DOI: 10.51790/2712-9942-2024-5-4-14

ПРИМЕНЕНИЕ ГИБРИДНЫХ ВЫЧИСЛИТЕЛЕЙ ДЛЯ МОДЕЛИРОВАНИЯ ЛАВОВОГО ПОТОКА

И. А. Цепелев[а], **А. И. Короткий**[б]
Институт математики и механики им. Н.Н. Красовского УрО РАН, г. Екатеринбург, Российская Федерация
[а] *ORCID: https://orcid.org/0000-0002-8236-9834, ✍ tsepelev@imm.uran.ru*
[б] *ORCID: https://orcid.org/0009-0000-4539-7169, korotkii@imm.uran.ru*

Аннотация: в данной работе рассматривается усредненная по глубине модель движения вязкой жидкости, предназначенная для компьютерного моделирования распространения вулканических лав. Описываются математические свойства модели, методы ее аппроксимации и алгоритмы численного решения, ориентированные на применение современных гибридных вычислительных кластеров. Математическая модель представляет собой систему уравнений гиперболического типа, которая отражает законы сохранения массы и импульса в приближении тонкого слоя. Данные рельефа, реологические свойства жидкости и экструзия вещества из кратера задаются как неоднородные члены в системе. Предлагаемая математическая модель имеет преимущество вычислительной эффективности из-за ее двумерности и включения высоты свободной поверхности в качестве переменной в основные уравнения. В работе обсуждаются параллельные компьютерные реализации рассматриваемой модели на основе OpenFOAM (MPI), OpenMP и OpenACC. Компьютерные коды реализованы на вычислительных кластерах с общей и распределенной памятью на CPU и GPU под управлением ОС Linux. Проведены верификация кодов на аналитическом решении задачи и профилирование кодов для многоядерных CPU с общей памятью и GPU.

Ключевые слова: параллельное программирование, моделирование, многоядерные процессы, графические сопроцессоры, эффективность программ, лавовые потоки.

Благодарности: при проведении работ был использован суперкомпьютер «Уран» ИММ УрО РАН, г. Екатеринбург.

Для цитирования: Цепелев И. А., Короткий А. И. Применение гибридных вычислителей для моделирования лавового потока. *Успехи кибернетики*. 2024;5(4):103–109. DOI: 10.51790/2712-9942-2024-5-4-14.

Поступила в редакцию: 09.10.2024. *В окончательном варианте: 10.11.2024.*

APPLICATION OF HYBRID COMPUTERS TO SIMULATE THE LAVA FLOW

I. A. Tsepelev[a], **A. I. Korotkii**[b]
N.N. Krasovskii Institute of Mathematics and Mechanics, Urals Branch, Russian Academy of Sciences, Yekaterinburg, Russian Federation
[a] *ORCID: https://orcid.org/0000-0002-8236-9834, ✍ tsepelev@imm.uran.ru*
[b] *ORCID: https://orcid.org/0009-0000-4539-7169, korotkii@imm.uran.ru*

Abstract: this study discusses a depth-averaged model of viscous flow motion used in computer simulation of volcanic lava propagation. The mathematical properties of the model, its approximation methods, and numerical solution algorithms fine-tuned for modern hybrid computing clusters are presented. The mathematical model is a system of hyperbolic equations that simulate the laws of conservation of mass and momentum in the thin layer approximation. The terrain and rheological properties of the liquid and substance extrusion rate from the crater are specified as inhomogeneous members of the system. The proposed mathematical model offers high computational efficiency due to its two-dimensionality and the inclusion of free surface height as a variable in the equations. The paper also discusses parallel computer implementations of the model with OpenFOAM (MPI), OpenMP, and OpenACC. The codes run on CPU/GPU clusters with shared and distributed memory and Linux OS. The codes were verified using the analytical solution of the problem. We also profiled the codes for multi-core CPUs with shared memory.

Keywords: parallel computing, simulation, multi-core processes, graphics coprocessors, high performance codes, lava flows.

Acknowledgements: computations were performed on the Uran supercomputer, N.N. Krasovskii Institute of Mathematics and Mechanics, Urals Branch, Russian Academy of Sciences.

Cite this article: Tsepelev I. A., Korotkii A. I. Application of Hybrid Computers to Simulate the Lava Flow. *Russian Journal of Cybernetics*. 2024;5(4):103–109. DOI: 10.51790/2712-9942-2024-5-4-14.

Original article submitted: 09.10.2024. *Revision submitted: 10.11.2024.*

Постановка задачи

Большинство геофизических потоков (например, снежные лавины, пирокластические потоки, селевые потоки) можно классифицировать как движение неньютоновской жидкости. Изучение поведения этих потоков не только очень интересно с чисто научной точки зрения, но и чрезвычайно важно на практике, учитывая сильное воздействие, которое они могут оказать на население и прилегающую инфраструктуру. Поэтому надежная модель поведения потока во времени необходима для того, чтобы иметь возможность быстро прогнозировать эволюцию таких природных явлений с целью смягчения последствий и снижения риска.

Численные методы и компьютерное моделирование могут дополнять лабораторные эксперименты благодаря предоставляемой ими возможности запускать тестовые сценарии на ЭВМ, обеспечивая безопасный испытательный стенд для экспериментов. Для организации такой работы необходимо разработать (выбрать) надежный численный метод, который позволяет дискретизировать соответствующие уравнения и реализовать его в виде компьютерных программ с учетом использования параллельных методов обработки данных. Далее, провести проверку программного кода, когда известно аналитическое решение уравнений. Проверенный код можно использовать для запуска нескольких тестовых примеров с различными параметрами в модели и сравнения результатов с полевыми измерениями или лабораторными экспериментами.

Рассмотрим процесс вытекания вязкой несжимаемой жидкости из кратера вулкана. Опишем математическую модель рассматриваемого движения жидкости. В качестве основных уравнений состояния жидкости примем двумерную усредненную по глубине модель движения вязкой жидкости. В модельной области $\Omega = (0,L_1) \times (0,L_2)$, $\Omega = \Omega_1 \cup \Omega_2$, (см. рис. 1) движение такой вязкой жидкости на промежутке времени $t \in [0,\vartheta]$, где $t = \theta$ — конечный момент времени наблюдения за процессом, представляется системой дифференциальных уравнений с частными производными (PDE) [1]:

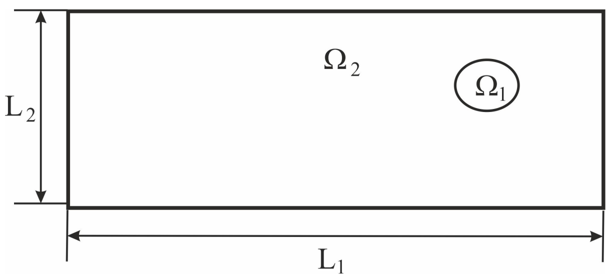


Рис. 1. *Модельная область*

$$\frac{\partial h}{\partial t} + \nabla \cdot (h\boldsymbol{u}) = w_{in}, \quad \boldsymbol{x} \in \Omega, \tag{1}$$

$$\frac{\partial (h\boldsymbol{u})}{\partial t} + \nabla \cdot \left(h\boldsymbol{u}\boldsymbol{u}^T\right) = \tag{2}$$

$$-\frac{1}{2}\nabla\left(gh^2\right) - gh\nabla H - \gamma\boldsymbol{u}, \quad \boldsymbol{x} \in \Omega_2,$$

$$h(0,\boldsymbol{x}) = u(0,\boldsymbol{x}) = v(0,\boldsymbol{x}) = 0, \quad \boldsymbol{x} \in \Omega, \tag{3}$$

где $\boldsymbol{x} = (x,y)$ – пространственная переменная, $h = h(t;\boldsymbol{x})$ – высота столба жидкости, измеренная от уровня поверхность местности $H = H(\boldsymbol{x})$ до поверхности взаимодействия жидкости с окружающей средой, $\boldsymbol{u}(t;x,y) = (u,v)$ — усредненная скорость жидкости, $\boldsymbol{U}(t;x,y,z)$ – скорость потока жидкости $\boldsymbol{u}(t;x,y) = h^{-1}(t;x,y) \int\limits_{H(x,y)}^{H(x,y)+h(x,y)} \boldsymbol{U}(t;x,y,z)dz$, $w_{in} = w_{in}(t;\boldsymbol{x}) > 0$, $\boldsymbol{x} \in \Omega_1$, $w_{in} = 0$, $\boldsymbol{x} \in \Omega_2$ — магнитуда скорости экструзии жидкости и $\gamma = \gamma(t;\boldsymbol{x})$ — коэффициент сопротивления, в котором учитываются зависимости от реологии жидкости, свойства самого потока и взаимодействие с поверхностью H, $g = 9.81$ — ускорение свободного падения.

Ключевым аспектом, определяющим поведение геофизических потоков, является их реология — соотношение между напряжением сдвига и скоростью деформации в жидкостях. Жидкость Бингама является простейшей аппроксимацией поведения пороговых жидкостей: такая жидкость остается в покое, пока приложенные напряжения сдвига, вызванные гравитационными силами, меньше, чем предел текучести. Закон Гершеля–Балкли является обобщением закона Бингама и степенного закона. В этом случае поток может характеризоваться фактической вязкостью $\mu_{eff} = \mu\dot{\varepsilon}^{n-1} + \tau_0\dot{\varepsilon}^{-1}$, $\dot{\varepsilon} = h^{-1}\|\boldsymbol{u}\|$,

$0 < n \leq 1$. Здесь μ (Pa s^n) — текучесть жидкости, определяемая химическими свойствами жидкости, и τ_0 (Pa) — предел текучести потока. Варьируя параметры, можно моделировать ньютоновскую, бингамовскую и жидкость Гершеля–Балкли [2], полагая

$$\gamma \boldsymbol{u} = c_n \rho^{-1} h^{-1} \boldsymbol{u} \mu_{eff}, \quad c_n = ((1+2n)/n)^n.$$

Программный модуль OpenFOAM

Для решения PDE (1), (2) явным или неявным консервативным методом конечных объемов используются соответствующие модули OpenFOAM [3]. С учетом того, что необходимо гарантировать ограниченность вариации решения задачи, применяются TVD (Total Variation Diminishing) схемы второго порядка точности по пространству. Коды тестировались на схемах аппроксимации градиента minmod и superBee [4]. Потери массы при этом не превосходили 3% на всем отрезке времени моделирования. Пакет OpenFOAM — это библиотека языка C++, обладает преимуществами, присущими объектно-ориентированному программированию. Пользователь в программных кодах реализует только алгоритм решения (1)–(3). Работа по дискретизации дифференциальных операторов, получение систем линейных алгебраических уравнений и их решение реализованы в пакете. OpenFOAM предоставляет широкий спектр инструментов для создания (импорта) и преобразования сетки. Для отладки программных кодов проведено сравнение с аналитическим решением задачи [5] и с результатами моделирования в пакете Fluent эволюции двухфазной несжимаемой жидкости в поле силы тяжести. Подробно тестовые эксперименты описаны в [6].

Преимущества: минимальные затраты на создание и сопровождение программных кодов. Аппроксимация уравнений и решатели линейных систем для разреженных матриц разнообразны. Переносимость на вычислитель произвольной архитектуры. Количество процессов не ограничено одним узлом вычислителя. Возможность исполнять последовательную и параллельную версии без модификации кодов.

Недостатки: достаточная квалификация разработчика в языке C++, возможностях пакета OpenFOAM. Реализация только под ОС Linux. Параллельность достигается только средствами MPI [7]. Повысить производительность за счет применения OpenMP и OpenACC затратно для разработчика.

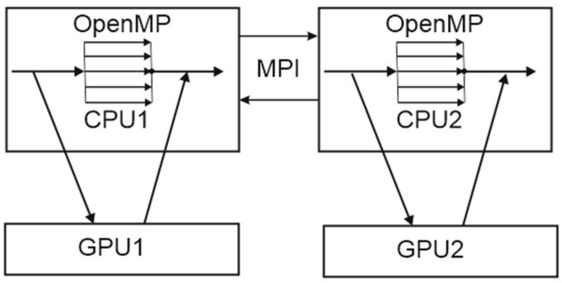


Рис. 2. *Схема организации гибридного вычислительного кластера, узел которого основан на многоядерном CPU с общей оперативной памятью и графическим сопроцессором. Такие узлы связаны в единый кластер с помощью высокоскоростных Интернет-каналов*

Программный модуль OpenMP

Модель (1)–(3) по своей сути параллельна. Это упрощает логику алгоритма в кодах, а время расчетов может быть уменьшено почти пропорционально количеству доступных ядер. В архитектурах с общей памятью все ядра обращаются к одной и той же памяти, даже одновременно. В отличие от MPI, каждый блок имеет глобальное представление памяти, т.е. адресное пространство, общее для всех ядер. Программисту не нужно беспокоиться о передаче данных между различными блоками обработки. В системах с общей памятью несоответствие между скоростью процессора и скоростью обмена с памятью всегда было основным узким местом для количества операций, которые процесс может выполнить за заданное время, поскольку медленная подкачка данных из оперативной памяти не обеспечит процесс нужной информацией. Главным образом, по этой причине при написании параллельного кода важно ограничивать количество обращений к памяти для поддержания высокой производительности.

Основная программа	HEqn_omp.H - Решатель уравнения (1) явной схемой

```
//Цикл по времени
while (currentTime<finalTime)
{
// Вычисление шага по времени tau
  #include "setTimeStep.H"
  currentTime += tau;
  currentInterval += tau;
  if (currentInterval>writeInterval)
  {
#include "writeData.H"
  }
// Вычисление правых частей в (1), (2)
  #include "update_fields.H"
// Решатели уравнений (1), (2)
  #include "UEqn_omp.H"
  #include "HEqn_omp.H"
}
```

```
#pragma omp parallel for
 for (int i = 0; i < _Nx*_Ny; ++i) {
// Реализация уравнения (1)
       double advX = convX(h,  Ux,  i);
       double advY = convY(h,  Uy,  i);
       h_new[i] = h[i] + tau * (-advX - advY + wIn[i]);

#pragma omp parallel for
  for (int i = 0; i < _Nx*_Ny; ++i) {
    h[i] = h_new[i];
   }
```

Рис. 3. *Фрагменты программы OpenMP*

OpenMP использует многопоточную модель (рис. 2). В этой модели процесс изначально состоит из одного потока, называемого начальным потоком. OpenMP позволяет указать некоторые области кода, которые могут выполняться параллельно. Когда поток сталкивается с одним из этих регионов, он многократно разветвляется и образует группу потоков, которые выполняются одновременно. Поток, породивший команду, называется главным потоком команды. Разветвленные потоки снова присоединяются к главному потоку после завершения параллельных операций. Для распараллеливания нужно сосредоточиться только на тех частях кода, которые больше влияют на производительность. OpenMP [9] скрывает низкоуровневые детали реализации (см. рис. 3), позволяя использовать один и тот же код для большего количества платформ, можно скомпилировать один и тот же исходный код для параллельного или последовательного создания (в этом случае компилятор игнорирует OpenMP директивы) версии программы. Для аппроксимации операторов конвективного переноса в (1), (2) применялась регуляризированная схема А.А. Самарского [8].

Программный модуль OpenACC

Вычисления на графическом процессоре предоставляют ученым расширенную поддержку для параллельного выполнения симуляций, касающихся решений систем PDE. Графические процессоры (видеокарты) особенно подходят для быстрого выполнения параллельных программ, а технологии Cuda и OpenACC [10] предоставляют способы доступа к возможностям графического сопроцессора. В структуре программный модуль для OpenACC представляет собой модуль OpenMP, который расположен внутри секции *acc_data_copy* (см. рис. 4). Данная секция до начала цикла по времени загружает необходимые данные на GPU, и в цикле по времени нет обмена информацией между центральным процессором и графической картой (за исключением переменных для сохранения на диск). Далее, секция *acc_parallel_loop_independent* сообщает компилятору, что следующий за ней цикл обрабатывается на GPU параллельно, и компилятор самостоятельно распределяет нагрузки между ядрами GPU.

Для отладки программных кодов в OpenMP и OpenACC проведено сравнение с результатами моделирования аналогичного теста в OpenFOAM.

Преимущества: реализация под любую ОС. Параллельность достигается средствами OpenMP(ACC). Переносимость на вычислитель произвольной архитектуры и тип графической карты. Последовательная и параллельная версии реализуются без модификации кодов.

Недостатки: разработчик сам аппроксимирует дифференциальные операторы и разрабатывает решатели систем линейных алгебраических уравнений для разреженных матриц. Ограниченность количества ядер для MP и возможность использовать одну графическую карту для ACC. Не все компиляторы поддерживают OpenACC. В данной реализации использован компилятор PGI v.16.5. Использо-

```
Основная программа
// Загрузка данных из памяти CPU в GPU
#pragma acc data copy (h, Ux, Uy, H, wIn)
{
//Основной цикл по времени
while (currentTime<finalTime)
{
// Вычисление шага по времени tau
 #include "setTimeStep.H"
 currentTime += tau;
 currentInterval += tau;
 if (currentInterval>writeInterval)
  {
// Выгрузка днных из GPU в память CPU
#pragma acc data update(СПИСОК ДАННЫХ
ДЛЯ ЗАПИСИ НА ДИСК)
#include "writeData.H"
  }
// Вычисление правых частей для (1), (2)
 #include "update_fields.H"
// Решатели уравнений (1)-(5)
 #include "UEqn_acc.H"
 #include "HEqn_acc.H"
} // while
} // Данные пересылаются на CPU, память GPU
освобождается
```

```
HEqn_omp.H - Решатель уравнения (1) явной схемой
 #pragma acc data present (h, Ux, Uy)
 {
 #pragma acc parallel loop independent
 for (int i = 0; i < _Nx*_Ny; ++i) {
      double advX = _X(h, Ux, i);
      double advY = _Y(h, Uy, i);
       h_new[i] = h[i] + tau*(-advX - advY + wIn[i]);
 }

 #pragma acc parallel loop independent
   for (int i = 0; i < _Nx*_Ny; ++i) {
     h[i] = h_new[i];
   }
 } // acc data present
```

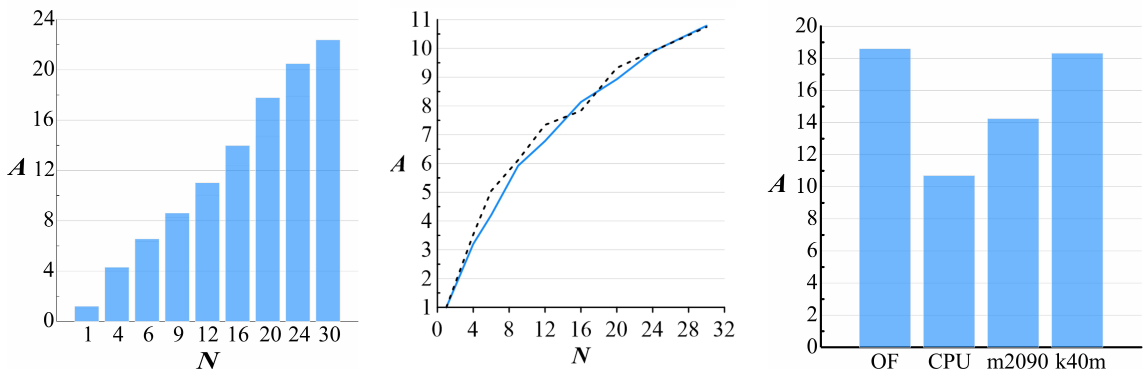
Рис. 4. *Фрагменты программы OpenACC*

вание пакетов BLAS (cuBLAS) и CUDA (OpenCL) может существенно повысить производительность кодов, но при этом также усложнить разработку и адаптацию кодов под реальную задачу.

Оценка эффективности кода

Эффективная вычислительная программа обеспечивает точные результаты при малых вычислительных затратах. Это особенно важно для кодов, используемых, когда необходимо получать результаты компьютерного моделирования в реальном времени и/или провести серии виртуальных экспериментов для создания прогнозов развития процесса и своевременного реагирования на быстро меняющиеся условия. Сравнение нескольких кодов в рамках одного бенчмарка требует общего показателя эффективности кода, независимого от возможностей вычислителя, используемого для тестирования. В данном исследовании все тесты проводились на одном вычислителе, который находился в общем доступе, и сравнивалось усредненное вычислительное время из 5 испытаний. В тестах OpenFOAM для сравнения с OpenMP (OpenACC) использовался явный метод решения уравнений (1)–(3). На рис. 5а демонстрируется ускорение $A = time(1core)/time(Ncore)$ расчетов в пакете OpenFOAM версии 7 (OF) для явной схемы аппроксимации уравнений (1), (2) в зависимости от количества используемых ядер CPU. Размерность задачи в этом тесте 600000 ячеек. Общее количество ядер CPU в расчетах ограничено $N = 36$, это число — максимальное количество ядер на одном узле ЭВМ, которые имеют общую оперативную память. При дальнейшем увеличении ресурсов расчет осуществляется на нескольких узлах ЭВМ, что значительно снижает эффективность расчетов, т.к. учитываются ресурсы на межпроцессорные обмены. На рис. 5b демонстрируется ускорение расчетов в решателе на основе OpenMP. Тесты проведены в условиях теста OF. В системах с общей памятью несоответствие между скоростью процессора и скоростью памяти всегда было основным узким местом для количества операций, которые процесс может выполнить за заданное время, поскольку медленная подкачка данных из оперативной памяти не может обеспечить процессор нужной информацией. Расчеты проводились на gcc(64) версии 8.3 на двух 18-ти ядерных процессорах Intel(R) Xeon(R) Gold 6254, RAM 64 GB. Для сравнения проведены расчеты на Intel Xeon CPU E5-2697. Для него время расчетов было больше примерно на 30%, что соответствует разнице в тактовой частоте процессора, но ускорение при этом практически не изменилось. Можно сделать предположение, что производительность можно увеличить, если более гибко оптимизировать загрузку кеш памяти процессора, использовав, например, пакет BLAS. На рис. 5c демонстрируются ускорения расчетов OF для неявной схемы аппроксимации уравнений (1), (2), OpenMP для $N = 32$

ядер CPU и модуля OpenACC, реализованного на графических сопроцессорах NVidia m2090 и NVidia k40m.



(a) (b) (c)

Рис. 5. *(a) Ускорение расчетов в пакете OpenFOAM-7. (b) Ускорение расчетов для модуля OpenMP. Сплошная линия — Intel Xeon E5-2697; пунктир — Intel Xeon Gold 6254; (c) Ускорение расчетов для OpenACC. Тестировались реализации OpenFOAM и OpenMP (32 ядра — Intel Xeon Gold 6254), NVidia m2090 и NVidia k40m*

Заключение

Параллельные вычисления представляют собой эффективное решение для организации вычислительных экспериментов сложных геофизических процессов, таких как потоки лавы, на современных вычислительных кластерах. Двумерные модели позволяют качественно моделировать такие процессы за приемлемое время, хотя и не отражают всех факторов, которые влияют на динамику этих процессов. Данные модели по своей сути параллельны, поэтому логика их реализации на компьютерах проста, продолжительность численного эксперимента может быть уменьшена пропорционально количеству доступных ядер. Использование неявных методов аппроксимации уравнений модели позволяет организовать надежный решатель проблемы, но время эксперимента и ресурсы системы при этом увеличиваются. Явные решатели менее требовательны к ресурсам, но результаты их работы требуют подтверждения на адекватность получаемого решения. При этом снижается надежность эксперимента.

В данном исследовании основное внимание уделялось простоте реализации алгоритмов численного решения в программных кодах для упрощения модификации (адаптации) кодов для решения конкретной прикладной задачи. Использование пакетов BLAS (cuBLAS) и CUDA (OpenCL) может существенно повысить эффективность реализации кодов, но при этом также усложнить разработку и адаптацию кодов под реальную задачу. Численные эксперименты демонстрируют соблюдение основных физических законов и приемлемую эффективность реализации кодов на ЭВМ.

ЛИТЕРАТУРА

1. Costa A., Macedonio G. Numerical Simulation of Lava Flows Based on Depth-Averaged Equations. *Geophysical Research Letters*. 2005;32:L05304. DOI: doi.org/10.1029/2004GL021817.
2. Ferrari S., Saleri F. A New Two-Dimensional Shallow Water Model Including Pressure Effects and Slow Varying Bottom Topography. *ESAIM Math. Modell. Numer. Anal.* 2004;38:211–234. DOI: doi.org/10.1051/m2an:2004010.
3. Jasak H. OpenFOAM: Open Source CFD in Research and Industry. *International Journal of Naval Architecture and Ocean Engineering* 2009;1(2):89-94. DOI: doi.org/10.2478/IJNAOE-2013-0011.
4. LeVeque R. J. *Finite Volume Methods for Hyperbolic Problems*. Cambridge University Press; 2002. 580 p.
5. Huppert H. E. The Propagation of Two-Dimensional and Axisymmetric Viscous Gravity Currents Over a Rigid Horizontal Surface. *J. of Fluid Mechanics*. 1982;121:43–58. DOI: 10.1017/S0022112082001797.
6. Короткий И. А., Цепелев И. А. Численное моделирование извержения вулкана Этна с применением усредненной по глубине модели потока лавы. *Выч. механика сплошных сред*. 2024;17(3):362–374. DOI: 10.7242/1999-6691/2024.17.3.30.

7. Snir M., Otto S., Huss-Lederman S., Walker D., Dongarra J. *MPI: The Complete Reference*. Cambridge: The MIT Press; 1995. 350 p.
8. Вабищевич П. Н., Первичко В. А., Самарский А. А., Чуданов В. В. Нелинейные регуляризованные разностные схемы для многомерного уравнения переноса. *Ж. вычисл. матем. и матем. физ.* 2000;40(6):900–907.
9. Chapman B., Jost G., van der Pas R. *Using OpenMP Portable Shared Memory Parallel Programming*. Cambridge: The MIT Press; 2007. 384 p.
10. Farber R. *Parallel Programming with OpenACC (1st. ed.)*. Cambridge: Morgan Kaufmann Publishers; 2016. 296 p.