УДК 519.63

Эффективность параллельных реализаций классических итерационных методов решения уравнения Пуассона в цилиндрических координатах

Логинов В.В., Пескова Е.Е.

Национальный исследовательский Мордовский государственный университет

Аннотация: В статье представлен сравнительный анализ эффективности параллельных реализаций классических итерационных методов (Якоби, Гаусса-Зейделя, SOR) для решения уравнения Пуассона в цилиндрической системе координат. В основу исследования положены методы ускорения вычислений с использованием технологии CUDA и библиотеки PyCUDA.

Ключевые слова: уравнение Пуассона, цилиндрическая система координат, GPU-ускорение, параллельные вычисления, итерационные методы, CUDA

1. Введение

Уравнение Пуассона является фундаментальным в математической физике и широко применяется для описания потенциалов, распределения температур, давления и других величин в различных областях науки и техники. Особое значение оно приобретает при моделировании процессов в цилиндрической системе координат (r, ϕ, z), которые часто используются для описания задач с геометрией цилиндрической формы [1].

Общее уравнение Пуассона в цилиндрической системе координат имеет вид:

$$\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial u}{\partial r}\right) + \frac{1}{r^2}\frac{\partial^2 u}{\partial \phi^2} + \frac{\partial^2 u}{\partial z^2} = f(r, \phi, z), \tag{1}$$

где $u(r, \phi, z)$ — искомая функция, $f(r, \phi, z)$ — заданная функция источника.

Численное решение этого уравнения требует дискретизации области и применения эффективных итерационных методов для решения возникающих систем уравнений. Учитывая большой объём вычислений при трёхмерных моделях, становится необходимым использовать высокопроизводительные вычислительные платформы, такие как графические ускорители.

2. Обзор методов решения уравнения Пуассона

Метод Якоби – это один из самых простых в реализации итерационных методов. Он основывается на том, что значение функции в каждой точке обновляется с использованием значений её соседей с предыдущей итерации. Для этого метода легко применимы технологии параллельных вычислений, так как каждый узел обновляется независимо от других. Однако нужно учитывать, что у метода довольно медленная сходимость, особенно на сетках большой размерности, что ограничивает его эффективность при решении сложных задач.

Введем обозначения: n – номер итерации, (i, j, k) – индексы пространственной ячейки по координатам r, ϕ и z, соответственно; h_r, h_ϕ и h_z – шаги интегрирования

по координатам r, ϕ и z, соответственно.

Итерационная формула для метода Якоби:

$$u_{(i,j,k)}^{(n+1)} = \left(\frac{u_{(i+1,j,k)}^{(n)} + u_{(i-1,j,k)}^{(n)}}{h_r^2} + \frac{u_{(i+1,j,k)}^{(n)} - u_{(i-1,j,k)}^{(n)}}{2r_i h_r^2} + \frac{u_{(i,j+1,k)}^{(n)} + u_{(i,j-1,k)}^{(n)}}{r_i^2 h_\phi^2} + \right.$$
$$\left. + \frac{u_{(i,j,k+1)}^{(n)} + u_{(i,j,k-1)}^{(n)}}{h_z^2} - f_{(i,j,k)} \right) \Big/ \left(\frac{2}{h_r^2} + \frac{2}{r_i^2 h_\phi^2} + \frac{2}{h_z^2} \right). \quad (2)$$

Метод Гаусса-Зейделя улучшает сходимость решения, поскольку на каждом шаге итерации используются уже обновлённые значения переменных, что позволяет ускорить процесс сходимости по сравнению с методом Якоби. Однако, это приводит к трудностям при параллельной реализации, поскольку обновление значений в каждой точке зависит от значений, вычисленных в других точках, что создаёт последовательные зависимости и нарушает независимость вычислений, необходимую для эффективного параллелизма.

Итерационная формула для метода Гаусса–Зейделя:

$$u_{(i,j,k)}^{(n+1)} = \left(\frac{u_{(i+1,j,k)}^{(n)} + u_{(i-1,j,k)}^{(n+1)}}{h_r^2} + \frac{u_{(i+1,j,k)}^{(n)} - u_{(i-1,j,k)}^{(n+1)}}{2r_i h_r^2} + \frac{u_{(i,j+1,k)}^{(n)} + u_{(i,j-1,k)}^{(n+1)}}{r_i^2 h_\phi^2} + \right.$$
$$\left. + \frac{u_{(i,j,k+1)}^{(n)} + u_{(i,j,k-1)}^{(n+1)}}{h_z^2} - f_{(i,j,k)} \right) \Big/ \left(\frac{2}{h_r^2} + \frac{2}{r_i^2 h_\phi^2} + \frac{2}{h_z^2} \right). \quad (3)$$

Метод SOR – это улучшенная версия метода Гаусса–Зейделя. В нём вводится дополнительный параметр релаксации ω, который регулирует скорость сходимости. При правильном выборе ω сходимость ускоряется. Параллельная реализация метода SOR сталкивается с такими же трудностями, как и метод Гаусса–Зейделя, из-за зависимостей между значениями, которые обновляются последовательно. Несмотря на это, метод остаётся мощным инструментом для решения больших задач, но требует дополнительной настройки для эффективной работы на параллельных системах.

Итерационная формула для метода SOR:

$$u_{(i,j,k)}^{(n+1)} = (1-\omega) u_{(i,j,k)}^{(n)} + \omega \left(\frac{u_{(i+1,j,k)}^{(n)} + u_{(i-1,j,k)}^{(n+1)}}{h_r^2} + \frac{u_{(i+1,j,k)}^{(n)} - u_{(i-1,j,k)}^{(n+1)}}{2r_i h_r^2} + \frac{u_{(i,j+1,k)}^{(n)} + u_{(i,j-1,k)}^{(n+1)}}{r_i^2 h_\phi^2} + \right.$$
$$\left. + \frac{u_{(i,j,k+1)}^{(n)} + u_{(i,j,k-1)}^{(n+1)}}{h_z^2} - f_{(i,j,k)} \right) \Big/ \left(\frac{2}{h_r^2} + \frac{2}{r_i^2 h_\phi^2} + \frac{2}{h_z^2} \right). \quad (4)$$

где ω – параметр релаксации, который регулирует скорость сходимости. Значения параметра выбираются в диапазоне от 1 до 2.

3. Результаты вычислительных экспериментов

Рассмотренные методы были реализованы с использованием библиотеки Numba [2] для вычислений на CPU. На рис. 1–2 представлены результаты сравнения описанных методов. Из проведённых экспериментов видно, что метод SOR (метод последовательной релаксации) имеет значительное преимущество перед методом Якоби и методом Гаусса-Зейделя как по скорости сходимости, так и по времени выполнения.

В частности, метод SOR показал пятикратное улучшение по сравнению с методом Якоби и трёхкратное по сравнению с методом Гаусса-Зейделя. Однако, несмотря на эти преимущества, при увеличении размерности сетки время выполнения растёт значительно, что требует оптимизации методов.

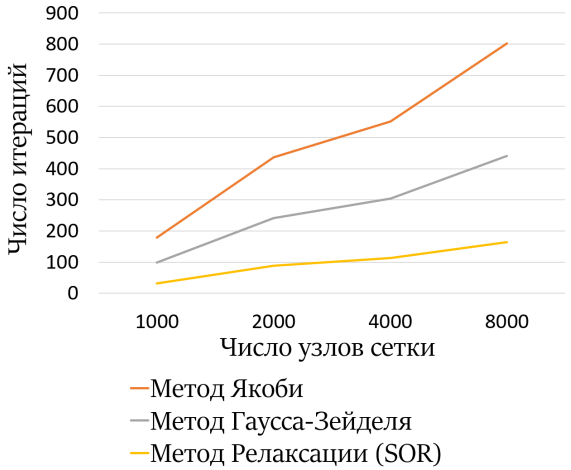

Рис. 1. Сравнение скорости сходимости.

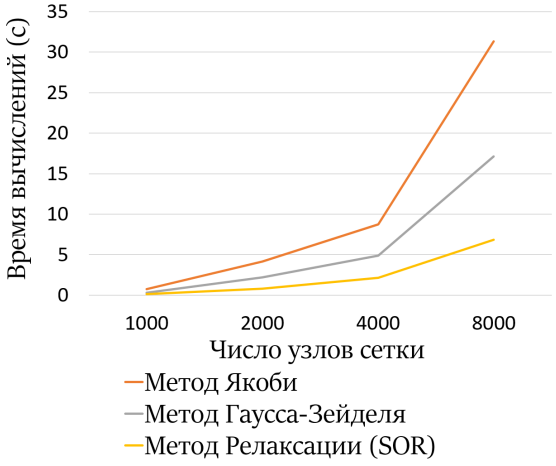

Рис. 2. Сравнение времени выполнения.

Использование GPU позволяет эффективно масштабировать задачу и сокращать время вычислений за счет массового параллелизма. Однако, как упоминалось ранее, некоторые методы не поддерживают параллельную реализацию. Одним из решений этой проблемы является красно-чёрное разбиение области [3], при котором сетка делится на красные и чёрные узлы, а вычисления выполняются поочерёдно. Однако при использовании на GPU такой подход даёт улучшения лишь в пределах погрешности из-за особенностей реализации параллельных вычислений на CUDA.

На рис. 3 представлено сравнение параллельного метода Якоби между CPU и GPU с использованием библиотеки PyCUDA [4]. Реализация параллельного метода на видеокарте дает 4 кратное ускорение при небольшом числе узлов и доходит до 10 кратного ускорения при увеличении числа узлов по сравнению с реализации параллельных вычислений на центральном процессоре.

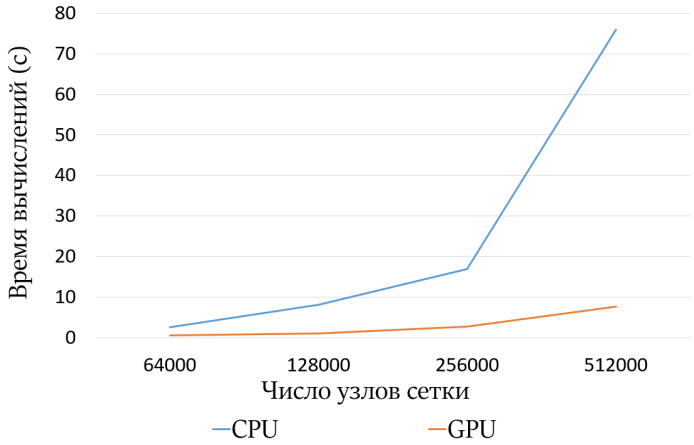

Рис. 3. Сравнение времени выполнения программы на CPU и GPU.

Литература

1. Peskova E. E., Snytnikov V. N. The Influence of Laser Radiation on the Laminar Flow of a Chemically Active Gas-Dust Medium in a Narrow Circular Tube // Theoretical Foundations of Chemical Engineering. 2024. Vol. 58, no. 3. P. 517-525.

2. Numba: A Just-In-Time Compiler for Python [Электронный ресурс]. URL: https://numba.pydata.org/ (дата обращения: 10.07.2025).

3. Ортега Дж. Введение в параллельные и векторные методы решения линейных систем / Пер. с англ. Москва: Мир, 1991. 368 с.

4. PyCUDA: Python interface to CUDA [Электронный ресурс]. URL: https://documen.tician.de/pycuda/ (дата обращения: 10.07.2025).

MSC 65L20

The efficiency of parallel implementations of classical iterative methods for solving the Poisson equation in cylindrical coordinates.

V.V. Loginov, E.E. Peskova

National Research Mordovia State University

Abstract: The article presents a comparative analysis of the efficiency of parallel implementations of classical iterative methods (Jacobi, Gauss-Seidel, SOR) for solving the Poisson equation in cylindrical coordinates on graphics processing units (GPUs). The study is based on computational acceleration techniques using CUDA technology and the PyCUDA library.

Keywords: Poisson equation, cylindrical coordinates, GPU acceleration, parallel computing, iterative methods, CUDA.

References

1. Peskova E. E., Snytnikov V. N. The Influence of Laser Radiation on the Laminar Flow of a Chemically Active Gas-Dust Medium in a Narrow Circular Tube // Theoretical Foundations of Chemical Engineering. 2024. Vol. 58, no. 3. P. 517-525.
2. Numba: A Just-In-Time Compiler for Python [Online]. Available: https://numba.pydata.org/ (accessed: 10.07.2025).
3. Ortega J. Vvedenie v parallelnye i vektornye metody resheniya lineinykh sistem [Introduction to Parallel and Vector Methods for Solving Linear Systems]. Moscow: Mir, 1991. 368 p. (in Russian)
4. PyCUDA: Python interface to CUDA [Online]. Available: https://documen.tician.de/pycuda/ (accessed: 10.07.2025).