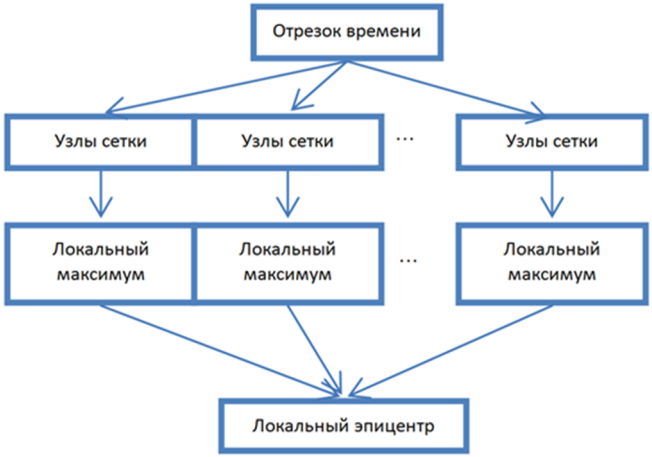


Рис.9. Схема распараллеливания в общей памяти

Внутри каждого потока ищется узел с максимальным результатом суммирования, который будем называть его локальным максимумом.

При таком распределении нагрузки потоки выполняются независимо друг от друга, т.е. отсутствует простой одних вычислительных устройств в ожидании завершения работы других, чем и обусловлены близкие к идеальным показатели эффективности распараллеливания (рис.10, 11).

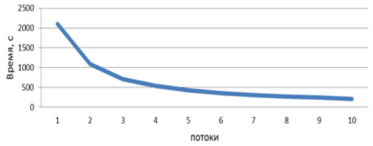


Рис.10. Зависимость времени от количества потоков

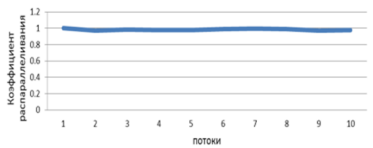


Рис.11. Эффективность распараллеливания

Для обработки больших объемов данных, которыми характеризуется задача, используется предварительная подгрузка данных в память на фоне обработки ранее загруженных данных (рис. 12).

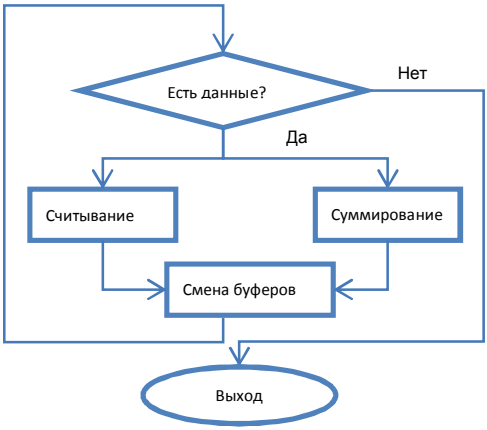


Рис.12. Схема подгрузки данных

Для организации подгрузки используются два буфера данных.

Порождаются два потока. Пока один проводит суммирование (используя первый буфер), второй тем временем готовит новую порцию данных (заполняя второй буфер).

После того как данные обработаны, буферы меняются местами.

Уточняющий алгоритм:

1) разбиваем исследуемую область на подобласти;
2) узнаем результат суммирования для центра подобластей;
3) выбираем ту подобласть, в которой значение оказалось максимальным;
4) теперь так же исследуем полученную подобласть;
5) делаем так до тех пор, пока исследуемая область не станет достаточно малой;

Ускорение достигается за счет непосредственного уменьшения числа обсчитываемых точек.

Заключение

Разработана архитектура и реализован прототип программного комплекса для визуального конструирования программ обработки геофизических данных, который позволяет конструировать

линейные цепочки процедур обработки данных. Реализован набор операций, решающий задачу поиска эпицентра микросейсмической активности методом когерентного суммирования. Реализован параллельный алгоритм когерентного суммирования.

Дальнейшая работа предполагает наполнение библиотеки эффективными реализациями процедур, специфицированных в Madagaskar, реализацию возможности конструировать нелинейные схемы программ, интеграцию инструмента визуального конструирования в среду HPC Community Cloud [2,3].

Литература

1. *Колесников Ю.И., Хогоев Е.А., Полозов С.В., Донцов М.В.* Применение сейсмоэмиссионной томографии для локализации сейсмических источников // Сборник докладов Международной конференции, посвященной 90-летию академика Пузырева Н.Н. «Сейсмические исследования Земной коры». Новосибирск, 2004, С. 129–134.
2. *Городничев М.А., Малышкин В.Э., Медведев Ю.Г.* HPC Community cloud: эффективная организация работы научно-образовательных суперкомпьютерных центров // Научный вестник НГТУ. 2013. №3(52). С. 91–96.
3. *Вайцель М. А. Городничев М.А.* HPC Community Cloud: разработка инструментария для повышения уровня взаимодействия пользователей с объединенными HPC-системами // Седьмая Сибирская конференция по параллельным и высокопроизводительным вычислениям. Программа и тезисы докладов. Томск: Изд-во Том. ун-та, 2013. С. 82–84.

Параллельный алгоритм разделения сеточного графа на домены

В.Н. Берцун

Томский государственный университет, Томск

Для решения прикладных задач на сеточных графах большой размерности с помощью многопроцессорных вычислительных систем часто используют декомпозицию расчетной области, в результате которой граф разделяется на домены (подграфы, части сетки). Это связано с необходимостью сбалансирования нагрузки для выделенных q процессоров или хранения многомерных сеток.

Объем передачи данных между процессорами зависит от числа ребер (веса сечения), соединяющих вершины, принадлежащие разным доменам, распределенным по процессорам. Для разделения графов используются следующие геометрические методы [1, 2]: покоординатное разбиение, рекурсивный инерционный метод деления пополам, деление сети с использованием кривых Пеано. Эти методы основываются на координатной информации об узлах графа и поэтому не оптимизируют связи между доменами. Разделение графа методом спектральной бисекции [3] предполагает, что каждой i-той вершине простого связного графа G_n ставится в соответствие компонента x_i вектора $\overline{x}$, равная +1 или –1, таким образом, что

$$\sum_{i=1}^{n} x_i = 0, \quad \sum_{i=1}^{n} x_i^2 = n. \tag{1}$$

При этих ограничениях минимизируется квадратичная форма

$$(L(G_n)\overline{x}, \overline{x}) = \sum_{(i,k)\in E} (x_i - x_k)^2, \tag{2}$$

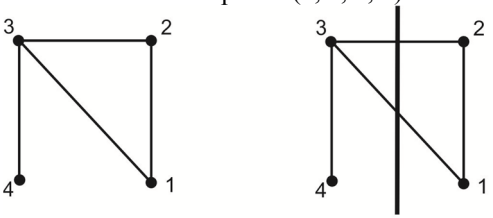
где $L=A–V$, V– нулевая матрица, содержащая на главной диагонали степени v_i вершин графа G_n, A – матрица смежности графа, L – симметричная матрица Лапласа с элементами

$$l_{ij} = \begin{cases} 1, \text{ если вершины } x_i \text{ и } x_j \text{ смежны,} \\ 0, \text{ если } i \neq j \text{ и вершины } x_i \text{ и } x_j \text{ не смежны,} \\ -v_i, \text{ если } i = j;\ i, j = \overline{1,n}. \end{cases}$$

Приближенное решение задачи (1)–(2) сводится к определению собственного вектора Фидлера $\overline{F}$, соответствующего максимальному ненулевому собственному значению λ_2 матрицы Лапласа L, затем координаты $\overline{F}$ отображаются на множество вершин графа V. Этот метод минимизирует суммарный вес ребер, соединяющих вершины из двух разных доменов V_1, V_2. При этом для четных n

$$V_1 \cup V_2 = \varnothing,\ n_1 + n_2 = n,\ |V_1| = |V_2|.$$

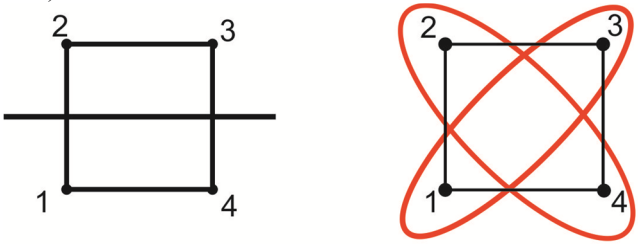
Например, для графа из рис. 1a вектор $\overline{F} = (-0.408, -0.408, 0, 0.816)$, а порядок номеров вершин соответствует компонентам вектора V= (1, 2, 3, 4).



a b

Рис. 1. Разделение графа на два домена для n=4

Используя этот вектор, можно распределить вершины графа в два домена V_1, V_2 так, что $d = n_1 / n_2$, $V_1 \cup V_2 = \varnothing$, $n_1 + n_2 = n$. Для d=1 такое разделение графа представлено на рис.1b, что соответствует оптимальному разбиению графа на два домена с двумя связями. Для простого цикла C_4 на рис. 2a представлено оптимальное разделение графа (λ_2=−2), а на рис.2b – не оптимальное (λ_4= −4).



a b

Рис. 2. Разделение графа на домены a – λ_2= −2; b – λ_4= −4

Как следует из этих рисунков, во втором случае два домена связаны уже не двумя, а четырьмя связями. Таким образом, вектор $\overline{F}$ действительно минимизирует связи между доменами [4, 5]. Используя алгоритм рекурсивной бисекции (*RSB*), граф можно последовательно разделить и на произвольное четное число частей.

Собственное значение λ_2 (или спектр R матрицы L) можно вычислить по ее характеристическому многочлену. Рассмотрим один из параллельных алгоритмов для нахождения коэффициентов характеристического многочлена матрицы Лапласа:

$$P_L(\lambda) = \det(L - \lambda E) = (-1)^n[\lambda^n + c_1\lambda^{n-1} + ... + c_n], \; n >> 1. \qquad (3)$$

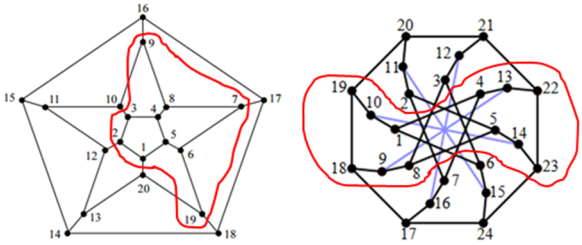
Такой многочлен можно построить методом Леверье, используя понятие следа *Sp*L матрицы L [6]. Если $R = [\lambda_i, \; i = \overline{1,n}\;]$ – спектр этой матрицы, то λ_i^k являются собственными значениями матрицы L^k. Обозначим

$$S_k = \lambda_1{}^k + \lambda_2{}^k + ... + \lambda_n{}^k = SpL^k \;,$$

тогда при $1 \le k \le n$ коэффициенты c_k, $k = \overline{1,n}$ характеристического многочлена (3) определяются по формуле

$$c_k = -\frac{1}{k}\left(S_k + c_1 S_{k-1} + c_2 S_{k-2} + ... + c_{k-1} S_1\right), \;\; S_k = 0, \text{ если } k \le 0.$$

Сложность этого алгоритма $O(n^4)$, а вычисление степеней и следов матрицы L для $n>>1$ осуществляется в параллельном режиме [2, 7].



a b

Рис.3. Разделение графа на два домена (a – граф G20, b – граф G24)

Определив коэффициенты характеристического многочлена и границы его спектра R по теореме Гершгорина, используем параллельный алгоритм сплайновой интерполяции для определения λ_2 и параллельный алгоритм для определения вектора Фидлера. В

качестве примера рассмотрим регулярные графы додекаэдра G_{20} и Макджи G_{24}, для которых оптимальное разделение на два домена представлено на рис.3.

Так как матрица Лапласа симметрична, то для решения полной проблемы собственных значений можно использовать параллельный метод вращений или метод Ланцоша приведения матрицы к трехдиагональной форме [8]. Если граф G_n является регулярным, то спектр R матрицы L можно вычислить по спектру матрицы смежности A.

Литература

1. *Волков К.Н.* Балансировка нагрузки процессоров при решении краевых задач механики жидкости и газа сеточными методами // Вычислительные методы и программирование. 2012. Т.13. С.107–129.
2. *Гергель В.П.* Высокопроизводительные вычисления для многопроцессорных многоядерных систем. М.: МГУ, 2010. 544 с.
3. *Fiedler M.* Eigenvectors of acyclic matrices // Czechoslovak Mathematical Journal. 1975. V.25(100). P. 607–618.
4. *Якобовский М.В.* Обработка сеточных данных на распределенных вычислительных системах // Вопросы атомной науки и техники. Сер. Математическое моделирование физических процессов. 2004. Вып. 2. С.40–53.
5. *Берцун В.Н.* Математическое моделирование на графах. Ч. 2. Томск: Изд–во Том. ун–та, 2013. 86 с.
6. *Березин И.С., Жидков Н.П.* Методы вычислений. М., 1959. Т. 2. 620 с.
7. *Старченко А.В., Берцун В.Н.* Методы параллельных вычислений. Томск: Изд-во Том. ун-та, 2013. 223 с.
8. *Квасов Б.И.* Численные методы анализа и линейной алгебры. Ч. 2: Линейная алгебра. Новосибирск, 2009. 132 с.

Средства для задания императивного управления во фрагментированных программах на примере задачи моделирования самогравитирующего вещества методом частиц-в-ячейках

А.А. Ткачёва

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск

В Лаборатории синтеза параллельных программ ИВМ и МГ СО РАН разрабатывается система фрагментированного программирования LuNA [1]. Технология фрагментированного программирования предназначена для реализации больших численных задач на суперкомпьютерах. В ней прикладная программа собирается из множества фрагментов, и фрагментированная структура программы сохраняется в ходе вычислений, что позволяет автоматизированно обеспечивать динамические свойства программы (динамическая балансировка загрузки процессоров, реализация коммуникаций на фоне счета, и т.д.). Такой подход к распараллеливанию позволяет программировать на более высоком уровне и уйти от некоторых сложностей системного параллельного программирования. Описание фрагментированного алгоритма является платформенно независимым, а настройку на конкретный вычислитель обеспечивает Runtime-система LuNA.

Был разработан модуль императивного управления (далее – прямого управления), его задачей является эффективная реализация фрагментированных подпрограмм. Главной чертой этого модуля является то, что он выполняет фрагментированную программу не по базовому алгоритму Runtime-системы LuNA, а в соответствии с дополнительно определенным прямым управлением. Кроме того, были разработаны средства представления прямого управления во фрагментированной программе.

Фрагментированная программа (ФП)

ФП представляет собой набор <*DF, CF, in, out*>, где *DF* – это множество фрагментов данных (ФД) единственного присваивания; *CF* – это множество фрагментов вычислений (ФВ) единственного

срабатывания. Для каждого $a \epsilon CF$ определены набор входных ФД *in(a)* и набор выходных ФД *out(a)*.

Выполнение ФВ состоит в вычислении его выходных ФД из значений входных ФД. ФВ реализуется некоторой функцией без побочных эффектов. Выполнение ФП заключается в выполнении всех его ФВ. Порядок выполнения ФВ в ФП основан на информационных зависимостях между ФВ.

Семантика базового алгоритма выполнения ФП

1. Среди множества всех ФВ выбираются те, чьи значения входных ФД уже вычислены. Такие ФВ называются готовыми к выполнению.
2. Из множества готовых к выполнению ФВ выбирается один или несколько ФВ на выполнение.
3. По мере выполнения ФВ какие-то ФД получают значения. При этом у некоторых ФВ все входные ФД оказываются вычисленными, и такие ФВ переходят во множество готовых к выполнению.
4. Пункты 2–3 повторяются, пока множество ФВ, готовых к выполнению, не пусто и ни один ФВ не выполняется.

Заметим, что, так как в общем случае порядок выполнения ФВ не детерминирован, базовый алгоритм выполнения ФП требует много времени и ресурсов. Чтобы уменьшить степень недетерминизма выполнения ФП, можно разработать и ввести дополнительные средства для задания императивного управления выполнения в ФП, которые позволили бы задавать желаемый порядок выполнения ФВ, но при этом какая-то степень недетерминизма еще оставалась бы для возможности параллельного выполнения на мультикомпьютере и обеспечения динамических свойств выполнения программы.

Целью работы является разработка средств задания прямого управления во ФП для системы LuNA и реализация их в виде программного модуля.

При этом эти средства должны удовлетворять следующим требованиям:

- позволять описывать желаемый порядок выполнения ФВ и распределение ресурсов для некоторого достаточно большого класса численных алгоритмов;

- представление алгоритма должно допускать эффективную реализацию на мультикомпьютере.

Управляющая сеть Петри

В качестве средств для задания прямого управления была выбрана модификация сети Петри (далее – управляющая сеть Петри (УСП)), а в качестве класса численных алгоритмов были выбраны итерационные алгоритмы (явная разностная схема, итерационные методы решения СЛАУ и т.д.). Модификацией сети Петри является то, что после срабатывания перехода не обязательно, что в каждом выходном месте появится фишка, это было введено для более прямого соответствия модели ФП.

Интерпретация УСП

Будем интерпретировать УСП следующим образом: Срабатыванию перехода сопоставим вычисление некоторого ФВ. Месту соотнесем некоторый буфер в памяти. Присутствие фишки в месте означает, что в этом буфере находится значения некоторого ФД. Ориентированные дуги между переходами и местами и местами и переходами будут сопоставляться с соответствующими наборами in и out для каждого ФВ. Порядок выполнения ФВ определяется функционированием УСП.

Использование УСП для задания прямого управления во ФП без использования базового алгоритма позволит существенно сократить накладные расходы. В то же время с помощью УСП можно задавать не только порядок выполнения ФВ, но и частичное распределение ресурсов, при этом к одной и той же ФП могут применяться различные УСП в зависимости от характеристик вычислительной среды.

Реализация программного модуля, осуществляющего прямое управление

В ходе работы был реализован программный модуль RuSh (Runtime Shell), который обеспечивает параллельное выполнение в общей памяти ФП, в которой порядок исполнения ФВ задан УСП. Язык реализации C++, для параллельного выполнения в общей памяти использовалась библиотека POSIX Threads.

Особенности и ограничения реализации: в разработанном программном модуле поддерживается только безопасная УСП (не более одной фишки в месте). Это условие позволяет в реализации ограничиться одним буфером для обмена ФД.

Моделирование самогравитирующего вещества методом частиц-в-ячейках

Исследование производительности и применимости разработанного модуля и УСП проводилось на задаче численного моделирования самогравитирующего вещества методом частиц-в-ячейках. Подробно эта задача описана в работе [2], а здесь приведем основные этапы алгоритма.

Моделирование разбивается на временные итерации, а каждая временная итерация состоит из следующих этапов:

- расчет плотности;
- расчет потенциала (решение уравнение Пуассона);
- расчет скорости;
- сдвиг частиц.

Фрагментация метода частиц-в-ячейках основана на декомпозиции области. В рассмотренном случае пространственная сетка разбивалась вдоль оси *Ox*. Этапы каждой временной итерации рассчитываются разными ФВ. На этапе расчета потенциала при решении уравнения Пуассона необходимо осуществлять обмен теневыми гранями потенциала, а также актуальными значениями плотности для границ. Также на каждой итерации происходит сдвиг, что влечет необходимость обмена частицами между ФВ.

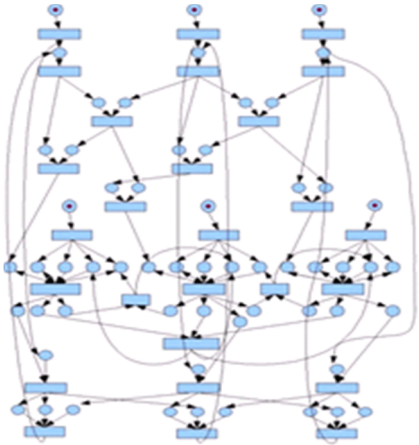


Рис. 1. УСП для метода частиц-в-ячейках при декомпозиции области на 3 подобласти

Для ФП метода частиц-в-ячейках была разработана УПС, показанная на рис.1. Заметим, что для хранения ФД различных временных итераций использовались одни и те же буферы памяти.

Исследование производительности и сравнительное тестирование

Исследование проводилось на 8-ядерном узле кластера, принадлежащего ССКЦ СО РАН (г. Новосибирск). Для тестирования метода частиц-в-ячейках использовалось два типа данных:

- данные А. Размер сетки 64x64x64, количество частиц 1000000, 1000 временных итераций моделирования;
- данные Б. Размер сетки 300x300x300, количество частиц 1000000, 100 временных итераций моделирования.

Такие параметры являются близкими к реально используемым [2] и поэтому позволяют судить о характеристиках программ в условиях, приближенных к реальным.

Тест 1. Сравнение с реализацией последовательного алгоритма

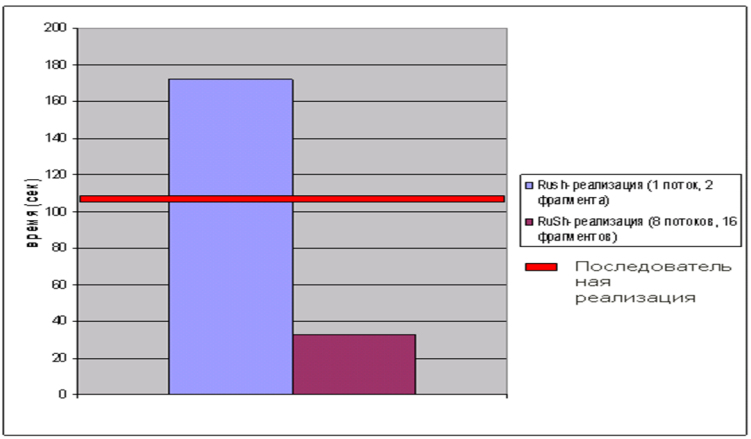


Рис.2. Последовательная реализация и реализации ФП в RuSh (данные А)

Целью данного теста было оценить накладные расходы, связанные с фрагментацией алгоритма, для этого было проведено сравнение времени работы реализации последовательного алгоритма и

реализации ФП в RuSh. Также было найдено оптимальное количество потоков и разбиение области, при котором достигается максимальное ускорение для оценки полученного выигрыша в производительности по сравнению с последовательным алгоритмом.

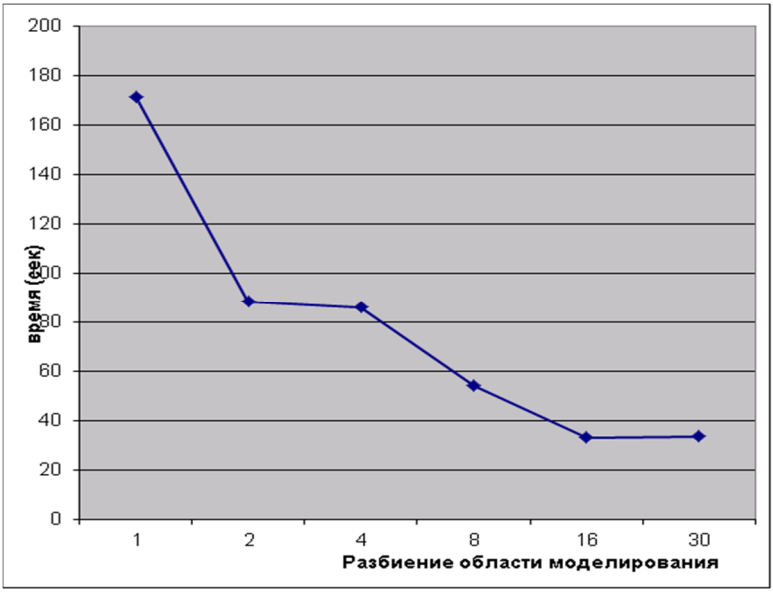


Рис. 3. Время работы Rush-реализации в зависимости от разбиения области моделирования для оптимального количества потоков (данные A)

На рис. 2 изображено время работы реализации ФП в RuSh при оптимальных параметрах. Ими оказались 8 потоков и декомпозиция области на 16 подобластей.

Результаты расчетов показывают, что реализация в RuSh на 1 потоке не сильно уступает реализации последовательного алгоритма. Накладные расходы, связанные с усложнением алгоритма после фрагментации, вполне терпимы. Параллельная реализация в RuSh работает быстрее, чем последовательная. При этом на 8-ядерном узле в идеале можно было ожидать 8-кратного ускорения. В нашем случае RuSh-реализация ускоряется максимум в 5,24 раза – это является хорошим результатом, так как решаемая задача относится к задачам с большим обращением к памяти, и

узким местом здесь выступает пропускная способность шины памяти.

Тест 2. Исследование масштабируемости

Под масштабируемостью в данном случае понимается способность системы справляться с увеличением рабочей нагрузки при добавлении ресурсов. Для её оценки сравнивалось время работы программы с увеличением степени декомпозиции области для оптимального количества потоков (рис. 3). Из рис. 3 видно, что реализация в RuSh справляется с нагрузкой, что говорит о её удовлетворительной масштабируемости при заданных параметрах.

Тест 3. Сравнительное тестирование с MPI-реализацией

Так как одним из требований к предлагаемым средствам для задания прямого управления во ФП является возможность эффективного исполнения на мультикомпьютере, то цель данного теста – провести сравнение производительности с реализацией на MPI, при этом для объективности сравнения все MPI процессы запускались на одном узле, чтобы свести к минимуму накладные расходы, связанные с доступом к памяти между процессами.
Из рис. 4 видно, что производительность RuSh-реализации практически совпадает с производительностью MPI-реализации, что

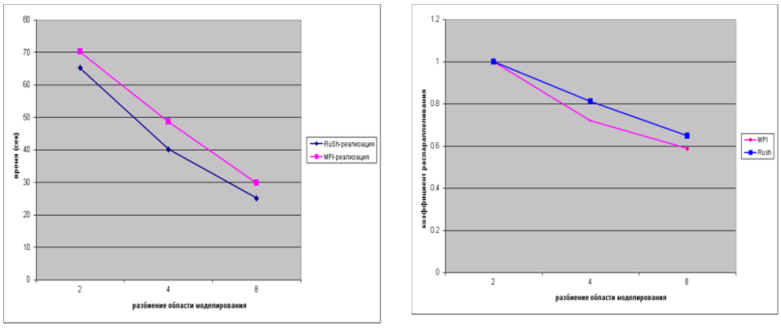


Рис.4. Сравнение времени работы и коэффициента распараллеливания Rush- и MPI- реализаций в зависимости от разбиения области моделирования (данные Б)

является хорошим результатом и говорит о том, что разработанный

модуль позволяет эффективное исполнение на мультикомпьютере ФП, в которой управление задано УСП.

Заключение

Были предложены средства для задания прямого управления при ФП для системы LuNA. Реализован программный модуль RuSh для параллельного выполнения на мультикомпьютерах с общей памятью ФП, в которой управление задано УСП.

Разработана УСП для задачи моделирования самогравитирующего вещества методом частиц-в-ячейках и реализована соответствующая ФП. Проведено сравнительное тестирование производительности между RuSh- и MPI-реализациями на многоядерном мультипроцессоре. Результаты сравнительного тестирования показали применимость предлагаемых средств.

Литература

1. *Malyshkin V.E., Perepelkin V.A.* LuNA Fragmented Programming System, Main Functions and Peculiarities of Run-Time Subsystem // Proceedings of the 11-th Conference on Parallel Computing Technologis, LNCS 6873. Springer, 2011. P. 53–61.
2. *Киреев С.Е.* Параллельная реализация метода частиц-в-ячейках для моделирования задач гравитационной космодинамики // Автометрия. 2006. Т.42, № 3. С.32–39.

Параллельные алгоритмы для решения обратных задач динамики небесных тел[1]

И.Н. Чувашов

НИИ прикладной математики и механики Томского государственного университета, Томск

Рассматривается возможность использования параллельных вычислений для решения обратных задач динамики небесных тел. Особенности и трудоемкость этой задачи позволяют на всем этапе численного моделирования применять параллельные алгоритмы, что приводит к значительному увеличению скорости работы приложения и дает возможность использовать разрядную сетку высокого порядка (128 бит) без существенных временных затрат.

Формулировка прямой и обратной задач динамики ИСЗ

Прямая задача динамики ИСЗ определяется уравнениями движения спутника с заданными начальными условиями. Как известно, движение искусственного спутника Земли можно представить как движение материальной частицы бесконечно малой массы в поле тяготения центрального тела с массой M под действием сил, определенных потенциальными функциями V и R, а также совокупности сил P, не имеющих потенциала, и записать в виде

$$\frac{d\mathbf{x}}{dt} = \dot{\mathbf{x}}, \quad \frac{d\dot{\mathbf{x}}}{dt} = Q\frac{\partial V}{\partial \mathbf{x}} + \frac{\partial R}{\partial \mathbf{x}} + \mathbf{P} \tag{1}$$

с начальными условиями

$$\mathbf{x}_0 = \mathbf{x}(t_0), \quad \dot{\mathbf{x}}_0 = \dot{\mathbf{x}}(t_0), \tag{2}$$

причем Q – матрица перехода из вращающейся системы координат в инерциальную систему [1].

[1] Работа выполнена по заданию Министерства образования и науки РФ № 8.4859.2011 и при частичной финансовой поддержке РФФИ, грант 11-02-00918-а

Обратная задача динамики ИСЗ и алгоритм ее решения могут быть сформулированы следующим образом. Пусть $\rho_j = \rho_j(q_i)$ – измеренные величины, а q_i – определяемые параметры, связь между которыми через решение уравнений (1) задается нелинейным соотношением

$$\rho_j = F_j(q_i),\ j = 1,2,...N,\ i = 1,2....m,$$

причем $j \gg i$. Следовательно, мы имеем избыточную нелинейную систему уравнений для определения неизвестных параметров. Решение этой системы возможно при условии существования минимума функции

$$\Phi(q) = \sum_{i=1}^{N} \left[\Delta\rho_i(q)\right]^2 = \min, \qquad (3)$$

где $\Delta\rho_i = \rho_o - \rho_c$ – невязки, представляющие собой разность наблюденных и вычисленных значений измеряемых величин. Задачу минимизации (3) принято называть задачей наименьших квадратов (НК). Решение нелинейной задачи НК производится методом Гаусса–Ньютона, путем нахождения дифференциальных поправок Δq в определяемые элементы.

Итерационный процесс считается завершенным при выполнении следующих условий:

$$\left|q_i^{n+1} - q_i^n\right| \prec \varepsilon, \qquad (4)$$

где ε – заданная точность вычислений.

Вычисление обратной матрицы осуществляется методом сингулярного анализа [2].

Особенности программной реализации алгоритма

При решении задач динамики небесных тел в среде параллельных вычислений используются два способа реализации: функциональная декомпозиция задачи и декомпозиция системы объектов. В первом случае параллельно вычисляются модели сил, действующих на объект. При таком подходе распараллеливания неизбежно возникает проблема балансировки, т. е. обеспечения равномерного распределения вычислительной нагрузки между

параллельными процессами. Кроме того, адаптация программного комплекса для суперкомпьютера может занять значительное время.

Во втором подходе система объектов разделяется на подсистемы, число которых равно числу процессов, что позволяет без существенных трудозатрат на модификацию программного кода применить программный комплекс для решения прямых и обратных задач.

Нами было сделано сравнение обоих этих способов решения обратных задач по быстродействию и эффективности.

Параллельное решение обратной задачи с помощью функциональной декомпозиция, реализованное в разработанном программном комплексе, позволяет задействовать не более четырех ядер (рис. 1).

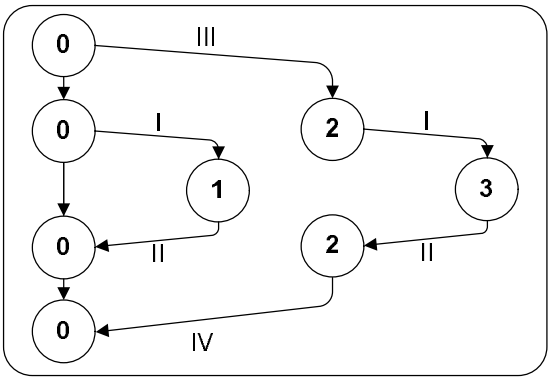


Рис. 1. Блок-схема программного комплекса. Этап I, II – обмен данными в функции правых частей; III, IV – обмен входными и выходными данными

Нами были рассмотрены промежуточные варианты реализации задачи для одного, двух и трех ядер. Использование одного потока для решения обратной задачи приведено в качестве сравнения с работой на персональном компьютере. Распараллеливание задачи на два потока позволило интегрировать уравнения (1) с прямым и обратным шагом. Решение задачи с использованием трех ядер позволило выделить отдельный поток для вычисления модели сил для первого и второго ядер, что приводит к сильной разбалансировке. Эта проблема решается с выделением еще одного ядра, и, таким образом, количество потоков увеличивается до четырех.

Результаты этих исследований приведены в табл. 1. В качестве примера было рассмотрено время решения обратной задачи для ИСЗ Лагеос на одной итерации.

Таблица 1.Зависимость времени работы от количества используемых ядер

Количество ядер	Время решения обратной задачи, сек
1	33,65
2	16,88
3	16,11
4	12,11

Видно, что время выполнения программного комплекса уменьшается и минимальное время достигается при решении задачи на четырех ядрах. Применение трех ядер для решения этой же задачи является нецелесообразным, так как на третье ядро ложится значительная вычислительная нагрузка, что вызывает ожидания на первых двух потоках. Это приводит к разбалансировке в решаемой задаче, а использование четырех ядер устраняет эту проблему. Дальнейшее увеличение количества ядер в рассматриваемой задаче приведет к большой разбалансировке и создаст дополнительные сложности в реализации программного комплекса. К тому же явным преимуществом решения задачи на четырех ядрах по сравнению с решением на большем количестве потоков является выполнение задачи на одном четырехъядерном узле без использования протокола пересылки данных между узлами, что тоже сильно влияет на быстродействие. Иногда в таких реализациях можно получить значительное ускорение, связанное с пересылкой информации через кэш-память процессора, не используя даже оперативную память узла.

Нами для полноты исследования параллельной реализации обратной задачи в небесной механике была внедрена в программный комплекс декомпозиция системы объектов. Так как в программном комплексе изохронные производные находятся численно варьированием соответствующей компоненты вектора состояния, система состоит из восьми траекторий: одна изначальная траектория объекта, шесть варьированных траекторий по координатам и скоростям и одна траектория, которая используется для определения ускорения вдоль орбиты объекта. Используя эту информацию, программный комплекс сначала был разделен на восемь ядер (рис. 2), а потом, чтобы корректно сравнить с

предыдущим исследованием, на четыре ядра – по две траектории на поток.

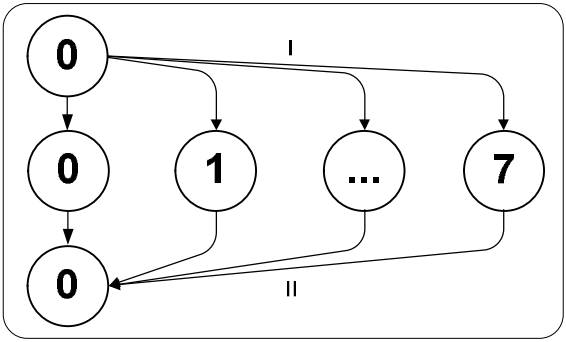


Рис 2. Информационный граф программного комплекса. I, II – обмен входными и выходными данными

Структурно информационные графы для четырех и восьми ядер не отличаются друг от друга, поэтому мы приводим только информационный граф для восьми ядер. На начальном этапе инициализации внутренних переменных программного комплекса происходит пересылка данных для нужного ядра (I), где выполняется независимое интегрирование уравнений движений (1). Когда моменты наблюдений совпадают с шагом интегрирования, происходит сборка информации по изохронным производным на нулевом ядре (II), где в дальнейшем, после интегрирования определяется новый вектор состояния.

После реализации двух совершенно разных подходов для решения обратных задач было проведено сравнение быстродействия (табл. 2).

Таблица 2. Сравнение быстродействия двух реализаций

Номер алгоритма	Время выполнения, сек	
	На 64-битной разрядной сетке	На 128-битной разрядной сетке
1	12,11	231,4
2	43,78	875,6
3	34,10	682,1

В табл. 2 номера алгоритмов означают: 1 – параллельная реализация функциональной декомпозиции на четырех потоках; 2 –

распределение системы объектов на четыре потока; 3 – распределение системы объектов на восемь потоков. Время в табл. 2 измерено для одной итерации. Видим, что программный комплекс, использующий параллельное вычисление модели сил, показывает быстродействие в три-четыре раза выше, чем другие реализации. Такое быстродействие можно объяснить тем, что информация между потоками обменивается через кэш-память процессора.

Результаты исследования показывают, что не всегда простая и быстрая параллельная реализации задач может принести эффективную работу приложения. Иногда нужно учитывать особенности программной реализации программного комплекса и аппаратной конфигурации кластера.

Литература

1. *Бордовицына Т.В., Авдюшев В.А.* Теория движения ИСЗ. Аналитические и численные методы. Томск: Изд-во Том. ун-та, 2007. 105 с.
2. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. М.: Мир, 1980. 279 с.
3. *Немнюгин С.А., Стесик О.Л.* Параллельное программирование для многопроцессорных вычислительных систем. СПб.: БХВ-Петербург, 2002.

Реализация модели многочастичного газа FHP-MP на гибридном кластере

А.С. Подстригайло

Институт вычислительной математики и математической геофизики СО РАН, Новосибирск
Новосибирский государственный университет

Введение

Для имитационного моделирования потоков жидкости и газа в 70-х годах прошлого века было разработано семейство дискретных клеточно-автоматных моделей Lattice Gas. Эти модели имеют ряд ограничений, в частности, граничные условия позволяют задавать только неподвижные твердые объекты (стенки), моделирование околозвуковых скоростей влечет искажение результата и т.д.

Для решения этих проблем в ИВМиМГ СО РАН была предложена новая КА модель, названная FHP-MP, подробно описанная в работе [1]. Она является обобщением классической булевой модели FHP, в которой допускается более одной частицы по каждому направлению скорости. Приведем краткое описание модели: пространство моделирования представляет собой гексагональную решетку, состояние каждой клетки представлено вектором целых чисел, частицы покоя в рассматриваемой модели отсутствуют. Процесс моделирования – итеративный. Каждая итерация состоит из двух фаз: столкновения и движения. На фазе сдвига каждая частица, обладающая ненулевой скоростью, перемещается в смежную ячейку, соответствующую направлению ее скорости; при этом скорость частицы в новой ячейке остается прежней. На фазе столкновения необходимо случайно равновероятно выбрать одно из состояний вектора такое, что выполняются законы сохранения массы и импульса частиц.

Характерной особенностью модели является большая вычислительная сложность фазы столкновения. В то же время данный алгоритм обладает высоким уровнем параллелизма, что позволяет, в частности, использовать модель вычислений на GPU.

Постановка задачи

В рамках работы была поставлена задача реализовать модель FHP-MP для одного узла гибридного кластера с несколькими

графическими ускорителями. Данная задача разбивается на несколько этапов:

1. Реализация модели FHP-MP непосредственно на одном графическом ускорителе.
2. Анализ степени дисбаланса вычислений на видеокарте в рамках данной модели.
3. Реализация модели FHP-MP на одном узле гибридного кластера.
4. Разработка и реализация алгоритма балансировки нагрузки между видеокартами одного узла кластера.

Параллельная реализация модели FHP-MP на одном графическом ускорителе

Для проверки существования дисбаланса в рамках одной видеокарты было реализовано два алгоритма столкновений разной сложности: кубической [2] и линейной [3].

Поскольку штатные средства профилировки не позволяют изучить загрузку мультипроцессоров видеокарты, был поставлен следующий вычислительный эксперимент:

- рассматривались поля размером 1024x512, 2048x1024, 4096x2048 клеток;
- на расстоянии четверти поля от левого края была поставлена отражающая стенка, не доходящая до верха и низа поля – таким образом, что частицы способны обходить ее;
- верхняя и нижняя границы поля – отражающие, правая и левая – проникающие, т.е. частицы, вылетающие за правую границу, прилетают с левой и наоборот.

Для сравнения были поставлены серии экспериментов двух типов.

В первом случае на поле выделялась область с «плотным» заполнением, когда по каждому направлению в этой клетке вылетало по 1, 10 или 100 частиц, в зависимости от эксперимента; в оставшейся области в каждой клетке в трех случайных направлениях разлеталось по 1 частице (рис. 1).

В экспериментах второго типа область равномерно заполнялась частицами, разлетающимися в случайных направлениях так, чтобы их суммарное количество соответствовало общему числу частиц в эксперименте первого типа.

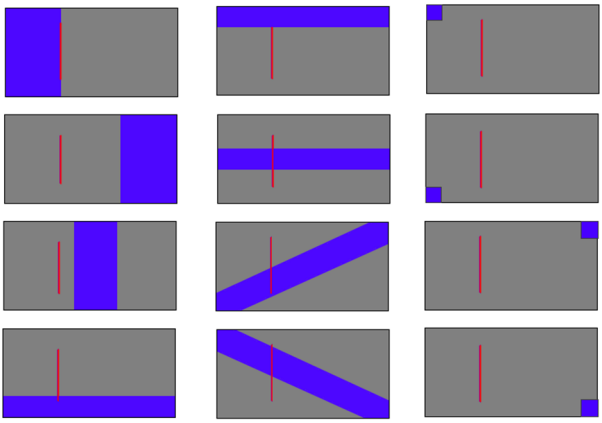


Рис. 1. Серия экспериментов I типа. Черным цветом показаны области с плотным заполнением, серым – с разреженным. Вертикальный отрезок показывает отражающую стенку

Приведем численные результаты экспериментов, рассчитанных на графическом ускорителе GTX 680, и сравним их с последовательной реализацией тех же алгоритмов на процессоре Core i7.

Заметим, что, хотя общее время обсчета модели при переходе к большей области закономерно возрастает, время обсчета одной клетки существенно не меняется, поэтому ограничимся результатами для области 1024x512 клеток.

Таблица 1. Численные результаты эксперимента I типа

Среднее время обсчета клетки (область 1024x512), мкс				
N	CPU-N3	CPU-N	GPU-N3	GPU-N
6	0,243	0,115	0,012	0,008
60	30,351	0,222	2,505	0,012
600	-	1,196	-	0,024

Таблица 2. Численные результаты эксперимента II типа

Среднее время обсчета клетки (область 1024x512), мкс				
M	CPU-N3	CPU-N	GPU-N3	GPU-N
3	0,200	0,114	0,009	0,008
17	5,408	0,236	0,328	0,011
152	-	1,215	-	0,020

Здесь N – количество частиц в области с плотным заполнением (в данном случае представлены результаты эксперимента, когда плотно заполнена четверть поля слева), M – количество частиц в каждой клетке при равномерном распределении суммарного количества частиц по области.

Прочерки в таблице объясняются тем, что общее время обсчета на CPU для кубического алгоритма на 50 итерациях уже для 60 частиц составляет порядка 15 мин, поэтому расчеты для 600 частиц были признаны нецелесообразными; соответствующие результаты для версии GPU также не вычислялись.

Таблица 3. Отношение времени вычислений экспериментов

Отношение времени вычислений			
CPU-N3	CPU-N	GPU-N3	GPU-N
1,216	*1,006*	1,285	*1,007*
5,613	*0,939*	7,631	*1,105*
-	*0,984*	-	*1,197*

Рассматривая отношения времени вычисления первого эксперимента ко второму (табл. 3), можно заметить, что следующая величина для CPU и GPU (выделено курсивом) почти не различается, что говорит о том, что в эксперименте первого типа на GPU нет дисбаланса вычислений: на CPU ввиду последовательной реализации дисбаланса очевидно нет, а если бы был дисбаланс на GPU, то эксперимент первого типа ускорился бы в разы меньше, чем второго, что на практике не реализуется.

Таким образом, было установлено, что в случае модели FHP-MP при вычислениях на одной видеокарте не наблюдается дисбаланса вычислительной нагрузки, т.е. планировщик блоков потоков справляется с распределением задачи по вычислительным ядрам

видеокарты самостоятельно и дополнительной балансировки нагрузки в рамках одного ускорителя не требуется.

Для дальнейшего развития модели был выбран алгоритм столкновений с линейной сложностью.

Параллельная реализация модели FHP-MP на узле гибридного кластера

В табл. 4 представлены результаты обсчета модели, расширенной для выполнения на нескольких видеокартах в рамках одного узла кластера НКС-30Т+GPU. Нижняя строчка таблицы показывает, что использование всех трех видеокарт дает существенное увеличение производительности.

Таблица 4. Численные результаты эксперимента I типа

Среднее время обсчета клетки (область 4096x2048), мкс				
N	CPU-N	GPU-N		
		Fermi x1	Fermi x2	Fermi x3
6	0,149	0,009	0,005	0,004
60	0,289	0,011	0,006	0,005
600	1,673	0,042	0,021	0,014

Однако существует ряд экспериментов, когда основная вычислительная нагрузка приходится на одну видеокарту, в этом случае, как показано в табл. 5, увеличения производительности от использования дополнительных графических ускорителей не наблюдается.

Таблица 5. Численные результаты эксперимента I типа для области с плотным заполнением частиц в нижней четверти поля

Среднее время обсчета клетки (область 4096x2048), мкс			
N	GPU-N		
	Fermi x1	Fermi x2	Fermi x3
6	0,007	0,005	0,004
60	0,010	0,008	0,006
600	0,038	0,036	0,034

Для компенсации этого факта был реализован следующий алгоритм динамической балансировки между видеокартами.

Во время выполнения итерации подсчитывается суммарное количество частиц по всем направлениям в каждой строке М. На основе этих данных можно рассчитать теоретическое «эталонное» количество частиц Мт, равное М, деленное на количество графических ускорителей в узле, которое нужно отдать каждой видеокарте для балансировки нагрузки. Опираясь на эту границу и зная, сколько «весит» в частицах каждая строка, алгоритм отдает для обсчета каждой видеокарте столько последовательных строк, чтобы разница между теоретическим Мт и практическим Мп была минимальна.

В табл. 6 представлены численные результаты того же эксперимента, что и в табл. 5, с условием включенной динамической балансировки. Видно, что алгоритм хорошо компенсирует замеченный дисбаланс нагрузки между графическими ускорителями в узле.

Таблица 6. Численные результаты эксперимента I типа для области с плотным заполнением частиц в нижней четверти поля, с балансировкой

Среднее время обсчета клетки (область 4096x2048), мкс			
N	GPU-N		
	Fermi x1	Fermi x2	Fermi x3
6	0,007	0,005	0,004
60	0,010	0,008	0,006
600	0,038	0,024	0,014

Заключение

В работе представлена реализация модели FHP-MP для нескольких видеокарт в рамках одного узла гибридного кластера. Был разработан и реализован алгоритм динамической балансировки нагрузки для данной модели между графическими ускорителями в одном узле, который позволяет компенсировать возникающий в процессе вычислений дисбаланс нагрузки видеокарт.

Литература

1. *Медведев Ю.Г.* Многочастичная клеточно-автоматная модель потока жидкости FHP-MP // Вестник Томского государственного университета, серия «Управление,

вычислительная техника и информатика». 2009. №1(6). С. 33–40.
2. *Kalgin K.* Optimization of transition rule computation algorithm in multiparticle lattice gas. // Proc. of Combinatorics. 2010. P.148.
3. *Дубовик А.С.* // Труды конференции «Новые информационные технологии в исследовании сложных структур». 2012.

Математическое моделирование процессов самоочищения реки с использованием модификации моделей Герберта и Стритера – Фелпса

Д.Г. Абеляшев, М.Д. Михайлов

Томский государственный университет, Томск

Проводится численное исследование модификации моделей Герберта и Стритера – Фелпса, переменными которых служат концентрация органического вещества L (в единицах БПК), концентрации микроорганизмов X и кислорода D.

Поступающие в водоем загрязнения вызывают в нем нарушение естественного равновесия. Способность водоема противостоять этому нарушению, освобождаться от вносимых загрязнений и составляет сущность процесса самоочищения, представляющего собой сложный комплекс физических, физико-химических, химических и биохимических явлений. Главную роль в окислении растворенных органических веществ играют бактерии. При этом видовой состав микроорганизмов определяется характером внесенных загрязнений. В воде развиваются виды, способные использовать те или иные загрязняющие вещества в качестве источников питания.

Вопросы описания роста микробных популяций давно привлекают внимание ученых. В конце 40-х годов 20-го столетия французский учёный Жак Моно обнаружил и обосновал связь между скоростью роста микроорганизмов и концентрацией лимитирующего субстрата в среде [1]. После этих исследований работы по моделированию роста культур микроорганизмов получили достаточно широкое распространение.

Приведём одну из модификаций модели Моно – модель Герберта – замкнутую систему кинетических уравнений, описывающих рост биомассы X на лимитирующем субстрате L [2]:

$$\begin{cases} \dfrac{dL}{dt} = -\dfrac{1}{Y}\mu(L)\,X, \\ \dfrac{dX}{dt} = \mu(L)\,X - b\,X, \\ L(0) = L_0\,,\, X(0) = X_0\,. \end{cases} \qquad (1)$$

Здесь $\mu(L) = \dfrac{\mu_m L}{K_L + L}$, b – константа скорости отмирания бактерий; μ_m – максимальная удельная скорость роста микроорганизмов, сут$^{-1}$; K_L – константа полунасыщения, равная концентрации субстрата L, при которой скорость процесса равна половине удельной максимальной, мг/л; Y – экономический коэффициент, показывающий, какая часть поглощенного субстрата идет на построение биомассы бактерий. Здесь при $b = 0$ система уравнений (1) представляет собой классическую модель Моно.

Ещё одной важной характеристикой для водных экосистем является содержание в воде растворенного кислорода. Даже небольшие изменения его концентрации в воде могут приводить к необратимым последствиям для всей экосистемы [3].

Рассмотрим кислородный баланс реки в условиях ее загрязнения легко окисляемыми отходами, например, органического происхождения. При попадании таких отходов в воду начинается процесс их биохимического разложения, который протекает с использованием растворенного в воде кислорода, что может привести к падению его концентрации в воде.

Исследуем взаимосвязь между концентрациями растворенного кислорода и органических отходов. Концентрация отходов измеряется величиной так называемого биохимического потребления кислорода (БПК), которая представляет собой количество кислорода на единицу объема воды, необходимое для разложения содержащихся в этом объеме органических примесей. Единица измерения БПК – мг/л O_2.

Опишем процессы, формирующие кислородный баланс реки при наличии загрязнения. Введем обозначения:

L – биохимическое потребление кислорода (БПК), мг/л, D_s – концентрация кислорода в воде при отсутствии отходов, мг/л, D –

концентрация кислорода в воде при наличии отходов, мг/л, K_1 – скорость потребления кислорода на разложение отходов, 1/сутки, K_2 – скорость реаэрации, 1/сутки.

В этих обозначениях процессы разложения отходов и формирования кислородного баланса в водоеме описываются следующими простыми уравнениями, получившими название модели Стритера – Фелпса, по именам ученых, впервые использовавших их для анализа такой ситуации:

$$\begin{cases} \dfrac{dL}{dt} = -K_1 L, \\ \dfrac{dD}{dt} = -K_1 L + K_2 \cdot \left(D_s - D\right), \\ L\left(0\right) = L_0, D\left(0\right) = D_S. \end{cases} \qquad (2)$$

Скорость разложения отходов пропорциональна их концентрации в воде при условии, что кислорода достаточно для их полного разложения. При отсутствии отходов концентрация кислорода в воде колеблется около равновесного (максимального для данных условий) значения. При наличии отходов реальная концентрация кислорода будет снижаться. Однако существуют процессы, обеспечивающие увеличение концентрации кислорода в воде, например реаэрация – растворение кислорода атмосферы в воде через поверхностный слой.

Для учёта обоих характеристик будем использовать модель, полученную путём объединения двух приведённых выше моделей:

$$\begin{cases} \dfrac{dL}{dt} = -\dfrac{1}{Y}\dfrac{\mu_m X L}{K_L + L} - K_1 L, \\ \dfrac{dX}{dt} = \dfrac{\mu_m X L}{K_L + L} - b X, \\ \dfrac{dD}{dt} = -K_1 L + K_2 \left(D_s - D\right), \\ L(0) = L_0,\ X(0) = X_0,\ D(0) = D_0,\ t \in \left[0, T\right]. \end{cases} \qquad (3)$$

Модификация построенной модели (3) на двумерный случай заключается в добавлении в данную систему оператора диффузии

$$\Gamma = d_1 \frac{\partial^2}{\partial x^2} + d_2 \frac{\partial^2}{\partial y^2}$$

и конвективного оператора

$$\Delta = V_1 \frac{\partial}{\partial x} + V_2 \frac{\partial}{\partial y}.$$

В результате модель принимает вид системы дифференциальных уравнений в частных производных

$$\begin{cases} \dfrac{\partial L}{\partial t} + V_1 \dfrac{\partial L}{\partial x} + V_2 \dfrac{\partial L}{\partial y} = d_1 \dfrac{\partial^2 L}{\partial x^2} + d_2 \dfrac{\partial^2 L}{\partial y^2} - \dfrac{1}{Y} \dfrac{\mu_m X L}{K_L + L} - K_1 L, \\ \dfrac{\partial X}{\partial t} + V_1 \dfrac{\partial X}{\partial x} + V_2 \dfrac{\partial X}{\partial y} = d_1 \dfrac{\partial^2 X}{\partial x^2} + d_2 \dfrac{\partial^2 X}{\partial y^2} + \dfrac{\mu_m X L}{K_L + L} - b X, \\ \dfrac{\partial D}{\partial t} + V_1 \dfrac{\partial D}{\partial x} + V_2 \dfrac{\partial D}{\partial y} = d_1 \dfrac{\partial^2 D}{\partial x^2} + d_2 \dfrac{\partial^2 D}{\partial y^2} - K_1 L + K_2 \left(D_S - D \right) \end{cases} \tag{4}$$

с соответствующими начальными

$$\begin{aligned} L(x, y, 0) &= L_0(x, y), \\ X(x, y, 0) &= X_0(x, y), \\ D(x, y, 0) &= D_0(x, y) \end{aligned} \tag{5}$$

и граничными

$$\begin{aligned} &\left.\frac{\partial L}{\partial x}\right|_{x=a} = \left.\frac{\partial L}{\partial x}\right|_{x=b} = 0, \ \left.\frac{\partial L}{\partial y}\right|_{y=c} = \left.\frac{\partial L}{\partial y}\right|_{y=d} = 0, \\ &\left.\frac{\partial X}{\partial x}\right|_{x=a} = \left.\frac{\partial X}{\partial x}\right|_{x=b} = 0, \ \left.\frac{\partial X}{\partial y}\right|_{y=c} = \left.\frac{\partial X}{\partial y}\right|_{y=d} = 0, \\ &\left.\frac{\partial D}{\partial x}\right|_{x=a} = \left.\frac{\partial D}{\partial x}\right|_{x=b} = 0, \ \left.\frac{\partial D}{\partial y}\right|_{y=c} = \left.\frac{\partial D}{\partial y}\right|_{y=d} = 0 \end{aligned} \tag{6}$$

условиями, где V_1, V_2 – компоненты вектора скорости течения реки, км/сут; d_1, d_2 – коэффициенты диффузии.

Численная реализация модели (4) – (6) осуществляется с помощью явной разностной схемы вида:

$$
\begin{cases}
\dfrac{L_{j,k}^{n+1}-L_{j,k}^{n}}{\tau}+V_1\dfrac{L_{j,k}^{n}-L_{j-1,k}^{n}}{h_x}+V_2\dfrac{L_{j,k}^{n}-L_{j,k-1}^{n}}{h_y}=d_1\dfrac{L_{j+1,k}^{n}-2L_{j,k}^{n}+L_{j-1,k}^{n}}{h_x^{\,2}}+ \\
+d_2\dfrac{L_{j,k+1}^{n}-2L_{j,k}^{n}+L_{j,k-1}^{n}}{h_y^{\,2}}-\dfrac{1}{Y}\dfrac{\mu_m\,L_{j,k}^{n}X_{j,k}^{n}}{K_L+L_{j,k}^{n}}-K_1L_{j,k}^{n}, \\
\dfrac{X_{j,k}^{n+1}-X_{j,k}^{n}}{\tau}+V_1\dfrac{X_{j,k}^{n}-X_{j-1,k}^{n}}{h_x}+V_2\dfrac{X_{j,k}^{n}-X_{j,k-1}^{n}}{h_y}=d_1\dfrac{X_{j+1,k}^{n}-2X_{j,k}^{n}+X_{j-1,k}^{n}}{h_x^{\,2}}+ \\
+d_2\dfrac{X_{j,k+1}^{n}-2X_{j,k}^{n}+X_{j,k-1}^{n}}{h_y^{\,2}}+\dfrac{\mu_m\,L_{j,k}^{n}X_{j,k}^{n}}{K_L+L_{j,k}^{n}}-bX_{j,k}^{n}, \\
\dfrac{D_{j,k}^{n+1}-D_{j,k}^{n}}{\tau}+V_1\dfrac{D_{j,k}^{n}-D_{j-1,k}^{n}}{h_x}+V_2\dfrac{D_{j,k}^{n}-D_{j,k-1}^{n}}{h_y}=d_1\dfrac{D_{j+1,k}^{n}-2D_{j,k}^{n}+D_{j-1,k}^{n}}{h_x^{\,2}}+ \\
+d_2\dfrac{D_{j,k+1}^{n}-2D_{j,k}^{n}+D_{j,k-1}^{n}}{h_y^{\,2}}-K_1L_{j,k}^{n}+K_2\cdot\left(D_s-D_{j,k}^{n}\right), \\
j=\overline{1,N_1-1},\ k=\overline{1,N_2-1},\ n=\overline{0,M-1},
\end{cases}
$$

с начальными и граничными условиями следующего вида:

$$
\begin{aligned}
&L_{j,k}^{0}=L^{0}\left(x_j,y_k\right),\ X_{j,k}^{0}=X^{0}\left(x_j,y_k\right),\ D_{j,k}^{0}=D^{0}\left(x_j,y_k\right); \\
&L_{0,k}^{n}=L_{1,k}^{n},\ L_{N_1,k}^{n}=L_{N_1-1,k}^{n},\ L_{j,0}^{n}=L_{j,1}^{n},\ L_{j,N_2}^{n}=L_{j,N_2-1}^{n}, \\
&X_{0,k}^{n}=X_{1,k}^{n},\ X_{N_1,k}^{n}=X_{N_1-1,k}^{n},\ X_{j,0}^{n}=X_{j,1}^{n},\ X_{j,N_2}^{n}=X_{j,N_2-1}^{n}, \\
&D_{0,k}^{n}=D_{1,k}^{n},\ D_{N_1,k}^{n}=D_{N_1-1,k}^{n},\ D_{j,0}^{n}=D_{j,1}^{n},\ D_{j,N_2}^{n}=D_{j,N_2-1}^{n}, \\
&j=\overline{0,N_1},\ k=\overline{0,N_2},\ n=\overline{1,M}.
\end{aligned}
$$

Разностная схема в линейном приближении исследовалась на устойчивость методом гармоник по начальным данным. В ходе

исследований было показано, что она является условно устойчивой, если $\tau \le 1/\left(\frac{V_1}{h_x}+\frac{V_2}{h_y}+2\frac{d_1}{h_x^2}+2\frac{d_2}{h_y^2}\right)$.

Кроме того, показано, что разностная схема аппроксимирует исходную дифференциальную задачу с первым порядком относительно шагов по пространству h_x, h_y и шагом по времени τ в норме пространства C_h. Из этого следует, что решение разностной задачи сходится к решению дифференциальной задачи (4) – (6).

Численная реализация данной задачи проводилась на ПЭВМ. Для этого была написана программа на языке С. При построении графиков использовался пакет MatLab.

Объектом исследования являлся участок реки Селенги от г. Улан-Уде до Кабанска длиной 50 км и шириной 600 м. В начальный момент область загрязнения составляла 10 км в длину и 60 м в ширину и его концентрация равнялась 9,7 мг/л. На всей площади реки концентрация кислорода составляла 8 мг/л, а концентрация микроорганизмов – 1,2 мг/л. Расчётная область покрывалась сеткой 100×100 точек.

На рис. 1 представлен график концентрации органического вещества L в момент времени $t = 5 сут$. К этому моменту времени загрязнитель распространился на всю расчётную область и максимальное его значение равнялось 0,1895 мг/л, что значительно меньше первоначального загрязнения.

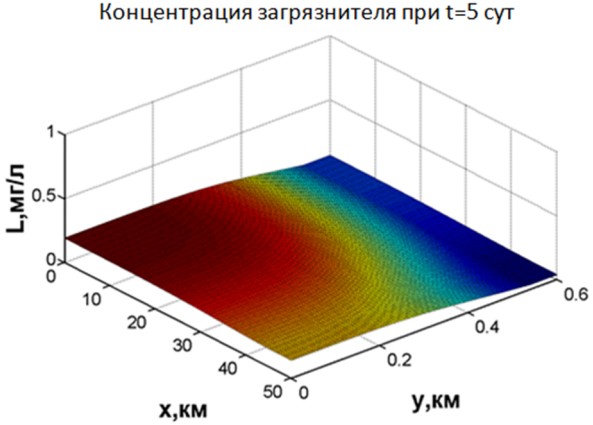


Рис. 1

Процесс очищения водной среды происходит путём окисления органического вещества как с помощью кислорода, так и с помощью бактерий.

При попадании загрязнителя в водную среду происходит его окисление содержащимся в воде кислородом, тем самым его концентрация падает. На рис. 2 представлен график концентрации кислорода D на 5-е сутки. Из графика видно, что в области с большей концентрацией загрязнения значение концентрации кислорода меньше, чем в области с меньшим загрязнением. Одновременно с кислородом органическое вещество окисляется и бактериями.

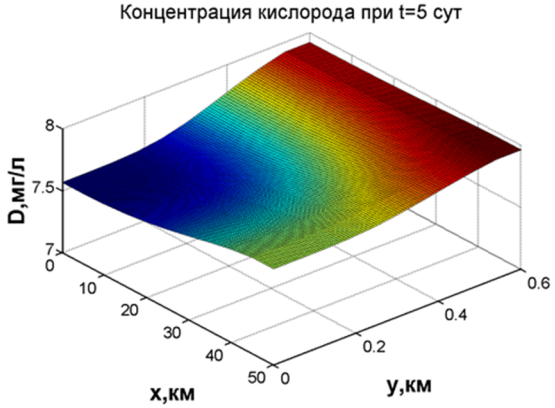


Рис. 2

На рис. 3 представлено распределение их концентрации на всём протяжении реки в тот же момент времени. Из графика видно, что концентрация микроорганизмов больше там, где больше концентрация загрязнения.

К 10-м суткам загрязнение распространяется на всю расчётную область, и к этому моменту времени достигается его предельно допустимая концентрация (ПДК). Концентрация кислорода при отсутствии отходов начнёт восстанавливаться, начиная с момента 2,7 сут, до своего первоначального равновесного состояния 8 мг/л.

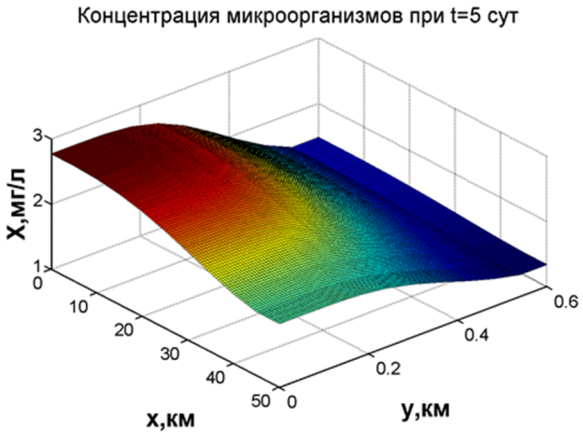


Рис. 3

К моменту времени 7,7 сут количество бактерий достигает своего максимума – 2,78 мг/л, а дальше наблюдается сокращение популяции бактерий, которая в дальнейшем установится на некотором постоянном уровне. Достоверность полученных результатов была подтверждена сравнением с экспериментальными данными из [4].

Литература

1. *Иерусалимский Н.Д.* Основы физиологии микробов. М. : Изд. АН СССР, 1963. 242 с.

2. *Хубларян М.Г.* Водные потоки: модели течений и качества суши. М. : Наука, 1991. 192 с.

3. *Вавилин В.А.* Нелинейные модели биологической очистки и процессов самоочищения в реках. М. : Наука, 1981. 160 с.

4. *Фундаментальные* проблемы воды и водных ресурсов // Материалы Третьей всероссийской конференции с международным участием. Введ. 2010-08-24. Барнаул : Изд-во АРТ, 2010. С.14–17.

Обобщение метода погранфункций для бисингулярно возмущенных эллиптических уравнений в случае, когда особенность появляется на границе и в центре круга

К. Алымкулов, Т.Д. Турсунов

Ошский государственный университет, Ош, Кыргызстан

Рассматривается задача Дирихле для бисингулярно возмущенного эллиптического уравнения. Для решения задачи построено равномерное асимптотическое разложение.

При математическом моделировании процессов конвективно-диффузионного переноса, химической кинетики и др. возникают краевые задачи для уравнений эллиптического типа второго порядка с малым параметром при старших производных. Явное решение этих задач построить в общем случае не удается, поэтому используют разные асимптотические методы. Основополагающими в этом направлении являются работы А.Н. Тихонова, А.Б. Васильевой, С.А. Ломова, В.Б. Бутузова, Л.И. Люстерника, М.И. Вишика, А.М. Ильина. В случае, когда соответствующее невозмущенное уравнение имеет негладкое решение, эти задачи, по терминологии А.М. Ильина [1], называют бисингулярными. Ранее для построения асимптотики бисингулярно возмущенных задач применялся метод сращивания, а метод пограничных функций не использовался напрямую. В работе предложена модификация метода пограничных функций [2], благодаря которой стало возможным построить асимптотику решения бисингулярно возмущенного эллиптического уравнения.

Целью исследования является развитие асимптотического метода пограничных функций для бисингулярно возмущенных задач. Применяя обобщенный метод пограничных функций, построено асимптотическое разложение решения бисингулярно возмущенного эллиптического уравнения в случае, когда предельное уравнение имеет особенность на граничных точках области. Задача рассматривается в круге.

Постановка задачи

Рассмотрим задачу Дирихле для бисингулярно возмущенного эллиптического уравнения

$\varepsilon\Delta u-(1-\rho)\rho u=f(\rho,\varphi,\varepsilon)$, $(\rho,\varphi)\in D=\{(\rho,\varphi)|0\le\rho<1,\ 0\le\varphi<2\pi\}$, (1)

$u(1,\varphi,\varepsilon)=\psi(\varphi,\varepsilon)$, (2)

где $\Delta=\dfrac{\partial^2}{\partial\rho^2}+\dfrac{1}{\rho}\dfrac{\partial}{\partial\rho}+\dfrac{\partial^2}{\partial\varphi^2}$ – оператор Лапласа, $\psi(\varphi,\varepsilon)=\sum\limits_{k=0}^{\infty}\psi_k(\varphi)\varepsilon^k$, $\psi(\varphi)\in C^{\infty}[0,\ 2\pi]$, $\psi(\varphi,\varepsilon)$, $f(\rho,\varphi,\varepsilon)$ – заданные функции, $u=u(\rho,\varphi,\varepsilon)$ – искомая функция.

(Условие **U**) Пусть $f(\rho,\varphi,\varepsilon)=\sum\limits_{k=0}^{\infty}f_k(\rho,\varphi)\varepsilon^k$, $f_k(\rho,\varphi)\in C^{(\infty,\infty)}_{(\rho,\varphi)}(\overline{D})$, $f(1,\varphi,0)\ne0$, $f(0,\varphi,0)\ne0$.

Как всегда, сначала рассмотрим структуру внешнего разложения решения задачи (1), которое ищем в виде:

$$V=\sum_{k=0}^{\infty}\varepsilon^k v_k(\rho,\varphi),\quad \varepsilon\to0. \tag{3}$$

После подстановки (3) в (1) и приравнивания коэффициентов при одинаковых степенях ε получим рекуррентную систему уравнений:

$-(1-\rho)\rho v_0(\rho,\varphi)=f_0(\rho,\varphi)$,

$(1-\rho)\rho v_k(\rho,\varphi)=\Delta v_{k-1}(\rho,\varphi)-f_k(\rho,\varphi)$, $k\in\mathbf{N}$.

Отсюда однозначно определяются все $v_k(\rho,\varphi)\in C^{(\infty,\infty)}_{(\rho,\varphi)}(D\setminus(0,0))$:

$v_0(\rho,\varphi)=-f_0(\rho,\varphi)/(1-\rho)\rho$,

$v_k(\rho,\varphi)=(\Delta v_{k-1}(\rho,\varphi)-f_k(\rho,\varphi))/(1-\rho)\rho$, $k\in\mathbf{N}$,

значит,

$$V=-\frac{f_0(\rho,\varphi)}{(1-\rho)\rho}+\frac{\varepsilon}{((1-\rho)\rho)}F_1(\rho,\varphi)+\ldots+\frac{\varepsilon^n}{((1-\rho)\rho)^{3n+1}}F_n(\rho,\varphi)+\ldots,\ \varepsilon\to0,$$

$F_k(\rho,\varphi)\in C^{(\infty,\infty)}_{(\rho,\varphi)}(\overline{D})$, $k\in\mathbf{N}$.

Заметим, что при $\rho=1$ и $\rho=0$ на границе круга D и в центре круга, все эти функции $v_k(\rho,\varphi)$ имеют нарастающие особенности:

$v_k(\rho,\varphi)=O(1/((1-\rho)\rho)^{3k+1})$.

Построение ФАР решения

Решение задачи (1) – (2) будем искать в виде

$u=v_0(\rho,\varphi)+\pi_{-1}(\tau,\varphi)/\mu+\pi_0(\tau,\varphi)+w_{-1}(\eta,\varphi)/\mu+w_0(\eta,\varphi)+R(\rho,\varphi)$, (4)

где $\tau=(1-\rho)/\mu, \eta=\rho/\mu$, $\varepsilon=\mu^3$.

Подставляя (4) в (1), получим:

$$\varepsilon\Delta v_0-(1-\rho)\rho v_0+\frac{\partial^2\pi_{-1}}{\partial\tau^2}-\tau\pi_{-1}+\mu\left(\frac{\partial^2\pi_0}{\partial\tau^2}-\tau\pi_0-\frac{\partial\pi_{-1}}{\partial\tau}+\tau^2\pi_{-1}\right)+\Delta_{\eta\varphi}w_{-1}-$$
$$-\eta w_{-1}+\mu\Delta_{\eta\varphi}w_0-\eta w_0+\eta^2 w_0+O(\mu^2)+\varepsilon\Delta R-(1-\rho)\rho R=f_0(\rho,\varphi)+O(\varepsilon)-$$
$$-H(\rho,\varphi)+H(\rho,\varphi), \qquad (5)$$

где $\Delta_{\eta\varphi}=\frac{\partial^2}{\partial\eta^2}+\frac{1}{\eta}\frac{\partial}{\partial\eta}+\frac{\partial^2}{\eta^2\partial\varphi^2}$.

Из граничного условия (2) имеем

$v_0(1,\varphi)+\pi_{-1}(0,\varphi)/\mu+\pi_0(0,\varphi)+w_{-1}(1/\mu,\varphi)/\mu+w_0(1/\mu,\varphi)+R(1,\varphi)=$
$=\psi_0(\varphi)+O(\varepsilon)$.

Отсюда получим:

$$\pi_{-1}(0,\varphi)=0, \qquad (6)$$
$$\pi_0(0,\varphi)=\psi_0(\varphi)-v_0(1,\varphi), \qquad (7)$$
$$w_{-1}(1/\mu,\varphi)=0, \qquad (8)$$
$$w_0(1/\mu,\varphi)=0, \qquad (9)$$
$$R(1,\varphi)=O(\varepsilon). \qquad (10)$$

Здесь в правую часть уравнения прибавили и убавили одну и ту же функцию $H(\rho,\varphi)$, которую определяем ниже.

Из (5) имеем

$-(1-\rho)\rho v_0=f_0(\rho,\varphi)-H(\rho,\varphi)$.

Отсюда

$v_0(\rho,\varphi)=-(f_0(\rho,\varphi)-H(\rho,\varphi))/(1-\rho)\rho$,

здесь мы определим неизвестную функцию $H(\rho,\varphi)$ так, чтобы $v_0(\rho,\varphi)\in C^{(\infty,\infty)}_{(\rho,\varphi)}(\overline{D})$. Отсюда $H_0(\rho,\varphi)=f_0(1,\varphi)\rho+f_0(0,\varphi)(1-\rho)$.

Следовательно,

$$v_0(\rho,\varphi)=-(f_0(\rho,\varphi)-f_0(1,\varphi)\rho-f_0(0,\varphi)(1-\rho))/(1-\rho)\rho, \qquad (11)$$
$$v_0(\rho,\varphi)\in C^{(\infty,\infty)}_{(\rho,\varphi)}(\overline{D}).$$

Учитывая условия (6)–(10), из (5) получим следующие задачи:

$$\frac{\partial^2\pi_{-1}}{\partial\tau^2}-\tau\pi_{-1}=f_0(1,\varphi),\ \pi_{-1}(0,\varphi)=0; \qquad (12)$$

$$\frac{\partial^2\pi_0}{\partial\tau^2}-\tau\pi_0=\frac{\partial\pi_{-1}}{\partial\tau}-(\tau\pi_{-1}+f_0(1,\varphi))\tau,\ \pi_0(0,\varphi)=\psi_0(\varphi)-v_0(1,\varphi)$$

или

$$\frac{\partial^2\pi_0}{\partial\tau^2}-\tau\pi_0=\frac{\partial\pi_{-1}}{\partial\tau}-\tau\ \frac{\partial^2\pi_{-1}}{\partial\tau^2},\ \pi_0(0,\varphi)=\psi_0(\varphi)-\nu_0(1,\varphi); \tag{13}$$

$$\Delta_{\eta\varphi}w_{-1}-\eta w_{-1}=f_0(0,\varphi),\ w_{-1}(1/\mu,\varphi)=0; \tag{14}$$

$$\Delta_{\eta\varphi}w_0-\eta w_0=\eta\Delta_{\eta\varphi}w_{-1},\ w_0(1/\mu,\varphi)=0; \tag{15}$$

$$\varepsilon\Delta R_0-(1-\rho)\rho R_0=-\varepsilon\Delta v_0+O(\varepsilon),\ R(1,\varphi)=O(\varepsilon). \tag{16}$$

Функции $\pi_{-1}(\tau,\varphi)$, $\pi_0(\tau,\varphi)$ являются обобщенными погранфункциями, поэтому на них накладываем дополнительные условия:

$\pi_{-1}(\tau,\varphi)\to 0$, $\pi_0(\tau,\varphi)\to 0$, при $\tau\to\infty$.

Если ввести оператор $l\equiv\frac{\partial^2}{\partial\tau^2}-\tau$, то имеем:

$$l\pi_{-1}=f_0(1,\varphi),\ \lim_{\tau\to\infty}\pi_{-1}(\tau,\varphi)=0,\ \pi_{-1}(0,\varphi)=0, \tag{17}$$

$$l\pi_0=\frac{\partial\pi_{-1}}{\partial\tau}-\tau\ \frac{\partial^2\pi_{-1}}{\partial\tau^2},\ \lim_{\tau\to\infty}\pi_0(\tau,\varphi)=0,\ \pi_0(0,\varphi)=\psi_0(1,\varphi)-\nu_0(1,\varphi). \tag{18}$$

Как нам известно, уравнение Эйри $z''-\tau z=0$ имеет два независимых решения $Ai(\tau)$ и $Bi(\tau)$.

Учитывая это, мы сможем записать решения задачи (17), (18):

$$\pi_{-1}(\tau,\varphi)=-\pi f_0(1,\varphi)\left(Ai(\tau)\int_0^\tau Bi(s)ds-\sqrt{3}Ai(\tau)\int_0^\infty Ai(s)ds+Bi(\tau)\int_\tau^\infty Ai(s)ds)\right),$$

$$\pi_0(\tau,\varphi)=-\pi(Ai(\tau)\left(\int_0^\tau Bi(s)\Phi(s,\varphi)ds-\sqrt{3}\int_0^\infty Ai(s)\Phi(s,\varphi)ds\right)+$$

$$+Bi(\tau)\int_\tau^\infty Ai(s)\Phi(s)ds)+\frac{\psi_0(1,\varphi)-\nu_0(1,\varphi)}{Ai(0)}Ai(\tau),$$

где $\Phi(\tau,\varphi)=\frac{\partial\pi_{-1}}{\partial\tau}-\tau\ \frac{\partial^2\pi_{-1}}{\partial\tau^2}$.

Асимптотика решения задачи (12), (13) при $\tau\to\infty$, имеет вид:

$\pi_{-1}(\tau,\varphi)=O(\tau^{-1})$,

$l\pi_0=O(\tau^{-2})\Rightarrow\pi_0(\tau,\varphi)=O(\tau^{-3})$.

Задачи (14) и (15) имеют единственные решения. При $\eta\to\infty$ имеем асимптотику:

$w_{-1}(\eta,\varphi)=O(\eta^{-1})$,

$$\frac{\partial^2 w_0}{\partial\eta^2}+\frac{\partial w_0}{\eta\partial\eta}+\frac{\partial^2 w_0}{\eta^2\partial\varphi^2}-\eta w_0=O(\eta^{-2})\Rightarrow w_0(\eta,\varphi)=O(\eta^{-3}).$$

Оценка остаточного члена $R(\rho,\varphi)$

Задачу (16) запишем в виде

$\varepsilon\Delta R-(1-\rho)\rho R=O(\varepsilon)$, $R(1,\varphi)=O(\varepsilon)$.

Отсюда получим

$|R(\rho,\varphi)|\le c\varepsilon^{2/3}$, $c>0$ – const.

Мы доказали следующую теорему.

Теорема. Пусть выполняется условие **U**, тогда для решения задачи (1) – (2) справедливо асимптотическое разложение

$u(\rho,\varphi,\varepsilon)=v_0(\rho,\varphi)+\pi_{-1}(\tau,\varphi)/\mu+\pi_0(\tau,\varphi)+w_{-1}(\eta,\varphi)/\mu+w_0(\eta,\varphi)+O(\varepsilon^{2/3})$,

где $\tau=(1-\rho)/\mu$, $\eta=\rho/\mu$, $\varepsilon=\mu^3$.

Литература

1. *Ильин А.М., Данилин А.Р.* Асимптотические методы в анализе. М.: Физматлит, 2009. 248 с.

2. *Alymkulov K.* Analog of Method of Boundary Layer Function for the Solution of the Lighthill's Model Equation with the regular Singular Point //American J.Math. & Statistics. 2013. V.3. N.1. P. 53–61.

Применение неявного итерационного полинейного рекуррентного метода для решения задачи о течении несжимаемой вязкой жидкости в плоской каверне

А.А. Фомин, Л.Н. Фомина

Кузбасский государственный технический университет, Кемерово
Кемеровский государственный университет, Кемерово

Рассматривается задача о течении несжимаемой вязкой жидкости в плоской каверне с подвижной верхней крышкой, которая движется с постоянной скоростью.

Целью настоящей работы является оценка эффективности применения неявного итерационного полинейного рекуррентного метода [1] при численном моделировании динамики несжимаемой вязкой жидкости на примере решения классической задачи о течении в плоской каверне с подвижной верхней крышкой. Данная задача удобна тем, что является своеобразным полигоном, на котором исследователи демонстрируют свои достижения в области численного моделирования динамики несжимаемой вязкой жидкости: новизну алгоритмов решения, аппроксимаций дифференциальной постановки задачи, методов решения СЛАУ, способов разрешения сингулярностей и разрывов в верхних угловых точках и многое другое. При этом необходимо заметить, что решение данной задачи давно уже вышло за рамки моделирования физического процесса, поскольку ещё в работе [2] было показано, что при числах Re ~ 1000 в подобных течениях возникают существенно трёхмерные неустойчивости, приводящие к турбулизации потока, в то время как большинство современных работ оперируют с числами Re от 1000 и выше, оставаясь при этом в рамках решения уравнений Навье–Стокса.

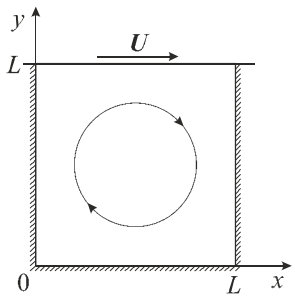


Рис. 1

На рис. 1 представлены система координат и схема течения рассматриваемой задачи. Математическая постановка, в которой в качестве искомого

решения выбрана тройка естественных параметров *u, v, p,* имеет следующий вид:

$$\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}=\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)-\frac{\partial p}{\partial x}; \quad (1)$$

$$\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}=\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right)-\frac{\partial p}{\partial y}; \quad (2)$$

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0. \quad (3)$$

На боковых и нижней границах имеют место условия

$$u=v=0;\frac{\partial p}{\partial n}=0; \quad (4)$$

на верхней границе

$$u=1;\quad v=0;\frac{\partial p}{\partial n}=0. \quad (5)$$

Здесь *x, y* – пространственные координаты, *t* – время, *u, v* – компоненты скорости $\vec{V}$ вдоль *x, y* координат соответственно, *p* – давление, *n* – нормаль к границе каверны; *Re* = *UL*/υ, где υ – кинематический коэффициент вязкости жидкости.

Расчетная область покрывается ортогональной сеткой, в узлах которой определяются элементы векторов решения. Разностная аппроксимация задачи (1) – (5) производится методом контрольного объема со вторым порядком точности по пространству и первым – по времени [3].

Алгоритм решения задачи представляет собой классическую трёхшаговую схему [4], на первом шаге которого определяются предварительные значения компонент скорости потока *u*, v** на базе градиентов давления с предыдущей итерации

$$u^*=u^n-\tau\left[u^{n-1}\frac{\partial u^*}{\partial x}+v^{n-1}\frac{\partial u^*}{\partial y}-\frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u^*}{\partial x^2}+\frac{\partial^2 u^*}{\partial y^2}\right)-\frac{\partial p^{n-1}}{\partial x}\right]; \quad (6)$$

$$v^* = v^n - \tau\left[u^{n-1}\frac{\partial v^*}{\partial x} + v^{n-1}\frac{\partial v^*}{\partial y} - \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 v^*}{\partial x^2} + \frac{\partial^2 u^*}{\partial y^2}\right) - \frac{\partial p^{n-1}}{\partial y}\right]. \qquad (7)$$

Здесь τ – шаг по временным слоям, верхний индекс «n» – номер слоя по времени.

На втором шаге путём решения уравнения Пуассона

$$\frac{\partial^2 p'}{\partial x^2} + \frac{\partial^2 p'}{\partial y^2} = \frac{1}{\tau}\mathrm{div}\vec{V}^* \qquad (8)$$

определяется поправка поля давления p', с помощью которой корректируются поля скорости и давления на третьем шаге алгоритма

$$u^n = u^* - \tau\frac{\partial p'}{\partial x}, \quad v^n = v^* - \tau\frac{\partial p'}{\partial y}, \quad p^n = p^{n-1} + \sigma p', \qquad (9)$$

где σ – итерационный параметр. Необходимо пояснить, что схема (6) – (9) изложена в варианте определения стационарного решения методом установления.

Реализация схемы (6) – (9) всегда была связана с двумя проблемами: 1) поточечной сходимостью векторов решения $\left\{u_{ij}^n\right\}$, $\left\{v_{ij}^n\right\}$, $\left\{p_{ij}^n\right\}$ при итерировании по временным слоям; 2) решением СЛАУ, полученной при разностной аппроксимации задачи Неймана на базе уравнения (8). И если первая проблема является естественной сутью используемой схемы решения задачи, то вторая может быть преодолена путём применения мощного метода решения разностных эллиптических СЛАУ, что и было сделано в рамках настоящего исследования. Для построения решений периодически возникающих на каждом слое по времени СЛАУ использовался неявный итерационный полинейный рекуррентный метод второго порядка экстраполяции, ускоренный в подпространствах Крылова, так называемый LR2sK [1].

Сходимость решений СЛАУ определялась по величине первой нормы разности двух последовательных приближений векторов решений, которая должна быть меньше наперёд заданной точности ξ, умноженной на первую норму вектора приближения решения. Для СЛАУ на базе уравнений движения (6) – (7) точность

принималась равной $\xi = 5\times10^{-7}$, а на базе уравнения Пуассона (8) – $\xi = 10^{-8}$. Решение считалось установившимся в случае, когда $\left\|\mathrm{div}\vec{V}^*\right\|_{\mathrm{I}} < \varepsilon$, где оценка ε производится на основе абсолютных минимальных значений функции тока ψ и характерного сеточного шага h (поле ψ определяется через найденные поля u и v для построения линий тока течения):

$$\varepsilon \sim \mathrm{div}\vec{V} \sim \frac{\delta V}{h} \sim \frac{\delta\psi}{h^2} \Rightarrow \varepsilon \sim \frac{\delta\psi}{h^2}, \qquad (10)$$

где δV и $\delta\psi$ – погрешности векторов компонент скорости и функции тока соответственно. В настоящем исследовании при ожидаемых $\psi \sim 10^{-10}\div10^{-12}$ и $h \sim 10^{-3}$ была использована величина $\varepsilon = 10^{-5}$.

Достоверность полученных решений задачи подтверждается путём сравнения с литературными данными. В частности, в табл. 1 представлены параметры центрального (первичного) вихря течения при Re = 1000.

Таблица 1

Источник	Параметры вихря			
	xc	yc	ψc	ωc
R. Schreiber, H. B. Keller [5]	–	–	−0,1160	−2,026
D. Kwak, S. E. Rogers [6]	–	–	−0,1171	−2,044
U. Ghia, K. N. Ghia, C. T. Shin [7]	0,5313	0,5625	−0,1179	−2,050
В. П. Шапеев, В. И. Исаев [8]	0,5308	0,5652	−0,1189	–
Данная работа	0,5315	0,5645	−0,1186	−2,059

Здесь x_c, y_c – координаты центра вихря, ψ_c и ω_c – экстремумы соответственно функции тока и завихренности в центре первичного вихря.

В работах других авторов неоднократно отмечалось, что решение СЛАУ на базе уравнений движения (6) – (7) не представляет трудностей. Действительно, в настоящем исследовании оценки чисел обусловленности по первой норме матриц таких СЛАУ в широком диапазоне чисел Re = 100÷1000 и сеточных разрешений от 65×65 до 1025×1025 дают значения в пределах нескольких сотен, что говорит об отсутствии каких-либо трудностей при построении решений данных систем уравнений. В среднем для достижения заданной точности требовалось от 4–5 на

начальном этапе расчёта варианта до 1–2 итераций на момент установления решения.

Совершенно иная картина складывается при решении СЛАУ на базе уравнения (8). Как уже указывалось ранее, решение подобных систем представляет собой одну из двух основных проблем при численном моделировании динамики несжимаемой вязкой жидкости. На рис. 2 представлены зависимости количества итераций k_p, необходимых для нахождения полей поправки давления p' от номера слоя по времени n методами LR2sK и бисопряженных градиентов со стабилизацией [9] с предобусловливанием на базе явного метода Н. И. Булеева (Bi-CGStab P B). Хорошо видно, что в среднем на каждом слое по времени для сходимости LR2sK требуется 10–20 итераций, а для Bi-CGStab P B – 200–400 итераций. Любопытно, что для обоих методов данный результат практически не зависит от сложности решаемой задачи, которая определяется величиной числа Re и сеточным разбиением области (чем больше, тем сложнее задача).

И хотя одна итерация метода Bi-CGStab P B выполняется в 2–3 раза быстрее одной итерации LR2sK, преимущество использования последнего не вызывает сомнений. Это преимущество также подтверждается данными табл. 2, в которой представлены абсолютные времена (сек) решения задачи при Re = 1000 при применении методов LR2sK и Bi-CGStab P B при различных сеточных разбиениях области.

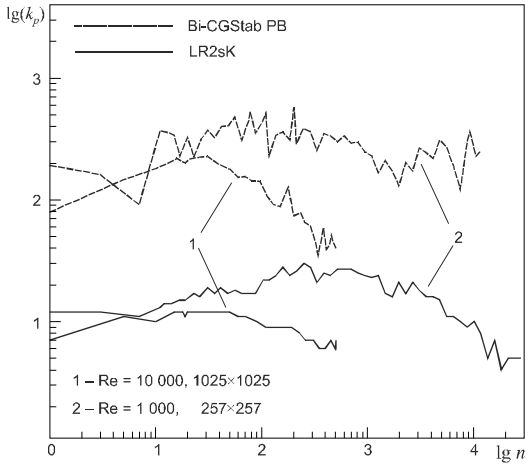


Рис. 2

Из данных табл. 2 следует, что трудоёмкость расчётов резко растёт по мере увеличения сеточного разбиения области: приблизительно на порядок при увеличении количества узлов сеточного разбиения в 4 раза. Данный факт в наилучшей степени иллюстрирует характер поточечной сходимости используемой трехшаговой схемы решения задачи.

Таблица 2

Метод	Сетка				
	65×65	129×129	257×257	513×513	1025×1025
LR2sK	3,0	16,0	149,0	1 903,8	23 396,0
Bi-**CGStab** P B	5,0	32,0	421,0	8 898,0	143 076,0

На основании изложенного можно сделать вывод о том, что использование неявного итерационного полинейного рекуррентного метода полностью себя оправдывает и, соответственно, позволяет снять одну из двух проблем численного моделирования динамики несжимаемой вязкой жидкости.

Литература

1. *Фомин А. А., Фомина Л. Н.* Неявный итерационный полинейный рекуррентный метод решения разностных эллиптических уравнений. Кемерово: КемГУ, 2012. 314 с.
2. *Koseff J. R., Street R. L.* Visualization studies of a shear driven three-dimensional recirculating flow // J. Fluids Engng. 1984. Vol. 106. P. 21–29.
3. *Патанкар С.* Численные методы решения задач теплообмена и динамики жидкости: Пер. с англ. М.: Энергоатомиздат,1984. 152 с.
4. *Белоцерковский О. М., Гущин В. А., Щенников В. В.* Метод расщепления в применении к решению задач динамики вязкой несжимаемой жидкости // ЖВМ и МФ. 1975. Т. 15, № 1. С. 197–207.
5. *Schreiber R., Keller H. B.* Driven Cavity Flows by Efficient Numerical Techniques // J. Comp. Physics. 1983. Vol. 49. P. 310–333.
6. *Rogers S. E., Kwak D.* An upwind differencing scheme for the incompressible Navier-Stokes equations // Applied Numerical Mathematics. 1991. Т. 8. С. 43–64.

7. *Ghia U., Ghia K. N., Shin C. T.* High-Re solution for incompressible flow using the Navier-Stokes equations and a multigrid method // Journal of Computational Physics. 1982. Vol. 48. P. 387–411.
8. *Исаев В. И., Шапеев В. П.* Варианты метода коллакации и наименьших квадратов повышенной точности для численного решения уравнения Навье–Стокса // ЖВМ и МФ. 2010. Т. 50, № 10. С. 1758–1770.
9. *Van der Vorst H. A.* Bi-CGStab: a fast and smoothly converging variant of BI-CG for solution of nonsymmetric linear systems // SIAM J. Sci. Stat. Comput. 1992. Vol. 13, № 2. P. 631–644.

Алгоритм SIMPLED согласования полей скорости и давления для численного моделирования термобара в глубоком озере*

Б.О. Цыденов, А.В. Старченко

Томский государственный университет, Томск

Для согласования полей скорости и давления разработана процедура SIMPLED для течений с плавучестью, представляющая собой модификацию известного метода SIMPLE Патанкара и Сполдинга.

При математическом моделировании термобара в глубоком озере для решения уравнений количества движения

$$\frac{\partial u}{\partial t}+\frac{\partial u^2}{\partial x}+\frac{\partial uw}{\partial z}=-\frac{1}{\rho_0}\frac{\partial p}{\partial x}+\frac{\partial}{\partial x}\left(K_x\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial u}{\partial z}\right)+ +2\Omega_z v-2\Omega_y w; \quad (1)$$

$$\frac{\partial v}{\partial t}+\frac{\partial uv}{\partial x}+\frac{\partial wv}{\partial z}=\frac{\partial}{\partial x}\left(K_x\frac{\partial v}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial v}{\partial z}\right)+2\Omega_x w-2\Omega_z u; \quad (2)$$

$$\frac{\partial w}{\partial t}+\frac{\partial uw}{\partial x}+\frac{\partial w^2}{\partial z}=-\frac{1}{\rho_0}\frac{\partial p}{\partial z}+\frac{\partial}{\partial x}\left(K_x\frac{\partial w}{\partial x}\right)+\frac{\partial}{\partial z}\left(K_z\frac{\partial w}{\partial z}\right)- -\frac{g\rho}{\rho_0}+2\Omega_y u-2\Omega_x v, \quad (3)$$

входящих в численную модель [1], необходимо знать распределение давления. Для согласования полей скорости и давления разработана процедура SIMPLED для течений с плавучестью, представляющая собой модификацию известного метода SIMPLE (*Semi-Implicit Method for Pressure-Linked Equations* – полунеявный метод для связывающих давление уравнений) Патанкара и Сполдинга [2]. Следует заметить, что при использовании процедуры SIMPLE в некоторых случаях скорость сходимости оказывается недостаточно быстрой. Причиной этому является то, что уравнение для поправки

* Работа выполнена при финансовой поддержке стипендии Президента РФ для молодых ученых и аспирантов (СП-71.2012.5).

давления даёт завышенные значения, несмотря на правдоподобность соответствующих поправок к составляющим скорости. Процедура SIMPLE основана на коррекции скорости только от поправки давления. Однако в численную модель термобара [1] входит уравнение состояния Чена – Миллеро, связывающее плотность воды с температурой, солёностью, давлением [3]:

$$\rho(T,S,p)=\frac{\rho^0(T,S)}{1-\dfrac{p}{K(T,S,p)}}, \tag{4}$$

где T – температура, S – солёность, p – давление, ρ – плотность, $\rho^0(T,S)$ – плотность воды на поверхности озера, $K(T,S,p)$ – объёмный модуль упругости.

Таким образом, уравнение состояния (4), включенное в математическую модель расчёта термобара [1], позволяет учитывать три фактора (температуру, солёность и давление), индуцирующие движение в озере.

Суть метода SIMPLED ("*D*" обозначает англ. слово "*density*" – плотность) заключается в циклической последовательности операций «предположение – коррекция». С помощью некоторого начального поля давления сначала вычисляются компоненты скорости по уравнению движения (1) – (3). Затем давление, плотность и компоненты скорости корректируются так, чтобы удовлетворять уравнению неразрывности

$$\frac{\partial u}{\partial x}+\frac{\partial w}{\partial z}=0. \tag{5}$$

Итерационный процесс продолжается до тех пор, пока не будет получено сходящееся решение.

Рассмотрим дискретные аналоги уравнений количества движения для составляющих скорости *u* и *w* (здесь и ниже используются обозначения, принятые в [2]):

$$a_e^u u_e=\sum_{nb} a_{nb}^u u_{nb}+b^u+(p_P-p_E)\Delta z; \tag{6}$$

$$a_n^w w_n=\sum_{nb} a_{nb}^w w_{nb}+b^w+(p_P-p_N)\Delta x-\frac{\rho_n g}{\rho_0}\Delta x\cdot\delta z, \tag{7}$$

где $\sum_{nb}$ означает суммирование по всем соседним узлам конечного объёма W, E, S и N; Δx и Δz – шаги сетки в соответствующем направлении (рис.1).

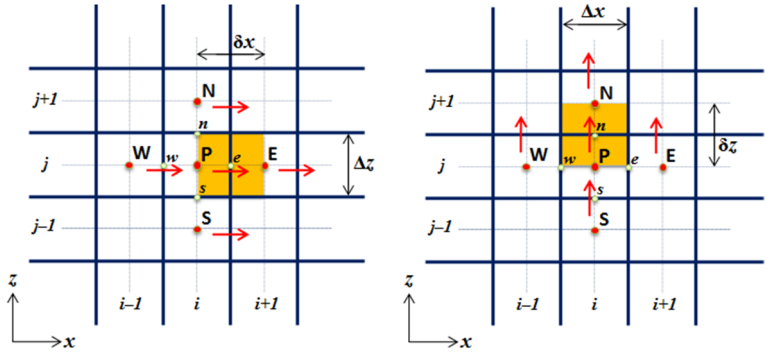


Рис. 1. Конечные объёмы для компонент скорости: для продольной составляющей u (слева), для вертикальной составляющей w (справа)

Выразим поле скорости через u^* и w^* при помощи приближенных полей давления p^* и плотности ρ^*:

$$a_e^u u_e^* = \sum_{nb} a_{nb}^u u_{nb}^* + b^u + (p_P^* - p_E^*)\Delta z; \tag{8}$$

$$a_n^w w_n^* = \sum_{nb} a_{nb}^w w_{nb}^* + b^w + (p_P^* - p_N^*)\Delta x - \frac{\rho_n^* g}{\rho_0}\Delta x \cdot \delta z. \tag{9}$$

Пусть истинное давление определяется из соотношения

$$p = p^* + p',$$

где p' – поправка давления.

Аналогичным образом введем поправки для компонент скорости и плотности:

$$u = u^* + u',$$
$$w = w^* + w',$$
$$\rho = \rho^* + \rho'.$$

Если из (8), (9) вычтем (6), (7) соответственно и отбросим члены $\sum_{nb} a_{nb}^u u_{nb}$ и $\sum_{nb} a_{nb}^w w_{nb}$, то получим поправочные формулы для поля скорости:

$$u_e = u_e^* + d_e(p'_P - p'_E); \tag{10}$$

$$w_n = w_n^* + d_n(p'_P - p'_N) - \frac{\rho'_n g}{\rho_0} c_n, \tag{11}$$

где $d_e = \frac{\Delta z}{a_e^u}$, $d_n = \frac{\Delta x}{a_n^w}$, $c_n = \frac{\Delta x \cdot \delta z}{a_n^w} = d_n \delta z$.

Аналогично можно выразить компоненты скорости:

$$u_w = u_w^* + d_w(p'_W - p'_P); \tag{10a}$$

$$w_s = w_s^* + d_s(p'_S - p'_P) - \frac{\rho'_s g}{\rho_0} c_s, \tag{11a}$$

где $d_w = \frac{\Delta z}{a_e^u}$, $d_s = \frac{\Delta x}{a_s^w}$, $c_s = \frac{\Delta x \cdot \delta z}{a_s^w} = d_s \delta z$.

Интегрируем уравнение неразрывности (5) по конечному объёму:

$$\left(u_e - u_w\right)\Delta z + \left(w_n - w_s\right)\Delta x = 0. \tag{12}$$

Если теперь вместо всех компонент скорости в (12) подставить их выражения из поправочных формул вида (10), (10а), (11), (11а), то после группировки соответствующих членов получим дискретный аналог уравнения для поправки давления:

$$a_P p'_P = a_E p'_E + a_W p'_W + a_N p'_N + a_S p'_S + b, \tag{13}$$

где $a_E = d_e \Delta z$, $a_W = d_w \Delta z$, $a_N = d_n \Delta x$, $a_S = d_s \Delta x$,

$$a_P = a_E + a_W + a_N + a_S,$$

$$b = \left(u_w^* - u_e^*\right)\Delta z + \left(w_s^* - w_n^*\right)\Delta x + \frac{g}{\rho_0}\left(c_n \rho'_n - c_s \rho'_s\right)\Delta x. \tag{14}$$

В основе решения систем линейных алгебраических уравнений вида (13) лежит метод неполной факторизации (явный метод Н. И. Булеева) [4].

Таким образом, алгоритм SIMPLED имеет следующую последовательность операций на каждом шаге по времени:

1. Задание приближенного поля давления p^*, температуры T^* и солёности S^*.

2. Решение уравнений количества движения для получения приближенных значений компонент скорости u^* и w^*.

3. Решение уравнений энергии (для получения T) и солёности (для получения S) и расчёт $\rho' = \rho(p^*, T, S) - \rho(p^*, T^*, S^*)$.

4. Решение уравнения для поправки давления p'.

5. Расчёт поля давления p из уравнения $p = p^* + p'$.

6. Корректировка компонент скорости u и w.

7. Решение уравнений энергии (для получения T) и солёности (для получения S) и расчёт $\rho = \rho(p, T, S)$.

8. Решение дискретных аналогов для других Ф (концентрации, турбулентных характеристик).

9. Возврат к пункту 2 и повтор расчётов до тех пор, пока не будет достигнута сходимость.

Следует признать, что одна итерация SIMPLED в отличие от SIMPLE требует больше расчётных усилий. Однако дополнительное усилие на итерацию более чем компенсируется экономией общего расчётного времени, так как SIMPLED позволяет увеличить шаг по времени в два раза.

Литература

1. *Цыденов Б.О., Старченко А.В.* Численная модель взаимодействия систем «река – озеро» на примере весеннего термобара в озере Камлупс // Вестн. Том. гос. ун-та. Математика и механика. 2013. № 5(25) С. 102–115.

2. *Патанкар С.* Численные методы решения задач теплообмена и динамики жидкости : пер. с англ.; под ред. В.Д. Виленского. М. : Энергоатомиздат, 1984. 124 с.

3. *Chen C.T., Millero F.G.* Precise thermodynamic properties for natural waters covering only limnologies range // Limnol. Oceanogr. 1986. Vol. 31, № 3. P. 657–662.

4. *Булеев Н.И.* Метод неполной факторизации для решения двумерных и трехмерных разностных уравнений типа диффузии // Журн. вычисл. матем. и матем. физ. 1970. Т. 10, № 4. С. 1042–1044.

Численное решение уравнения конвекции-диффузии на неструктурированных сетках[1]

В.В. Чуруксаева, А.В. Старченко

Томский государственный университет, Томск

Рассматривается разностная схема для решения нестационарного уравнения конвекции-диффузии на неструктурированных сетках. На основе полученного решения задачи о точечном источнике показаны преимущества использования неструктурированных сеток и корректность построенного численного метода.

Задачи расчета распространения загрязнителей в естественных средах часто возникают в различных прикладных науках. Но в силу того, что экспериментальное исследование сопряжено с рядом экономических и технологических трудностей, все чаще применяются возможности математического моделирования.

При моделировании процессов, протекающих в естественных условиях, геометрия рассматриваемой области, как правило, довольно сложна, потому предпочтительным становится использование неструктурированных сеток.

Одной из актуальных задач современных наук о Земле является изучение распространения выбросов в атмосферу или водоем от промышленных предприятий. Важной частью таких расчетов является решение конвективно-диффузионного уравнения.

В качестве приближения описанной проблемы рассмотрим задачу распространения в прямоугольной области $D = [0, L_x] \times [0, L_z]$ инертной примеси, выбрасываемой мгновенным источником, расположение и мощность которого известны.

Математически процесс переноса примеси определяется решением краевой задачи для нестационарного двумерного конвективно-диффузионного уравнения

[1] Работа выполнена при финансовой поддержке Минобрнауки РФ в рамках государственного задания №2014/223 (код проекта 2382).

$$\frac{\partial \Phi}{\partial t} + div\vec{\mathrm{I}} = S_{\Phi};\quad \vec{\mathrm{I}} = \vec{V}\Phi - \Gamma grad\Phi,$$
$$S_{\Phi} = Q\delta(t-t_0)\delta(x-x_0)\delta(z-z_(\tag{1}$$

с начальными и граничными условиями

$$t = 0:\quad \Phi = 0;$$
$$x = 0:\ \Phi = 0;\quad x = L_x : \frac{\partial \Phi}{\partial x} = 0; \tag{2}$$
$$z = 0 : \Phi = 0;\quad z = L_z : \frac{\partial \Phi}{\partial z} = 0.$$

Здесь функция $\Phi(x,z,t)$ описывает концентрацию примеси, $S_{\Phi}(x,z,t)$ – распределение и интенсивность источника, $\vec{V} = (u,w)$ – вектор скорости, Γ – коэффициент диффузии, Q – мощность источника, x_0, z_0, t_0 – координаты источника и момент выброса примеси.

Существование и единственность решения поставленной задачи доказаны в [1].

Генерация неструктурированных, в частности треугольных, сеток сегодня требует существенно меньше времени вычислителя, сравнимого со временем построения структурированных или блочно-структурированных сеток, а современные производительные вычислительные системы позволяют генерировать неструктурированные сетки большого объема. Такие сетки можно сгущать в подобластях с большими градиентами решения, не перестраивая на всей области.

Для решения поставленной задачи с помощью сеточного генератора Gambit на области поиска решения строится триангуляция, не содержащая тупоугольных треугольников.

Дискретный аналог задачи (1)–(2) построим методом конечного объема, позволяющим гарантировать выполнение законов сохранения для каждой ячейки сетки и, соответственно, для всей области.

В качестве конечного объема выберем ячейку Дирихле (рис.1): для каждого узла сетки строится контур из отрезков, соединяющих точки пересечения срединных перпендикуляров треугольников $C_1 ... C_k$. Таким образом, ячейка Дирихле представляет собой

множество точек области, более близких к данному узлу, чем к любому другому.

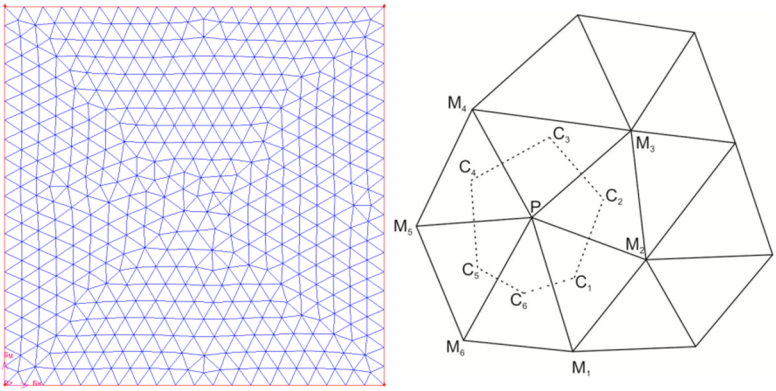


Рис.1. Расчетная сетка на области и изображение контрольного объема

Неизвестные значения концентрации предполагается определять в узлах сетки, а потоки через грани конечного объема аппроксимировать через эти значения. Компоненты скорости u, w заданы в точке пересечения отрезков, соединяющих центры описанных окружностей смежных треугольников C_k, C_{k-1} и их общей стороны PM_k.

Для определения значений неизвестных концентраций в узлах триангуляции M_i исходное уравнение интегрируется по соответствующей ячейке:

$$\int_{D_i}\left[\frac{\partial\Phi}{\partial t}+div\bar{\mathrm{I}}-S_\Phi\right]ds=0.$$

Воспользовавшись формулой Грина и теоремой о среднем, перейдем от интеграла по площади к интегралу по ограничивающему ее контуру и получим соотношение

$$\frac{d}{dt}\overline{\Phi}S_i+\int_{\partial D_i}\left[(\overline{V}\Phi)_n-\Gamma\frac{\partial\Phi}{\partial n}\right]dl-S_\Phi S_i=0.$$

Здесь S_i – площадь конечного объема.

Аппроксимируя полученные значения с помощью приближенных формул, получим дискретный аналог уравнения (1):

$$\frac{\tilde{\Phi}_P - \Phi_P}{\Delta t} S_P + \sum_{k=1}^{N_P} V_n \Phi_n \left|C_k C_{k-1}\right| - \Gamma \sum_{k=1}^{N_P} \frac{\Phi_{M_k} - \Phi_P}{\left|PM_k\right|} \left|C_k C_{k-1}\right| = S_\Phi S_P,$$

P – текущий узел триангуляции, N_P – число треугольников, содержащих общую вершину P, S_P – площадь ячейки Дирихле для P.

Для аппроксимации конвективного слагаемого здесь применяется схема с разностями против потока: значение на грани конечного объема рассчитывается в зависимости от направления потока через эту грань:

$$V_n \Phi_n = \begin{cases} V_n \Phi_P, \; V_n \geq 0, \\ V_n \Phi_{M_k}, \; V_n < 0. \end{cases}$$

$$V_n \Phi_n \left|C_k C_{k-1}\right| = \left[\max\left(un_x + wn_z, 0\right) \Phi_P - \max\left(-un_x - wn_z, 0\right) \Phi_{M_k} \right] \left|C_k C_{k-1}\right|.$$

$$n_x = \frac{x_{M_k} - x_P}{\left|PM_k\right|}, n_z = \frac{y_{M_k} - y_P}{\left|PM_k\right|}.$$

В итоге вычислительная формула для значения концентрации в текущем узле P на временном слое n+1 будет иметь вид (3):

$$\tilde{\Phi}_P = \Phi_P \left(1 - \frac{\Delta t}{S_P} \sum_{k=1}^{N_P} \left[\max\left(un_x + wn_z, 0\right) \left|C_k C_{k-1}\right| + \Gamma \frac{\left|C_k C_{k-1}\right|}{\left|PM_k\right|} \right] + \right.$$
$$\left. \frac{\Delta t}{S_P} \sum_{k=1}^{N_P} \Phi_{M_k} \left[\max\left(-un_x - wn_z, 0\right) \left|C_k C_{k-1}\right| + \Gamma \frac{\left|C_k C_{k-1}\right|}{\left|PM_k\right|} \right] \right) + S_\Phi. \qquad (4)$$

Замечание. В случае, когда текущая вершина P лежит на границе расчетной области, ячейку предлагается замыкать через отрезки, соединяющие узел P с серединами граничных ребер C_1^*, C_5^* (рис. 2).

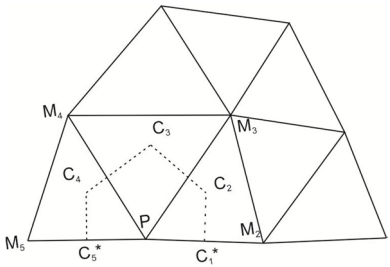


Рис. 2

Таким образом, получена явная по времени консервативная разностная схема первого порядка аппроксимации.

Использование явных схем является предпочтительным для проведения расчета на областях большой площади, когда количество узлов сетки велико и становится оправданным распараллеливание алгоритма решения. Однако при использовании явных схем следует учитывать ограничения, накладываемые условием устойчивости на шаг по времени. Для данного случая достаточное условие сходимости метода получается из условия наличия диагонального преобладания у матрицы полученной системы линейных алгебраических уравнений и записывается в виде $\Delta t \le \min_i \frac{S_i}{\sum_{k=0}^{N_i} a_{ik}}$. Здесь N_i – число треугольных элементов, образующих ячейку Дирихле, S_i – площадь ячейки Дирихле для текущего узла, a_{ik} – элемент матрицы системы (4).

Тестирование построенного численного метода проводилось с помощью задачи о мгновенном точечном источнике, аналитическое решение которой имеет вид

$$\Phi(x,y,t) = \frac{Q}{4\pi\Gamma(t-t_0)} \exp\left(-\frac{(x-x_0-u(t-t_0))^2}{4\Gamma(t-t_0)}\right) \exp\left(-\frac{(z-z_0-w(t-t_0))^2}{4\Gamma(t-t_0)}\right).$$

Численное решение поставленной задачи находилось для случая

$u = 3\,м/с;\ w = 0,5\ м/с; \Gamma = 0,5 м^2/\text{с};\ \ \text{Q} = 1 кг;$

$x_0 = 100\,м;\ z_0 = 90 м;\ t_0 = 10\,с; L_x = L_x = 180$

на треугольной сетке, содержащей 489 элементов, и представлено на рис. 3.

Из рис. 3 видно, что с течением времени примесь переносится в направлении сносящего потока, задаваемого полем скорости, а также распространяется с уменьшением концентрации за счет диффузии. Таким образом, полученное численное решение соответствует физике изучаемого процесса и, как видно из рисунка, хорошо согласуется с аналитическим.

Для выявления зависимости качества получаемого решения от выбора сетки проведем расчет по противопотоковой схеме, построенной для структурированной сетки $22 \times 22 = 484$ узла.

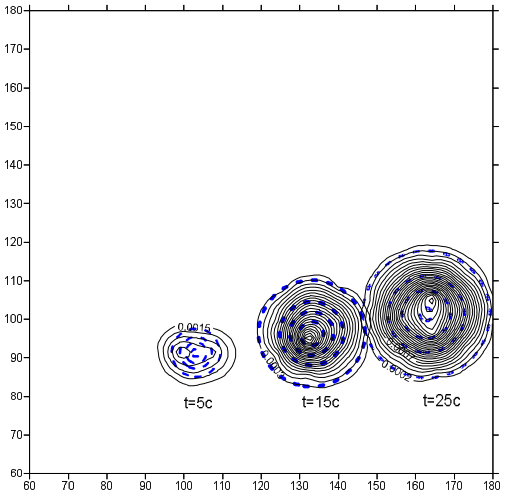


Рис. 3. Численное решение в указанные моменты времени. (Штриховой линией обозначено точное решение)

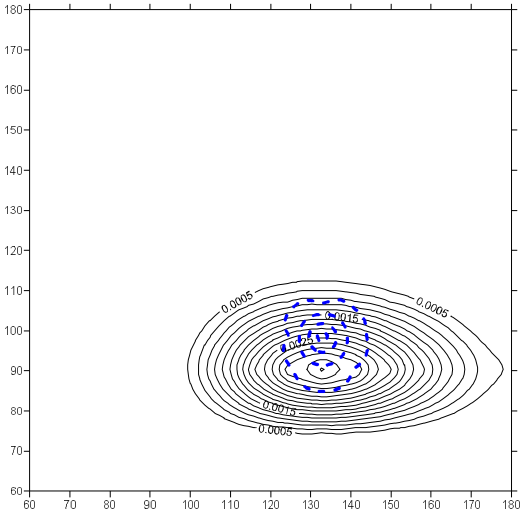


Рис. 4. Численное решение задачи на структурированной сетке при $t = 15\text{с}$. (Штриховой линией обозначено точное решение)

Результат тестового расчета для момента времени $t = 15\,\text{с}$ приведен на рис. 4.

Видно, что полученное численное решение значительно отличается от аналитического. Достижение более точного решения на структурированной сетке требует большего количества ее узлов.

В результате можно отметить: с помощью метода конечного объема на неструктурированной сетке построена явная разностная схема первого порядка. Полученный численный метод протестирован с помощью задачи о мгновенном точечном источнике в двумерной прямоугольной области. Сравнение с расчетом на структурированной сетке показывает, что треугольная сетка позволяет получить более точное решение при приблизительно равном числе узлов. Дальнейшие исследования в данном направлении предполагают проведение расчетов для областей сложной геометрии, задание переменного поля скоростей и построение схем более высокого порядка.

Литература

1. *Марчук Г.И.* Математическое моделирование в проблеме окружающей среды. М.: Наука, 1982. 320 с.

К программной реализации решения задачи геометрического программирования

М.А. Тынкевич

Кузбасский государственный технический университет,
Кемерово

Рассматривается проблематика компьютерной реализации задач геометрического программирования и возможные пути ее полного решения.

Как известно, задача геометрического программирования сводится к поиску наименьшего значения позинома

$$g_0(x_1, x_2, ..., x_n) = \sum_{k=1}^{m_0} c_k x_1^{\alpha_{1k}} x_2^{\alpha_{2k}} ... x_n^{\alpha_{nk}} \quad (c_k > 0 \text{ при всех } k) \qquad (1)$$

при $x_1>0$, .., $x_n>0$ и так называемых вынужденных ограничениях $g_k(x_1, .., x_n) \le 1$, $k=1 \div p$, где все функции $g_k(x_1, .., x_n)$ – позиномы [1].

Если обозначить $m = \sum_{k=0}^{p} m_k$, где m_k – количество слагаемых в соответствующих позиномах, и ввести множества индексов:

$J_0 = \{1, 2, ..., m_0\}$, $J_1 = \{m_0 + 1, m_0 + 2, ..., m_0 + m_1\}$, …,

$$J_p = \left\{ \sum_{i=0}^{p-1} m_i + 1, \sum_{i=0}^{p-1} m_i + 2, ..., \sum_{i=0}^{p-1} m_i + m_p \right\},$$

то все позиномы запишутся в виде

$$g_k(x_1, x_2, ..., x_n) = \sum_{i \in J_k} c_k x_1^{\alpha_{1i}} x_2^{\alpha_{2i}} ... x_n^{\alpha_{ni}}$$

и информация о решаемой задаче представима вектором коэффициентов $c_k>0$ ($k=1,2,\ldots,m$) и *матрицей экспонент* (показателей степени), составленной из n строк и m столбцов.

Двойственная задача состоит в максимизации функции

$$V(\delta) = \left[\prod_{k=1}^{m} \left(\frac{c_k}{\delta_k} \right)^{\delta_k} \right] \prod_{s=1}^{p} \lambda_s(\delta)^{\lambda_s(\delta)} \quad , \qquad (2)$$

где

$$\lambda_s(\delta) = \sum_{j \in J(s)} \delta_j,\ s=1 \div p\ ;\ \delta_k \geq 0\ (k=1 \div m); \tag{3}$$

$$\delta_1 + \delta_2 + \ldots \delta_{m_0} = 1\ ,\ \sum_{i=1}^{m} \alpha_{ij} \delta_i = 0\ , j=1,2,..,n\ . \tag{4}$$

(функция $V(\delta)$ непрерывна при всех δ, поскольку при δ=0 соблюдается $\delta^{\pm\delta}$=1).

Теоремы двойственности геометрического программирования [1] утверждают, что при условии сильной совместности ограничений прямой задачи и существовании точки X^*, в которой достигается минимум $g_0(X)$ (хотя бы локальный), существует точка δ^*, в которой достигается максимум $V(\delta)$ (глобальный или локальный), причем $g_0(X^*) = V(\delta^*)$.

Для оптимального решения двойственной задачи δ^* любая точка минимума X^* прямой задачи удовлетворяет системе

$$c_i \prod_{s=1}^{n} x_s^{\alpha_s^{is}} = \begin{cases} \delta_i^* \times V(\delta^*),\ i \in J(0) \\ \delta_i^* / \lambda_k(\delta^*),\ i \in J(k), \end{cases} \tag{5}$$

где k принимает все положительные значения такие, что $\lambda_k(\delta^*)>0$.

Алгоритмическая реализация решения задачи сталкивается с принципиальными трудностями, связанными с соблюдением условия $m=n+1$ для минимизируемого позинома.

Так, при решении задачи [1. С.16] минимизации $g_0(X)=x_2$ при ограничениях $g_1(x_1, x_2) = \dfrac{x_1^4}{x_2^4} + \dfrac{\sqrt{x_2}}{x_1} \leq 1$; $x_1, x_2>0$, обнаруживаем m=3; соответственно, $J(0)=\{\ 1\ \}$, $J(1)=\{\ 2\ 3\ \}$, вектор $C=[1\ 1\ 1]$ и матрицу экспонент

$$A = \begin{pmatrix} 0 & 4 & -1 \\ 1 & -4 & 0{,}5 \end{pmatrix},$$

откуда получаем

$$V(\delta) = \left(\frac{1}{\delta_1}\right)^{\delta_1} \left(\frac{1}{\delta_2}\right)^{\delta_2} \left(\frac{1}{\delta_3}\right)^{\delta_3} (\delta_2 + \delta_3)^{\delta_2 + \delta_3}$$

и множество планов, определяемое положительными (в пределе неотрицательными) решениями системы $n+1=3$ уравнений с $m=3$ неизвестными

$$\begin{aligned} \delta_1 &= 1, \\ 4\,\delta_2 - \delta_3 &= 0, \\ \delta_1 - 4\,\delta_2 + 0{,}5\,\delta_3 &= 0. \end{aligned}$$

Обнаружив без труда таковые δ = [1 0,5 2], убеждаемся в существовании решения обеих задач и равенства $V(\delta) = g_0(X) = 3{,}4939$. Обратившись к (5), получаем нелинейную, но сводимую логарифмированием к линейной, систему m=3 зависимых уравнений с n=2 неизвестными (заметьте, что в роли коэффициентов этой системы выступает транспонированная матрица экспонент):

$$x_2 = \delta_1 \times V(\delta) = 3{,}4939\ ;\ x_1^4 \Big/ x_2^4 = \left(\frac{\delta_2}{\delta_2+\delta_3}\right) = 0{,}2\ ;\ \sqrt{x_2} \Big/ x_1 = \left(\frac{\delta_3}{\delta_2+\delta_3}\right) = 0{,}8,$$

откуда $z_1 = \lg(x_1) = 0{,}3683$, $z_2 = \lg(x_2) = 0{,}5433$ и, соответственно, $x_1 = 2{,}34$, $x_2 = 3{,}49$.

Однако решая, на первый взгляд, более простую задачу минимизации

$$g_0(X) = \frac{40}{x_1 x_2} + 40x_2 + 20x_1 + 10x_1x_2$$

при отсутствии вынужденных ограничений (так называемый коэффициент трудности $s=m-(n+1)=4-3>0$), приходим к максимизации

$$V(\delta) = \left(\frac{40}{\delta_1}\right)^{\delta_1} \left(\frac{40}{\delta_2}\right)^{\delta_2} \left(\frac{20}{\delta_3}\right)^{\delta_3} \left(\frac{10}{\delta_4}\right)^{\delta_4}$$

на множестве неотрицательных решений системы (m=4, n=2)

$$\begin{aligned} \delta_1 + \delta_2 + \delta_3 + \delta_4 &= 1, \\ -\delta_1 + \delta_3 + \delta_4 &= 0, \\ -\delta_1 + \delta_2 + \delta_4 &= 0. \end{aligned}$$

Если при нулевой трудности $s=m-(n+1)=0$ программирование вычислительного процесса действительно не вызывает особых трудностей, то при $s>0$, построив аналогичную систему относительно δ_k, мы вынуждены выбирать дальнейший путь поиска оптимального решения.

Один из вариантов связан с отказом от «компьютерного продолжения» и использованием человеческого знания и интуиции на пути понижения размерности (громоздкая подготовительная работа уже выполнена) – подобрать некоторый набор n базисных

переменных и выразить остальные через них. Однако при s >>1 этот вариант чреват возникновением не менее сложной задачи.

Другой вариант предполагает использование процедуры Монте-Карло (еще 20 лет назад нереальной при сколь-нибудь значительной размерности, не говоря уже о «детских годах» геометрического программирования).

Есть и третий путь, базирующийся на идеях методов квадратического программирования (например, метода Вулфа – Фрэнка). Здесь в рамках симплексной процедуры с полным начальным искусственным базисом отыскивается начальный опорный план, находятся компоненты градиента $V(\delta)$

$$\frac{\partial V}{\partial \delta_k} V(\delta) \times \left(\begin{array}{l} \left(\dfrac{c_k}{\delta_k} \right)^{\delta_k} \ln \dfrac{c_k}{\delta_k} \left(-\dfrac{c_k}{\delta_k^2} \right) + \\ + \begin{cases} \lambda_s(\delta)^{\lambda_s(\delta)} \ln \lambda_s(\delta), & k \in J(s) \\ 0, & k \notin J(s) \end{cases} \end{array} \right), k = \overline{1,m},$$

принимаемые за компоненты «линейной формы», и выполняется переход к очередному опорному плану в направлении ее возрастания. Увы, поскольку вогнутость $V(\delta)$ в общем случае не гарантирована, не гарантировано и получение глобального оптимума.

Литература

1. *Даффин Р., Питерсон Э., Зенер К.* Геометрическое программирование. М.: Мир, 1972. 311 с.
2. *Тынкевич М.А., Вересова К.А.* К решению задачи геометрического программирования // Вестник КузГТУ. 2009. №3. С. 141–146.

Организация безопасных вычислений на распределенных мобильных устройствах

Н.А. Беляев

Новосибирский государственный технический университет, Новосибирск

Представлен проект и прототип реализации системы распределенных вычислений на мобильных устройствах. Обеспечивается предотвращение зловредных действий со стороны вычислительных программ на мобильных устройствах пользователей.

Введение

Все больше растут потребности человечества в вычислениях для решения множества фундаментальных и прикладных задач. Ряд задач успешно решается в волонтерских сетях распределенных вычислений, таких как World Community Grid, работающих на основе программного обеспечения BOINC[1]. Пользователи персональных компьютеров устанавливают у себя клиентский компонент системы BOINC, который обращается на сервер World Community Grid, скачивает вычислительное задание, выполняет его и отдает серверу результат. Сеть широко используется для решения задач анализа медицинских изображений, структуры молекул белков, подбора лекарств, проектирования новых материалов.

Быстрый рост вычислительной производительности мобильных устройств и количества таких устройств в мире делает целесообразным вовлечение ресурсов мобильных устройств в волонтерские вычислительные сети. В 2013 г. в рамках BOINC было разработано клиентское программное обеспечение для платформы Android [1].

Участники вычислений в таких проектах доверяют организаторам вычислений, и вопрос о проверке безопасности вычислений не ставится. Подобные сети могли бы использоваться для решения задач, поставляемых произвольными заказчиками, и участники расчетов могли бы получать некоторое вознаграждение за предоставление ресурсов. При этом возникает необходимость

[1]BOINC: http://boinc.berkeley.edu

обеспечения безопасности вычислительных устройств и их пользователей.

В настоящей работе предлагается проект системы безопасных отказоустойчивых вычислений на устройствах, работающих на платформе Android, и представлен прототип системы. При организации такой системы распределенных вычислений требуется решение многих проблем, основными из которых являются ограниченность ресурсов мобильных устройств, необходимость обеспечения безопасности вычислений для предотвращения попадания на устройства вредоносных программ, а также обеспечение равномерного планирования вычислительной нагрузки по узлам сети. Также важной проблемой является обеспечение отказоустойчивости в вычислительной сети.

Архитектура системы

Модель организации вычислений. Система состоит из сервера, осуществляющего планирование задач, и клиентских устройств, выполняющих вычисления. На клиентские устройства устанавливается программное обеспечение, которое позволяет пользователю устройства выбрать задачи, в решении которых он хочет участвовать, загружает исполняемые файлы программ, осуществляющих вычисления, и загружает порции данных для обработки этими программами. По мере готовности результаты вычислений отправляются на сервер.

Для выполнения расчетов конкретной прикладной задачи разработчик программы загружает исполняемый файл и исходные данные, разбитые на части, на сервер распределенных вычислений, который производит планирование нагрузки, отправку исходных данных на клиентские устройства, а также сбор результатов.

Принцип обеспечения безопасности. Для обеспечения безопасности предлагается следующий подход. Исполняемый файл, попадая на сервер, проходит ряд модификаций. Сначала производится поиск в файле специальной секции, описывающей импортируемые функции. Она предназначена для того, чтобы у программ была возможность коммуникаций с операционной системой, и представляет собой по сути таблицу, в которой каждому символьному имени функции или переменной соответствует некоторое относительное смещение, из которого в процессе загрузки операционная система вычисляет виртуальный адрес данной функции или переменной. Собственно модификации

исполняемого файла, производимые сервером распределенных вычислений, представляют собой удаление информации о всех импортируемых данных исполняемым файлом функций операционной системы, что предотвращает попытку их прямого вызова и, как следствие, попытку выполнить на клиентском устройстве какие-либо вредоносные действия. Важно отметить, что данный метод не гарантирует полную безопасность клиентских устройств, он основан на том факте, что в момент запуска исполняемого файла задачи на клиентском устройстве неизвестны фактические адреса API операционной системы. Таким образом, при наличии уязвимостей в операционной системе существует возможность обхода данного механизма. Также следует отметить тот факт, что модификация исполняемого файла сама по себе не обеспечивает безопасность вычислений. Существует необходимость обеспечения правильного исполнения данных файлов на клиентских устройствах. В настоящем проекте оно обеспечивается благодаря своеобразной «песочнице». Механизм исполнения заданий проиллюстрирован на рис.1. Программа, полученная с сервера распределенных вычислений, не имеет возможности напрямую обращаться к операционной системе. Вместо этого она обращается к API песочницы, которая, в свою очередь, проведя необходимые проверки входных параметров, обращается к соответствующим API операционной системы.

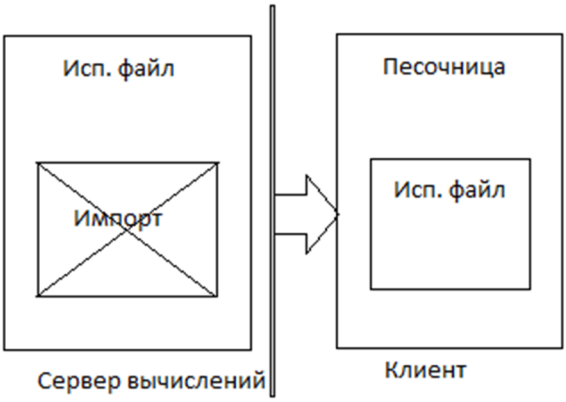


Рис. 1

Механизм планирования

Помимо проблемы обеспечения безопасности, существует необходимость планирования вычислительной нагрузки, распределяемой по узлам. Стоит отметить, что в таких крупных проектах, как BOINC, порции данных, отправляемые для обработки устройствами, имеют одинаковый размер и распределяются по вычислительным узлам вне зависимости от производительности этих ресурсов. Предлагается механизм двухстадийного планирования вычислительной нагрузки. Идея его заключается в том, что изначально известен вычислительный вес каждого фрагмента исходных данных. На первой стадии планирования все множество фрагментов разбивается на подмножества таким образом, что в каждом из подмножеств представлены фрагменты данных, пригодные к вычислению за разумное время на соответствующем наборе устройств, имеющих примерно одинаковую производительность. Так, например, при подобном делении можно учесть различные классы устройств – от высокопроизводительных до бюджетных. Вторая же стадия планирования осуществляется уже непосредственно в момент отправки задания на клиентское устройство и представляет собой, по сути, выбор первого фрагмента входных данных из подмножества, соответствующего данному устройству, который еще не был обработан. Такой подход обеспечивает малые затраты на планирование и в то же время исключает возможность попадания фрагмента, имеющего большой вес, на устройство с низкой производительностью.

Заключение

Разработан проект и реализован прототип системы безопасных распределенных вычислений на мобильных устройствах, работающих на платформе Android. Постановку вычислительных задач осуществляют заказчики, предоставляя исполняемый файл и наборы данных для обработки. Пользователи мобильных устройств, устанавливая специальное программное обеспечение, становятся участниками распределенной вычислительной сети и принимают расчетные задания на свои мобильные устройства. Система предотвращает возможность вредоносных действий со стороны программ заказчиков на мобильных устройствах участниках за счет модификации программ и исполнения их в специальной

виртуальной среде – «песочнице». Обеспечивается планирование распределения вычислений в соответствии с производительностью вычислительных устройств.

Дальнейшие планы включают обеспечение возможности параллельных расчетов на мобильных устройствах и разработку системы отказоустойчивости таких вычислений.

Литература

1. *Fritzsch J.* BOINC on Android: state & outlook // The 9th BOINC Workshop. Grenoble, Inria, 25-27 September 2013 http://boinc.berkeley.edu/trac/raw-attachment/wiki/WorkShop13/boa-state_and_outlook.pdf

Анализ статистических данных отказов кластерных систем национальной лаборатории Лос-Аламоса

Е.С. Пименов, А.Ю. Поляков

Сибирский государственный университет телекоммуникаций и информатики, Новосибирск
Институт физики полупроводников СО РАН, Новосибирск

В данной работе описан созданный расширяемый инструментарий анализа статистических данных Лос-Аламоса, позволяющий определить такие параметры, как среднее время наработки на отказ, среднее время восстановления узла и т.п.

Введение

Распределенные вычислительные системы (ВС) являются основным инструментом высокопроизводительной обработки информации. Современные ВС большемасштабны, они формируются из $10^5 – 10^6$ вычислительных ядер, а их производительность достигает десятков PFLOPS [1]. При этом ВС – сложные технические комплексы, для организации эффективного отказоустойчивого функционирования которых требуется анализ и постоянное совершенствование архитектурных и алгоритмических решений.

Основной задачей распределенных ВС является решение поступающих задач, при этом в процессе их функционирования возможно возникновение отказов. Согласно статистическим исследованиям [2], основными причинами отказов ВС являются аппаратурные ошибки вычислительных узлов. В 2005 году национальной лабораторией Лос-Аламоса (США) были опубликованы наборы статистических данных, включающие в себя информацию о функционировании 23 вычислительных кластеров различных архитектур и конфигураций, собранную в течение 10 лет. Данная информация представляет большой интерес с точки зрения исследования распределенных ВС и может быть использована при построении их имитационных моделей, а также при принятии решений по их конфигурированию, например для определения оптимального интервала между формированием контрольных точек.

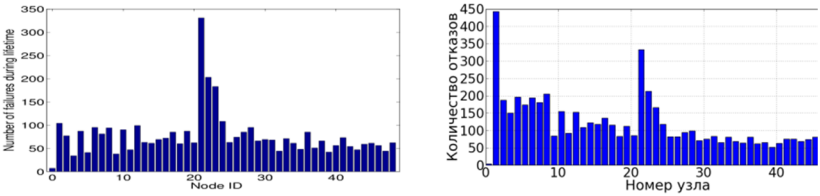
Отказы были классифицированы по следующим категориям:

1) аппаратурные сбои;
2) программные ошибки;

3) сетевые ошибки;
4) человеческий фактор;
5) ошибки в окружении;
6) неопределенные.

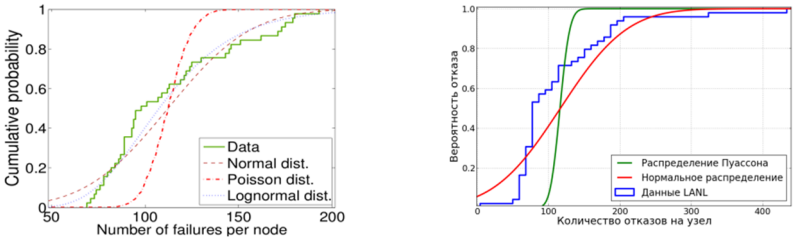
Процесс сбора статистики состоял из нескольких этапов. Ошибка обнаруживалась автоматической системой мониторинга. Оператор получал уведомление и добавлял в базу запись об отказе. Затем инженер устранял проблему, и оператор возвращал узел в систему, дополняя созданную запись информацией о времени устранения неисправности и ее причине. Поскольку в процессе присутствовал человеческий фактор, предоставленная статистика может быть неполной или некорректной. Следовательно, для ее анализа требуется фильтрация входных данных.

Наиболее полные исследования статистики были приведены в статье [2]. Авторами представлены различные показатели функционирования ВС. В частности, приведен график распределения количества отказов по вычислительным узлам системы №2 (рис. 1,а). Также была построена функция распределения количества отказов на узел и выполнено ее приближение тремя различными распределениями (рис. 2,а). Основной вывод авторов заключается в том, что нормальное и логнормальное распределение являются лучшим приближением по сравнению с распределением Пуассона.



а) б)

Рис. 1. Распределение количества отказов по узлам системы №2:
а) результаты из статьи [2], б) результаты, полученные разработанным инструментарием HPCStatTool



а) б)

Рис. 2. Функция распределения отказов по вычислительным узлам системы №2: а) результаты из статьи [2]; б) результаты, полученные разработанным инструментарием HPCStatTool

Инструментарий HPCStatTool

Авторами [2] выполнен анализ лишь части данных, доступных в статистике Лос-Аламоса. В частности, рассмотрено всего несколько систем. Инструментарий, с помощью которого были получены результаты, не был размещен в свободном доступе.

Для более подробного анализа доступных данных потребовалась разработка собственного инструментария, который был назван HPCStatTools. Созданный программный пакет позволяет:

1) гибко определять набор входных данных;
2) задавать методы их анализа;
3) выполнять визуализацию результатов.

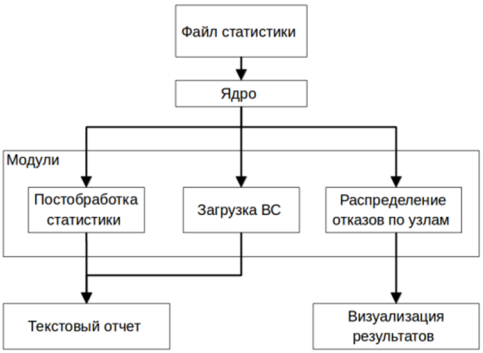


Рис. 3. Архитектура инструментария HPCStatTools

Архитектура разработанного инструментария представлена на рис. 3. Он состоит из ядра и набора расширений, обеспечивающих анализ различных аспектов статистических данных. На уровне ядра производится построчное чтение и проверка корректности входных данных из журналов статистики. Прошедшие проверку данные передаются независимо функционирующим модулям обработки статистики. Эти модули должны реализовывать интерфейс для обработки порции данных и формирования отчета.

Анализ отказов Лос-Аламосской национальной лаборатории

В данной работе приведены результаты исследований для систем из таблицы.

Таблица. Исследуемые системы Лос-Аламосской национальной лаборатории

Номер системы	Количество узлов	Количество ядер	Объем ОЗУ на одном узле, Гб
2	49	6152	128
3	128	512	4
16	16	2048	32
18	1024	4096	16
20	512	2048	16

С помощью разработанного инструментария было получено распределение отказов по вычислительным узлам системы №2 (рис. 1,б). Полученный график отличается от исходного, поскольку в оригинальной статье не был обозначен метод фильтрации исходных данных. Несмотря на это, результаты сопоставимы, а в большей части графика удалось получить точное совпадение результатов. Большой интерес представляет аналогичный график для системы №20, представленный на рис. 4. Очевиден неравномерный характер количества отказов в системе: узлы 1–256 отказывали значительно чаще. Нами было выдвинуто предположение о неравномерной загрузке системы, которое было подтверждено с применением имитационного моделирования, детали будут описаны далее. Система была загружена лишь наполовину, а в качестве алгоритма планирования использовался планировщик типа FIFO, использующий при распределении задач первые свободные узлы системы.

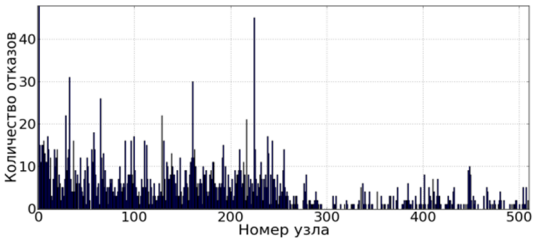


Рис. 4. Распределение отказов по вычислительным узлам системы №20

Также в работе было выполнено построение графиков функций распределения количества отказов на узел. Выводы в работе [2] справедливы не для всех случаев: согласно нашим наблюдениям нормальное распределение является лучшим приближением для SMP-систем, в то время как для кластерных ВС распределение Пуассона является достаточно качественным (рис. 5).

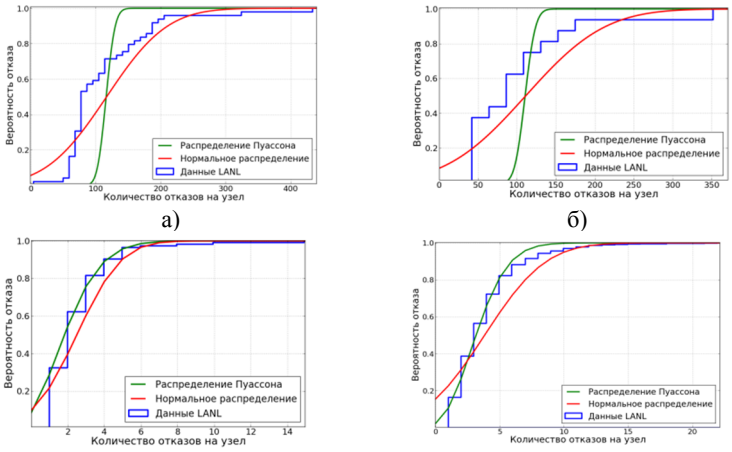


Рис. 5. Функции распределения отказов по узлам и их приближения для систем: а) 2, б) 16, в) 3, г) 18

Анализ загрузки ВС Лос-Аламоса

Помимо информации об отказах статистика Лос-Аламоса содержит данные загрузки систем. Разработанный в данной работе

инструментарий позволяет оценить характеристики потоков задач из представленной статистики.

На рис. 6 показано распределение времени решения задач, поступавших на систему №20. Значительное количество задач требовало 1, 5 и 5,5 часов для решения.

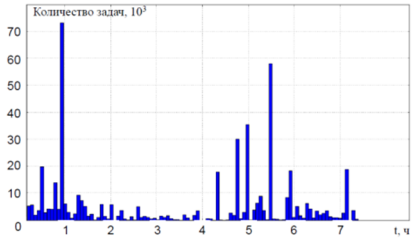


Рис. 6. Распределение времени решения задач на ВС №20

Разработанный инструментарий позволяет обработать статистику потока задач и привести ее к виду, удобному для использования в качестве входных данных имитационной модели. Также предусмотрена возможность получить восстановленную загрузку без проведения моделирования для проверки адекватности модели.

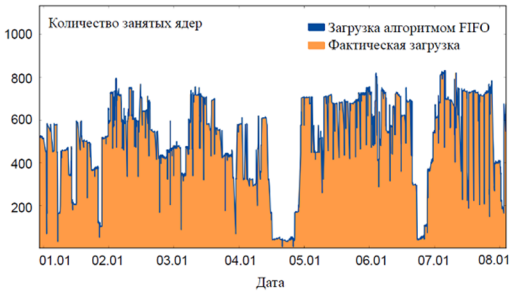


Рис. 7. Восстановленная и моделируемая загрузка системы №20

По извлеченной статистике было проведено имитационное моделирование системы, функционирующей под управлением FIFO-планировщика. На рис. 7 видно, что моделируемая загрузка полностью совпадает с восстановленной, что подтверждает корректность модели. Вместе с этим заметим, что пиковая загрузка

системы не превышает половину узлов, что подтверждает высказанное ранее предположение о причине неравномерного распределения отказов по узлам данной ВС.

Заключение

Разработан программный инструментарий HPCStatTools, который был адаптирован для обработки статистических данных об эксплуатации ВС лаборатории Лос-Аламоса. Выполнен анализ статистики отказов, который согласуется с результатами, опубликованными в работах других исследователей. Сделаны выводы по поводу причин дисбаланса распределения отказов в системе №20. Проведено исследование алгоритма распределения ресурсов на системе №20. Показано, что использовался простой планировщик FIFO. Выполнена визуальная оценка отказов в SMP–кластерах и кластерах, построенных на базе серийного оборудования. Полученные результаты свидетельствуют о корректности созданного инструментария.

Литература

1. *TOP500* [Электронный ресурс] – Режим доступа: http://www.top500.org/
2. *Schroeder B.* A large-scale study of failures in high-performance computing systems / Bianca Schroeder, Garth A. Gibson // Proceedings of the International Conference on Dependable Systems and Networks. – IEEE Computer Society Washington, DC, USA, 2006. P. 249–258.

Анализ эффективности использования средств распределенных вычислений для систем с общей памятью при моделировании течений вязкой жидкости методом граничных элементов*

М.А. Пономарева

Томский государственный университет, Томск

Представлены результаты применения средств распределенных вычислений для систем с общей памятью для параллельной реализации алгоритма решения задач динамики вязкой жидкости при малых числах Рейнольдса методом граничных элементов.

Введение

С появлением многопроцессорной вычислительной техники актуальной становится разработка эффективных параллельных реализаций существующих вычислительных алгоритмов решения научно-технических задач, в том числе задач динамики жидкости. В то же время известно, что использование многих средств организации распределенных вычислений сопряжено со значительным преобразованием программного кода (до 100% в случае использования GPU) и ограниченностью использования полученной параллельной версии программы при отсутствии вычислительных систем с параллельной архитектурой [1–3]. В некоторых случаях добиться значительного повышения эффективности работы алгоритма возможно, частично избежав упомянутых трудностей. Для этого достаточно задействовать вычислительный потенциал наиболее широко распространенной, многопроцессорной SMP-машины, воспользовавшись средствами распределенных вычислений для систем с общей памятью (например, OpenMP), в том числе существующими оптимизированными низкоуровневыми математическими библиотеками, имеющими параллельную реализацию (например, Intel® Math Kernel Library). В настоящей работе такой подход

*Исследование выполнено при поддержке гранта Президента РФ (МК-3687.2014.1) и РФФИ в рамках научного проекта №14-08-31579 мол_а и проекта № 12-08-00313а.

использован для оптимизации программы расчета, реализующей непрямой метод граничных элементов (НМГЭ) для моделирования плоских течений вязкой жидкости [4]. НМГЭ относится к классу численных методов расчета динамики жидкости с явным выделением границы, что делает его применение весьма эффективным в случаях, когда требуется максимально точное выполнение граничных условий, например при наличии свободной поверхности и учете межфазного взаимодействия (поверхностного натяжения, смачиваемости и т.п.). К преимуществам данного подхода можно также отнести снижение размерности задачи на единицу и непрерывное моделирование полей в области решения. Результаты расчетов с использованием многопоточных вычислений методом граничных элементов (МГЭ) (включая расчеты на кластерных системах и графических процессорах) приведены в работах К.Е. Афанасьева, Н.А. Гумерова, A.J. Davies, Y.Y. Wu, M.S. Ingber, Y.J. Liu, S.R. Subia, D. Li и др. для случаев идеальной жидкости, многофазных течений и, конечно же, в теории упругости, где МГЭ получил более широкое распространение. То, что алгоритм, реализующий МГЭ, имеет степень параллелизма, близкую к идеальной, является несомненным преимуществом гранично-интегрального подхода и также положительно сказывается в случае реализации параллельных вычислений для моделирования течений вязкой жидкости.

Постановка задачи

В качестве тестовой выбрана задача о стабилизированном течении вязкой жидкости в плоском канале (рис. 1), имеющая аналитическое решение. Задача решается в безразмерных переменных. В качестве характерных масштабов длины, скорости и давления выбраны полуширина канала R, средняя скорость течения U, $R/\mu U$ (μ – коэффициент динамической вязкости). Система уравнений (уравнения движения совместно с уравнением неразрывности) и граничные условия имеют вид:

$$\begin{cases} \dfrac{\partial \Pi_{ij}}{\partial x_j} = 0, \quad i, j = 1, 2 \\ \dfrac{\partial u_i}{\partial x_i} = 0. \end{cases} \tag{1}$$

$$u_1 = 3/2\left(1-x_2^2\right), \quad u_2 = 0, \quad x \in \Gamma_1$$
$$u_i = 0, \quad x \in \Gamma_2 \qquad (2)$$
$$t_1 = 0, \quad u_2 = 0, \quad x \in \Gamma_3$$

где $\Pi_{ij} = -p\delta_{ij} + 2\dot{e}_{ij}$ – тензор напряжений, $\dot{e}_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$ – тензор скоростей деформаций, p – давление, u_i – компоненты вектора скорости, $t_i = \Pi_{ij} n_j$ – усилия, n_i – компоненты внешней нормали.

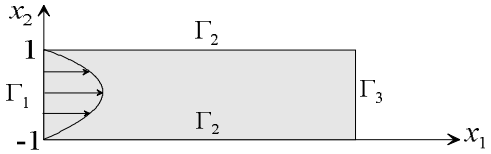


Рис. 1. Область решения

Метод решения

Система (1) заменяется эквивалентной системой граничных интегральных уравнений [4]. Дискретный аналог последней, в случае, если граница аппроксимируется N постоянными элементами, представляет собой систему линейных алгебраических уравнений (СЛАУ):

$$\underbrace{\begin{bmatrix} \Delta U_{ij}^{pq} \\ \Delta T_{ij}^{pq} \end{bmatrix}}_{2N\times 2N} \times \underbrace{\left[\varphi_j^q\right]}_{2N} = \underbrace{\begin{bmatrix} u_i^p \\ t_i^p \end{bmatrix}}_{2N}, \qquad (3)$$

коэффициенты которой

$$\Delta U_{ij}^{pq} = \int_{\Delta\Gamma} U_{ij}(x_0^p, \xi^q) d\Gamma(\xi^q), \quad \Delta T_{ij}^{pq} = \int_{\Delta\Gamma} T_{ij}(x_0^p, \xi^q) d\Gamma(\xi^q) \qquad (4)$$

являются интегралами от фундаментальных сингулярных решений линеаризованной системы уравнений Навье – Стокса для скоростей U_{ij} и усилий T_{ij} соответственно, по элементам границы области решения $\Delta\Gamma$; p и q – номера элементов, $x_0^p \in \Gamma$ – точка приложения нагрузки, $\xi^q \in \Gamma$ – точка наблюдения, u_i^p, t_i^p – значения скоростей и

усилий, известные на границе. Матрица системы (3) является несимметричной, полностью заполненной с диагональным преобладанием (рис. 2). В результате решения СЛАУ (3) находится вектор-столбец φ_j^q, свертка которого с соответствующими фундаментальными решениями дает значения скоростей и напряжений в любой точке области. В рассматриваемой задаче решением является профиль скорости u_1 на Γ_3.

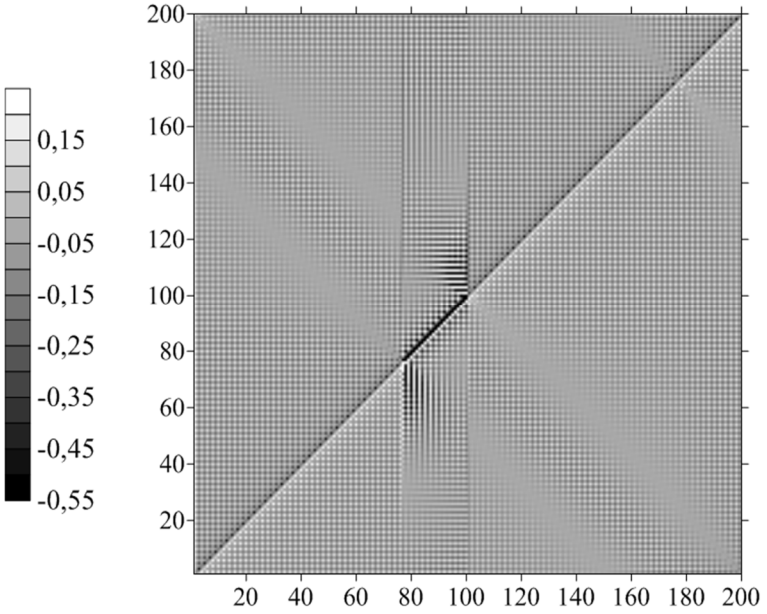


Рис. 2. Портрет матрицы при N=100

Алгоритм, в соответствии с которым работает программа расчета, выглядит следующим образом:

1. Инициализация данных: чтение параметров расчета, задание области решения и граничных условий;

2. Формирование матрицы СЛАУ: вычисление коэффициентов.

3. Решение СЛАУ в два этапа: факторизация матрицы и прямое решение СЛАУ с факторизованной матрицей;

4. Расчет профиля скорости на выходе из канала, вычисление отклонения от аналитического решения.

Результаты работы последовательного алгоритма

В последовательном алгоритме для решения СЛАУ использовались стандартные последовательные процедуры, реализующие метод Гаусса с выбором главного элемента [5]. Расчеты проведены на суперкомпьютере СКИФ «Cyberia». Для каждого расчета задействован узел, включающий 2 6-ядерных процессора Intel® Xeon™ 3670 2.99 ГГц. Время выполнения алгоритма в секундах увеличивается с ростом числа граничных элементов как $t_p \approx 5 \cdot 10^{-9} N^{2.87}$. Отношения времен выполнения отдельных блоков алгоритма ко времени вычислений показаны на рис. 3.

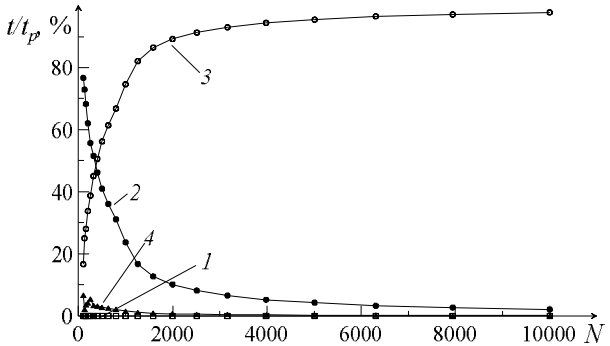


Рис. 3. Зависимость относительных времен работы отдельных блоков алгоритма (*1* – инициализация данных, *2* – формирование матрицы, *3* – решение СЛАУ, *4* – расчет профиля на выходе)

Из рис. 3 следует, что основное время тратится на формирование матрицы и решение СЛАУ.

Результаты работы параллельного алгоритма

На этапе выполнения двух наиболее ресурсоемких частей алгоритма реализованы параллельные вычисления с использованием средств распределенных вычислений для систем с общей памятью, таким образом, что для последовательного и параллельного расчета использовался один и тот же код. Для решения СЛАУ использован модуль линейной алгебры LAPACK из библиотеки Intel® Math Kernel Library [6], входящей в Intel® Visual Fortran Composer XE 2011, его последовательная и параллельная

реализации. Решение СЛАУ осуществляется посредством последовательного вызова двух библиотечных процедур. Для процедуры формирования матрицы СЛАУ использована технология OpenMP.

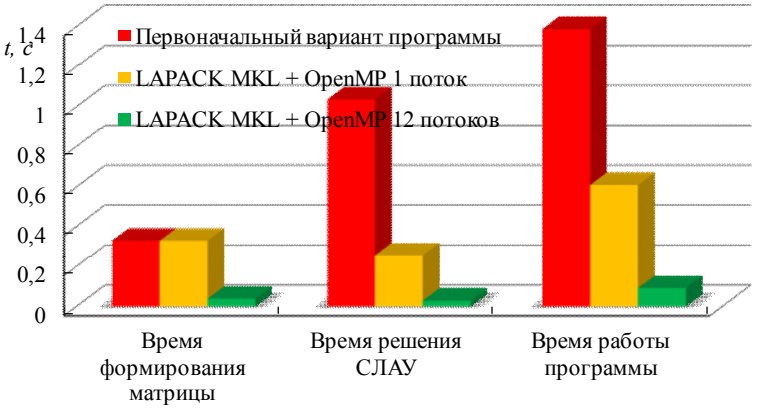


Рис. 4. Время работы программы и ее отдельных блоков при $N = 1000$

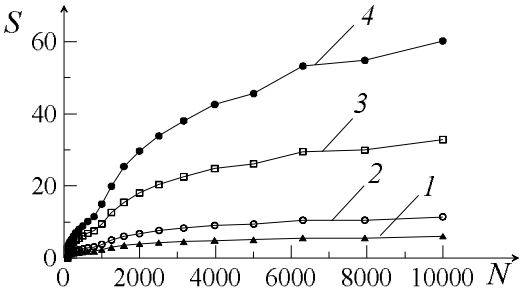


Рис. 5. Ускорение вычислений по сравнению с первоначальным вариантом программы (*1* – 1 поток, *2* – 2 потока, *3* – 6 потоков, *4* – 12 потоков)

Итерации цикла, в котором рассчитываются значения коэффициентов матрицы СЛАУ, распределяются между доступными потоками, для этого используется всего несколько директив OpenMP. Вычислительный эксперимент показал, что оптимальным является статическое распределение итераций

соответствующего цикла (до 50% эффективнее динамического распределения). Количество потоков выбиралось равным числу задействованных ядер процессоров. В случае превышения числа потоков над числом доступных ядер время расчетов не увеличивалось. Результаты представлены на рис. 4–6.

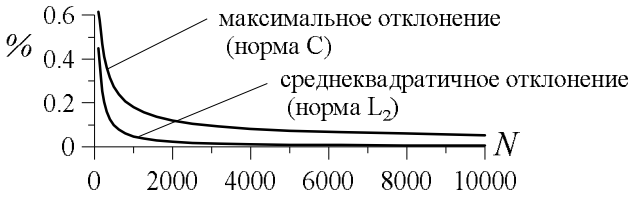


Рис. 6. Сравнение с аналитическим решением

Таким образом, применение параллельных вычислений для рассматриваемого алгоритма позволило добиться значительного ускорения проведения расчетов с использованием многоядерной архитектуры системы с общей памятью, не прибегая к значительному изменению программного кода. Так, использование модуля LAPACK [7] даже в однопоточном режиме ускоряет время решения СЛАУ приблизительно в 4–5 раз (рис. 5, кривая *1*). Суммарно полученное ускорение в случае совместного использования модуля LAPACK для решения СЛАУ и OpenMP для формирования матрицы на указанной системе достигает нескольких десятков раз.

Литература

1. *Гергель В.П.* Современные языки и технологии параллельного программирования: Учебник / Предисл.: В.А. Садовничий. М.: Изд-во МГУ, 2012. 408 с.
2. *Антонов А.С.* Технологии параллельного программирования MPI и OpenMP: Учеб. пособие / Предисл.: В.А. Садовничий. М.: Изд-во МГУ, 2012. 344 с.
3. *Беликов Д.А., Говязов И.В., Данилкин Е.А. и др.* Высокопроизводительные вычисления на кластерах: Учеб. пособие/ под ред. А.В. Старченко. – Томск: Изд-во Том. ун-та, 2008. – 198 с.
4. *Пономарева М.А., Якутенок В.А.* Метод граничных элементов для решения уравнений Стокса. Моделирование плоских

течений вязкой жидкости со свободной поверхностью. LAMBERT Academic Publishing, 2013. 160 с.
5. *Форсайт Дж., Малькольм М., Моулер К.* Машинные методы математических вычислений. М.: Мир, 1980.
6. http://software.intel.com/sites/products/documentation/doclib/mkl_sa/11/mkl_lapack_examples

Научное издание

Седьмая Сибирская конференция
по параллельным и высокопроизводительным вычислениям

Томск, 12−14 ноября 2013 года

Редактор **В.Г. Лихачёва**
Компьютерная верстка **А.А. Барт**

Подписано в печать 18.04.13 г.
Формат 60x84 1/16. Бумага офсетная №1.
Печать офсетная. Гарнитура «Times».
Печ. л. 8,9; уч.-изд. л. 8,5; усл. печ. л. 8,0.
Заказ Тираж 100 экз.

ОАО «Издательство ТГУ», 634029, г. Томск, ул. Никитина, 4
Типография ООО «Интегральный переплет», 634040, г. Томск, ул. Высоцкого, 28, стр.1

1878–2013

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

VII СКПВВ

12 - 14 ноября 2013, Томск

http://ssspc.math.tsu.ru

МЕХАНИКО МАТЕМАТИЧЕСКИЙ факультет

65 лет ММФ