УДК 004.42, 004.051
DOI: 10.15827/0236-235X.141.130-143
Дата подачи статьи: 30.09.22, после доработки: 10.11.22
2023. Т. 36. № 1. С. 130–143

Векторизация трехмерного метода погруженных границ для повышения эффективности расчетов на микропроцессорах Intel

А.А. Рыбаков [1], *к.ф.-м.н., ведущий научный сотрудник, rybakov@jscc.ru*
А.О. Мещеряков [1], *младший научный сотрудник, alex2501@jscc.ru*

[1] *Межведомственный суперкомпьютерный центр РАН (МСЦ РАН) – филиал ФНЦ НИИСИ РАН, г. Москва, 119991, Россия*

Работа посвящена повышению эффективности выполнения современных расчетных приложений на высокопроизводительных вычислительных системах. В качестве инструмента повышения эффективности рассматривается векторизация программного кода. С ее помощью однотипные скалярные операции объединяются в векторные аналоги, кратно повышая производительность. Целевой платформой являются современные микропроцессоры Intel, для которых поддержан уникальный набор векторных инструкций AVX-512.

Предлагается подход к векторизации газодинамического решателя, использующего метод погруженных границ и противопотоковую схему Steger-Warming в трехмерном виде. Решатель обладает сложным программным контекстом, автоматическая векторизация которого невозможна. Рассматриваются реализация решателя, а также подходы к организации кода и приведению его к виду, пригодному для автоматической векторизации компилятором icc.

Для обеспечения автоматического применения векторизации к программному коду решателя были применены три основных эквивалентных преобразования. Во-первых, вычисления, одинаковые для всех итераций проведения расчетов, включая матричные операции, были локализованы и вынесены на этап подготовки вычислений. Во-вторых, основные функции решателя были организованы в виде плоских циклов, а структуры данных представлены в виде наборов массивов. В-третьих, к гнездам циклов была применена оптимизация расщепления по условию, с помощью которой можно уменьшить степень разветвленности управления внутри тела цикла. Данные преобразования позволяют компилятору автоматически применять векторизацию кода.

В результате выполненной работы достигнуто ускорение решателя в три раза за счет векторизации при вычислениях на вещественных числах двойной точности.

Ключевые слова*: векторизация, оптимизация, газовая динамика, метод погруженных границ, AVX-512.*

При численном решении задач газовой динамики часто приходится сталкиваться с расчетными областями со сложной или изменяющейся геометрией (например, обтекание различных тел или протекание потока внутри элементов силовых установок летательных аппаратов). Во время таких расчетов основные проблемы связаны с вычислениями, выполняемыми вблизи сложных границ, – для сложных границ крайне проблематично построение согласованных расчетных сеток. Проблема отчасти может быть решена путем использования неструктурированных и гибридных сеток. В работе [1] представлен подход к генерации гибридной расчетной сетки для численного решения вязкого течения в условиях обтекания тела со сложной геометрией. В основе данного подхода лежит применение триангуляции поверхности обтекаемого тела для построения расчетной сетки пристеночного слоя, формируемой из слоев согласованных с поверхностью призматических ячеек, тогда как для остального объема используется простая неструктурированная расчетная сетка. Главной проблемой данного подхода является сложность создания слоев призматических ячеек с помощью метода наступающего слоя, при котором необходимо рассчитывать направления движения узлов сетки, размеры шагов и предотвращать самопересечения фронта при обработке вогнутых областей.

Альтернативой является метод погруженных границ [2], с помощью которого можно избежать трудоемкого процесса создания и обслуживания согласованных на поверхности расчетных сеток. Выделим два основных подхода к разрешению граничных условий в методах погруженных границ, различающихся спо-

собом выполнения расчетов на границе: задание граничных условий посредством внешних (источниковых) членов и методы, использующие фиктивные (вспомогательные в расчетах) ячейки (или узлы). При первом из этих подходов в дифференциальные уравнения, описывающие эволюцию течения жидкости или газа, добавляются дополнительные члены, влияние которых подобно присутствию твердого обтекаемого тела, тем самым обозначая его границу. В методе штрафных функций Бринкмана для моделирования обтекаемого твердого тела используется модель пористой среды с приближающимся к нулю значением проницаемости [3].

Вторым подходом к реализации метода погруженных границ является применение так называемых фиктивных расчетных ячеек, которые используются только для вычисления газодинамических величин в других ячейках расчетной сетки. Значения физических величин в фиктивных ячейках вычисляются с помощью аппроксимации по соседним ячейкам расчетной сетки [4]. Как правило, метод погруженных границ с применением фиктивных ячеек подразумевает использование декартовых расчетных сеток, чтобы избежать усложнения шаблонов аппроксимации величин в фиктивных ячейках. Методы, использующие фиктивные ячейки, различаются способами аппроксимации значений в фиктивных ячейках. В работе [5] значения физических величин привязаны к узлам расчетной сетки, а значения в фиктивных узлах, находящихся внутри обтекаемого тела, аппроксимируются с помощью полинома второй степени через так называемую зеркальную точку (то есть точку, симметричную рассматриваемому фиктивному узлу относительно границы раздела среда–тело). В работе [6] рассматривается похожий подход, однако для моделирования течения используются конечно-объемная расчетная сетка и метод Годунова первого порядка с применением приближенного римановского решателя. Таким образом, газодинамические величины привязаны к центрам ячеек, а для аппроксимации значений в центре фиктивной ячейки используются взятые с противоположным знаком значения в зеркальной точке. Другим отличием от метода, предложенного в работе [5], является принудительное отодвижение зеркальной точки от границы в случае, если центр фиктивной ячейки лежит слишком близко к границе, для обеспечения более точной аппроксимации газодинамических величин в ней. Таким образом, в работах [5, 6] задача аппроксимации значений в фиктивных ячейках сводится к задаче аппроксимации значений внутри поля течения жидкости или газа уже без учета наличия границы.

В настоящей работе рассматривается метод погруженных границ с фиктивными ячейками, используя который, можно ограничиться простыми структурированными расчетными сетками и даже декартовыми сетками с ячейками правильной формы. Основной идеей данного метода является построение структурированной сетки, которая может пересекать сложную границу, при этом некоторые ячейки данной сетки оказываются полностью внутри расчетной области, тогда как другие – только частично. Для ячеек, целиком находящихся в расчетной области, вычисления проводятся привычным образом, а для ячеек, пересекающих границу, выполняются дополнительные вычисления, связанные с учетом граничных условий. Чтобы осуществить учет граничных условий в ячейках, пересекающих расчетную область, необходимо провести коррекцию газодинамических параметров в этих ячейках на основании данных соседних ячеек – выполнить аппроксимацию величин, участвующих в вычислениях. Использование метода погруженной границы с фиктивными ячейками позволяет упростить логику программы, перестроить программный код таким образом, чтобы он оказался более пригодным для декомпозиции задачи, а также для повышения производительности вычислений путем векторизации. За исключением подготовительных действий, а также действий, связанных с аппроксимацией газодинамических параметров вблизи сложной границы, вычисления проводятся таким образом, словно работа ведется просто с обычной структурированной расчетной сеткой. В отличие от рассмотренных методов, использующих зеркальные точки, предлагаемый в данной работе метод выполняет аппроксимацию величин в центрах фиктивных ячеек непосредственно через значения в окружающих ячейках и систему уравнений, порождаемую конфигурацией аппроксимационного шаблона и граничными условиями.

Описание трехмерного метода погруженной границы с использованием фиктивных ячеек

Реализация метода погруженной границы с фиктивными ячейками начинается с построе-

ния структурированной расчетной сетки и разметки ячеек, которые будут участвовать в вычислительном процессе. Для расчетов будем использовать наиболее простой вид сеток – декартову сетку с кубическими ячейками. Разметка ячеек производится в две фазы. На первой фазе ячейки делятся на четыре типа: обычные (COMMON) – ячейки, полностью расположенные в расчетной области, в них выполняется расчет газодинамических параметров; внутренние (INNER) – ячейки, полностью расположенные внутри обтекаемого тела, в них не происходит никаких газодинамических процессов; граничные (BORDER) – ячейки, пересекающие сложную границу, однако большей частью находящиеся вне обтекаемого тела; фиктивные (GHOST) – ячейки, пересекающие сложную границу, большая часть которых все же находится внутри обтекаемого тела (рис. 1, слева). Условно пригодными для расчетов будем считать ячейки COMMON и BORDER, тогда как ячейки GHOST и INNER относятся, скорее, к обтекаемому телу. Разметка ячеек является самостоятельной задачей и сводится к задаче пересечения объемной и поверхностной расчетных сеток, которая также может быть эффективно векторизована [7], однако в данной работе этот аспект реализации метода не рассматривается. Для разграничения ячеек типов BORDER и GHOST вместо оценки доли объема ячейки, лежащей внутри или вне обтекаемого тела, будем также использовать более простой критерий: если центр ячейки лежит вне тела, она помечается как BORDER, в противном случае – как GHOST (данное упрощение уместно при условии, что берется достаточно мелкая расчетная сетка, линейный размер ячеек в которой меньше изломов поверхности).

После первой фазы разметки ячеек требуется провести корректировку. Во время выполнения расчетов конечно-объемным методом для ячеек COMMON и BORDER нужно будет вычислять потоки вещества, импульса и энергии через все границы этих ячеек, а значит, все их соседи должны содержать газодинамические параметры. Поэтому на фазе коррекции расчетных ячеек все соседи ячеек COMMON и BORDER, которые размечены как INNER, должны быть переквалифицированы в GHOST. После выполнения этого действия нет необходимости различать ячейки COMMON и BORDER, а значит, все ячейки BORDER можно переквалифицировать в COMMON. Таким образом, после выполнения фазы коррекции значение имеют только два типа ячеек: COMMON, для которых проводятся газодинамические расчеты, и GHOST, для которых газодинамические расчеты не проводятся, а выполняется аппроксимация газодинамических параметров с учетом значений соседних ячеек COMMON и граничных условий (рис 1, справа). На рисунке 2 продемонстрирован результат разметки ячеек для тестовой расчетной области (прямоугольная расчетная область, внутри которой выполняется расчет процесса обтекания случайно расположенных сферических объектов различных радиусов).

Центральной задачей реализации метода погруженной границы является выполнение

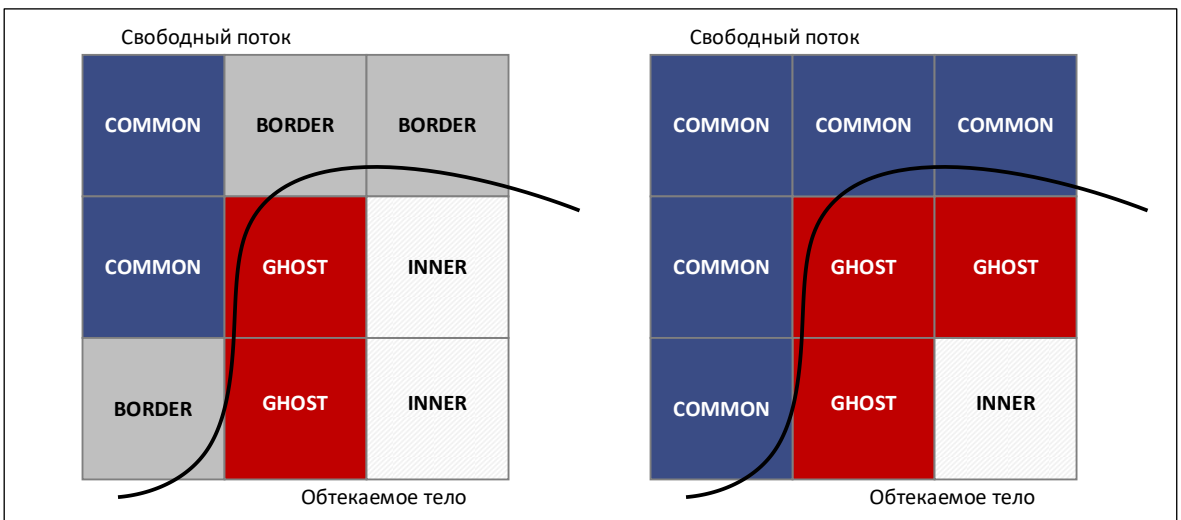

Рис. 1. Двумерная иллюстрация разметки расчетных ячеек

Fig. 1. A two-dimensional illustration of the calculation cell marking

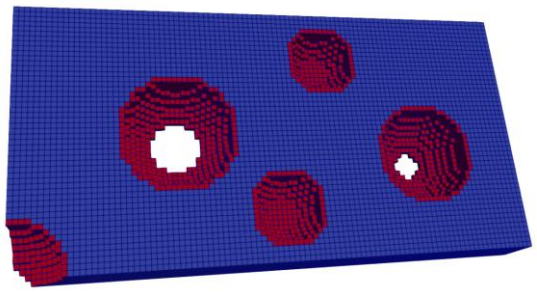

Рис. 2. Разделение ячеек на COMMON и GHOST для тестовой расчетной области

Fig. 2. Cell classification into COMMON and GHOST for the test calculation area

аппроксимации газодинамических величин GHOST-ячеек на основе данных соседних COMMON-ячеек, а также граничных условий (в данном случае подразумевается граничное условие Неймана). При этом аппроксимацию требуется выполнять как для скалярных величин (плотность и давление), так и для векторной величины скорости. Рассмотрим формулы аппроксимации газодинамических величин для GHOST-ячеек. Данные формулы были получены путем обобщения двумерных уравнений на трехмерный случай, их подробный вывод и описание приведены в [8].

Пусть в пространстве задана скалярная величина $\varphi(x, y, z)$. Если известно значение этой величины в четырех точках общего положения $\varphi_i = \varphi(x_i, y_i, z_i)$ для $i = 0, \ldots, 3$, то можно выполнить ее аппроксимацию в произвольной точке пространства с помощью линейной функции $\varphi(x, y, z) = ([1, x, y, z], \bar{a})$, $\bar{a} = [a_0, a_1, a_2, a_3]^T$ путем решения системы линейных уравнений:

$$\begin{cases} a_0 + a_1 x_0 + a_2 y_0 + a_3 z_0 = \varphi_0, \\ a_0 + a_1 x_1 + a_2 y_1 + a_3 z_1 = \varphi_1, \\ a_0 + a_1 x_2 + a_2 y_2 + a_3 z_2 = \varphi_2, \\ a_0 + a_1 x_3 + a_2 y_3 + a_3 z_3 = \varphi_3, \end{cases}$$

то есть $\bar{a} = B^{-1}_{\langle 0123\rangle}\bar{\varphi}_{\langle 0123\rangle}$, где $\bar{\varphi}_{\langle 0123\rangle} = [\varphi_0, \varphi_1, \varphi_2, \varphi_3]^T$, а матрица $B_{\langle 0123\rangle}$ имеет следующий вид:

$$\begin{bmatrix} 1 & x_0 & y_0 & z_0 \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{bmatrix}.$$

При реализации метода погруженных границ для аппроксимации величины φ используем три COMMON-ячейки (в качестве опорных точек взяты их центры $[x_1, y_1, z_1]$, $[x_2, y_2, z_2]$, $[x_3, y_3, z_3]$), в которых известны значения этой величины. В качестве дополнительной точки рассмотрим точку границы $[x_0, y_0, z_0]$, ближайшую к GHOST-ячейке, а для этой точки границы – граничное условие Неймана, то есть условие равенства нулю производной аппроксимируемой величины по направлению нормали к поверхности в точке $[x_0, y_0, z_0]$:

$$\frac{\partial\varphi}{\partial\bar{e}_0}(x_0, y_0, z_0) = \varphi' = 0 = \frac{\partial\varphi}{\partial x}e_{0,x} + \frac{\partial\varphi}{\partial x}e_{0,x} + \frac{\partial\varphi}{\partial x}e_{0,x} =$$

$$= a_1 e_{0,x} + a_2 e_{0,y} + a_3 e_{0,z}.$$

Таким образом, вектор коэффициентов аппроксимации определяется как $\bar{a} = B^{-1}_{\langle 0'123\rangle}\varphi_{\langle 0'123\rangle}$,

где $B_{\langle 0'123\rangle} = \begin{bmatrix} 0 & e_{0,x} & e_{0,y} & e_{0,z} \\ 1 & x_1 & y_1 & z_1 \\ 1 & x_2 & y_2 & z_2 \\ 1 & x_3 & y_3 & z_3 \end{bmatrix}$.

При аппроксимации векторной величины скорости $\bar{v} = [v_x, v_y, v_z]$ будем считать, что значение этой величины известно в трех соседних COMMON-ячейках (значения скорости в их центрах равны $\bar{v}_1$, $\bar{v}_2$ и $\bar{v}_3$), а в ближайшей к рассматриваемой GHOST-ячейке точке поверхности (точка с индексом 0) выполняется условие следующего вида: нормальная составляющая вектора скорости и производная тангенциальной составляющей по направлению нормали в точке поверхности равны нулю. Равенство нулю нормальной составляющей $\bar{v}_0$ можно записать в виде $v_{0,x}e_{0,x} + v_{0,y}e_{0,y} + v_{0,z}e_{0,z} = 0$, откуда, аппроксимировав составляющие вектора $\bar{v}_0$ через центр рассматриваемой GHOST-ячейки и центры трех соседних с ней COMMON-ячеек, получим следующее уравнение относительно неизвестных $v_{G,x}$, $v_{G,y}$ и $v_{G,z}$:

$$[1, x_0, y_0, z_0]B^{-1}_{\langle G123\rangle}\left(\begin{bmatrix} v_{G,x} \\ v_{1,x} \\ v_{2,x} \\ v_{3,x} \end{bmatrix}e_{0,x} + \begin{bmatrix} v_{G,y} \\ v_{1,y} \\ v_{2,y} \\ v_{3,y} \end{bmatrix}e_{0,y} + \begin{bmatrix} v_{G,z} \\ v_{1,z} \\ v_{2,z} \\ v_{3,z} \end{bmatrix}e_{0,z}\right) = 0,$$

в котором после ввода обозначения $\bar{d} = [d_G, d_1, d_2, d_3] = [1, x_0, y_0, z_0]B^{-1}_{\langle G123\rangle}$ получим

$$v_{G,x}e_{0,x} + v_{G,y}e_{0,y} + v_{G,y}e_{0,y} = -\left(\sum_{i=1}^{3} d_i(\bar{v}_i, \bar{e}_0)\right)/d_G = Q.$$

Что касается условия на тангенциальную составляющую скорости в точке поверхности, то вместо нее будем рассматривать три вектора, перпендикулярные нормали:

$$\bar{f}_1 = [-e_{0,y}, e_{0,x}, 0],$$

$$\bar{f}_2 = [-e_{0,z}, 0, e_{0,x}],$$

$$\bar{f}_3 = [0, -e_{0,z}, e_{0,y}],$$

и линейную аппроксимацию скалярных величин $(\bar{f}_i, \bar{v}_G)$ с учетом граничного условия Неймана:

$$\begin{cases} -v_{G,x}e_{0,y} + v_{G,y}e_{0,x} = \bar{t}[0, -v_{1,x}e_{0,y} + v_{1,y}e_{0,x}, -v_{2,x}e_{0,y} + \\ +v_{2,y}e_{0,x}, -v_{3,x}e_{0,y} + v_{3,y}e_{0,x}]^T = T_{xy}, \\ -v_{G,x}e_{0,z} + v_{G,z}e_{0,x} = \bar{t}[0, -v_{1,x}e_{0,z} + v_{1,z}e_{0,x}, -v_{2,x}e_{0,z} + \\ +v_{2,z}e_{0,x}, -v_{3,x}e_{0,z} + v_{3,z}e_{0,x}]^T = T_{xz}, \\ -v_{G,y}e_{0,z} + v_{G,z}e_{0,y} = \bar{t}[0, -v_{1,y}e_{0,z} + v_{1,z}e_{0,y}, -v_{2,y}e_{0,z} + \\ +v_{2,z}e_{0,y}, -v_{3,y}e_{0,z} + v_{3,z}e_{0,y}]^T = T_{yz}, \end{cases}$$

где $\bar{t} = [1, x_G, y_G, z_G]B^{-1}_{\langle 0'123\rangle}$.

Таким образом, для определения составляющих скорости $\bar{v}_G$ сформировалась следующая система уравнений:

$$\begin{bmatrix} e_{0,x} & e_{0,y} & e_{0,z} \\ -e_{0,y} & e_{0,x} & 0 \\ -e_{0,z} & 0 & e_{0,x} \\ 0 & -e_{0,z} & e_{0,y} \end{bmatrix} \begin{bmatrix} v_{G,x} \\ v_{G,y} \\ v_{G,z} \end{bmatrix} = \begin{bmatrix} Q \\ T_{xy} \\ T_{xz} \\ T_{yz} \end{bmatrix}.$$

Эта система из четырех уравнений содержит три неизвестных, то есть одно уравнение является избыточным. На самом деле избыточным является одно из трех последних уравнений, и из них можно удалить то, в которое входит минимальная по модулю составляющая вектора $\bar{e}_0$.

Несмотря на то, что аппроксимацию газодинамических величин во время расчетов нужно выполнять для каждой GHOST-ячейки на каждой итерации, полученные в приведенных формулах значения векторов $\bar{d}$ и $\bar{t}$ (а также матрица из компонентов вектора $\bar{e}_0$) зависят только от координат центров ячеек и направлений нормалей к поверхности, то есть изменяются только при изменении геометрии обтекаемого объекта, и в случае неизменяемой геометрии данные значения считаются только один раз на этапе подготовки вычислений. Это существенно, так как только при вычислении этих величин необходимо получать обратные матрицы, что является достаточно дорогой с точки зрения ресурсов процедурой.

Во время выполнения аппроксимации газодинамических величин в GHOST-ячейке возникает вопрос корректности данных вычислений. Аппроксимация может быть выполнена, если существуют обратные матрицы $B^{-1}_{\langle G123\rangle}$ и $B^{-1}_{\langle 0'123\rangle}$. Эти матрицы зависят только от выбора соседних COMMON-точек, по которым выполняется аппроксимация, а также от направления нормали к поверхности обтекаемого тела в точке поверхности, ближайшей к центру рассматриваемой GHOST-ячейки. Выбранные COMMON-точки для выполнения аппроксимации назовем шаблоном. Существование обратной матрицы $B^{-1}_{\langle G123\rangle}$ означает, что точки с индексами G, 1, 2 и 3 должны быть точками общего положения, то есть не лежать в одной плоскости. Существование же обратной матрицы $B^{-1}_{\langle 0'123\rangle}$ означает, что нормаль из точки поверхности, ближайшей к рассматриваемой GHOST-ячейке, не должна быть параллельной плоскости, образованной точками с индексами 1, 2 и 3 (которые не лежат на одной прямой).

На рисунке 3 проиллюстрирована схема выбора шаблона аппроксимации газодинамических величин в GHOST-ячейке (центр обозначен черной точкой). Если шаблон выбирается только среди ближайших соседей (индекс ячеек вдоль каждого из направлений структурированной сетки i, j, k отличается не более чем на единицу), то всего существует 26 ячеек, которые претендуют на включение в шаблон. В качестве претендентов рассматриваются только ячейки COMMON, что сразу снижает количество претендентов. Среди оставшихся ячеек нужно перебрать все комбинации троек ячеек и выбрать любую тройку, для которой существуют обратные матрицы $B^{-1}_{\langle G123\rangle}$ и $B^{-1}_{\langle 0'123\rangle}$. С большой вероятностью среди ближайших соседей можно найти подходящий шаблон аппроксимации. Если же среди них не удалось найти тройку COMMON-ячеек, которые удовлетворяют всем условиям, то можно расширить поиск.

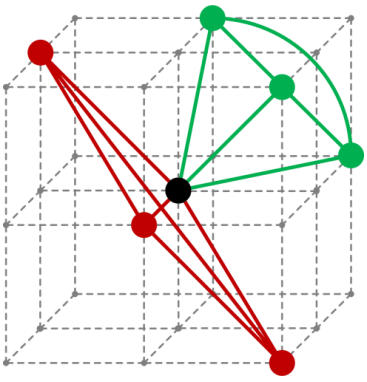

Рис. 3. Пример корректного (зеленый) и некорректного (красный) шаблонов относительно матрицы $B^{-1}_{\langle G123\rangle}$

Fig. 3. The example of correct (green) and incorrect (red) patterns relative to the matrix $B^{-1}_{\langle G123\rangle}$

Постановка газодинамической задачи и локализация пригодных для векторизации участков программного кода

Рассмотрим систему уравнений газовой динамики в трехмерном виде:

$$\frac{\partial}{\partial e}U+\frac{\partial}{\partial x}F+\frac{\partial}{\partial y}G+\frac{\partial}{\partial z}H=0,$$

$$U=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{pmatrix},\ F=\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ u(E+p)\end{pmatrix},$$

$$G=\begin{pmatrix}\rho v\\ \rho uv\\ \rho v^2+p\\ \rho vw\\ v(E+p)\end{pmatrix},\ H=\begin{pmatrix}\rho w\\ \rho uw\\ \rho vw\\ \rho w^2+p\\ w(E+p)\end{pmatrix},$$

$$E=\rho\left(\frac{V^2}{2}+e\right),\ V^2=u^2+v^2+w^2,$$

$$e(p,\rho)=\frac{p}{\rho(\gamma-1)}.$$

Данную систему будем численно решать с помощью метода конечных объемов путем вычисления потоков вектора F через грани ячейки, перпендикулярные направлению сетки i, вектора G через грани ячейки, перпендикулярные направлению сетки j, вектора H через грани ячейки, перпендикулярные направлению сетки k (при вычислении используем декартову сетку с кубическими ячейками, то есть дискретизация пространства выполняется с одним и тем же шагом по всем трем направлениям):

$$U_i^{n+1}=U_i^n-\frac{\Delta t}{\Delta h}(F_{i+1/2}-F_{i-1/2}+$$
$$+G_{j+1/2}-G_{j-1/2}+H_{k+1/2}-H_{k-1/2}).$$

Для вычисления потоков используем противопоточную схему Steger-Warming [9], в которой потоки получаются следующим образом (на примере потока F):

$$F_{i+1/2}=F_i^+(U_i^n)+F_i^-(U_{i+1}^n),$$

$$F^\pm=\frac{\rho}{2\gamma}\begin{pmatrix}\lambda_1^\pm+2(\gamma-1)\lambda_2^\pm+\lambda_5^\pm\\ (u-a)\lambda_1^\pm+2(\gamma-1)u\lambda_2^\pm+(u+a)\lambda_5^\pm\\ v(\lambda_1^\pm+2(\gamma-1)\lambda_2^\pm+\lambda_5^\pm)\\ w(\lambda_1^\pm+2(\gamma-1)\lambda_2^\pm+\lambda_5^\pm)\\ (H-ua)\lambda_1^\pm+(\gamma-1)V^2\lambda_2^\pm+(H+ua)\lambda_5^\pm\end{pmatrix},$$

$$\lambda_i^\pm=(\lambda_i\pm|\lambda_i|)/2,\ \lambda_1=u-a,\ \lambda_2=\lambda_3=\lambda_4=u,$$
$$\lambda_5=u+a,\ H=(E+p)/\rho.$$

С учетом приведенных формул один шаг выполнения расчетов с использованием метода погруженных границ должен содержать следующие необходимые действия.

1. Для всех GHOST-ячеек сетки аппроксимация примитивных газодинамических величин $D=[\rho, u, v, w, p]$ на основе единожды вычисленных шаблонов (подготовка шаблонов и предварительное вычисление матриц не рассматриваются, так как это однократные действия, вклад которых в общий процесс расчета стремится к нулю при увеличении времени расчетов). Данное действие обозначим approximate_values.

2. Для всех GHOST- и COMMON-ячеек сетки получение из вектора примитивных газодинамических величин $D=[\rho, u, v, w, p]$ вектора консервативных величин $U=[\rho, \rho u, \rho v, \rho w, E]$. Данное действие обозначим d_to_u.

3. Для всех GHOST- и COMMON-ячеек сетки вычисление векторов потоков $F^\pm$, $G^\pm$ и $H^\pm$. Эти действия обозначим как calc_fgh.

4. Для всех COMMON-ячеек корректировка вектора консервативных величин $U=[\rho, \rho u, \rho v, \rho w, E]$ с помощью потоков. Обозначим это действие calc_flows.

5. Для всех COMMON-ячеек обратный пересчет вектора консервативных величин $U=[\rho, \rho u, \rho v, \rho w, E]$ в вектор примитивных величин $D=[\rho, u, v, w, p]$. Данное действие обозначим u_to_d.

Схема выполнения расчетов приведена на рисунке 4.

Использование метода погруженных границ с фиктивными ячейками позволяет проводить газодинамические расчеты вблизи сложных границ без необходимости построения согласованных сеток. С помощью описанного метода и с применением схемы расчета Steger-Warming были проведены расчеты модельной задачи обтекания свободным потоком множества случайных сфер (при этом граница обтекаемых объектов для простоты задавалась аналитически, а не с помощью поверхностной сетки). Реализация модельной задачи находится в публичном репозитории (URL: https://github.com/r-aax/ibm_vec). Для расчетов использовалась простая декартова расчетная сетка с кубическими ячейками. Во время расчетов геометрическая конфигурация оставалась неизменной, что сделало возможным однократное вычисление вспомогательных объектов для аппроксимации газодинамических величин в фиктивных ячейках (в том числе

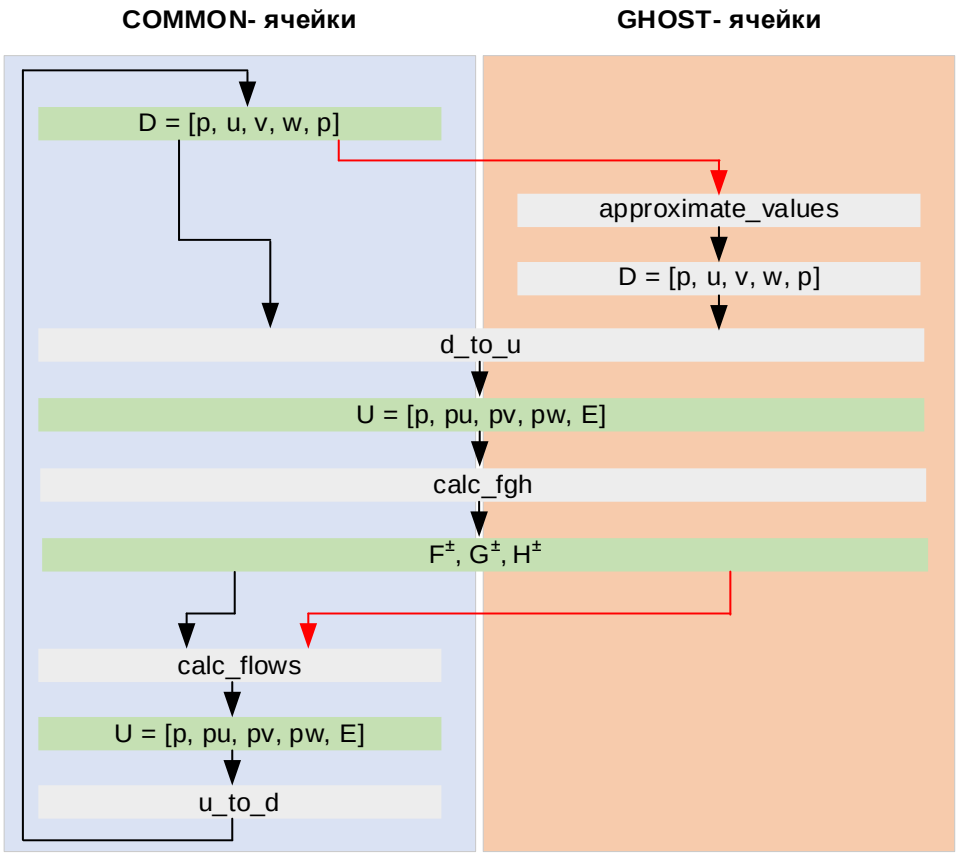

Рис. 4. Схема выполнения шага расчетов методом погруженных границ с использованием схемы Steger-Warming

Fig. 4. A scheme for performing a calculation step by the immersed boundary method using the Steger-Warming scheme

речь идет о матрицах $B^{-1}_{\langle G123\rangle}$ и $B^{-1}_{\langle 0'123\rangle}$). С помощью метода погруженных границ на декартовой расчетной сетке можно получить качественно правильную картину обтекания, что проиллюстрировано на рисунке 5.

Векторизация и результаты эксперимента

Вычисления на достаточно подробных расчетных сетках с высоким уровнем дискретизации по пространству и времени требуют серьезных вычислительных ресурсов. Для обработки крупных расчетных областей целесообразно применение высокопроизводительных вычислительных кластеров с использованием различных средств повышения быстродействия приложений и распараллеливания вычислений. Основными уровнями распараллеливания являются организация вычислений на распределенной памяти (например, с применением MPI [10]), использование многопоточного программирования (например, с применением средств OpenMP [11]), а также векторизация программного кода [12], что является низкоуровневой оптимизацией, способной кратно увеличить эффективность исполняемого кода.

Многие современные микропроцессорные архитектуры содержат расширения инструкций для векторизации вычислений. Для ARM наиболее актуальным набором векторных команд на текущий момент является расширенный набор команд под названием NEON – 128-битный расширенный SIMD, представленный а архитектурах ARMv7 и ARMv8. Набор команд NEON поддерживает векторные операции по работе с 8-, 16-, 32- и 64-битными знаковыми и беззнаковыми целыми числами, а также с 16-, 32- и 64-битными вещественными числами с плавающей точкой [13]. В современных микропроцессорах архитектуры «Эльбрус» существует поддержка коротких вектор-

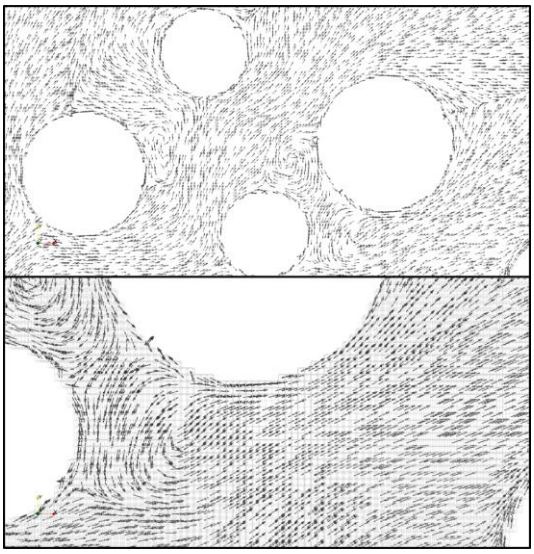

Рис. 5. Визуализация поля скоростей (двумерный срез), полученного с помощью метода погруженной границы на расчетной сетке с рисунка 2

Fig. 5. Visualization of the velocity field (two-dimensional slice) obtained using the immersed boundary method on the calculation grid from Fig. 2

ных инструкций, работающих с 8-байтными векторами [14]. Однако наиболее мощная поддержка векторных вычислений реализована в современных линейках микропроцессоров Intel. Это набор векторных инструкций AVX-512, представляющий собой 512-битное расширение 256-битных AVX-инструкций. Набор векторных инструкций AVX-512 содержит огромное количество команд, включая арифметические операции, операции множественного обращения в память, специализированные операции для работы с нейросетями и AES-шифрованием, специальные битовые операции, комбинированные операции вида $\pm a \cdot b \pm c$ и многие другие. Однако наиболее значимым преимуществом набора команд AVX-512 является наличие специальных масочных регистров, с помощью которых можно определить подмножество элементов векторов, над которыми требуется произвести векторную операцию. Использование таких масочных регистров позволяет легко векторизовать программный код, представленный в предикатной форме, и, таким образом, выполнять векторизацию кода, содержащего сложное разветвленное управление.

Автоматическая векторизация средствами оптимизирующего компилятора не может быть применена к программному контексту произвольного вида. Для принудительного применения векторизации могут быть использованы специальные функции-интринсики, которые при компиляции кода раскрываются в конкретные векторные инструкции, последовательности векторных инструкций или библиотечные вызовы (функции-интринсики подключаются к программному коду на языке C/C++ с помощью заголовочного файла <immintrin.h>). Однако прямое ручное создание векторного кода требует серьезных усилий разработчика и такой код сложен для сопровождения. Поэтому потенциальным направлением для повышения эффективности программного кода средствами векторизации является правильная организация кода, чтобы оптимизирующий компилятор мог выполнить векторизацию в автоматическом режиме. Наиболее удобным для выполнения автоматической векторизации программным контекстом является так называемый плоский цикл. Плоским циклом, реализованным на языке программирования C, будем называть цикл for, отвечающий следующим требованиям: все итерации плоского цикла независимы друг от друга (то есть могут выполняться в любом порядке, в том числе и одновременно); внутри плоского цикла осуществляется работа с массивами атомарных данных, причем на i-й итерации цикла к массивам выполняются только обращения вида $a[i]$ (во многих случаях это условие можно выполнить в полном объеме, в других несколько ослабить, жертвуя эффективностью векторизации). Также к программному коду могут предъявляться дополнительные требования, например, выровненность массивов данных в памяти на одну и ту же величину, равную ширине векторного регистра (однако это требование также можно обойти). При организации программного кода должным образом в виде набора плоских циклов компилятор способен выполнить автоматическую векторизацию кода.

В рамках данной работы проводился эксперимент по организации программного кода газодинамического решателя, работающего с данными в формате double (вещественные числа размером 64 бита) с целью векторизации под микропроцессоры Intel Xeon Phi Knights Landing [15]. Данный микропроцессор был выбран, так как, кроме обычных векторных арифметических операций набора AVX-512F, он поддерживает также такие полезные наборы, как AVX-512PF – операции предварительной подкачки данных из памяти и AVX-512ER –

векторные команды для вычисления экспоненты и обратных значений, которые крайне полезны в физических приложениях.

Рассмотрим на примере реализации функции d_to_u из общей схемы вычислений, представленной на рисунке 4, способы организации вычислений интуитивным способом и в виде плоского цикла. Итак, функция d_to_u переводит вектор примитивных газодинамических величин $D = [\rho, u, v, w, p]$ в вектор консервативных величин $U = [\rho, \rho u, \rho v, \rho w, E]$ для каждой расчетной ячейки. Интуитивным способом вычисления могут быть организованы в виде следующей схемы: для каждой ячейки создается структура, которая содержит все необходимые величины, и действия по переводу вектора D в вектор U выполняются над полями данной структуры (листинг 1, слева). При попытке векторизации данного цикла во время объединения нескольких последовательных итераций (объединяться должны 8 итераций, так как размер элемента данных равен 64 битам, а длина векторного регистра 512 битам) вместо чтения из памяти скалярного значения (например, плотности) должно происходить чтение значений плотности из нескольких последовательно расположенных структур данных. То есть чтение вектора должно осуществляться не из последовательной области памяти. Для таких целей в наборе векторных инструкций AVX-512 предусмотрены инструкции gather/scatter, однако их эффективность гораздо ниже чтения последовательной области памяти размером 512 бит даже при использовании предварительной подкачки данных.

Естественным решением оптимизации вычислений является организация расположения данных в памяти в виде набора массивов (листинг 1, справа). После объединения нескольких итераций цикла команды скалярного доступа в память трансформируются в векторные аналоги доступа к последовательной области памяти размером 512 бит. Вся же арифметика, которая присутствует в коде функции (операции сложения, вычитания, умножения, деления, взятие обратной величины), имеет свои векторные аналоги в наборе инструкций AVX-512.

```
struct Cell
{
    double rho;
    double u;
    double v;
    double w;         // организация
    double p;         // данных в виде
    double rho_u;     // массива структур
    double rho_v;
    double rho_w;
    double E;
};

Cell cells[N];

void d_to_u()
{
    for (int i = 0; i < N; i++)
    {
        double rho = cells[i].rho;
        double u = cells[i].u;
        double v = cells[i].v;
        double w = cells[i].w;
        double p = cells[i].p;

        cells[i].rho_u = rho * u;
        cells[i].rho_v = rho * v;
        cells[i].rho_w = rho * w;
        cells[i].E = 0.5 * rho
                     * (u * u + v * v + w * w)
                     + p / (GAMMA - 1.0);
    }
}
```

```
double rhos[N];
double us[N];
double vs[N];
double ws[N];         // организация
double ps[N];         // данных в виде
double rho_us[N];     // набора массивов
double rho_vs[N];
double rho_ws[N];
double Es[N];

void d_to_u()
{
    for (int i = 0; i < N; i++)
    {
        double rho = rhos[i];
        double u = us[i];
        double v = vs[i];
        double w = ws[i];
        double p = ps[i];

        rho_us[i] = rho * u;
        rho_vs[i] = rho * v;
        rho_ws[i] = rho * w;
        Es[i] = 0.5 * rho
                * (u * u + v * v + w * w)
                + p / (GAMMA - 1.0);
    }
}
```

Листинг 1. Организация данных для программной реализации функции d_to_u в виде массива структур (слева) и набора отдельных массивов (справа)

Listing 1. Data organization for the software implementation of the d_to_u function in the form of an structure array (left) and as a set of separate arrays (right)

Другой критической проблемой, влияющей на эффективность векторизации кода, является наличие условных операций внутри векторизуемого плоского цикла. Конечно, набор инструкций AVX-512 содержит специальные масочные аргументы, с помощью которых можно векторизовать программный код с управлением практически любой сложности (разветвленное управление, гнезда циклов, циклы с вызовами функций, циклы с нерегулярным числом итераций) [16], однако при увеличении количества условий в коде эффективность векторизации снижается.

Как пример такого негативного эффекта можно рассмотреть логику гнезда циклов из функции calc_flows, в которой корректируются консервативные величины с помощью потоков через все грани ячейки. Пусть рассматриваемая расчетная область представлена структурированной сеткой размера $NX \times NY \times NZ$ по направлениям i, j, k соответственно. Тогда при учете потоков через левую грань каждой ячейки необходимо отдельно обрабатывать случай $i = 0$, что соответствует граничному условию расчетной области, причем данное граничное условие может быть разных видов (свободное протекание, условие жесткой стенки) (см. листинг 2, слева). Аналогично нужно рассматривать особые случаи для всех шести граней ячейки, что резко увеличивает количество условий внутри цикла и снижает эффективность векторизации.

Можно заметить, что условия обработки границ расчетной области не уникальны для тройки координат ячейки (i, j, k), но являются константными для некоторого среза ячеек сетки. Это означает, что для гнезда циклов такое условие является частично константным, и гнездо может быть разбито по этому условию. На листинге 2 справа продемонстрировано разбиение гнезда циклов по условию $i = 0$, после чего итоговое гнездо, содержащее основную часть вычислений, освобождается от условия. Аналогичным образом можно выполнить разбиение по остальным условиям, освободив от них основное гнездо циклов, которое после этого успешно векторизуется.

На рисунке 4 приведена общая схема выполнения расчетов для одной итерации численного метода. Основными функциями в этих расчетах являются approximate_values, d_to_u, calc_fgh, calc_flows, u_to_d. Во время проведения эксперимента на модельной задаче был собран профиль исполнения до векторизации и после нее, результаты распределения времени исполнения между этими основными функциями представлены на рисунке 6.

Результаты ускорения после векторизации каждой функции в отдельности и суммарного ускорения всего расчетного кода отражены на рисунке 7. Заметим, что во время векторизации расчетного кода, работающего с вещественными числами двойной точности, один векторный регистр содержит 8 элементов данных.

```
for (int k = 0; k < NZ; k++)
{
    for (int j = 0; j < NY; j++)
    {
        for (int i = 0; i < NX; i++)
        {
            if (i == 0)
            {
                // левое граничное условие
                ...
            }

            // остальной код
            ...
        }
    }
                    }
```

```
for (int k = 0; k < NZ; k++)
{
    for (int j = 0; j < NY; j++)
    {
        // левое граничное условие с i = 0
        ...
    }
}

for (int k = 0; k < NZ; k++)
{
    for (int j = 0; j < NY; j++)
    {
        for (int i = 1; i < NX; i++)
        {
            // остальной код
            ...
        }
    }
                        }
```

Листинг 2. Схема разбиения гнезда циклов по условию на примере обработки левого граничного условия. Гнездо циклов в изначальном виде (слева) и после вынесения обработки граничного условия при i = 0 (справа)

Listing 2. The scheme of splitting the cycle nest by a condition on the example of processing the left boundary condition. The cycle nest in its original form (left) and after processing the boundary condition at i = 0 (right)

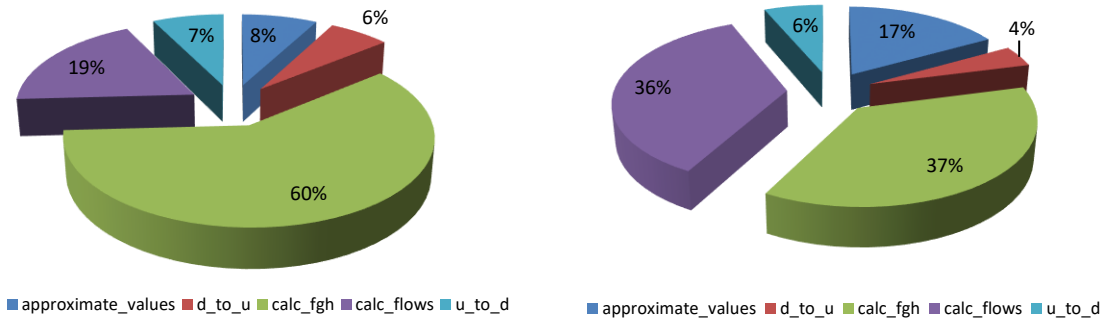

Рис. 6. Распределение времени выполнения между основными функциями до (слева) и после (справа) векторизации

Fig. 6. Distribution of execution time between the main functions before (left) and after (right) vectorization

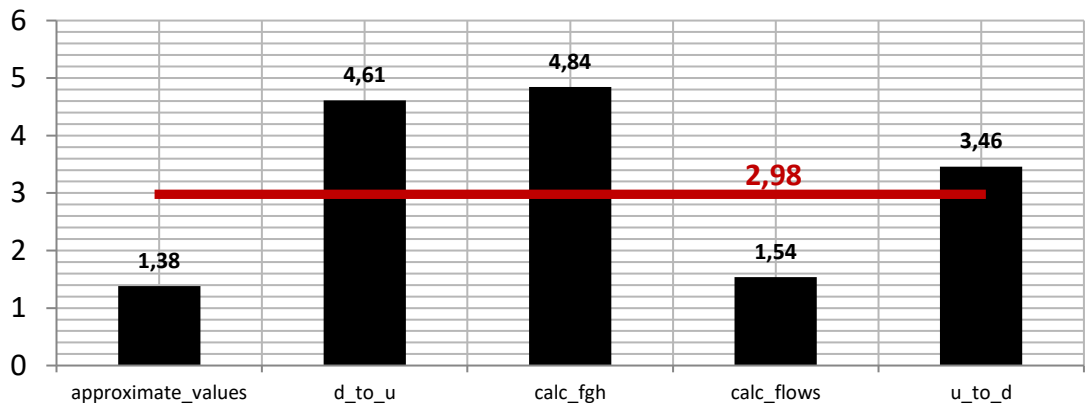

Рис. 7. Ускорение отдельных функций и суммарное ускорение

Fig. 7. Acceleration of individual functions and total acceleration

Можно эмпирически считать, что при замене в плоском цикле всех скалярных инструкций на точные векторные аналоги логическое ускорение составит 8 раз (количество инструкций в итерации цикла останется неизменным, но количество итераций в плоском цикле уменьшится в 8 раз). Однако на практике ускорение в 4 раза уже можно считать успешным.

Следует заметить, что ускорения различных функций в результате векторизации сильно отличаются друг от друга. Причиной являются конкретные особенности каждой из этих функций.

Наибольшее ускорение продемонстрировала функция calc_fgh, реализующая вычисление потоков $F^{\pm}$, $G^{\pm}$, $H^{\pm}$. Этому способствовали следующие ее особенности. Во-первых, функция содержит плоский цикл, полностью удовлетворяющий всем условиям, что делает возможной замену всех скалярных операций на векторные аналоги. Во-вторых, в скалярной версии используются библиотечные вызовы abs (вычисление абсолютного значения) и sqrt (вычисление квадратного корня). В наборе векторных инструкций AVX-512 есть векторные операции VAND (с помощью побитовой операции над вещественным числом можно обратить в ноль бит знака данного числа) и VSQRT, которые реализуют эти действия просто одной операцией, что сильно ускоряет исполнение. Наконец, обилие операций умножения и сложения делает возможным применение векторных комбинированных операций вида $\pm a \cdot b \pm c$.

Ускорение выше среднего продемонстрировали функции d_to_u и u_to_d. Они также состоят из плоских циклов, поэтому скалярные операции могут быть заменены на векторные аналоги. Более низкое ускорение функции u_to_d объясняется наличием операций деления, которые выполняются медленнее операций сложения и умножения.

Показатель ускорения функции calc_flows составил менее двух раз даже после избавления от всех условий, связанных с обработкой границ расчетной области. Такая низкая эффективность векторизации объясняется тем, что внутри этой функции присутствуют циклы, которые не являются плоскими. Например, в функции calc_flows для корректировки консервативных величин внутри ячейки (i, j, k) требуется обращаться за данными ко всем смежным по граням ячейкам: ($i \pm 1, j, k$), ($i, j \pm 1, k$), ($i, j, k \pm 1$). Это нарушает требование по унификации обращения к массивам данных на одной итерации плоского цикла и немедленно приводит к появлению операций gather/scatter, понижая эффективность векторизации. Для функции approximate_values ситуация выглядит аналогичной, так как для выполнения аппроксимации данных в ячейке (i, j, k) требуется обращаться за данными в ячейки, относящиеся к шаблону аппроксимации (на рисунке 4 нарушение требования унификации обращения за данными продемонстрировано красными стрелками, означающими, что при обработке одной ячейки сетки возникает необходимость обращаться за данными к другой ячейке).

Несмотря на низкие показатели эффективности векторизации функций, содержащих циклы, не являющиеся плоскими, общая эффективность от векторизации метода погруженных границ составила примерно три раза, при этом удалось ограничиться только реорганизацией кода без использования ассемблера или функций-интринсиков для форсированного использования векторных инструкций.

Заключение

В работе рассмотрен метод погруженных границ для выполнения газодинамических расчетов обтекания тел со сложной геометрией. Данный метод реализован в трехмерной постановке в условиях постоянной геометрии и аналитически заданных границ обтекаемых тел, разработан программный код на языке C. Для тестовой задачи проведены расчеты с использованием противопотоковой схемы Steger-Warming.

Изучены вопросы векторизации программного кода для микропроцессоров Intel с поддержкой набора инструкций AVX-512, предложены подходы по реорганизации программного кода для обеспечения автоматической векторизации кода с помощью оптимизирующего компилятора icc для микропроцессора Intel Xeon Phi Knights Landing. В результате экспериментов достигнуто ускорение расчетного кода за счет векторизации при использовании вещественных чисел двойной точности.

Предложенные методы подготовки кода для векторизации могут быть использованы для повышения эффективности любых расчетных приложений при представлении логики программы в виде объединения плоских циклов.

Работа выполнена в МСЦ РАН в рамках госзадания по теме FNEF-2022-0016. В исследованиях использовался суперкомпьютер МВС-10П.

Литература

1. Ye H., Liu Y., Chen B., Liu Z., Zheng J., Pang Y., Chen J. Hybrid grid generation for viscous flow simulations in complex geometries. Advances in Aerodynamics, 2020, vol. 2, art. 17. DOI: 10.1186/s42774 020-00042-x.

2. Цветкова В.О., Абалакин И.В., Бобков В.Г., Жданова Н.С., Козубская Т.К., Кудрявцева Л.Н. Моделирование обтекания винта на адаптивной неструктурированной сетке с использованием метода погруженных границ // Математическое моделирование. 2021. Т. 33. № 8. С. 59–82. DOI: 10.20948/mm-2021-08-04.

3. Жданова Н.С., Абалакин И.В., Васильев О.В. Расширение метода штрафных функций Бринкмана для сжимаемых течений вокруг подвижных твердых тел // Математическое моделирование. 2022. Т. 34. № 2. С. 41–57. DOI: 10.20948/mm-2022-02-04.

4. Chi C., Abdelsamie A., Thévenin D. A directional ghost-cell immersed boundary method for incompressible flows. J. of Comput. Phys., 2020, vol. 404, art. 109122. DOI: 10.1016/j.jcp.2019.109122.

5. Yousefzadeh M., Battiato I. High-order ghost-cell immersed boundary method for generalized boundary conditions. Int. J. of Heat and Mass Transfer, 2019, vol. 137, pp. 585–598. DOI: 10.1016/j.ijheatmasstransfer.2019.03.061.

6. Chi C., Lee B.J., Im H.G. An improved ghost-cell immersed boundary method for compressible flow simulations. Int. J. for Numerical Methods in Fluids, 2017, vol. 83, pp. 132–148. DOI: 10.1002/fld.4262.

7. Рыбаков А.А. Векторизация нахождения пересечения объемной и поверхностной расчетных сеток для микропроцессоров с поддержкой AVX-512 // Тр. НИИСИ РАН. 2019. Т. 9. № 5. С. 5–14.

8. Рыбаков А.А. Метод погруженной границы с использованием фиктивных ячеек в трехмерной постановке // Современные информационные технологии и ИТ-образование. 2020. Т. 16. № 2. С. 321–330.

9. Смирнова Н.С. Сравнение схем с расщеплением потока для численного решения уравнений Эйлера сжимаемого газа // Тр. МФТИ. 2018. Т. 10. № 1. С. 122–141.

10. Chunduri S., Parker S., Balaji P., Harms K., Kumaran K. Characterization of MPI usage on a production supercomputer. Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis, 2018, pp. 386–400. DOI: 10.1109/SC.2018.00033.

11. Majeed R., Farrukh R., Riaz O., Ali S., Samad A., Khan M. Parallel implementation of FEM solver for shared memory. Math. Problems in Engineering, 2022, vol. 2022, art. 2375102. DOI: 10.1155/2022/2375102.

12. Tian X., Saito H., Preis S.V., Garcia E.N., Kozhukhov S.S. et al. Effective SIMD vectorization for Intel Xeon Phi coprocessors. Sci. Programming, 2015, art. 269764. DOI: 10.1155/2015/269764.

13. Arm Developer. Introducing NEON Development Article. URL: https://developer.arm.com/documentation/dht0002/a/Introducing-NEON/What-is-NEON- (дата обращения: 28.09.2022).

14. Волконский В.Ю., Брегер А.В., Бучнев А.Ю., Грабежной А.В., Ермолицкий А.В., Муханов Л.Е. и др. Методы распараллеливания программ в оптимизирующем компиляторе // Вопросы радиоэлектроники. 2012. Т. 4. № 3. С. 63–88.

15. Jeffers J., Reinders J., Sodani A. Intel Xeon Phi Processor High Performance Programming. Morgan Kaufmann Publ., 2016, 662 p.

16. Savin G.I., Shabanov B.M., Rybakov A.A., Shumilin S.S. Vectorization of flat loops of arbitrary structure using instructions AVX-512. Lobachevskii J. of Math., 2020, vol. 41, no. 12, pp. 2566–2574. DOI: 10.1134/S1995080220120331.

Software & Systems
DOI: 10.15827/0236-235X.141.130-143

Received 30.09.22, Revised 10.11.22
2023, vol. 36, no. 1, pp. 130–143

Vectorization of the three-dimensional immersed boundary method for improving the efficiency of calculations on Intel microprocessors

A.A. Rybakov [1], *Ph.D. (Physics and Mathematics), Leading Researcher, rybakov@jscc.ru*
A.O. Meshcheryakov [1], *Junior Researcher, alexmeshr@jscc.ru*

[1] *Joint Supercomputer Center of the Russian Academy of Sciences – JSCC, Moscow, 119991, Russian Federation*

Abstract. The work is devoted to increasing the efficiency of modern computational applications on high-performance computing systems. The authors consider program code vectorization as a tool for increasing efficiency. Vectorization helps combining scalar operations of the same type into vector analogs, significantly increasing performance. Modern Intel microprocessors were chosen as the target platform, for which a unique set of vector instructions AVX-512 is supported.

The paper considers an approach to vectorization of a gas dynamic solver using the immersed boundary method and the Steger-Warming upwind scheme in 3D case. This solver has a complex programming context that cannot be vectorized automatically. The paper considers the implementation of the solver, as well as approaches to organizing the code and bringing it to a form suitable for automatic vectorization by the icc compiler.

To ensure automatic application of vectorization to the solver code, three basic equivalent transformations were applied. First, the calculations, which are the same for all iterations including matrix operations, were localized and brought to the stage of preparing calculations. Second, the main functions of the solver were organized as flat loops, and the data structures were presented as sets of arrays. Third, splitting by condition optimization was applied to loop nests, which can be used to reduce the degree of control branching inside the loop body. These transformations allow the compiler to automatically apply code vectorization.

As a result of the work performed, the solver was accelerated by a factor of 3 due to vectorization when performing calculations on double-precision real numbers.

Keywords: vectorization, optimization, fluid dynamics, immersed boundary method, AVX-512.

Acknowledgements. *The work was carried out at the Joint Supercomputer Center of the Russian Academy of Sciences within the framework of the state assignment on the topic FNEF 2022 0016. The authors used MVS-10P supercomputer in the studies.*

References

1. Ye H., Liu Y., Chen B., Liu Z., Zheng J., Pang Y., Chen J. Hybrid grid generation for viscous flow simulations in complex geometries. *Advances in Aerodynamics*, 2020, vol. 2, art. 17. DOI: 10.1186/s42774 020-00042-x.
2. Cvetkova V.O., Abalakin I.V., Bobkov V.G., Zhdanova N.S., Kozubskaya T.K., Kudryavceva L.N. Simulation of flow near rotating propeller on adaptive unstructured meshes using immersed boundary method. *Math. Models Comput. Simul*., 2021, vol. 33, no. 8, pp. 59–82. DOI: 10.20948/mm-2021-08-04 (in Russ.).
3. Zhdanova N.S., Abalakin I.V., Vasilev O.V. Generalized Brinkman volume penalization method for compressible flows around moving obstacles. *Math. Models Comput. Simul*., 2022, vol. 34, no. 2, pp. 41–57. DOI: 10.20948/mm-2022-02-04 (in Russ.).
4. Chi C., Abdelsamie A., Thévenin D. A directional ghost-cell immersed boundary method for incompressible flows. *J. of Comput. Phys*., 2020, vol. 404, art. 109122. DOI: 10.1016/j.jcp.2019.109122.
5. Yousefzadeh M., Battiato I. High-order ghost-cell immersed boundary method for generalized boundary conditions. *Int. J. of Heat and Mass Transfer*, 2019, vol. 137, pp. 585–598. DOI: 10.1016/j.ijheatmasstransfer.2019.03.061.
6. Chi C., Lee B.J., Im H.G. An improved ghost-cell immersed boundary method for compressible flow simulations. *Int. J. for Numerical Methods in Fluids*, 2017, vol. 83, pp. 132–148. DOI: 10.1002/fld.4262.
7. Rybakov A.A. Vectorization of finding the intersection of volume grid and surface grid for microprocessors with AVX-512 support. *Proc. SRISA RAS*, 2019, vol. 9, no. 5, pp. 5–14 (in Russ.).
8. Rybakov A.A. Immersed boundary method using ghost cells in a three-dimensional case. *Modern Information Technologies and IT-Education*, 2020, vol. 16, no. 2, pp. 321–330 (in Russ.).
9. Smirnova N.S. Comparison of flux splitting schemes for numerical solution of the compressible Euler equations. *Proc. MIPT*, 2018, vol. 10, no. 1, pp. 122–141 (in Russ.).
10. Chunduri S., Parker S., Balaji P., Harms K., Kumaran K. Characterization of MPI usage on a production supercomputer. *Proc. Int. Conf. for High Performance Computing, Networking, Storage and Analysis*, 2018, pp. 386–400. DOI: 10.1109/SC.2018.00033.
11. Majeed R., Farrukh R., Riaz O., Ali S., Samad A., Khan M. Parallel implementation of FEM solver for shared memory. *Math. Problems in Engineering*, 2022, vol. 2022, art. 2375102. DOI: 10.1155/2022/2375102.
12. Tian X., Saito H., Preis S.V., Garcia E.N., Kozhukhov S.S. et al. Effective SIMD vectorization for Intel Xeon Phi coprocessors. *Sci. Programming*, 2015, art. 269764. DOI: 10.1155/2015/269764.
13. *Arm Developer. Introducing NEON Development Article*. Available at: https://developer.arm.com/documentation/dht0002/a/Introducing-NEON/What-is-NEON- (accessed September 28, 2022).
14. Volkonsky V.Yu., Breger A.V., Buchnev A.Yu., Grabezhnoy A.V., Ermolicky A.V., Mukhanov L.E. et al. Program parallelization methods implemented in optimizing compiler. *Issues of Radio Electronics*, 2012, vol. 4, no. 3, pp. 63–88 (in Russ.).
15. Jeffers J., Reinders J., Sodani A. *Intel Xeon Phi Processor High Performance Programming*. Morgan Kaufmann Publ., 2016, 662 p.
16. Savin G.I., Shabanov B.M., Rybakov A.A., Shumilin S.S. Vectorization of flat loops of arbitrary structure using instructions AVX-512. *Lobachevskii J. of Math.*, 2020, vol. 41, no. 12, pp. 2566–2574. DOI: 10.1134/S1995080220120331.

Для цитирования

Рыбаков А.А., Мещеряков А.О. Векторизация трехмерного метода погруженных границ для повышения эффективности расчетов на микропроцессорах Intel // Программные продукты и системы. 2023. Т. 36. № 1. С. 130–143. DOI: 10.15827/0236-235X.141.130-143.

For citation

Rybakov A.A., Meshcheryakov A.O. Vectorization of the three-dimensional immersed boundary method for improving the efficiency of calculations on Intel microprocessors. *Software & Systems*, 2023, vol. 36, no. 1, pp. 130–143 (in Russ.). DOI: 10.15827/0236-235X.141.130-143.