ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”
(СПбГУ)

Кафедра вычислительной физики

Направление «Физика»

Метод представления дискретных переменных для решения квантовой задачи нескольких частиц

Магистерская диссертация студента

Тимошенко Владимира Андреевича

Научный руководитель:

к.ф.-м.н., доц. **Яревский Е.А.**

Рецензент:

к.ф.-м.н. **Колганова Е.А.**

Санкт–Петербург

2017

Оглавление

Введение

Существует множество примеров квантово-механических систем нескольких частиц: состоящие из нескольких кластеров системы в ядерной физике, атомы и молекулы в атомной физике или квантовые точки для нескольких электронов в физике твердого тела[1]. Описание данных систем — нетривиальная задача и требует применения различных методов для решения уравнения Шредингера нескольких частиц.

В данной работе были рассмотрены системы частиц, энергия связи которых мала, а волновая функция связанного состояния значительно распространена в пространстве: He_2, Ne_2, ^{6}Li-He, ^{7}Li-He, He_3, Ne_3, ^{6}Li-He_2, ^{7}Li-He_2, для которых были определены энергии связного состояния. Данные системы представляют особый интерес и были исследованы в других работах с результатами которых, можно ознакомиться в статьях [2–7]. Так как для данных молекул потенциал взаимодействия слабый, задача по нахождению энергий связи и волновых функций имеет дополнительные трудности при решении. При малых изменениях входных параметров или неточности вычислений результат может сильно отличаться от истинного. С другой стороны, в данных системах имеет место интересное явление — эффект Ефимова[8]. Эффект Ефимова возникает, когда энергия связи пары частиц близка к нулю, т.е. длина рассеяния $a \to \pm\infty$. Данное явление представляет собой существование последовательности энергий связи для трех частиц – состояний Ефимова[9].

Квантово-механическая система описывается операторм Гамильтона и его собственной волновой функцией. Волновая функция зависит от позиции в пространстве и других степеней свободы частиц. В случае трехчастичных систем нескольких частиц можно ограничится сравнительно небольшим набором координат, используя координаты Якоби (Рис. 1)[1]. Например, система трех частиц описывается набором координат $\{x, y, z = \cos\varphi\}$. Здесь использовалось то, что рассматриваемые состояния систем имеют нулевой угловой

момент и задачу можно решать в плоскости, в которой находятся частицы.

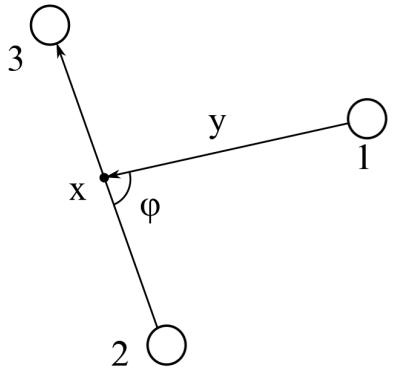

Рис. 1. Координаты Якоби

Исследование данных систем является непростой задачей и требует различных подходов, методов решения и дополнительных вычислительных мощностей. Даже при нулевом угловом моменте вычисления резонансных связных состояний для слабо связанных систем, которые рассматриваются в данной работе, трудно выполнить с хорошей точностью. Обширные исследования слабо связанных тримеров были проведены за последние десятилетия.

Различные методы были успешно использованы для решения поставленной задачи, включая метод конечных элементов (МКЭ)[2], другие вариационные методы[10], методы Монте Карло[11], методы конечных разностей[12]. Также авторами использовались различные системы координат, например, гиперсферические[7] или координаты Якоби[2, 10].

Основной частью данной работы является разработка и реализация метода дискретных переменных или DVR метода (Descrete Variable Representation) [13]. Вариационные методы хорошо себя зарекомендовали, но для точных расчетов вычисления трудоемки, требуют больших ресурсов и много времени. Для ускорения вычислений можно использовать представление дискретных переменных, основаное на базисе функций, которые, в некотором смысле, локализованы на сетке в координатном пространстве. Чтобы построить DVR-функции необходимо преобразовать набор ортонормированных базис-

ных функций, которые определены на промежутке в другой ортонормированый набор функций, которые локализованы в одной точке на сетке в пространстве. Данные точки являются точками квадратурной формулы Гаусса. За счет переход к DVR-функциям можно значительно сократить время вычислений.

В 1982 году Лилл, Паркер и Лайт[14] впервые описали данное представление, в котором оператор, зависящий от координат, является диагональным. Данный подход был независимо представлен Блэкмором и Шизгалом[15] в 1984 году под названием "метод дискретных ординат". Затем данный метод был применен для решения многомерной задачи[16–18]. Метод представления дискретных переменных, который был применен к слабо связанным тримерам подробно описан в статьях [19, 20].

Целью работы является разработка и реализация алгоритма, который позволяет значительно сократить время вычислений. Это позволит проводить расчеты при меньших вычислительных ресурсах без потери точности, уменьшить время ожидания вычислений, как следствие, проводить научные исследования быстрее и с меньшими финансовыми затратами.

Для выполнения цели были поставлены следующие задачи:

- Разработка и реализация эффективного алгоритма решения квантовой задачи двух частиц, используя естественное разложение по ортогональным функциям. В качестве данных функций были выбраны присоединенные полиномы Лагерра, так как они, домноженные на экспоненту, являются собственными функциями атома водорода. Результаты работы программы были проверены на атоме водорода.

- Исследование молекул двух частиц с помощью написанной программы, определение энергий связи и волновых функций для Ne_2 и He_2. Сравнение полученных значений с результатами других авторов.

- Реализация метода представления дискретных переменных, который

позволяет производить расчеты значительно быстрее без потери точности. Проведение анализа времени работы и полученных результатов.

- Модификация алгоритма вычисления матричных элементов, реализация DVR метода в программе решения трехчастичных квантовых задач ACESPA[21, 22]. Проверка работы программы на тримере неона, сравнение результатов и времени при использовании разложения по DVR-функциям и полиномам Лежандра.

- Исследование систем трех частиц ^{6}Li-He_2, ^{7}Li-He_2 и He_2. Сравнение энергий со значениями других авторов.

Глава 1

Метод представления дискретных переменных

В ходе работы был разработан и реализован алгоритм, который позволяет ускорить вычисление собственных энергий для квантово-механических систем, состоящех из нескольких атомов, таких как тример гелия $^4He_3$. Данный метод основан на определении функций, точек и весов квадратурной формулы таким образом, что значения функций в этих точках, кроме одной, равны нулю. Это позволяет избавиться от интеграла по одной из переменных. Благодаря данному переходу время вычисления матричной формы гамильтониана сокращается в несколько раз.

1.1. Квадратурная формула Гаусса

Квадратурная формула Гаусса в общем виде[23]

$$\int_a^b \rho(x) f(x)\ dx \approx \sum_{i=1}^{n} \omega_i f(x_i), \tag{1.1}$$

определяется узлами x_i и весами ω_i так, что формула точна для функций

$$f(x) = \sum_{i=0}^{2n-1} c_i p_i(x), \tag{1.2}$$

где $\{p_i\}$ — набор линейно независимых полиномы, $\rho(x)$ — положительная весовая функция. Погрешность $R_n(f)$ квадратурной формулы Гаусса определяется:

$$R_n(f) = \frac{f^{(2n)}(\xi)}{(2n)!} \int_a^b \rho(x) Q_n(x)\ dx. \tag{1.3}$$

Здесь $\xi \in [a, b]$, $Q_n(x)$ — полином степени n, корнями которого являются узлы квадратурной формулы. Если $f(x)$ — полином степени не выше $2n-1$,

то квадратурная формула точна.

В частности, $a = -1,\ b = 1, \rho = 1$ для полиномов Лежандра [24]

$$P_n(x) = \frac{1}{2^n n!}\frac{d^n}{dx^n}(x^2 - 1)^n. \tag{1.4}$$

Узлы в данном случае — нули полинома Лежандра $P_n(x)$, а веса ω_i определяются соотношением

$$\omega_i = \frac{2}{(1 - x_i^2)[P_n'(x_i)]^2}. \tag{1.5}$$

Такую квадратурную формулу называют формулой Гаусса-Лежандра, существуют квадратурные формулы Гаусса-Лагерра, Гауса-Чебышева и другие, построенные по соответствующим полиномам.

1.2. Дискретные дельта-функции

Основной задачей при выполнении данной работы было построение и применение метода, основанного на использовании дискретных дельта-функций или DVR-функций (Discrete Variable Representation)[25]. DVR-функции связаны с квадратурной формулой Гаусса и определяются соотношением:

$$\varphi_i(x) = \frac{P_n(x)}{P_n'(x_i)(x - x_i)}, \tag{1.6}$$

где $x_1, \ldots, x_n$ — корни полинома $P_n(x)$. Заметим, что

$$\varphi_i(x_k) = \delta_{ik}. \tag{1.7}$$

Также были использованы производные DVR-функций.

$$\varphi_i'(x) = \left(\frac{P_n(x)}{P_n'(x_i)(x - x_i)}\right)' = \frac{P_n'(x)(x - x_i) - P_n(x)}{P_n'(x_i)(x - x_i)^2}. \tag{1.8}$$

Учитывая равенство $P_n(x_k) = 0$, при $k \neq i$:

$$\varphi_i'(x_k) = \frac{P_n'(x_k)}{P_n'(x_i)(x_k - x_i)}. \tag{1.9}$$

Для того, чтобы определить $\varphi_i'(x_i)$, необходимо разложить функцию $P_n(x)$ в ряд в окрестности x_i:

$$P_n(x) = P_n(x_i) + (x - x_i)P_n'(x_i) + \frac{(x - x_i)^2}{2}P_n''(x_i) + O((x - x_i)^3). \quad (1.10)$$

После подстановки данного разложения в уравнение (1.8) получим :

$$\varphi_i'(x_i) = -\frac{P_n''(x_i)}{2P_n'(x_i)}. \quad (1.11)$$

1.3. Одномерная задача

В случае взаимодействия двух частиц нет необходимости решать трехмерную задачу, достаточно одной координаты, чтобы описать систему. Это упрощение возникает из-за сферически-симметричного потенциала, при котором трехмерное уравнение распадается на одномерные, и было использовано для разработки алгоритма нахождения энергии связи частиц.

Из множества экспериментов получен потенциал взаимодействия рассматриваемых парных частиц в виде сферически-симметричной функции. Таким образом, зная потенциал, можно записать радиальное уравнение. Неизвестная волновая функция в данном уравнении раскладывается по присоединенным палиномам Лагерра[24, 26], которые являются, домноженные на экспоненту, собственными волновыми функциями для атома водорода[27]. Таким образом, можно получить хорошее приближение точного решения исходной задачи.

1.3.1. Радиальное уравнение

Запишем радиальное уравнение Шредингера [27]:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{dR(r)}{dr}\right) - \frac{l(l+1)}{r^2}R(r) + \frac{2m}{\hbar^2}[E - U(r)]R(r) = 0 \quad (1.12)$$

Рассмотрим случай $l = 0$ и домножим уравнение на $-r^2$:

$$-\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)+\frac{2m}{\hbar^2}r^2U(r)R=\frac{2mE}{\hbar^2}r^2R \tag{1.13}$$

Для рассматриваемых систем данное допущение обосновано. Так как энергия связи очень мала, были рассмотрены лишь самые глубокие энергетические уровни, т.е. при $n=0$, а следовательно и при $l=0$.

Перейдя к атомным единицам, получим:

$$-\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)+2r^2U(r)R=2r^2ER \tag{1.14}$$

Масса m может отличаться от массы электрона m_e, таким образом в уравнении должен быть множитель m. Этот множитель учтен в потенциале и энергии, то есть $mU\to U$ и $mE\to E$.

Домножим уравнение на дифференцируемую функцию F и возьмем интеграл:

$$-\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)+2r^2U(r)R=2r^2ER\ \Big|\ \int_0^{+\infty}\cdot F(r)dr \tag{1.15}$$

$$-\int_0^{+\infty}\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)Fdr+2\int_0^{+\infty}r^2U(r)RFdr=2E\int_0^{+\infty}r^2RFdr \tag{1.16}$$

Преобразуем первое слагаемое, взяв интеграл по частям:

$$-\int_0^{+\infty}F\cdot\frac{d}{dr}\left(r^2\frac{dR}{dr}\right)dr=-F\cdot r^2\frac{dR}{dr}\Big|_0^{+\infty}+\int_0^{+\infty}\frac{dF}{dr}\cdot r^2\frac{dR}{dr}dr=\int_0^{+\infty}r^2F'R'dr, \tag{1.17}$$

$$\int_0^{+\infty}r^2R'F'dr+2\int_0^{+\infty}r^2U(r)RFdr=2E\int_0^{+\infty}r^2RFdr. \tag{1.18}$$

Произведя замену переменной $x = \lambda r$, получим уравнение, записанное в виде:

$$\frac{\lambda^2}{2}\int\limits_0^{+\infty} x^2 R' F' dx + \int\limits_0^{+\infty} x^2 U\left(\frac{x}{\lambda}\right) RF dx = E\int\limits_0^{+\infty} x^2 RF dx. \tag{1.19}$$

Разложим решение радиального уравнения по ортогональным функциям, а в качестве функции F выберем одну из них:

$$R = \sum_i c_i \psi_i = \sum_i c_i e^{-x/2} L_i^2; \quad F = \psi_j = e^{-x/2} L_j^2, \tag{1.20}$$

где L_i^2 – нормированный обобщенный полином Лагерра[26]. Заметим, что

$$\langle \psi_i | \psi_j \rangle = \int\limits_0^{+\infty} x^2 L_i^2 L_j^2 e^{-x} dx = \delta_{ij}. \tag{1.21}$$

Подставляя (1.20) в уравнение (1.19), получим следующее равенство:

$$\sum_i c_i \underbrace{\frac{\lambda^2}{2}\int\limits_0^{+\infty}\left(L_i' - \frac{L_i}{2}\right)\left(L_j' - \frac{L_j}{2}\right) x^2 e^{-x} dc}_{T_{ij}} +$$

$$+ \sum_i c_i \underbrace{\int\limits_0^{+\infty} U\left(\frac{x}{\lambda}\right) L_i L_j x^2 e^{-x} dx}_{V_{ij}} = E c_j \ , \ \forall\ j \tag{1.22}$$

Таким образом, это уравнение можно записать в матричном виде, получив обобщенную задачу на собственные значения:

$$\hat{H}\mathbf{c} = E\mathbf{c}\,, \qquad \hat{H} = \hat{T} + \hat{V}. \tag{1.23}$$

После нахождения собственных значений и собственных функций поставленная задача будет решена.

1.3.2. Квадратурная формула Гаусса — Лагерра

Как видно из соотношения (1.22), необходимо вычислять большое количество интегралов на интервале $[0;\infty)$. Для этого удобно использовать квадратурную формулу Гаусса — Лагерра [28], которая является частным случаем квадратурной формулы Гаусса. Данный метод аппроксимирует значения интегралов вида

$$\int_0^{+\infty} e^{-x} f(x)\, dx \tag{1.24}$$

рядом по n точкам:

$$\int_0^{+\infty} e^{-x} f(x)\, dx \approx \sum_{i=1}^{n} w_i f(x_i), \tag{1.25}$$

где x_i — это i-й корень палинома Лагерра $L_n(x)$. Коэффициенты w_i определяются соотношением:

$$w_i = \frac{x_i}{(n+1)^2 [L_{n+1}(x_i)]^2}. \tag{1.26}$$

Ошибка приближенного вычисления интеграла равна:

$$R_n = \frac{n!}{(2n)!} f^{(2n)}(\xi),\ (0 < \xi < \infty). \tag{1.27}$$

Заметно, что данный метод вычисления интегралов естественно вписывается в решение нашей задачи.

Вычисления можно значительно ускорить, если перейти к DVR-функциям. Для этого вместо того, чтобы раскладывать решение по полиномам Лагерра, разложим его по нормированным DVR-функциям:

$$\psi_i = \frac{\varphi_i}{x_i \sqrt{w_i}}, \tag{1.28}$$

$$\varphi_i = \frac{L_n(x)}{L_n'(x_i)(x - x_i)}. \tag{1.29}$$

Здесь значения $x_1, \ldots, x_n$ — корни уравнения $L_n(x) = 0$, и вместе с коэффициентами $w_1, \ldots, w_n$ определяют квадратурную формулу Гаусса-Лаггера.

Заметим, что

$$\varphi_i(x_k) = \delta_{ik}. \tag{1.30}$$

Благодаря последнему равенству матрица, отвечающая потенциальной энергии, является диагональной:

$$V_{ij} = \int_0^{+\infty} x^2 U\left(\frac{x}{\lambda}\right) \psi_i(x)\psi_j(x)e^{-x}dx =$$

$$= \sum_k w_k x_k^2 U\left(\frac{x_k}{\lambda}\right) \frac{\varphi_i(x_k)}{x_i\sqrt{w_i}} \frac{\varphi_j(x_k)}{x_j\sqrt{w_j}} = U\left(\frac{x_i}{\lambda}\right)\delta_{ij}. \tag{1.31}$$

Для того, чтобы определить матрицу кинетической энергии, необходимо вычислить значения производной $\psi_i'(x)$ в точках $x_1, \ldots, x_n$. Сначала найдем производную $\varphi_i'(x)$:

$$\varphi_i'(x) = \left(\frac{L_n(x)}{L_n'(x_i)(x - x_i)}\right)' = \frac{L_n'(x)(x - x_i) - L_n(x)}{L_n'(x_j)(x - x_j)^2}. \tag{1.32}$$

Учитывая $L_n(x_k) = 0$, при $k \neq i$ получаем:

$$\varphi_i'(x_k) = \frac{L_n'(x_k)}{L_n'(x_i)(x_k - x_i)}. \tag{1.33}$$

Для того, чтобы определить $\varphi_i(x_i)$, необходимо разложить функцию $L_n(x)$ в ряд в окрестности x_i:

$$L_n(x) = L_n(x_i) + (x - x_i)L_n'(x) + \frac{(x - x_i)^2}{2}L_n'' + O((x - x_i)^3). \tag{1.34}$$

После подстановки данного разложения в уравнение (1.32) получим :

$$\varphi_i'(x_i) = -\frac{L_n''(x_i)}{2L_n'(x_i)}. \tag{1.35}$$

Далее, используя уравнения для полиномов Лагерра, можно выразить $L_n''(x)$ через $L_n'(x)$.

$$xL_n''(x) + (1 - x)L_n'(x) + nL_n(x) = 0, \tag{1.36}$$

$$L_n''(x) = -\frac{(1 - x)L_n'(x) + nL_n(x)}{x}. \tag{1.37}$$

В точке x_i:

$$L_n''(x_i) = -\frac{(1-x_i)L_n'(x_i)}{x_i}. \tag{1.38}$$

Подставляя (1.38) в (1.35), получим:

$$\varphi_i'(x_i) = \frac{1-x_i}{2x_i}. \tag{1.39}$$

После того, как определены производные функций разложения, можно перейти к вычислению матрицы, отвечающей кинетической энергии:

$$T_{ij} = \int_0^{+\infty} x^2 \left(\psi_i'(x) - \frac{\psi_i(x)}{2}\right)\left(\psi_i'(x) - \frac{\psi_i(x)}{2}\right) e^{-x} dx =$$

$$= \sum_k w_k x_k^2 \frac{2\varphi_i'(x_k) - \delta_{ik}}{2x_i\sqrt{w_i}} \frac{2\varphi_j'(x_k) - \delta_{jk}}{2x_j\sqrt{w_j}}. \tag{1.40}$$

В данной сумме значение w_i быстро убывает с ростом i, это приводит к проблемам при реализации данного алгоритма. Для того, чтобы избежать этого, необходимо воспользоваться формулой для w_i:

$$w_i = \frac{1}{x_i[L_n'(x_i)]^2}. \tag{1.41}$$

Подставляя данное значение, получим:

$$\psi_i'(x_k) = \frac{\varphi_i'(x_k)}{x_i\sqrt{w_i}} = \frac{\operatorname{sign}(L_n'(x_i))L_n'(x_k)}{\sqrt{x_i}(x_k - x_i)} \quad \text{при } k \neq i, \tag{1.42}$$

$$\psi_i'(x_i) = \frac{\varphi_i'(x_i)}{x_i\sqrt{w_i}} = \frac{(1-x_i)|L_n'(x_i)|}{2x_i^{3/2}}. \tag{1.43}$$

Следовательно,

$$\psi_i'(x_k) - \frac{\psi_i(x_k)}{2} = \frac{\operatorname{sign}(L_n'(x_i))L_n'(x_k)}{\sqrt{x_i}(x_k - x_i)} \quad \text{при } k \neq i. \tag{1.44}$$

$$\psi_i'(x_i) - \frac{\psi_i(x_i)}{2} = \frac{|L_n'(x_i)|(1-2x_i)}{2x^{3/2}}. \tag{1.45}$$

Также в сумме (1.40) необходимо разбить множитель $w_k x_k^2$, тогда получаем:

$$\sum_i c_i \underbrace{\frac{\lambda^2}{2}\sum_k \left(\sqrt{w_k}x_k\left[\psi_i'(x_k) - \frac{\psi_i(x_k)}{2}\right]\right)\left(\sqrt{w_k}x_k\left[\psi_j'(x_k) - \frac{\psi_j(x_k)}{2}\right]\right)}_{T_{ij}} +$$

$$+\sum_i c_i \underbrace{U\left(\frac{x_i}{\lambda}\right)\delta_{ij}}_{V_{ij}} = Ec_j\ ,\ \forall\ j. \tag{1.46}$$

Заметим, что вторая сумма равна $c_j U\left(\frac{x_j}{\lambda}\right)$. В этом и есть преимущество применение DVR-функций.

1.4. Трехмерная задача

1.4.1. Уравнение Шредингера задачи трех частиц

Запишем уравнение Шредингера для нулевого углового момента[2]:

$$H = \left(-\frac{1}{\mu_{1,23}}\frac{1}{y}\frac{\partial^2}{\partial y^2}y + \frac{1}{\mu_{23}}\frac{1}{x}\frac{\partial^2}{\partial x^2}x\right) +$$

$$+\left(\frac{1}{\mu_{1,23}y^2} + \frac{1}{\mu_{23}x^2}\right)\left(\frac{\partial^2}{\partial\varphi^2} + \operatorname{ctg}\varphi\frac{\partial}{\partial\varphi}\right) + V(x,y,\varphi) \tag{1.47}$$

Здесь потенциал $V = V(x,y,\varphi)$ является суммой двухчастичных сферически-симметричных потенциалов, которые зависят от расстояний между частицами, а приведенные массы μ определены массами частиц $m_1,\ m_2,\ m_3$:

$$\mu_{23} = \frac{m_2 m_3}{m_2+m_3}, \qquad \mu_{1,23} = \frac{m_1(m_2+m_3)}{m_1+(m_2+m_3)}. \tag{1.48}$$

Ускорение вычислений произойдет при интегрировании по координате $z = \cos\varphi$. Перепишем оператор кинетической энергии для координаты φ в терминах z:

$$\frac{d^2}{d\varphi^2} + \operatorname{ctg}\varphi\frac{d}{d\varphi} = (1-z^2)\frac{d^2}{dz^2} - 2z\frac{d}{dz}. \tag{1.49}$$

Подействуем оператором на дифференцируемую функцию g, домножим на f и возьмем интеграл:

$$\int_{-1}^{1} f\left[(1-z^2)\frac{d^2}{dz^2} - 2z\frac{d}{dz}\right] g\ dz. \tag{1.50}$$

Рассмотрим слагаемое с производной второй степени, проинтегрируем его по частям:

$$\int_{-1}^{1} f(1-z^2)g''\,dz = f(1-z^2)g'\Big|_{-1}^{1} - \int_{-1}^{1} f'(1-z^2)g'\,dz + \int_{-1}^{1} 2zfg'\,dz. \quad (1.51)$$

Таким образом,

$$\int_{-1}^{1} f\left[(1-z^2)\frac{d^2}{dz^2} - 2z\frac{d}{dz}\right] g\,dz = -\int_{-1}^{1} f'g'(1-z^2)\,dz. \quad (1.52)$$

Далее данное выражение будет использована для вычисления матричных элементов оператора кинетической энергии.

1.4.2. DVR-функции и полиномы Лежандра

При помощи полиномов Лежандра определены DVR-функции, которые локализованы в узлах квадратурной формулы. Полиномы Лежандра $P_n(z)$ определяются уравнением:

$$P_n(z): \qquad (1-z^2)\frac{d^2P_n(z)}{dz^2} - 2z\frac{dP_n(z)}{dz} + n(n+1)P_n(z) = 0. \quad (1.53)$$

Условие ортогональности полиномов на отрезке $[-1,\ 1]$:

$$\int_{-1}^{1} P_k(z)P_l(z)\,dz = \frac{2}{2k+1}\delta_{kl}, \quad (1.54)$$

где δ_{kl} — символ Кронекера. Построим DVR-функцию при помощи полиномов Лежандра. Функция φ_i является нормированной DVR-функцией:

$$\psi_i = \frac{\varphi_i}{\sqrt{w_i}}, \quad (1.55)$$

$$\varphi_i(z) = \frac{P_n(z)}{P_n'(z_i)(z-z_i)}. \quad (1.56)$$

Заметим, что

$$\varphi_i(z_k) = \delta_{ik}. \quad (1.57)$$

Теперь решение уравнения (1.47) можно представить как линейную комбинацию DVR-функций:

$$f(z) = \sum_i c_i \psi_i(z). \tag{1.58}$$

Благодаря равенству (1.57) матрица, отвечающая потенциальной, энергии является диагональной:

$$V_{ij} = \sum_k w_k \, V(x, y, z_k) \frac{\varphi_i(z_k)}{\sqrt{w_i}} \frac{\varphi_j(z_k)}{\sqrt{w_j}} = V(x, y, z_i)\delta_{ij}. \tag{1.59}$$

Для того, чтобы определить матрицу кинетической энергии, необходимо вычислить значения производной $\psi_i'(z)$ в точках $z_1, \ldots, z_n$. По аналогии с преобразованием выражений (1.32 - 1.39) получаем значения для производных:

$$\varphi_i'(z_k) = \frac{P_n'(z_k)}{P_n'(z_i)(z_k - z_i)}, \qquad i \neq k; \tag{1.60}$$

$$\varphi_i'(z_i) = \frac{z_i}{1 - z_i^2}. \tag{1.61}$$

После того, как определены производные функций разложения, можно перейти к вычислению матрицы, отвечающей кинетической энергии, используя полученное выражение (1.52):

$$T_{ij} = \int_{-1}^{1} \psi_i'(z)\psi_j'(z)(1 - z^2)dz = \sum_k w_k \frac{\varphi_i'(z_k)}{\sqrt{w_i}} \frac{\varphi_j'(z_k)}{\sqrt{w_j}} (1 - z_k^2). \tag{1.62}$$

В данной сумме значение w_i определено формулой (1.5). Подставляя данное значение, получим:

$$\psi_i'(z_k) = \frac{\varphi_i'(z_k)}{\sqrt{w_i}} = \operatorname{sign}(P_n'(z_i)) P_n'(z_k) \frac{\sqrt{1 - z_i^2}}{\sqrt{2}} \quad \text{при } k \neq i, \tag{1.63}$$

$$\psi_i'(z_i) = \frac{\varphi_i'(z_i)}{\sqrt{w_i}} = \frac{z_i |P_n'(z_i)|}{\sqrt{2(1 - z_i^2)}}. \tag{1.64}$$

После того, как операторы записаны в матричной форме, можно перейти к решению задачи на собственные числа.

1.4.3. DVR-функции и полиномы Чебышева

Заметим что при координате Якоби $\varphi \longrightarrow 0; \pi$ или $z = \cos\varphi \longrightarrow \pm 1$ может возникнуть особенность в потенциале парного взаимодействия частиц порядка $\sim (1 - z^2)^{-m}$, где m — положительно. Например, в случае кулоновского взаимодействия $m = 1/2$. Чтобы компенсировать данную особенность, обратимся к полиномам Чебышева первого рода[24], построим по ним DVR-функции. Полиномы Чебышева $T_n(z)$ определяются уравнением:

$$T_n(z): \qquad (1 - z^2)\frac{d^2T_n(z)}{dz^2} - z\frac{dT_n(z)}{dz} + n^2T_n(z) = 0. \tag{1.65}$$

Условие ортогональности этих полиномов на отрезке $[-1,\ 1]$:

$$\int_{-1}^{1} T_k(z)T_l(z)\,\frac{dz}{\sqrt{1 - z^2}} = \frac{\pi}{2}\delta_{kl}, \qquad l, k \neq 0, \tag{1.66}$$

где δ_{kl} — символ Кронекера. Именно благодаря весу $\dfrac{1}{\sqrt{1 - z^2}}$ и удается сгладить особенность в потенциале. Интеграл на отрезке $[-1,\ 1]$ можно вычислять с помощью квадратурной формулы

$$\int_{-1}^{1} f(z)\,dz = \sum_{i=1}^{n} w_i f(z_i), \tag{1.67}$$

где $z_1, \ldots, z_n$ — корни уравнения $T_n(z) = 0$, w_i — веса, равные константе $\dfrac{\pi}{2}$. Построим DVR-функции и их производные при помощи полиномов Чебышева аналогично построению для полиномов Лежандра:

$$\varphi_i(z) = \frac{T_n(z)}{T_n'(z_i)(z - z_i)}. \tag{1.68}$$

Заметим, что

$$\varphi_i(z_k) = \delta_{ik}. \tag{1.69}$$

Производные в точках вычисляются по формулам:

$$\varphi_i'(z_k) = \frac{T_n'(z_k)}{T_n'(z_i)(z_k - z_i)}. \tag{1.70}$$

$$\varphi_i'(z_i) = \frac{z_i}{2(1 - z_i^2)}. \tag{1.71}$$

Глава 2

Результаты

2.1. Реализация метода представления дискретных переменных для решения задачи нескольких частиц

2.1.1. Программа для исследования систем двух частиц

Алгоритм нахождения собственных чисел и функций для пар частиц был полностью самостоятельно реализован на языке c++ (Приложение Б). Для расчетов интегралов была использована библиотека (John Burkardt), содержащая функции для работы с полиномами Лагерра[29], несколько функций было дописано и одна исправлена. Также использовалась библиотека ALGLIB[30] для решения спектральной задачи. Корректность работы программы была проверена, используя известное решения уравнения Шредингера атома водорода.

В замене переменных (1.19) параметр λ можно выбирать произвольным. Но для увеличения точности расчетов была реализована возможность выбора λ, описывающего асимптотическое поведение волновой функции. Таким образом, можно получить результат, близкий к точному решению при меньшем количестве полиномов в разложении (1.20). Данная возможность была реализована путем последовательных приближений. Связь λ и энергии определяется соотношением:

$$\lambda = 2\sqrt{-2Em}, \tag{2.1}$$

где m — приведенная масса. Так для атома водорода λ для n-ой волновой функции $\lambda = 2/n$.

Вычисления производились на компьютере MacBookPro (Mid 2012) с процессором 2.5 GHz Intel Core i5 и оперативной памятью 4 GB 1600 MHz DDR3.

2.1.2. Оптимизация программы для исследования систем нескольких частиц

Следующим шагом стали реализация и использование метода представления дискретных переменных в программе решения трехчастичных квантовых задач ACESPA[21, 22], которая реализована на языке fortran90 (Приложение В).

Вычисления производятся с помощью трехмерного метода конечных элементов (МКЭ)[2]. Трехмерное конфигурационное пространство разделено на набор прямоугольных параллелепипедов. В каждом элементе выбран набор линейно независимых полиномиальных базисных функций. Волновая функция разложена по данным базисным функциям. В случае угловой координаты разложение было произведено по полиномам Лежандра. Таким образом осуществляется переход от уравнения Шредингера к задаче на нахождение собственных чисел.

$$\hat{H}\mathbf{c} = E\hat{S}\mathbf{c}. \tag{2.2}$$

Здесь вектор $\mathbf{c}$ определяется коэффициентами разложения волновой функции.

Благодаря локализации базисных функций матрица Гамильтониана $\hat{H}$ и матрица перекрытия $\hat{S}$ являются разреженными. Линейная задача решается при помощи метода Арнольди. Также используются технологии параллельных вычислений OPENMP и MPI.

Функции в разложении по координате z (1.58) были заменены на дискретные дельта-функции. Была написана подпрограмма, вычисляющая DVR-функции и их производные и модифицирована основная подпрограмма вычисления матричных элементов.

Для исследования систем ^{6}Li-He,^{7}Li-He,^{6}Li-He_2,^{7}Li-He_2 была описана функция определения потенциала [31] взаимодействия и реализовано программное описание данных систем.

Вычисления производились на удаленной машине ресурсного центра СПбГУ. Параметры машины: 6-ти ядерный процессор Intel(R) Xeon(R) X5670 2.93GHz, оперативная память 32 GB.

2.2. Системы двух частиц

2.2.1. Атом водорода

Корректность работы программы, написанной на языке c++, для вычисления энергий связи для двухчастичных систем была проверена, используя известное решение уравнения Шредингера атома водорода. В данном случае потенциал имеет кулоновский вид:

$$U = -\frac{1}{x}, \tag{2.3}$$

Обобщенные полиномы Лагерра, домноженные на экспоненту, являются собственными функциями для данной системы:

$$R_n = const \cdot e^{-x/2} L_n^1(x), \tag{2.4}$$

а собственные значения энергий:

$$E_n = -\frac{1}{2n^2},\ n = 1, 2, \ldots. \tag{2.5}$$

Таким образом, ожидалось получить те же волновые функции и собственные значения энергий.

В таблице 2.1 представлены полученные значения энергии связи протона и электрона в атоме водорода, также приведены теоретические значения. Видно, что они совпадают с машинной точностью. Также были построены волновые функции (Рис. 2.1) и распределения вероятности (Рис. 2.2), соответствующие первым трем состояниям. Зависимости совпадают с хорошо известными кривыми [32]. Таким образом, можно сделать вывод, что численные расчеты верны и программа работает корректно. Для исследования зависи-

Таблица 2.1. Энергетические уровни атома водорода

n	$E_n, a.e.$	$-1/2n^2, a.e.$
1	-0.50000	-0.5
2	-0.12500	-0.125
3	-0.05556	-0.05556
4	-0.03125	-0.03125
5	-0.02000	-0.02

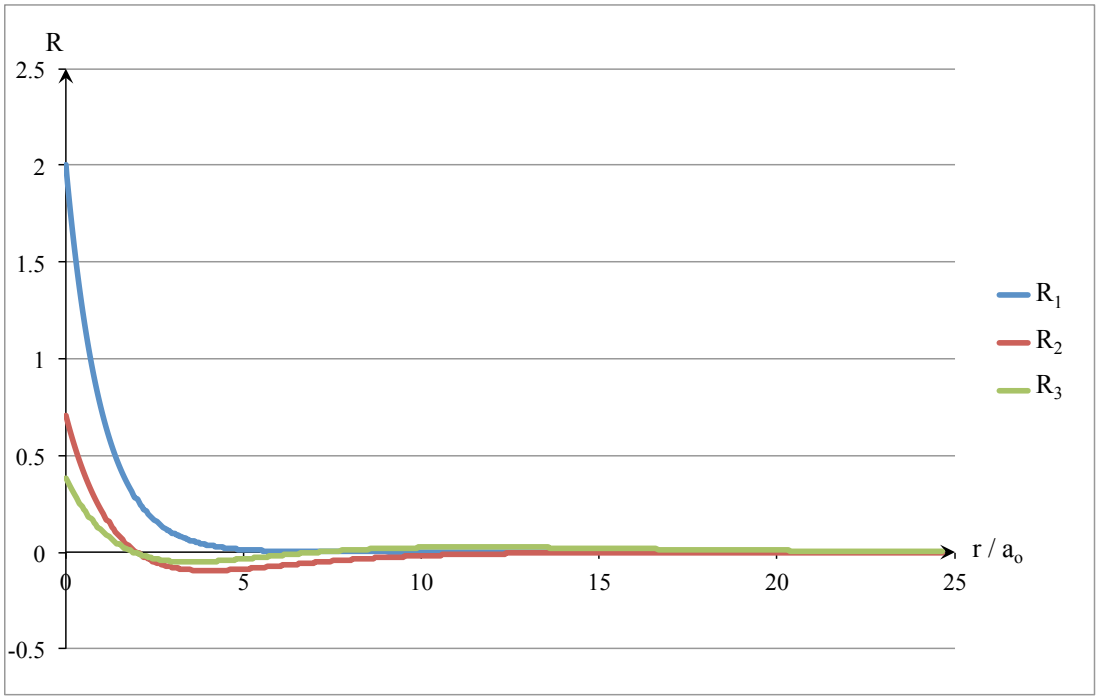

Рис. 2.1. Вычисленные радиальные функции для атома водорода

мости погрешности вычислений от числа полиномов в разложении (1.19) был выбран параметр $\lambda = 5$ для основного состояния. Напомним, что точное решение достигается при $\lambda = 2$. Далее изменялось количество полиномов n в разложении и определялась зависимость невязки δE от n для двух первых стационарных состояний. Данная зависимость представлена в таблице 2.2 и на рисунке 2.3. Вертикальная ось представлена в логарифмическом масштабе. Аппроксимирующая функция является экспоненциальной. Таким обра-

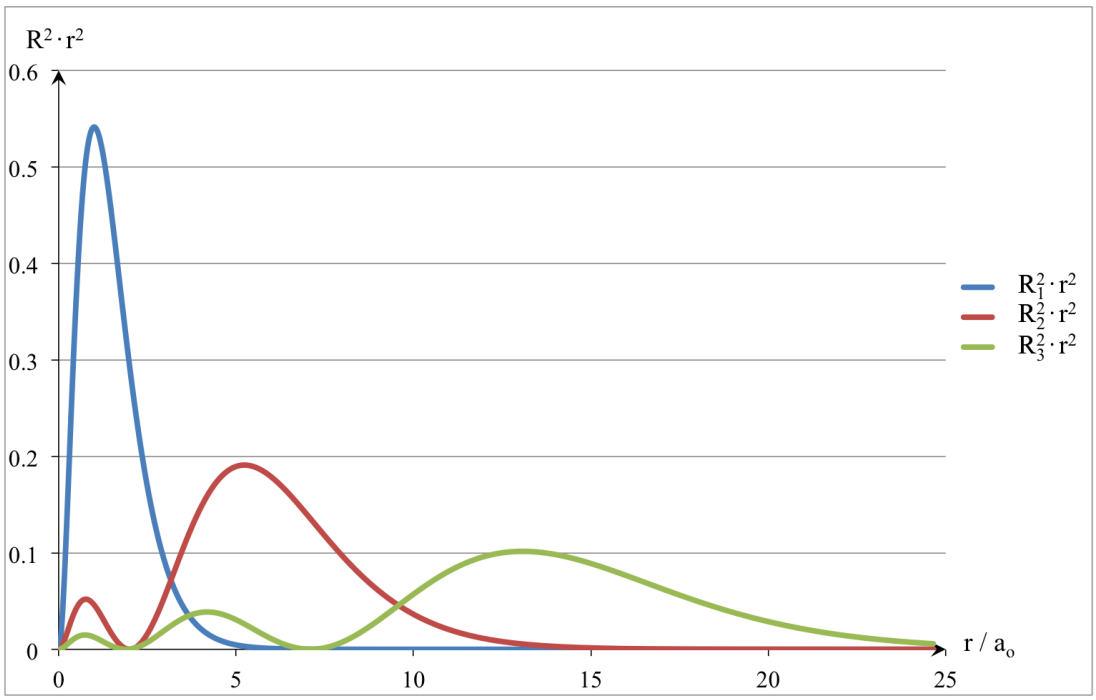

Рис. 2.2. Радиальное распределение вероятности

зом, можно сказать о быстрой сходимости результата к точному решению при увеличении количества полиномов Лагерра.

Также интерес представляет зависимость погрешности от λ. В таблице 2.3 приведены параметр λ, вычисленная энергия и погрешность на каждой итерации при количестве полиномов $n = 10$. Начальное значение интересующего нас параметра было положено равное 0.1. Как видно из таблицы 2.3, погрешность быстро убывает при приближении параметра к оптимальному значению.

Таблица 2.2. Зависимость погрешности вычислений от количества полиномов в разложении

n	1	2	3	4	5	6	7	8	9	10
$\delta E_1, a.e.$	1.12500	0.20371	0.05186	0.01411	0.00379	0.00098	0.00024	0.00006	0.00001	$3.04{\cdot}10^{-6}$
$\delta E_2, a.e.$	0.12500	5.83796	1.60888	0.69738	0.35558	0.19654	0.11382	0.06783	0.04113	0.02517
n	11	12	13	14	15	16	17	18	19	20
$\delta E_1, a.e.$	$6.73{\cdot}10^{-7}$	$1.47{\cdot}10^{-7}$	$3.16{\cdot}10^{-8}$	$6.71{\cdot}10^{-9}$	$1.41{\cdot}10^{-9}$	$2.95{\cdot}10^{-10}$	$6.10{\cdot}10^{-11}$	$1.26{\cdot}10^{-11}$	$2.56{\cdot}10^{-12}$	$5.21{\cdot}10^{-13}$
$\delta E_2, a.e.$	0.01544	0.00946	0.00576	0.00347	0.00207	0.00122	0.00071	0.00041	0.00023	0.00013

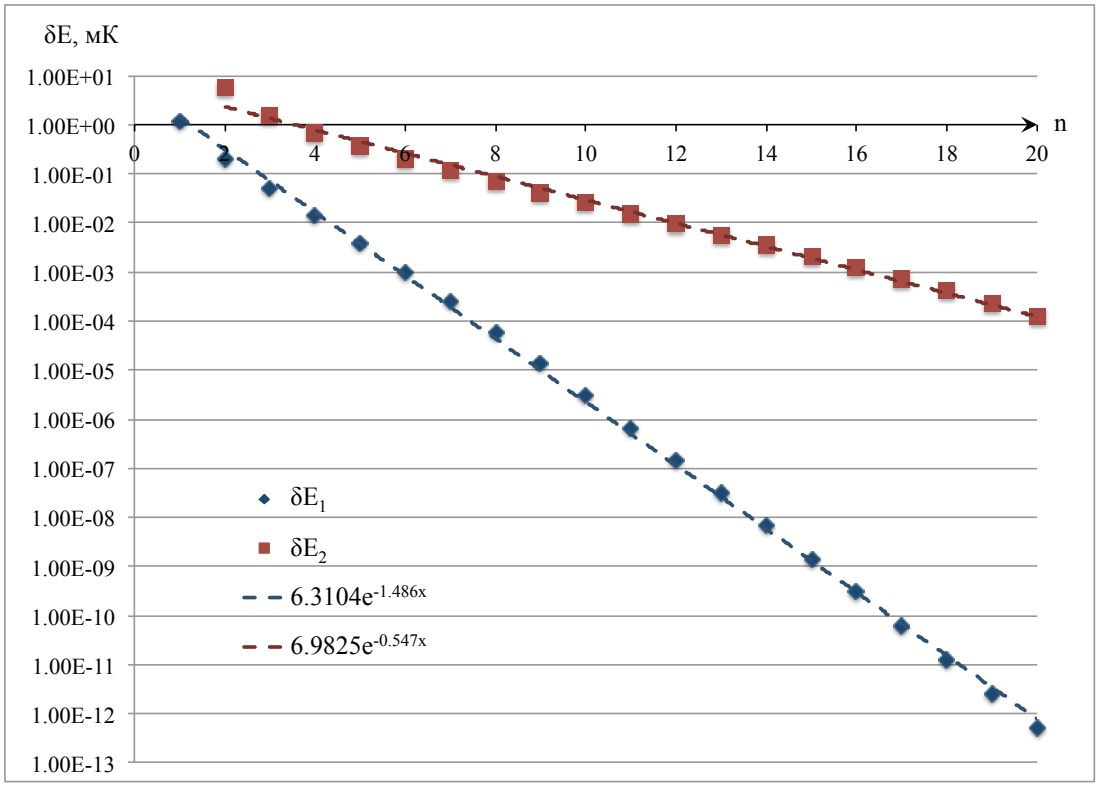

Рис. 2.3. Зависимость погрешности вычислений энергий первого и второго стационарного состояния от количества полиномов в разложении

Таблица 2.3. Результаты работы программы на каждой итерации последовательного приближения параметра λ

№ итерации	λ	$E_1, a.e.$	$\delta E_1, a.e.$
1	0.10000	-0.14647	0.35353
2	1.08249	-0.49987	0.00013
3	1.99974	-0.50000	$-2.22 \cdot 10^{-16}$
4	2.00000	-0.50000	$-4.44 \cdot 10^{-16}$

2.2.2. Димер неона

Далее был рассмотрен димер неона. Напомним, что неон — инертный газ с атомной массой 20.015 а.е. Потенциал димера [2] представлен на рисунке 2.4. Эта задача является относительно простой для численных расчетов, потенциальная яма глубока, энергии связи достаточно велики. Глубина потенциальной ямы равна 42.1 K при радиусе $5.9a_0$. Для неона мы получили два значения энергии связи: $E_1 = -23.7565$ K $= -16.5116$ см$^{-1}$ и $E_2 = -2.6357$ K $= -1.8319$ см$^{-1}$. Также были построены волновые функции, соответсвующие данным значениям энергий (Рис. 2.5). Значения энергий очень близки к значениям, полученным в других работах [2] ($E_1 = -16.512$ см$^{-1}$ и $E_2 = -1.832$ см$^{-1}$). Таким образом, можно утверждать, что метод разложения по полиномам Лагерра успешно работает и для систем, значительно отличающихся от атома водорода.

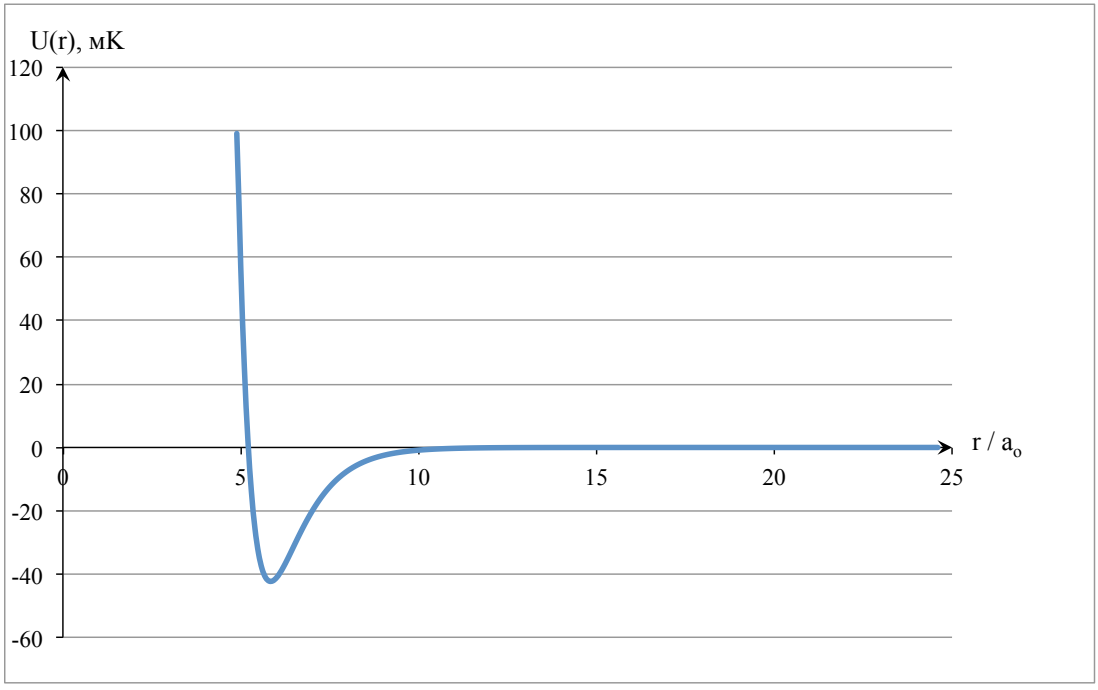

Рис. 2.4. Потенциальная энергия димера неона

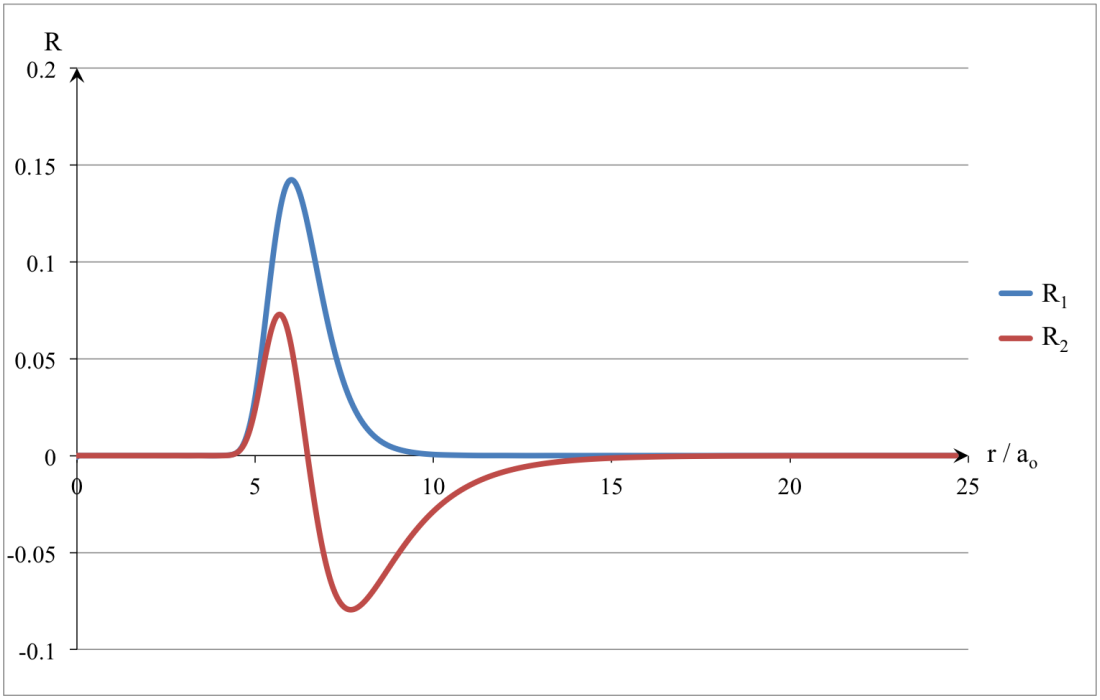

Рис. 2.5. Волновые функции димера неона

2.2.3. Димер гелия

Особый интерес представляет димер гелия в силу очень слабого взаимодействия. С другой стороны, малая энергия связи также приводит к трудностям при численных расчетах. При малых изменениях входных параметров относительное изменение результата велико. Для решения поставленной задачи необходимо описать потенциал взаимодействия. Ниже приведем некоторые обозначения наиболее распространенных потенциалов, которые были использованы:

- HFD-B — R.A. Aziz, F.R.W. McCourt, C.C.K. Wong[33];
- LM2M2 — R.A. Aziz, M.J. Slaman, J. Chem[34];
- TTY — K.T. Tang, J.P. Toennies, C.L. Yiu[35].

Эти потенциалы почти совпадают за исключением значений при радиусе, близком к нулю, где потенциалы велики. Глубина ямы для этих потенциалов порядка -10.9 мК при радиусе $r = 5.6a_0$, эти значения немного варьируются

у разных авторов. В силу того, что потенциал растет слишком быстро при стремлении радиуса к нулю, потенциал TTY был ограничен. Ниже приведен график, соответствующий потенциалу LM2M2 (Рис. 2.6). Расчеты были произведены для трех потенциалов: HFD-B, LM2M2, TTY. В результате были получены энергии связи, приведенные в таблице 2.4. Как и ожидалось, энергия очень мала, связанное состояние может возникнуть только при очень низких температурах. Была построена волновая функция связанного состояния (Рис. 2.7). Для наглядности также представлен потенциал димера гелия.

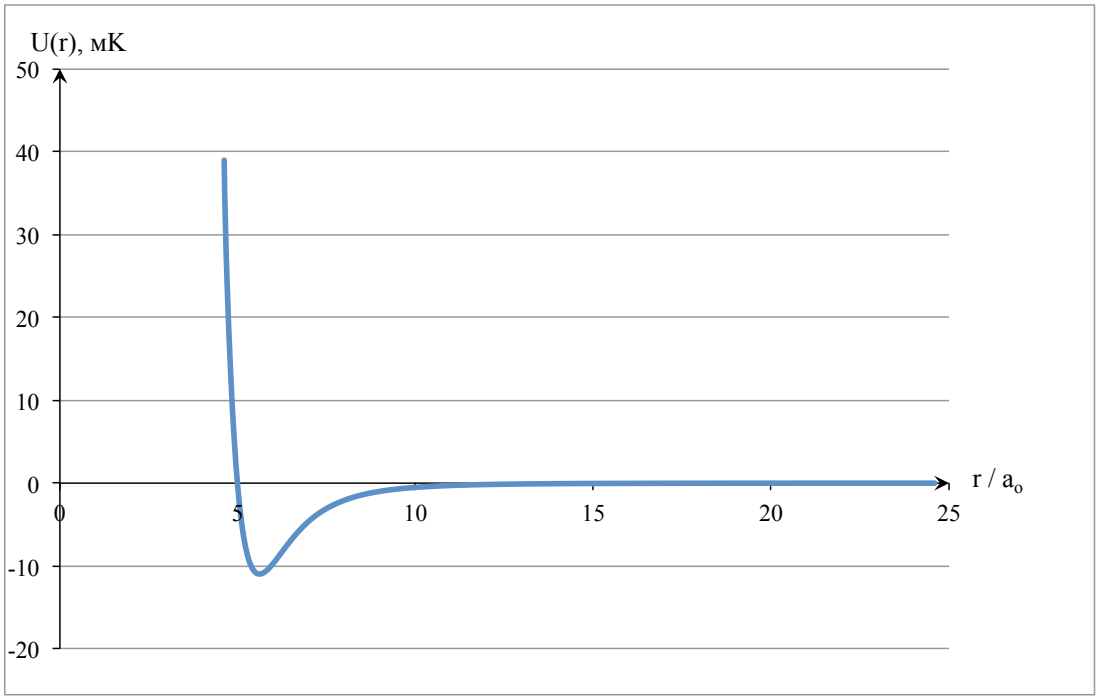

Рис. 2.6. Потенциальная энергия димера гелия

Таблица 2.4. Значения энергий для различных потенциалов в мК

	$HFD-B$	$LM2M2$	TTY
Motovilov A. et. al.[3]	-1.6853	-1.3032	-1.3091
Данная работа	-1.6854	-1.3096	-1.3096

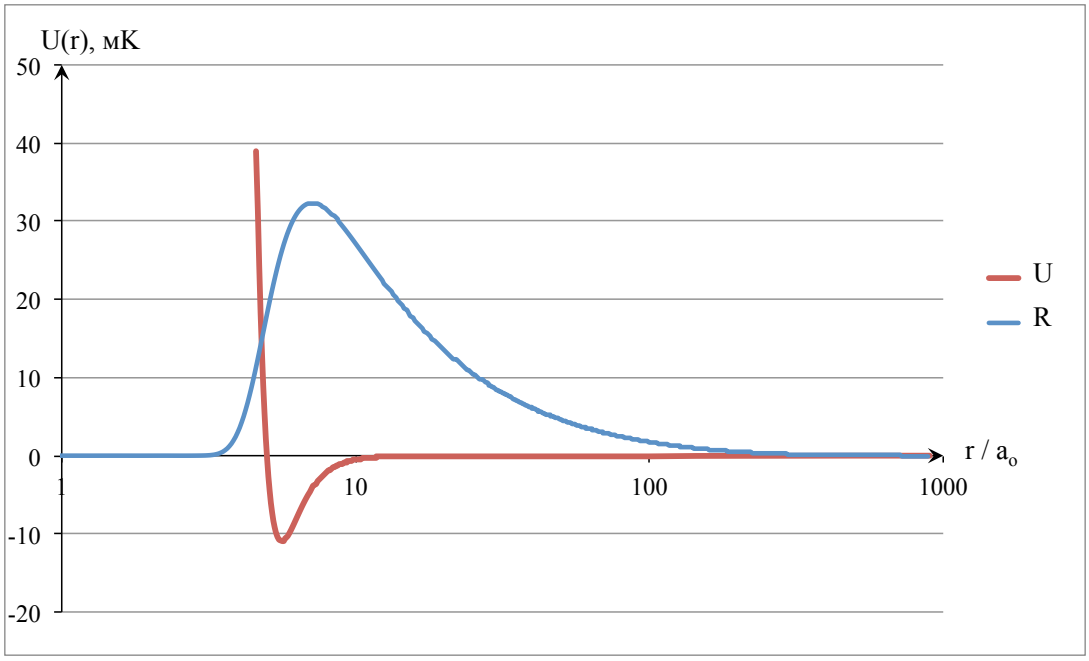

Рис. 2.7. Волновая функция и потенциал димера гелия

Несмотря на то, что потенциал короткодействующий, глубина потенциальной ямы достигается при $r = 2.963$ Å, распространение волновой функции велико и достигает сотни ангстрем.

Следующим шагом был реализован метод дискретных переменных, который описан в разделе 1.3. Данный алгоритм был реализован и проверен на примере димера гелия. Для сравнения также приведем результаты метода, основанного на разложении радиальной функции по присоединенным полиномам Лагерра $L_n^{(2)}(x)$. В таблице 2.5 приведены результаты работы программы при различной размерности матриц n (количестве функций в разложении) для потенциала TTY. Была определена относительная ошибка и время интегрирования, которое занимает подавляющий объем вычислений. За истинное значение была выбрана энергия, полученная ранее $E = -1.3096$ мК при $n = 300$. Здесь и далее время определялась, как наименьшее из 3-5 запусков программы с постоянными параметрами.

По данным таблицы построен график 2.8 зависимости времени вычисления матричных элементов от числа полиномов в разложении в логарифми-

ческом масштабе. Можно сделать вывод, что DVR-метод позволяет ускорить вычисления в десятки раз почти без потери точности.

Таблица 2.5. Сравнение результатов для метода разложения по полиномам Лагерра и метода представления дискретных переменных

	Разложение по $L_n^{(2)}$			DVR			
n	E, мК	δE	t, c	E, мК	δE	t, c	Коэфф. ускорения
10	90.1981	69.76585	0.0023	90.99950	70.37682	0.0004	6.1
25	−9.53053	6.26595	0.0267	−9.59525	6.31529	0.0021	12.4
50	−2.02091	0.54072	0.2067	−2.02118	0.54072	0.0087	23.8
100	−1.28987	0.01662	1.5002	−1.28992	0.01658	0.0434	34.6
150	−1.31096	0.00054	4.8912	−1.31119	0.00037	0.1111	44.0
200	−1.31176	0.00007	11.5434	−1.31206	0.00030	0.2372	48.7
250	−1.31164	0.00002	21.7746	−1.31197	0.00023	0.4208	51.7

2.3. Системы трех частиц

Основным результатом работы является применение метода дискретных переменных в программе ACESPA, разработанной группой ученых и студентов, позволяющей производить вычисления энергий связи квантово-механических систем. После изучения структуры программы и реализации DVR метода работа программы успешно проверена на примере тримера неона. Для этого было взято различное количество функций в разложении по угловой координате для метода, реализованного ранее, и DVR разложения. Обратим внимание, данный метод является модифицированным, алгоритм оптимизирован. Для сравнения приведены также результаты неоптимизированного кода, в котором разложение и интегрирование описано без дополнительных модификаций и производится по полиномам Лежандра. Именно этот код был взят за основу при реализации метода дискретных переменных.

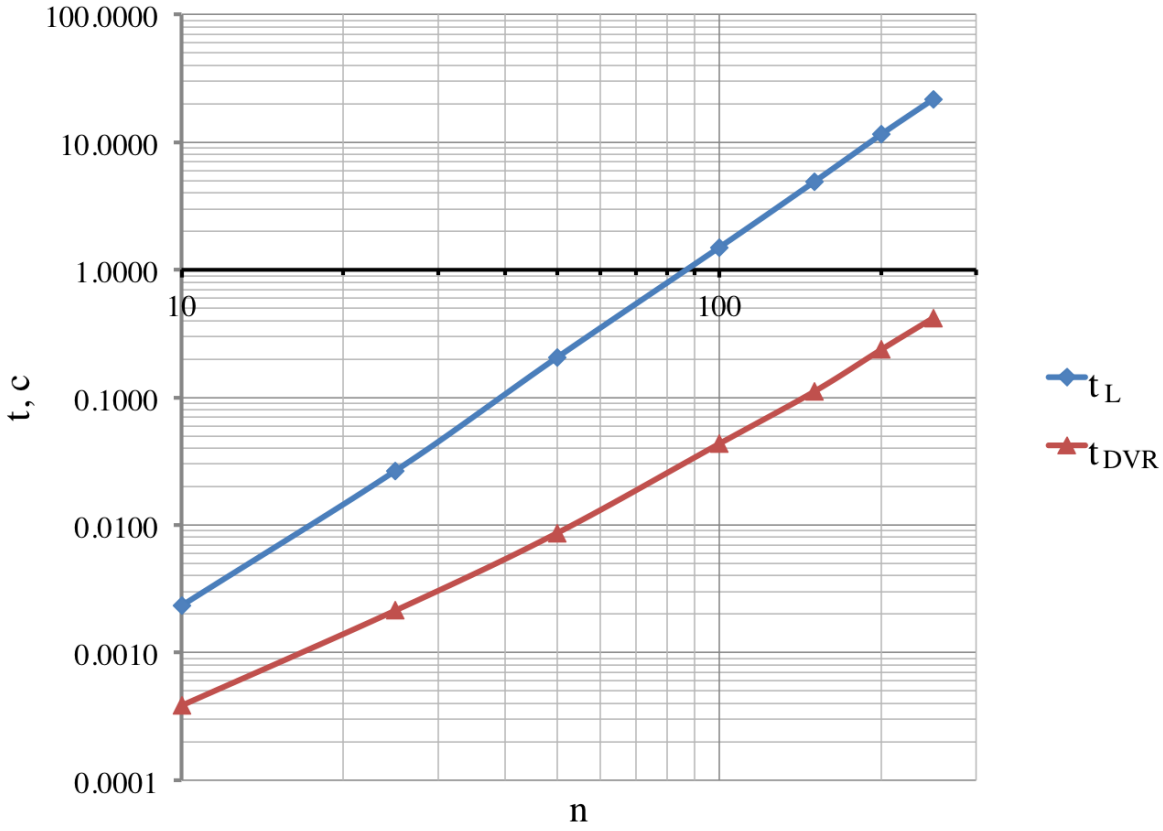

Рис. 2.8. Зависимость времени вычисления матричных элементов от количества полиномов в разложении

2.3.1. Тример неона

В таблице 2.6 приведены абсолютные значения энергий связи тримера неона и относительная ошибка при различном количестве функций n в разложении. Для всех трех методов результат совпал с машинной точностью. Это связано с тем, что задача по нахождению энергий связи тримера неона относительно проста, а потенциал гладкий и хорошо интегрируем. Для наглядности значения приведены для первого, десятого и двадцатого энергетического уровня тримера неона. В качестве верного значения выбиралась энергия связи, вычисленная при $n = 40$. Время вычисления матричных элементов операторов для стандартного, модифицированного и DVR метода значительно отличается. Из таблицы видно, что время вычислений значительно сократилось. Таким образом, можно считать, что поставленная задача

успешно решена. Для наглядности приведены графики сходимости (Рис. 2.9) и зависимости времени вычислений (Рис. 2.10) от количества полиномов в логарифмическом масштабе.

Таблица 2.6. Энергии связи и относительные ошибки при различном количестве функций в разложении

n	3	5	10	15	20	25	30
E_1, см $^{-1}$	59.3785	57.4215	49.3707	50.1440	50.1015	50.1016	50.1016
E_{10}, см $^{-1}$	24.6638	30.4413	29.3888	29.7554	29.7205	29.7204	29.7204
E_{20}, см $^{-1}$	19.3429	20.3997	20.2850	20.2446	20.2423	20.2421	20.2421
δE_1	$1.85{\cdot}10^{-1}$	$1.46{\cdot}10^{-1}$	$1.46{\cdot}10^{-2}$	$8.47{\cdot}10^{-4}$	$2.05{\cdot}10^{-6}$	$1.36{\cdot}10^{-8}$	$1.40{\cdot}10^{-9}$
δE_{10}	$1.70{\cdot}10^{-1}$	$2.43{\cdot}10^{-2}$	$1.12{\cdot}10^{-2}$	$1.18{\cdot}10^{-3}$	$2.79{\cdot}10^{-6}$	$1.40{\cdot}10^{-7}$	$8.08{\cdot}10^{-9}$
δE_{20}	$4.44{\cdot}10^{-2}$	$7.78{\cdot}10^{-3}$	$2.12{\cdot}10^{-3}$	$1.24{\cdot}10^{-4}$	$6.28{\cdot}10^{-6}$	$6.09{\cdot}10^{-7}$	$5.14{\cdot}10^{-8}$

Таблица 2.7. Время работы стандартного $t_{\text{ст.}}$, модифицированного $t_{\text{мод.}}$ метода разложения по полиномам Лежандра и метода дискретных переменных t_{DVR}

n	3	5	10	15	20	25	30
$t_{\text{ст.}}$, с	0.5	1.3	8.5	26.8	61.5	143.1	279.5
$t_{\text{мод.}}$, с	0.4	1.3	7.4	22.5	50.9	98.5	169.9
t_{DVR}, с	0.2	0.6	2.2	5.0	9.3	14.9	22.0
$t_{\text{ст.}}/t_{DVR}$	2.5	2.2	3.9	5.4	6.6	9.6	12.7
$t_{\text{мод.}}/t_{DVR}$	2.0	2.2	3.4	4.5	5.5	6.6	7.7

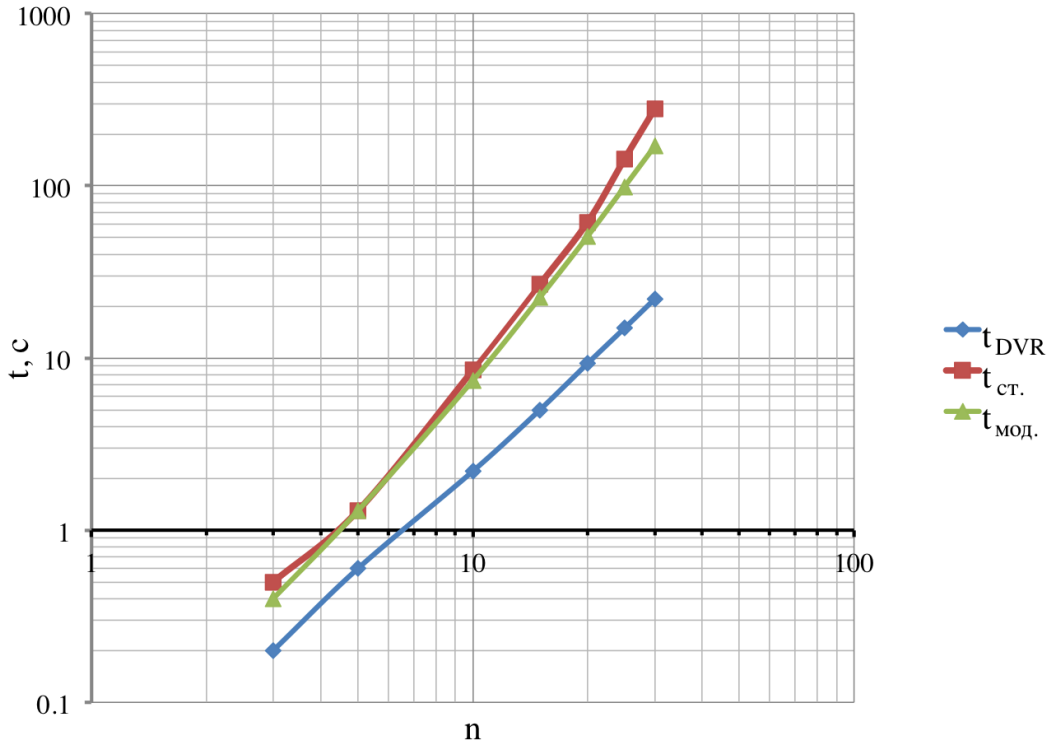

Рис. 2.9. Зависимость времени вычисления матричных элементов от количества полиномов в разложении для различных методов

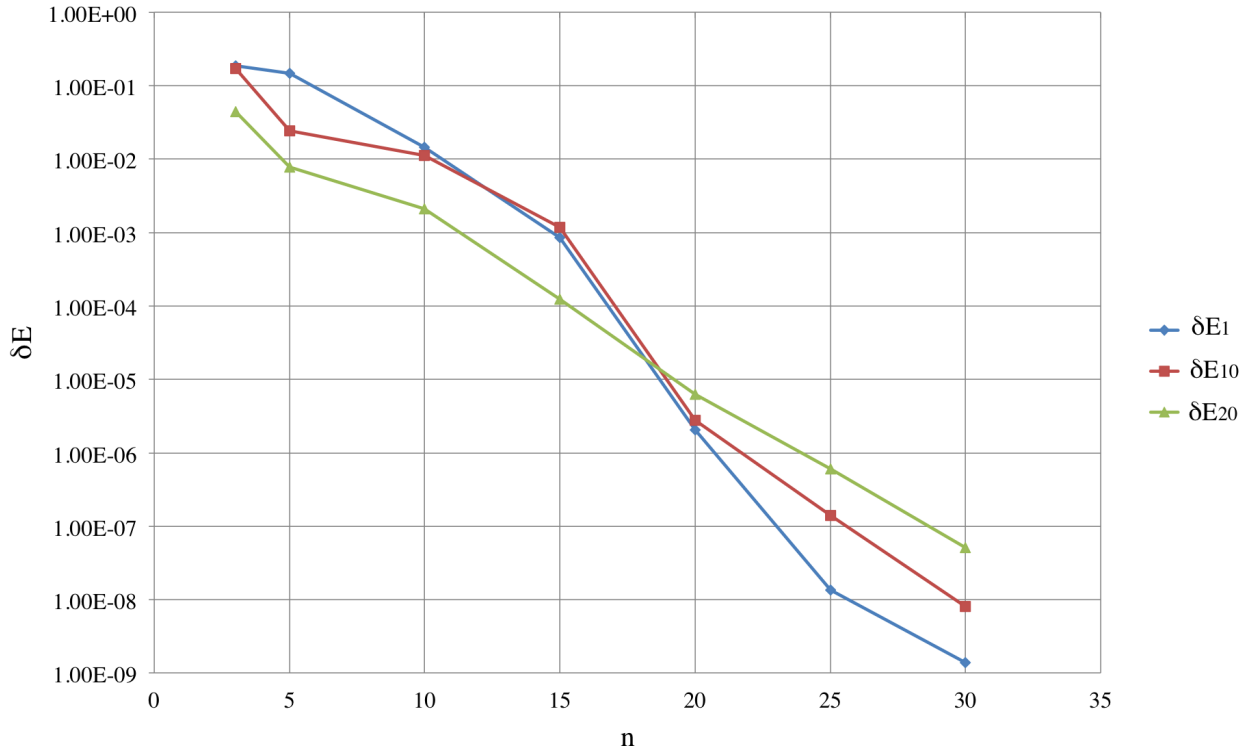

Рис. 2.10. Зависимость относительной ошибки от количества полиномов в разложении

2.3.2. Система литий-гелий

Для того, чтобы исследовать связные состояния, необходимо описать потенциал[31] и параметры системы. Формула расчета и график потенциала приведены в приложении А. Была написана функция, возвращающая потенциал взаимодействия атомов лития и гелия в произвольной точке пространства. В начале были получены энергии связи для молекул ^{6}Li-He и ^{7}Li-He. Здесь ^{6}Li и ^{7}Li — изотопы лития с атомными массами 6.015 а.е. и 7.016 а.е. соответсвенно. Результаты вычислений приведены в таблице 2.8. Также приведены значения энергий, полученные другими авторами. Заметим, что энергия связи для данной пары очень мала. Далее были получены энергии для

Таблица 2.8. Энергия связи молекулы Li-He в см$^{-1}$

	^{6}Li-He	^{7}Li-He
Данная работа	-0.023·10^{-2}	-0.195·10^{-2}
J. Yuan and C. D. Lin[4]	-0.008·10^{-2}	-0.150·10^{-2}
I. Baccarelli et al.[5]	-0.023·10^{-2}	-0.195·10^{-2}

систем, состоящих из изотопов лития и пары гелия. В таблице 2.9 приведены значения, полученные данной работе и результаты других авторов. Заметим, что лучше всего энергии согласуются со значениями представленными в статье[6]. Был проведен анализ времени вычисления матричных элементов при использовании метода дискретных переменных и разложения по полиномам Лежандра для системы ^{7}Li-He$_2$. В таблице 2.10 представлены значения энергий системы ^{7}Li-He$_2$, относительных ошибок и времени вычислений для различного количества функций в разложении. Результаты для ^{6}Li-He$_2$ качественно совпадают. Для вычисления ошибки за истинное значение была принята энергия при $n = 35$, полученная при помощи разложения по полиномам Лежандра. Из полученных данных можно сделать вывод, что работа программы значительно ускорилась без потери точности. Более того, при ма-

Таблица 2.9. Энергия связи системы Li-He_2 в $см^{-1}$

	^{6}Li-He_2	^{7}Li-He_2
Данная работа	-2.62·10^{-2}	-4.07·10^{-2}
J. Yuan and C. D. Lin[4]	-2.18·10^{-2}	-3.18·10^{-2}
I. Baccarelli et al.[5]	-3.61·10^{-2}	-5.10·10^{-2}
E. A. Kolganova [6]	-2.46·10^{-2}	-3.54·10^{-2}

лом количестве функций DVR метод дает более точный результат, ошибка вычислений значительно меньше.

Таблица 2.10. Энергии связи ^{7}Li-He_2 и относительные ошибки при различном количестве функций в разложении

n	5	10	15	20	25
Разложение по полиномам Лежандра					
E_1, $см^{-1}$	-3.44·10^{-7}	-3.00·10^{-2}	-3.94·10^{-2}	-4.07·10^{-2}	-4.12·10^{-2}
δE_1	1.00	2.75·10^{-1}	4.67·10^{-2}	1.49·10^{-2}	2.26·10^{-3}
t, с	6.0	36.6	119.6	287.9	553.5
Метод представления дискретных переменных					
E_1, $см^{-1}$	-4.81·10^{-2}	-4.67·10^{-2}	-4.19·10^{-2}	-4.15·10^{-2}	-4.14·10^{-2}
δE_1	1.63·10^{-1}	1.31·10^{-1}	1.41·10^{-2}	5.10·10^{-3}	1.96·10^{-3}
t, с	2.2	8.2	19.1	35.2	65.8
Коэффициент ускорения					
	2.7	4.5	6.3	8.2	8.4

2.3.3. Тример гелия

Последней рассматриваемой системой является тример гелия. Были также получены значения энергий связи и времени вычислений. Данные приведены в таблице 2.11. Максимально точное значение, которое удалось получить $E_1 = -0.08246$ см$^{-1} = -0.11792$ K, что согласуется с результатами, полученными другими авторами [3, 7]. В то же время, благодаря использованию метода дискретных переменных удалось сократить время вычислений почти в 17 раз.

Таблица 2.11. Энергии связи и их относительные ошибки при различном количестве функций в разложении

n	10	20	30	40
Разложение по полиномам Лежандра				
E_1, см$^{-1}$	-3.13·10^{-3}	-7.05·10^{-2}	-7.99·10^{-2}	-8.18·10^{-2}
δE_1	9.62·10^{-1}	1.42·10^{-1}	2.80·10^{-2}	5.34·10^{-3}
t, с	12.3	84.4	277.9	662.9
Метод представления дискретных переменных				
E_1, см$^{-1}$	-1.29·10^{-1}	-9.72·10^{-2}	-8.35·10^{-2}	-8.26·10^{-2}
δE_1	5.69·10^{-1}	1.81·10^{-1}	1.59·10^{-2}	4.21·10^{-3}
t, с	2.4	9.5	21.6	40.0
Коэффициент ускорения				
	5.1	8.9	12.9	16.6

Заключение

Поставленную цель можно считать успешно достигнутой. Разработан и реализован алгоритм, который позволяет значительно сократить время вычислений связных состояний систем нескольких частиц. В ходе роботы выполнены следующие задачи:

- Реализована программа, позволяющая определять энергии связи для пары частиц при помощи разложения по присоединенным полиномам Лагерра и метода дискретных переменных. Программа успешно проверена на атоме водорода, вычисленные значения совпадают с теоретическими.

- Получены энергии связи пар частиц Ne_2 и He_2, вычисленные при помощи метода представления дискретных переменных и классического спектрального метода. Использование дискретных переменных позволило укорить вычисления в десятки раз.

- Модифицирована программа расчета квантово-механических систем, DVR метод был успешно применен и протестирован на тримере неона.

- Получены энергии связи систем, состоящих из нескольких частиц Ne_3, ^{6}Li-He, ^{7}Li-He, ^{6}Li-He_2, ^{7}Li-He_2 и He_3. Благодаря применению метода представления дискретных переменных время вычислений значительно сократилось без потери точности.

Промежуточные результаты данной работы были успешно представлены на международной студенческой конференции “Science and Progress”, проходившей в Санкт-Петербурге в октябре 2016 года[36].

В дальнейшем планируется:

- Реализовать алгоритм построения DVR-функций и вычисления матричных элементов, используя полиномы Чебышева.

- Модифицировать код программы для исследования состояний с ненулевым полным угловым моментом.
- Рассмотреть комплексную задачу для резонансных состояний и процессов рассеяния.

Список литературы

1. Suzuki Y., Varga K. Stochastic Variational Approach to Quantum-Mechanical Few-Body Problems. Germany: Springer, 1998.
2. Salci M., Levin S. B., Elander N., Yarevsky E. A. A theoretical study of the rovibrational levels of the bosonic van der Waals neon trimer // J. Chem. Phys. 2008. Vol. 129. P. 134304.
3. Motovilov A., Sandhas W., Sofianos S., Kolganova E. Binding Energies and Scattering Observables in the $^4He_3$ Atomic System // Eur. Phys. J. D. 2001. Vol. 13. P. 34–41.
4. Yuan J., Lin C. D. Weakly bound triatomic He_2Li and He_2Na molecules // J. Phys. B: At. Mol. Opt. Phys. 1998. Vol. 31. P. 647–645.
5. Baccarelli I., Delgado-Barrio G., Gianturco F. A. et al. The weakly bound ground state of the $LiHe_2$ triatomic system // Phys. Chem. Chem. Phys., 2000, 2, 4067-4073. 2000. Vol. 2. P. 4067–4073.
6. Kolganova E. Weakly Bound Li He_2 Molecules // Few-Body Syst. 2017. Vol. 58. P. 57.
7. Esry B. D., Lin C. D., Greene C. H. Adiabatic Hyperspherical Study of the Helium Trimer // The American Physical Society. 1996. Vol. 54. P. 394–401.
8. Efimov V. Energy Levels Arising from Resonant Two-Body Forces in a Three-body system // Physics Letters. 1970. Vol. 33B. P. 563–564.
9. E. Braaten H.-W. H. Universality in few-body systems with large scattering length // Phys. Rep. 2006. Vol. 428. P. 259–390.
10. Gonzalez-Lezana T., Rubayo-Soneira J., Miret-Artés S. et al. Comparative configurational study for He, Ne, and Ar trimers // The Journal of Chemical Physics. 1999. Vol. 110, no. 18. P. 9000–9010.
11. Rick S. W., Lynch D. L., Doll J. D. A variational Monte Carlo study of argon, neon, and helium clusters // The Journal of Chemical Physics. 1991. Vol. 95, no. 5. P. 3506–3520.

12. Simos T., Williams P. A finite-difference method for the numerical solution of the Schrodinger equation // J. Comp. and Appl. Math. 1997. Vol. 79. P. 189 – 205.
13. Light J., Jr. C. T. Discrete-Variable representations and their utilization // Advances in Chemical Physics. John Wiley Sons Inc., 2007. P. 263–310.
14. Lill J., Parker G., J.C.Light. The discrete variable–finite basis approach to quantum scattering // J. Chem. Phys. 1986. Vol. 85, no. 2. P. 900 – 910.
15. Blackmore R., Shizgal B. Discrete-ordinate method of solution of Fokker-Planck equations with nonlinear coefficients // Phys. Rev. A. 1985. Vol. 31. P. 1855–1868.
16. Bacic Z., Light J. Highly excited vibrational levels of "floppy" triatomic molecules: A discrete variable representation—Distributed Gaussian basis approach // J. Chem. Phys. 1986. Vol. 85, no. 8. P. 4594–4604.
17. Bacic Z., Whitnell R., Brown D., Light J. Localized representations for large amplitude molecular vibrations // Comp. Phys. Comm. 1988. Vol. 51. P. 35 – 47.
18. Light J. C., Whitnell R. M., Parkand T. J., Choi S. E. // Supercomputer Algorithms for Reactivity, Dynamics and Kinetics of Small Molecules, Ed. by A. Lagana. NATO ASI Series C, 1989. Vol. 277. P. 187.
19. Manolopoulos D., Wyatt R. Quantum scattering via the log derivative version of the Kohn variational principle // Chemical Physics Letters. 1988. Vol. 152. P. 23 – 32.
20. Rescigno T. N., McCurdy C. W. Numerical grid methods for quantum-mechanical scattering problems // Phys. Rev. A. 2000. Vol. 62. P. 032706.
21. Elander N., Yarevsky E. Finite-element calculations of the antiprotonic helium atom including relativistic and QED corrections // Phys. Rev. A. 1997. Vol. 56. P. 1855–1864.
22. Yarevsky E. A. Parallel Numerical Calculations of Quantum Trimer Systems // LNCS 7125, Mathematical Modeling and Computational Science,

Gheorghe A., Buša J., and Hnatic, M. (Eds.). 2012. P. 290–295.

23. Canuto C., Quarteroni A., Hussaini M. Y., Zang T. A. Spectral Methods. Germani: Springer, 2006.
24. Gottlieb D., Orszag S. A. Numerical Analysis of Spectral Methods: Theory and Applications. Montpelier: Capital City Press, 1977.
25. Salci M. A Theoretical Study of Atomic Trimers in the Critical Stability Region: Thesis for the deg. of ph.d. in phys. / Department of Physics, Stockholm University. 1996.
26. Konovalova D., I.Bray. Calculation of Electron-Impact Ionization using the J-matrix Method // Phys. Rev. A. 2010. Vol. 82.
27. Ландау Л. Д., Лифшиц Е. М. Квантовая механика (нерелятивистская теория). Москва: Наука, 1989.
28. Abramowitz M. Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables,. Dover Publications, Incorporated, 1974. ISBN: 0486612724.
29. Burkardt J. Laguerre Polynomials. URL: `http://people.sc.fsu.edu/~jburkardt/cpp_src/laguerre_polynomial/laguerre_polynomial.html`.
30. S. B. ALGLIB. URL: `http://www.alglib.net`.
31. Cvetko D., Lausi A., Morgante A. et al. A new model for atom–atom potentials // J. Chem . Phys. 1994. Vol. 100. P. 2052.
32. Миникин В.И., Симкин Б.Я., Миняев Р.М. Теория строения молекул. Москва: Высшая школа, 1979.
33. Aziz R. A., McCourt F. R. W., Wong C. C. K. A new determination of the ground state interatomic potential for He_2 // Mol. Phys. 1987. Vol. 61. P. 1487–1511.
34. Aziz R., Slaman M. An examination of ab initio results for the helium potential energy curve // J. Chem. Phys. 1991. Vol. 84. P. 8047–8053.
35. Tang K., Toennies J., Yiu C. Accurate analytical He-He van der Waals po-

tential based on perturbation theory // Phys. Rev. Lett. 1995. Vol. 74. P. 1546–1549.

36. Timoshenko V. Quantum Few-Body Problem // Conference Abstracts of International Student Conference “Science and Progress”. 2016. P. 219.

Приложение А

Потенциал взаимодействия Li-He

Потенциал взаимодействия пары частиц Li-He описывается формулами[31]

$$v(r) = \frac{C_6}{120}\left(\frac{b}{3}\right)^6 (ae^{-br} - \chi e^{-(2/3)br} - e^{-(1/3)br}), \quad \text{если} br \leqslant 16.6, \qquad (A.1)$$

$$v(r) = \frac{C_6}{120}\left(\frac{b}{3}\right)^6 ae^{-br} - \frac{C_6}{r^6 - Q^2 r^4}, \quad \text{если} br \geqslant 16.6, \qquad (A.2)$$

где χ определяется непрерывностью потенциала $v(r)$ при br $= 16.6$, а параметры $C_6 = 22.5,\ Q = 7.1,\ b = 1.06,\ a = 4700.$

На рисунке А.1 построен график данного потенциала.

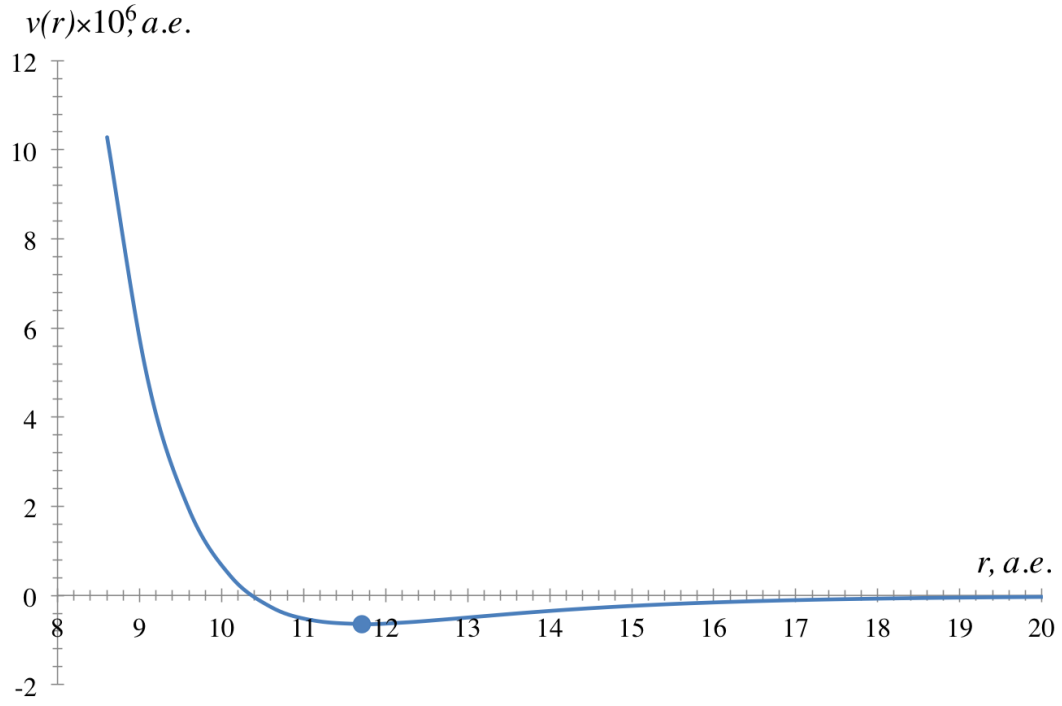

Рис. А.1. Зависимость потенциала от расстояния между частицами пары Li-He

Глубина потенциальной ямы равна $v_{min} = -6.42 \cdot 10^{-6}$ а.е. при радиусе $r_{min} = 11.71$ а.е.

Приложение Б

Программа для исследования систем двух частиц

Код вычисления потенциалов

```
double Gaussian(double r)
{
    double const bohr=0.5291772083;
    double x = r;// / bohr;
    return -1.227 * exp(-pow((x/10.03),2.0));
}
double HFDB(double r)
{
    double const bohr=0.5291772083;
    double xe = (r / 2.963) * bohr;
    double xe2i = pow(xe,-2.0);
    double vf1 = 1.0;
    if (std::abs(xe)<=1.4826)
    {
        vf1 = exp(-pow((1.4826/xe - 1.0),2.0));
    }
    double vp1 = 184431.01*exp(-10.43329537*xe-2.27965105*xe*xe)
    -vf1*(1.36745214+(0.42123807+0.17473318*xe2i)*xe2i)*pow(xe2i,3.0);

    return 10.948 * vp1;
}
double molhe2pot_neon_morse (double r)
{
    //routine that computes Neon dimer Morse potential
    double const bohr=0.5291772083;
    double const D=1.337731704e-4, alpha=2.088*bohr;
    double const r_e=3.091/bohr;
    double tmp;
    tmp = exp(-alpha*(r-r_e));
    return D * (pow(tmp,2.0)-2.0*tmp);

}
double LM2M2 (double r)
{
    double const bohr=0.5291772083, ksi1=1.003535949, ksi2=1.454790369;
    double xe = (r / 2.9695) * bohr;
    double vf1 = 1.0;
    if (xe <= 1.4088) vf1 = exp(-pow((1.4088/xe - 1.0),2.0));
        double vp1 = 189635.353*exp(-10.70203539*xe-1.90740649*xe*xe)
    -vf1*(1.34687065+0.41308398/xe/xe+0.17060159/pow(xe,4))/pow(xe,6);
    double vf2 = 0.0;
        if((ksi1 <= xe) && (xe <= ksi2))
```

```
            vf2=0.0026*(sin(2*M_PI*(xe-ksi1)/(ksi2-ksi1)-0.5*M_PI)+1.0);
    return   10.97 * (vp1 + vf2);
}
double  TTY(double r)
{
// Temporary to exclude huge values at origin:
    double const beta=1.3443, vp=7.0/(2.0*beta)-1.0;
    double rt = fmax(r,0.22);
    double xe = rt;
    double vf1=7.449*pow(xe,vp)*exp(-2.0*beta*xe);
    double bx = xe*(2.0*beta-vp/xe);
    double ebx = exp(-bx);
    double cn[13], av[25], fn[13];
    double ak, ak1, sk, vf2;
    for (int n; n <= 12; n++)
        cn[n] = fn[n] = 0.0;
    cn[3]=1.461;
    cn[4]=14.11;
    cn[5]=183.5;
    for (int n = 6; n <= 12; n++)
        cn[n]=pow((cn[n-1]/cn[n-2]),3.0)*cn[n-3];
    if(fabs(bx)>vp)
    {
        av[0] = 1.0;
        for (int n = 1; n <= 24; n++)
        {
            av[n]=av[n-1]*bx/double(n);
        }
        fn[1]=av[0]+av[1]+av[2];
        for (int n = 2; n <= 12; n++)
            fn[n] = fn[n-1]+av[2*n-1]+av[2*n];
        for (int n = 1; n <= 12; n++)
            fn[n] = 1.0-ebx*fn[n];
    }
    else
        for (int n = 1; n <= 12; n++)
        {
            ak = bx/(fact(2*n+1));
            ak1 = fmin(1.0,fmax(fabs(ak),1e-30));
            sk = ak;
            for (int j=2; j <= 100; j++)
            {
                ak = ak * bx / double(2*n+j);
                sk = sk + ak;
                if(fabs(ak) <= 1e-15*ak1) break;
            }
            fn[n] = ebx * pow(bx,(2.0*n)) * sk;
        }
    vf2 = 0.0;
    for (int n = 3; n <= 12; n++)
    {
```

```
        vf2 = vf2 + cn[n]*fn[n]/pow(xe,2.0*n);
    }
    return 315766.2067*(vf1 - vf2);
}
```

Код main-файла

```
int main (int argc, char * const argv[])
{
    int n = 100; // number of polynomials for decomposition (n <= 200)
    int m = n; // number of points for Gauissian quadrature
    unit::m = 0.5 * 4.0026 * 1822.77261; //mass parameter
    unit::energy = 3.1668153e-6; //from kelvin to hartree (3.1668153e-6)
    int number_of_eigenvalue = 1; //eigenvalues are in ascending order, the first is 1
    double lambda = 2; // x = lambda*r (lambda = 2*sqrt(-2*E*m) for n-th eigenfunction of radial equ
    double T1[n][n];
    double V1[n][n];
    double S1[n][n];
    double H1[n][n];
    double x[m]; // roots of the n-th Laguerre polynomial
    double x1[1000];
    double w[m];
    double *l2;
    double *dl2;
    double *dl0;
    double *dphy;
    double E;
    print_potenrial(4);
    lambda = 2.0*sqrt(2.0*1.94684857e-05*unit::m);
    alglib::real_1d_array eigenValues;
    alglib::real_2d_array eigenVectors;
    alglib::real_2d_array H;
    alglib::real_2d_array Y;
    alglib::real_2d_array YT;
    alglib::real_2d_array S;
    alglib::real_2d_array V;
    alglib::real_2d_array T;
    alglib::real_2d_array T_dvr;
    alglib::real_2d_array V_dvr;
    alglib::real_2d_array H_dvr;

    alglib::ae_int_t info;
    alglib::matinvreport rep;

    eigenValues.setlength(n);
    eigenVectors.setlength(n, n);
    Y.setlength(n, n);
    YT.setlength(n, n);
    S.setlength(n, n);
```

```
T_dvr.setlength(n, n);
T.setlength(n, n);
V_dvr.setlength(n, n);
H_dvr.setlength(n, n);
V.setlength(n, n);

H.setlength(n, n);

l_quadrature_rule(m, x, w);

dphy = new double[n*m];

dl0 = lf_function_derevative(m, n, 0.0, x);
//associated Laguerre polynomials Lf(n,2,x)

unsigned int start = clock();

l2 = lf_function(m, n, 2.0, x);
dl2 = lf_function_derevative(m, n, 2.0, x);

for (int i = 0; i < n; i ++)
{
    for (int k = 0; k < m; k++)
    {
        l2[i*m+k] = 1.0/sqrt((i+1)*(i+2))*l2[i*m+k];
        dl2[i*m+k] = 1.0/sqrt((i+1)*(i+2))*dl2[i*m+k];
    }
}
for (int i = 0; i < n; i ++) //filling the matrices
{
    for (int j = 0; j < n; j ++)
    {
        V[i][j] = 0.0;
        T[i][j] = 0.0;
        for (int k = 0; k < m; k++)
        {
            V[i][j] += w[k]*l2[i*m+k]*l2[j*m+k]*U(x[k]/lambda)*x[k]*x[k];
            T[i][j] += w[k]*(0.5*l2[i*m+k] - dl2[i*m+k])*(0.5*l2[j*m+k] - dl2[j*m+k])*x[k]*x[k];
        }
        H[i][j] = 0.5*lambda*lambda*T[i][j]+V[i][j];
    }
}
alglib::smatrixevd(H, n, 1, 1, eigenValues, eigenVectors);
cout << "n = " << n << endl;
cout << "L2: " << "E = " << eigenValues[number_of_eigenvalue-1]/unit::m/unit::energy*1e3 << "; "
cout << "time = " << float(clock()-start)/CLOCKS_PER_SEC << endl;

start = clock();

//dvr-functions
for (int i = 0; i < n; i ++)
```

```
    {
        for (int k = 0; k < n; k ++)
        {
            if (k == i) { dphy[i*m+i] = -0.5*abs(dl0[n*m+i])/pow(x[i],1.5);}
            else { dphy[i*m+k] = dl0[n*m+k]/(x[k]-x[i])/sqrt(x[i])*(1-2*signbit(dl0[n*m+i]));}
        }
    }
    for (int i = 0; i < n; i ++) //filling the matrices
    {
        for (int j = 0; j < n; j ++)
        {
            T_dvr[i][j] = 0.0;
            V_dvr[i][j] = U(x[i]/lambda)*(i==j);
            for (int k = 0; k < m; k++)
            {
                T_dvr[i][j] += (sqrt(w[k])*x[k]*dphy[i*m+k])*(sqrt(w[k])*x[k]*dphy[j*m+k]);
            }
            H_dvr[i][j] = 0.5*lambda*lambda*T_dvr[i][j]+V_dvr[i][j];
        }
    }

    alglib::smatrixevd(H_dvr, n, 1, 1, eigenValues, eigenVectors);

    cout << "DVR: " << "E = " << eigenValues[number_of_eigenvalue-1]/unit::m/unit::energy*1e3 << ";
    cout << "time = " << float(clock()-start)/CLOCKS_PER_SEC << endl;

    for (int i = 0; i < n; i ++) //filling the matrices
    {
        for (int j = 0; j < n; j ++)
        {
            T1[i][j]=T_dvr[i][j];
            H1[i][j]=H_dvr[i][j];
            V1[i][j]=V_dvr[i][j];
        }
    }
    return 0;
}
void matrixpower (int n, const alglib::real_2d_array &a, double p)
{
    alglib::real_2d_array U_svd;
    alglib::real_2d_array VT_svd;
    alglib::real_1d_array W_svd;
    alglib::real_2d_array W_svd_2d;

    U_svd.setlength(n, n);
    VT_svd.setlength(n, n);
    W_svd_2d.setlength(n, n);
    W_svd.setlength(n);

    alglib::rmatrixsvd(a, n, n, 2, 2, 1, W_svd, U_svd, VT_svd);
    for (int i=0; i<n ; i++)
```

```
    {
        W_svd_2d[i][i]=pow(W_svd[i], p);
    }
    alglib::rmatrixgemm(n, n, n, 1, U_svd, 0, 0, 1, W_svd_2d, 0, 0, 0, 0, a, 0, 0);
    alglib::rmatrixgemm(n, n, n, 1, a, 0, 0, 0, VT_svd, 0, 0, 1, 0, a, 0, 0);
};
```

Приложение B

Программа для исследования систем нескольких частиц частиц

Код вычисления потенциалов

```
! Contains the potential which is used in the ongoing calculations.
! Every new potential should be copied into (or replaced by) this file.
!  The potential is calculated in various combimations:
!  ITYPE=0         sum of all three potentials
!  ITYPE=1,2,3     potential V(x1), V(x2), V(x3) resp.
!  ITYPE=4         sum of potentials V(x1)+V(x3)
!  ITYPE=5         sum of potentials V(x2)+V(x3)
!
MODULE POTPOT
use constants
implicit none
private
!    the system under investigation
     character*20 :: system_invest1 = 'HelDimer'
     character*20 :: system_invest2 = 'HelTrimer'
     character*20 :: system_invest3 = 'Li6He'
     character*20 :: system_invest4 = 'Li7He'

     logical, save :: firstcall=.true.
!
public dpotpot,zpotpot,lihepot
!
contains
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!C
     double precision function dpotpot(r1,r2,r3,ITYPE,nw)
!  The d.real values of zpotpot().
     implicit none
     double precision, intent(IN) :: r1,r2,r3
     integer, intent(IN) :: ITYPE
     integer, intent(IN), optional :: nw
     dpotpot = zpotpot(dcmplx(r1),dcmplx(r2),dcmplx(r3),ITYPE,nw)
     return
     end function dpotpot
!C
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!C
     complex(8) function zpotpot(r1,r2,r3,ITYPE,nw)
!c  This function calculates potential for a system
!c  depending on interparticle distances (see usepot).
!c  complex(8) variant.
```

```
      use usecom
      use SCATTVARS, ONLY: Rsplt
      implicit none
      complex(8) r1,r2,r3,v1,v2,v3
      integer, intent(IN) :: ITYPE
      integer, intent(IN), optional :: nw
!
      if(firstcall)then
          if( (system /= system_invest1) .and. (system /= system_invest2) .and. &
              (system /= system_invest3) .and. (system /= system_invest4)) then
              write(*,*) 'The wrong potential is used for the system ',system
              stop 'The wrong potential is used for the system '
          endif
          firstcall = .false.
      endif
      v1 = convfac*zheltrim(r1,2)
      v3 = convfac*zheltrim(r3,2)
      if (system == 'Li6He' .or. system == 'Li7He') then
      v2 = convfac * lihepot(real(r2))
      endif
      if(ITYPE == 0)then
          zpotpot=v1+v2+v3
      elseif(ITYPE == 1)then
          zpotpot=v1
      elseif(ITYPE == 2)then
          zpotpot=v2
      elseif(ITYPE == 3)then
          zpotpot=v3
      elseif(ITYPE == 4)then
          zpotpot=v1+v3
      elseif(ITYPE == 5)then
          zpotpot=v2+v3
      endif
      return
      end function zpotpot
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
!
complex(8) function zheltrim(r,i)
!c Calculates different potentials for helium trimer:
!c 0 - Gaussian type potential
!c 1 - HFD-B potential
!c 2 - LM2M2 potential
!c 3 - TTY potential
      implicit none
      double precision, parameter :: aua=1.d0/0.52917d0
      double precision, parameter :: beta=1.3443d0, vp=7.d0/(2.d0*beta)-1.d0, &
        ksi1=1.003535949d0, ksi2=1.454790369d0
      complex(8) r,rt,xe,xe2i,vp1,vf2,vf1,bx,ebx,fn(12),av(0:24),ak,sk
      double precision ak1,fact,cn(12)
      integer i,n,j
```

```
!c
      if (i.eq.0) then
        xe = r / bohr
!c  Gaussian
        zheltrim = -1.227d0 * exp(-(xe/10.03d0)**2)
!c  Exponential
!c          zheltrim = -3.909d0 * exp(-xe/(2.1*4.117d0))
      elseif (i.eq.1)then
!c r - in \o A
        xe = (r / 2.963d0) * bohr
        xe2i = 1.d0/xe**2
        vf1 = 1.d0
        if (abs(xe) .le. 1.4826d0) then
            vf1 = exp(-(1.4826d0/xe - 1.d0)**2)
        endif
        vp1 = 184431.01d0*exp(-10.43329537d0*xe-2.27965105d0*xe*xe)      &
          -vf1*(1.36745214d0+(0.42123807d0+0.17473318d0*xe2i)*xe2i)*xe2i**3
        zheltrim = 10.948d0 * vp1
      elseif (i.eq.2) then
!c r - in \o A
        xe = (r / 2.9695d0) * bohr
        xe2i = 1.d0/xe**2
        vf1 = 1.d0
        if (abs(xe) .le. 1.4088d0) then
            vf1 = exp(-(1.4088d0/xe - 1.d0)**2)
        endif
        vp1 = 189635.353d0*exp(-10.70203539d0*xe-1.90740649d0*xe*xe)     &
          -vf1*(1.34687065d0+0.41308398d0*xe2i+0.17060159d0*xe2i**2)*xe2i**3
        vf2 = 0.d0
        if((ksi1.le.abs(xe)) .and. (abs(xe).le.ksi2)) then
        vf2=0.0026d0*(sin(2*pi*(xe-ksi1)/(ksi2-ksi1)-0.5d0*pi)+1.d0)
        endif
        zheltrim = 10.97d0 * (vp1 + vf2)
      elseif (i.eq.3) then
! Temporary to exclude huge values at origin:
        rt = r
        if(abs(r) < 0.15d0) rt = 0.15d0
        xe = rt
        vf1=7.449d0*xe**vp*exp(-2.d0*beta*xe)
        bx = xe*2.d0*beta-vp
        ebx = exp(-bx)
        cn(3)=1.461d0; cn(4)=14.11d0; cn(5)=183.5d0
        do n = 6,12
            cn(n)=(cn(n-1)/cn(n-2))**3*cn(n-3)
        enddo

        if(abs(bx) .gt. vp)then
            av(0)=1.d0
            do n = 1,24
                av(n)=av(n-1)*bx/dble(n)
            enddo
```

```
            fn(1)=av(0)+av(1)+av(2)
            do n = 2,12
                fn(n) = fn(n-1)+av(2*n-1)+av(2*n)
            enddo
            do n = 1,12
                fn(n) = 1.d0-ebx*fn(n)
            enddo
        else
            do n = 1,12
                ak = bx/(fact(2*n+1))
                ak1 = min(1.d0,max(abs(ak),1.d-30))
                sk = ak
                do j=2,100
                    ak = ak * bx / dble(2*n+j)
                    sk = sk + ak
                    if(abs(ak) .le. 1.d-15*ak1) goto 201
                enddo
201             continue
                fn(n) = ebx * bx**(2*n) * sk
            enddo
        endif
        vf2 = 0.d0
        do n = 3,12
            vf2 = vf2 + cn(n)*fn(n)/xe**(2*n)
        enddo
        zheltrim = 315766.2067d0 * (vf1 - vf2)
      else
        call acestp('zheltrim: wrong number of potential')
      endif
      return
      end function zheltrim
!
!CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC


real(8) function lihepot(r)
!c Calculates different potentials for helium trimer:
      implicit none
      real(8) r, br, chi
    !Li
          double precision, parameter :: C6=22.5d0,  Q=7.d0, b=1.06d0, a=4700.d0, Qb=7.42d0

      br = r*b !in atomic units
      if (abs(br) .le. 16.6d0) then
        chi=(14.6d0+0.918d0*Qb**2)/(1.d0-(Qb/16.6d0)**2)
        lihepot = C6/120.d0*((b/3.d0)**6)*(a*exp(-br)-chi*exp(-2.d0/3.d0*br)-exp(-br/3.0))
      else
        lihepot = C6/120.d0*((b/3.d0)**6)*a*exp(-br)-C6/(r**6-(Q**2)*(r**4))
      endif
      lihepot = lihepot/convfac
      return
```

```
      end function lihepot

!
END MODULE POTPOT
```

Подпрограмма вычисления DVR-функций и их производных

```
subroutine dvrcalc(npoiZ,xz,fdvr,ddvr)
!  Calculates DVR functions at integration points
        implicit none
        integer npoiZ,i1,i2,i,k
        double precision mleg(npoiZ),dleg(npoiZ)
        double precision fdvr(npoiZ,npoiZ),ddvr(npoiZ,npoiZ)
        double precision xz(npoiZ),xv
        double precision vv1(0:npoiZ)
!
        mleg = 0.d0
        dleg = 0.d0
        vv1(0)=1.d0
        do i2=1,npoiZ
          xv=xz(i2)
          vv1(1)=xv
          do i1=2,npoiZ
            vv1(i1)=((2*i1-1)*xv*vv1(i1-1)-(i1-1)*vv1(i1-2))/dble(i1)
          enddo
          mleg(i2)=vv1(npoiZ)
          dleg(i2)=npoiZ/(1.d0-xv*xv)*(vv1(npoiZ-1)-xv*vv1(npoiZ))
        enddo
        do i=1,npoiZ
          xv=xz(i)
          do k=1,npoiZ
            if(i.eq.k) then
              fdvr(i,k)=1.d0
              ddvr(i,k)=xv/(1.d0-xv*xv)
            else
              fdvr(i,k)=0.d0
              ddvr(i,k)=dleg(k)/(dleg(i)*(xz(k)-xv))
            endif
          enddo
        enddo
  return
end
!
```

Код вычисления матричных элементов при помощи DVR-функций

```
    subroutine jsubele(iel,jsub,nbas,jbas,jelf)
!
```

```
! calculates the contributions from the subelement jsub
! to the matrix elements between all functions defineb by jbas
!
! If offdiag=.true. the off-diagonal term is calculated.
! The corresponding matrix is NONSYMMETRIC!
!
! the integrations are normally performed numerically
! if one of the subelement boundaries is found beyond
! practical infinity ("dinfty") a branch is made to
! analytical integration
!
! jelf is an auxiliary numbering of the elemenary functions
!
! base  base coordinate for integration in each direction
! weig  weigths for integration in each direction
! jaco  factors of jacobi determinant (assumed to factorize)
! pote  array for the potential values at base points
! poba  temporary storage for potential * basis function
! vbas  1-dim elementary function values at base points
! dbas  1-dim genaralized derivative values at base points
! ekin  factorized kinetic energy matrix elements
! amet  factorized metrical matrix elements (overlaps)
! ovba  .true. if basis function has overlap with subelement
!
    use aceglob
    use mpimod
    implicit none
    logical, save :: call1 = .true.
    logical ovba(maxbas),ovrlap,fastjump,lpotcalc
    integer jsub(6),nbas,jbas(12,maxbas),jelf(maxbas,3)
    double precision base(maxpoi,3),weig(maxpoi,3),jaco(maxpoi,3)
    double precision vwrk(maxpoi),dwrk(maxpoi),rwrk(maxpoi),qwrk(maxpoi)
    double precision ddot,kron,lamminus
!
! auxiliary variables
    integer iel,iax,i0,i1,i2,i3,iba,jba,limup, i4
    integer iacu, SwB
    double precision qmas(3)
    double precision a1,a2,a3
    complex(8) scal,scq2
!CC_EY
    integer jelfj1,jelfj2,jelfj3,jelfi1,jelfi2,jelfi3
    integer i11,il,i111,imxi,ivw1,ierr,ivw2
    integer ipow(maxfun)
    double precision vsp1,vw1,vw2,vw3,vw4,vwoff,pvsp1,pvsp2
    double precision vsp1m(maxfun),vsp2m(maxpoi),av3(maxpoi)
    double precision, allocatable :: pote(:,:,:), poba(:,:,:)
    double precision, allocatable :: fdvr(:,:), ddvr(:,:)
!=======================================================
    fastjump = offdiag
    lpotcalc = potint=='3d-numer' .or. potint=='3d-coreZ'
```

```
      if(.not. lpotcalc) then
         call acestp('jsubele: integr. method undefined for nonzero J')
      endif
! coordinate-dependent code
      if(call1)then
        call1=.false.
        if(mpi_rank==0) write(6,*) '   ***  ASSOCIATED INTEGRAL CALCULATION  ***'
      endif
!-------------------------------------------------------------
! calculate mass factors:
! jacobi coordinates:
      if(coor.eq.'jacobi  ')then
        qmas(1)=0.5d0*(ama1+ama2+ama3)/(ama1*(ama2+ama3))
        qmas(2)=0.5d0*(ama2+ama3)/(ama2*ama3)
        qmas(3)=1.d0
      else
        write(6,'(//'' !!! coordinate type "'',a8,''" not implemented !!! '')')coor
        call acestp('JSUBELE: unknown type of coordinates')
      endif
!=============================================================
      pvsp1=(jproj+ispar)**2
      pvsp2=jvalue*(jvalue+1)-2.d0*pvsp1
! exterior scaling
      scal=1.d0
      scq2=1.d0
! check a possibility of associated integration
      if(abs(hypo(jsub(5),3)+1.d0).gt.1.0d-8.or.  &
         abs(hypo(jsub(6),3)-1.d0).gt.1.0d-8)     &
            call acestp(' ASSOC: non-single interval for angle')
!=============================================================
! numerical integration:
! for each coordinate iax:
      do iax=1,3
        npoi(iax)=npoa(iax,iacu(jsub))
! get base points
        a1=hypo(jsub(2*(iax-1)+1),iax)
        a2=hypo(jsub(2*(iax-1)+2),iax)
        call getpoi(iax,a1,a2,npoi(iax),base(1,iax),weig(1,iax))
        if (iax.eq.3) then
          allocate (fdvr(npoi(iax),npoi(iax)), ddvr(npoi(iax),npoi(iax)))
          call dvrcalc(npoi(iax),base(:,iax),fdvr,ddvr)
! fdvr(i,k), ddvr(i,k) - DVR functions and their derevatives at integration points,
! where is a number of function, k is a number of a point
        endif
!
! evaluate 1-dim elementary functions and
! the generalized derivatives
        i3=0
        do iba=1,nbas
          if(jelf(iba,iax).gt.i3+1)call acestp('SUBELE: bug')
          if(jelf(iba,iax).eq.i3+1)then
```

```
          i3=i3+1
          if(i3.gt.maxelf) call acestp('SUBELE: i3.gt.maxelf')
          i0=jbas(6+iax,iba)
          i2=jbas(9+iax,iba)
          do i1=1,npoi(iax)
            if (iax.eq.3) then
              vbas(i1,i3,iax)=fdvr(i2,i1)/sqrt(weig(i2,iax))
              dbas(i1,i3,iax)=ddvr(i2,i1)/sqrt(weig(i2,iax))
            else
             call getvba(iax,i0,i2,base(i1,iax),vbas(i1,i3,iax),dbas(i1,i3,iax))
            end if
            if(offdiag.AND.RB) then
              if((RBParity.AND.(mod(jproj,2)==0)).OR. &
                 ((.NOT.RBParity).AND.(mod(jproj,2)==1))) then
                vbas_ev(:,:,:)=vbas(:,:,:)
                dbas_ev(:,:,:)=dbas(:,:,:)
                do i4=1, ifun(0,3,iel)
                  ifun(i4,3,iel)=ifun(i4,3,iel)+1
                enddo
                call getvba(iax,i0,i2,base(i1,iax),vbas_od(i1,i3,iax),dbas_od(i1,i3,iax))
                do i4=1, ifun(0,3,iel)
                  ifun(i4,3,iel)=ifun(i4,3,iel)-1
                enddo
              else
                vbas_od(:,:,:)=vbas(:,:,:)
                dbas_od(:,:,:)=dbas(:,:,:)
                do i4=1, ifun(0,3,iel)
                  ifun(i4,3,iel)=ifun(i4,3,iel)-1
                enddo
                call getvba(iax,i0,i2,base(i1,iax),vbas_ev(i1,i3,iax),dbas_ev(i1,i3,iax))
                do i4=1, ifun(0,3,iel)
                  ifun(i4,3,iel)=ifun(i4,3,iel)+1
                enddo
              endif
            endif
          enddo
!   Recalculating func. and derivat. for nonzero momentum;
!   only for coordinate-parameter, only for radial coordinates and
!   only for Jacobi coordinates. ANGULAR VARIABLE MUST NOT BE SCALED!
!   Extracted from JRENORM.
          if(iax==1.or.iax==2)then
            do i1=1,npoi(iax)
              call fren(base(i1,iax),vbas(i1,i3,iax),dbas(i1,i3,iax),qmas,iax)
              if(offdiag.AND.RB) then
                call fren(base(i1,iax),vbas_od(i1,i3,iax),dbas_od(i1,i3,iax),qmas,iax)
                call fren(base(i1,iax),vbas_ev(i1,i3,iax),dbas_ev(i1,i3,iax),qmas,iax)
              endif
            enddo
          endif
        endif
      enddo
```

```
    enddo
!=======================================================================
!
    allocate(pote(npoi(1),npoi(2),npoi(3)))
    allocate(poba(npoi(1),npoi(2),npoi(3)))
!-----------------------------------------------------------------------
! jacobi coordinates
    if(coor.eq.'jacobi__')then
! jacobi determinant (factorized)
     jaco(1:npoi(1),1)=base(:,1)**2
     jaco(1:npoi(2),2)=base(:,2)**2
     jaco(1:npoi(3),3)=1.d0
    do iax=1,3
! calculate the factorized  kinetic energy and overlaps
    i1=0
    do jba=1,nbas
! calculate only if new 1-dimensional elementary function
    if(jelf(jba,iax).eq.i1+1)then
    i1=i1+1
    i2=0
! multiply by weights
    vwrk(1:npoi(iax))=weig(:,iax)*vbas(:,i1,iax)*jaco(:,iax)
    dwrk(1:npoi(iax))=weig(:,iax)*dbas(:,i1,iax)*jaco(:,iax)
! 3rd coordinate is cos(theta), derivatives are by theta:
       if(iax.eq.3)then
         do i3=1,npoi(iax)
            vsp1=1.d0-base(i3,iax)*base(i3,iax)
            dwrk(i3)=dwrk(i3)*vsp1
!            rwrk=0.d0;
!            qwrk=0.d0;
            rwrk(i3)=(1-jproj-ispar)*vwrk(i3)*base(i3,iax)-dwrk(i3)
            qwrk(i3)=vwrk(i3)/vsp1
         enddo
        else
          rwrk(1:npoi(iax))=vwrk(:)/(base(:,iax)**2)
        endif
! set limit for nonsymmetric matrix
            limup=jba
            if(offdiag) limup=nbas
            do iba=1,limup
! calculate only if new 1-dimensional elementary function
              if(jelf(iba,iax).eq.i2+1)then
                i2=i2+1
                vw1=0.d0
                vw2=0.d0
                vwoff=0.d0
                if(iax.ne.3)then
                  if (offdiag.AND.RB) then
                    if((RBParity.AND.(mod(jproj,2)==0)).OR. &
                    ((.NOT.RBParity).AND.(mod(jproj,2)==1))) then
                      vw2=ddot(npoi(iax),dwrk,1,dbas_od(1,i2,iax),1)
```

```
                vw1=ddot(npoi(iax),vwrk,1,vbas_od(1,i2,iax),1)
                vwoff=ddot(npoi(iax),rwrk,1,vbas_od(1,i2,iax),1)
              else
                vw2=ddot(npoi(iax),dwrk,1,dbas_ev(1,i2,iax),1)
                vw1=ddot(npoi(iax),vwrk,1,vbas_ev(1,i2,iax),1)
                vwoff=ddot(npoi(iax),rwrk,1,vbas_ev(1,i2,iax),1)
              endif
            else
              vw2=ddot(npoi(iax),dwrk,1,dbas(1,i2,iax),1)
              vw1=ddot(npoi(iax),vwrk,1,vbas(1,i2,iax),1)
              vwoff=ddot(npoi(iax),rwrk,1,vbas(1,i2,iax),1)
            endif
          else
            vw1=ddot(npoi(iax),vwrk,1,vbas(1,i2,iax),1)
            vw2=ddot(npoi(iax),dwrk,1,dbas(1,i2,iax),1)+ &
                pvsp1*ddot(npoi(iax),qwrk,1,vbas(1,i2,iax),1)
            vwoff=ddot(npoi(iax),rwrk,1,vbas(1,i2,iax),1)
          endif
          amet(i1,i2,iax)=vw1
          ekin(i1,i2,iax)=vw2*qmas(iax)
          qrsq(i1,i2,iax)=vwoff*qmas(iax)
          if(.not.offdiag)then
            ekin(i2,i1,iax)=ekin(i1,i2,iax)
            qrsq(i2,i1,iax)=qrsq(i1,i2,iax)
            amet(i2,i1,iax)=amet(i1,i2,iax)
          endif
        endif
      enddo
    endif
  enddo
enddo
!
  else
    call acestp('JSUBELE.AS_SP:␣coordinate␣system␣unknown')
  endif
!=======================================================
! *** matrix elements of potential ***
!=======================================================
  if(.not.fastjump)then
! get potential values at all base points
    if(lpotcalc)then
      call usepote(maxpoi,npoi,base,pote,rext,scal)
!c
!c multiply potential by weights and jacobi-determinant
      do i1=1,npoi(1)
      do i2=1,npoi(2)
        vsp1=weig(i1,1)*jaco(i1,1)*weig(i2,2)*jaco(i2,2)
        do i3=1,npoi(3)
          pote(i1,i2,i3)=pote(i1,i2,i3)*vsp1*weig(i3,3)*jaco(i3,3)
        enddo
      enddo
```

```
        enddo
!c
      else
        stop  ' JSUBELE: forbidden value of potint for nonzero J'
      endif
   endif
!-----------------------------------------------------------------------
! assemble matrix elements
    do jba=1,nbas
      ovba(jba)=ovrlap(jsub,jbas(1,jba))
    enddo
    do jba=1,nbas
    i3=jbas(12,jba)
! only if function is non-zero on subelement
      if(ovba(jba))then
        jelfj1=jelf(jba,1)
        jelfj2=jelf(jba,2)
        jelfj3=jelf(jba,3)
!CC
        if(fastjump) goto 557
!CC
        if(potint.eq.'3d-numer' .or. potint.eq.'3d-coreZ')then
! multiply potential by right hand side basis function
        poba=0.d0
        do i1=1,npoi(1)
          do i2=1,npoi(2)
            poba(i1,i2,i3)=pote(i1,i2,i3)*vbas(i1,jelfj1,1)*vbas(i2,jelfj2,2)*vbas(i3,jelfj3,3)
          enddo
        enddo
      endif
557 CONTINUE
!
! loop over left side functions
! set limit for nonsymmetric matrix
    limup=jba
    if(offdiag) limup=nbas
!
    do iba=1,limup
    i3=jbas(12,iba)
    if(ovba(iba))then
      a1=0.d0
        jelfi1=jelf(iba,1)
        jelfi2=jelf(iba,2)
        jelfi3=jelf(iba,3)
!CC
    if(fastjump) goto 556
!CC
! if overlap multiply from the left and sum up
    if(potint.eq.'3d-numer' .or. potint.eq.'3d-coreZ')then
       do i1=1,npoi(1)
         do i2=1,npoi(2)
```

```
            a1=a1+poba(i1,i2,i3)*vbas(i3,jelfi3,3)*vbas(i1,jelfi1,1)*vbas(i2,jelfi2,2)
          enddo
        enddo
      else
        call acestp('JSUBELE.AS_SP: integr. method undefined for nonzero J')
    endif
!
!=======================================================================
! *** matrix elements of kinetic energy ***
!=======================================================================
!CC
 556    continue
!CC
! jacobi coordinates
      if(coor.eq.'jacobi  ')then
        if(offdiag.or.(eqnsys.and.(isysind.ne.jsysind)))then
          a1=0.d0
        else
      a1=a1+                                &
        (ekin(jelfj1,jelfi1,1)              &
        +qrsq(jelfj1,jelfi1,1)*pvsp2)       &
        *amet(jelfj2,jelfi2,2)              &
        *amet(jelf(jba,3),jelf(iba,3),3)    &
!
        +ekin(jelfj2,jelfi2,2)              &
        *amet(jelf(jba,3),jelf(iba,3),3)    &
        *amet(jelfj1,jelfi1,1)              &
!
        +ekin(jelf(jba,3),jelf(iba,3),3)*   &
        (amet(jelfj1,jelfi1,1)*qrsq(jelfj2,jelfi2,2)+       &
         qrsq(jelfj1,jelfi1,1)*amet(jelfj2,jelfi2,2))
        endif
!
! get overlap matrix element
        a2=amet(jelfj1,jelfi1,1)*amet(jelfj2,jelfi2,2)*     &
           amet(jelf(jba,3),jelf(iba,3),3)
!
! get off-diagonal term
        if(offdiag)then
          a3=qrsq(jelfj1,jelfi1,1)*amet(jelfj2,jelfi2,2)*   &
             qrsq(jelf(jba,3),jelf(iba,3),3)
        endif
!
      else
        call acestp('JSUBELE.AS_SP: coordinate system unknown')
      endif

.....................................................................
```