УДК 519.688

Параллельный алгоритм для полунеявного метода частиц в ячейках с сохранением энергии и заряда*

Е.А. Берендеев[1,2], И.В. Тимофеев[1,2]

[1]Новосибирский национальный исследовательский государственный университет (НГУ), ул. Пирогова, 1, Новосибирск, 630090

[2]Институт ядерной физики им. Г.И. Будкера СО РАН, просп. Акад. Лаврентьева, 11, Новосибирск, 630090

E-mail: beren@inp.nsk.su (Берендеев Е.А.)

Английская версия этой статьи печатается в журнале "Numerical Analysis and Applications" № 4, Vol. 17, 2024.

Берендеев Е.А., Тимофеев И.В. Параллельный алгоритм для полунеявного метода частиц в ячейках с сохранением энергии и заряда // Сиб. журн. вычисл. математики / РАН. Сиб. отд-ние. — Новосибирск, 2024. — Т. 27, № 4. — С. 365–378.

Статья посвящена вопросам построения параллельного алгоритма для расчёта динамики плазмы методом частиц в ячейках с использованием полунеявной схемы, сохраняющей энергию и заряд. Данная схема представляет собой двухстадийный предиктор–корректор, где на этапе предсказания используется полунеявный метод Лапенты, в котором сохраняющий энергию линейный ток не удовлетворяет локальному закону Гаусса, а на этапе коррекции токи, электромагнитные поля и скорости частиц подправляются так, чтобы разностные законы сохранения энергии и заряда выполнялись точно. Этот подход оказывается эффективным для моделирования разномасштабных явлений с достаточно большим временным шагом, однако является ресурсоёмким, поскольку требует не только решения двух систем линейных уравнений за шаг, но и перестроения всей матрицы системы. Авторами разработан матрично-операторный алгоритм для программной реализации этой схемы, позволяющий эффективно распараллелить вычисления, а также использовать различные библиотеки для работы с матрицами и решателями систем линейных уравнений. Для построения матрицы использован алгоритм построчного хранения с поиском элементов через хэш-таблицу, что уменьшает объём используемой памяти, число синхронизаций потоков и позволяет существенно ускорить вычисления. Рассматриваемый алгоритм успешно применён в коде Beren3D.

DOI: 10.15372/SJNM20240401
EDN: MIQMZD

Ключевые слова: *параллельный алгоритм, метод частиц в ячейках, решение систем линейных алгебраических уравнений, высокопроизводительные вычисления.*

Berendeev E.A., Timofeev I.V. Parallel algorithm for semi-implicit particle-in-cell method with energy and charge conservation // Siberian J. Num. Math. / Sib. Branch of Russ. Acad. of Sci. — Novosibirsk, 2024. — Vol. 27, № 4. — P. 365–378.

The article is devoted to the construction of a parallel algorithm for calculating plasma dynamics by the particle-in-cell method using a semi-implicit scheme that conserves energy and charge. This is a two-stage predictor–corrector scheme. At the prediction stage a semi-implicit Lapenta-type method is used in which an energy-conserving linear current does not satisfy the local Gauss law. At the correction stage the currents, electromagnetic fields, and particle velocities are corrected so that difference laws of energy and charge conservation are satisfied exactly. This approach turns out to be efficient in modeling of multi-scale phenomena with a sufficiently large time step. However, the method is computer time-consuming, since it requires not only solving two systems of linear equations per step, but also reconstructing the entire matrix

*Исследование выполнено за счет гранта РНФ (проект № 21-72-10071).

of the system. The authors have developed a matrix-operator software implementation algorithm for this scheme, which allows efficient paralleling of the calculations and using the various available libraries for work with matrices and solvers for systems of linear equations. To construct the matrix, a row-by-row storage algorithm is used with search for the elements via a hash table, which reduces the memory capacity required, the number of thread synchronizations, and can significantly speed up the calculations. This algorithm has been successfully applied in a computer code, Beren3D.

Keywords: *parallel algorithm, particle-in-cell method, solving systems of linear algebraic equations, high-performance computing.*

Введение

Основной проблемой при численном моделировании установок для управляемого термоядерного синтеза является огромный динамический диапазон между микро- и макромасштабами исследуемых процессов. Наиболее подробное кинетическое описание плазмы можно получить, используя модели частиц в ячейках (PIC), требующие разрешения ларморовского вращения электронов или их быстрых колебаний на плазменной частоте на протяжении всего времени эксперимента. Использование стандартных явных схем здесь ограничивается не только малым шагом по времени, который посредством условия Куранта–Фридрихса–Леви привязан к пространственному шагу, обычно разрешающему дебаевский радиус в плазме, но и накоплением со временем численных ошибок, связанных с неточным сохранением энергии. В последние годы появилось множество неявных схем интегрирования системы уравнений Власова–Максвелла, обладающих хорошей устойчивостью даже при больших шагах по времени. Их можно разделить на две группы: полностью неявные [1, 2] и полунеявные схемы [3–8]. В первом случае уравнения движения частиц и уравнения Максвелла для полей решаются совместно с помощью нелинейных итераций Ньютона–Крылова, а энергия системы может сохраняться с любой наперёд заданной точностью. В полунеявном подходе вычислительный цикл строится так же, как и в явных PIC-схемах (particle-in-cell или PIC), а отклик частиц на поле в будущем учитывается в уравнениях Максвелла через линейный ток. В то время как в прямом неявном методе (Direct Implicit Method или DIM) [3, 4] и неявном методе моментов (Implicit Moment Method или IMM) [5–7] отклик частиц является линейным только приближённо, что приводит к несохранению энергии, в полунеявном методе Лапенты (Energy Conserving Semi-Implicit Method или ECSIM) [8] линейность тока не является результатом каких-либо приближений, что позволяет точно сохранить энергию на дискретных шагах по времени.

Стоит отметить, что возможность точного сохранения энергии весьма важна для моделирования столкновительных эффектов в плазме, поскольку это позволяет сохранить энергию, когда PIC-алгоритмы работают совместно с алгоритмом кулоновских столкновений Монте-Карло [9]. Очевидно, что из-за отсутствия нелинейных итераций полунеявные схемы оказываются в несколько раз более эффективными с вычислительной точки зрения, чем полностью неявные. Однако даже при линейном отклике среды конечный размер частиц делает его нелокальным, что сводит решение уравнений Максвелла к обращению недиагональной разреженной матрицы на каждом шаге по времени, что является весьма трудоёмкой задачей. Более того, заполнение этой матрицы необходимо производить при каждом решении уравнения Максвелла, поскольку значения и положения элементов определяются скоростями и координатами частиц. Также стоит заметить, что выбранный в оригинальном методе ECSIM способ вычисления тока несовместим с выполнением уравнения непрерывности, а значит, и закона Гаусса, что

приводит к повышенному уровню электростатических шумов. Важный шаг по исправлению этого изъяна был сделан в работе [10], где одновременное сохранение энергии и заряда было предложено достигать за счёт использования дополнительного этапа коррекции, на котором удовлетворяющий закону Гаусса ток вычислялся по предсказанным положениям частиц. В этой работе пространственная дискретизация была основана на методе конечных элементов, а коррекции подвергалось только электрическое поле, что, конечно, не обеспечивало точного сохранения энергии из-за игнорирования вклада электромагнитных флуктуаций. Кроме того, оставалось неясным, как этот метод применить к практически важному случаю, когда токи в плазме создаются сразу несколькими сортами частиц.

В работе [11] мы переформулировали предложенную в [10] схему предиктор–корректор для стандартной пространственной сетки Йи, существенно модифицируя корректирующий шаг. В нашем варианте предсказание неизвестных величин на новом временном шаге основано на сохраняющем энергию методе ECSIM [8], а сохранение заряда обеспечивается переходом к току, вычисляемому напрямую из уравнения непрерывности с помощью метода декомпозиции плотности Есиркепова [12]. В отличие от работы [10], коррекции подвергается не только электрическое, но и магнитное поле, а сам метод легко обобщается на случай генерации тока несколькими сортами частиц.

Эта схема была реализована в коде Beren3D для моделирования динамики плазмы в открытых ловушках с высоким β; в работе [11] с помощью этого кода на примере задачи вейбелевской неустойчивости было показано сохранение полной энергии системы до 10^{-13}. Однако в процессе реализации данной модели мы столкнулись с тем, что заполнение разреженной матрицы для решения системы уравнений Максвелла затрудняет использование параллельных алгоритмов, обычно применяющихся при реализации метода частиц. Действительно, представление разреженной матрицы, как правило, включает в себя несколько массивов, один из которых хранит значения ненулевых элементов, а другие — их положение в матрице. Поскольку элементы матрицы определяются координатами и скоростями модельных частиц, то при параллельной обработке частиц возникает необходимость одновременного обновления всех массивов, представляющих матрицу. Это приводит к необходимости синхронизации большого объёма данных, так как каждая частица вносит вклад сразу в несколько элементов. Таким образом, построение матрицы занимает основную часть вычислений. Например, в работе [13] отмечено, что построение подобной матрицы занимает 90 % всего времени расчёта (в 10 раз больше, чем перемещение частиц, и в 30 раз больше, чем решение СЛАУ). В данной статье мы разработали параллельный алгоритм, связывающий частицы с матрицей на вычислительной сетке таким образом, чтобы избежать конкуренции данных и лишних синхронизаций в матрице. Также нами был предложен специальный алгоритм хранения, поиска и вставки элементов, основанный на неупорядоченном словаре, который позволяет существенно ускорить заполнение матрицы. А поскольку в разработанной нами схеме с сохранением энергии и заряда необходимо решение 2-х СЛАУ, мы предложили переписать большую часть вычислений в матрично-векторном виде, что позволило свести основные параллельные вычисления к умножению матрицы на вектор и сложению векторов, а также использовать стандартные библиотеки для работы с линейной алгеброй.

1. Численная модель

Сформулируем сначала предложенную в [11] полунеявную PIC-модель, а затем перепишем её в виде последовательного алгоритма. На первом этапе мы предсказываем

неизвестные значения координат $\mathbf{x}_p^{n+1}$ и скоростей $\tilde{\mathbf{v}}_p^{n+1}$ частиц, как и значения электромагнитных (ЭМ) полей $\tilde{\mathbf{E}}_g^{n+1}$ и $\tilde{\mathbf{B}}_g^{n+1}$ на сетке на следующем временном шаге $t_{n+1} = (n+1)\Delta t$, используя полунеявную ECSIM-схему [8], которая сохраняет энергию, но не сохраняет заряд. Затем, на шаге коррекции, ток $\tilde{\mathbf{J}}_p^{n+1}$, который не удовлетворяет уравнению непрерывности, заменим на ток $\mathbf{J}_p^{n+1}$, рассчитанный по схеме Есиркепова [12]. После этого скорректируем поля $\tilde{\mathbf{E}}_p^{n+1}$, $\tilde{\mathbf{B}}_p^{n+1} \to \mathbf{E}_p^{n+1}$, $\mathbf{B}_p^{n+1}$ и проведём локальную перенормировку скоростей $\tilde{\mathbf{v}}_p^{n+1} \to \mathbf{v}_p^{n+1}$ так, чтобы восстановить закон сохранения энергии.

На шаге предсказания для движения частиц мы используем следующую конечно-разностную схему:

$$\mathbf{x}_p^{n+1/2} = \mathbf{x}_p^n + \frac{\Delta t}{2}\mathbf{v}_p^n, \tag{1}$$

$$\tilde{\mathbf{v}}_p^{n+1} = \mathbf{v}_p^n + \frac{q_p \Delta t}{m_p}\left(\tilde{\mathbf{E}}_p^{n+1/2} + \left[\tilde{\mathbf{v}}_p^{n+1/2} \times \mathbf{B}_p^n\right]\right), \tag{2}$$

где скорость $\tilde{\mathbf{v}}_p^{n+1/2} = (\mathbf{v}_p^n + \tilde{\mathbf{v}}_p^{n+1})/2$ представляет собой полусумму значений скоростей на старом и новом шагах, а электромагнитные поля, действующие на макрочастицу с форм-фактором W, определяются с помощью интерполяции на сетку:

$$\tilde{\mathbf{E}}_p^{n+1/2}\big(\mathbf{x}_p^{n+1/2}\big) = \sum_g \left(\frac{\mathbf{E}_g^n + \tilde{\mathbf{E}}_g^{n+1}}{2}\right) W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g\big), \tag{3}$$

$$\mathbf{B}_p^n\big(\mathbf{x}_p^{n+1/2}\big) = \sum_g \mathbf{B}_g^n W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g\big). \tag{4}$$

Здесь и далее время измеряется в обратных плазменных частотах электронов ($\omega_p = \sqrt{4\pi e^2 n_0/m_e}$), скорости и координаты частиц — в скоростях света c и c/ω_p, заряд q_p и масса m_p частиц — в единицах заряда e и массы m_e электрона, поля измеряются в единицах $m_e c\omega_p/e$, плотность частиц и тока — в единицах n_0 и en_0c. Для решения уравнений Максвелла в конечно-разностной форме

$$\tilde{\mathbf{B}}_g^{n+1} = \mathbf{B}_g^n - \Delta t\left(\operatorname{rot}\tilde{\mathbf{E}}^{n+1/2}\right)_g, \tag{5}$$

$$\tilde{\mathbf{E}}_g^{n+1} = \mathbf{E}_g^n - \Delta t\tilde{\mathbf{J}}_g^{n+1/2} + \Delta t\left(\operatorname{rot}\tilde{\mathbf{B}}^{n+1/2}\right)_g, \tag{6}$$

где $\tilde{\mathbf{B}}_g^{n+1/2} = \big(\mathbf{B}_g^n + \tilde{\mathbf{B}}_g^{n+1}\big)/2$, необходимо вычислить плотность электрического тока, создаваемого частицами на промежуточном временном шаге:

$$\tilde{\mathbf{J}}_g^{n+1/2} = \sum_p q_p \tilde{\mathbf{v}}_p^{n+1/2} W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g\big). \tag{7}$$

Используя уравнение (2), промежуточную скорость $\tilde{\mathbf{v}}_p^{n+1/2}$ выражаем через электрическое поле $\tilde{\mathbf{E}}_p^{n+1/2}$, что позволяет установить линейную связь между сеточными значениями тока и электрического поля в будущем:

$$\tilde{\mathbf{v}}_p^{n+1/2} = \frac{1}{1+\alpha_p^2}\big[\mathbf{v}_p^n + \alpha_p\left[\mathbf{v}_p^n \times \mathbf{h}\right] + \alpha_p^2\mathbf{h}\big(\mathbf{h}\mathbf{v}_p^n\big)\big] + \\ \frac{1}{1+\alpha_p^2}\left[\frac{\beta_p\tilde{\mathbf{E}}_p^{n+1/2}}{2} + \alpha_p\left[\tilde{\mathbf{E}}_p^{n+1/2}, \mathbf{h}\right] + \alpha_p^2\big(\mathbf{h}\tilde{\mathbf{E}}_p^{n+1/2}\big)\mathbf{h}\right], \tag{8}$$

$$\tilde{\mathbf{J}}_g^{n+1/2} = \mathbf{I}_g + \frac{\Delta t}{4}\sum_{g'}\hat{\mathbf{L}}_{gg'}\big(\mathbf{E}_{g'}^n + \tilde{\mathbf{E}}_{g'}^{n+1}\big), \tag{9}$$

$$\mathbf{I}_g = \sum_p \frac{q_p}{1+\alpha_p^2}\big[\mathbf{v}_p^n + \alpha_p\left[\mathbf{v}_p^n \times \mathbf{h}\right] + \alpha_p^2\mathbf{h}\big(\mathbf{h}\mathbf{v}_p^n\big)\big]W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g\big), \tag{10}$$

$$\hat{\mathbf{L}}_{gg'} = L_{gg'}^{ij} = \sum_p \frac{q_p^2}{m_p(1+\alpha_p^2)}\left[\delta_{ij} + \alpha_p e_{ijm}h_m + \alpha_p^2 h_i h_j\right] \times W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g\big)W\big(\mathbf{x}_p^{n+1/2} - \mathbf{x}_{g'}\big), \tag{11}$$

$$\alpha_p = \frac{q_p\Delta t}{2m_p}|\mathbf{B}_p^n|, \qquad \beta_p = \frac{q_p\Delta t}{m_p}, \qquad \mathbf{h} = \frac{\mathbf{B}_p^n}{|\mathbf{B}_p^n|}. \tag{12}$$

Здесь под δ_{ij} и e_{ijm} понимаются единичные и абсолютно антисимметричные тензоры (в трёхмерном декартовом пространстве индексы i и j пробегают значения x, y, z, а g и g' нумеруют узлы пространственной сетки). Следует отметить, что, в отличие от других полунеявных схем [4,6,7], линейный отклик частиц на ЭМ-поля в методе ECSIM не является результатом каких-либо аппроксимаций. Подставив ток (9) в уравнения Максвелла, а также исключив из них магнитное поле, получим систему линейных алгебраических уравнений для сеточных электрических полей $\tilde{E}_g^{n+1}$ на новом шаге. Решив эту систему, мы затем вычисляем предварительные скорости для всех частиц $\tilde{\mathbf{v}}_p^{n+1}$. Завершая вычислительный цикл, определяем положения частиц на шаге $n+1$:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^{n+1/2} + \frac{\Delta t}{2}\tilde{\mathbf{v}}_p^{n+1}. \tag{13}$$

Представление тока в виде (7) удобно для обеспечения точного сохранения энергии, но если вычислить скорость изменения плотности заряда по полученным положениям частиц, то окажется, что это значение не удовлетворяет уравнению непрерывности с этим током. Это означает, что в такой схеме не выполняется закон Гаусса и в системе могут нарастать паразитные флуктуации электрического поля. Ток, сохраняющий заряд, должен удовлетворять конечно-разностному уравнению непрерывности

$$\Delta\rho_g = \sum_p q_p\left[W(\mathbf{x}_p^{n+1} - \mathbf{x}_g) - W(\mathbf{x}_p^n - \mathbf{x}_g)\right] = -\Delta t\left(\operatorname{div}\mathbf{J}^{n+1/2}\right)_g. \tag{14}$$

Однако, если новое значение тока $\mathbf{J}_g^{n+1/2}$ вычислить методом Есиркепова [12] непосредственно из уравнения (14) с той же формой частицы W, что и на этапе предсказания, то мы столкнёмся со следующей трудностью. Дело в том, что ток Есиркепова от каждой частицы вносит ненулевой вклад в меньшее количество узлов, чем ток (7), предсказанный по форме частицы в середине её траектории. Как показано в [11], большая локальная разница в плотности тока приводит к необходимости сильных поправок к локальному полю и вызывает численную неустойчивость при попытке внести эти поправки в энергию частиц. Проблема решается, если на этапе коррекции использовать более гладкое ядро частиц $\tilde{W}$. Например, численная схема остается устойчивой, если ток Есиркепова рассчитывается с использованием параболического ядра вместо линейного. Кроме того, поскольку траектория частицы на каждом временном шаге состоит из двух прямолинейных участков, для повышения точности расчёт тока Есиркепова состоит из двух этапов, соответствующих каждому из этих участков:

$$\Delta\rho_g = \sum_p q_p \left[\tilde{W}(\mathbf{x}_p^{n+1} - \mathbf{x}_g) - \tilde{W}(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g)\right] + \sum_p q_p \left[\tilde{W}(\mathbf{x}_p^{n+1/2} - \mathbf{x}_g) - \tilde{W}(\mathbf{x}_p^{n} - \mathbf{x}_g)\right]. \tag{15}$$

Тогда, решив уравнения Максвелла с новым током, получим скорректированные пространственные распределения ЭМ-полей $\mathbf{E}_g^{n+1}$ и $\mathbf{B}_g^{n+1}$. Величина, сохраняемая уравнениями Максвелла

$$\sum_g \left[\frac{1}{2}\left(\left|\mathbf{E}_g^{n+1}\right|^2 - \left|\mathbf{E}_g^{n}\right|^2 + \left|\mathbf{B}_g^{n+1}\right|^2 - \left|\mathbf{B}_g^{n}\right|^2\right) + \Delta t \mathbf{J}_g^{n+1/2}\,\mathbf{E}_g^{n+1/2} + \Delta t\,\nabla_g \left[\mathbf{E}_g^{n+1/2} \times \mathbf{B}_g^{n+1/2}\right]\right] = 0, \tag{16}$$

имеет смысл закона сохранения энергии, если работа скорректированного электрического поля над новым током равна изменению кинетической энергии частиц:

$$\sum_g \Delta t \mathbf{J}_g^{n+1/2}\,\mathbf{E}_g^{n+1/2} = \sum_p \frac{m_p}{2}\left(\left|\mathbf{v}_p^{n+1}\right|^2 - \left|\mathbf{v}_p^{n}\right|^2\right). \tag{17}$$

Поскольку то же равенство справедливо и на этапе предсказания, поправка на скорости частиц должна определяться изменением работы:

$$\sum_p \frac{m_p}{2}\left(\left|\mathbf{v}_p^{n+1}\right|^2 - \left|\tilde{\mathbf{v}}_p^{n+1}\right|^2\right) = \Delta t \sum_g \left[\mathbf{J}_g^{n+1/2}\,\mathbf{E}_g^{n+1/2} - \tilde{\mathbf{J}}_g^{n+1/2}\,\tilde{\mathbf{E}}_g^{n+1/2}\right]. \tag{18}$$

Самый простой способ восстановить закон сохранения энергии на корректирующем шаге — глобальная корректировка скоростей частиц $\mathbf{v}_p^{n+1} = \lambda\tilde{\mathbf{v}}_p^{n+1}$ с единым для всех коэффициентом

$$\lambda^2 = 1 + \frac{\Delta t \sum\limits_g \left[\mathbf{J}_g^{n+1/2}\,\mathbf{E}_g^{n+1/2} - \tilde{\mathbf{J}}_g^{n+1/2}\,\tilde{\mathbf{E}}_g^{n+1/2}\right]}{\sum\limits_p m_p\left|\tilde{\mathbf{v}}_p^{n+1}\right|^2/2}. \tag{19}$$

Однако в связи с тем, что ток в методе Есиркепова рассчитывается как сумма вкладов отдельных частиц $\mathbf{J}_g^{n+1/2} = \sum_p \mathbf{J}_g^p$, коррекцию энергии при необходимости можно проводить локально в соответствии с той работой, которая совершается над частицами в данной ячейке.

Матрично-операторное представление. Формулы (5), (6) можно представить в матрично-операторном виде. Для этого представим, например, 3D-векторное поле $\mathbf{E}$ в виде вектора

$$\mathbf{E} = \left[0\ \ldots\ \mathrm{Ind}(i,j,k,x)\ \ldots\ \mathrm{Ind}(i,j,k,y)\ \ldots\ \mathrm{Ind}(i,j,k,z)\ \ldots\ \mathrm{Ind}(Ni,Nj,Nk,z)\right]^\top,$$

где функция $\mathrm{Ind}(i,j,k,d)$ переводит 3-мерное сеточное векторное пространство в одномерное, например $\mathrm{Ind}(i,j,k,d) = d + 3 * (k + N_k * (j + N_j * i))$. Здесь d отвечает за номер пространственной координаты вектора ($d = 0$ для E_x, $d = 1$ для E_y и $d = 2$ для E_z), $i = [0, N_i]$, $j = [0, N_j]$, $k = [0, N_k]$, а N_i, N_j, N_k — числа узлов сетки по каждому направлению. Таким образом, задав функцию Ind, мы определяем обход сетки g в уравнениях (5)–(11). На сетке Йи, используя стандартную дискретизацию, перепишем $(\mathrm{rot}\,\mathbf{E})_g$ и $(\mathrm{rot}\,\mathbf{B})_g$ в виде умножения матрицы на вектор:

$$(\mathrm{rot}\,\mathbf{E})_g = \mathbf{curlE} * \mathbf{E}, \tag{20}$$

$$(\mathrm{rot}\,\mathbf{B})_g = \mathbf{curlB} * \mathbf{B}. \tag{21}$$

Матрицы **curlE** и **curlB** легко получить из разностной схемы. Например, если

$$(\mathrm{rot}\,\mathbf{E}_{i,j,k})(x) = \frac{Ez_{i,j+1,k} - Ez_{i,j,k}}{h_y} - \frac{Ey_{i,j,k+1} - Ey_{i,j,k}}{h_z},$$

где h_x, h_y, h_z — шаги сетки, то соответствующая часть матрицы **curlE** определяется как

$$\begin{aligned}
&\mathbf{curlE}\,[\mathrm{Ind}(i,j,k,z), \mathrm{Ind}(i,j+1,k,z)] = 1/h_y,\\
&\mathbf{curlE}\,[\mathrm{Ind}(i,j,k,z), \mathrm{Ind}(i,j,k,z)] = -1/h_y,\\
&\mathbf{curlE}\,[\mathrm{Ind}(i,j,k,y), \mathrm{Ind}(i,j,k+1,y)] = 1/h_z,\\
&\mathbf{curlE}\,[\mathrm{Ind}(i,j,k,y), \mathrm{Ind}(i,j,k,y)] = -1/h_z.
\end{aligned}$$

Аналогично можно получить остальные компоненты матрицы **curlE**, а также матрицы **curlB**.

Опустив индексы g и подставив полученные матрицы, а также уравнения (5) и (9) в (6), мы получим систему линейных алгебраических уравнения в матрично-операторном виде для этапа предсказания

$$\tilde{\mathbf{E}}^{n+1} = \mathbf{E}^n - \Delta t\,\mathbf{I}^{n+1/2} + \Delta t\big(\mathbf{curlB} * \tilde{\mathbf{B}}^n\big) + \frac{\Delta t}{4}(\mathbf{L} - \mathbf{curlB} * \mathbf{curlE}) * \big(\mathbf{E}^n + \tilde{\mathbf{E}}^{n+1}\big) \tag{22}$$

и этапа коррекции

$$\mathbf{E}^{n+1} = \mathbf{E}^n - \Delta t\,\mathbf{J}^{n+1/2} + \Delta t\big(\mathbf{curlB} * \tilde{\mathbf{B}}^n\big) - \frac{\Delta t}{4}\mathbf{curlB} * \mathbf{curlE} * \big(\mathbf{E}^n + \mathbf{E}^{n+1}\big). \tag{23}$$

Таким образом, полный вычислительный шаг рассматриваемой модели состоит из следующих этапов:

1) обновление координат частиц $\mathbf{x}_p^n \to \mathbf{x}_p^{n+1/2}$ по формуле (1);
2) вычисление сохраняющего заряд тока $\mathbf{J}^{n+1/4}$ в схеме Есиркепова по полученным на этапе (1) положениям частиц;
3) вычисление тока $\mathbf{I}_g$ по формуле (10);
4) вычисление матрицы $\mathbf{L}$ по формуле (11);
5) вычисление предсказанного электрического поля $\tilde{\mathbf{E}}^{n+1}$ через решение СЛАУ из уравнения (22);
6) вычисление скорости частиц $\tilde{\mathbf{v}}_p^{n+1/2}$ по предсказанному электрическому полю, согласно формуле (8);
7) обновление координат частиц $\mathbf{x}_p^{n+1/2} \to \mathbf{x}_p^{n+1}$ и скоростей $\tilde{\mathbf{v}}_p^{n+1/2} \to \tilde{\mathbf{v}}_p^{n+1}$;
8) вычисление сохраняющего заряд тока $\mathbf{J}^{n+3/4}$ в схеме Есиркепова по полученным на этапе (7) координатам частиц;
9) вычисление полного тока, сохраняющего заряд $\mathbf{J}^{n+1/2} = (\mathbf{J}^{n+1/4} + \mathbf{J}^{n+3/4})/2$;
10) вычисление нового электрического поля через решение СЛАУ из уравнения (23);

11) корректировка скоростей частиц с целью сохранения энергии по формуле (19);
12) вычисление магнитного поля $\mathbf{B}^{n+1} = \mathbf{B}^n - \Delta t\, \mathbf{curlE} * (\mathbf{E}^{n+1} + \mathbf{E}^n)/2$.

Рассмотрим теперь построение параллельного алгоритма для описанной выше модели.

2. Параллельный алгоритм

Поскольку нам удалось представить этапы (5), (6), (11) в матрично-операторном виде, то параллельный алгоритм сводится к простым операциям умножения разреженной матрицы на вектор, сложения векторов и решения СЛАУ. При этом здесь можно использовать как готовые библиотеки для работы с матрицами, такие как Eigen или PETSC, так и свои алгоритмы.

В случае распределённой памяти эффективность распараллеливания будет зависеть главным образом от реализации коммуникаций между элементами сетки и от распределения частиц по процессорам. Заметим, что при декомпозиции области расчёт всех сеточных величин, за исключением решения СЛАУ, происходит локально в каждой подобласти и все коммуникации сводятся только к обмену граничными значениями. Поэтому здесь можно использовать стандартные алгоритмы декомпозиции с балансировкой частиц, например, применённые в [14], а основные элементы работы с матрицами переложить на оптимизированную стороннюю библиотеку, например PETSC.

Однако, в случае с общей памятью, ключевым вопросом становится взаимодействие потоков с общими элементами сетки. При этом основной сложностью для рассматриваемого полунеявного метода является заполнение разреженной матрицы $\mathbf{L}$ на каждом шаге. Далее рассмотрим способы параллельного построения этой матрицы и синхронизации данных при параллельной обработке частиц в условиях конкуренции потоков за общие данные.

Для тестирования производительности здесь и далее мы будем использовать разработанный нами код Beren3D, написанный на языке C++.

Построение матрицы системы. Для работы с матрицами мы используем библиотеку Eigen, которая поддерживает многопоточность с помощью OpenMP, а также векторизацию вычислений. Все сеточные значения (кроме матрицы $\mathbf{L}$) представлены в виде векторов Eigen::VectorXd с функцией преобразования Ind, определённой выше. Таким образом, базовые операции с векторами распараллелены уже внутри библиотеки Eigen. Остальной параллелизм в коде Beren3D также обеспечивается за счёт применения OpenMP. Для решения СЛАУ мы выбрали метод BiCGSTAB, который можно легко распараллелить. В качестве предобуславливателя предлагается диагональная матрица.

Шаблоны матриц **curlE** и **curlB** зависят только от используемой разностной схемы, поэтому их можно определить один раз в самом начале моделирования. При этом все матрицы в уравнениях (22) и (23) имеют разреженную структуру. Матрица системы (23) также может быть вычислена в начале моделирования. Число ненулевых элементов в этой матрице фиксировано и равно $12\,N^3$ без учёта граничных условий, где N — полное число узлов сетки. Матрица $\mathbf{L}$ из уравнения (11) зависит от формы и положения частиц, поэтому её (а значит, и всю матрицу системы (22)) необходимо пересчитывать на каждом шаге по времени.

На рисунке 1 показана схема матрицы системы (22) для случая распределения частиц по всей области и локально. Видно, что число ненулевых элементов при этом существенно меняется, поэтому формат хранения матрицы в разреженном виде имеет решающую роль.

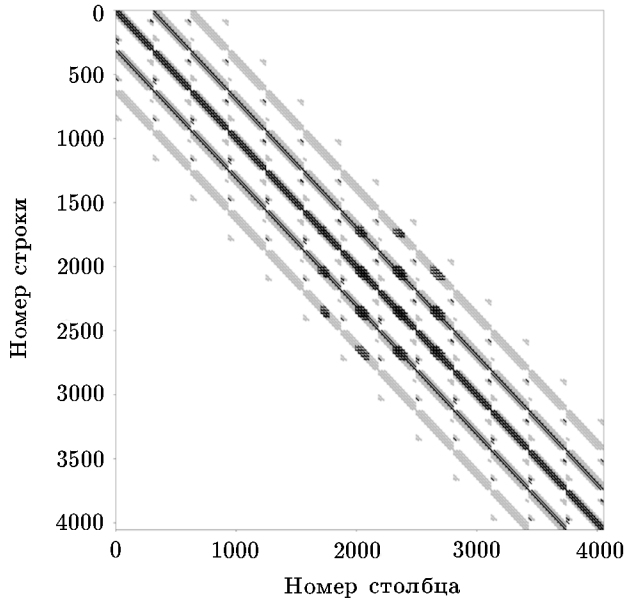

Рис. 1. Схема матрицы для (22). Серым показано положение ненулевых значений для распределения частиц по всей области, чёрным — распределение частиц в пределах нескольких ячеек

Существует несколько способов параллельного заполнения матрицы $\mathbf{L}$. Рассмотрим их все. Разреженное матричное представление модуля Eigen предлагает высокую производительность и низкое использование памяти. Он реализует более универсальный вариант широко используемой схемы хранения сжатых столбцов (или строк). Он состоит из четырех компактных массивов:

- Values хранит ненулевые значения коэффициентов;
- InnerIndices хранит индексы строк (соответственно столбцов) ненулевых значений;
- OuterStarts сохраняет для каждого столбца (соответственно строки) индекс первого ненулевого значения в двух предыдущих массивах;
- InnerNNZs хранит количество ненулевых значений в каждом столбце (соответственно строке).

Слово “Inner” относится к внутреннему вектору, который является столбцом для матрицы, хранящейся по столбцам, или строкой для матрицы, хранящейся построчно. Слово “Outer” относится к другому направлению. Мы выбрали построчный формат хранения, поскольку он обеспечивает лучший параллелизм для решения СЛАУ. Таким образом, распределение данных между потоками также будет построчное. Предполагая, что перераспределение не требуется, вставка случайного элемента происходит в $O(nnz_j)$, где nnz_j — количество ненулевых значений соответствующего внутреннего вектора. Кроме того, добавление элемента влияет одновременно на все 4 массива, поэтому любая случайная вставка при параллельной сборке матрицы $\mathbf{L}$ по формуле (11) предполагает блокировку всей матрицы потоком (в OpenMP для этого применяется критическая секция). Сразу заметим, что каждый поток не может иметь свою копию матрицы, поскольку для этого требуется очень большой объём памяти (в матрице $\mathbf{L}$ порядка $100\,N^3$ ненулевых элементов).

Чтобы избавиться от зависимости по данным, мы предлагаем использовать различные форматы хранения матрицы $\mathbf{L}$ для этапа 4 и решения СЛАУ на этапе 5. Поскольку

при решении СЛАУ предпочтителен формат сжатых строк (Compressed Row Storage или CRS), мы также будем рассматривать матрицу как набор строк. Для исключения зависимости по данным на этапе 4 мы будем хранить каждую строку независимо от другой, т. е. представим матрицу в виде массива строк. Заметим, что на сетке Йи из-за сдвига положения узлов сетки относительно друг друга нельзя только лишь по номеру ячейки (т. е. по номеру строки в матрице) однозначно определить номера столбцов с ненулевыми элементами. Это означает, что использование одного массива для хранения ненулевых значений в строке, как сделано, например, в работе [13], затруднительно. Таким образом, для каждой строки мы должны знать номер столбца ненулевого элемента и его значение.

Организовать хранение этих данных можно несколькими способами. Самый простой способ — хранить неупорядоченный или отсортированный массив пар столбец–значение, а при заполнении матрицы по положению частицы определять номер столбца и с помощью, например, линейного поиска определить индекс пары в этом массиве. Однако наиболее эффективным способом хранения является не массив, а неупорядоченный словарь unordered_map из стандартной библиотеки C++. Он позволяет выполнять операции поиска и вставки за $O(1)$ благодаря таблице хэшей. Тем не менее, все эти варианты также требуют блокировки данных (только уже не матрицы, а строки).

На рис. 2 а приведено время заполнения матрицы $\mathbf{L}$ для различного способа хранения и поиска данных для следующих параметров: число узлов сетки $N_x = N_y = N_z = 100$, общее число частиц 10^8. Частицы распределены внутри цилиндра, радиусом 30 ячеек. Вычисления выполнены на сервере ИЯФ СО РАН (два 64-ядерных процессора AMD EPYC 7773X 2.2 GHz). Для последовательного заполнения используется одно ядро в режиме с повышенной частотой (3.6 GHz), для параллельного — все 128 ядер. Мы сравнили заполнение матрицы в исходном формате (матрица Eigen) и хранение матрицы как массива строк различного вида: строка матрицы — массив с линейным поиском элементов; строка матрицы — словарь (мы выбрали в качестве словаря контейнеры map и unordered_map из стандартной библиотеки C++). Также мы измерили среднее время копирования матрицы из построчного формата в формат CRS, который используется в дальнейшем при решении СЛАУ. Как видно из рис. 2 а, хранение матрицы как массива неупорядоченных словарей (unordered_map) является предпочтительным и позволяет ускорить исходные вычисления почти в 20 раз, несмотря на расходы на копирование в формат CRS.

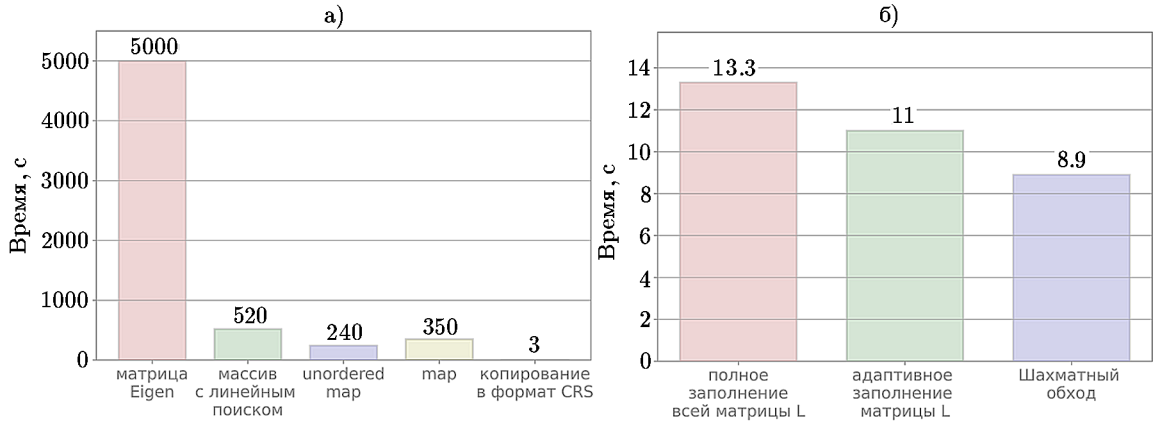

Рис. 2. а) время последовательного заполнения матрицы $\mathbf{L}$ для различных способов хранения и поиска данных, 1 поток; б) время параллельного заполнения матрицы $\mathbf{L}$ в формате вектора неупорядоченных словарей для различных способов синхронизации данных, 128 потоков

Обход частиц. Частицы в нашем коде представлены единой структурой, содержащей по три компоненты координат и скоростей. Поскольку на многих этапах вычислений нам одновременно необходимы эти значения на текущем и предыдущем шагах, то мы также храним в этой структуре начальные значения координат и скоростей для каждого шага.

Существует несколько подходов к распределению частиц между потоками. Самый простой — хранение частиц единым массивом и распределение элементов этого массива между потоками. Такой подход обеспечивает практически идеальную балансировку нагрузки, прост в реализации, однако требует работы с общими элементами сетки, для чего необходимо устранить гонку данных. К счастью, поскольку все сеточные значения представлены в виде плотных векторов, а все используемые матрицы, кроме матрицы **L**, определены заранее, то все операции в данном случае можно свести к атомарным (мы используем для этого omp_atomic). Только для матрицы **L**, как уже было сказано, необходима блокировка строк (через omp_critical). Однако, зная заранее максимальное число ненулевых элементов в строке, можно выделить память с запасом и зафиксировать структуру словаря (в данном случае это эквивалентно заполнению ключей словаря с присвоенным нулевым значением). Это позволяет использовать атомарную операцию вместо критической секции и в этом случае, поскольку мы таким образом исключаем перевыделение памяти и перехеширование таблицы.

Возвращаясь к рис. 1, можно увидеть, что этот способ существенно увеличит расходы на память, кроме того, время добавления числа в строку также может увеличиться из-за роста размера строки. Поэтому мы также рассмотрели динамическое заполнение матрицы **L**. Перед каждым шагом мы просматриваем только те ячейки, где уже есть частицы или куда они могут перелететь за шаг, и заполняем нулями значения матрицы для этих ячеек (если соответствующие ключи словарей ещё не заполнены). Это можно делать в один поток, поскольку таких ячеек не очень много. После этого можно обновлять матрицы параллельно с использованием атомарных операций.

Второй подход к распараллеливанию частиц состоит в том, чтобы хранить частицы по ячейкам и распределять между потоками ячейки целиком. Такой способ позволяет улучшить кэширование данных на сетке за счёт их большей локальности для потока, но требует дополнительного управления памятью при перемещении частиц между ячейками. Также при большом числе потоков эффективность может снижаться из-за неравномерного распределения ячеек между потоками. Однако, если несколько потоков обрабатывают ячейки, не имеющие общих соседей, упрощается не только перемещение частиц, но и исключаются общие элементы в матрице **L**. Чтобы реализовать такой обход ячеек параллельно, мы предлагаем использовать шахматное распределение задач. Ячейки раскрашиваются по цветам в шахматном порядке с шагом 2, что обеспечивает отсутствие нескольких соседей одного цвета. На каждом этапе потоки выбирают ячейки из множества ячеек, имеющих один цвет, пока все ячейки выбранного цвета не будут обработаны. Затем потоки переходят к следующему цвету.

На рис. 2б показано время заполнения матрицы **L** в параллельном варианте. Мы сравнили заполнение матрицы с заранее заполненными нулями строками (здесь используется атомарное обновление данных) с динамическим заполнением матрицы (по мере появления частиц в ячейке) и шахматным обходом ячеек. Несмотря на дополнительную синхронизацию потоков перед обработкой нового цвета, шахматный обход работает существенно быстрее. Схема обхода представлена на рис. 3. Обработку ячеек для перемещения частиц между ячейками также можно организовать шахматным способом. В остальных случаях, например при вычислении тока, мы используем обычный обход ячеек с атомарными операциями обновления данных.

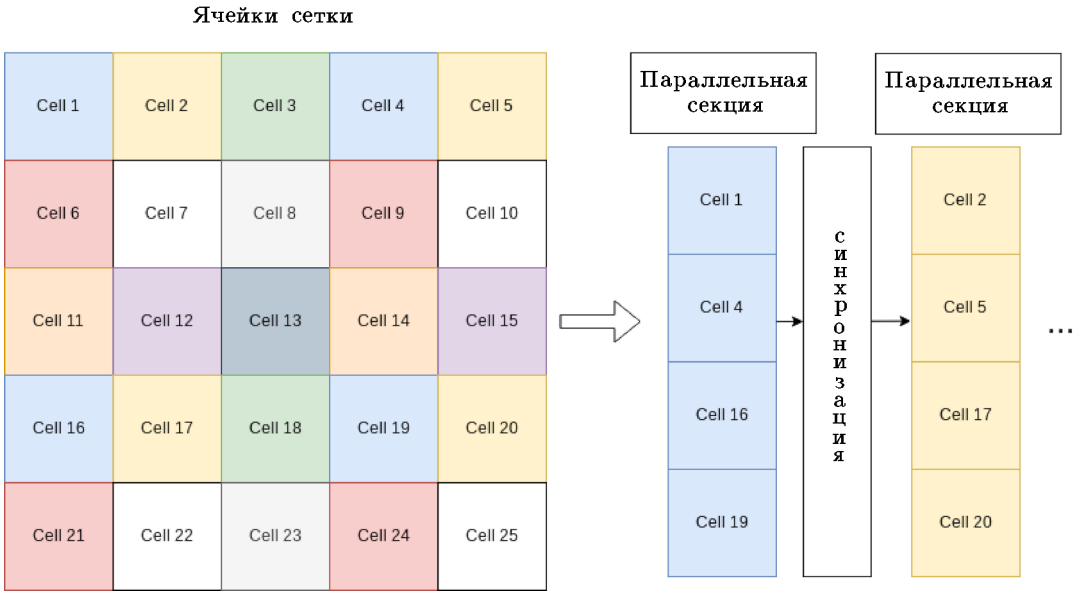

Рис. 3. Шахматная схема обхода частиц. Ячейки раскрашиваются так, чтобы каждый цвет не имел общих соседей одного цвета. После этого все одноцветные ячейки можно обрабатывать независимо друг от друга. При переходе между цветами необходима синхронизация

Производительность алгоритма. Оценим масштабируемость кода Beren3D и используемого параллельного алгоритма. На рис. 4 а приведено время выполнения основных процедур программы, а на рис. 4 б — график ускорения в зависимости от числа потоков. Как видно из рисунков, мы существенно снизили затраты на заполнение матрицы **L** (по сравнению с той же работой в [13]), а сам алгоритм достаточно хорошо масштабируется до 128 ядер (ускорение в 54 раза), несмотря на дополнительную синхронизацию. Ускорение кода в целом чуть меньше за счёт невысокой эффективности решения СЛАУ.

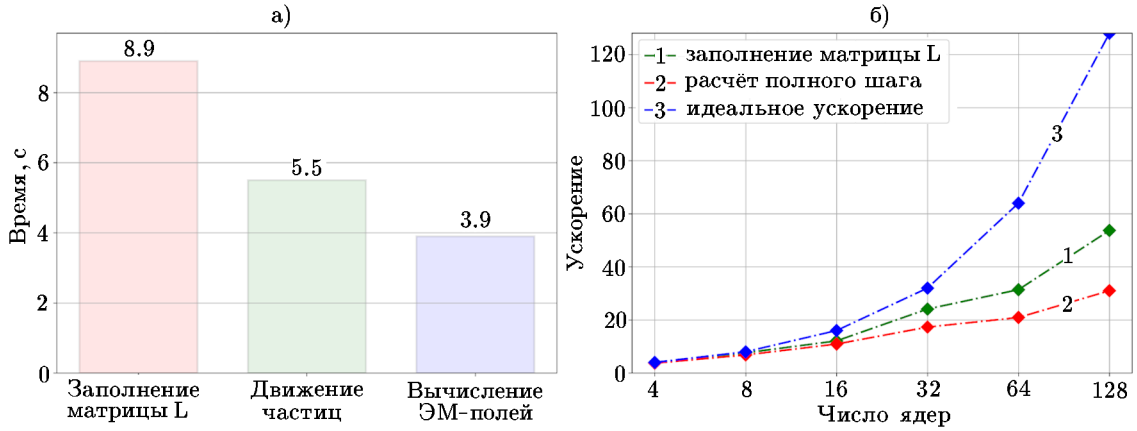

Рис. 4. а) время работы основных функций кода Beren3D за 1 шаг, 128 потоков; б) эффективность распараллеливания

Оценим эффективность рассматриваемого алгоритма для полунеявной схемы в сравнении с полностью неявной и явной схемами для типичных параметров эксперимента КОТ (Компактный Осесимметричный Тороид) ИЯФ СО РАН [15]. Поскольку полностью неявная схема при больших временных шагах (больше циклотронной частоты) неправильно воспроизводит магнитный момент и зеркальную силу, то ограничения на шаг по

времени для этой схемы полностью соответствуют ограничениям для полунеявной. Однако на каждую временную итерацию полностью неявной схемы приходится несколько итераций с движением частиц и решением СЛАУ для полей. Обычно необходимо не менее четырёх таких итераций. С учётом того, что, согласно данным из рис. 4 а, заполнение матрицы $\mathbf{L}$ в полунеявной схеме составляет 1.6 времени от движения частиц, то шаг по времени неявной схемы будет выполняться в $4/(1+1.6) = 1.5$ раза медленнее (а с учётом дополнительных решений СЛАУ этот фактор ещё больше).

Ограничения на шаги по времени и пространству составляют: для явной схемы

$$h_{\,\mathrm{ex}} < \lambda_{\mathrm{D}}, \qquad \Delta t_{\,\mathrm{ex}} < \frac{\lambda_{\mathrm{D}}}{2c} = \frac{v_{\mathrm{TE}}}{2c}\,\frac{1}{\omega_p},$$

для полунеявной

$$\Delta t_{\,\mathrm{im}} < \frac{h_{\,\mathrm{im}}}{6\lambda_{\mathrm{D}}\omega_p} < \frac{1}{\Omega_e},$$

где $h_{\,\mathrm{ex}}$, $\Delta t_{\,\mathrm{ex}}$, $h_{\,\mathrm{im}}$, $\Delta t_{\,\mathrm{im}}$ — шаги по пространству и времени для явной и полунеявной схем, λ_{D} — радиус Дебая, v_{TE} — тепловая скорость электронов, Ω_e — ларморовская частота электронов. Таким образом, соотношение шагов имеет вид

$$\frac{\Delta t_{\,\mathrm{im}}}{\Delta t_{\,\mathrm{ex}}} = 2\frac{\omega_p/\Omega_e}{v_{\mathrm{TE}}/c}, \qquad \frac{h_{\,\mathrm{im}}}{h_{\,\mathrm{ex}}} = 6\frac{\omega_p}{\Omega_e}.$$

Для параметров КОТ $v_{\mathrm{TE}}/c \approx 0.01$, $\omega_p/\Omega_e \approx 3$, т. е. $\dfrac{\Delta t_{\,\mathrm{im}}}{\Delta t_{\,\mathrm{ex}}} \approx 600$, $\dfrac{h_{\,\mathrm{im}}}{h_{\,\mathrm{ex}}} \approx 18$.

Заметим, что при этом полунеявная схема по сравнению с явной включает в себя дополнительные этапы движения частиц (для коррекции тока), решения СЛАУ для полей и заполнения матрицы $\mathbf{L}$. Из данных рис. 4 а можно видеть, что время, затраченное на эти дополнительные этапы, составляет $8.9+5.5+3.9 = 18.3$ секунд за шаг против $5.5/2 = 2.2$ секунд за шаг явной схемы, т. е. медленнее всего в 6.5 раза (решением уравнения для полей в явной схеме можно пренебречь). Если рассмотреть, например, решение задачи инжекции плазмы в прямоугольный слой, описанной в работе [11], то расчёт явным параллельным 2D-кодом занял более месяца, в то время как полунеявный параллельный метод в 3D-постановке потребовал около суток. При этом шумовое электрическое поле в расчёте было существенно меньше.

Таким образом, благодаря разработанному нами параллельному алгоритму, полунеявная схема не только уменьшает шумы за счёт точного сохранения энергии и заряда, но также и работает существенно быстрее полностью неявной и явной схем с учётом ограничений на шаг по времени.

Заключение

В статье рассмотрен параллельный алгоритм для полунеявного метода частиц в ячейках, сохраняющего энергию и заряд. Данный алгоритм применяется в коде Beren3D для моделирования динамики плазмы в ловушках с высоким бета. В коде реализован матрично-операторный подход к вычислениям, позволяющий использовать популярные библиотеки работы с векторами и свести вопрос параллелизма к простым матрично-векторным операциям. Для построения матрицы системы линейных алгебраических уравнений для неявной схемы при решении уравнений Максвелла мы рассмотрели несколько способов хранения и обработки данных. В качестве оптимального контейнера для матрицы был выбран построчный алгоритм поиска и вставки элементов с хэш-таблицей, реализованный в виде неупорядоченного словаря. Такой подход не только

обеспечивает ускорение в 20 раз по сравнению с обычным алгоритмом построения разреженной матрицы, но и позволяет добиться высокой масштабируемости вычислений (до 54 раз при 128 ядрах) за счёт шахматного алгоритма обхода ячеек.

Литература

1. **Markidis S., Lapenta G.** The energy conserving particle-in-cell method // J. Comput. Physics. — 2011. — Vol. 230. — P. 7037–7052. — https://doi.org/10.1016/j.jcp.2011.05.033.
2. **Ricketson L.F., Chacon L.** An energy-conserving and asymptotic-preserving charged-particle orbit implicit time integrator for arbitrary electromagnetic fields // J. Comput. Physics. — 2020. — Vol. 418. — Article № 109639. — https://doi.org/10.1016/j.jcp.2020.109639.
3. **Langdon A.B., Cohen B.I., Friedman A.** Direct implicit large time-step particle simulation of plasmas // J. Comput. Physics. — 1983. — Vol. 51. — P. 107–138.
4. **Welch D.R., Rose D.V., Clark R.E., Genoni T.C., Hughes T.** Implementation of an non-iterative implicit electromagnetic field solver for dense plasma simulation // Comput. Phys. Commun. — 2004. — Vol. 164, iss. 1-3. — P. 183–188. — https://doi.org/10.1016/j.cpc.2004.06.028.
5. **Brackbill J., Forslund D.** An implicit method for electromagnetic plasma simulation in two dimensions // J. Comput. Physics. — 1982. — Vol. 46, iss. 2. — P. 271–308. — https://doi.org/10.1016/0021-9991(82)90016-X.
6. **Noguchi K., Tronci C., Zuccaro G., Lapenta G.** Formulation of the relativistic moment implicit particle-in-cell method // Phys. Plasmas. — 2007. — Vol. 14. — Article № 042308. — https://doi.org/10.1063/1.2721083.
7. **Kempf A., Kilian P., Ganse U., Schreiner C., Spanier F.** PICPANTHER: A simple, concise implementation of the relativistic moment implicit particle-in-cell method // Comput. Phys. Commun. — 2015. — Vol. 188. — P. 198–207. — http://dx.doi.org/10.1016/j.cpc.2014.11.010.
8. **Lapenta G.** Exactly energy conserving semi-implicit particle in cell formulation // J. Comput. Physics. — 2017. — Vol. 334. — P. 349–366.
9. **Angus J.R., Link A., Friedman A., Ghosh D., Johnson J.D.** On numerical energy conservation for an implicit particle-in-cell method coupled with a binary Monte-Carlo algorithm for Coulomb collisions // J. Comput. Physics. — 2022. — Vol. 456. — Article № 111030.
10. **Campos Pinto M., Pagés V.** A semi-implicit electromagnetic FEM-PIC scheme with exact energy and charge conservation // J. Comput. Physics. — 2022. — Vol. 453. — Article № 110912. — https://doi.org/10.1016/j.jcp.2021.110912.
11. **Berendeev E.A., Timofeev I.V., Kurshakov V.A.** Energy and charge conserving semi-implicit particle-in-cell model for simulations of high-pressure plasmas in magnetic traps // Comput. Phys. Commun. — 2024. — Vol. 295. — Article № 109020.
12. **Esirkepov T.Zh.** Exact charge conservation scheme for particle-in-cell simulation with an arbitrary form-factor // Comput. Phys. Commun. — 2001. — Vol. 135. — P. 144–153.
13. **Gonzalez-Herrero D., Boella E., Lapenta G.** Performance analysis and implementation details of the energy conserving semi-implicit method code (ECsim) // Comput. Phys. Commun. — 2018. — Vol. 229. — P. 162–169.
14. **Derouillat J., Beck A., Perez F. at al.** Smilei: A collaborative, open-source, multi-purpose particle-in-cell code for plasma simulation // Comput. Phys. Commun. — 2018. — Vol. 222. — P. 351–373.
15. **Bagryansky P.A., Akhmetov T.D., Chernoshtanov I.S. et al.** Status of the experiment on magnetic eld reversal at BINP // AIP Conf. Proc. — 2016. — Vol. 1771. — Article № 030015.

Поступила в редакцию 15 февраля 2024 г.
После исправления 26 марта 2024 г.
Принята к печати 26 августа 2024 г.