УДК 519.612:632.4

МОДЕЛИРОВАНИЕ ТРЕХМЕРНЫХ ТЕЧЕНИЙ МЕТОДОМ РАСЩЕПЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ ПАРАЛЛЕЛЬНОЙ АРХИТЕКТУРЫ ГПУ

С. Б. Березин[1], **В. М. Пасконов**[1], **Н. А. Сахарных**[1]

Предложена эффективная реализация неявного численного метода покоординатного расщепления в трехмерной области с применением графических процессоров (ГПУ). Рассмотрены применение метода покоординатного расщепления для решения полной системы уравнений Навье–Стокса, а также основные детали реализации быстрого алгоритма для решения трехдиагональных систем на CUDA. Выполнено сравнение производительности ГПУ и ЦПУ на модельной задаче и продемонстрировано, что ГПУ позволяют достичь ускорения расчетов в двойной точности на порядок по сравнению с последними моделями многоядерных ЦПУ. Работа выполнена при поддержке РФФИ (коды проектов 10–01–00288-а и 09–07–00424-а).

Ключевые слова: метод расщепления, параллельные вычисления, CUDA, ГПУ, уравнения Навье–Стокса.

1. Введение. Моделирование аэродинамических течений является одной из самых сложных областей моделирования явлений природы, требующих огромного числа вычислительных ресурсов и высокой пропускной способности памяти устройства. Даже самые мощные суперкомпьютеры, состоящие из большого числа вычислительных узлов, позволяют получить только приближенные решения движений потоков в общем случае.

Несколько лет назад появились графические ускорители, которые затем эволюционировали в массивно-параллельные процессоры общего назначения (ГПУ). Быстрый рост производительности графических процессоров в течение последнего десятилетия открыл новые возможности для использования ГПУ в реальных приложениях, требующих больших вычислительных мощностей. Ранее к решению подобных задач можно было прийти только путем использования суперкомпьютеров и кластеров с большим числом процессоров. Сейчас в нашем распоряжении находятся более энергоэффективные и компактные ГПУ с массивно-параллельной архитектурой, которые могут использоваться как сопроцессоры, значительно ускоряя расчеты.

На текущий момент существует множество удобных инструментов для разработки приложений общего назначения для ГПУ. Зарекомендовавшей себя программной моделью является CUDA, каждая новая версия которой отражает последние нововведения в архитектуре моделей ГПУ [1]. Кроме того, существуют удобные отладчики и профилировщики для программирования на ГПУ. Эти средства значительно упростили разработку сложных приложений, эффективно использующих параллельную архитектуру ГПУ.

Методы расщепления хорошо известны в литературе. Методы расщепления по физическим процессам успешно применялись для решения климатических задач и расчета циркуляции океана [2]. Кроме того, схемы, основанные на специальном расщеплении операторов, эффективно использовались при решении уравнений Навье–Стокса [3]. Метод покоординатного расщепления применялся для исследования течений в плоском двумерном канале в работе [4]. В настоящей статье предложена эффективная реализация метода покоординатного расщепления для полной системы уравнений Навье–Стокса в трехмерной области на массивно-параллельной архитектуре ГПУ.

2. Сравнение методов моделирования. Существует множество различных подходов к моделированию течений; условно их можно разбить на три группы: прямое моделирование (DNS), моделирование больших вихрей (LES) и усредненные уравнения Навье–Стокса (RANS). Каждый из этих методов имеет как свои преимущества, так и недостатки. Здесь мы будем рассматривать только прямое моделирование. Однако следует отметить, что схожие алгоритмы для ГПУ можно использовать и для ускорения других методов.

[1] Московский государственный университет им. М. В. Ломоносова, факультет вычислительной математики и кибернетики, Ленинские горы, д. 1, стр. 52, 119991, Москва; С. Б. Березин, доцент, e-mail: bs@mslab.cs.msu.su; В. М. Пасконов, профессор, e-mail: paskonov@cs.msu.su; Н. А. Сахарных, аспирант, e-mail: nikolai.sakharnykh@gmail.com

Подробное описание метода прямого моделирования, его преимуществ и недостатков приведено в [5]. Это — наиболее точный подход, учитывающий все масштабы турбулентности, напрямую решающий уравнения без использования дополнительных моделей турбулентности. С другой стороны, этот метод требует значительной вычислительной мощности и использования сеток высокого разрешения. С увеличением числа Рейнольдса отношение размеров больших вихрей к малым начинает расти, что создает дополнительные трудности для прямого моделирования турбулентности таких потоков. Таким образом, для больших чисел Рейнольдса приближенные решения, моделирующие только вихри, соответствующие большим энергиям (такие как LES), являются наиболее предпочтительными. Прямое моделирование можно рассматривать как наилучшее решение для исследования турбулентного потока, хотя оно и требует наибольших вычислительных затрат, за ним следует LES — менее сложное и далее RANS — наиболее простое решение (но в то же время наименее точное).

Таким образом, метод прямого моделирования хорошо подходит для детального исследования течений, изучения свойств и поиска закономерностей, но редко используется для прикладных инженерных задач. Рост производительности ГПУ делает возможной разработку новых численных методов, с помощью которых можно преодолеть этот недостаток и получать результаты прямого моделирования гораздо быстрее.

3. Полная система уравнений Навье–Стокса. Прямое моделирование предполагает использование полной системы уравнений Навье–Стокса. Для вязких газов такая система состоит из уравнений неразрывности, движения и энергии. Дополнительно вводится уравнение состояния, которое линейно связывает давление и температуру, тем самым замыкая нашу систему относительно скорости и температуры потока. Мы будем рассматривать течения, скорость которых гораздо меньше скорости звука, поэтому изменением плотности можно пренебречь. Все величины нормированы, а плотность положена равной 1. Тогда систему уравнений можно записать в безразмерных величинах следующим образом:

$$\operatorname{div} \boldsymbol{u} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u}\,\nabla \boldsymbol{u} = -\nabla T + \frac{1}{\mathrm{Re}}\,\nabla^2 \boldsymbol{u}, \tag{2}$$

$$\frac{\partial T}{\partial t} + \boldsymbol{u}\,\nabla T = \frac{1}{\mathrm{Re}\ \mathrm{Pr}}\,\Delta T + \frac{\gamma - 1}{\gamma\ \mathrm{Re}}\,\Phi, \tag{3}$$

$$p = \rho R T = R T.$$

Здесь $\boldsymbol{u}$ — вектор скорости, T — температура, p — давление, ρ — плотность. Кроме того, в системе присутствует несколько параметров подобия потока, таких как число Рейнольдса (Re), число Прандтля (Pr), газовая постоянная (R) и удельная теплоемкость (γ). Диссипативная функция Φ, возникающая из-за необратимой работы вязких сил, может быть представлена в виде $\Phi = \Phi_x + \Phi_y + \Phi_z$, $\Phi_x = 2\left(\dfrac{\partial u}{\partial x}\right)^2 + \left(\dfrac{\partial v}{\partial x}\right)^2 + \left(\dfrac{\partial w}{\partial x}\right)^2 + \dfrac{\partial u}{\partial y}\,\dfrac{\partial v}{\partial x} + \dfrac{\partial u}{\partial z}\,\dfrac{\partial w}{\partial x}$, $\Phi_y = \left(\dfrac{\partial u}{\partial y}\right)^2 + 2\left(\dfrac{\partial v}{\partial y}\right)^2 + \left(\dfrac{\partial w}{\partial y}\right)^2 + \dfrac{\partial v}{\partial x}\,\dfrac{\partial u}{\partial y} + \dfrac{\partial v}{\partial z}\,\dfrac{\partial w}{\partial y}$, $\Phi_z = \left(\dfrac{\partial u}{\partial z}\right)^2 + \left(\dfrac{\partial v}{\partial z}\right)^2 + 2\left(\dfrac{\partial w}{\partial z}\right)^2 + \dfrac{\partial w}{\partial x}\,\dfrac{\partial u}{\partial z} + \dfrac{\partial w}{\partial y}\,\dfrac{\partial v}{\partial z}$.

Для нахождения неизвестных скоростей и температуры необходимо решить нелинейные уравнения (2) и (3). Уравнение (1) используется для проверки численной ошибки после каждого временно́го шага. Для нахождения ошибки рассчитывается усредненный интеграл от дивергенции скорости по поверхности каждого узла сетки. Подробный вывод уравнений, их описание и применение можно найти в [6].

4. Численный метод расщепления. Численный метод покоординатного расщепления представляет собой конечно-разностный метод, который чаще всего применяется для решения параболических или эллиптических уравнений в частных производных. В общем случае этот метод может быть применен к задаче любой размерности. Различные применения методов расщепления и их характеристики описаны в [7]. Основная идея метода заключается в расщеплении конечно-разностных уравнений на несколько более простых — по уравнению вдоль каждой из осей координат. Эта операция проводится таким образом, что производные вдоль соответствующего направления берутся неявно, а остальные координаты считаются постоянными. В случае системы Навье–Стокса уравнение

$$\frac{\partial u}{\partial t} + u\,\frac{\partial u}{\partial x} + v\,\frac{\partial u}{\partial y} + w\,\frac{\partial u}{\partial z} = -\frac{\partial T}{\partial x} + \frac{1}{\mathrm{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

для X-компоненты скорости для векторного уравнения (2) после применения данного метода будет рас-

щепляться на три уравнения следующим образом:

$$\frac{\partial u}{3\partial t} + u\frac{\partial u}{\partial x} = -\frac{\partial T}{\partial x} + \frac{1}{\mathrm{Re}}\frac{\partial^2 u}{\partial x^2}, \tag{4}$$

$$\frac{\partial u}{3\partial t} + v\frac{\partial u}{\partial y} = \frac{1}{\mathrm{Re}}\frac{\partial^2 u}{\partial y^2}, \tag{5}$$

$$\frac{\partial u}{3\partial t} + w\frac{\partial u}{\partial z} = \frac{1}{\mathrm{Re}}\frac{\partial^2 u}{\partial z^2}. \tag{6}$$

Остальные уравнения также могут быть разложены подобным образом. На каждом шаге расщепления уравнения (4)–(6) решаются последовательно таким образом, что для расчетов используются уже посчитанные значения величин с предыдущего слоя. Для решения каждого из уравнений используется неявная конечно-разностная схема первого порядка по времени и второго порядка по пространственным величинам. После применения дискретизации для каждого направления получается набор независимых трехдиагональных систем. Например, для расщепления X для каждой пары координат Y и Z получается новая трехдиагональная система. Решение этих систем является ядром метода и его основной вычислительной частью.

Поскольку уравнения Навье–Стокса являются нелинейными, необходимо также проводить итерации для обновления нелинейных коэффициентов. В текущей реализации проводятся итерации двух типов — глобальные и локальные. Глобальные итерации применяются для целого временно́го шага для всей системы, локальные — для каждого дробного шага, соответствующего одному из направлений.

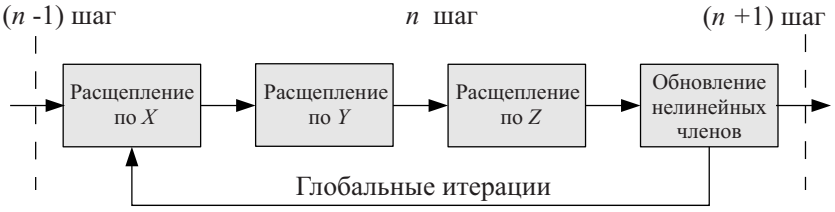

Рис. 1. Целый шаг метода покоординатного расщепления

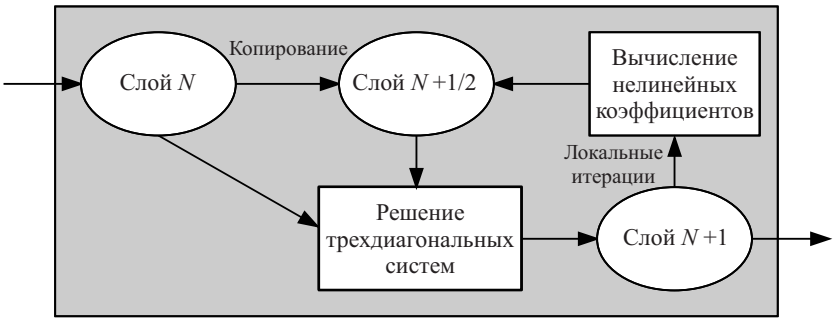

Рис. 2. Дробный шаг расщепления

Общая схема алгоритма приведена на рис. 1. На каждом целом временно́м шаге мы расщепляем нашу систему по трем направлениям и решаем новые системы последовательно, используя глобальные итерации. Число глобальных итераций выбирается так, чтобы ошибка после каждого шага не превышала установленную величину. Внутри каждого дробного шага расщепления соответствующие уравнения решаются при использовании конечно-разностной схемы. На рис. 2 показана модель данных и расчетов для отдельного шага расщепления. Основное вычислительное ядро — решатель наборов трехдиагональных систем. Полученные решения систем используются для обновления нелинейных коэффициентов. После этого проводится несколько локальных итераций для уменьшения численной ошибки.

Подобная схема использовалась впервые для решения уравнений Навье–Стокса в двумерном случае в работе [8]. В дальнейшем развитие этой идеи послужило реализации численного метода для трехмерного случая и разработке динамической системы визуализации в [9]. В настоящей статье мы подробно рассмотрим эффективную параллельную реализацию метода покоординатного расщепления и решения трехдиагональных систем на архитектуре графических процессоров.

5. Обзор архитектуры GPU. Графические процессоры изначально создавались и были предназначены для задач отрисовки графики, активно использующих параллельные вычисления и многочисленные арифметические операции. С течением времени ГПУ эволюционировали в массивно-параллельную архитектуру с большой пропускной способностью памяти и высокой пиковой производительностью вычислений с плавающей точкой. Далее мы приведем небольшой обзор архитектуры ГПУ на примере современной модели Fermi и обсудим ключевые характеристики, а также особенности разработки программ численного моделирования под архитектуру ГПУ.

Архитектура Fermi состоит из иерархического массива процессоров и связанных с ними систем памяти, как показано на рис. 3. Fermi состоит из нескольких потоковых мультипроцессоров (streaming

multiprocessor). Каждый мультипроцессор состоит из 32 CUDA-ядер, которые независимо выполняют операции с плавающей точкой. Например, модель Tesla C2070 имеет 14 потоковых мультипроцессоров, что в конечном итоге дает 448 вычислительных ядер на одно устройство. Параллельные потоки (threads) объединяются в группы по 32; эти группы называются варпами (warps). Варпы исполняются в соответствии с ОКМД-моделью (одна команда — много данных), одновременно работая на всех ядрах одного мультипроцессора. В такой конфигурации одна инструкция может быть исполнена для 32 потоков за один цикл. Потоки внутри одного варпа могут идти по любому пути исполнения кода. Однако поскольку все 32 ядра внутри одного мультипроцессора разделяют выборку инструкции и логику декодирования, за один цикл только одна инструкция может быть направлена 32 нитям одного варпа. Если нити одного варпа исполняют разные пути, то происходит существенное снижение производительности, зависящее от числа уникальных путей исполнения. Такая модель исполнения называется ОКМП (одна команда — много потоков).

Большинство реализаций численных методов моделирования течений интенсивно используют доступную память; таким образом, скорость вычислений сильно зависит от доступа к памяти и ее пропускной способности. В этой ситуации очень важно понимать, как работает система памяти в ГПУ и что необходимо для достижения максимальной пропускной способности.

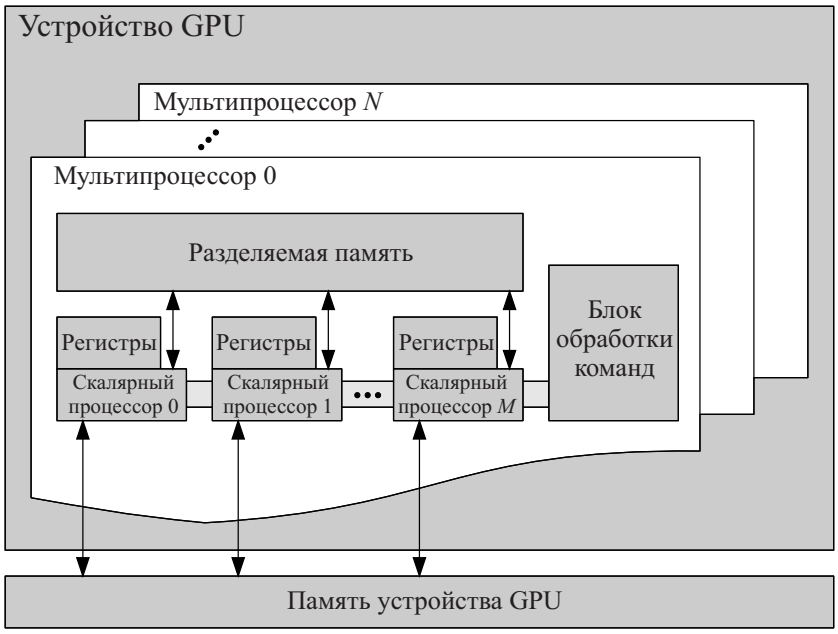

Рис. 3. Архитектура NVIDIA GPU

На устройстве Tesla C2070 пиковая пропускная способность памяти равна 144 ГБ/с. Однако эта скорость может быть достигнута лишь при использовании всех дорожек шины ОЗУ. Для достижения максимальной пропускной способности памяти важно выбирать правильный шаблон доступа. Если потоки варпа считывают информацию из одной области памяти в 128 байт, то эти процессы объединяются в единый запрос; такой процесс называется объединением запросов (coalescing). Поскольку объединение работает на уровне варпов, при неправильном доступе отдельные нити будут создавать дополнительные запросы, при этом будет существенно снижаться пропускная способность памяти. Следовательно, наиболее важной оптимизацией для приложений, интенсивно использующих память, является организация работы потоков таким образом, чтобы потоки внутри одного варпа обращались к последовательным областям памяти в одно и то же время.

6. Детали реализации на CUDA. CUDA — это новая модель программирования, разработанная NVIDIA с целью связать вычислительные мощности своих ГПУ. CUDA является расширением языка программирования Си и позволяет разработчику управлять массивными параллельными вычислениями на ГПУ. Более подробно о программной модели можно узнать в справочнике по программированию на CUDA [1]. В этом разделе мы раскроем некоторые технические детали реализации численного метода на CUDA, а также расскажем о проблемах, с которыми нам пришлось столкнуться по ходу переноса программы на CUDA, и приведем их решение.

В целом программа состоит из трех ядер: трехдиагональный солвер, оценка диссипативной функции и нелинейное обновление. Все массивы данных хранятся в ГПУ с целью снижения количества дорогостоящих транзакций PCI-E между ЦПУ и ГПУ до нуля. Мы остановим внимание на ядре трехдиагонального солвера, который является наиболее затратным и отражает общую идею метода.

Основная вычислительная часть предложенного численного метода решает множество независимых трехдиагональных систем, полученных после применения конечно-разностной дискретизации. Существует несколько алгоритмов, разработанных для решения трехдиагональных систем. Самые общеупотребимые из них — это метод прогонки (sweep) и метод циклической редукции (cyclic reduction). Метод прогонки — это самый быстрый последовательный алгоритм; он состоит из двух шагов: расчет прогоночных коэф-

фициентов и последовательное восстановление вектора решения. С другой стороны, метод циклической редукции можно эффективно распараллелить.

В работе [10] представлено несколько параллельных реализаций трехдиагональных солверов, использующих алгоритм циклической редукции и его гибридные версии. Тем не менее, эти методы имеют известные ограничения, что делает их неприменимыми по отношению к моделированию течений в больших областях. Например, трехдиагональная система должна иметь не более 512 элементов — из-за ограничений на размер разделяемой памяти в существующих ГПУ. Кроме того, методы, основанные на циклической редукции, требуют выполнения большего количества операций и менее эффективны для большого количества относительно малых по размеру систем, чем прогонка.

Общая идея алгоритма для ГПУ состоит в том, чтобы передать каждую из трехдиагональных систем одному независимому потоку и затем решить эти системы одновременно методом прогонки. Количество систем на каждом шаге разбиения равно квадрату размерности сетки. Таким способом мы можем запустить тысячи потоков параллельно на ГПУ, полностью загружая устройство.

Все трехдиагональные системы хранятся прямо на трехмерной сетке данных в линейной памяти ГПУ. Трехмерный массив будет линеаризован следующим образом: $X + Y \dim X + Z \dim X \dim Y$, где $[\dim X, \dim Y, \dim Z]$ — размеры сетки. Таким образом, соседние элементы по направлению X будут лежать рядом в глобальной памяти ГПУ. Заметим, что, при расщеплении по направлению X, нити одного варпа будут решать разные системы и, соответственно, в один момент времени будут обращаться к удаленным местам в памяти со сдвигом $\dim X$. Это приводит к необъединенным доступам, в то время как при разбиении по осям Y и Z чтение и запись могут объединяться для каждого варпа.

Ниже представлен фрагмент программы, демонстрирующий основное вычислительное ядро для ГПУ, которое решает набор трехдиагональных систем. Полный исходный код выложен в свободном доступе на сайте проекта [11].

```
__device__ void solve_tridiagonal(
        FTYPE *a, FTYPE *b, FTYPE *c, FTYPE *d, FTYPE *x,
        int num, int id, int num_seg, int max_n )
{
    get(c,num-1) = 0.0;

    get(c,0) = get(c,0) / get(b,0);
    get(d,0) = get(d,0) / get(b,0);

    // прямой ход прогонки
    for (int i = 1; i < num; i++)
    {
      get(c,i) = get(c,i) / (get(b,i) - get(a,i) * get(c,i-1));
      get(d,i) = (get(d,i) - get(d,i-1) * get(a,i)) /
                 (get(b,i) - get(a,i) * get(c,i-1));
    }

    get(x,num-1) = get(d,num-1);

    // обратный ход прогонки
    for (int i = num-2; i >= 0; i--)
      get(x,i) = get(d,i) - get(c,i) * get(x,i+1);
}
```

Для увеличения пропускной способности памяти мы предоставляем дополнительные шаги переупорядочивания данных и запускаем ядро разбиения по оси Y вместо разбиения по оси X. Переупорядочивание данных позволяет транспонировать нужные массивы данных, используя эффективные алгоритмы транспонирования, пользующиеся преимуществами разделяемой памяти ГПУ. Расщепление по Y имеет оптимальный доступ к памяти с объединением запросов. Эта схема улучшает производительность трехдиагонального солвера по направлению X приблизительно в два раза, включая дополнительные затраты на переупорядочивание данных.

В табл. 1 приведены данные по времени работы расщепления и затраты на транспонирование при использовании 4 локальных итераций. Следует отметить, что транспонирование данных нужно делать всего один раз для нескольких локальных итераций. Таким образом, при увеличении числа итераций влияние

переупорядочивания данных на производительность будет уменьшаться. Экспериментальным путем было установлено, что порядка 10 локальных итераций достаточно, чтобы полностью скрыть дополнительные затраты на транспонирование.

В этом разделе были предложены реализация алгоритма решения большого числа независимых трехдиагональных систем на ГПУ и схема разделения области расчетов для последующей параллельной обработки. В общем случае гибридные алгоритмы, состоящие из прогонки и циклической редукции, могут еще более качественно справляться с большими сетками за счет более эффективного распараллеливания. Это одно из возможных направлений развития алгоритма в дальнейшем.

Таблица 1
Затраты на решение трехдиагональных систем и транспонирование для модельной задачи на сетке 192 в кубе на Tesla C2070

Ядро	Время (с)
Направление Z	5.6
Направление Y	5.6
Направление X (без трансп.)	15.8
Направление X (с трансп.)	5.6
Транспонирование данных	1.8

7. Анализ производительности. В этом разделе мы проведем подробное исследование эффективности предложенного алгоритма для ГПУ. Мы представим сравнение производительности современных многоядерных ЦПУ и ГПУ на примере предложенного трехмерного численного метода покоординатного расщепления. Обе версии — многоядерные ЦПУ и ГПУ — будут тестироваться на одних и тех же конфигурациях и модельной задаче.

На ЦПУ распараллеливание достигается за счет использования директив OpenMP. Оптимальное число потоков для ЦПУ выбиралось экспериментальным путем. В табл. 2 показана зависимость ускорения расчетов на ЦПУ от количества параллельных потоков для 12-ядерного процессора Intel Core i7. Можно сделать вывод, что насыщение подсистемы памяти ЦПУ наступает при использовании 6 потоков и выше, при этом достигается ускорение порядка 4 раз по сравнению с однопоточной реализацией. Использование векторных команд для ЦПУ (SSE) не дает видимого ускорения, так как метод прогонки на ЦПУ ограничен скоростью доступа к оперативной памяти.

Таблица 2
Зависимость скорости расчетов на ЦПУ от числа используемых OpenMP потоков

Число потоков ЦПУ	1	2	4	6	8	10	12
Общее ускорение	1.0	1.9	3.0	3.7	3.2	3.6	4.0

Для изучения производительности мы использовали регулярную декартову сетку размерности $192 \times 192 \times 192$. В наших тестах размеры сетки позволяют моделировать движение газа с числом Рейнольдса до 500. Численные результаты демонстрируют ламинарно-турбулентное течение внутри канала. Кроме того, мы протестировали ЦПУ- и ГПУ-реализации метода с одинарной и двойной точностью.

Таблица 3
Одинарная точность расчетов. Число обработанных систем в секунду

Ядро	Intel Core i7	Tesla C2070	Ускорение ГПУ
Расщепление по X	472	2326	4.9
Расщепление по Y	427	2315	5.4
Расщепление по Z	242	2312	9.5
Обновление нелин. коэф.	644	6314	9.8
Весь процесс	226	1576	7.0

Таблица 4
Двойная точность расчетов. Число обработанных систем в секунду

Ядро	Intel Core i7	Tesla C2070	Ускорение ГПУ
Расщепление по X	314	1400	4.4
Расщепление по Y	288	1386	4.8
Расщепление по Z	187	1391	7.4
Обновление нелин. коэф.	371	3436	9.3
Весь процесс	149	920	6.1

В табл. 3 и 4 представлено сравнение временны́х показателей исполнения отдельных блоков программы на ГПУ и ЦПУ при расчетах с одинарной и двойной точностью соответственно. Результаты приводятся отдельно по трем направлениям расщепления, обновлению нелинейных коэффициентов, а также по общему процессу с учетом транспонирования для ГПУ. Результаты ГПУ представлены для NVIDIA Tesla C2070. Результаты ЦПУ были получены на процессоре Intel Core i7-3930K с частотой 3.2 ГГц с использованием оптимального числа потоков.

ГПУ-реализация выигрывает по производительности у современных многоядерных ЦПУ на порядок по всем этапам расщепления и по общему процессу. Это позволяет выполнять одни и те же вычисления

значительно быстрее на ГПУ, чем на ЦПУ.

Расчеты с двойной точностью выполняются примерно в 2 раза медленнее, чем с одинарной. При тестировании на моделях предыдущего поколения Tesla C1060 были получены похожие соотношения времен для одинарной и двойной точности, хотя пропускная способность вычислений в случае двойной точности в 8 раз медленнее. Таким образом, на Tesla C1060 метод ограничен доступом к памяти. При использовании профилировщика Visual Profiler можно сделать вывод, что скорость работы метода на Tesla C2070 также ограничена пропускной способностью доступа к памяти для каждого этапа расщепления и обновления нелинейных коэффициентов. Результаты также подтверждают исследования влияния двойной точности на производительность в работе [12]. ГПУ имеют бóльшую пропускную способность памяти, чем ЦПУ, поэтому при правильном шаблоне доступа удается достичь на порядок большей производительности.

8. Заключение. В настоящей статье была представлена эффективная ГПУ-реализация метода покоординатного расщепления для решения полной системы уравнений Навье–Стокса в трехмерной области. Этот метод имеет может применяться для прямого моделирования течений. Численный метод был протестирован на модельной задаче и продемонстрировал хорошие результаты по производительности.

Предложенная реализация использует мелкозернистый параллелизм для эффективной утилизации вычислительных ресурсов ГПУ. Учитывая особенности архитектуры и используя специальный доступ к памяти, удается использовать бóльшую часть пропускной способности памяти.

В дальнейшем планируется улучшение численного метода для моделирования течений в областях сложной формы с нетривиальной границей.

Другая, не менее важная область для исследования в будущем — это использование нескольких ГПУ и кластеров из ГПУ. Так как для единичного устройства, будь то ГПУ или обычный процессор, всегда существуют ограничения по объему памяти и производительности, очень важно эффективно распределять вычисления между несколькими устройствами. Реализация и распределение расчетов трехдиагональных матриц между несколькими ГПУ представляют собой дополнительные сложности, схожие с теми, что возникают в распределенных системах и кластерах. Решение этих проблем на одном узле с несколькими ГПУ или даже на кластере с несколькими компьютерами с ГПУ — очень важная область для дальнейшего исследования.

СПИСОК ЛИТЕРАТУРЫ

1. NVIDIA Inc. NVIDIA CUDA programming guide, version 4.0.
2. *Марчук Г.И.* Методы расщепления для решения нестационарных задач // Ж. вычисл. матем. и матем. физ. 1995. **35**, № 6. 843–849.
3. *Ковеня В.М., Слюняев А.Ю.* Алгоритмы расщепления при решении уравнений Навье–Стокса // Ж. вычисл. матем. и матем. физ. 2009. **49**, № 4. 700–714.
4. *Березин С.Б., Пасконов В.М.* Численное исследование вдува вязкого несжимаемого газа в плоский канал на основе уравнений Навье–Стокса // Вычислительные методы и программирование. 2003. **4**, № 1. 5–17.
5. *Moin P., Mahesh K.* Direct numerical simulation: a tool in turbulence research // Anuual Review in Fluid Mech. 1998. **30**. 539–578.
6. *Флетчер К.* Вычислительные методы в динамике жидкостей. Т. 2. М.: Мир, 1991.
7. *Яненко Н.Н.* Метод дробных шагов решения многомерных задач математической физики. Новосибирск: Наука, 1967.
8. *Березин С.Б., Пасконов В.М.* Неклассические решения классической задачи о течении вязкой несжимаемой жидкости в плоском канале // Прикладная математика и информатика. Вып. 17. М.: МАКС Пресс, 2004.
9. *Березин С.Б., Корухова Е.С., Пасконов В.М.* Динамическая система визуализации для многопроцессорных компьютеров с общей памятью и ее применение для численного моделирования турбулентных течений вязких жидкостей // Вестн. Моск. ун–та. Сер. 15: Вычислит. матем. и кибернетика. 2007. № 1. 7–16.
10. *Zhang Y., Cohen J., Owens J.D.* Fast tridiagonal solvers on the GPU // Proc. of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP 2010). New York: ACM, 2010. 127–136.
11. Google Code project (http://code.google.com/p/cmc-fluid-solver/).
12. *Cohen J., Molemaker J.* A fast double precision CFD code using CUDA // Parallel Computational Fluid Dynamics: Recent Advances and Future Directions. Lancaster: DEStech Publications, 2010. 414–429.

Поступила в редакцию
06.06.2012