

РЕАЛИЗАЦИЯ АЛГОРИТМА КЛЕТОЧНО-АВТОМАТНОЙ ИНТЕРФЕРЕНЦИИ НА МУЛЬТИКОМПЬЮТЕРАХ

В. П. Маркова, М. Б. Остапкевич

Институт вычислительной математики и математической геофизики СО РАН, 630090, Новосибирск

УДК 519.688

В работе предлагается параллельная реализация алгоритма клеточно-автоматной интерференции двух волн с использованием технологии фрагментированного программирования и базирующейся на ней системы LuNA. Технология базируется на стратегии управления потоками данных. В отличие от большинства современных технологий, технология фрагментированного программирования и LuNA наряду с такими технологиями, как StreamIt и Lift реализуют унифицированный подход к реализации параллельных программ на гетерогенных мультимикросистемах. Программа на LuNA содержит описания фрагментов данных, фрагментов вычислений и информационные зависимости между ними. Результаты сравнения параллельных реализаций в LuNA и MPI показали, что время выполнения программы в текущей версии LuNA больше, чем в MPI. Это связано с особенностями алгоритмов распределения, поиска и пересылок фрагментов данных и фрагментов вычислений на кластере. Сложность построения LuNA программы существенно ниже, чем программы.

Ключевые слова: параллельное программирование, фрагментированное программирование, система LuNA, клеточный автомат, решеточный газ, моделирование интерференции.

Введение

В настоящее время для написания параллельных программ на мультимикросистемах (компьютерах с распределенной памятью) используется стандарт MPI. Он получил самое широкое распространение на большинстве мультимикросистем с однородными узлами благодаря тому, что он позволяет создавать переносимые программы. Используя MPI, программист при написании программ помимо описания собственно вычислений должен программировать выделение ресурсов, межузловые коммуникации, балансировку нагрузки между узлами и определять порядок вычислений.

С появлением компьютеров с гетерогенными узлами задача распараллеливания усложнилась, так как разные вычислители в составе узла имеют разную архитектуру и каждый из них программируется с использованием отдельного интерфейса (технологии). Построение эффективной программы для таких компьютеров выполняется двумя способами. Первый заключается в использовании MPI для организации межузлового параллелизма, а OpenMP, OpenCL, CUDA, HLS — для организации внутриузлового параллелизма.

Второй способ предполагает построение единой технологии реализации параллельных программ на гетерогенном мультимикросистеме. Примерами таких технологий являются StreamIt [1, 2] и Lift [3]. В рамках второго способа в ИВМиМГ СО РАН была реализована технология фрагментированного программирования, основанная на стратегии управления потоками данных. На ее основе построена система программирования LuNA [4].

В статье исследованы реализации алгоритма клеточно-автоматной интерференции двух волн. Одна реализация построена с использованием MPI, а другая — на базе системы LuNA. перечислены характеристики системы LuNA. Проведено их сравнение, и сформулированы преимущества использования системы LuNA.

1 Основные определения и характеристики системы LuNA

Система LuNA — это инструмент для построения параллельных программ на базе технологии фрагментированного программирования. LuNA система состоит из транслятора языка сборки и подсистемы исполнения фрагментированных программ. Базовыми понятиями в системе являются фрагменты данных, фрагменты кода и фрагменты вычислений. *Фрагмент данных* — это множество соседних ячеек массива заданного размера. *Фрагмент кода* — это функция, которая получает значения некоторых входных фрагментов данных и вычисляет по ним значения выходных фрагментов данных. При построении фрагментированных программ (ФП) используются два типа фрагментов кода: атомарные и структурированные. *Атомарные фрагменты кода* в системе представляются как функции Си программ. *Структурированные фрагменты кода* содержат сборку из фрагментов вычислений. *Фрагмент вычислений* — это обращение (вызов на исполнение) к фрагменту кода с указанием имен всех входных и выходных фрагментов данных. Написание LuNA-программы состоит из следующих шагов.

- В исходной последовательной программе на языках Си/Си++ выделяются участки кода, которые отвечают за вычисления. В этих участках кода обработка произвольных объектов данных заменяется обработкой фрагментов данных, а сами участки оформляются как атомарные фрагменты кода.
- На языке сборки LuNA описываются структурированные фрагменты кода и информационные зависимости между фрагментами вычислений и фрагментами данных. В отличие от MPI программы, в сборочной программе (это синоним LuNA-программы) нет необходимости жестко задавать порядок вычислений, управлять выделением ресурсов и программировать межузловые коммуникации (см. табл. 1).
- Вставляются инструкции, которые обеспечивают освобождение памяти, занимаемой фрагментами данных, которые больше не используются.
- Определяются размеры фрагментов данных, при которых время выполнения минимально.

На вход системы LuNA поступает фрагментированная программа на языке сборки и модуль на Си/Си++, реализующий атомарные фрагменты кода. Эта программа содержит все фрагменты кода, фрагменты данных и информационные зависимости между ними. Система LuNA выполняет следующие шаги.

- Транслятор системы преобразует исходную LuNA-программу во внутреннее представление, с которым может работать исполнительная подсистема.
- Исполнительная подсистема LuNA автоматически распределяет фрагменты кода и данных по узлам в соответствии с имеющимися ресурсами мультимедийного компьютера. При программировании с использованием MPI программист должен сам проводить такое распределение (см. табл. 1).
- При выполнении фрагментированной программы исполнительная подсистема определяет окончательный порядок выполнения фрагментов вычислений. Для этого она ведет учет фрагментов кода, готовых к исполнению, и распределяет их на исполнение по потокам. При выполнении программы на нескольких узлах система автоматически обеспечивает необходимую пересылку фрагментов данных между узлами. При написании же MPI программы разработчик в явном виде определяет порядок выполнения и описывает все межузловые пересылки данных (см. табл. 1). При несбалансированности вычислительной нагрузки на узлы, исполнительная подсистема перераспределяет нагрузку между узлами. В MPI такую балансировку нагрузки реализует сам программист в своей программе.
- Исполнение фрагментированной программы продолжается, пока есть исполняющиеся или готовые к исполнению фрагменты кода.

2 Клеточно-автоматное моделирование волновых процессов

Для моделирования интерференции двух волн используется простейший класс одночастичных недетерминированных клеточных автоматов с одной частицей покоя массой 2, известный в литературе как HPP1rp [5]. Клеточные автоматы (КА) описывают природные явления множеством гипотетических частиц, которые

Таблица 1: Реализация системных функций в MPI и LuNA

Функция	MPI	LuNA
порядок вычислений	Жестко задается программистом, неизменен при выполнении.	Окончательный порядок определяется уже при исполнении, когда есть информация о фрагментах кода, для которых готовы и собраны в одном узле все исходные данные и которые могут запускаться.
управление выделением ресурсов в узле	Реализуется программистом.	Реализуется системой LuNA.
распределение данных по узлам	Реализуется программистом.	Выполняется системой на основании информации о доступных ресурсах вычислительной системы.
межузловые коммуникации	Явно кодируются программистом.	Активизируются подсистемой исполнения на основе информации о местонахождении затребованных данных.
балансировка нагрузки	Не реализуется или реализуется программистом.	Реализуется системой LuNA.

движутся в решеточном пространстве по некоторым правилам. Эти правила представляют моделируемое явление на микроуровне, исходя из общих законов физики.

Интерес к клеточно-автоматному (КА) моделированию объясняется рядом его 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.

Клеточный автомат HPP1gr определен на 2D решетке. Каждый узел решетки соединен с соседями единичными решеточными векторами e_i , $i = 1, 2, 3, 4$, и содержит 4 движущиеся частицы и частицу покоя. Движущиеся частицы имеют единичную массу и единичную скоростью c_i направленную вдоль одного из четырех векторов решетки. В каждом направлении может двигаться только одна из частиц, находящихся в узле решетки. Частица покоя имеет массу 2 и нулевую скорость.

Каждому узлу решетки с именем r поставлена в соответствие клетка с тем же именем. Множество частиц в клетке определяет ее состояние $s(r)$. Вектор $s(r)$ состоит из 5 элементов. Значение первых четырех элементов вектора s показывает наличие ($s_i(r)=1$) или отсутствие ($s_i(r)=0$) движущейся частицы со скоростью c_i в клетке с именем r последний элемент вектора $s(r)$ показывает наличие $s_i(r)=1$ или отсутствие $s_i(r)=0$ частицы покоя массой 2 в клетке с именем r . Пара (s, r) называется *клеткой*. Общая сумма масс частиц в клетке с именем r называется *модельной плотностью* $\rho(r)$ Множество клеток, в котором все клетки имеют уникальные имена, образуют *клеточный массив*. Множество состояний всех клеток массива $\omega(t)$ в момент времени t называется *глобальным состоянием* автомата. Смена глобальных состояний автомата описывает *эволюцию* КА.

Клеточный автомат работает синхронно: все клетки автомата меняют свои состояния одновременно на каждом итерационном шаге. Шаг состоит из 2-х фаз: столкновение и сдвиг. На фазе *столкновения* частицы в каждой клетке соударяются друг с другом таким образом, что суммарные масса и импульс частиц сохраняются. Функция столкновения создает или разрушает движущуюся частицу со скоростью в клетке с именем r в момент времени t и зависит только от ее исходного состояния в данный момент времени. Например, клетка в состоянии $(00101_2=5_{10})$ на фазе столкновения меняет его на одно из трех состояний: $(00101_2=10_{10})$ с вероятностью $p_{5 \rightarrow 10}$ (рис. 1а), $(10000_2=16_{10})$ с вероятностью $p_{5 \rightarrow 16}$ (рис.1б) и остается в своем состоянии с вероятностью $p_{5 \rightarrow 5}$ (рис.1в). На фазе *сдвига* движущиеся частицы в каждой клетке сдвигаются в сторону ближайшего соседа с единичной скоростью.

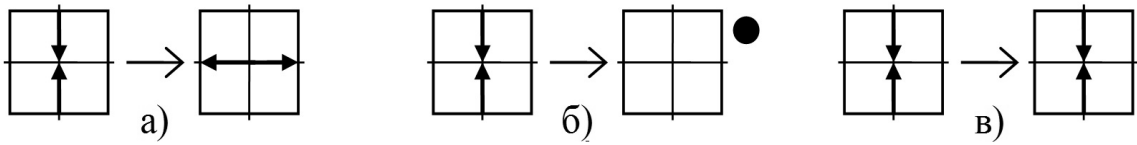


Рис. 1: Пример правила столкновения: а) переход $5 \rightarrow 10$, б) переход $5 \rightarrow 16$, в) переход $5 \rightarrow 5$.

3 Моделирование интерференции двух волн

Интерференция двух волн моделируется эволюцией клеточного автомата размера 1400×1200 и плотности клеток среды 3.3. Два круговых источника периодических волн находятся на расстоянии четырех длин волны (целое число длин волн — условие существования конструктивной интерференции [6]). На рис. 2 показаны две похожие интерференционные картины двух волн, которые получены разными способами и на разных шагах моделирования. Первая промоделирована традиционным способом на шаге времени 100 (рис. 2а), а вторая — клеточно-автоматным способом на 975-м шаге эволюции клеточного автомата (рис. 2б).

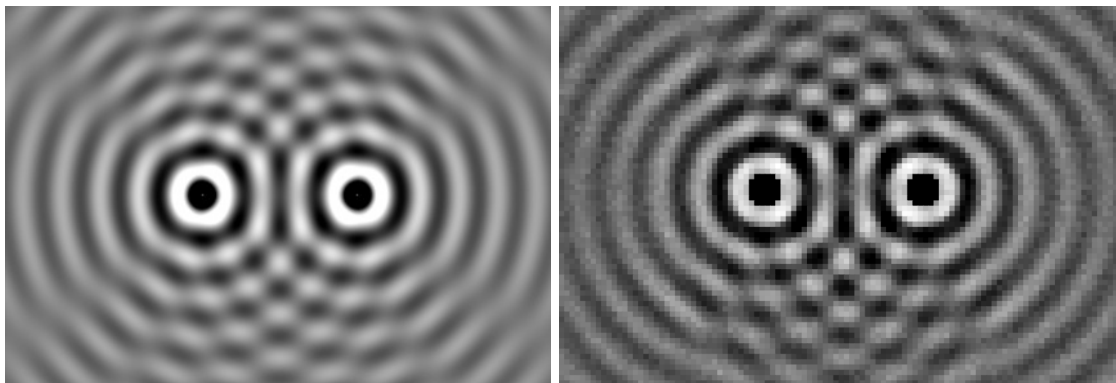


Рис. 2: Интерференция по формуле сферической волны (а) и в модели HPPgr (б)

4 Описание фрагментированного алгоритма КА-интерференции

В реализации исходный клеточный массив разбивается на фрагменты данных с топологией линейка. Главными фрагментами данных в реализации являются:

- $a[t][x]$ — фрагменты, хранящие начальное состояние клеточного массива на шаге моделирования t ,
- $b[t][x]$ — фрагменты, хранящие состояние клеточного массива после вычисления столкновения частиц на шаге моделирования t .
- $bu[t][x]$, $bd[t][x]$ — фрагменты для хранения теневых граней.

Во фрагменте $bu[t][x]$ дублируется верхняя строка фрагмента $b[t][x]$. Во фрагменте $bd[t][x]$ дублируется нижняя строка фрагмента $b[t][x]$. Введение фрагментов данных $bu[t][x]$, $bd[t][x]$ позволяет уменьшить объем пересылок между узлами, так как их размер существенно меньше, чем у фрагмента $b[t][x]$.

Индекс x определяет положение фрагмента в пространстве. Для фрагмента, содержащего самые верхние строки клеточного массива, значение x равно 0. x для фрагмента, содержащего самые нижние строки массива, равно числу фрагментов, на которое разбит массив минус единица.

Фрагментированная реализация алгоритма КА-интерференции представлена фрагментами кода:

- `init` — инициализация начального состояния клеточного массива,
- `collision` — вычисление операции столкновения частиц в клетках,

- propagation — вычисление перелета частиц между клетками,
- print — печать результата моделирования в файл.

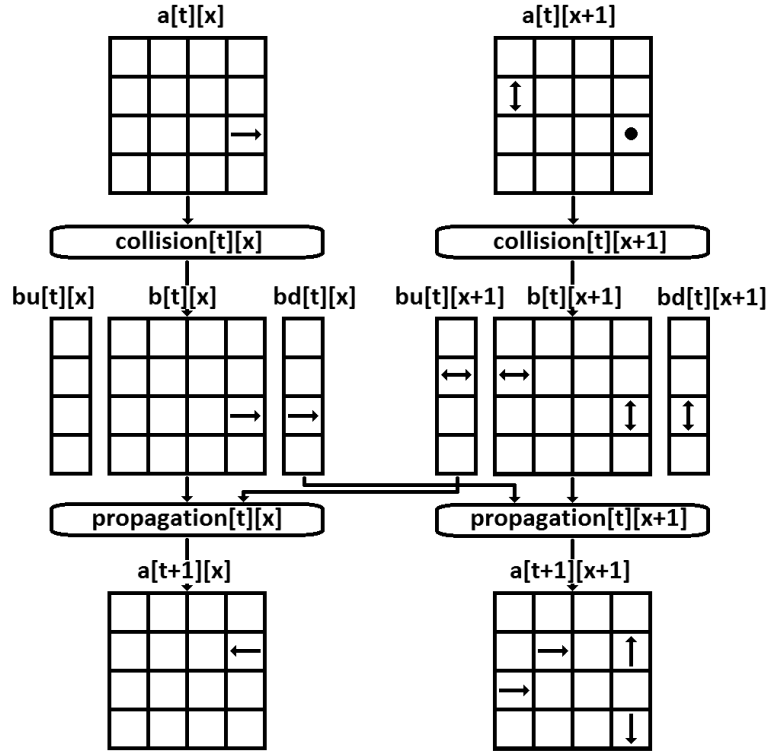


Рис. 3: Часть графа информационных зависимостей реализации HPPrg в LuNA

На рис. 3 показан граф информационных зависимостей для двух исходных фрагментов данных $a[t][x]$ и $a[t][x+1]$ на шаге времени t . Они являются исходными для фрагментов вычислений $\text{collision}[t][x]$ и $\text{collision}[t][x+1]$. Как только $a[t][x]$ и $a[t][x+1]$ вычислены, $\text{collision}[t][x]$ и $\text{collision}[t][x+1]$ готовы к выполнению. В какой-то момент времени они запускаются на исполнение. После завершения выполнения $\text{collision}[t][x]$ создаются фрагменты данных $b[t][x]$, $bu[t][x]$, $bd[t][x]$. Они хранят состояние частей клеточного массива на шаге t после применения правила коллизии. Аналогично, после завершения $\text{collision}[t][x+1]$ создаются $b[t][x+1]$, $bu[t][x+1]$, $bd[t][x+1]$. В этот момент $a[t][x]$ и $a[t][x+1]$ больше не будут использоваться никакими фрагментами вычислений, и могут быть удалены из системы, а освободившаяся память использоваться для хранения новых данных. Когда вычислены $bd[t][x-1]$, $b[t][x]$, $bu[t][x+1]$, фрагмент вычислений $\text{propagation}[t][x]$ готов к исполнению. Аналогично, $\text{propagation}[t][x+1]$ может запускаться, когда посчитаны $bd[t][x]$, $b[t][x+1]$, $bu[t][x+2]$. Результатом работы $\text{propagation}[t][x]$, $\text{propagation}[t][x+1]$ являются $a[t+1][x]$ и $a[t+1][x+1]$, хранящие состояние клеточного массива на шаге времени $t+1$.

LuNA-программа реализует КА-интерференцию в клеточном массиве размером 4096×4096 . На рис. 4 показана зависимость времени выполнения программы от размера фрагментов. Замеры для выявления этой зависимости проводились при числе узлов 1, потоков 8. Минимальное время наблюдается при размере фрагментов 256. При уменьшении размера фрагментов увеличивается их число и растут накладные расходы системы на их обработку. При увеличении размера фрагментов количество обрабатываемых потоком фрагментов становится слишком маленьким, и система менее качественно балансирует вычислительную нагрузку между потоками.

Результаты сравнения времени выполнения реализаций на MPI и LuNA представлены на рис. 5. Время выполнения LuNA реализации при использовании нескольких узлов существенно превышает время выполнения MPI реализации. Уменьшение времени выполнения может быть достигнуто за счет оптимизации алгоритмов распределения, поиска и передачи фрагментов данных между узлами мультикомпьютеров и алгоритмов управления выделением памяти для фрагментов данных внутри узлов.

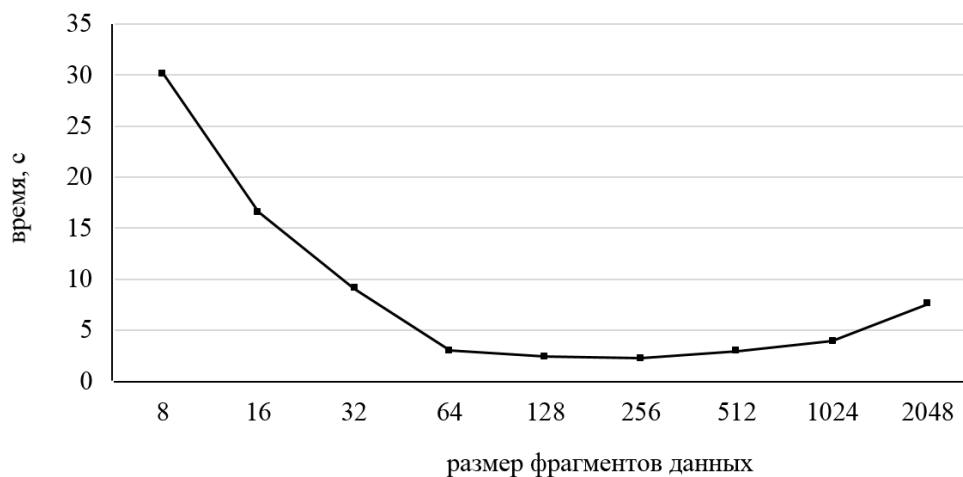


Рис. 4: Зависимость времени выполнения программы от размера фрагмента

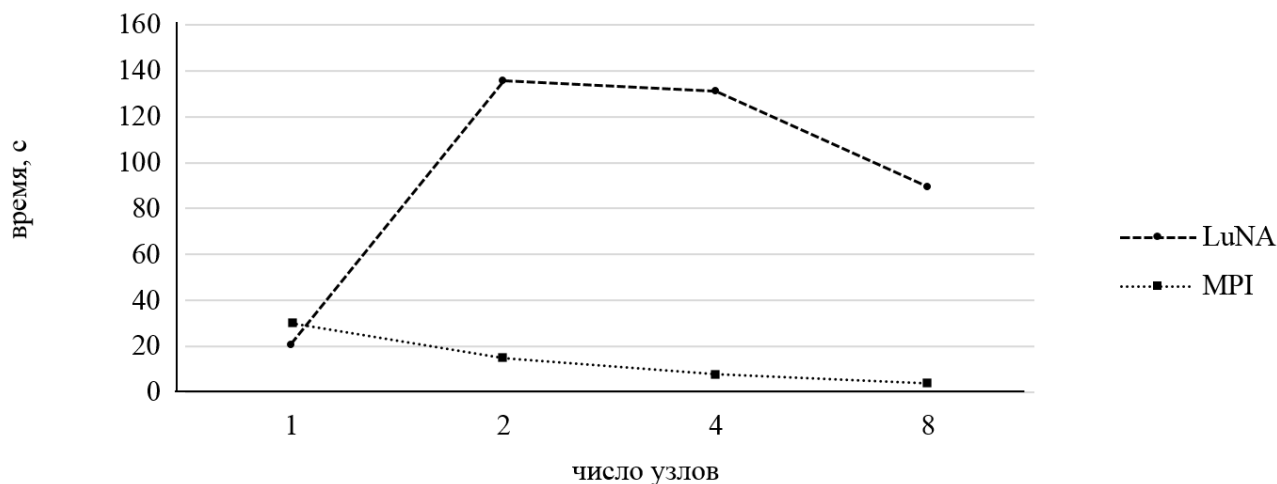


Рис. 5: Зависимость времени выполнения программы от числа используемых узлов

Заключение

Опыт реализации программ в системе LuNA выявил ее основные достоинства и недостатки. Главным недостатком является то, что текущая реализация LuNA существенно проигрывает MPI в плане времени выполнения программ. Рассмотрим достоинства системы LuNA.

Для задач с неравномерной вычислительной нагрузкой на узлы использование встроенной в LuNA динамической балансировки дает сопоставимый по времени выполнения результат по сравнению с ручной реализацией балансировки в MPI программе [7].

LuNA-программы не имеют такой привязки к архитектуре и доступным ресурсам компьютера, как MPI-программы. Поэтому, LuNA программы обладают существенно более высокой переносимостью.

Отсутствие необходимости задавать порядок вычислений и программировать межузловые коммуникации существенно упрощает программирование в LuNA по сравнению с MPI. Эти же особенности LuNA устраняют появление некоторых видов ошибок, свойственных при программировании в MPI. Описание информационных зависимостей между фрагментами имеет локальный характер, в отличие от задания порядка вычислений в MPI. Это упрощает отладку LuNA программ. Все это вместе уменьшает время разработки параллельных реализаций вычислительных алгоритмов в системе LuNA по сравнению с MPI.

Список литературы

- [1] StreamIt Project Homepage, <http://groups.csail.mit.edu/cag/streamit/>
- [2] William Thies, Michal Karczmarek, Saman P. Amarasinghe. StreamIt: A Language for Streaming Applications. Proceeding CC '02 Proceedings of the 11th International Conference on Compiler Construction, 2002, P. 179–196.
- [3] Michel Steuwer, Toomas Remmelg, Chistophe Dubach.: Lift: a functional data-parallel IR for high-performance GPU code generation. CGO'17 Proceedings of the 2017 International Symposium on Code Generation and Optimization, P. 74–85.
- [4] Victor Malyshkin.: Active Knowledge, LuNA and Literacy for Oncoming Centuries. LNCS, V. 9465. P. 292–303. Springer, Heidelberg, 2015.
- [5] M. Zhang, D. Cule, L. Shafai, G. Bridges and N. Simons. Computing Electromagnetic Fields in Inhomogeneous Media Using Lattice Gas Automata. In: Proceedings of 1998 Symposium on Antenna Technology and Applied Electromagnetics, Aug. 14–16, Ottawa, Canada.
- [6] Conditions for interference, http://physics.bu.edu/~duffy/sc545_notes09/interference_conditions.html
- [7] V. Malyshkin, V. Perepelkin The PIC implementation in LuNA system of fragmented programming // Journal of Supercomputing. 2014. V. 69, iss. 1. P. 89–97

*Валентина Петровна Маркова — к.ф.-м.н., ст. науч.сотр. Института
вычислительной математики и математической геофизики СО РАН;
e-mail: markova@ssd.sccc.ru;*

*Михаил Борисович Остапкевич — мл. науч.сотр. Института
вычислительной математики и математической геофизики СО РАН;
e-mail: ostap@ssd.sccc.ru.*

Дата поступления — 30 мая 2017 г.