ЖУРНАЛ ФИЗИЧЕСКОЙ ХИМИИ, 2024, том 98, с. 74–85

СТРОЕНИЕ ВЕЩЕСТВА И КВАНТОВАЯ ХИМИЯ

УДК 544.183

ЭФФЕКТИВНЫЙ АЛГОРИТМ МЕТОДА ХАРТРИ–ФОКА С ХРАНЕНИЕМ ДВУХЭЛЕКТРОННЫХ ИНТЕГРАЛОВ В ПРИБЛИЖЕНИИ РАЗЛОЖЕНИЯ ЕДИНИЧНОГО ОПЕРАТОРА[1]

© 2024 г. И. О. Глебов[a], *, В. В. Поддубный[a]

[a]Московский государственный университет им. М.В. Ломоносова, Химический факультет, 119991, Москва, Россия

**e-mail: glebov_io@phys.chem.msu.ru*

Поступила в редакцию 12.09.2023
После доработки 13.10.2023
Принята к публикации 16.10.2023

В данной работе реализован стандартный вариант ограниченного метода Хартри–Фока с хранением двухэлектронных интегралов в приближении разложения единичного оператора или разложения Холецкого. Наиболее трудоемкие стадии алгоритма сформулированы как матрично-векторные операции, что позволяет эффективно использовать современные библиотеки линейной алгебры. Показано, что алгоритм имеет высокую производительность и хорошую параллельность. Проведено сравнение с прямым вариантом, основанным на многократном пересчете двухэлектронных интегралов без их хранения в памяти. Показано, что разработанный алгоритм более эффективен для расчетов с использованием больших базисных наборов. В случае маленьких базисных наборов данный алгоритм более эффективен, чем прямой, для малых и средних молекул.

Ключевые слова: метод Хартри–Фока, разложение единичного оператора (resolution of identity), двухэлектронные интегралы.

DOI: 10.31857/S0044453724040096, **EDN:** QEXFVU

ВВЕДЕНИЕ

Метод Хартри–Фока (Hartree–Fock method – HF), или самосогласованного поля (self consistent field – SCF) [1, 2], – один из базовых методов квантовой химии. Несмотря на то что квантовая химия далеко шагнула вперед и было создано множество постхартрифоковских методов, он до сих пор остается актуальным. Сейчас он используется в качестве основы для расчетов методами теории возмущений MP2, конфигурационного взаимодействия и связанных кластеров. Поэтому этот метод должен быть наиболее простым во всей последовательности расчетов. Казалось бы, в наше время нет ни необходимости, ни возможности для улучшения метода Хартри–Фока. Тем не менее некоторые техники, широко используемые в постхартрифоковских методах, оказываются плохо приспособленными для метода HF. Например, разложение Холецкого [3] и разложение единичного оператора (Resolution of Identity – RI) [4] оказываются плохо реализованными для метода HF во многих широко используемых квантово-химических программах даже несмотря на то, что теоретические основы прекрасно известны [5]. Это приводит нас к ситуации, когда некоторые хорошо оптимизированные постхартрифоковские методы, например DLPNO-MP2 [6] (domain-based local pair natural orbital MP2) и DLPNO-CC [7] (domain-based local pair natural orbital coupled clusters), оказываются быстрее, чем метод Хартри–Фока.

Наиболее затратной по времени частью метода HF является расчет двухэлектронных интегралов, которые нужны для кулоновской и обменной составляющих оператора Фока. Стандартный подход заключается в расчете интегралов, записи их в оперативную память

[1] Дополнительные материалы к статье размещены на сайте https://elibrary.ru.

или на жесткий диск и последующем использовании в ходе повторяющихся итераций метода SCF. Однако общее количество интегралов составляет N_{AO}^4 (N_{AO} – количество атомных орбиталей), что для практически значимых расчетов невозможно уместить в оперативную память компьютера. Использование симметрии тензора двухэлектронных интегралов и различных схем, позволяющих не хранить пренебрежимо малые интегралы, не помогает решить эту проблему. В связи с этим был разработан так называемый прямой подход [8], в котором интегралы пересчитываются на каждой итерации SCF и используются в момент получения без записи на любой тип носителя. Также существует комбинированный «полупрямой» подход, в котором часть интегралов хранится в памяти, а часть пересчитывается. Пересчет интегралов на каждой итерации кратно увеличивает время расчета.

Такие приближения, как разложение Холецкого [3] и разложение единичного оператора [4], позволяют снизить как вычислительные затраты, так и требования оперативной памяти для расчета двухэлектронных интегралов. Они успешно применяются для ускорения таких квантово-химических методов, как MP2 [10], CASPT2 [11], CCSD [12] и многие другие. Для метода HF снижение требований памяти может оказаться даже более важным. Преимущество прямого и полупрямого подходов над стандартным перестает быть значимым, и многократный пересчет интегралов становится не нужен.

В настоящей статье показано, как можно реализовать метод Хартри–Фока в приближении разложения единичного оператора с записью интегралов в оперативную память. Описаны изменения, которые нужно внести в алгоритм. Показано, что после разложения тензор двухэлектронных интегралов может быть записан в оперативную память современных компьютеров, а использование оптимизированных библиотек линейной алгебры, например Intel MKL, позволяет эффективно реализовать данный алгоритм.

Статья имеет следующую структуру: сначала дана общая теория и описаны необходимые модификации используемых формул, далее приведены детали тестовых расчетов, в заключительной части дан сравнительный анализ производительности расчетов различными вариантами метода Хартри–Фока.

ТЕОРИЯ МЕТОДА

Ограниченный вариант метода Хартри–Фока (RHF)

Метод Хартри–Фока – это вариационный метод решения уравнения Шредингера для основного состояния, волновая функция которого приближенно считается определителем Слейтера. Одноэлектронные волновые функции (молекулярные орбитали – МО) описываются в приближении МО-ЛКАО:

$$\varphi_i = \sum_{\alpha} C_{i\alpha}\chi_{\alpha}, \qquad (1)$$

где χ_{α} – базисная функция (атомная орбиталь), а $C_{i\alpha}$ – коэффициенты, получаемые решением задачи на собственные функции оператора Фока:

$$\hat{F}\varphi_i = \varepsilon_i\varphi_i. \qquad (2)$$

В ограниченном варианте (restricted Hartree–Fock method – RHF) для системы с закрытыми оболочками оператор Фока является суммой одноэлектронного гамильтониана $\hat{h}$, кулоновского $\hat{J}$ и обменного $\hat{K}$операторов:

$$\hat{F} = \hat{h} + \sum_{i\in occ.}\left(2\hat{J}_i - \hat{K}_i\right) = \hat{h} + 2\hat{J} - \hat{K}, \qquad (3)$$

Матричные элементы $\hat{J}$ и $\hat{K}$

$$\left\langle \alpha\left|\hat{J}\right|\beta\right\rangle = \sum_{\gamma\delta} D_{\gamma\delta}(\alpha\beta \mid \gamma\delta), \qquad (4)$$

$$\left\langle \alpha\left|\hat{K}\right|\beta\right\rangle = \sum_{\gamma\delta} D_{\gamma\delta}(\alpha\gamma \mid \beta\delta). \qquad (5)$$

могут быть выражены через элементы матрицы плотности:

$$D_{\alpha\beta} = \sum_{i\in occ.} C_{i\alpha}C_{i\beta} \qquad (6)$$

и двухэлектронные интегралы:

$$(\alpha\beta \mid \gamma\delta) = \iiint \frac{\chi_{\alpha}(\vec{r}_1)\chi_{\beta}(\vec{r}_1)\chi_{\gamma}(\vec{r}_2)\chi_{\delta}(\vec{r}_2)}{\left|\vec{r}_1 - \vec{r}_2\right|} d\vec{r}_1 d\vec{r}_2. \qquad (7)$$

Лимитирующей стадией метода RHF является расчет двухэлектронных интегралов (7).

Именно здесь возникает необходимость хранения огромного объема данных в стандартном варианте или многократного повторения расчетов в прямом. Оптимизация этой стадии и является целью данной работы.

Разложение тензора двухэлектронных интегралов

Время расчета интегралов и объем используемой оперативной памяти могут быть снижены при использовании следующего разложения:

$$(\alpha\beta \mid \gamma\delta) \approx \sum_A L^A_{\alpha\beta} L^A_{\gamma\delta}. \quad (8)$$

Этого можно добиться разложением Холецкого [3] или с помощью внутреннего проектора (разложением единичного оператора) [4]:

$$(\alpha\beta \mid \gamma\delta) \approx \sum_{B,C} (\alpha\beta \mid B)\left(\mathbf{V}^{-1}\right)_{BC} (\gamma\delta \mid C). \quad (9)$$

Последнее обычно реализуется с помощью вспомогательных атомных базисных функций ρ_B [13, 14] и расчета трех- и двуцентровых интегралов:

$$(\alpha\beta \mid B) = \iiint \frac{\chi_\alpha(\vec{r}_1)\chi_\beta(\vec{r}_1)\rho_B(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|} d\vec{r}_1 d\vec{r}_2, \quad (10)$$

$$V_{BC} = \iiint \frac{\rho_B(\vec{r}_1)\rho_C(\vec{r}_2)}{|\vec{r}_1 - \vec{r}_2|} d\vec{r}_1 d\vec{r}_2. \quad (11)$$

Умножение тензора трехцентровых интегралов на «квадратный корень» обратной матрицы двуцентровых интегралов $\mathbf{V}^{-1}$ дает вектора L разложения (8):

$$L^A_{\alpha\beta} = \sum_B (\alpha\beta \mid B)\left(\mathbf{V}^{-1/2}\right)_{BA}. \quad (12)$$

Таким образом, разложение Холецкого и разложение единичного оператора – это два разных приближения, приводящих к форме (8).

Если количество векторов Холецкого равно N^2_{AO}, то разложение является точным, а размер тензора L равен размеру тензора $(\alpha\beta \mid \gamma\delta)$. Однако их количество может быть значительно снижено без существенной потери точности. То же верно и для разложения единичного оператора, так как размерность вспомогательного базисного набора (N_{Aux}) также меньше максимальной размерности ($N_{Aux} < N^2_{AO}$). Таким образом, тензоры $L^A_{\alpha\beta}$ и $(\alpha\beta \mid B)$ размера $N^2_{AO} N_{Aux}$ могут быть записаны в оперативную память современных вычислительных машин и рассмотренные приближения открывают возможность реализации стандартного варианта RHF без многократного пересчета двухэлектронных интегралов.

Далее рассмотрим метод RHF в приближении разложения единичного оператора (RI) по формуле (9).

Метод RI–RHF

Описанный в данном разделе алгоритм основан на уже известном подходе RI–RHF [5]. Он был модифицирован для применения к стандартной схеме (хранение интегралов без пересчета). Тензор двухэлектронных интегралов и другие промежуточные данные в описываемом алгоритме хранятся в матричной или векторной формах, что позволяет использовать эффективные алгоритмы из библиотеки линейной алгебры BLAS, сводя большую часть операций к умножению матриц.

Кулоновский оператор

Подстановка (9) в (4) дает:

$$\left\langle \alpha \middle| \hat{J} \middle| \beta \right\rangle = \sum_{\gamma,\delta,B,C} D_{\gamma\delta} (\alpha\beta \mid B)\left(\mathbf{V}^{-1}\right)_{BC} (\gamma\delta \mid C). \quad (13)$$

(Примечание: все приближенные формулы будут записаны как точные, чтобы не затруднять чтения. Никаких дополнительных приближений, кроме вышеописанных, использовано не будет.)

Расчет $\langle \alpha \mid \hat{J} \mid \beta \rangle$ можно разделить на следующие действия:

– умножение матрицы плотности на RI-тензор:

$$D_C = \sum_{\gamma,\delta} D_{\gamma\delta} (\gamma\delta \mid C); \quad (14)$$

– умножение результата D_C на $\mathbf{V}^{-1}$:

$$\left(DV^{-1}\right)_B = \sum_C \left(\mathbf{V}^{-1}\right)_{BC} D_C; \quad (15)$$

– умножение результата $\left(DV^{-1}\right)_B$ на RI-тензор:

$$\left\langle \alpha \left| \hat{J} \right| \beta \right\rangle = \sum_B (\alpha\beta \mid B)\left(DV^{-1}\right)_B. \quad (16)$$

Все операции можно свести к матричному умножению и использовать стандартные функции библиотеки BLAS. Матрица плотности представляется как вектор длины N_{AO}^2, $(\alpha\beta \mid B)$ – матрица размера $N_{AO}^2 \times N_{Aux}$, $\mathbf{V}^{-1}$ – матрица размера $N_{Aux} \times N_{Aux}$, а все промежуточные данные – вектора длины N_{Aux}. Эти векторно-матричные умножения требуют

$$N_{mult,J} = 2N_{AO}^2 N_{Aux} + N_{Aux}^2 \quad (17)$$

умножений действительных чисел, что является минимально возможным для расчета кулоновского оператора, заданного по уравнению (13), без использования дополнительных приближений.

Обменный оператор

Подстановка (9) в (5):

$$\left\langle \alpha \left| \hat{K} \right| \beta \right\rangle = \sum_{\gamma,\delta,B,C} D_{\gamma\delta} (\alpha\gamma \mid B)\left(\mathbf{V}^{-1}\right)_{BC} (\beta\delta \mid C) \quad (18)$$

приводит к выражению, которое не позволяет действовать так же, как в случае кулоновского оператора. Преобразование, подобное (14), приведет к суммированию только по одному индексу:

$$O_{\gamma\beta}^C = \sum_\delta D_{\gamma\delta} (\delta\beta \mid C). \quad (19)$$

Это требует больших вычислительных затрат – $N_{AO}^3 N_{Aux}$ умножений и хранения большего объема данных – $N_{AO}^2 N_{Aux}$ действительных чисел (увеличивает потребление памяти примерно в два раза).

Однако можно воспользоваться уравнением для матрицы плотности:

$$\left\langle \alpha \left| \hat{K} \right| \beta \right\rangle = \sum_{\gamma,\delta,B,C,i} C_{i\gamma} C_{i\delta} (\alpha\gamma \mid B)\left(\mathbf{V}^{-1}\right)_{BC} (\beta\delta \mid C). \quad (20)$$

Расчет обменного оператора можно начать со свертки тензора $(\gamma\alpha \mid B)$ с матрицей коэффициентов занятых орбиталей $C_{i\gamma}$, как было предложено ранее [5]. В отличие от прошлых работ, этот шаг реализован как одно матричное умножение с хранением данных в оперативной памяти. Алгоритм можно разбить на следующие действия:

– частичное преобразование трехцентровых интегралов к базису занятых молекулярных орбиталей:

$$(i\alpha \mid B) = \sum_\gamma C_{i\gamma} (\gamma\alpha \mid B). \quad (21)$$

Этот шаг требует $N_{occ} N_{AO}^2 N_{Aux}$ умножений и $N_{occ} N_{AO} N_{Aux}$ записей в память, N_{occ} – число занятых молекулярных орбиталей, которое значительно меньше, чем N_{AO}, особенно в случае большого базисного набора;

– транспонирование:

$$(i\alpha \mid B) \to (\alpha i \mid B). \quad (22)$$

Этот шаг требует $N_{occ} N_{AO} N_{Aux}$ записей;

– умножение на $\mathbf{V}^{-1/2}$:

$$M_{\alpha i}^A = \sum_B (\alpha i \mid B)\left(\mathbf{V}^{-1/2}\right)_{BA}. \quad (23)$$

Этот шаг требует $N_{occ} N_{AO} N_{Aux}^2$ умножений;

– расчет матрицы обменного оператора с использованием M:

$$\left\langle \alpha \left| \hat{K} \right| \beta \right\rangle = \sum_{A,i} M_{\alpha i}^A M_{\beta i}^A. \quad (24)$$

Этот шаг реализован как умножение матриц $M[\alpha, i \otimes A]$ и $M^T[i \otimes A, \beta]$. Один индекс матрицы – номер атомной орбитали α или β, а второй – обобщенный индекс пары занятой и вспомогательной орбиталей $i \otimes A$. Именно для того, чтобы было возможно реализовать этот шаг с помощью BLAS как умножение матриц, M должна иметь форму, где внешним индексом нумерации будет α. И именно для этого необходимо транспонирование (22).

Все данные, так же как и для кулоновского оператора, приведены к матричному виду, а операции матричного умножения реализованы через функции библиотеки BLAS. Общее число умножений составляет

$$N_{mult,K} = N_{occ} N_{AO} N_{Aux}^2 + 2N_{occ} N_{AO}^2 N_{Aux}. \quad (25)$$

Сравнение RI и разложения Холецкого

Алгоритм, построенный на разложении Холецкого и формуле (8), был бы даже проще, так как в нем не требуются операции

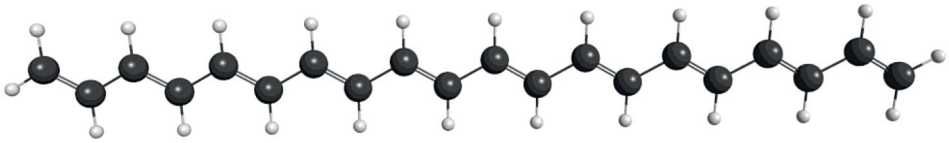

Рис. 1. Структура $C_{20}H_{22}$. Рисунок сделан в программе wxmacmolplt [16]. Атомы углерода изображены черным, водорода – серым. Два различных типа связей C–C изображены как двойные и одинарные.

(15) и (23) с матрицей **V**. Того же результата можно добиться, используя формулу (12) для перехода от трехцентровых интегралов $(\alpha\beta\,|\,B)$ к аналогу векторов Холецкого $L^{A}_{\alpha\beta}$. Однако такое преобразование потребует $N^2_{AO}N^2_{Aux}$ умножений, что может быть затратным. Во избежание этого на каждой итерации используются формулы (15) и (23), в которых умножаются матрицы меньшего размера.

Формула (15) требует N^2_{Aux} умножений, и ее удаление не принесет значимого облегчения расчета.

Преобразование (23) требует $N_{occ}N_{AO}N^2_{Aux}$ умножений и повторяется каждую итерацию. Сравнивая это с $N^2_{AO}N^2_{Aux}$-умножениями по формуле (12), которые производятся один раз, можно прийти к выводу, что это обосновано, если $N_{AO}\,/\,N_{occ}$ больше, чем ожидаемое число итераций. Для большого базисного набора это должно быть так.

В последующих расчетах использован алгоритм, описанный в пунктах «*Кулоновский оператор*» и «*Обменный оператор*».

ДЕТАЛИ РАСЧЕТОВ

Разработанный алгоритм был протестирован на предмет производительности, параллельности и масштабируемости. В качестве тестовых объектов были выбраны молекулы, для которых число базисных функций, размер молекулы и количество атомов линейно связаны друг с другом. Таким критериям в равной степени соответствуют многие гомологические ряды органических молекул, например алканы, полиены и т.д. В данной работе был выбран ряд линейных полиенов $C_{2n}H_{2n+2}$ от $C_{20}H_{22}$ до $C_{70}H_{72}$. Все структуры взяты плоскими, двойные связи имели *транс*-конфигурацию. Все валентные углы заданы по 120°, а длины связей C–H и обоих видов C–C взяты из геометрии бутадиена определенной микроволновой спектроскопией [15]. Структура $C_{20}H_{22}$ приведена на рис. 1.

Производительность алгоритма была сравнена с производительностью стандартных вариантов RHF реализованных в пакете программ Orca [17, 18] версии 5.0.3 с использованием гибридной схемы расчета интегралов с библиотеками SHARK [19] и LIBINT [20]. Использованы следующие варианты:

– **базовый** – традиционная схема расчета интегралов без разложения тензора и его хранения в памяти (прямой);

– **RIJK прямой** – приближение RI для двухэлектронных интегралов для расчета кулоновского (J) и обменного (K) операторов, прямой подход к работе с интегралами (без хранения);

– **RIJK стандартный** – то же, что и в прошлом пункте, но с хранением интегралов. Все временные файлы были помещены в оперативную память во избежание потери скорости из-за обращений к жесткому диску. Временные папки пакета Orca были также размещены в оперативной памяти (раздел tmpfs системы linux). Лимиты использования оперативной памяти были установлены на максимум.

В расчетах программой Orca были использованы настройки по умолчанию для метода Хартри–Фока. Стартовое приближение для молекулярных орбиталей было получено на основе модельного потенциала (Pmodel). Для сходимости SCF использована комбинация DIIS [21] и SOSCF [22]. Для пренебрежения околонулевыми двухэлектронными интегралами использовались критерии, заданные в программе Orca по умолчанию.

Разработанный алгоритм был реализован в **NOPT** – программе, написанной авторами данной работы на языке C++ для расчетов методами неортогонального конфигурационного взаимодействия [23] и их уточнения по теории возмущений [24] (исходный код программы доступен по запросу у авторов). В этой программе

были использованы библиотека LIBINT [20] версии 2.4.2 для расчета электронных интегралов и разные варианты BLAS для умножения матриц. Выбор библиотеки для расчета интегралов и ее версии не является принципиальным, т.к. эта стадия не лимитирует скорости расчета в стандартном варианте RHF. С другой стороны, выбор библиотеки линейной алгебры является определяющим, т.к. наиболее трудоемкие части расчета формализованы как произведение матриц большого размера. Рассмотрены два варианта библиотеки BLAS:

– **OpenBLAS**;

– **MKL**: Intel Math Kernel Library.

Стартовое приближение для молекулярных орбиталей было получено расширенным методом Хюккеля [25] в базисе MINI-1 Хузинаги [26, 27, 28]. Для сходимости SCF использован подход SOSCF [22]. В качестве критерия малости для пренебрежения двухэлектронными интегралами использовалось пороговое значение, выбранное в библиотеке LIBINT по умолчанию. Предварительный отбор интегралов по неравенству Шварца [9, 29] не проводился.

Выбор подходов для сравнения производительности объясняется следующим:

– базовый подход из программы Orca нужен для сравнения приближения RI с точным расчетом;

– разные варианты RIJK из программы Orca нужны для сравнения реализации прямого и стандартного вариантов RHF, написанных в рамках одной логики;

– альтернативные варианты RI из программы Orca не были рассмотрены: вариант RI только для кулоновского оператора (RIJ) не был рассмотрен, так как основная трудность заключается в расчете обменного вклада (см. пункты «*Кулоновский оператор*» и «*Обменный оператор*»); приближение цепи сфер для обменного оператора (RIJCOSX) [30] не рассмотрено, так как это дополнительное приближение, которое не использовано в рассмотренном алгоритме;

– рассматриваемый в данной статье алгоритм может быть назван RIJK в терминах программы Orca.

Все расчеты проведены с использованием базисных наборов семейства Def2 [31, 32]. Использованы наборы с двукратным и трехкратным ξ-расщеплением, дополненные диффузными и поляризационными функциями:

– Def2-SVP;

– Def2-SVPD;

– Def2-TZVP;

– Def2-TZVPP;

– Def2-TZVPD;

– Def2-TZVPPD.

Для разложения единичного оператора использован стандартный для программы Orca вспомогательный базисный набор Def2/J [33]. Параметры для всех базисных наборов были взяты из базы данных (BSE) [34, 35, 36].

Все тестовые расчеты проводились на компьютере с двумя центральными процессорами Intel Xeon E5-2697 v3 с частотой 2.60 ГГц и 14 ядрами на каждый процессор и оперативной памятью 196 ГБ. Максимальное потребление памяти при расчетах разработанным алгоритмом составило 176.5 ГБ для $C_{70}H_{72}$ в базисе Def2-SVPD. Для $C_{20}H_{22}$ в базисе Def2-TZVPPD потребовалось 18.4 ГБ. Подробная информация о потреблении памяти дана в Сопроводительных материалах.

ОБСУЖДЕНИЕ РЕЗУЛЬТАТОВ

Общие сведения

Во всех проведенных расчетах получены одинаковые энергии для одного и того же объекта в одном и том же базисе. Разница между величинами, полученными при расчетах в Orca и NOPT, была пренебрежимо малой. Разница между величинами, рассчитанными в приближении RI и базовым вариантом RHF, была адекватной для RI приближения.

Все стадии, предшествующие итерационному циклу SCF, занимали меньшее время, чем итерации SCF. Поэтому в данной статье основное внимание уделено времени итераций SCF. Использование различных схем сходимости SCF и стартовых приближений в Orca и NOPT приводит к различному числу итераций. Так же число итераций будет зависеть от критериев сходимости, которые могут быть более жесткими или более мягкими, в зависимости от задачи. Поэтому основной характеристикой скорости алгоритма было выбрано среднее время одной итерации и анализ был проведен на основании

Таблица 1. Величины α и R^2 для уравнения (26) для расчета $C_{20}H_{22}$ в базисе Def2-SVP

Программа	Метод	[1; 14]		[1; 28]	
		α	R^2	α	R^2
Orca	Базовый	0.937	0.999	0.898	0. 996
	RI прямой	0.705	0.989	0.648	0.982
	RI стандартный	0.365	0.940	0.265	0.789
NOPT	OpenBLAS	0.560	0.955	0.472	0.922
	MKL	0.757	0.991	0.669	0.970

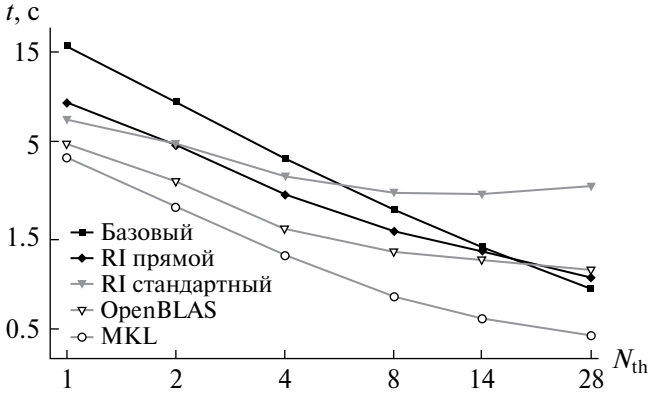

Рис. 2. Зависимости времени итерации от числа потоков N_{th} для $C_{20}H_{22}$ в базисе Def2-SVP для различных вариантов метода RHF. Оси имеют логарифмическую шкалу.

этого параметра. Данные о полном времени расчета, времени на стадии до цикла SCF и внутри цикла, а также число итераций можно найти в Дополнительных материалах.

Тестирование параллельности

Все рассмотренные варианты метода RHF были протестированы на параллельность. Были проведены расчеты для $C_{20}H_{22}$ всеми вариантами RHF с различным числом потоков. Результаты для расчетов в базисе Def2-SVP приведены на рис. 2.

На рис. 2 можно увидеть, что для непараллельного расчета (N_{th}=1) базовый вариант оказывается самым медленным, прямой RI – быстрее, а все варианты стандартного RI – еще быстрее. Оба варианта разработанного алгоритма быстрее, чем Orca, а MKL быстрее, чем OpenBLAS. Однако в случае параллельных вычислений ситуация меняется: базовый вариант становится быстрее, чем все варианты RI, кроме MKL, что вызвано лучшим параллелизмом алгоритма. Стандартный вариант RI в Orca имеет наихудший параллелизм, что, вероятно, связано со скоростью работы с оперативной памятью. Ситуация насколько лучше для прямого RI в Orca и разработанного алгоритма с использованием OpenBLAS. В то же время в случае использования библиотеки MKL разработанный алгоритм остается самым быстрым.

Численно охарактеризовать качество параллельности алгоритма можно, аппроксимировав полученные зависимости уравнением:

$$t \sim N_{th}^{-\alpha}. \qquad (26)$$

Полученные α и R^2 аппроксимированных кривых даны в табл. 1. Аппроксимация проведена для интервалов $N_{th} \in [1; 14]$, где в расчете можно использовать только один центральный процессор, и $N_{th} \in [1; 28]$, где в последней точке обязательно используются оба. Таблица 1 дает численное подтверждение вышеизложенным рассуждениям. Также видно, что переход от N_{th}=14 к N_{th}=28 дает небольшой прирост скорости расчета. Этот факт также подтверждает необходимость использовать хорошо оптимизированные библиотечные функции для линейной алгебры. При этом базовый алгоритм лишен этих недостатков. Тем не менее использование MKL также дает неплохой параллелизм и показывает наибольшую скорость для представленных расчетов.

Похожие результаты получены для $C_{20}H_{22}$ при использовании более крупных базисных наборов и для $C_{40}H_{42}$. На рис. 3 и 4 приведены зависимости среднего времени итераций от числа потоков для $C_{20}H_{22}$ в базисе Def2-TZVPPD и $C_{40}H_{42}$ в базисе Def2-SVP.

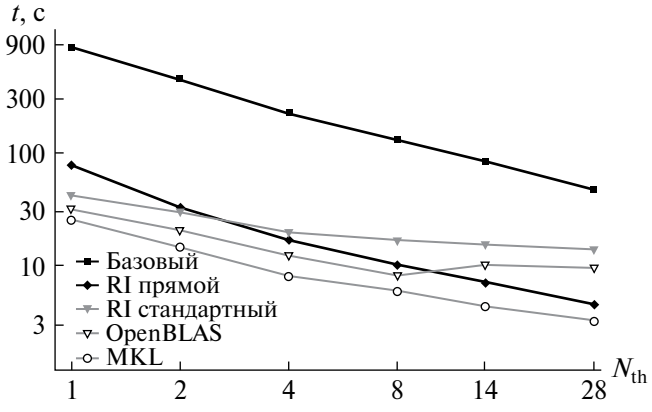

Рис. 3. Зависимости времени итерации от числа потоков N_{th} для $C_{20}H_{22}$ в базисе Def2-TZVPPD для различных вариантов метода RHF. Оси имеют логарифмическую шкалу.

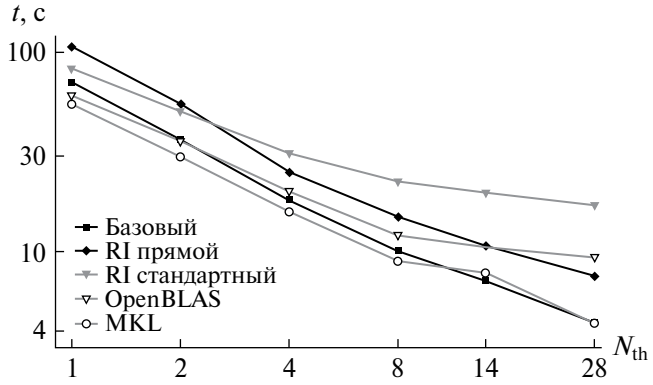

Рис. 4. Зависимости времени итерации от числа потоков N_{th} для $C_{40}H_{42}$ в базисе Def2-SVP для различных вариантов метода RHF. Оси имеют логарифмическую шкалу.

Из полученных результатов можно сделать следующие выводы:

– стандартный вариант RI-RHF в Orca плохо приспособлен к параллельным расчетам – вероятно, из-за неоптимальной работы с оперативной памятью;

– использование оптимизированных библиотек линейной алгебры снижает время расчета и улучшает параллельность стандартного варианта метода RHF;

– использовать преимущества библиотек BLAS можно только для алгоритмов, построенных на матрично-векторных операциях.

В дальнейшем будет использован только MKL, так как OpenBLAS уступает ему как по скорости, так и по параллельности.

«Тестирование масштабируемости: Базисный набор»

Расширение базисного набора по-разному влияет на производительность различных методов. Скорость расчета в стандартном варианте RI ограничена построением обменного оператора. Согласно формуле (25), время должно быть пропорционально $N_{AO}^2 N_{Aux}$. При использовании универсального вспомогательного базиса постоянной размерности время будет возрастать пропорционально N_{AO}^2. Для базового варианта время должно расти пропорционально N_{AO}^4, однако ситуация может быть лучше благодаря предварительному отбору и пренебрежению околонулевыми интегралами.

Зависимость среднего времени итераций от размерности базиса (N_{AO}) для параллельного (N_{th}=28) и непараллельного (N_{th}=1) расчетов дана на рис. 5 и 6 соответственно.

Рисунок 5 показывает, что базовый вариант становится самым медленным уже на второй точке – базисный набор Def2-SVPD. Кривая продолжает идти монотонно вверх и делает резкий скачок между точками № 4 (Def2-TZVPP, N_{AO}=928) и 5 (Def2-TZVPD, N_{AO}=938). Это явление вызвано заменой поляризационных функций диффузными, для которых число пренебрежимо малых (неиспользуемых в расчете) интегралов сокращается. Этот же эффект можно наблюдать при использовании прямого RI подхода (см. рис. 6). При использовании стандартного варианта метода RHF подобных скачков не наблюдается, так как интегралы не пересчитываются.

Зависимость времени итерации от размерности базиса была аппроксимирована функцией:

$$t \sim N_{AO}^{\beta}. \qquad (27)$$

Полученные параметры для параллельных и непараллельных расчетов даны в табл. 2.

Полученные β полностью соответствуют исходным предположениям: все RI-варианты замедляются пропорционально N_{AO}^2, а базовый – пропорционально N_{AO}^4. Разработанный алгоритм показывает наилучшую производительность особенно в случае базисных наборов, содержащих диффузные функции.

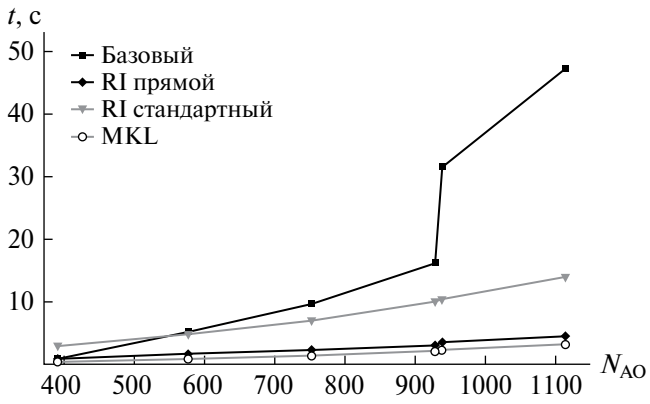

Рис. 5. Зависимости времени итерации от размерности базисного набора для $C_{20}H_{22}$ для различных вариантов метода RHF, N_{th}=28.

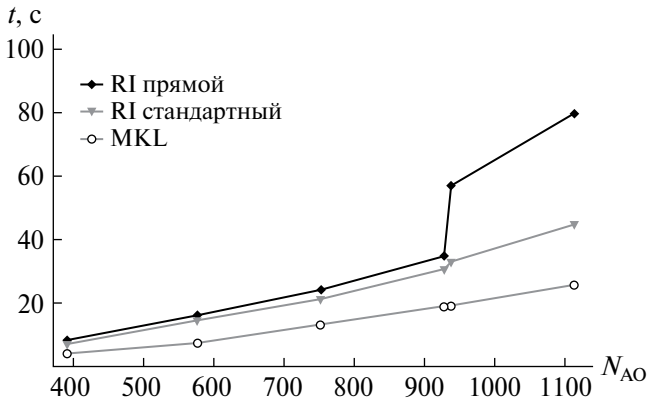

Рис. 6. Зависимости времени итерации от размерности базисного набора для $C_{20}H_{22}$ для различных вариантов метода RHF, N_{th}=1.

Таблица 2. β для уравнения (27) для параллельного и непараллельного расчетов $C_{20}H_{22}$ в различных базисах

Программа	Метод	N_{th}=28	N_{th}=1
Orca	Базовый	3.72	3.66
	RI прямой	1.47	2.09
	RI стандартный	1.50	1.73
NOPT	MKL	1.86	1.79

«Тестирование масштабируемости: размер молекулы»

Тестирование масштабируемости относительно размера молекулы было проведено для ряда полиенов от $C_{20}H_{22}$ до $C_{70}H_{72}$ с шагом 10 атомов углерода. Результаты расчетов в базисе Def2-SVP приведены на рис. 7.

Из рис. 7 видно, что разработанный алгоритм оказывается самым быстрым, вплоть до $C_{50}H_{52}$. Однако лучшая масштабируемость базового варианта RHF делает его быстрее для длинных молекул. Согласно данным раздела «*Тестирование масштабируемости: Базисный набор*», расширение базисного набора должно сдвинуть точку пересечения в сторону больших объектов. Результаты аналогичных расчетов в базисе Def2-SVPD даны на рис. 8.

Из рисунка видно, что уже при переходе к базису Def2-SVPD разработанный алгоритм становится самым быстрым на всем рассмотренном наборе молекул. Лучшая масштабируемость базового варианта сохраняется, но точка пересечения сдвигается в сторону экстремально больших объектов.

Зависимость среднего времени итерации от числа атомов углерода была аппроксимирована функцией:

$$t \sim N_C^{\gamma}. \qquad (28)$$

Полученные параметры для базисов Def2-SVP и Def2-SVPD даны в табл. 3.

Согласно уравнению (25) можно ожидать, что время расчета обменного вклада растет как N_C^4, однако лучшая масштабируемость кулоновского вклада несколько снижает γ относительно ожидаемого значения – 4. Однако все варианты метода в программе Orca характеризуются меньшими величинами γ в связи со снижением вычислительной работы из-за пренебрежения частью интегралов. Из этого следует, что при больших величинах N_C расчеты в программе Orca будут быстрее. На основе параметров уравнения (28) можно предсказать, что реализованный алгоритм будет быстрее базового вплоть до $C_{90}H_{92}$, прямого RI – до $C_{84}H_{86}$ и стандартного RI – до $C_{340}H_{342}$. Однако для молекул малого и среднего размера данный алгоритм будет самым быстрым. Это также справедливо для больших молекул в случае большого базисного набора.

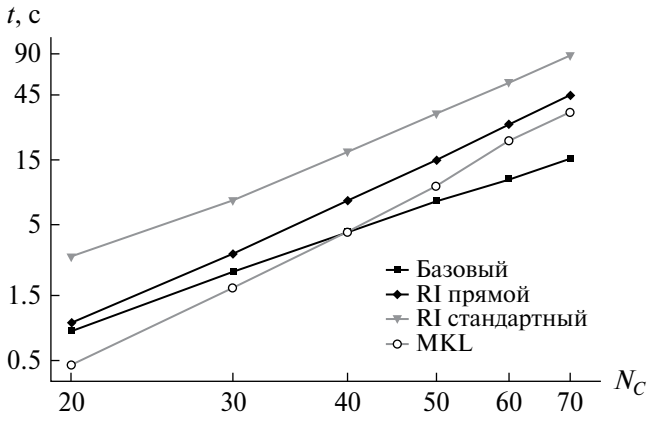

Рис. 7. Зависимости времени итерации от числа атомов углерода для расчетов различными вариантами метода RHF в базисе Def2-SVP, N_{th}=28. Оси имеют логарифмическую шкалу.

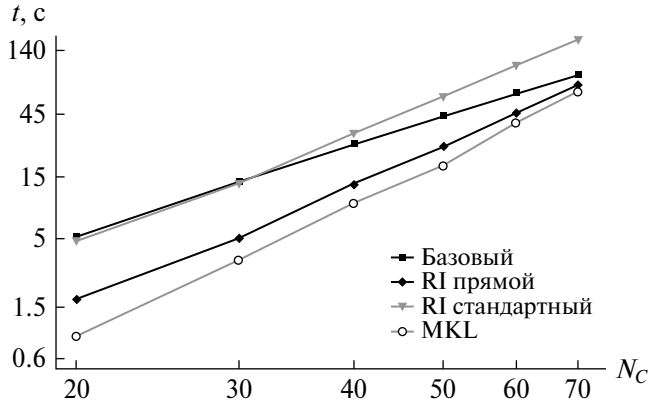

Рис. 8. Зависимости времени итерации от числа атомов углерода для расчетов различными вариантами метода RHF в базисе Def2-SVPD, N_{th}=28. Оси имеют логарифмическую шкалу.

Таблица 3. Величина γ для уравнения (28) для расчетов различных полиенов в базисах Def2-SVP и Def2-SVPD

Программа	Метод	γ_{SVP}	γ_{SVPD}
Orca	Базовый	2.32	2.29
	RI прямой	3.08	3.02
	RI стандартный	2.74	2.85
NOPT	MKL	3.44	3.44

ЗАКЛЮЧЕНИЕ

Проведенные расчеты показывают, что приближение RI менее эффективно для больших линейных молекул. Этот эффект играет важную роль в случае маленьких базисных наборов, не содержащих диффузных функций. В этом случае прямые методы становятся более эффективными из-за пренебрежения большим количеством двухэлектронных интегралов. В приближении разложения единичного оператора подобные действия не так эффективны, так как количество околонулевых трехцентровых интегралов вида (10) не так велико, как для четырехцентровых (7). В реализованном в данной статье алгоритме этот эффект роли не играет, так как расчет интегралов производится однократно. Все последующие вычисления проводятся с полноразмерными матрицами, и доля нулевых элементов влияет на время расчета незначительно. Применение техник работы с разряженными матрицами может улучшить алгоритм в этой части, но в данной работе этого сделано не было.

Тестирование производительности, приведенное в данной статье, можно обобщить следующими утверждениями:

– разработанный алгоритм является самым быстрым в случае расчетов для молекул среднего размера;

– разработанный алгоритм имеет лучшую масштабируемость при увеличении размерности базисного набора по сравнению с базовым вариантом RHF (β ~ 1.9 против β ~ 3.6);

– разработанный алгоритм имеет худшую масштабируемость при увеличении размера молекулы по сравнению с базовым вариантом RHF (γ ~ 3.4 против γ ~ 2.3), но более высокая производительность позволяет ему быть быстрее для малых и средних молекул;

– расширение базисного набора увеличивает интервал, на котором данный алгоритм быстрее базового.

В заключение мы можем рекомендовать разработанный алгоритм для расчетов во всех случаях, кроме очень больших молекул в малых базисах.

ВЫВОДЫ

В данной работе представлен модифицированный вариант метода Хартри–Фока в приближении разложения единичного оператора для тензора двухэлектронных интегралов. Этот алгоритм основан на хранении полученных трехцентровых интегралов в оперативной памяти и может быть классифицирован как стандартный вариант метода Хартри–Фока. В отличие от прямых вариантов метода, этот вариант метода не требует пересчета двухэлектронных интегралов на каждой итерации метода SCF, что делает его быстрее, чем общепринятые алгоритмы. Разложение тензора двухэлектронных интегралов по Холецкому или с помощью внутреннего проектора (разложение единичного оператора) снижает требования метода к оперативной памяти, что делает возможным проведение таких расчетов на современных вычислительных машинах с достаточным количеством оперативной памяти.

Тестирование численной эффективности с молекулами линейных полиенов в качестве объекта показало, что данный алгоритм при использовании эффективных библиотек линейной алгебры (Intel MKL) более эффективен, чем аналоги, реализованные в широко используемом квантово-химическом пакете Orca. Данный алгоритм имеет лучшую масштабируемость относительно увеличения размерности базисного набора, но худшую масштабируемость относительно увеличения размера молекулы. Последнее может быть объяснено отсутствием предварительного отбора двухэлектронных интегралов по неравенству Шварца. Это делает данный алгоритм менее эффективным по сравнению с базовым вариантом для расчета больших молекул в малых базисах. Однако проведенные тесты показали, что даже для Def2-SVP (простой базис с двойным ζ-расщеплением) данный алгоритм работает быстрее для $C_{40}H_{42}$. В случае молекул большего размера данный алгоритм быстрее базового при использовании большего базисного набора. Например, расчеты в базисе Def2-SVPD (базис с двойным ζ-расщеплением, дополненный диффузными функциями) быстрее проводить данным алгоритмом вплоть до $C_{90}H_{92}$. Таким образом, мы рекомендуем использовать данный алгоритм для всех ситуаций, кроме очень больших молекул и очень маленьких базисных наборов.

Работа выполнена при поддержке Минобрнауки России, госбюджетная тема № 121031300176-3.

СПИСОК ЛИТЕРАТУРЫ

1. *Roothaan C.C.J.* // Rev. Mod. Phys. 1951. V. 23. № 2. P. 69.
2. *Roothaan C.C J.* // Ibid. 1960. V. 32. № 2. P. 179.
3. *Beebe N.H.F.* // Int. J. Quantum Chem. 1977. V. 12. № 4. P. 683.
4. *Vahtras O., Almlöf J.* // Chem. Phys. Lett. 1993. V. 213. № 5–6. P. 514.
5. *Weigend F.* // Phys. Chem. Chem. Phys. 2002. V. 4. № 4. P. 4285.
6. *Pinski P., Riplinger C., Valeev E.F., Neese F.* // J. Chem. Phys. 2015. V. 143. № 3. P. 034108.
7. *Riplinger C., Pinski P., Becker U. et al.* // Ibid. 2016. V. 144. № 2. P. 024109.
8. *Almlöf J., Faegri K., Korsell K.* // J. Comput. Chem. 1982. V. 3. № 3. P. 385.
9. *Häser M., Ahlrichs R.* // Ibid. 1989. V. 10. № 1. P. 104.
10. *Weigend F., Häser M., Patzelt H. et al.* // Chem. Phys. Lett. 1998. V. 294. № 1–3. P. 143.
11. *Aquilante F., Malmqvist P.-Å., Pedersen T.B. et al.* // J. Chem. Theory Comput. 2008. V. 4. № 5. P. 694.
12. *Folkestad S.D., Kjønstad E.F., Koch H.* // J. Chem. Phys. 2019. V. 150. № 19. P. 194112.
13. *Eichkorn K., Treutler O., Öhm H. et al.* // Chem. Phys. Lett. 1995. V. 240. № 4. P. 283.
14. *Eichkorn K., Weigend F., Treutler O., Ahlrichs R.* // Theor. Chem. Acc. 1997. V. 97. № 1–4. P. 119.
15. *Craig N.C., Groner P., McKean D.C.* // J. Phys. Chem. A. 2006. V. 110. № 23. P. 7461.
16. *Bode B.M., Gordon M.S.* // J. Mol. Graph. Model. 1998. V. 16. № 3. P. 133.
17. *Neese F.* // WIREs Computational Molecular Science. 2011. V. 2. № 1. P. 73.
18. *Neese F.* // Ibid. 2017. V. 8. № 1. P. e1327.
19 *Neese F.* // J. Comput. Chem. 2022. V. 44. № 3. P. 381.
20. *Valeev E.F.* http://libint.valeyev.net/ 2021. version 2.7.1. https://doi.org/10.5281/zenodo.5516568
21. *Pulay P.* // Chem. Phys. Lett. 1980. V. 73. № 2. P. 393.
22. *Chaban G., Schmidt M.W., Gordon M.S.* // Theor. Chem. Acc. 1997. V. 97. № 1–4. P. 88.
23. *Glebov I.O., Kozlov M.I., Poddubnyy V.V.* // Comput. Theor. Chem. 2019. V. 1153. P. 12.

24. *Glebov I.O., Poddubnyy V.V., Khokhlov D.V.* // J. Phys. Chem. A. 2022. V. 126. № 34. P. 5800.

25. *Hoffmann R.* // J. Chem. Phys. 1963. V. 39. № 6. P. 1397.

26. *Tatewaki H., Huzinaga S.* // J. Comput. Chem. 1980. V. 1. № 3. P. 205.

27. *Andzelm J., Huzinaga S., Klobukowski M. et al.* // Phys. Sci. Data. 1984. V. 16. P. 27.

28. *van Duijneveldt F.B.* Gaussian basis sets for the atoms H-Ne for use in molecular calculations. IBM Research Laboratory, 1971.

29. *Horn H., Weiß H., Háser M. et al.* // J. Comput. Chem. 1991. V. 12. № 9. P. 1058.

30. *Neese F., Wennmohs F., Hansen A., Becker U.* // Chem. Phys. 2009. V. 356. № 1–3. P. 98.

31. *Weigend F., Ahlrichs R.* // Phys. Chem. Chem. Phys. 2005. V. 7. P. 3297.

32. *Rappoport D., Furche F.* // J. Chem. Phys. 2010. V. 133. № 13. P. 134105.

33. *Weigend F.* // Phys. Chem. Chem. Phys. 2006. V. 8. P. 1057.

34. *Pritchard B.P., Altarawy D., Didier B. et al.* // J. Chem. Inf. Model. 2019. V. 59. P. 4814.

35. *Feller D.* // J. Comput. Chem. 1996. V. 17. P. 1571.

36. *Schuchardt K.L., Didier B.T., Elsethagen T. et al.* // J. Chem. Inf. Model. 2007. V. 47. P. 1045.