9. Smalyuk V. A. et al. Experimental results of radiation-driven, layered deuterium-tritium implosions with adiabat-shaped drives at the National Ignition Facility // Phys.Plasmas. – 2016. – 23. – 102703.

10. Haan S. W. et al. Point design targets, specifications, and requirements for the 2010 ignition camaign on the National Ignition Facility // Phys.Plasmas. – 2011. – 18. – 051001.

11. Michel et al. // Phys.Plasmas. – 2013. – 20. – 056308.

12. Софронов И. Д., Бельков С. А., Винокуров О. А., Мхитарьян Л. С., Рябикина Н. А. Методика расчета спектрального переноса излучения в двумерном комплексе МИМОЗА-НД // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. – 2000. – Вып. 1 – 8.

## 3D COMPUTATION OF THE EXPERIMENT AT NIF FACILITY: A SCHEME OF INTRODUCE LASER RADIATION INTO THE HOHLRAUM AND COMPRESSION OF AN ICF TARGET

*A. S. Gnutov, S. A. Dontsov, D. M. Linnik, L. F. Potapkina, P. V. Rybachenko*

Russian Federal Nuclear Center –
All-Russian Research Institute of Experimental Physics, Sarov

This paper describes program realization for numerical simulation of implosion process for a thermonuclear target under operation condition at National Ignition Facility (NIF), the results of 3D computations in a shock-capturing approach are provided, as well as comparison with available experimental results.

*Key words*: laser target, hohlraum, laser radiation, spectral radiation transfer, parallel computations.

УДК 519.6

## ВЕКТОРИЗОВАННЫЕ АЛГОРИТМЫ ДЛЯ ОДНОЙ РАЗНОСТНОЙ СХЕМЫ ГАЗОВОЙ ДИНАМИКИ НА НЕСТРУКТУРИРОВАННЫХ МНОГОГРАННЫХ СЕТКАХ В ЛАГРАНЖЕВОЙ МЕТОДИКЕ ТИМ-3D

*Ф. О. Голомидов, А. А. Воропинов*

Российский федеральный ядерный центр –
Всероссийский НИИ экспериментальной физики, Саров

В данной работе описываются векторизованные алгоритмы для расчета уравнений газовой динамики на неструктурированной многогранной сетке произвольного вида, используемой в методике ТИМ-3D. Приводится описание одной из разностных схем, реализованных в методике на основе лагранжевого подхода, и реализующие ее традиционные скалярные алгоритмы.

Описывается преобразование вычислительных блоков для векторизации разностной схемы. В основу положен подход, основанный на тетраэдрализации ячеек исходной сетки, что позволяет единым образом представить сетку из ячеек произвольного вида. В результате проведения тестовых расчетов были получены результаты, согласующиеся с результатами, полученными по методике ТИМ-3D, а ускорение от векторизации составило ~2.5 раза на сопроцессоре Intel Xeon Phi, и ~1.1 раза на универсальном процессоре Intel Haswell.

*Ключевые слова:* неструктурированная многогранная лагранжева сетка, векторизация, лагранжева газовая динамика, Intel Xeon Phi.

## Введение

Методика ТИМ-3D [1] предназначена для расчета нестационарных трехмерных задач механики сплошной среды на неструктурированных многогранных сетках произвольного вида на основе лагранжевого подхода. Примеры сеток представлены на рис. 1.

В последнее время широкое распространение в СуперЭВМ получили вычислительные устройства с векторизацией. К данному типу устройств можно отнести современные универсальные процессоры, различного рода сопроцессоры и арифметические ускорители. Наиболее удобными являются сопроцессоры, основанные на архитектуре x86, что позволяет не использовать специфичные конструкции и средства разработки программ. Более того, разработка и оптимизация алгоритмов под сопроцессоры данной архитектуры, зачастую положительно сказывается и на эффективности алгоритмов для универсальных процессоров.

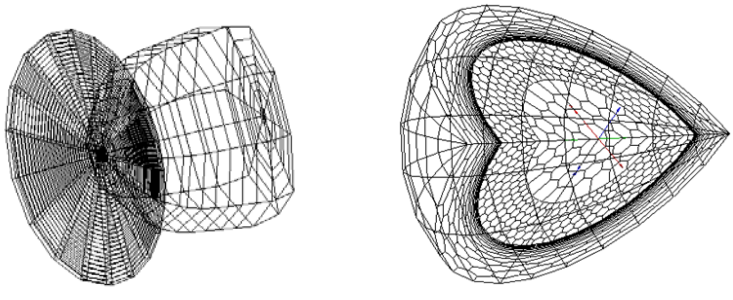

Рис. 1. Примеры начальных расчетных сеток методики ТИМ-3D

В ходе проведения работ по тестированию и доработке алгоритмов OpenMP распараллеливания методики ТИМ-3D [2] на вычислительной системе «Каскад» [3], стало очевидно, что добиться высоких значений ускорения алгоритмов без использования такого ресурса, как векторизация, затруднительно. Это связано с тем, что в состав вычислительной системы «Каскад» входят сопроцессоры Intel Xeon Phi [4], которые ориентированы на работу с векторными инструкциями шириной 512-бит. Само вычислительное устройство Xeon Phi обладает большим набором «легковесных» ядер, что в сочетании с широким векторным регистром позволяет достигать высокой суммарной производительности. При этом векторизация потенциально позволяет увеличить производительность в несколько раз.

## 1. Разностная схема методики ТИМ-3D

За основу в данной работе была взята неконсервативная газодинамическая разностная схема методики ТМК-УП, описанная в работе [5]. Данная схема реализована и для методики ТИМ-3D, где обобщена на случай произвольных сеток. Следует отметить, что векторные алгоритмы, описанные в данной работе, нацелены на дальнейшее обобщение для других блоков методики ТИМ-3D.

### 1.1. Основные положения

Для получения разностной схемы используются уравнения газовой динамики с использованием лагранжева подхода [5].

Область интегрирования разбивается на ячейки – произвольные многогранники (произвольное количество граней, у граней произвольное количество узлов, в узле сходится две или более граней), в узлах может сходиться произвольное количество ребер и граней. Разностная схема, используемая в методике для аппроксимации дифференциальных уравнений механики сплошной (в газодинамическом приближении) среды, строится аналогично схеме Неймана. Каждый узел разностной сетки (относится к вершине многогранника – ячейке) описывается координатами, имеет массу и скорость, которая считается постоянной в окрестности узла. Масса ячейки, удельная внутренняя энергия, плотность вещества, искусственная вязкость относятся к ячейкам сетки и считаются постоянными внутри.

При построении конечно-разностных соотношений считается, что:

1. Давление $p$, плотность ρ, удельная внутренняя энергия $\varepsilon$, вязкость $q$ определены внутри ячеек, и постоянны внутри них.
2. Каждой ячейке соответствует масса вещества $M$, заключенная в её объёме.
3. Узлы характеризуются координатами $\vec{R}$ и вектором скорости $\vec{U}$.
4. Каждому узлу приписывается некая масса m, которая состоит из частей масс ячеек, примыкающих к данному узлу и которая в процессе счета не меняется.
5. Ускорения постоянны внутри фигуры, ограничивающей объем узла.
6. При движении ребра ячеек остаются прямолинейными.
7. Грани сетки в общем случае являются неплоскими. Поэтому при построении схемы грань представляется в виде совокупности треугольников.

Расчет уравнений газовой динамики состоит из двух основных этапов:

1. Расчет узловых величин (скорости и координаты)
2. Расчет ячеечных величин (объем, плотность, энергия и давление).

### 1.2. Вычисление объема ячеек

В описанном выше подходе грани являются произвольными пространственными многоугольниками, не являющимися плоскими. Для того чтобы воспользоваться положениями теории многогранников (в частности вычисления объема) необходимо привести грани к плоскому виду. Поэтому грань представляется в виде набора треугольников. Для этого используется дополнительная точка, называемая центром грани. Центр грани соединяется с вершинами отрезками (рис. 2). Треугольники образуются центром грани и двумя последовательными узлами грани. Совокупность треугольников, полученных при таком разбиении грани и считается триангуляцией грани и поверхностью, разделяющей две смежные ячейки.

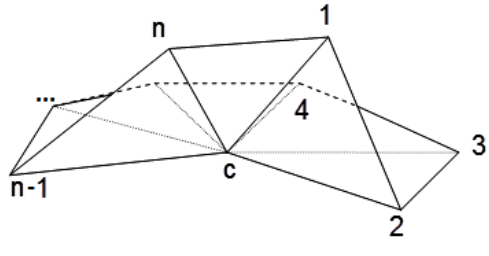


Рис. 2. Триангуляция грани: 1, 2,…,n – вершины; c – центр грани

Центр грани определяется через среднее арифметическое координат ее вершин

$$\vec{R}_c = \frac{1}{n}\sum_{i=1}^{n}\vec{R}_i \,.$$

Выполнив триангуляцию всех граней, рассматриваемая ячейка превращается в многогранник, состоящий из треугольных граней. Данный многогранник является ориентируемым и поэтому для него можно определить объем. Однако для произвольного многогранника прямое определение объема затруднительно, поэтому его необходимо разбить на более простые тела – тетраэдры (треугольные пирамиды). Для этого выбирается еще одна точка, не обязательно находящаяся внутри многогранника (можно выбрать центр ячейки). Объем ячейки определяется как сумма объемов всех тетраэдров, вершинами каждого из которых являются: центр ячейки, центр грани и две соседние вершины этой грани (рис. 3), т. е.

$$V = \frac{1}{6}\sum_{j=1}^{l}\sum_{i=1}^{n^j}\left\langle \vec{R}_c^{\,j}, \vec{R}_i^{\,j}, \vec{R}_{i+1}^{\,j} \right\rangle \qquad (1)$$

где $l$ – число граней ячейки; $n^j$ – количество вершин $j$-й грани; $\vec{R}_c^{\,j}, \vec{R}_i^{\,j}, \vec{R}_{i+1}^{\,j}$ – векторы, начала которых лежат в центре ячейки, а концы, соответственно, в центре $j$-й грани и в двух последовательных ее вершинах; $\left\langle \vec{R}_c^{\,j}, \vec{R}_i^{\,j}, \vec{R}_{i+1}^{\,j} \right\rangle$ – смешанное произведение векторов. Направление обхода грани при последовательном выборе ее вершин – против часовой стрелки, если смотреть снаружи ячейки.

Координаты центра ячейки вычисляются как среднее арифметическое от координат центров граней.

Необходимо отметить, что в традиционной схеме методики ТИМ-3D триангуляция граней и тетраэдрализация ячеек производится «на лету» при вычислении объемов и в явном виде на протяжении счета не хранится. Это же касается и расчета узловых величин – там триангуляция грани делается неявно через получение соседства в узле. В предложенных в данной работе алгоритмах, тетраэдры хранятся на протяжении всего счета, для них после каждого счетного шага обновляются значения величин в вершинах. Следует отметить, что подобный подход удорожает алгоритмы с точки зрения используемой памяти, например для 1 шестигранной ячейки порождается 24 тетраэдра. Для шестиугольной призмы требуется 36 тетраэдров.

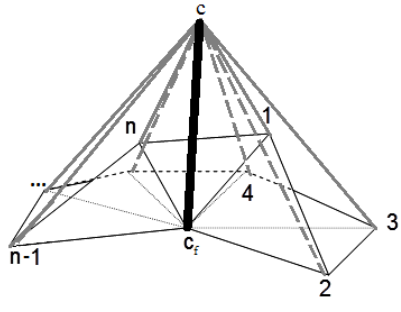


Рис. 3. Разбиение ячейки на тетраэдры со стороны одной грани

### 1.3. Расчет узловых величин

К узлам относятся величины скорости и координаты, а также массы узлов.
Расчет скоростей узлов в разностном виде выглядит следующим образом:

$$m_i \frac{\vec{U}_i^{n+1} - \vec{U}_i^{n}}{\tau} = \sum_{k \in Ш_i} \left( S_{k_1}^n \vec{n}_{k_1} + S_{k_2}^n \vec{n}_{k_2} \right)\left( P_{k_H}^n - P_{k_B}^n \right)$$

При этом нормали направлены из ячейки «$H$» в ячейку «$B$»; $Ш_i$ – шаблон-указатель связей узлов, ячеек и граней для узла $i$; $r$ – шаг по времени; $S_{k_1}^n \vec{n}_{k_1} + S_{k_2}^n \vec{n}_{k_2}$ – сумма площадей, умноженных на единичные нормали к этим площадям, $P_{k_H}^n - P_{k_B}^n$ – разность давлений в соседних ячейках,

разделенных $S^n_{k_1}\vec{n}_{k_1} + S^n_{k_2}\vec{n}_{k_2}$. Площадь $S^n_{k_1}$ является треугольным фрагментом грани, разделяющим ячейки «*H*» и «*B*» и образуется узлом *V*, центром грани *c*, и точкой $V'$ - серединой ребра $VV_1$, $\vec{n}_{k_1}$ – единичный вектор нормали к этой площади, аналогично определяется площадь $S^n_{k_2}$ для второго соседнего узла $V_2$. Узлы $V_1$, $V$, $V_2$ являются подряд идущими в грани разделяющей ячейки «*H*» и «*B*» против часовой стрелки со стороны ячейки «*H*» (см. рис. 2).

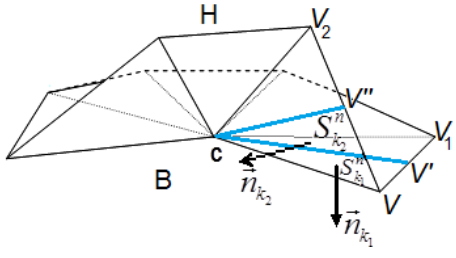


Рис. 4. Определение поверхности, примыкающей к узлу для уравнения движения

Необходимость представления границы раздела двух ячеек в виде суммы двух составляющих обусловлена тем, что грань между двумя многогранниками может быть неплоской. Поэтому в сумме участвуют половины двух треугольников триангуляции грани, примыкающих к узлу.

Положение узлов в пространстве на следующий момент времени интегрирования определяется по формуле:

$$\vec{R}_j^{n+1} = \vec{R}_j^n + \vec{U}_j^{n+1}\tau \tag{2}$$

**Учет внешних границ**

На этапе расчета узловых величин производится учет внешних границ задачи. В данной работе рассмотрим два основных типа внешних границ для расчета уравнений газовой динамики:

- свободная граница, на которой может быть задано внешнее давление;
- плоская жесткая стенка, определяемая вектором нормали.

Учет границ осуществляется следующим образом.

Для узла, лежащего на свободной границе, для соответствующих граней вместо соответствующего давления из ячейки $P^n_{k_H}$ или $P^n_{k_B}$ (см. (1)) используется давление, заданное на соответствующей свободной границе.

Для узлов на жесткой стенке производится корректировка скорости по формуле:

$$\vec{v} = \vec{v} - (\vec{v}, \vec{n})\vec{n} \tag{3}$$

где $\vec{n}$ – вектор внешней нормали к жесткой стенке. Для узлов на угловой линии, образованной двумя жесткими стенками корректировка скорости производится по формуле:

$$\vec{v} = \vec{v} - (\vec{v}, \vec{n}_1)\vec{n}_1 - (\vec{v}, \vec{n}_2)\vec{n}_2 \tag{4}$$

где $\vec{n}_1, \vec{n}_2$ – вектора внешних нормалей к жестким стенкам. Для угловых узлов, в которых сходится 3 (и более) жестких стенок скорость задается равной 0:

$$\vec{v} = 0 \tag{5}$$

Отметим, что граничный узел может лежать на угловой линии, образованной свободной границей и жесткой стенкой, также могут быть угловые узлы, в которых сходится три грани с двух жестких стенок и свободной границы. Такие узлы рассчитываются по алгоритмам, соответствующим свободной границе, а затем производится корректировка скорости по количеству жестких стенок, сходящихся в узле по формулам (4) – (6).

### 1.4. Расчет ячеечных величин

К ячейкам относятся массы ячеек, плотности, удельная внутренняя энергия и давление.

Плотность вычисляется из уравнения неразрывности:

$$\rho_j^{n+1} = \frac{M_j}{V_j^{n+1}}, \tag{6}$$

где $V_j$ – объем $j$-ой ячейки.

Уравнение сохранения энергии записывается в неявном виде:

$$\varepsilon_j^{n+1} = \varepsilon_j^n - p_j^{n+\frac{1}{2}} \left( \frac{1}{\rho_j^{n+1}} - \frac{1}{\rho_j^n} \right), \text{ где } p_j^{n+\frac{1}{2}} = \frac{p_j^{n+1} + p_j^n}{2} \tag{7}$$

Для разрешения неявности используется уравнение состояния, которое в разностном виде выглядит следующим образом.

$$p_j^{n+1} = p\left(\rho_j^{n+1}, \varepsilon_j^{n+1}\right) \tag{8}$$

Система уравнений (8) – (9) решается методом предиктор-корректор.

## 2. Векторные алгоритмы

Следует отметить, что в самой методике ТИМ-3D используется подход, в котором для одного узла или ячейки рассчитываются все уравнения. Данный способ хорошо зарекомендовал себя при использовании классических алгоритмов (как последовательных, так и параллельных), однако для векторизации этот подход подходит плохо, так как программный код содержит большое количество условных операций значительно снижающих производительность векторного устройства. При большом количестве условных операций выигрыша от векторизации может вообще не наблюдаться. Поэтому было решено изменить саму идеологию алгоритмов и перейти от обсчета всех уравнений для одного узла или ячейки к расчету одного уравнения для всех соответственно узлов или ячеек. При этом для достижения лучшей производительности часто приходится производить расщепление и самих уравнений на более простые.

Вторая проблема заключается в работе с неструктурированными сетками произвольного вида. Сами алгоритмы для неструктурированных сеток характеризуются нерегулярным шаблоном доступа к памяти, что приводит к существенной фрагментации обращений в поле памяти. По этой причине алгоритмы для неструктурированных сеток, как правило, плохо используют КЭШ-память.

Однако существует ряд работ по векторизации на неструктурированных тетраэдральных сетках (см. например [6]). Использование тетраэдров позволяет упорядочить работу с памятью, так как тетраэдр всегда содержит 4 вершины и 4 треугольных грани. Фиксированность позволяет разработать алгоритмы, использующие регулярный шаблон доступа к памяти и такие алгоритмы поддаются эффективной векторизации.

Основная идея векторизованных алгоритмов заключается во введении явного тетраэдрального представления сетки и работе с ним. При этом сама расчетная сетка остается произвольной, однако вместо неявной триангуляции граней и тетраэдрализации ячеек используется заранее подготовленная информация.

### 2.1. Процедура тетраэдрализации

После зачитывания исходной сетки и формирования топологической информации, производится тетраэдрализация ячеек. Для этого выполняется триангуляция каждой грани счетной ячейки, затем вычисляется центр ячейки. Таким образом, перебрав все грани ячейки и узлы каждой грани, формируется набор тетраэдров для ячейки.

Как было отмечено ранее, тетраэдры формируются заранее, и сохраняется на протяжении расчета. Тетраэдры хранятся в виде 4 массивов координат точек: 1 – центр ячейки, 2 – центр грани, 3,4 – узлы основной сетки (против часовой стрелки со стороны ячейки). Также для каждого тетраэдра сохраняется номер ячейки основной сетки, из которой он получен. Для всех массивов используется выравнивание по ширине векторных регистров (для Xeon Phi – 512 бит). Причем каждый набор координат по X, Y, Z хранится в своем массиве.

В блоке расчета узловых величин используются ориентированные площади треугольных фрагментов граней. При этом используется (см. рис. 2) сам узел, центр грани и середина ребра, соединяющего его с соседним узлом данной грани. То есть треугольник является половиной основания треугольной пирамиды, полученной при тетраэдрализации ячейки. Благодаря этому при расчете узловых величин можно использовать единый блок тетраэдрализации с расчетом ячеечных величин. Для этого достаточно вычислить площади оснований треугольных пирамид (2 – 4 узлы тетраэдров). При этом достаточно умножать данные площади на 0,5 в вычислении скоростей для получения корректных результатов. Использование единой тетраэдрализации позволяет уменьшить объем требуемой памяти и увеличить скорость работы.

Таким образом после того как набор тетраэдров сформирован для каждой ячейки счетной сетки, у каждого тетраэдра производится расчет площадей оснований.

### 2.2. Расщепление по тетраэдрам для расчета скорости

Уравнение (1) расщепляется по тетраэдрам: для каждого тетраэдра рассчитывается вклад в расчет силы, прикладываемой к паре узлов. Таким образом, расчет уравнения движения выглядит следующим образом:

$$m_i \frac{\vec{U}_i^{n+1} - \vec{U}_i^{n}}{\tau} = \sum_{k=1}^{l} S_k^n \vec{n}_k P_k^n \,, \tag{9}$$

где $l$ – количество тетраэдров окружающих узел. Таким образом, каждая внутренняя грань обсчитывается дважды – со стороны каждой ячейки. Необходимо отметить, что в формуле (10) объем вычислений по сравнению с исходной формулой (2) существенно возрастает, поскольку в (10) ориентированная площадь рассчитывается для каждого треугольника отдельно, в отличие от (2), где ориентированная площадь рассчитывается сразу для пары треугольников. Кроме того, в исходной схеме грань обрабатывается один раз, а в векторизованной схеме внутренние грани обсчитываются дважды – отдельно со стороны каждой ячейки, окружающей узел.

Итоговое уравнение для расчета скорости выглядит следующим образом:

$$\vec{U}_i^{n+1} = \vec{U}_i^{n} + \frac{\tau}{m_i} \sum_{k=1}^{l} S_k^n \vec{n}_k P_k^n \tag{10}$$

Кроме скорости производится расчет нового положения узлов по формуле (3).

### 2.3. Расчет внутренних узлов

Рассмотрим уравнение (11). Прямое программирование этого уравнения не позволит использовать векторизацию. Это объясняется использованием в уравнении величин, относящихся к разным элементам сетки: скорости и массы к узлам, площади фрагментов – к граням, примыкающим к узлу, давления – к ячейкам. Усложняет векторизацию и суммирование от 1 до $l$. Поэтому в таком виде уравнение (11) в векторных алгоритмах использовать нельзя. Необходимо выполнить расщепление на более мелкие операции по элементам сетки. Например, при вычислении площадей выполняется произведение двух векторов, при этом в качестве первого этапа вычисляется значение двух векторов, затем отдельно их произведение. Такая реализация оказалась более быстрой, чем в случае их объединения в более крупные блоки.

Расщепление уравнения (11) производится на следующие этапы:

1. Вычисление ориентированных площадей оснований тетраэдров $\vec{S}_k = S_k^n \vec{n}_k = \frac{1}{4}\left[\vec{R}_{V_1} - \vec{R}_f, \vec{R}_{V_2} - \vec{R}_f\right]$, где $\vec{R}_{V_1}, \vec{R}_{V_2}$ – координаты узлов, $\vec{R}_f$ – координата центра грани.

2. Вычисление вклада в силу, приложенную к паре узлов со стороны данного тетраэдра $\vec{f}_k = \vec{S}_k P_k^n$.

3. Суммирование для узла вкладов сил из окружающих его тетраэдров $\vec{F}_i = \sum_{k=1}^{l} \vec{f}_k$

4. Вычисление ускорений для узлов сетки: $\vec{a}_i = \frac{\vec{F}_i}{m}$.

5. Вычисление новых скоростей для узлов сетки: $\vec{U}_i^{n+1} = \vec{U}_i^n + \tau \vec{a}_i$.

6. Вычисление новых координат узлов сетки: $\vec{R}_i^{n+1} = \vec{R}_i^n + \vec{U}_i^{n+1}\tau$.

В описанной последовательности операций шаги 1 – 2 делаются по тетраэдрам, шаги 3 – 6 по узлам.

### 2.4. Расчет узлов на внешних границах

Обсчет граничных узлов происходит особым образом. На первом этапе определяется, к какому типу относится узел: внутренний, или граничный. Если узел внутренний происходит расчет уравнений для данного узла, если же узел - граничный, то происходит определение, к какому типу граничных условий относится данный узел: жесткая стенка или свободная граница. В зависимости от условия происходит расчет узла с учетом того или иного типа граничных условий. Данный подход, с точки зрения векторизации неэффективен, так как обилие логических операторов при анализе принадлежности узла к граничным или внутренним, приводит к замедлению работы векторного алгоритма. Поэтому было решено в рамках векторных алгоритмов переработать данный подход, для того чтобы прийти к единообразию, с минимальным набором логики и чтобы можно было рассчитывать как граничные, так и внутренние узлы векторно.

Расчет узлов на свободных границах

Для того чтобы учесть давление со стороны свободной границы, для каждого треугольника на свободной границе вводится фиктивный тетраэдр. В качестве центра ячейки у таких тетраэдров выступает фиктивная точка (с координатами равными «$-\infty$»). А в качестве давления – соответствующее значение на свободной границе. Фиктивные тетраэдры не относятся ни к одной ячейке, но приписываются к паре узлов также как и внутренние тетраэдры. Таким образом, фиктивные тетраэдры участвуют в расчете уравнения движения (11), но не участвуют в расчете объемов ячеек. Такая схема позволяет с одной стороны производить корректный расчет свободных границ, с другой стороны позволяет использовать единый алгоритм расчета тетраэдров в уравнении движения без дополнительных ветвлений в алгоритмах.

Расчет узлов на жестких стенках

На жестких стенках производится корректировка вектора скорости – убирается его нормальная составляющая:

$$\left(\vec{U}_j^{n+1}\right)' = \begin{cases} \vec{U}_j^{n+1}, & \text{если } p = 0 \\ \vec{U}_j^{n+1} - \vec{n}_1\left(\vec{U}_j^{n+1}, \vec{n}_1\right), & \text{если } p = 1 \\ \vec{U}_j^{n+1} - \vec{n}_1\left(\vec{U}_j^{n+1}, \vec{n}_1\right) - \vec{n}_2\left(\vec{U}_j^{n+1}, \vec{n}_2\right), & \text{если } p = 2 \\ 0, & \text{если } p \geq 3 \end{cases}, \quad (11)$$

где $\vec{n}_1, \vec{n}_2$ – вектора единичных нормалей к соответствующей жесткой стенке, а $p$ – количество жестких стенок для узла, 0 – для внутреннего узла, 1 – на одной жесткой стенке, 2 – на угловой линии, образованной двумя жесткими стенками, 3 – угловой узел, образованный тремя (и более) жесткими стенками. Отметим, что наличие свободной границы не влияет на корректировку

скорости. То есть для узлов на угловых линиях, образованных свободной границей и жесткой стенкой $p = 1$, а для угловых узлов образованных двумя жесткими стенками и свободной границей $p = 2$. При этом для таких узлов учет свободной границы уже произведен до корректировки скорости на жестких стенках.

Формула (12) для векторизации оказалась плохо подходящей. Связано это с наличием дополнительных условий, снижающих эффективность работы векторного вычислительного устройства.

В связи с этим данная формула была переработана. Поскольку на жесткой стенке убирается нормальная составляющая вектора скорости, то остается тангенциальная составляющая. Таким образом, правую часть третьего уравнения формулы (12) можно записать в виде

$$\vec{U}_j^{n+1} - \vec{n}_1\left(\vec{U}_j^{n+1}, \vec{n}_1\right) - \vec{n}_2\left(\vec{U}_j^{n+1}, \vec{n}_2\right) = \vec{\vartheta}_j\left(\vec{U}_j^{n+1}, \vec{\vartheta}_j\right),$$

где $\vec{\vartheta}_j = \left[\vec{n}_1 \times \vec{n}_2\right]$ – единичный вектор направления угловой линии.

Таким образом, положив:

$$a = \left[\left|2p'_j - 1\right| \bmod 3\right] \qquad \vec{\eta}_j = \begin{cases} 0, & \text{если } p_j = 0, p_j \geq 3 \\ \vec{n}_1, & \text{если } p_j = 1 \\ \left[\vec{n}_1 \times \vec{n}_2\right], & \text{если } p_j = 2 \end{cases},$$

$$b = \left[\left(2p'_j - 3\right)\min\left(p'_j, 1\right)\right]$$

где $p' = \left[p - 4\left\lfloor \frac{\min(p,3)}{3} \right\rfloor\right]$, получим единую формулу для вычисления корректировки скорости на жесткой стенке:

$$\left(\vec{U}_j^{n+1}\right)' = \left[\left|2p'_j - 1\right| \bmod 3\right] \cdot \vec{U}_j^{n+1} + \left[\left(2p'_j - 3\right)\min\left(p'_j, 1\right)\right] \cdot \vec{\eta}_j\left(\vec{U}_j^{n+1}, \vec{\eta}_j\right)$$

Здесь выражения в полуквадратных скобках обозначают целую часть, т. е. например, $\left\lfloor \frac{5}{3} \right\rfloor = 1$;

a mod – операция «остаток от деления».

Данная формула подходит для векторизации и не требует условных операторов на стадии вычислений. Значения $p_j$ и $\vec{\eta}_j$ для каждого из узлов в процессе счета не меняются (для плоских жестких стенок) и поэтому их расчет производится один раз при подготовке данных.

Следует отметить, что при программной реализации данной формулы возник ряд сложностей. Признак $p_j$ имеет целый тип, тогда как остальные данные представлены вещественным типом с двойной точностью. На сопроцессорах Intel Xeon Phi подобное смешение типов в векторной операции приводит к замедлению, так как компилятор для подобных конструкций начинает генерировать инструкции преобразования типов. Существует несколько путей решения данной проблемы: во-первых, можно явно использовать низкоуровневую «intrinsic» функцию dble, чтобы явно преобразовать типы данных, используемые в данных вычислениях. Однако такой подход приводит к проявлению новой проблемы – возникновению инструкций сборки и распределения (gather/scatter), которые осуществляют сбор и запись данных в памяти с непрямым доступом. Поэтому массив признаков $p_j$ описан как вещественный с двойной точностью, что решает все вышеперечисленные проблемы, однако приводит к увеличению используемой памяти.

Отметим, что операции получения минимума, остатка от деления и модуля, используемые в данном алгоритме, также берутся из встроенных функций, для которых имеется векторная реализация в компиляторах.

## 3. Расчет термодинамических величин

Как было отмечено ранее, в блоке расчета термодинамических величин для вычисления объемов так же используется процедура тетраэдрализации ячеек, точнее набор тетраэдров, сформированный на подготовительном этапе.

Вычисляется объем каждого тетраэдра, а затем объем каждой ячейки, за счет редуцирования объемов тетраэдров, полученных при тетраэдрализации соответствующей ячейки.

В блоке расчета ячеечных величин алгоритмы реализованы в схожем ключе с блоком расчета узловых величин. То есть вместо расчета всех уравнений для одной ячейки, происходит вычисление одного уравнения для всех ячеек. Подобная организация алгоритмов позволяет векторизовать производимые вычисления. В следующем примере продемонстрирован подход с реорганизацией алгоритмов:

```
!DEC$IF DEFINED (__MIC__)
 !dir$ simd vectorlength(8)
!DEC$ ELSE
 !dir$ simd vectorlength(4)
!DEC$ ENDIF
 !dir$ vector aligned nontemporal(RO)
!DEC$IF DEFINED (__MIC__)
 !dir$ prefetch * : 1 : 32
 !dir$ prefetch * : 0 : 8
 !dir$ noprefetch RO
!DEC$ ENDIF
  do iCell = 1,MathAreas(nArea)%kAreaCells
    RO(iCell) = dMass(iCell) / dVolume(iCell)
    enddo
```

В представленном листинге видно, что, благодаря директивам условной компиляции, регулируется ширина вектора, для которой выполняется последующий цикл. Далее используется директива «aligned», которая указывает компилятору, что данные выровнены в памяти. Так же следует отметить использование директив «prefetch». Использование подобного инструмента позволяет управлять предвыборкой данных в кэш-память. Сначала указывается список массивов, для которых будет осуществляться предвыборка, в случае, представленном в листинге, указано значение «*», что означает предвыборку для всех переменных. Далее после двоеточия, указан уровень кэш-памяти, в которую будет происходить загрузка. Под цифрой 1 – подразумевается кэш L2 размер, которого для данного сопроцессора составляет 512 кб, а под цифрой 0 – кэш L1, который состоит из двух частей 32 кб кэш-инструкций и 32 кб кэш данных. Далее после двоеточия указывается количество итераций, которые должны быть подгружены в память. В данном примере было решено подгружать в кэш L2 по 512 байт, т. е. 32 итерации по 2 элемента, каждый из которых по 8 байт, в результате получаем 512. В кэш нулевого уровня данные подгружаем в меньшем объеме.

В качестве уравнения состояния было использовано уравнение состояния для идеального газа, для которого был реализован векторный вариант.

## 4. Результаты тестовых расчетов

Все тестовые расчеты были проведены на вычислительной системе «Каскад» [5]. Расчеты «Без векторизации» осуществлялись кодом, собранным с ключами –qno-vec, -qno-simd, -qno-openmp-simd. Использование подобных ключей позволяет отключить все векторные оптимизации компилятора, что позволяет оценить коэффициент ускорения от векторизации.

В качестве тестовой задачи была выбрана задача о распространении плоской волны. Число ячеек составляло миллион, использовалась шестигранная (см. рис. 5 слева). Для получения временных замеров и коэффициентов ускорения от векторизации замерялось время счета 10 шагов.

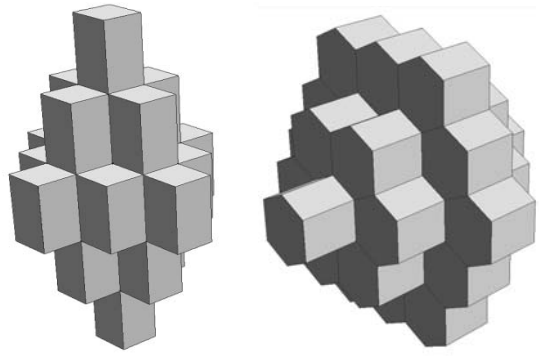

Рис. 5. Примеры сеток, на которых проводились тестовые расчеты

Запуск производился на 1 ядре сопроцессора Intel Xeon Phi. Полученные данные для сопроцессоров на шестигранной сетке представлены в табл. 1.

В табл. 1 представлены замеры времени для векторизованных блоков расчета узловых величин и термодинамических величин для 10 шагов.

Таблица 1

Значения полученных ускорений от векторизации для сопроцессоров Intel Xeon Phi на шестигранной сетке из миллиона ячеек

| Блок | Без векторизации, с | С векторизацией, с | Ускорение, раз. |
|---|---|---|---|
| Расчет узловых величин | 46,995 | 24,885 | 1,9 |
| Расчет термодинамических величин | 91,012 | 27,987 | 3,25 |
| Общее время расчета уравнений газодинамики | 138,007 | 52,872 | 2,61 |

Общее ускорение векторизованных блоков составило чуть более 2,6 раз, что является удовлетворительным показателем для вещественной арифметики с двойной точностью. Дальнейшая оптимизация алгоритмов позволит улучшить показатели ускорения от векторизации.

Современные универсальные процессоры также являются векторными вычислительными устройствами, поэтому замеры были выполнены и для них. Результаты замеров приведены в табл. 2.

Таблица 2

Значения полученных ускорений от векторизации для универсальных процессоров на шестигранной сетке из миллиона ячеек

| Блок | Без векторизации, с | С векторизацией, с | Ускорение, раз. |
|---|---|---|---|
| Расчет узловых величин | 5,018 | 4,756 | 1,05 |
| Расчет термодинамических величин | 6,981 | 6,438 | 1,08 |
| Общее время расчета уравнений газодинамики | 11,999 | 11,194 | 1,07 |

В качестве второй тестовой задачи была выбрана задача о распространении плоской волны, но на сетке из шестиугольных призм. Число ячеек составило 500 тысяч (см. рис. 5 справа). Для получения коэффициентов ускорения от векторизации был сделан замер времени расчета 10 шагов. Следует отметить, что число точек в данном тесте пришлось уменьшить так как по сравнению с шестигранной сеткой число тетраэдров на одну ячейку возрастает в 1,5 раза, следовательно и затраты памяти на сопроцессоре для хранения данной сетки возрастают. Полученные данные для Intel Xeon Phi представлены в табл. 3.

Таблица 3

Значения полученных ускорений от векторизации для сопроцессоров Intel Xeon Phi на сетке из шестиугольных призм на 500 тысяч ячеек

| Блок | Без векторизации, с. | С векторизацией, с. | Ускорение, раз. |
|---|---|---|---|
| Расчет узловых величин | 45,135 | 26,763 | 1,68 |
| Расчет термодинамических величин | 62,109 | 15,789 | 3,93 |
| Общее время расчета уравнений газодинамики | 107,224 | 42,552 | 2,52 |

В табл. 4 представлены данные по тестированию векторных алгоритмов на сетке из шестиугольных призм для универсальных процессоров.

Таблица 4

Значения полученных ускорений от векторизации для универсальных процессоров на сетке из шестиугольных призм на 500 тысяч ячеек

| Блок | Без векторизации, с. | С векторизацией, с. | Ускорение, раз. |
|---|---|---|---|
| Расчет узловых величин | 4,366 | 4,09 | 1,07 |
| Расчет термодинамических величин | 4,937 | 4,456 | 1,11 |
| Общее время расчета уравнений газодинамики | 9,303 | 8,546 | 1,09 |

Сравнивая результаты на разных типах сеток, приведенные в таблицах 1, 2 и 3, 4 можно сделать вывод о том, что на сетке из шестиугольных призм ускорение оказывается несколько ниже по сравнению с расчетом на шестигранной сетке. Это можно объяснить существенно возрастающим количеством тетраэдров для восьмигранных ячеек в результате время, затраченное на редукцию при вычислении объемов, увеличивается. Коэффициент ускорения для данного алгоритма дает невысокое значение, что приводит к некоторому снижению и общего показателя. В алгоритме редукции при вычислении объема ячейки используется механизм gather/scatter, который позволяет осуществить выборку тетраэдров из списка для текущей ячейки, что приводит к произвольному доступу к памяти. Произвольный доступ к памяти в векторных алгоритмах существенно влияет на коэффициент ускорения, что и наблюдается в представленных таблицах. В дальнейшем предполагается выполнить доработку и оптимизацию алгоритмов расчета объемов, с целью сокращения времени данного блока.

## Заключение

В работе представлено описание векторизованных алгоритмов для одной из разностных схем расчета уравнений газовой динамики методом конечных разностей на произвольных неструктурированных многогранных лагранжевых сетках в методике ТИМ-3D. Полученные результаты расчетов по векторным алгоритмам согласуются с результатами расчетов по скалярным алгоритмам с точностью до машинной арифметики (отличие в 12 знаке).

В работе представлен анализ проблем, возникающих при векторизации алгоритмов, сделан анализ разностной схемы с точки зрения векторизации и предложен подход на основе явного хранения триангуляции граней и тетраэдрализации ячеек.

Общее ускорение от векторизации алгоритмов составило около 2,5 раз для сопроцессора Intel Xeon Phi, и 1,1 для универсального процессора.

В дальнейшем планируется разработать и реализовать векторные алгоритмы для других разностных схем и приближений методики ТИМ-3D. Оптимизировать процедуру вычисления объемов, для увеличения коэффициента ускорения от векторизации. Также разработанные алгоритмы планируется расширить для использования на множестве ядер и вычислительных узлов в смешанной модели памяти, а также с использованием различных уравнений состояний и приближений.

## Литература

1. Соколов С. С., Панов А. И., Воропинов А. А., и др.: Методика ТИМ расчета трехмерных задач механики сплошных сред на неструктурированных многогранных сетках // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. – 2005. – Вып. 3. – С. 37 – 52.

2. Голомидов Ф. О., Воропинов А. А., Новиков И. Г., Развитие OpenMP распараллеливания в методике ТИМ // XVI Международная конференция «Супервычисления и математическое моделирование»: сб. науч. тр. / под ред. Р. М. Шагалиева. – Саров: ФГУП «РФЯЦ-ВНИИЭФ», 2017. – С. 100 – 102.

3. «Центр компетенций и обучения». Вычислительная система «Каскад» [Электронный ресурс]: Режим доступа: http://www.compcenter.org

4. Intel Xeon Phi coprocessor [Electronic resource]. Mode of access: http://www.intel.com.

5. Соколов С. С. Методика решения нестационарных упругопластических задач на нерегулярных многогранных лагранжевых сетках // Вопросы атомной науки и техники. Серия: Математическое моделирование физических процессов. – 2002. – Вып. 4. – С. 23 – 36.

6. Горобец А. В., Сков С. А., Богданов П. Б., На пути к освоению гетерогенных супервычислений в газовой динамике // Информационные технологии и вычислительные системы. – 2013. – Вып. 4. – С. 15 – 26.

## VECTORIZED ALGORITHMS FOR ONE OF THE DIFFERENCE SCHEME OF GAS DYNAMICS ON UNSTRUCTURED POLYHEDRAL MESHES IN LAGRANGIAN TIM-3D METHOD

*F. O. Golomidov, A. A. Voropinov*

Russian Federal Nuclear Center –
All-Russian Research Institute of Experimental Physics, Sarov

This paper describes vectorized algorithms for calculation of gas dynamics equations on unstructured Lagrangian polyhedral meshes used in TIM-3D method. Besides, one of the difference schemes used in TIM method that realizes its standard scalar algorithms is described. This paper shows the

transformation of calculation blocks for the difference scheme vectorization. The main idea of the vectorization approach is based on tetrahedralization of the cells of the initial mesh. This approach makes vectorization on arbitrary meshes possible. Testing of vectorized algorithms shows that the computation results produced by two algorithms (the standard one used in TIM method and the vectorized one) are the same, and the speedup from the vectorization obtained on Intel Xeon Phi coprocessor is ~2.5 times, and it is ~1.1 times on Intel Haswell central processing unit.

*Keywords*: unstructured Lagrangian polyhedral mesh, vectorization, Lagrangian gas dynamics, Intel Xeon Phi.

УДК 004.94

# КРОСС-ВЕРИФИКАЦИЯ ПАКЕТА ПРОГРАММ ЛОГОС НА ПОЛНОМАСШТАБНЫХ РАСЧЕТАХ СМЕШЕНИЯ НЕИЗОТЕРМИЧЕСКИХ ПОТОКОВ ТЕПЛОНОСИТЕЛЯ В НАПОРНОЙ КАМЕРЕ СУДОВОЙ РУ

*Е. А. Данилов*[1], *А. А. Деулин*[1], *О. В. Денисова*[1], *В. В. Курулин*[1], *О. О. Шестак*[1], *Е. В. Глазунова*[1], *Д. Н. Свешников*[2]

[1] Российский федеральный ядерный центр – Всероссийский НИИ экспериментальной физики, Саров
[2] АО «ОКБМ Африкантов», Нижний Новгород

В настоящей работе представлены результаты моделирования смешения неизотермических потоков в напорной камере РУ РИТМ 200 при отключении парогенератора, а также представлено сравнение результатов моделирования по пакету программ ЛОГОС с данными, полученными с использованием коммерческого CFD-кода ANSYS CFX. Постановка задачи, исходная геометрия и результаты расчетов по ANSYS CFX для выполнения кросс-верификации предоставлены АО «ОКБМ Африкантов». В расчетах рассмотрены различные варианты компоновки оборудования напорной камеры для обоснования конструкторского решения по установке смесительного оборудования.

Сравнительный анализ результатов моделирования, выполненного с использованием пакетов ЛОГОС и ANSYS CFX, продемонстрировал их удовлетворительное согласие по всем оцениваемым параметрам.

*Ключевые слова*: пакет программ ЛОГОС, кросс-верификация, судовая РУ, РИТМ 200, турбулентное течение, смешение неизотермических потов, турбулентная конвекция, CFD-коды, напорная камера, Ansys CFX.

## Введение

Одним из актуальных направлений применения пакетов программ трехмерного теплогидравлического расчета (CFD-кодов) является их использование при обосновании режимов эксплуатации реакторных установок (РУ) при парциальном составе работающего оборудования (при отключении парогенератора, секции парогенератора и т. п.).