ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение Иркутский Государственный Университет

Международная организация Объединенный Институт Ядерных Исследований

Лаборатория Теоретической Физики

Физический факультет
Кафедра теоретической физики
Заведующий кафедрой
Профессор Валл А. Н.

Дипломная работа

Ядерные переходы $O^{16}H \to F^{17}$ и эффекты электронного экранирования

Руководитель:
_____ к.ф.-м.н. Пеньков Ф. М.. (ОИЯИ),
Студент группы 1521
_____ Наумов Д.В.
Работа защищена
с оценкой __ОТЛИЧНО__
"15" __июня__ 1997

Рецензент:
_____

Нормоконтролер
_____ доц. Персиков В. М.

Дубна-Иркутск 1997 г.

В настоящей работе рассматривается влияние атомных электронов на протекание ядерных реакций как в молекулярных состояниях, так и в состоянии рассеяния ядер.

Методом Хартри - Фока - Рутана решается задача для определения терма молекулы, являющегося эффективной потенциальной энергией для движения ядер в молекуле. При этом для ядер предполагается выполненным приближение Борна - Оппенгеймера. Методом Нумерова находится решение дифференциального уравнения Шредингера для определения волновой функции ядер в молекуле. Колебательный спектр ядер находится методом стрельбы. В работе получены оценки на скорости ядерных реакций как из молекулярного состояния, так и из состояния рассеяния.

Объем диплома составляет 39 страниц. Диплом состоит из Введения, трех глав, Заключения и 7 приложений. Кроме этого, отдельно на 68 страницах приведен написанный комплекс программ. В дипломной работе приведены 4 таблицы и 9 рисунков.

Оглавление

Глава 1

Введение

Сравнительно недавно были проведены исследования по регистрации спектров гамма излучения от различных объектов в космосе. При этом было обнаружено, что от некоторых молекулярных облаков, например, в созвездии Ориона [1] идет излучение фотонов МэВ - ных энергий. В комплексе Ориона есть огромные молекулярные облака и именно оттуда наблюдается поток высокоэнергичных гамма квантов с энергией от 3 до 7 МэВ. Что может излучать в холодных молекулярных облаках столь высокоэнергичные фотоны ? Очевидно, что эти фотоны - результат ядерных переходов из возбужденного в основное состояние. Интересно выяснить механизм возбуждения ядер.

Поскольку во Вселенной существуют потоки космических частиц, то можно предположить, что космические лучи возбуждают ядра в молекулярных образованиях. Но эта гипотеза не согласуется с экспериментальным наблюдением. А именно, интенсивность линий с энергиями 4.44 и 6.13 МэВ, что соответствует переходу из возбужденного состояния ядер ^{12} и ^{16}O, примерно на два порядка превышает то, что предсказывается теорией, основанной на знании спектра космических лучей. Более того, относительное содержание и должно существенно превышать долю более тяжелых элементов для того, чтобы объяснить малое количество линий в области энергий от 1 до 3 МэВ (в этой области энрегий излучают в основном именно тяжелые элементы). В тоже время, средняя энергия излучаемого спектра гамма квантов требует увеличения содержания ядер ^{12} и ^{16}O относительно легких ядер и . Однако, до сих пор не ясно, почему содержание именно этих элементов должно быть столь велико. Возможное объяснение проблемы излучения гамма квантов могло бы быть в том, что спектр космических лучей именно в той области галактики отличается от среднего. Это предположение основывается на различных механизмах ускорения частиц, но пока механизм этого ускорения до конца не понят.

Существует, однако, еще одна гипотеза, которая также может претендовать на объяснение этой проблемы. Дело в том, что для определенных молекул может быть существенно отличная от нуля вероятность протекания в ней ядерных реакций. Очевидно, что из - за кулоновского отталкивания ядер вероятность для ядер подойти близко друг к другу сильно подавлена потенциальным барьером. Но, оказывается, что существуют молекулы, полная энергия которых очень близка к энергии объединенного ядра. Например, это ^{17}OH и $^{18}F^*$. [1] Это означает, что рассматриваемая система обладает вырожденной энергией: то есть двум разным волновым функциям соответствует од-

[1] значения энергии для этой и других молекулярных состояний с похожими свойствами приведены в Приложении 7

на энергия. [2] Но в таком случае вероятность подбаерьрного перехода резко возрастает (см., например [2]). Такие задачи рассматривались в ряде работ : ([3, 4, 5, 6] см., также цитированную там литературу.) В работе Х. Пиккера [6] наблюдался эффект увеличения на несколько порядков вероятности ядерной реакции из высоковозбужденных молекулярных состояний. В настоящей работе также наблюдается подобный эффект.

Прежде чем переходить к постановке задачи, рассмотрим простой пример, демонстрирующий существование эффекта экспоненциального увеличения вероятности туннелирования под потенциальным барьером. Пусть имеется такой потенциал и налетающая справа частица (см. Рис. 1.1)

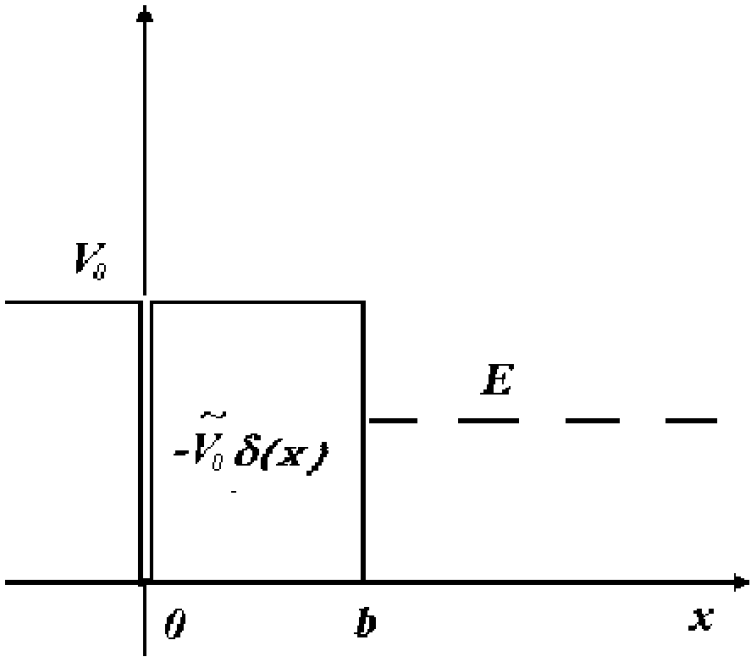

Рис. 1.1: Частица с энергией E налетает на потенциальную стенку с δ ямой в начале координат.

Как видно на рисунке, потенциал содержит на расстоянии b от правого края δ - яму. Если бы ее не было, то вероятность проникновения частицы экспоненциально уменьшалась бы с расстоянием. Однако, наличие притягивающего потенциала внутри, в корне меняет дело. Вместо этого, при определенной энергии налетающей частицы, а именно, при энергии, равной энергии связанного состояния в δ - яме, вероятность $\omega(x)$ туннелирования возрастает резонансным образом. Эффективно получается, что стенка как бы отодвигается от правого края - влево на расстояние b, что легко видно аналитически:

$$\omega(x) \simeq \frac{4}{1+\frac{\chi_0^2}{\chi_0^2+\rho^2}} \cdot e^{2\chi_0 b} \cdot e^{-2\chi_0|x|} \quad \text{при} \quad x \longrightarrow -\infty, \tag{1.1}$$

где $\chi_0^2 = \frac{m\tilde{V}_0}{\hbar^2}$ и $\rho^2 = \frac{2mV_0}{\hbar^2}$.

Качественно ситуацию легко понять, рассматривая вероятность вылета частицы из квазистационарного состояния в яме. Очевидно, что это одно и тоже явление. Хорошо

[2] Конечно, в действительности эти энергии не могут быть в точности равными друг другу, поскольку возбужденное состояние ядра обладает конечной шириной порядка КэВ

известный пример -это вылет α частицы из тяжелого ядра. Ясно, правда, что хотя вероятности туннелирования и вылета одинаковы, скорости же реакций разные, что связано с разными потоками. Итак, мы рассмотрели простой пример, в котором можно до конца провести аналитические вычисления и убедились в существовании эффекта *резонансного* увеличения вероятности туннелирования под барьером.

Поскольку прямоугольные потенциалы не описывают никакой реальной физической системы, необходимо получить реальный потенциал для движения ядер в молекуле. Причем, поскольку ожидается наблюдение резонансного туннелирования, то это означает, что ядерная волновая функция составного ядра не сосредоточена полностью в объеме ядра, а имеет довольно длинный "хвост"на больших, по сравнению с ядерными, расстояниях. Таким образом, приближение чисто кулоновского отталкивания между ядрами явно недостаточно и нужно решать задачу с учетом всех электронов в молекуле.

Начиная с середины - 20 века усилиями целой плеяды исследователей были развиты методы для решения молекулярных задач в рамках квантовой механики. Наиболее популярным, в настоящее время, является метод молекулярных орбиталей, краткое изложение которого будет в следующей главе.

Теперь сформулируем задачу, которая решается в данной работе. Рассматривается молекула ^{17}, для нее нужно найти терм (энергия молекулы при фиксированном расстоянии R между ядрами) для всех R, с этим термом решить уравнение Шредингера для ядер и найти их волновую функцию. Определив эту волновую функцию, оценить вероятность ядерной реакции:
$^{17}O + H \longrightarrow {}^{18}F^*$.

При этом, рассматриваются скорости реакций как из связанного молекулярного состояния, так и из непрерывного спектра. Последняя задача имеет важные приложения для расчетов скоростей ядерных реакций на солнце. Дело в том, что при солнечных энергиях сравнительно легкие ядра не имеют электронов, последние перешли в солнечную плазму, однако, для более тяжелых атомов ситуация меняется, и у них, как правило, остаются низколежащие электроны. Таким образом, при рассмотрении ядерных реакций этих ядер необходимо учитывать наличие электронов, которое существенно влияет на значение волновой функции ядер при $R = 0$. Простые оценочные соображения показывают, что боровский радиус системы с тяжелыми ядрами порядка ядерных размеров. При этом, из квазиклассического приближения следует, что насыщение интегралов, определяющих значение волновой функции в нуле происходит на расстояниях, меньших боровского радиуса. Таким образом, получается, что электроны из солнечной плазмы не оказывают существенного влияния на скорость таких ядерных реакций, поскольку в электронейтральной плазме электроны не могут быть локализованы на столь малых расстояниях. Все это приводит к тому, что необходимо учитывать электроны, находящиеся в связанном состоянии. В данной работе это учитывается, решая задачу на связанное состояние для электронов и непрерывного спектра ядер. Полученное таким образом значение $\Psi(0)$ сравнивается с чисто кулоновским фактором (без учета электронов) и наблюдается существенное отличие между ними при солнечных энергиях.

Глава 2

Квантовая механика молекул

2.1 Гамильтониан

Гамильтониан системы с N электронами и M ядрами, записанный в атомной системе единиц имеет вид:

$$\hat{H} = -\sum_{\alpha=1}^{M} \frac{1}{2M_\alpha} \triangle_\alpha - \sum_{i=1}^{N} \frac{1}{2} \triangle_i + V(R_\alpha, r_i), \tag{2.1}$$

$$V(R_\alpha, r_i) = -\sum_{\alpha,i} \frac{Z_\alpha}{r_{\alpha i}} + \sum_{i,j} \frac{1}{r_{ij}} + \sum_{\alpha,\beta} \frac{Z_\alpha Z_\beta}{R_{\alpha\beta}}.$$

Здесь $r_{\alpha i}$ - расстояние между i - электроном и α - ядром, r_{ij} - расстояние между i и j - электронами, $R_{\alpha\beta}$ - расстояние между α и β ядрами. M_α - масса α ядра.

В этом гамильтониане не учтены релятивистские эффекты, так же как и эффекты спин - спинового и спин - орбитального взаимодействия, что также является нерелятивистским приближением. Релятивистские поправки начинают существенно сказываться для тяжелых ядер с $Z \geq 30$ (см., например: [7]).

При необходимости эти эффекты можно учесть по теории возмущений.

Точное аналитическое решение это уравнение допускает только в случае двух неподвижных ядер и одного электрона. [1] Переменные уравнения в этом случае разделяются в сфероидальных координатах. Однако, для всех остальных систем приходится решать задачу приближенно и, по большей части, численно.

Прямое решение этого дифференциального уравнения тоже невозможно, в чем легко убедиться простыми оценками для нужного количества чисел, табулирующих волновую функцию - решение уравнения Шредингера. Легко найти, что, скажем, взяв 20 квадратурных точек по каждой переменной (что будет весьма грубым приближением), для N - электронов надо будет получить матрицу из 20^{3N} чисел. Если отводить под каждое число хотя бы по байту, то, соответственно, понадобятся 20^{3N} байт. У современных *workstation* типа *SUN* память на жестком диске, как правило, не превышает 200 Гб $= 2 \cdot 10^{11}$ байт. Решая задачу, например, с 10 электронами, получаем, что нам нужно будет хранить массив из 20^{30} байт!

Это убедительный пример в пользу развития эффективных методов расчета молекул, справляющихся с этими трудностями и правильно описывающих физику дела.

[1] разумеется вместе со случаем водородоподобных атомов.

2.2 Приближение Борна - Оппенгеймера

Первым приближением при решении задачи является *приближение Борна - Оппенгеймера* (см., например:[8]), суть которого заключается в следующем. Поскольку масса ядер в несколько тысяч раз превышает массу электрона, то кинетические энергии ядер будут много меньше кинетических энергий электронов, и значит, при движении ядер, электроны успеют много раз перераспределиться таким образом, как если бы ядра были неподвижны. То есть, должно быть справедливым *адиабатическое приближение*. Таким образом, при решении уравнения Шредингера:

$$\hat{H}\Psi(R_\alpha; r_i) = E\Psi(R_\alpha; r_i), \tag{2.1}$$

с гамильтонианом (2.1), можно искать волновую функцию в виде линейной комбинации функций, зависящих только от координат ядер или только координат электронов.

$$\Psi(R_\alpha; r_i) = \sum_k u_k(r_i)v_k(R_\alpha). \tag{2.2}$$

Легко получить условия применимости этого приближения (см., например: [9]); для этого, действительно, достаточно большой иерархии масс. Для электронной волновой функции получается уравнение:

$$\left[-\frac{1}{2}\sum_{i=1}^{N}\triangle_i + V(R_\alpha, r_i)\right] u(r_i) = E(R_\alpha)u(r_i). \tag{2.3}$$

В этом уравнении нет операторов кинетической энергии ядер. $E(R_\alpha)$ зависит от межъядерных расстояний параметрически и называется *электронным термом* молекулы. Этот терм является потенциальной энергией для ядер, именно он и определяет волновую функцию ядер в молекуле. Основная задача квантовой механики молекул - это определение терма. Найдя его, нужно будет решить уравнение для определения волновой функции ядер в молекуле (см., например:[8]):

$$\left[-\sum_{\alpha=1}^{M}\frac{1}{2M_\alpha}\triangle_\alpha + E(R_\alpha)\right] v(R_\alpha) = \varepsilon v(R_\alpha). \tag{2.4}$$

2.3 Молекулярные Орбитали

Основным методом квантовомеханического расчета молекул является *приближение молекулярных орбиталей* (МО). Этот метод был развит Хартри, Фоком, Мюлликеном и, наиболее последовательно, Рутаном (см., например: [7, 10, 11]). В этом подходе электрон в молекуле описывается одноэлектронной волновой функцией, как если бы он двигался по определенным орбиталям в молекуле. Все остальные электроны и ядра создают потенциальное поле, в котором двигается "рассматриваемый"[2] электрон. Поскольку, это поле, в свою очередь, определяется, через МО электронов в молекуле, то окончательным будет некое *самосогласованное* распределение электронов в молекуле. Итак, пусть $\psi_i(\xi)$ - i - ая *спин-орбиталь*, а ξ - совокупность пространственных и спиновых координат электрона. Полная волновая функция электронов в молекуле должна подчиняться принципу Паули, то есть, менять знак при перестановке частиц. С другой

[2]конечно, мы не можем пометить какой нибудь электрон, что отражается в антисимметризации МО, см. дальше.

стороны, ясно, что она должна быть некой линейной комбинацией произведений всех *спин-орбиталей* в молекуле. Этим двум требованиям удовлетворяет функция, составленная в виде детерминанта, называемая еще детерминантом Слэтера:

$$\Psi(r_i) = \frac{1}{\sqrt{N!}} det(\psi_1(\xi_1)\psi_2(\xi_2)\dots\psi_N(\xi_N)). \tag{2.1}$$

Если число электронов в молекуле четное, то можно считать, что электроны с противоположно направленными спинами попарно занимают одинаковые пространственные орбитали. То есть: $\psi_i(\xi_i) = \phi_i(\vec{r}_i)\cdot\gamma(\sigma)$, где $\phi_i(\vec{r}_i)$- пространственная орбиталь, а $\gamma(\sigma)$ - спиновая функция. И тогда, например,

$\psi_1(\xi) = \phi_1(\vec{r})\cdot\alpha(\sigma)$, $\psi_2(\xi) = \phi_1(\vec{r})\cdot\beta(\sigma)$ и т.д., [3] где, $\alpha(\sigma)$ и $\beta(\sigma)$ описывают спин "вверх"и "вниз"соответственно.

Поскольку, как легко увидеть, волновая функция (2.1) остается инвариантной относительно преобразования (если $detU = 1$):

$\psi_i(\xi) \longrightarrow \psi_i'(\xi) = U_{ij}\psi_j(\xi)$,

то, подходящим линейным преобразованием спин - орбитали можно выбрать такими, что $< \psi_i|\psi_j >= \delta_{ij}$. Введенная таким образом волновая функция (2.1) является также собственной функцией $\mathbf{S_z}$ и $\mathbf{S^2}$, в чем можно убедиться явным образом (см., например: [7]).

Уравнения для определения $\psi_i(\xi)$ получаются из требования минимума полной энергии системы:

$$E = \frac{< \Psi|\hat{H}|\Psi >}{< \Psi|\Psi >}. \tag{2.2}$$

Подставив выражение для волновой функции (2.1) в определение энергии, получим:

$$E = \sum_i^N H_i + \frac{1}{2}\sum_{i\neq j=1}^N (2J_{ij} - K_{ij}). \tag{2.3}$$

Входящие сюда матричные элементы даны в приложении 2. Причем, поскольку мы работаем в адиабатическом приближении, когда положения ядер считаются фиксированными, то из гамильтониана исключены члены, отвечающие кулоновскому отталкиванию ядер, так как они постоянны по величине и могут быть включены в подсчет энергии позже. Входящие в (2.3) матричные элементы определяются следующим образом (см. например: [7]):

$$H_i =< \psi_i(\vec{r}_1)|\hat{h}(\vec{r}_1)|\psi_i(\vec{r}_1) >, \tag{2.4}$$

$$J_{ij} =< \psi_i(1)\psi_j(2)|\frac{1}{r_{12}}|\psi_i(1)\psi_j(2) >, \tag{2.5}$$

$$K_{ij} =< \psi_i(1)\psi_j(2)|\frac{1}{r_{12}}|\psi_j(1)\psi_i(2) >, \tag{2.6}$$

$$\text{где,}\quad \hat{h}(\vec{r}_i) = -\frac{1}{2}\triangle_i - \sum_\alpha \frac{Z_\alpha}{r_{i\alpha}}. \tag{2.7}$$

[3] в случае нечетного числа электронов нет аргументов в пользу выбора одинаковых пространственных орбиталей для электронов с противоположно направленными спинами, так как электрон с нескомпенсированным спином взаимодействует по разному с остальными электронами. Поэтому должна появиться разница между пространственным распределением разных спин - орбиталей

Применим теперь вариационный принцип к среднему значению энергии молекулы. Для этого нужно проварьировать по спин - орбитальным функциям, ограничиваясь такими вариациями, когда они остаются ортонормированными. Получаются интегро - дифференциальные уравнения на определение этих спин - орбиталей [7, 12]. Полученные уравнения носят название уравнений **Хартри -Фока** и имеют следующий вид:

$$\hat{F}|\psi_i >= \varepsilon_i|\psi_i > \tag{2.8}$$

Оператор Фока $\hat{F}$ выглядит следующим образом:

$$\hat{F} = -\frac{1}{2}\triangle - \sum_{\alpha}\frac{Z_\alpha}{r_{i\alpha}} + \sum_j < \psi_j|\frac{1}{r_{12}}|\psi_j > - \sum_j < \psi_j|\frac{1}{r_{12}}|\psi_i > . \tag{2.9}$$

Следуя [7, 8, 12] вывод этого уравнения приведен в приложении 3, а сейчас кратко обсудим его особенности и способ его решения.

Смысл уравнения ХФ состоит в том, что для каждой спин орбитали ψ_i мы имеем отдельное уравнение, каждое их которых напоминает обычное уравнение Шредингера для волновой функции электрона. Качественно можно утверждать, что величины ε_i, являющиеся решением уравнений на собственные значения (2.8), называемые *орбитальными энергиями*, равны энергии, необходимой для удаления электрона с i -ой орбитали (см., например: [7, 8]). [4]

Член $-\frac{1}{2}\triangle$ представляет собой оператор кинетической энергии электрона, величина $-\sum_\alpha \frac{Z_\alpha}{r_{i\alpha}}$ представляет собой сумму по всем ядрам энергий кулоновского притяжения между этим электроном и всеми ядрами. Выражение $\sum_j < \psi_j|\frac{1}{r_{12}}|\psi_j >$ представляет собой для точки, в которой находиться электрон, значение потенциальной энергии полного электронного распределения с плотностью заряда $\sum_j \psi_j\psi_j$. Смысл последней суммы в уравнении (2.9) заключается в том, чтобы не учитывать взаимодействие электрона с самим собой. Это, так называемая, обменная поправка, которая описывает взаимодействие электрона с плотностью распределения отрицательно заряженного заряда единичной величины. Это можно увидеть просто взяв интеграл по всему пространству от плотности распределения этого заряда, учитывая ортонормированность орбиталей (см., например: [8]). Итак, окончательно, рассматриваемый нами электрон взаимодействует с $N-1$ электронами и M ядрами, как и должно быть.

В случае атома, где присутствует сферическая симметрия, сравнительно просто решить уравнения (2.8), эти расчеты выполнены и получено хорошее согласие с экспериментом (см. например: [7]). Решением этих уравнений являются, как уже упоминалось, ε_i и сами орбитали. При этом полная энергия молекулы выражается посредством:

$$E = \sum_{i=1}^{N/2}(\varepsilon_i + H_i). \tag{2.10}$$

Однако, при расчете молекул эту схему реализовать уже не удается. Здесь приходиться полагаться на приближенные методы, представляя молекулярные орбитали в виде *Линейной Комбинации Атомных Орбиталей* (МО ЛКАО).

К изложению этого метода, следуя ([7, 8, 12]), мы и переходим.

[4] разумеется, взятой со знаком минус

2.4 Метод Хартри - Фока - Рутана

Схема приближенного решения уравнений самосогласованного поля посредством линейной комбинации атомных орбиталей впервые была предложена Рутаном [13]. В качестве базиса орбиталей могут быть взяты любые функции, но, исходя из физических соображений, понятно, что наилучшее приближение при минимальной размерности базиса, будет в случае выбора базиса из атомных волновых функций, имеющихся в задаче. Не конкретизируя явный вид базисных функций, запишем разложение искомых молекулярных спин - орбиталей по базису из L - спин орбиталей (см., например: [7, 12]):

$$\psi_i = \sum_{k=1}^{L} C_i^k u_k. \tag{2.1}$$

Из ψ_i орбиталей нужно составить детерминантную волновую функцию, поэтому очевидно, что L должна быть, по крайней мере, не меньше, чем N (число электронов), и должно быть больше, если мы хотим получить более надежное разложение. Тогда, дополнительные спин - орбитали будут представлять собой возбужденные состояния системы (см., например: [7]).

Функции u_k не предполагаются ортонормированными, так как представляют собой атомные функции, принадлежащие разным атомам в молекуле. Запишем их матрицу перекрытия в виде:

$$S_{ij} \equiv < u_i | u_j > = \int d\vec{r} u_i^*(\vec{r}) u_j(\vec{r}). \tag{2.2}$$

Очевидно, что в этой формуле производится интегрирование по пространству и суммирование по спиновым переменным, так что для атомных орбиталей с разным спином матрица перекрытия равна нулю.

Подставляя в выражение для энергии (2.2) спин - орбитали (2.1) и варьируя по параметрам C_i^k, приходим к уравнениям **Хартри - Фока - Рутана** [11] на определение неизвестных коэффициентов C_i^k:

$$\sum_{m=1}^{L} F_{km} C_i^m = \varepsilon_i \sum_{m=1}^{L} S_{km} C_i^m, \tag{2.3}$$

где,

$$F_{km} = H_{km} + \sum_{l,n=1}^{L} D_{ln}(2[km|ln] - [kn|lm]), \tag{2.4}$$

$$D_{ln} = \sum_{i=1}^{N/2} {C_i}^{l*} {C_i}^{n}, \tag{2.5}$$

$$H_{km} = < u_k | \hat{h} | u_m > = \int d\vec{r} {u_k}^*(\vec{r}) \hat{h}(\vec{r}) u_m(\vec{r}), \tag{2.6}$$

$$[km|ln] = \int\int d\vec{r}_1 d\vec{r}_2 \frac{{u_k}^*(\vec{r}_1) u_m(\vec{r}_1) {u_l}^*(\vec{r}_2) u_n(\vec{r}_2)}{r_{12}}. \tag{2.7}$$

Следуя [12], вывод этих уравнений приведен в Приложении 3. Равенство (2.5) определяет компоненты *матрицы плотности*. [5]

[5] суммирование проводится только по занятым орбиталям.

Для нахождения нетривиального решения этой системы уравнений необходимо положить детерминант системы уравнений равным нулю и, получив алгебраическое уравнение на определение ε_i, найти его корни. Однако, то обстоятельство, что сама матрица F_{km} определяется с помощью матрицы плотности, которая, в свою очередь, записывается через неизвестные коэффициенты $C_i{}^k$, делает задачу более сложной. Чтобы ее решить, необходимо задаться неким первоначальным набором коэффициентов, подставить их в уравнение (2.3), найти новую совокупность коэффициентов, опять подставить их в уравнение и действовать так до тех пор, пока не получиться самосогласованное решение уравнения (см., например: [7, 8, 11]). Конечно, эта процедура тесно связана с постулатом теории Хартри - Фока о самосогласованном поле.

Найдя собственные значения, энергия молекулы находится согласно (см., например: [12]):

$$E = \sum_{i=1}^{N/2} \varepsilon^i + \sum_{k,m} D_{km} H_{mk}. \tag{2.8}$$

Процедура, предложенная Рутаном, сделала революцию в квантовомеханичеких расчетах молекул, потому что от интегродифференциального уравнения Хартри - Фока, решить которое не представляется возможным, перешли к алгебраической системе уравнений, для которой найти решение уже гораздо легче. Правда, в этом методе все трудности перенеслись на расчет огромного количества всевозможных интегралов и, связанным с этим, оптимальным выбором атомных орбиталей. Однако, действуя именно этим способом удалось рассчитать огромное количество молекул.

Сложность решения такой задачи зависит также и от размерности матрицы. Оказывается, что решение уравнения $det(F - \varepsilon_i S) = 0$ разбивается на несколько независимых уравнений для каждого типа симметрии, присущего данной молекуле. Это сильно упрощает дело и поэтому перейдем к описанию того, как это происходит.

2.5 Симметрия молекулы

Гамильтониан молекулы инвариантен относительно некоторой группы пространственных преобразований. Волновые функции молекулы принадлежат базису неприводимого представления этой группы. Например, в случае двухатомных молекул имеется симметрия относительно вращения вокруг оси, соединяющей ядра. Значит, зависимость волновых функций от угла будет $e^{im\phi}$, где m- целое число, и имеется вырождение для функций, отличающихся только знаком m. Рассматривая матричные элементы оператора Фока между функций с разными m, обнаружим, что они равны нулю. В общем случае это можно доказать используя теорию групп. Пусть ϕ_a и ϕ_b, функции, принадлежащие базису неприводимых представлений Γ_a и Γ_b точечной группы симметрии молекулы. Рассмотрим матричный элемент

$$I = < \phi_a | \hat{F} | \phi_b > \tag{2.1}$$

Очевидно, что этот матричный элемент, являясь C - числом, должен оставаться инвариантным при групповых преобразованиях точечной группы симметрии молекулы. Поэтому, при формальном действии на это число оператором группы $\hat{R}$, под чем понимается групповое преобразование функций ϕ_i и оператора $\hat{F}$, матричный элемент I должен остаться неизменным:

$$\hat{R} I = I = \hat{R} < \phi_a | \hat{F} | \phi_b > . \tag{2.2}$$

Для того, чтобы это число не изменилось, необходимо, чтобы произведение $(\phi_a \hat{F} \phi_b)$ разлагалось на неприводимые представления согласно схеме:

$$\Gamma_a \oplus \Gamma_F \oplus \Gamma_b \oplus = \Gamma_1 \oplus \Gamma_2 \oplus \Gamma_3 \oplus \dots \tag{2.3}$$

и, при этом, чтобы среди представлений $\Gamma_1, \Gamma_2, \Gamma_3 \dots$, были тождественные. Иначе, это может быть выполнено только если $I = 0$. Говоря языком теории групп, это утверждение можно сформулировать следующим образом: Если представление Γ_F не содержится в разложении $\Gamma_a \oplus \Gamma_b$ на неприводимые представления, то матричный элемент I равен нулю [7].

Это очень важный результат, помогающий существенно упростить задачу. Все рассмотренное здесь относилось к *замкнутым оболочкам*, [6] однако, аналогичные рассуждения можно применить и для случая незаполненных, то есть *открытых оболочек*. Задача, которую мы решаем соответствует именно этому случаю, поэтому перейдем к описанию уравнений в этом случае.

2.6 Открытые оболочки

Случай открытых оболочек более сложный прежде всего потому, что волновая функция молекулы не всегда может быть представлена в виде (2.1) и даже в виде линейной суперпозиции таких детерминантов.Энергия, также, не всегда может быть выражена согласно (2.3). Однако, для некоторого типа молекул обобщение может быть проведено и на случай открытых оболочек. Это, прежде всего, молекулы с полузакрытой оболочкой, когда все состояния, относящиеся к какому - либо вырожденному набору, однократно заняты электронами с параллельными спинами. В молекулярной системе можно выделить совокупность двукратно занятых орбиталей $\phi_c(\vec{r})$ и однократно занятых орбиталей $\phi_o(\vec{r})$. Ожидаемое значение энергии системы изменяется по сравнению с (2.3) за счет добавления слагаемых, соответствующих однократно занятым орбиталям [11]:

$$E = 2\sum_k H_k + \sum_{k,l}(2J_{kl} - K_{kl}) + f\left\{2\sum_m H_m + \right.$$
$$\left. + f\sum_{m,n}(2\alpha J_{mn} - bK_{mn}) + 2\sum_{k,m}(2J_{km} - K_{km})\right\}. \tag{2.1}$$

Индексы k, l нумеруют МО из набора $\phi_c(\vec{r})$, индексы m, n -из набора $\phi_o(\vec{r})$. Матричные элементы J_{kl} и K_{kl} определяются формулами (2.4). Константы f и α и b зависят от заселенности открытой оболочки и специфики терма. Например, для систем с полузакрытой оболочкой в основном состоянии $f = \frac{1}{2}$, $\alpha = 1$ и $b = 2$.

Аналогично, варьируя по орбитальным волновым функциям с целью достижения минимума энергии, можно получить уравнение, впервые полученное Рутаном: ([11]).

$$\hat{F}_c \phi_k = \varepsilon_k \phi_k, \quad \hat{F}_o \phi_m = \varepsilon_m \phi_m. \tag{2.2}$$

Явный вид операторов $\hat{F}_c$ и $\hat{F}_o$ в данной работе не приводиться, поскольку он довольно громоздкий и его, при желании, можно найти в упомянутой статье Рутана.

[6] то есть, случаю четного числа электронов

Принципиально новых членов в нем нет, кроме тех, что связаны с незаполненной оболочкой. В настоящей работе используется другой подход для решения молекулярной задачи [14]. А именно, мы не будем ограничивать себя такими спин - орбиталями, которые попарно заняты. Предполагается различие в пространственном распределении орбиталей для электронов с разнонаправленными спинами.

Итак, спин - орбитали ищутся в виде:

$$\psi_i^\alpha = \sum_k C_{ik}^\alpha u_k, \quad \psi_i^\beta = \sum_k C_{ik}^\beta u_k. \tag{2.3}$$

Повторяя те же действия по минимизации энергии, получаются уравнения:

$$\sum_m F_{km}^\alpha C_{im}^\alpha = \varepsilon_i^\alpha \sum_m S_{km} C_{im}^\alpha, \quad \sum_m F_{km}^\beta C_{im}^\beta = \varepsilon_i^\beta \sum_m S_{km} C_{im}^\beta, \tag{2.4}$$

где,

$$F_{km}^\alpha = H_{km} + \sum_{l,n} \left(D_{ln}^\alpha + D_{ln}^\beta \right) [km|ln] - D_{ln}^\alpha [kn|lm],$$

$$D_{ln}^\alpha = \sum_{i=1}^{n_\alpha} C_{il}{}^{\alpha *} C_{in}{}^\alpha. \tag{2.5}$$

(И аналогично для F_{km}^β и D_{ln}^β). Числа n_α и n_β обозначают количества электронов со спином "вверх"и "вниз"соответственно. Эти две системы уравнений нужно решать одновременно, поскольку операторы уравнения определяются друг через друга. Как правило, численное решение этих уравнений дает очень близкие значения как для орбитальных энергий, так и для самих коэффициентов разложения. Видно, что условие того, чтобы считать орбитали одинаковыми для частиц с разным спином - малость обменного интеграла. В этом случае можно приближенно считать, что $D_{ln}^\alpha \simeq D_{ln}^\beta$. Тогда уравнения (2.5) сводятся к уравнениям (2.3) [15]. Мы же будем стремиться не делать дальнейших упрощений и решать уравнения (2.5). Уравнения решаются также, как описано выше - итерированием. Определив орбитальные энергии и параметры разложения C_{in}, полная энергия молекулы находиться по формуле(см., например: [12]):

$$E = \sum_{k,m} \left(D_{ln}^\alpha + D_{ln}^\beta \right) H_{mk} + \sum_{k,m,l,n} \left\{ \frac{1}{2} \left(D_{km}^\alpha D_{ln}^\alpha + \right. \right.$$

$$\left. \left. + D_{km}^\beta D_{ln}^\beta \right) \left([km|ln] - [kn|lm] \right) + D_{km}^\alpha D_{ln}^\beta [km|ln] \right\}. \tag{2.6}$$

В описанном только что приближении, то есть в случае разных пространственных орбиталей для частиц с противоположно направленным спином, есть один недостаток. Волновая функция молекулы, построенная из таких орбиталей, являясь собственной функцией $\mathbf{S_z}$, не является собственной функцией оператора $\mathbf{S^2}$. [7] Это означает, что в действительности, она не описывает системы частиц с определенным спином. [8] Качественно совершенно понятно, что для того, чтобы, введенная таким образом, волновая функция молекулы была также собственной функцией $\mathbf{S^2}$, необходимо, чтобы пространственные орбитали для электронов с разнонаправленными спинами, совпадали. И, если пренебречь обменными интегралами в матрице плотности D_{km}^α, то получиться именно этот случай. Поэтому, хотя, строго говоря, такая волновая функция не описывает систему электронов с определенным спином, реальная система не будет сильно отличаться

[7] $\mathbf{S_z}$ и $\mathbf{S^2}$ - третья проекция и полный квадрат спина всех электронов в молекуле, соответственно.

[8] Этот недостаток волновой функции устраняют, вводя проекционные спиновые операторы [7, 12].

от рассматриваемой таким образом [12, 15]. Исходный гамильтониан (2.1) вообще не содержит спиновых переменных, поэтому при вычислении физических величин, слабо зависящих от ориентации спина, таких как энергия или орбитальные заселенности, небольшое различие в пространственном распределении электронов с разнонаправленным спином, которое и приводит к тому, что волновая функция не является собственной функцией $\mathbf{S^2}$, не приведет к заметной ошибке при вычислении этих физических величин.

Итак, мы описали как решаются задачи квантовой механики молекул, и теперь приступаем к конкретному расчету молекулы OH.

Глава 3

Расчет молекулы

3.1 Основное состояние

Электронная конфигурация атома кислорода $(1s)^2(2S)^2(2p)^4$, волновая функция водорода в основном состоянии: $1s$. Таким образом, молекула OH представляет собой систему с открытой оболочкой, то есть с нечетным числом электронов, равным девяти.

Двухатомная молекула с разными ядрами обладает цилиндрической симметрией [1] и поэтому состояния молекулы классифицируются согласно:

$\sigma \to m = 0, \quad \pi \to m = 1, \ldots$

Здесь $m-$ проекция орбитального момента на ось z.

При расчете молекул методом Хартри - Фока - Рутана немаловажным является вопрос о выборе базисных спин - орбиталей. В принципе, можно взять некий набор функций типа Слэтеровского или Гауссова вида, которые зависят от свободных параметров, подлежащих определению, посчитать все интегралы, решить уравнения ХФР и варьировать по этим параметрам, определяя их оптимальный набор, соответствующий наименьшей энергии системы. При этом оказывается, что для наилучшей аппроксимации молекулярной волновой функции приходиться брать большое количество пробных функций и расчеты становятся очень трудоемкими. [2]

Кажется наиболее естественным взять в качестве базисных функций атомные орбитали, участвующие в задаче. Искомые МО будут их линейной суперпозицией и физически ясно, что при различных предельных случаях именно атомные орбитали будут давать наилучшее приближение к МО. Например, при $R \longrightarrow \infty$ нижние орбитали кислорода останутся "чистыми"атомными орбиталями без примеси водородной орбитали, а эффект связанного состояния будет целиком определяться смесью наивысшей орбитали атома кислорода и водородной атомной орбиталью.

С другой стороны, рассматривая энергетические спектры атомов, можно заключить, что поскольку электрон в атоме водорода имеет энергию $-\frac{1}{2}$ ат. ед., а энергия $2p$- состояния близка к этому, то этому электрону энергетически наиболее выгодно попасть именно в эту потенциальную яму, то есть, другими словами, связываться в основном будут именно эти уровни.

Итак, мы выбираем в качестве базисных функций для расчета молекулы следующие

[1] Двухатомные молекулы с одинаковыми ядрами обладают еще дополнительной симметрией волновой функции, связанной с отражением относительно центра молекулы

[2] например, приходится работать с матрицами большой размерности.

атомные орбитали:

$$\begin{aligned} 1s &= \frac{P_{1s}}{r} \cdot Y_{00}(\theta,\varphi), \\ 2S &= \frac{P_{2S}}{r} \cdot Y_{00}(\theta,\varphi), \\ 2p_0 &= \frac{P_{2p}}{r} \cdot Y_{10}(\theta,\varphi), \\ 2p_\pm &= \frac{P_{2p}}{r} \cdot Y_{1\pm}(\theta,\varphi), \\ 1h &= \frac{e^{-r'}}{r'} \cdot Y_{00}(\theta',\varphi'). \end{aligned} \tag{3.1}$$

Волновые функции P_i не имеют, конечно, явного аналитического вида, мы их возьмем как решение соответствующих уравнений Хартри - Фока для атома кислорода.

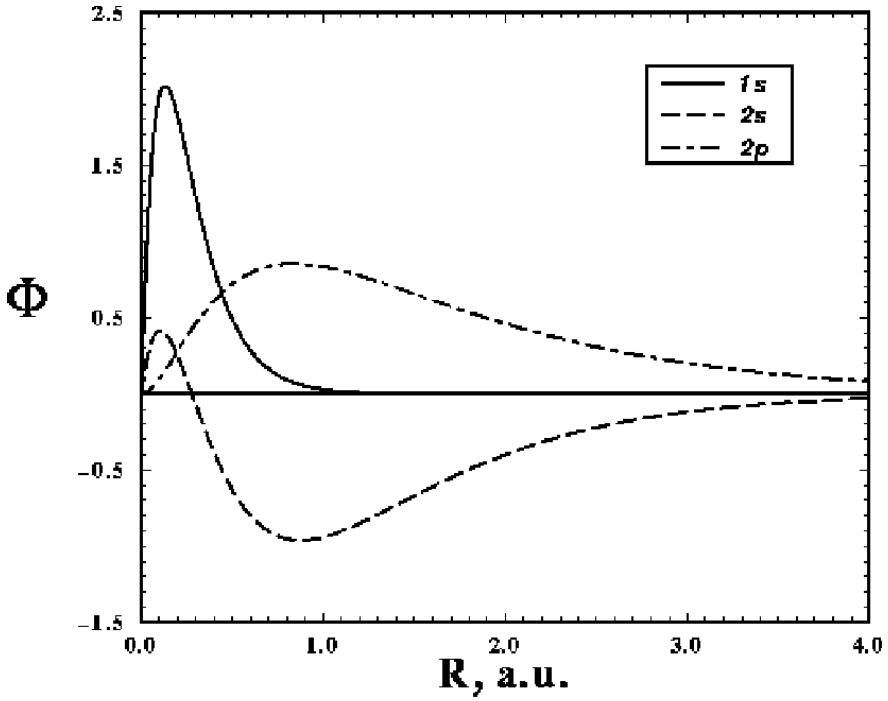

Рис. 3.1: Орбитальные функции атома кислорода

На Рис. 3.1 изображены используемые в настоящей работе базисные функции. [3] Из этих атомных орбиталей составим молекулярные орбитали :

$$\begin{cases} m\sigma &= C_{m1}(1s) + C_{m2}(2S) + C_{m3}(2p_0) + C_{m4}(1h), \\ \pi^+ &= (2p_+), \\ \pi^- &= (2p_-). \end{cases} \tag{3.2}$$

Итого, мы получаем четыре σ - МО орбитали (с нулевой проекцией орбитального момента) и две π МО орбитали. Последние остались чистыми $2p$ - орбиталями атома кислорода, так как мы предположили, что основное состояние молекулы строится, когда водород также находится в основном состоянии, что довольно разумное предположение.

[3] для нахождения этих волновых функций я воспользовался программой rcn.f, написанной проф. R.Cowan на фортране. Эта программа вычисляет также энергетический спектр и прочие полезные характеристики атомов. Она доступна по ftp: //t4.lanl.gov/pub/cowan в директории /pub/cowan/

Заранее неочевидно, что каждая из σ орбиталей будет занята электронами в молекуле, поэтому необходимо сделать несколько расчетов и выбрать то состояние молекулы, в котором ее энергия минимальна. Это нужно сделать по той причине, что в принципе, можно разместить 8 электронов на σ орбиталях и последний электрон разместить на π - орбитали. Однако, забегая вперед, скажем, что на самом деле основное состояние молекулы будет иметь незанятой 4σ орбиталь и выглядит таким образом:

$(1\sigma)^2(2\sigma)^2(3\sigma)^2(\pi^+)^2\pi^-$.

Итак, определив МО и тип основного состояния, приступим к решению уравнений ХФР (2.5). При этом, как уже обсуждалось в первой главе, разделе о симметрии молекул, уравнения расщепляются на три независимые системы уравнений, каждая для своего вида симметрии:

Для σ орбиталей:

$$\sum_{m=1}^{4}\left[H_{km}+\sum_{l,n=1}^{6}\left(D_{ln}^{\alpha}+D_{ln}^{\beta}\right)[km|ln]-D_{ln}^{\alpha}[kn|lm]\right]C_{im}^{\alpha}=\varepsilon_i^{\alpha}\sum_{m=1}^{4}S_{km}C_{im}^{\alpha},$$

$$\text{где,}\quad D_{ln}^{\alpha}=\sum_{i=1}^{n_\alpha}C_{il}{}^{\alpha *}C_{in}{}^{\alpha}. \tag{3.3}$$

Аналогичные уравнения получаем для орбиталей с другим спином, с учетом того, что $n_\alpha = 5$, а $n_\beta = 4$. Эти две системы, строго говоря, неэквивалентны из - за дополнительного слагаемого в матрице плотности, [4] хотя, как показывает расчет, для энергии молекулы это различие практически несущественно.

Для π - орбиталей уравнения много проще, и их энергия целиком определяется решением первой системы уравнений. Сами коэффициенты разложения для π - орбиталей определяются из условий нормировки МО. Итак, уравнения имеют вид:

$$H_{55}+\sum_{l,n=1}^{6}\left(D_{ln}^{\alpha}+D_{ln}^{\beta}\right)[55|ln]-D_{ln}^{\alpha}[5n|l5]=\varepsilon_5^{\alpha}. \tag{3.4}$$

Для удобства все орбитали пронумерованы, начиная от σ орбиталей, до π - орбиталей. Итого, у нас 6 орбиталей для каждого значения спина. Повторимся, что из - за симметрии молекулы, задача на определение коэффициентов с матрицей 6×6 свелась к задаче с матрицей 4×4.

Условия ортонормированности МО приводят к следующим соотношениям:

$$<\phi_i|\phi_i>=\sum_{k,m=1}^{6}C_{im}{}^{*}C_{ik}<u_m|u_k>=\sum_{k,m=1}^{6}C_{im}{}^{*}C_{ik}S_{km}, \tag{3.5}$$

где S_{km} - элементы матрицы перекрытия атомных орбиталей. Отсюда легко определить коэффицинты для π - орбиталей. Итак, для того, чтобы решить эту задачу нужно знать элементы матриц, то есть, посчитать все необходимые интегралы.

3.2 Расчет интегралов

В нашей задаче встречаются трехкратные и шестикратные интегралы. В случае атомов, то есть, в случае сферической симметрии, интегралы по углам снимаются аналитически

[4] суммирование в матрице плотности происходит только по занятым орбиталям

и расчет оставшихся одно- и двухкратных интегралов сравнительно просто провести численно. Поскольку в случае молекул такой симметрии нет, то, как это не удивительно, расчет интегралов становится наибольшей проблемой и наиболее время затратным процессом. К тому же, число необходимых интегралов сильно возрастает, по сравнению с атомными расчетами. Более того, поскольку нам нужен молекулярный терм для всех межъядерных расстояний, то количество требуемых интегралов возрастает еще пропорционально числу точек.

Мы будем различать интегралы, не зависящие и зависящие от межъядерного расстояния R. Все расчеты проводим в системе координат, с центром на атоме кислорода, как показано на Рис. 3.2.

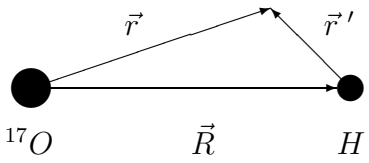

Рис. 3.2: Векторы $\vec{r}$ и $\vec{r}\,'$ в системе координат, центрированной на атоме кислорода

Для того, чтобы снять все интегралы по углам водородную волновую функцию представим в виде [15]:

$$e^{-r'} = \sum_{n=0}^{10} \frac{\alpha_n(r,R)}{r} P_n(\cos\theta). \tag{3.1}$$

Здесь $r' = \sqrt{r^2 + R^2 - 2rR\cos\theta}$, $\alpha_n(r,R)$ - коэффициенты разложения, являющиеся линейной комбинацией модифицированных функций Бесселя полуцелого порядка. $P_n(\cos\theta)$ - полиномы Лежандра.

Применяя разложение (3.1), удается снять все интегралы по углам и, тем самым, сильно упростить себе задачу взятия интегралов, поскольку теперь они становятся одно - и двухмерными. Реккурентные соотношения для $\alpha_n(r,R)$ получены в приложении 4. Эта функция, а также ее производная, которая понадобиться при вычислении кинетических интегралов были запрограммированы и многократно протестированны.

Количество необходимых функций для правильной аппроксимации экспоненты зависит от межъядерного расстояния, что неудивительно. Легко получить, что при малых R достаточно взять небольшое количество $\alpha_n(r,R)$, чтобы получить хорошее приближение, с другой стороны, при увеличении расстояния, значение самой экспоненты сильно падает и в этой области R значение интегралов очень мало. Далее, для большинства интегралов, оказывается, что нужны всего лишь некоторые из $\alpha_n(r,R)$, что следует из возникающих δ - функций в интегралах по углам. [5] Таким образом, ограничение одиннадцатью функциями вполне удовлетворительно аппроксимирует водородную волновую функцию.

[5]Приложения 5,6

3.3 Результаты расчета

Орбитальные энергии, а также, полная энергия молекулы в зависимости от межъядерного расстояния, приведены в таблице 3.1.

Как видно из таблицы, 4σ уровень молекулы, действительно, не занят, поскольку имеет положительную энергию. Он будет являться возбужденным молекулярным уровнем. Молекулярная энергия достигает минимума при $R \simeq 2.2$ ат.ед. Молекула находиться в основном состоянии, разумеется, при этом межъядерном расстоянии. Глядя на Таб. 3.2, видим, что получено хорошее согласие с экспериментальным значением энергии молекулы OH в основном состоянии. Абсолютная ошибка порядка $1Ry$ является характерной для метода ХФР [15, 16, 17].

В молекулярных расчетах есть проблема нахождения энергии на малых межъядерных расстояниях. Дело в том, что энергия молекулы при $R \longrightarrow 0$ должна стремиться к энергии объединенного атома. Однако, часто бывает, что симметрии объединенного атома и исходной молекулы не совпадают, и, вследствии этого, при малых межъядерных расстояниях нужно рассматривать уже другую конфигурацию молекулы с нужной симметрией, чтобы получить энергию объединенного атома. Однако, в нашем случае, эти симметрии совпадают, и поэтому полученный нами терм гладко переходит в энергию атома фтора при $R = 0$. Чтобы это увидеть наиболее ясно, изобразим терм без учета кулоновского отталкивания ядер. При ядерных расстояниях кулоновские силы ничтожно малы по сравнению с ядерными, и поэтому рассматривать нужно именно такой терм. (см. Рис. 3.3)

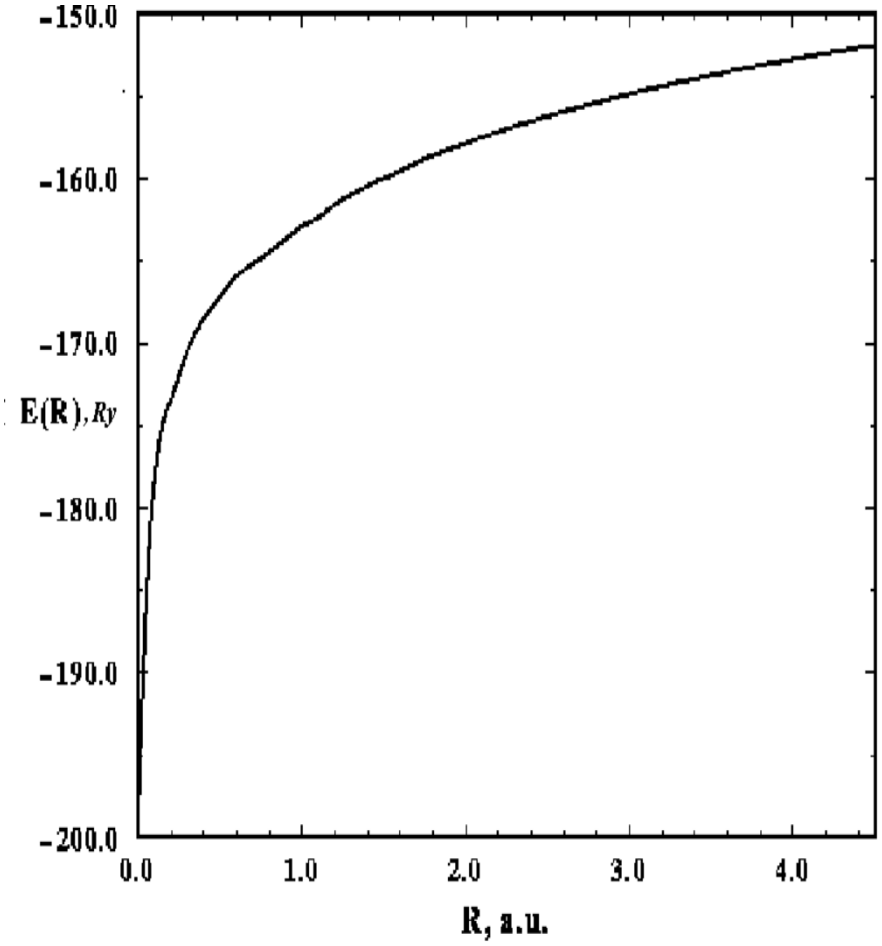

Рис. 3.3: Молекулярный терм без учета кулоновского отталкивания ядер

Таблица 3.1: **Энергетический спектр молекулы**

межъядерное	Энергии орбиталей					энергия
расстояние	$E_{1\sigma}$	$E_{2\sigma}$	$E_{3\sigma}$	$E_{4\sigma}$	$E_{2\pi}$	молекулы
0.1	-23.93035	-1.80296	-1.02915	14.29213	-2.40540	-18.08710
0.2	-22.78009	-1.85151	-1.02457	16.42680	-2.37936	-93.38817
0.3	-22.07776	-1.93136	-0.99539	18.01502	-2.34375	-117.09488
0.4	-21.64229	-2.01679	-0.95283	18.94097	-2.30363	-128.43725
0.5	-21.36735	-2.09717	-0.90643	19.18899	-2.26175	-134.98895
0.6	-21.19294	-2.16955	-0.86130	18.82717	-2.21936	-139.19935
0.7	-21.19577	-2.33415	-0.83172	20.02354	-2.17353	-142.53458
0.8	-21.08394	-2.34988	-0.78793	17.83090	-2.13401	-144.44525
0.9	-21.00104	-2.35706	-0.74937	15.76019	-2.09618	-145.82526
1.0	-20.93923	-2.35754	-0.71613	13.88725	-2.06013	-146.84563
1.1	-20.91512	-2.45219	-0.72472	9.98891	-2.04413	-147.85435
1.2	-20.85781	-2.34311	-0.66423	10.80261	-1.99300	-148.19651
1.3	-20.83130	-2.32995	-0.64450	9.56644	-1.96174	-148.64363
1.4	-20.81113	-2.31359	-0.62815	8.50163	-1.93184	-148.98659
1.5	-20.79603	-2.29610	-0.61488	7.59342	-1.90340	-149.25188
1.6	-20.78863	-2.30392	-0.64413	7.03928	-1.87956	-149.58108
1.7	-20.77470	-2.25123	-0.59468	6.11598	-1.84921	-149.59881
1.8	-20.76722	-2.22678	-0.58729	5.52054	-1.82364	-149.70643
1.9	-20.76179	-2.20174	-0.58143	5.00485	-1.79901	-149.78341
2.0	-20.75711	-2.17508	-0.57659	4.55130	-1.77511	-149.83114
2.1	-20.75330	-2.14779	-0.57275	4.15115	-1.75210	-149.85702
2.2	-20.75002	-2.11945	-0.56958	3.79736	-1.72981	-149.86315
2.3	-20.74707	-2.09082	-0.56705	3.48345	-1.70817	-149.85419
2.4	-20.74427	-2.06160	-0.56490	3.20417	-1.68713	-149.83129
2.5	-20.74154	-2.03206	-0.56306	2.95450	-1.66669	-149.79722

Таблица 3.1: **Энергетический спектр молекулы Продолжение**

межъядерное	Энергии орбиталей					энергия
расстояние	$E_{1\sigma}$	$E_{2\sigma}$	$E_{3\sigma}$	$E_{4\sigma}$	$E_{2\pi}$	молекулы
2.6	-20.73886	-2.00235	-0.56148	2.73040	-1.64691	-149.75387
2.7	-20.73615	-1.97252	-0.56006	2.52843	-1.62775	-149.70259
2.8	-20.73334	-1.94267	-0.55872	2.34622	-1.60919	-149.64478
2.9	-20.73042	-1.91275	-0.55742	2.18093	-1.59122	-149.58085
3.0	-20.72740	-1.88305	-0.55616	2.03077	-1.57389	-149.51277
3.1	-20.72427	-1.85355	-0.55484	1.89414	-1.55713	-149.44086
3.2	-20.72093	-1.82418	-0.55341	1.76938	-1.54093	-149.36561
3.3	-20.71751	-1.79519	-0.55191	1.65517	-1.52534	-149.28810
3.4	-20.71387	-1.76645	-0.55024	1.55063	-1.51028	-149.20837
3.5	-20.71080	-1.74026	-0.55276	1.46026	-1.49621	-149.13957
3.6	-20.70704	-1.71076	-0.54671	1.36514	-1.48212	-149.04715
3.7	-20.70260	-1.68314	-0.54453	1.28401	-1.46866	-148.96354

Таблица 3.2: Основное состояние молекулы

$-E_{\cdot}, Ry$	$-E_{\cdot}, Ry$	$\frac{E_{\cdot}}{E_{\cdot}}$
149.86	151.56	0.988

Итак, мы решили задачу для молекулы OH, получили молекулярный терм, который хорошо совпадает с экспериментальным значением энергии молекулы в основном состоянии, а также гладко переходит в энергию объединенного атома - атома фтора. Энергия атома фтора $\simeq -199Ry$.

Теперь мы можем поставить и решить задачу для ядер в этом потенциале.

Глава 4

Уравнение Шредингера для ядер в молекуле

Напомним, что в используемом нами приближении Борна -Оппенгеймера, движение ядер отделяется от движения электронов, и для ядер решается уравнение с потенциалом, равным молекулярному терму, полученному в предыдущей главе. В этом потенциале можно поставить задачу как на связанное состояние, так и на состояние рассеяния. Потенциал изображен на Рис. 4.1. Для наглядности он сдвинут на энергию свободных атомов, чтобы при $R \longrightarrow \infty$ он стремился к нулю.

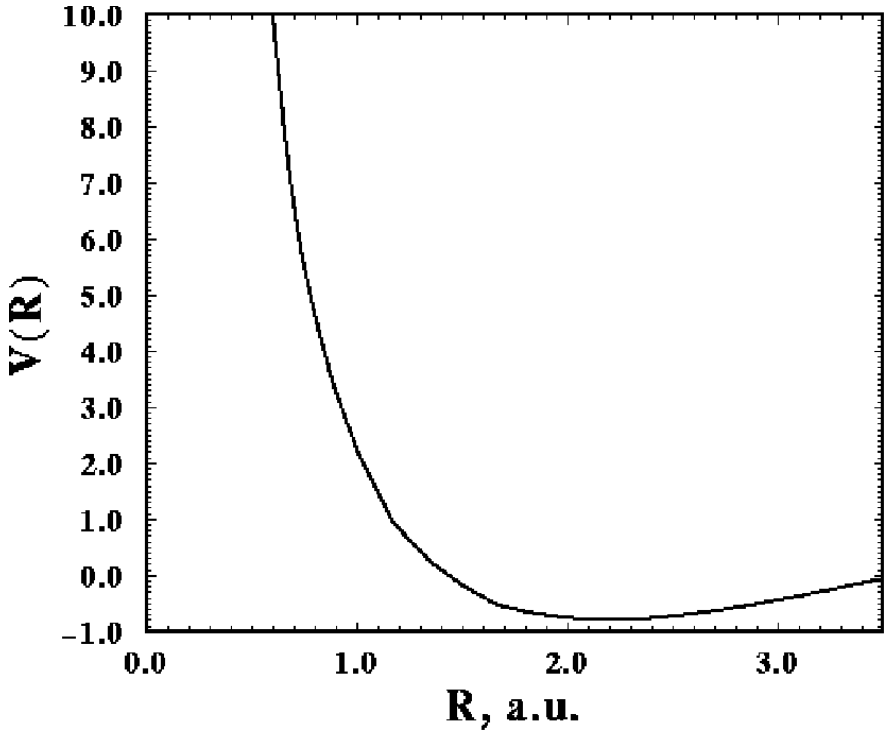

Рис. 4.1: Эффективный потенциал, действующий между ядрами

Радиальное уравнение Шредингера для ядер в молекуле имеет вид:

$$[\frac{d^2}{dr^2} + 2m^*(E - U(r))]\Phi(r) = 0, \quad \Phi(r) = r\Psi(r). \tag{4.1}$$

Таблица 4.1: Дискретный спектр ядер и значения $\Psi(0)$ в потенциале 4.1

уровень, N	энергия	$\Psi(0)$
0	-0.764	$1.0 \cdot 10^{-113}$
1	-0.721	$9.6 \cdot 10^{-113}$
2	-0.678	$6.5 \cdot 10^{-112}$
3	-0.636	$3.7 \cdot 10^{-111}$
4	-0.594	$1.8 \cdot 10^{-110}$
5	-0.552	$8.8 \cdot 10^{-110}$
6	-0.511	$4.0 \cdot 10^{-109}$
7	-0.474	$1.5 \cdot 10^{-108}$
8	-0.438	$4.0 \cdot 10^{-108}$
9	-0.401	$8.9 \cdot 10^{-108}$
10	-0.360	$2.0 \cdot 10^{-107}$
11	-0.319	$4.8 \cdot 10^{-107}$
12	-0.277	$1.2 \cdot 10^{-106}$
13	-0.236	$3.0 \cdot 10^{-106}$
14	-0.195	$7.7 \cdot 10^{-106}$
15	-0.154	$1.9 \cdot 10^{-105}$
16	-0.114	$4.7 \cdot 10^{-105}$

4.1 Дискретный спектр

В этом потенциале решается обычное уравнение Шредингера, для определения дискретного спектра энергии, а также волновых функций ядер. Задача решалась численно, интегрирование дифференциального уравнения осуществлялось по разностной схеме Нумерова. Для определения спектра энергии использовался метод пристрелки (см. [18]). Исходя из физических соображений, ясно, что дискретный спектр должен быть очень похожим на осцилляторный, поскольку эффективная масса ядер очень велика по сравнению с электронной и возбуждения должны быть локализованы вблизи минимума потенциала, где практически любой потенциал похож на осцилляторный. Поскольку нас интересует поведение волновой функции в нуле, то рассматривается только орбитальный момент $L = 0$, поскольку для всех остальных $\Psi(0) = 0$.

Видим, что спектр, действительно практически совпадает с осцилляторным, то есть, он эквидистантный с шагом $\simeq 0.04Ry$. Это колебательный спектр молекулы. Посмотрим теперь на вид волновых функций (см. Рис. 4.2).

Чтобы не загромождать рисунок, волновые функции приведены только для основного состояния, первого и шестнадцатого. Их легко узнать по количеству нулей функции. Мы видим, что уже шестнадцатый возбужденный уровень лежит гораздо ближе к началу координат, и отношение $\Psi_{16}(0)/\Psi_0(0)$ порядка 10^8! Таким образом, в настоящей работе (так же как и в работе [6]) наблюдается увеличение вероятности ядерных реакций, просто рассматривая возбужденные состояния молекулы. Причем увеличение существенное, на 16 порядков! Однако, сами значения $\Psi(0)$ очень малы, см. Таб. 4.1

В работах [5, 19] получено выражение для скоростей реакций из квазистационарного молекулярного спектра. Для того, чтобы охарактеризовать скорость ядерных пе-

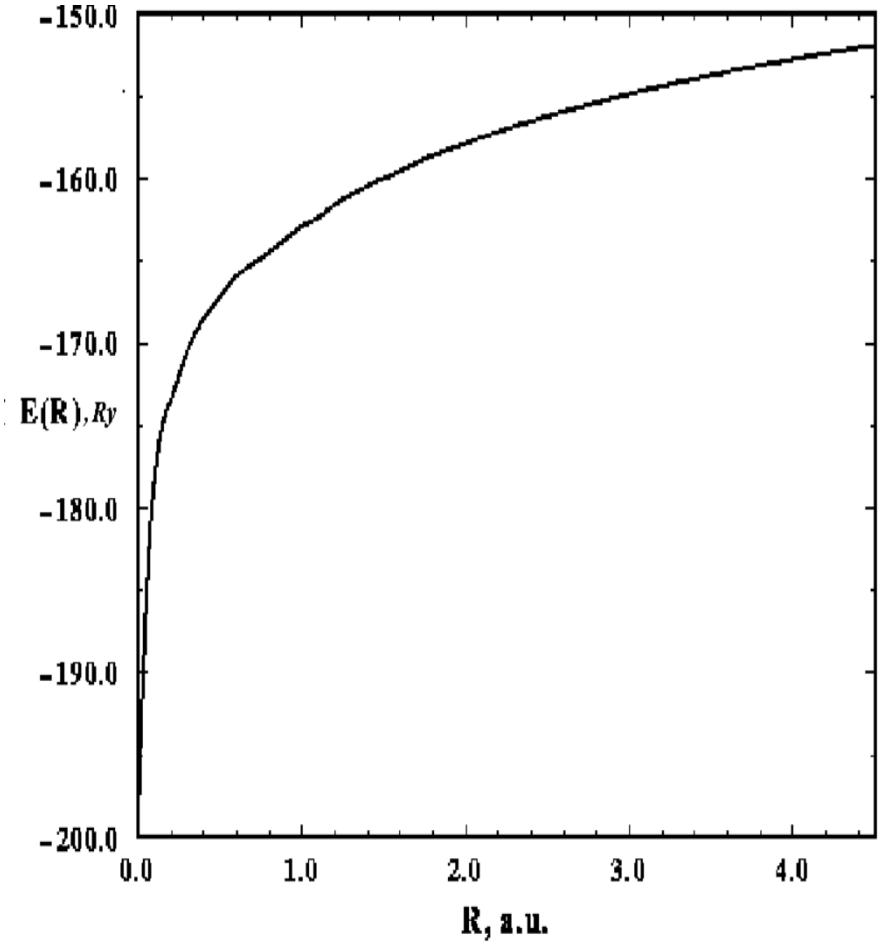

Рис. 4.2: Волновые функции дискретного спектра ядер в молекуле

реходов, вводится параметр λ, просто равный ширине молекулярного резонанса Γ_n, возникающего из - за наличия ядерных переходов. Тогда для связанных состояний:

$$\lambda = A_r |\Psi(0)|^2, \tag{4.1}$$

где, $A_r = S(E_{res})/(\pi m^* Z_1 Z_2)$. Здесь $S(E_{res})$ кулон - ядерный S -фактор. Очевидно, что весь эффект резонансной ядерной реакции заключается в этой величине, которую мы, однако, не вычисляем, так как это отдельная и весьма сложная задача. Нас же интересует оценка, которая следует из значения $\Psi(0)$. Приводя эту величину к правильной размерности (для этого ее нужно умножить на $\hbar$), получаем, что:

$$\lambda = 1.175 \cdot 10^9 S(E_{res}) \cdot |\Psi(0)|^2 \cdot \mathrm{c}^{-1}. \tag{4.2}$$

Вместо $S(E_{res})$ нужно подставлять безразмерное число, когда мы измеряем этот фактор в Мэв $\cdot$ барн. А для квадрата модуля волновой функции берутся безразмерные числа, когда мы измеряем волновую функцию в атомной системе единиц.

Итак, для того, чтобы существовала сколько - нибудь существенная вероятность ($\lambda \simeq 10^{-25}$ c^{-1}) протекания ядерной реакции в молекуле, необходимо, чтобы эффект ядерного резонанса увеличивал скорость реакции на 180 порядков.

4.2 Непрерывный спектр

Скорость реакции из непрерывного спектра определяется стандартным образом:

$$\lambda =< n_1 \cdot n_2 \cdot v \cdot \sigma > . \tag{4.1}$$

Здесь n_i - концентрации частиц, v - их остительная скорость, σ - сечение реакции.

Фиксируем асимпотику волновой функции в виде:

$$\Psi(r) \simeq \frac{sin(pr + \delta)}{r}. \tag{4.2}$$

Решая уравнение Шредингера для этого случая, получаем значения волновой функции в нуле для орбитального момента равного нулю. Эта величина сравнивается с чисто кулоновским фактором (см. например: [9])

$$|\Psi(0)|^2 = e^{-2\pi\eta} 2\pi\eta p^2. \tag{4.3}$$

Здесь $\eta = m^* Z_1 Z_2 / a_0 p$, a_0 - боровский радиус.

На Рис. 4.3 приведена волновая функция $\Phi(r) = r\Psi(r)$. С увеличением энергии растет как частота осцилляций в области действия потенциала, так и значение волновой функции в нуле. В настоящей работе для расчета значения $\Psi_{\cdot}(0)$ в качестве нуля использовалось значение $r = 10^{-5}$ ат. ед., что является характерным ядерным расстоянием. На Рис. 4.4 показаны значения $\Psi_{\cdot}(0)$ и $\Psi_{\cdot}(0)$.

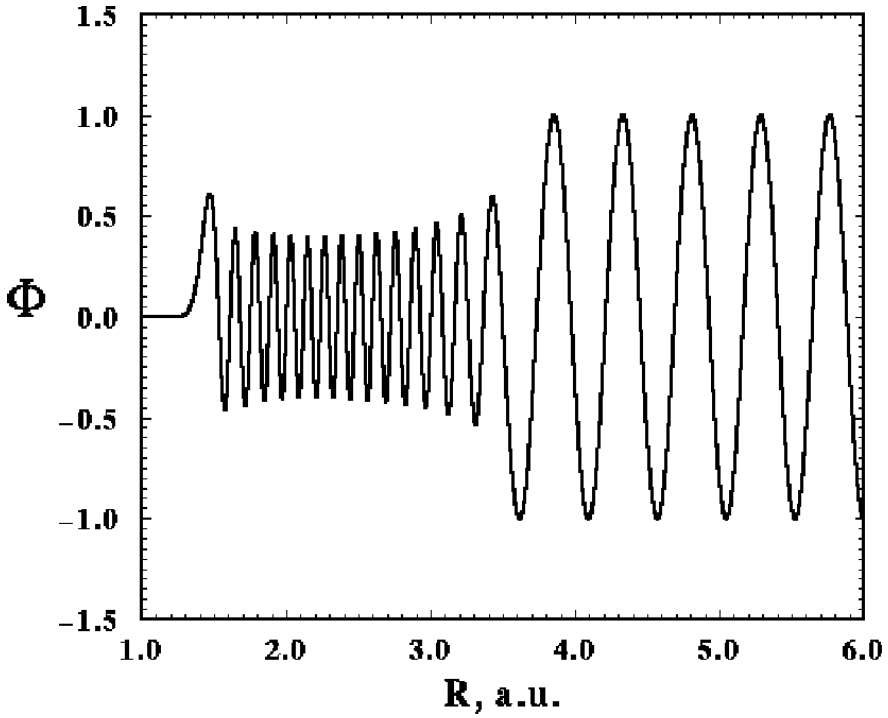

Рис. 4.3: Волновая функция рассеяния для энергии $E = 0.1Ry$

При малых энергиях ядер, волновая функция, вычисленная с учетом экранировки, на много порядков больше просто кулоновской функции в нуле. Очевидно, что при очень больших энергиях ядер, электроны не должны оказывать существенного влияния на значение $\Psi_{\cdot}(0)$. Однако, в применении к реальным физическим условиям, например, к ядерным реакциям, идущим на солнце, видим, что учет влияния электронов существенно изменяет скорость ядерной реакции (см. Рис. 4.5).

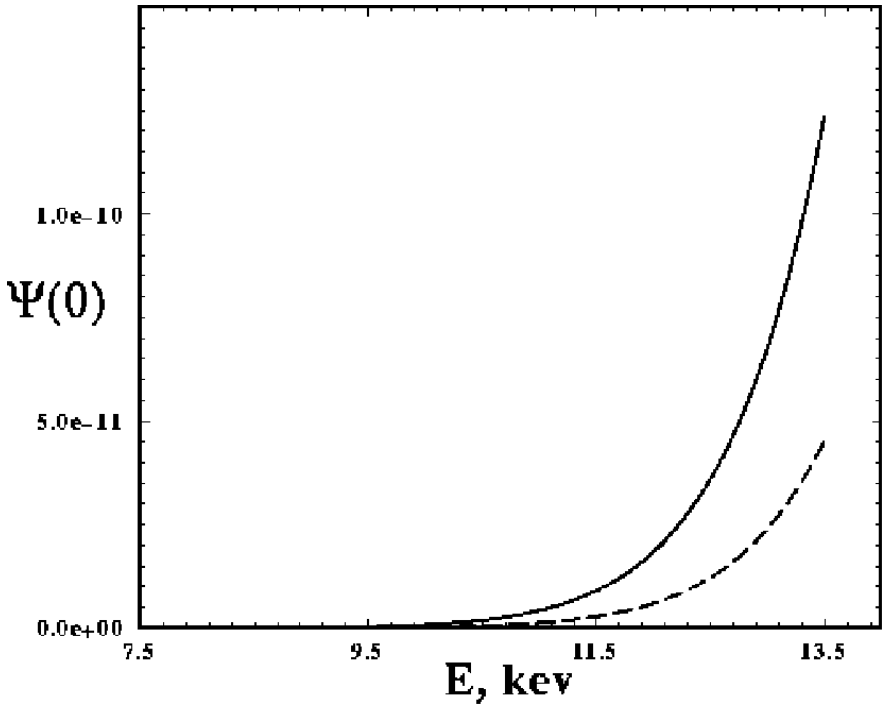

Рис. 4.4: Волновая функция рассеяния в нуле для различных энергий

Средняя энергия ядер на солнце порядка 1.4 КэВ, и при этих энергиях отношение скоростей ядерных реакций составляет около 8 порядков. При энергии ядер порядка 10 КэВ это отношение составляет около девяти. И только при энергиях порядка Мэв, атомные электроны не оказывают существенного влияния на значение $\Psi(0)$. Как видим, учет экранирования атомными электронами приводит к критическому изменению скорости ядерных реакций.

Качественно ситуацию можно понять следующим образом. Пусть два ядра рассеиваются друг на друге, тогда $|\Psi(0)|^2 = e^{-2\pi\eta} 2\pi\eta p^2$. Где $p = \sqrt{m^* E}$ [1] В случае рассеяния друг на друге двух атомов, начальная энергия атомов и конечная энергия объединенного атома различаются на некую $\triangle E$, и из закона сохранения энергии следует, что эта величина уходит на возбуждение составного ядра. Приближенно можно считать, что вся эта энергия ушла на кинетическую энергию ядер. А это значит, что при оценке $|\Psi(0)|^2$ нужно заменить E на $E + \triangle E$. Поскольку эта величина находиться в показателе экспоненты, вместе с большим множителем $m^* Z_1 Z_2$, то $|\Psi(0)|^2$ изменяется очень сильно.

[1] напомним, что в данной работе используются $Ry = \frac{1}{2}$ ат. ед

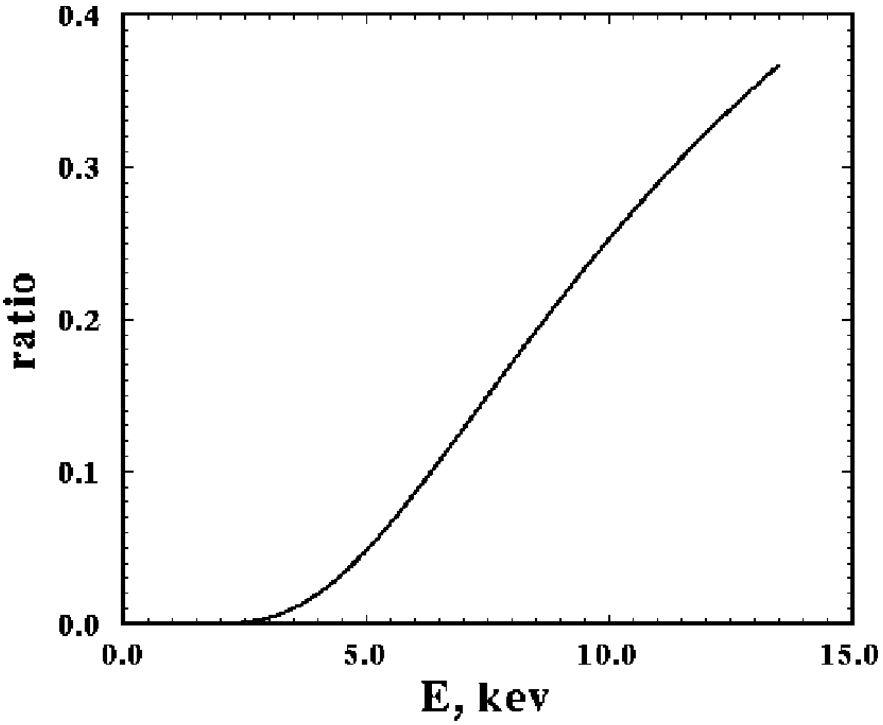

Рис. 4.5: Отношение $\Psi(0)/\Psi(0)$ для различных энергий

Заключение

В настоящей дипломной работе на примере конкретной молекулы $^{17}O^1H$, проведено исследование влияние атомных электронов на протекание ядерных процессов, как в молекулярном состоянии, так и в состоянии рассеяния ядер.

При этом были получены следующие результаты:

• Найден электронный терм молекулы $^{17}O^1H$ для всех межъядерных расстояний.

• Определены волновые функции связанного состояния и состояния рассеяния ядер в этом терме. Также найден колебательный спектр молекулы.

• Получена оценка для скорости ядерных реакций из молекулярного состояния ($\lambda \simeq S(E_.) \cdot 10^{-200}$ сек$^{-1}$), что говорит, по всей видимости, о том, что ядерные переходы из молекулярного состояния $^{17}O^1H$ невозможны. Исследовано влияние экранирования атомными электронами кулоновского отталкивания ядер, и найдено, что учет этих электронов существенно изменяет скорость ядерных реакций. Например, при средней энергии ядер на солнце, отношение $|\Psi(0)|^2/|\Psi_.|^2 \simeq 10^8$, а при энергии, порядка 10 КэВ, составляет фактор 9.

Приложением настоящей работы являются расчет резонансных ядерных реакций в молекулах с энергией, совпадающей с молекулярной, а также корректное вычисление скоростей ядерных процессов, идущих на солнце.

В заключение, считаю своим приятным долгом поблагодарить проф. В.Б. Беляева за постановку задачи и за предоставленную возможность учиться и работать в секторе малочастичных систем ОИЯИ, научного руководителя к.ф.н. Ф.М. Пенькова за большую помощь в работе и терпеливое объяснение многих непонятных для меня вопросов, Д.Е. Монахова и В.Е. Кузнецова за большую помощь в овладении операционной системой UNIX, проф. В.А. Наумова за неподдельный интерес к работе и за предоставление написанной им программы на фортране для вычисления многомерных интегралов, М.А. Писарева за большую помощь при верстке текста с использованием текстового редактора LaTeX, проф. А.Н. Валла за постоянный интерес к работе, добрые советы и помощь в решении финансовых проблем, а также к.ф.н. В.В. Пупышева за скрупулезное рецензирование настоящей работы, благодаря которому было исправлено множество стилистических опечаток и неточностей.

Приложение 1. Атомная система единиц В атомной системе единиц единица массы - масса электрона, единица длины - боровский радиус, единица заряда - заряд электрона.

$$\begin{aligned} [M] &= m_e, \\ [L] &= \frac{\hbar^2}{m_e e^2} \equiv a_b, \\ [Q] &= e. \end{aligned}$$

Зная значение постоянной тонкой структуры $\alpha = \frac{e^2}{\hbar c} = \frac{1}{137}$ находим, что $c = 137$. Поскольку $\frac{\hbar^2}{m_e e^2} = 1$, то $\hbar = 1$.

При этом единицей энергии называют величину $\frac{e^2}{a_b}$ равной примерно 27.212 эв. Она носит название *атомной единицы энергии*, представляя собой удвоенный потенциал ионизации атома водорода.

Иногда используют в два раза меньшую величину - Ридберг.

Приложение 2. Матричные элементы гамильтониана

В этом Приложении следуем [10]. Запишем волновую функцию (2.1) в виде:

$$\Psi(\xi_i) = \varepsilon_{\alpha\beta\ldots\pi}\psi_\alpha(1)\psi_\beta(2)\ldots\psi_\pi(N). \tag{4}$$

Гамильтониан представим как:

$$\hat{H} = \sum_i \hat{h}_i + \sum_{i>j} \hat{g}_{ij}, \tag{5}$$

где,

$$\hat{h}_i = -\frac{1}{2}\triangle_i - \sum_\alpha \frac{Z_\alpha}{r_{i\alpha}}, \quad \hat{g}_{ij} = \frac{1}{r_{ij}}. \tag{6}$$

Для расчета энергии нужно посчитать величину $< \Psi|\hat{H}|\Psi >$.

Итак, по порядку:

сначала одноцентровой член -$< \Psi|\sum_i \hat{h}_i|\Psi >$. Рассмотрим по отдельности каждое слагаемое, начав с первого:

$$\begin{aligned} < \Psi|\hat{h}_1|\Psi > &= \frac{1}{N!}\varepsilon_{\alpha'\beta'\ldots\pi'}\varepsilon_{\alpha\beta\ldots\pi} < \psi_{\alpha'}(1)|\hat{h}_1|\psi_\alpha(1) >< \psi_{\beta'}(2)|\psi_\beta(2) > \ldots \\ \ldots &< \psi_{\pi'}(N)|\psi_\pi(N) >= \frac{1}{N} < \psi_\alpha(1)|\hat{h}_1|\psi_\alpha(1) > . \end{aligned} \tag{7}$$

По немым индексам подразумевается суммирование и складывая все слагаемые, получаем:

$$< \Psi|\sum_i \hat{h}_i|\Psi > = < \psi_\alpha(\vec{r})|\hat{h}|\psi_\alpha(\vec{r}) > . \tag{8}$$

Рассмотрим теперь двух - центровые слагаемые, начав с $\hat{g}_{12}$:

$$\begin{aligned} < \Psi|\hat{g}_{12}|\Psi >= \frac{1}{N(N-1)} \quad [\quad &< \psi_\alpha(1)\psi_\beta(2)|\hat{g}_{12}|\psi_\alpha(1)\psi_\beta(2) > - \\ - \quad &< \psi_\alpha(1)\psi_\alpha(2)|\hat{g}_{12}|\psi_\beta(1)\psi_\beta(2) >]. \end{aligned} \tag{9}$$

При выводе этих соотношений использовались следующие, легко выводимые равенства :

$$\begin{aligned} \varepsilon_{\alpha'\beta'\ldots\pi'}\varepsilon_{\alpha\beta\ldots\pi'} &= (N-2)! \cdot (\delta_{\alpha\alpha'}\delta_{\beta\beta'} - \delta_{\alpha\beta'}\delta_{\alpha'\beta}), \\ \varepsilon_{\alpha'\beta'\ldots\pi'}\varepsilon_{\alpha\beta'\ldots\pi'} &= (N-1)! \cdot \delta_{\alpha\alpha'}. \end{aligned}$$

Теперь нам нужно просуммировать по всем индексам $i > j$. Число пар таких индексов равно $\frac{N(N-1)}{2}$, так что этот множитель сокращается и мы приходим к выражению для энергии (2.2).

Приложение 3. Уравнение Хартри - Фока

Следуя [12], введем линейные и эрмитовые операторы:

$$\hat{J}_i(\vec{r})\varphi(\vec{r}) = \left(\int \frac{\varphi_i(\vec{r}_1){\varphi_i}^*(\vec{r}_1)}{|\vec{r} - \vec{r}_1|} \right) \varphi(\vec{r}), \tag{1}$$

$$\hat{K}_i(\vec{r})\varphi(\vec{r}) = \left(\int \frac{{\varphi_i}^*(\vec{r}_1)\varphi(\vec{r}_1)}{|\vec{r} - \vec{r}_1|} \right) \varphi_i(\vec{r}). \tag{2}$$

С их помощью кулоновские и обменные интегралы могут быть записаны в форме:

$$J_{ij} = \int d\vec{r}{\varphi_i}^*(\vec{r})\hat{J}_i(\vec{r})\varphi_i(\vec{r}), \tag{3}$$

$$K_{ij} = \int d\vec{r}{\varphi_i}^*(\vec{r})\hat{K}_i(\vec{r})\varphi_i(\vec{r}). \tag{4}$$

Составим функционал энергии:

$$I = E - \sum_{i,j} \varepsilon_{ij} < \varphi_i | \varphi_j > . \tag{5}$$

Приравняем вариацию этого функционала по φ_i нулю для того, чтобы получить уравнения на φ_i. Получаем:

$$\sum_i \int d\vec{r}(\delta{\varphi_i}^*) \left[\left\{ \hat{H} + \sum_j (2\hat{J}_j - K_j) \right\} \varphi_i - \sum_j \varphi_j \varepsilon_{ji} \right] +$$

$$\sum_i \int d\vec{r}(\delta\varphi_i) \left[\left\{ \hat{H}^* + \sum_j (2\hat{J}_j{}^* - K_j{}^*) \right\} \varphi_i - \sum_j {\varphi_j}^* \varepsilon_{ij} \right] = 0. \tag{6}$$

Из независимости вариаций ${\varphi_i}^*$ и φ_i следует:

$$\left\{ \hat{H} + \sum_j (2\hat{J}_j - K_j) \right\} \varphi_i = \sum_j \varphi_j \varepsilon_{ji}, \tag{7}$$

$$\left\{ \hat{H}^* + \sum_j (2\hat{J}_j{}^* - K_j{}^*) \right\} \varphi_i = \sum_j {\varphi_j}^* \varepsilon_{ij}. \tag{8}$$

Легко убедиться в том, что матрица ε - эрмитовая. Поскольку молекулярная волновая функция инвариантна относительно унитарного преобразования, то можно совершить такое преобразование, при котором матрица ε станет диагональной. Тогда, введя оператор Фока согласно:

$$\hat{F} = \hat{H} + \sum_i (2\hat{J}_i - \hat{K}_i), \tag{9}$$

получаем

$$\hat{F}\varphi_i = \varepsilon_i \varphi_i. \tag{10}$$

Это и есть уравнение Хартри - Фока. Чтобы получить уравнения Хартри - Фока-Рутана можно действовать двумя способами. Во первых, представив каждую орбиталь

в виде разложения по неким базисным функциям, варьировать по неизвестным коэффициентам разложения. Можно же, однако, получить тот же ответ проще [7]:

$$\varphi_i = \sum_k C_{ik} u_k. \tag{11}$$

Умножим уравнение ХФ на ${u_m}^*$ и проинтегрируем. Тогда получим:

$$\sum_k C_{ik} < u_m | \hat{F} | u_k >= \varepsilon_i \sum_k C_{ik} S_{km}. \tag{12}$$

Раскрыв матричный элемент оператора Фока, получим те же уравнения, что приведены в основном тексте.

Случай разных пространственных орбиталей для частиц с разным спином рассматривается аналогично.

Приложение 4. Разложение по кислородному базису

Пусть $r' = \sqrt{r^2 + R^2 - 2rR\cos\theta}$, тогда можно получить формулу сложения ([20]):

$$\frac{e^{-\lambda r'}}{\lambda r'} = \frac{2}{\pi}\sum_{n=0}^{\infty}\left[(2n+1)\sqrt{\frac{\pi}{2\lambda r_<}}I_{n+\frac{1}{2}}(\lambda r_<) \times \sqrt{\frac{\pi}{2\lambda r_>}}K_{n+\frac{1}{2}}(\lambda r_>).\right] P_n(\cos\theta). \tag{1}$$

Здесь, $r_<, \quad r_>$- соответственно, меньшее и большее расстояние из r и R.

$I_{n+\frac{1}{2}}$ и $K_{n+\frac{1}{2}}$ - модифицированные функции Бесселя полуцелого порядка. Они являются линейно независимыми решениями дифференциального уравнения:

$$z^2w'' + 2zw' - [z^2 + n(n+1)]w = 0. \tag{2}$$

Введем обозначения:

$$f_n(z) = \sqrt{\frac{\pi}{2z}}I_{n+\frac{1}{2}}(z), \quad g_n(z) = \frac{2}{\pi}\sqrt{\frac{\pi}{2z}}K_{n+\frac{1}{2}}(z). \tag{3}$$

Для этих функций существуют реккурентные соотношения:

$$\begin{aligned} f_{n-1}(z) - f_{n+1}(z) &= (2n+1)\,\frac{f_n(z)}{z}, \\ g_{n-1}(z) - g_{n+1}(z) &= -(2n+1)\,\frac{g_n(z)}{z}, \\ z\,f_n'(z) &= z\,f_{n+1}(z) + n\,f_n(z), \\ z\,g_n'(z) &= -z\,g_{n+1} + n\,g_n. \end{aligned} \tag{4}$$

Несколько первых значений для этих функций:

$$\begin{aligned} f_0(z) &= \frac{sh(z)}{z}, \\ f_1(z) &= -\frac{sh(z)}{z^2} + \frac{ch(z)}{z}, \\ g_0(z) &= \frac{e^{-z}}{z}, \\ g_1(z) &= \frac{e^{-z}}{z}(1 + \frac{1}{z}). \end{aligned}$$

Чтобы получить выражение для самой экспоненты, без множителя $\frac{1}{r'}$, продифференцируем по λ, тогда :

$$\begin{aligned} e^{-\lambda r'} &= -\frac{\partial}{\partial\lambda}\frac{e^{-\lambda r'}}{r'} = \\ &= -\frac{\partial}{\partial\lambda}\lambda\sum_{n=0}^{\infty}\left[(2n+1)\sqrt{\frac{\pi}{2\lambda r_<}}I_{n+\frac{1}{2}}(\lambda r_<) \times \sqrt{\frac{\pi}{2\lambda r_>}}K_{n+\frac{1}{2}}(\lambda r_>)\right] P_n(\cos\theta). \end{aligned} \tag{5}$$

Проделав это и воспользовавшись реккурентными соотношениями, получим:

$$-\frac{\alpha_n(z_1, z_2)}{r} = (2n+1)f_n(z_1)g_n(z_2) + z_1 f_{n+1}(z_1)g_n(z_2) - z_2 g_{n+1}(z_2)f_n(z_1), \tag{6}$$

где $z_1 = \lambda r_<, z_2 = \lambda r_>$.

Для расчета кинетических интегралов понадобиться знание производной этой величины по r. Проделав все вычисления, получим:

при $r < R$:

$$-\frac{\partial \alpha_n(z_1, z_2)}{\partial r} = (2n+1)(n+1) f_n(z_1) g_n(z_2) + (3n+4) z_1 f_{n+1}(z_1) g_n(z_2) -$$
$$-(n+1) z_2 g_{n+1}(z_2) f_n(z_1) - z_1 z_2 g_{n+1}(z_2) f_{n+1}(z_1) + z_1^2 g_n(z_2) f_{n+2}(z_1). \quad (7)$$

при $r > R$:

$$-\frac{\partial \alpha_n(z_1, z_2)}{\partial r} = (2n+1)(n+1) f_n(z_2) g_n(z_1) - (3n+4) z_1 f_n(z_2) g_{n+1}(z_1) +$$
$$+(n+1) z_2 g_n(z_1) f_{n+1}(z_2) - z_1 z_2 g_{n+1}(z_1) f_{n+1}(z_2) + z_1^2 g_n(z_1) f_{n+2}(z_2). \quad (8)$$

Эти формулы были запрограммированы, причем из - за потери численной точности, связанной с машинным округлением пришлось перейти к quadruple precision, что позволяет сделать фортрановский компилятор на workstation *SUN*.

Приложение 5. Однократные интегралы

Для удобства все волновые функции пронумерованы так же, как в основном тексте. То есть:

$P_1 \equiv P_{1s}$,
$P_2 \equiv P_{2S}$,
$P_3 \equiv P_{2p_0}$,
$P_4 \equiv 1h$.

• **Интегралы перекрытия**.

$$\begin{array}{lcll} < k|m > & = & \delta_{l_k,l_m} \int dr P_k(r) P_m(r), & \forall k,m \in [1,3] \\ < k|4 > & = & 2\sqrt{2l_k+1} \int dr P_k(r) \alpha_{l_k}(r,R), & \forall k \in [1,3] \\ < 4|4 > & = & 1. & \end{array} \quad (1)$$

Здесь и ниже l_k - третья проекция орбитального момента k волновой функции.

• **Кулоновские интегралы взаимодействия с ядром кислорода**

$$\begin{array}{lcll} < k|\frac{1}{r}|m > & = & \delta_{l_k,l_m} \int dr \frac{P_k(r)P_m(r)}{r}, & \forall k,m \in [1,3], \\ < k|\frac{1}{r}|4 > & = & 2\sqrt{2l_k+1} \int dr \frac{P_k(r)\alpha_{l_k}(r,R)}{r}, & \forall k \in [1,3], \\ < 4|\frac{1}{r}|4 > & = & \frac{1}{R} - e^{-2R}(1+\frac{1}{R}). & \end{array} \quad (2)$$

• **Кулоновские интегралы взаимодействия с ядром водорода**

$$< k|\frac{1}{r'}|m >= \sqrt{(2l_k+1)(2l_m+1)} \sum_{l=|l_k-l_m|,2}^{l_k+l_m} \begin{pmatrix} l_k & l_m & l \\ 0 & 0 & 0 \end{pmatrix}^2 \times \quad (3)$$

$$\times \int_0^1 dx \left\{ x^l P_k(Rx) P_m(Rx) + (\frac{\Lambda}{R}-1) \frac{P_k((\Lambda-R)x+R) P_m((\Lambda-R)x+R)}{((\frac{\Lambda}{R}-1)x+1)^{l+1}} \right\}.$$

Индекс k пробегает значения от 1 до 3. Появившийся здесь параметр Λ происходит из - за того, что интегрирование проводится по конечному промежутку $(0,\Lambda)$, дело в том, что кислородные волновые функции, рассчитанные методом ХФ обрезаются естественным образом при $r = 22$ атомные единицы. Разумеется, значения интегралов не зависит от Λ. В настоящей работе, значение Λ равно 23.

Интегралы с водородной функцией также удается свести к одномерным даже без разложения по кислородному базису. Для нужных нам частных случаев с $l_k = 0$ и $l_k = 1$ они имеют следующий вид:

• $\mathbf{l_k = 0, k = 1, 2}$

$$< k|\frac{1}{r'}|4 >= 2\sqrt{2l_k+1} \int dr r P_k(r) \left\{ e^{-\frac{|r-R|}{\sqrt{2rR}}} - e^{-\frac{r+R}{\sqrt{2rR}}} \right\}. \quad (4)$$

• $\mathbf{l_k = 1, k = 3}$

$$< k|\frac{1}{r'}|4 >= \frac{2\sqrt{2l_k+1}}{\sqrt{2R}} \int dr \sqrt{r} P_k(r) \times \quad (5)$$

$$\times \quad \left\{ e^{-\frac{r+R}{\sqrt{2rR}}} \left(3\sqrt{2rR} + 2(r+R) \right) - e^{-\frac{|r-R|}{\sqrt{2rR}}} \left(\sqrt{2rR} + 2|r-R| \right) \right\}.$$

• **Кинетические интегралы**

$$
\begin{aligned}
< k| -\tfrac{1}{2}\triangle |m > &= \tfrac{1}{2}\delta_{l_k,l_m}\int dr\left\{P'_k(r)P'_m(r) + \tfrac{l_m(l_m+1)}{r^2}P_k(r)P_m(r)\right\},\\
< k| -\tfrac{1}{2}\triangle |4 > &= \sqrt{2l_k+1}\int dr\left\{P'_k(r)\alpha'_{l_k}(r,R) + \tfrac{l_m(l_m+1)}{r^2}P_k(r)\alpha_{l_k}(r,R)\right\},\\
&\forall k,m \in [1,3],\\
< 4| -\tfrac{1}{2}\triangle |4 > &= \tfrac{1}{2}.
\end{aligned}
$$

Приложение 6. Двукратные интегралы

При выводе этих интегралов нам понадобятся следующие известные формулы: [2]

$$\frac{1}{r_{12}} = \sum_{l=0}^{\infty} \frac{4\pi}{2l+1} \cdot \frac{r_<^l}{r_>^{l+1}} \sum_{m=-l}^{l} {Y_{lm}}^*(\theta_1, \varphi_1) Y_{lm}(\theta_2, \varphi_2). \tag{1}$$

$$\int d\Omega \qquad Y_{l_1 m_1}(\theta,\varphi) Y_{l_2 m_2}(\theta,\varphi) Y_{l_3 m_3}(\theta,\varphi) = \tag{2}$$
$$= \sqrt{\frac{(2l_1+1)(2l_2+1)(2l_3+1)}{4\pi}} \begin{pmatrix} l_1 l_2 l_3 \\ m_1 m_2 m_3 \end{pmatrix} \begin{pmatrix} l_1 l_2 l_3 \\ 000 \end{pmatrix}.$$

Рассматриваемые нами интегралы выглядят так:

$$[km|ln] = \int\int d\vec{r}_1 d\vec{r}_2 {u_k}^*(\vec{r}_1) u_m(\vec{r}_1) \frac{1}{r_{12}} {u_l}^*(\vec{r}_2) u_n(\vec{r}_2). \tag{3}$$

Проинтегрировав по углам, получим:

• **Интегралы с кислородными функциями**

$$[km|ln] = \sqrt{(2l_k+1)(2l_m+1)(2l_l+1)(2l_n+1)} \times \tag{4}$$
$$\times \sum_{l=0}^{2} \sum_{m=-l}^{l} (-1)^{m_k+m_l+m} \begin{pmatrix} l_k\ l_m\ l \\ -m_k m_m - m \end{pmatrix} \begin{pmatrix} l_k l_m l \\ 000 \end{pmatrix} \begin{pmatrix} l_l l_n l \\ -m_l m_n m \end{pmatrix} \begin{pmatrix} l_l l_n l \\ 000 \end{pmatrix} \times$$
$$\times \int_0^{\infty} dr_1 \int_0^{r_1} dr_2 \frac{r_2^l}{r_1^{l+1}} \left\{ P_k(r_1) P_m(r_1) P_l(r_2) P_n(r_2) + P_k(r_2) P_m(r_2) P_l(r_1) P_n(r_1) \right\}.$$
$\forall k, m, l, n \neq 4.$

• **Интегралы с одной водородной функцией**

$$[km|l4] = 2\sqrt{(2l_k+1)(2l_m+1)(2l_l+1)} \sum_{l=|l_k-l_m|,2}^{l_k+l_m} \sum_{j=|l-l_l|,2}^{l+l_l} \times \tag{5}$$
$$\times \sum_{m=-l}^{l} (-1)^{m_k+m_l+m} \begin{pmatrix} l_k\ l_m\ l \\ -m_k m_m - m \end{pmatrix} \begin{pmatrix} l_k\ l_m\ l \\ 000 \end{pmatrix} \begin{pmatrix} l_l\ j\ l \\ -m_l 0 m \end{pmatrix} \begin{pmatrix} l_l\ j\ l \\ 000 \end{pmatrix} \times$$
$$\times \int\int dr_1 dr_2 \frac{r_<^l}{r_>^{l+1}} P_k(r_1) P_m(r_1) P_l(r_2) \alpha_j(r_2, R).$$

Остальные интегралы этого типа отличаются фазовым множителем, возникающим после интегрирования по углам.

• **Интегралы с двумя водородными функциями**

$$[km|44] = 4\sqrt{(2l_k+1)(2l_m+1)} \sum_{j=|l_k-l_m|,2}^{l_k+l_m} (-1)_k^m \begin{pmatrix} l_k\ l_m\ j \\ -m_k m_m 0 \end{pmatrix} \begin{pmatrix} l_k\ l_m\ j \\ 000 \end{pmatrix} \times$$
$$\times \int\int dr_1 dr_2 \frac{r_<^j}{r_>^{j+1}} r_2 P_k(r_1) P_m(r_1) \alpha_j(2r_2, R_2). \tag{6}$$

[2] см. например: ([9])

$$[k4|m4] = (-1)^{m_k+m_m} 4\sqrt{(2l_k+1)(2l_m+1)} \sum_{l=0} \sum_{m=-l}^{l} \sum_{j_1=0} \sum_{j_2=0} \tag{7}$$

$$(-1)^m (2j_1+1)(2j_2+1) \begin{pmatrix} l_k\ l\ j_1 \\ -m_k - m 0 \end{pmatrix} \begin{pmatrix} l_k\ l\ j_1 \\ 000 \end{pmatrix} \begin{pmatrix} l_m\ l\ j_2 \\ -m_m m 0 \end{pmatrix} \begin{pmatrix} l_m\ l\ j_2 \\ 000 \end{pmatrix} \times$$

$$\times \int\int dr_1 dr_2 \frac{r_<^l}{r_>^{l+1}} P_k(r_1) \alpha_{j_1}(r_1, R) P_m(r_2) \alpha_{j_2}(r_2, R).$$

$$\forall k, m \neq 4$$

Аналогично, остальные интегралы с двумя водородными функциями отличаются фазовым множителем.

• **Интегралы с тремя водородными функциями**

$$[k4|44] = 8\Lambda\sqrt{2l_k+1}\delta_{m_k,0} \sum_{j_1=0}^{10} \sum_{j_2=|j_1-l_k|,2}^{j_1+l_k} (2j_1+1) \begin{pmatrix} l_k\ j_1\ j_2 \\ 000 \end{pmatrix}^2 \times \tag{8}$$

$$\times \int_0^1 dx \int_0^1 dy y^{j_2} \left\{ r_2 \alpha_{j_1}(r_1, R) \alpha_{j_2}(2r_2, 2R) P_k(r_1) + r_1 \alpha_{j_1}(r_2, R) \alpha_{j_2}(2r_1, 2R) P_k(r_2) \right\}.$$

Последний интеграл вычисляется аналитически:

$$[44|44] = \frac{5}{8}. \tag{9}$$

Приложение 7. Составное ядро

Фрагменты		Составное ядро				Год
$A_1(J^\pi;T)+A_2$	Энергия МэВ(КэВ)	$A_{12}(J^\pi,T)$	Энергия МэВ(КэВ)	Ширина КэВ(КэВ)	Каналы	
$^{7}Li(\frac{3}{2}^-;\frac{1}{2})+d$	16.695	$^{9}Be(?;?)$	16.671(8)	41(?)	γ	1979
$^{9}Be(\frac{3}{2}^-;\frac{1}{2})+p$	6.586	$^{10}B(4^-;?)$	6.561(1.2)	25(1.1)	α	1979
$^{9}Be(\frac{3}{2}^-;\frac{1}{2})+d$	15.817	$^{11}B(?;?)$	15.8(?)	200(?)	n,α	1980*
$^{10}B(3^+;0)+p$	8.691	$^{11}C(\frac{5}{2}^+;?)$	8.701(20)	15(1)	γ,p	1985
$^{10}B(3^+;0)+d$	25.189	$^{12}C(1^-;1)$	25.3(150)	510(10)	n,p	1985
$^{12}C(0^+;0)+d$	10.273	$^{14}N(1^-;0)$	10.226(8)	80(15)	p,γ	1991
$^{13}C(\frac{1}{2}^-;\frac{1}{2})+d$	16.160	$^{15}N(\frac{3}{2}^+;?)$	16.190(10)	450(100)	γ,n,p,t,α	1991
$^{14}N(1^+;0)+p$	7.297	$^{15}O(\frac{7}{2}^+;?)$	7.276(.6)	.7(.15)ps	γ	1991
$^{14}N(1^+;0)+d$	20.736	$^{16}O(?;?)$	20.8(?)	60(?)	n,p,γ	1982
$^{14}N(1^+;0)+d$	20.736	$^{16}O(0^-;?)$	20.857(14)	900(100)	α	1982
$^{15}N(\frac{1}{2}^-;\frac{1}{2})+p$	12.128	$^{16}O(0^+;0)$	12.049(4)	1.5(.5)	γ,α	1982
$^{14}N(1^+;0)+2p$	6.761	$^{16}F(?;?)$	6.678(?)	?(?)	?	1982
$^{15}N(\frac{1}{2}^-;\frac{1}{2})+d$	14.047	$^{17}O(\frac{9}{2}^+;\frac{11}{2})$	14.15(100)	100(?)	?	1982
$^{16}O(0^+;2)+d$	7.526	$^{18}F(2^-;1)$	7.530(2)	16.5(3)	γ,p,α	1983
$^{17}O(\frac{5}{2}^+;\frac{1}{2})+p$	5.607	$^{18}F(1^-;0+1)$	5.605(.28)	<1.2	γ,α	1983
$^{17}O(\frac{5}{2}^+;\frac{1}{2})+p$	5.607	$^{18}F(1^+;?)$	5.604(.27)	15(10)fs	γ	1983
$^{16}O(0^+;2)+2p$	4.522	$^{18}Ne(1^-;?)$	4.519(8)	<20	p	1983
$^{17}O(\frac{5}{2}^+;\frac{1}{2})+d$	13.814	$^{19}F(\frac{1}{2}^+;?)$	13.878(15)	101(?)	t	1983
$^{18}O(0^+;1)+p$	7.994	$^{19}F(\frac{5}{2}^+;?)$	8.015(2)	?	?	1983
$^{19}F(\frac{1}{2}^+;\frac{1}{2})+p$	12.848	$^{20}Ne(?;?)$	12.83(30)	55(?)	α	1983

* По данным 1985 года уровня нет

```
*********************************************************************
*     This program calculates all overlap integrals
*********************************************************************

      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=91,MODE=0)
      dimension Pnl(0:strings,3),r(0:strings),over(6,6),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)
      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lm,ln
      Common /distance/distance
      external OXIG,alpha,hybrid,dcspln

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

*********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
*********************************************************************
      do 1 i=1,MaxDim
```

```
      Xlow(i)=0.d0
    1 Xupp(i)=1.d0

      eps=1.d-5

      open (10,file='over.dat')
      open (1,file='over.err')

      over(1,1)=0.100000010781D1   ! these are results of multyply
      over(2,2)=0.100000010772D1   ! checked calculations
      over(3,3)=0.100000010763D1
      over(4,4)=1.D0
      over(1,2)=0.699308560228D-03
      over(2,1)=over(1,2)
      over(5,5)=over(3,3)
      over(6,6)=over(3,3)

      IntDim=1
      call MulSet(hybrid,result,eps,MinCal,IntDim,*1000)

      do ln=1,3
       lm=(ln-1)*(ln-2)/2
       call MulInt(hybrid,result,*1000)
       over(ln,4)=2*cut*dsqrt(1.d0*(2*lm+1))*result
       over(4,ln)=over(ln,4)
       call MulInf
      enddo


      write (10,100) ((over(i,j),j=1,6),i=1,6)
  100 format (6(D20.12))

 1000 stop
      END
```

```
************************************************************************
*     This is the function to be integrate for hybridization
************************************************************************

      double precision  function hybrid(var)
      implicit real*8 (a-h,o-z)
      dimension var(*)
      Common /moment/lm,ln
      Common /cut/cut
      x=var(1)
      r=cut*x
      hybrid=alpha(1.q0*r,1,lm)*oxig(r,ln)
      return
      END

************************************************************************
*     These are interpolated functions OXIG(x,m) .
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
************************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
```

```
      if (x.lt.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*********************************************************************
*     This program calculates integrals <u(i)|1/r|u(j)>
*********************************************************************
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)

      dimension Pnl(0:strings,3),r(0:strings),CulO(6,6),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)

      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lm,ln
      Common /distance/distance

      external alpha,dcspln,square,hybrid,OXIG

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)
```

```
*********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
*********************************************************************
      do 1 i=1,MaxDim
      Xlow(i)=0.d0
    1 Xupp(i)=1.d0

      eps=1.d-3
      open (1,file='CulO.err')
      open (2,file='CulO.dat')

      CulO(1,1)= 0.764252770332D+01    !
      CulO(1,2)= 0.108480451579D+01    ! These are results of calculations
      CulO(1,3)= 0.926147090979D+00    ! independent of distance
      CulO(2,2)= 0.126709356721D+01    !
      CulO(2,3)=-0.913805996698D+00
      CulO(3,3)= 0.110723159178D+01

      do 2 i=1,3
      do 2 j=i,3
    2 CulO(j,i)=CulO(i,j)

*********************************************************************
*      This unit below is just for checking purposes and I don't
*      use it in calculations becouse I did it before.
*********************************************************************


      IntDim=1
      call MulSet(hybrid,result,eps,MinCal,IntDim,*1000)

c      do lm=1,3
c       do ln=lm,3
c        call MulInt(square,result,*1000)
```

```
c        CulO(lm,ln)=result
c        CulO(ln,lm)=CulO(lm,ln)
c       call MulInf
c       enddo
c      enddo

*********************************************************************
*     Here it is <u(k)|1/r|u(4)>
*********************************************************************

      do ln=1,3
       lm=(ln-1)*(ln-2)/2
       factor=2*dsqrt(1.d0*(2*lm+1))
       call MulInt(hybrid,result,*1000)
       CulO(ln,4)=factor*result
       CulO(4,ln)=CulO(ln,4)
      enddo

      rd=1.d0/distance
      CulO(4,4)= rd - (1.d0+rd)*dexp(-2*distance)

      CulO(5,5)=CulO(3,3)
      CulO(6,6)=CulO(3,3)

      write (2,100) ((CulO(i,j),j=1,6),i=1,6)
  100 format (6(D20.12))

 1000 stop
      END

*********************************************************************
*     This is the function for <u(ln)|1/r|u(lm)> for lm,ln=1,3
*********************************************************************

      double precision  function square(var)
```

```
      implicit real*8 (a-h,o-z)
      parameter (delta=1.d-20)
      Common /cut/cut
      Common /moment/lm,ln
      dimension var(*)
      x=var(1)
      r=cut*x
      square=oxig(r,lm)*oxig(r,ln)/(x+delta)
      return
      END

**********************************************************************
*     This is the function to be integrate for hybridization
**********************************************************************

      double precision function hybrid(var)
      implicit real*8 (a-h,o-z)
      parameter (delta=1.d-20)
      dimension var(*)
      Common /moment/lm,ln
      Common /cut/cut
      x=var(1)
      r=x*cut
      hybrid=alpha(1.q0*r,1,lm)*oxig(r,ln)/(x+delta)
      return
      END

**********************************************************************
*     These are interpolated functions OXIG(x,m) .
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
**********************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
```

```
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*********************************************************************
*     This program calculates integrals <u(k)|1/(r-R)|u(m)>
*                  for k,m=1,6
*********************************************************************

      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)
      dimension Pnl(0:strings,3),r(0:strings),CulH(6,6),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)
      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lm,ln,l
      Common /distance/distance

      external OXIG,square,hybrid,phydro,alpha,dcspln,Vigner

      open (unit=10,file='waves',form='unformatted')
```

```
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************
      do 1 i=1,MaxDim
      Xlow(i)=0.d0
    1 Xupp(i)=1.d0

      distance=1.8d0

      eps=1.d-5

      open (1,file='CulH.err')
      open (2,file='CulH.dat')

      IntDim=1
      call MulSet(square,result,eps,MinCal,IntDim,*1000)

**********************************************************************
*      Here it calculates integrals <u(k)|1/(r-R)|u(m)> for k,m=(1,3)
**********************************************************************

      do 2 lm=1,3
       l1=(lm-1)*(lm-2)/2
```

```
        aa=dsqrt(2.d0*l1+1.d0)

        do 2 ln=lm,3
         l2=(ln-1)*(ln-2)/2
         bb=dsqrt(2.d0*l2+1.d0)
         lmin=abs(l1-l2)
         lmax=l1+l2

         do l=lmin,lmax,2
          call MulInt(square,result,*1000)
          call MulInf
          CulH(lm,ln)=CulH(lm,ln)+Vigner(l1,l2,l)*result
         enddo
        CulH(lm,ln)=aa*bb*CulH(lm,ln)
    2 CulH(ln,lm)=CulH(lm,ln)

**********************************************************************
*     Here it calculates integrals <u(4)|1/(r-R)|u(m)> for m=(1,3)
**********************************************************************

       do lm=1,3
        ln=(lm-1)*(lm-2)/2
        a1=2.d0*dsqrt(2.d0*ln+1.d0)
        call MulInt(hybrid,result,*1000)
        call MulInf
        CulH(lm,4)=result*cut
        CulH(4,lm)=CulH(lm,4)
       end do


       CulH(4,4)=1.d0

       l=0
        call MulInt(phydro,result,*1000)
        call MulInf
```

```
       CulH(5,5)=result
      l=2
       call MulInt(phydro,result,*1000)
       call MulInf
       CulH(5,5)=CulH(5,5) -result/5

      CulH(6,6)=CulH(5,5)
      write (2,100) ((CulH(i,j),j=1,6),i=1,6)

  100 format (6(D20.12))

 1000 stop
      END

**********************************************************************
*     This is the function for <u(ln)|1/|r-R||u(lm)> for lm,ln=1,3
**********************************************************************

      double precision function square(var)
      implicit real*8 (a-h,o-z)
      Common /cut/cut
      Common /moment/lm,ln,l
      Common /distance/R
      dimension var(*)
      x=var(1)
      r1=x*R
      r2=R+(cut-R)*x
      t=R/r2
      y=cut/R-1.d0

      square=x**l*oxig(r1,lm)*oxig(r1,ln)+
     +       y*oxig(r2,lm)*oxig(r2,ln)*t**(l+1)

      return
      END
```

```
************************************************************************
*     This is the function to be integrate for hybridization
************************************************************************

      double precision function hybrid(var)
      implicit real*8 (a-h,o-z)
      dimension var(*)
      Common /moment/lm,ln,l
      Common /cut/cut
      Common /distance/R
      x=var(1)
      r1=cut*x
      rmod=dabs(r1-R)
      rsum=r1+R
      rprod=dsqrt(2*r1*R)
      f1=dexp(-rsum/rprod)
      f2=dexp(-rmod/rprod)

      IF (ln.eq.0) THEN

       hybrid=r1*oxig(r1,lm)*(f2-f1)

                  ELSE

       hybrid=dsqrt(r1/2/R)*(f1*(3*rprod+2*rsum)-f2*(rprod+2*rmod))*
     *        oxig(r1,lm)

      END IF

      return
      END

************************************************************************
*     This is <u(5)|1/|r-R||u(5)> function for integration
```

```
************************************************************************

      double precision function phydro(var)
      implicit real*8 (a-h,o-z)
      dimension var(*)
      common /distance/R
      common /cut/cut
      common /moment/lm,ln,l

      x=var(1)
      const=cut-R
      r1=R*x
      r2=const*x + R
      y=R/r2
      const=const/R

      phydro=x**l*oxig(r1,3)**2 + const*oxig(r2,3)**2*y**(l+1)
      return
      end



************************************************************************
*     These are interpolated functions OXIG(x,m) .
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
************************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r
```

```
      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
       if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*********************************************************************
*     This program calculates kinetic integrals
*********************************************************************

      program kinetic
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=1)
      dimension Pnl(0:strings,3),r(0:strings),Q(6,6),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)
      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm
      Common /distance/distance

      external OXIG1,OXIG2,doxig1,doxig2,hybrid,dcspln,qin,alpha,alder

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
```

```
      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************

      do i=1,MaxDim
         Xlow(i)=0.d0
         Xupp(i)=1.d0
      enddo

      eps=1.d-5

      distance=1.8d0
      open (1,file='kinetic.err')
      open (2,file='kinetic.dat')

      IntDim=1
      call MulSet(hybrid,result,eps,MinCal,IntDim,*1000)

**********************************************************************
*     These integrals does not depend on R and this unit below is
*     made for checking purposes ;in calculations I use just before
*     calculated data
**********************************************************************


      do lk=1,3
       l1=(lk-1)*(lk-2)/2
```

```
       do lm=lk,3
        l2=(lm-1)*(lm-2)/2

        IF (l1.eq.l2) THEN

         call MulInt(qin,result,*1000)
         call MulInf
         Q(lk,lm)=cut*result/2
         Q(lm,lk)=Q(lk,lm)
        END IF
       enddo
      enddo

c       Q(1,1)=0.291087277423D+02 !
c       Q(1,2)=0.802054312411D+01 !
c       Q(2,2)=0.311295217047D+01 ! This data is result of previouse
c       Q(3,3)=0.252707655640D+01 ! calculations.
c       Q(2,1)=Q(1,2)             !
c       Q(4,4)=0.500000000000D+00 !
c       Q(5,5)=Q(3,3)             !
c       Q(6,6)=Q(3,3)             !
c
**********************************************************************
*     These integrals depend on R
**********************************************************************

c      do lk=1,3
c       l1=(lk-1)*(lk-2)/2
c       a1=dsqrt(1.d0*(2*l1+1))
c       call MulInt(hybrid,result,*1000)
c       call MulInf
c       Q(lk,4)=a1*cut*result
c       Q(4,lk)=Q(lk,4)
c     enddo
```

```
      write (2,100) ((Q(i,j),j=1,6),i=1,6)

  100 format (6(D20.12))
 1000 stop
      END


*********************************************************************
*     this is the function to be integrate for lk,lm=(1,3)
*********************************************************************

      double precision function qin(var)
      implicit real*8 (a-h,o-z)
      dimension var(*)
      parameter (deltar=1.d-10)
      common /moment/lk,lm
      common /cut/cut
      external oxig1,oxig2,doxig1,doxig2
      x=var(1)
      r=cut*x
      l1=(lm-1)*(lm-2)/2
      qin=r*r*doxig1(r)*doxig2(r)+
     +    l1*(l1+1)*oxig1(r)*oxig2(r)
      return
      END


*********************************************************************
*      This is the function for <u(4)|laplacian|u(m)>
*********************************************************************

      double precision  function hybrid(var)
      implicit real*8 (a-h,o-z)
      common /moment/lk,lm
```

```
      common /cut/cut
      parameter (deltar=1.d-10)
      dimension var(*)
      external oxig1,doxig1,alpha,alder

      x=var(1)
      r=cut*x
      l1=(lk-1)*(lk-2)/2

      hybrid=alder(1.q0*r,1,l1)*doxig1(r)+
     +       l1*(l1+1)*alpha(1.q0*r,1,l1)*oxig1(r)/(r*r+deltar)

      return
      END

*********************************************************************
*     These are derivates of oxig1(x)
*********************************************************************

      double precision  function dOXIG1(x)
      implicit real*8 (a-h,o-z)
      external oxig1,dderiv
      delta=1.d0
      call DDERIV(oxig1,x,delta,dfdx,rerr)
      dOXIG1=dfdx
      return
      END
*********************************************************************
*     These are derivates of oxig2(x)
*********************************************************************

      double precision  function dOXIG2(x)
      implicit real*8 (a-h,o-z)
      external oxig2,dderiv
      delta=1.d0
```

```
      call DDERIV(oxig2,x,delta,dfdx,rerr)
      dOXIG2=dfdx
      return
      END

*********************************************************************
*     These are interpolated functions OXIG1(x) .
*     for lm=1,2,3 we have 1s,2s,2p wave functions correspondely
*********************************************************************

      double precision function OXIG1(x)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r
      Common /moment/lk,lm

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG1=A(i,lk) + B(i,lk)*(x-r(i-1))+ C(i,lk)*(x-r(i-1))**2+
     +       D(i,lk)*(x-r(i-1))**3
       endif
    1 continue

      IF (x.le.r(0).or.x.gt.r(strings))  OXIG1=0.d0
      OXIG1= OXIG1/x
      RETURN
      END

*********************************************************************
*     These are interpolated functions OXIG2(x) .
*     for lk=1,2,3 we have 1s,2s,2p wave functions correspondely
```

```
************************************************************************

      double precision function OXIG2(x)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r
      Common /moment/lk,lm

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG2=A(i,lm) + B(i,lm)*(x-r(i-1))+ C(i,lm)*(x-r(i-1))**2+
     +       D(i,lm)*(x-r(i-1))**3
       endif
    1 continue
      IF (x.le.r(0).or.x.gt.r(strings)) OXIG2=0.d0
      OXIG2= OXIG2/x

      RETURN
      END
************************************************************************
*     This program calculates all integrals [km|ln] for k,m,l,n =(1,3)
*     and writes them into files oooo.dat and oooo.unf.
************************************************************************

      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)
      dimension Pnl(0:strings,3),r(0:strings),RES(4,4,4,4),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)
      Common /spline/A,B,C,D
      Common /r/r
```

```
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      Common /distance/distance

      external OXIG,oooo,Vigner,dcspln

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************

      do  i=1,MaxDim
          Xlow(i)=0.d0
          Xupp(i)=1.d0
      enddo


      eps=1.d-3
      open (1,file='oooo.err')
      open (2,file='oooo.dat')
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
      IntDim=2
      call MulSet (oooo,result,eps,MinCal,IntDim,*1000)
ccccccccccccccccccccccccccccccccccccccccccccccccccccc

      do 1 lk=1,3
       l1=(lk-1)*(lk-2)/2
       a1=dsqrt(1.d0*(2*l1+1))

      do 1 lm=lk,3
       l2=(lm-1)*(lm-2)/2
       b1=dsqrt(1.d0*(2*l2+1))
       number1=10*lk+lm

      do 1 ll=1,3
       l3=(ll-1)*(ll-2)/2
       c1=dsqrt(1.d0*(2*l3+1))

      do 1 ln=ll,3
       l4=(ln-1)*(ln-2)/2
       d1=dsqrt(1.d0*(2*l4+1))
       number2=10*ll+ln

         IF (number1.LE.number2) THEN

          do 2 l=0,2
           coef=Vigner(l1,l2,l)*Vigner(l3,l4,l)

           IF (coef.NE.0.d0) THEN
            call MulInt(oooo,result,*1000)
            call MulInf
            RES(lk,lm,ll,ln)=RES(lk,lm,ll,ln) +result*coef
           ENDIF
    2     continue
```

```
          RES(lk,lm,ll,ln)=a1*b1*c1*d1*cut*RES(lk,lm,ll,ln)

          write (2,100) RES(lk,lm,ll,ln)
         ENDIF
    1 continue
  100 format (D20.12)
 1000 stop
      END

*********************************************************************
*     This function is [km|ln] for k,m,l,n =(1,3)
*********************************************************************

      double precision function oooo(var)
      implicit real*8 (a-h,o-z)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      dimension var(*)
      external oxig
      x=var(1)
      y=var(2)
      r1=cut*x
      r2=r1*y
      t1=oxig(r1,lk)*oxig(r1,lm)*oxig(r2,ll)*oxig(r2,ln)
      t2=oxig(r2,lk)*oxig(r2,lm)*oxig(r1,ll)*oxig(r1,ln)
      oooo=(t1+t2)*y**l
      return
      END

*********************************************************************
*     These are interpolated functions OXIG(x,m) .
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
*********************************************************************

      double precision function OXIG(x,m)
```

```
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*********************************************************************
*     This program calculates integrals [km|l4]
*********************************************************************
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)

      dimension Pnl(0:strings,3),r(0:strings),RES(4,4,4,4),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)

      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      Common /distance/distance

      external OXIG,alpha,oooh,dcspln
```

```
      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************
      do  i=1,MaxDim
          Xlow(i)=0.d0
          Xupp(i)=1.d0
      enddo

      open (1,file='oooh.err')  ! here I put error's messages from Mul
      open (2,file='oooh.dat')  ! here I put data in formated way

      eps=1.d-3
      IntDim=2

      call MulSet(oooh,result,eps,MinCal,IntDim,*1000)

      do 1 lk=1,3
       l1=(lk-1)*(lk-2)/2
       a1=dsqrt(1.d0*(2*l1+1))
```

```
      do 1 lm=lk,3
       l2=(lm-1)*(lm-2)/2
       b1=dsqrt(1.d0*(2*l2+1))
       lmin=abs(l1-l2)
       lmax=l1+l2

      do 1 ll=1,3
       l3=(ll-1)*(ll-2)/2
       c1=dsqrt(1.d0*(2*l3+1))

        do 2 l=lmin,lmax,2
         lnmin=abs(l-l3)
         lnmax=l+l3
         coef=Vigner(l1,l2,l)
         Rint=0.d0

        do 3 ln=lnmin,lnmax,2
         coef=coef*Vigner(l3,l,ln)

         IF (coef.ne.0.d0) THEN
          call MulInt(oooh,result,*1000)
          call MulInf
          factor=coef*(2*ln+1)
          Rint=Rint+factor*result
         END IF
   3   continue
   2   RES(lk,lm,ll,4)=RES(lk,lm,ll,4)+Rint
       RES(lk,lm,ll,4)=2*cut*a1*b1*c1*RES(lk,lm,ll,4)
   1   write (2,100) RES(lk,lm,ll,4)

 100 format (D20.12)
1000 stop
     END
```

****

```
*     This is the function for [km|l4] to be integrate
************************************************************************

      double precision function oooh(var)
      implicit double precision (a-h,o-z)
      external oxig,alpha
      dimension var(*)
      common /cut/cut
      common /moment/lk,lm,ll,ln,l
      x=var(1)
      y=var(2)
      r1=cut*x
      r2=r1*y
      t1=oxig(r1,lk)*oxig(r1,lm)*oxig(r2,ll)*alpha(r2*1.q0,1,ln)
      t2=oxig(r2,lk)*oxig(r2,lm)*oxig(r1,ll)*alpha(r1*1.q0,1,ln)
      oooh=y**l*(t1+t2)
      RETURN
      END

************************************************************************
*     These are interpolated wave functions Pnl(x,m)
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
************************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
```

```
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
        endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*********************************************************************
*     This program calculates integrals [km|44]
*********************************************************************
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)

      dimension Pnl(0:strings,3),r(0:strings),RES(4,4,4,4),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)

      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      Common /distance/distance

      external OXIG,alpha,oohh,dcspln

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
```

```
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************
      do  i=1,MaxDim
          Xlow(i)=0.d0
          Xupp(i)=1.d0
      enddo

      open (1,file='oohh.err')  ! here I put error's messages from Mul
      open (2,file='oohh.dat')  ! here I put data in formated way

      eps=1.d-3
      IntDim=2
      call MulSet(oohh,result,eps,MinCal,IntDim,*1000)

      do 1 lk=1,3
       l1=(lk-1)*(lk-2)/2
       a1=dsqrt((2*l1+1)*1.d0)

       do 1 lm=lk,3
        l2=(lm-1)*(lm-2)/2
        b1=dsqrt((2*l2+1)*1.d0)
        lmin=abs(l1-l2)
        lmax=l1+l2

        do 2 l=lmin,lmax,2
         call MulInt(oohh,result,*1000)
    2   RES(lk,lm,4,4)=RES(lk,lm,4,4)+Vigner(l1,l2,l)*result
        RES(lk,lm,4,4)=4*cut*a1*b1*RES(lk,lm,4,4)
    1 write (2,100) RES(lk,lm,4,4)
```

```
  100 format (D20.12)
 1000 stop
      END

*********************************************************************
*     This is the function for [km|44] to be integrate
*********************************************************************

      double precision function oohh(var)
      implicit double precision (a-h,o-z)
      external oxig,alpha
      dimension var(*)
      common /cut/cut
      common /moment/lk,lm,ll,ln,l
      x=var(1)
      y=var(2)
      r1=cut*x
      r2=r1*y
      t1=r2*oxig(r1,lk)*oxig(r1,lm)*alpha(r2*1.q0,2,l)
      t2=r1*oxig(r2,lk)*oxig(r2,lm)*alpha(r1*1.q0,2,l)
      oohh=y**l*(t1+t2)
      RETURN
      END

*********************************************************************
*     These are interpolated wave functions Pnl(x,m)
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
*********************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
```

```
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*****************************************************************
*     This program calculates integrals [k4|l4]
*****************************************************************
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)

      dimension Pnl(0:strings,3),r(0:strings),RES(4,4,4,4),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)

      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      Common /distance/distance

      external OXIG,alpha,ohoh,dcspln

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
```

```
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
      NDIM=strings
      M=components
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

**********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
**********************************************************************
      do  i=1,MaxDim
          Xlow(i)=0.d0
          Xupp(i)=1.d0
      enddo

      open (1,file='ohoh.err')  ! here I put error's messages from Mul
      open (2,file='ohoh.dat')  ! here I put data in formated way

      eps=1.d-3
      IntDim=2
      call MulSet(ohoh,result,eps,MinCal,IntDim,*1000)

      do 1 lk=1,3
       l1=(lk-1)*(lk-2)/2
       a1=dsqrt(1.d0*(2*l1+1))

       do 1 ll=lk,3
        l2=(ll-1)*(ll-2)/2
        b1=dsqrt(1.d0*(2*l2+1))

        do 2 lm=0,9
```

```
         n1=2*lm+1
         Rint1=0.d0

         do 3 ln=0,9
          n2=2*ln+1
          Rint2=0.d0

          do 4 l=0,10
           coef=Vigner(l1,lm,l)*Vigner(l2,ln,l)

           IF (coef.NE.0.d0) THEN
            call MulInt(ohoh,result,*1000)
            call MulInf
            Rint2=Rint2+coef*result
           END IF

    4     continue

    3    Rint1=Rint1+Rint2*n2
    2   RES(lk,4,ll,4)=RES(lk,4,ll,4)+Rint1*n1

      RES(lk,4,ll,4)=4*cut*a1*b1*RES(lk,4,ll,4)
    1 write (2,100) RES(lk,4,ll,4)

  100 format (D20.12)
 1000 stop
      END

**********************************************************************
*     This is the function for [k4|l4] to be integrate
**********************************************************************

      double precision function ohoh(var)
      implicit double precision (a-h,o-z)
      external oxig,alpha
```

```
      dimension var(*)
      common /cut/cut
      common /moment/lk,lm,ll,ln,l
      x=var(1)
      y=var(2)
      r1=cut*x
      r2=r1*y
      t1=alpha(1.q0*r1,1,lm)*alpha(1.q0*r2,1,ln)*
     *                  oxig(r1,lk)*oxig(r2,ll)
      t2=alpha(1.q0*r2,1,lm)*alpha(1.q0*r1,1,ln)*
     *                  oxig(r2,lk)*oxig(r1,ll)

      ohoh=(t1+t2)*y**l

      RETURN
      END

*********************************************************************
*     These are interpolated wave functions Pnl(x,m)
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
*********************************************************************

      double precision function OXIG(x,m)
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
```

```
       endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*****************************************************************
*     This program calculates integrals [k4|44]
*****************************************************************
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3,MaxDim=2,MinCal=1,MODE=0)

      dimension Pnl(0:strings,3),r(0:strings),RES(4,4,4,4),
     ,          A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components)

      Common /spline/A,B,C,D
      Common /r/r
      Common /MulLim/Xlow(MaxDim),Xupp(MaxDim)
      Common /cut/cut
      Common /moment/lk,lm,ll,ln,l
      Common /distance/distance

      external OXIG,alpha,ohhh,dcspln

      open (unit=10,file='waves',form='unformatted')
      read (10) ((Pnl(i,m),i=0,strings),m=1,3)
      read (10) (r(i),i=0,strings)
      close (10)

      cut=23.d0
      distance=1.8d0

      N=strings
      NDIM=strings
      M=components
```

```
      call dcspln (N,r,M,Pnl,NDIM,MODE,A,B,C,D)

*********************************************************************
*     Here it was reading wave functions and radii from file 'waves'
*     and spline parameters setting was done
*********************************************************************
      do  i=1,MaxDim
          Xlow(i)=0.d0
          Xupp(i)=1.d0
      enddo

      open (1,file='ohhh.err')  ! here I put error's messages from Mul
      open (2,file='ohhh.dat')  ! here I put data in formated way

      eps=1.d-3
      IntDim=2
      call MulSet(ohhh,result,eps,MinCal,IntDim,*1000)

      do lk=1,3
       l1=(lk-1)*(lk-2)/2
       a1=dsqrt(1.d0*(2*l1+1))

       do 1 lm=0,9
        n1=2*lm+1
        llmin=abs(lm-l1)
        llmax=lm+l1
        Rint=0.d0

        do 2 ll=llmin,llmax,2

         call MulInt(ohhh,result,*1000)
         call MulInf
    2   Rint=Rint+Vigner(l1,lm,ll)*result

    1  RES(lk,4,4,4)=RES(lk,4,4,4)+Rint*n1
```

```
       RES(lk,4,4,4)=8*cut*a1*RES(lk,4,4,4)
       write (2,100) RES(lk,4,4,4)
      enddo


  100 format (D20.12)
 1000 stop
      END

*********************************************************************
*     This is the function for [k4|44] to be integrate
*********************************************************************

      double precision function ohhh(var)
      implicit double precision (a-h,o-z)
      external oxig,alpha
      dimension var(*)
      common /cut/cut
      common /moment/lk,lm,ll,ln,l
      x=var(1)
      y=var(2)
      r1=cut*x
      r2=r1*y
      t1=r2*oxig(r1,lk)*alpha(1.q0*r1,1,lm)*alpha(1.q0*r2,2,ll)
      t2=r1*oxig(r2,lk)*alpha(1.q0*r2,1,lm)*alpha(1.q0*r1,2,ll)
      ohhh=y**ll*(t1+t2)
      RETURN
      END

*********************************************************************
*     These are interpolated wave functions Pnl(x,m)
*     for m=1,2,3 we have 1s,2s,2p wave functions correspondely
*********************************************************************

      double precision function OXIG(x,m)
```

```
      implicit real*8 (a-h,o-z)
      parameter (strings=400,components=3)
      dimension A(strings,components),B(strings,components),
     ,          C(strings,components),D(strings,components),
     ,          r(0:strings)
      Common /spline/A,B,C,D
      Common /r/r

      do 1 i=1,strings
       if (x.gt.r(i-1).and.x.le.r(i)) then
        OXIG=A(i,m) + B(i,m)*(x-r(i-1))+ C(i,m)*(x-r(i-1))**2+
     +       D(i,m)*(x-r(i-1))**3
       endif
    1 continue
      if (x.le.r(0).or.x.gt.r(strings)) OXIG=0.d0
      return
      END
*************************************************************************
*     This is the alpha function that figures in expansion :
*                      infty
*        exp(-l*R)=1/r SUM  (2n+1)*alpha(l*r,l*rho)Pn(cos(theta))
*                      n=0
*
*     where R=dsqrt(r**2+rho**2-2*r*rho*cos(theta))
*************************************************************************

      double precision function alpha(z,l,n)
      implicit real*16 (b-h,o-z)
      real*8 a
      integer l,n,i
      parameter (delta=2.5q-2)
      dimension f(0:40),g(0:40)
      common /distance/a
      R=qextd(a)
      z1=l*qmin1(z,R)
```

```
  z2=l*qmax1(z,R)

  IF (z1.le.delta) GOTO 2

  rz1=1.q0/z1
  rz2=1.q0/z2

  f(0)=qsinh(z1)*rz1
  f(1)=(qcosh(z1)-f(0))*rz1
  g(0)=qexp(-z2)*rz2
  g(1)=g(0)*(1.q0+rz2)

  IF (n.gt.0) GOTO 1

  beta=f(0)*g(0)+g(0)*z1*f(1)-z2*f(0)*g(1)
  alpha=-dbleq(beta*z)

  RETURN

1 do i=1,n
     f(i+1)=f(i-1)-(2.q0*i+1.q0)*rz1*f(i)
     g(i+1)=g(i-1)+(2.q0*i+1.q0)*rz2*g(i)
  enddo
  beta=f(n)*g(n)+g(n)*(z1*f(n+1)+n*f(n))+
 +                f(n)*(-z2*g(n+1)+n*g(n))
  alpha=-dbleq(beta*z)
  RETURN
2 IF (n.eq.0) THEN
     alpha=dbleq(qexp(-z2)*z)
              ELSE
     alpha=0.d0
  ENDIF
  RETURN
  END
```

```
************************************************************************
*     This is the derivative of alpha function that figures in expansion :
*                       infty
*        exp(-l*R)=1/r SUM  (2n+1)*alpha(l*r,l*rho)Pn(cos(theta))
*                       n=0
*
*     where R=dsqrt(r**2+rho**2-2*r*rho*cos(theta))
************************************************************************

      double precision function alder(z,l,n)
      implicit real*16 (b-h,o-z)
      real*8 a
      integer l,n,i
      parameter (delta=2.5q-2)
      dimension f(0:40),g(0:40)
      common /distance/a
      R=qextd(a)
      z1=l*qmin1(z,R)
      z2=l*qmax1(z,R)

      IF (z1.le.delta) GOTO 2

      rz1=1.q0/z1
      rz2=1.q0/z2

      f(0)=qsinh(z1)*rz1
      f(1)=(qcosh(z1)-f(0))*rz1
      f(2)=f(0)-3*rz1*f(1)
      g(0)=qexp(-z2)*rz2
      g(1)=g(0)*(1.q0+rz2)
      g(2)=g(0)+3*rz2*g(1)


      IF (n.gt.0) GOTO 1
```

```
      IF (z.le.R) THEN

      beta=f(0)*g(0)+4*z1*f(1)*g(0)-z2*g(1)*f(0)-
     -           z1*z2*f(1)*g(1)+z1*z1*f(2)*g(0)
      alder=-dbleq(beta)

                  ELSE

      beta=f(0)*g(0)-4*z2*g(1)*f(0)+z1*f(1)*g(0)-
     -           z1*z2*f(1)*g(1)+z2*z2*f(0)*g(2)
      alder=-dbleq(beta)

      END IF

      RETURN

    1 do i=2,n+1
         f(i+1)=f(i-1)-(2.q0*i+1.q0)*rz1*f(i)
         g(i+1)=g(i-1)+(2.q0*i+1.q0)*rz2*g(i)
      enddo

      IF (z.le.R) THEN

      beta=(2*n+1)*(n+1)*f(n)*g(n)+(3*n+4)*z1*f(n+1)*g(n)-
     -     (n+1)*z2*g(n+1)*f(n)-z1*z2*f(n+1)*g(n+1)+z1*z1*f(n+2)*g(n)
      alder=-dbleq(beta)

                  ELSE

      beta=(2*n+1)*(n+1)*f(n)*g(n)-(3*n+4)*z2*g(n+1)*f(n)+
     +     (n+1)*z1*f(n+1)*g(n)-z1*z2*f(n+1)*g(n+1)+z2*z2*g(n+2)*f(n)
      alder=-dbleq(beta)

      END IF
```

```
      RETURN

    2 IF (n.eq.0) THEN
         alder=dbleq(qexp(-z2))
                  ELSE
         alder=0.d0
      ENDIF
      RETURN
      END


*************************************************************************
*    here are two real*8 functions Vigner(i,j,k) :i,j,k are integer*4, and
*    fac(n): n is integer*4. fac(n) -calculates factorial of n, so fac(n)=n!
*    and Vigner(i,j,k) is squared 3j symbol with zero magnet numbers.
*************************************************************************

      double precision function Vigner(i,j,k)
      implicit real*8 (V,r,f)
      implicit integer*4 (a,b,c,s,i,j,k)
      external fac
      Vigner=0.d0
      sum=i+j+k
      if (sum.eq.0) then
       Vigner=1.d0
       return
      endif
       if(mod(sum,2).eq.0) then
        a=sum -2*i
        b=sum -2*j
        c=sum -2*k
         if(a.ge.0.and.b.ge.0.and.c.ge.0) then
          Vigner=fac(a)*fac(b)*fac(c)/fac(sum+1)
          residue=fac(a/2)*fac(b/2)*fac(c/2)/fac(sum/2)
```

```
       Vigner=Vigner/residue/residue
      endif
     endif
    return
    END

    double precision function fac(n)
    implicit integer*4 (n)
    fac=1.d0
    if(n) 1,2,3
1   fac=0.d0
2   return
3   do 4 i=1,n
     fac=fac*i
4   continue
    return
    end
    program matrix
    implicit real*8 (a-h,o-z)
    parameter (charge1=8,charge2=1)

    Common /INTEGRALS/CulO(6,6),CulH(6,6),Q(6,6),
   *                  over(4,4),ee(6,6,6,6)
    Common /CHECKING/ Trace(4,4),VR(4),over1(4,4),C(4,4), D(6,6)
    Common /distance/distance
    Common /outINFO/Energy(4),E5,Etot

    dimension overinv(4,4),F(4,4),H(6,6)
    dimension G(4,4),VI(4),WORK(10),dnorm(4)


    external  input,check,DINV,EISRG1

    open (1,file='CulO.dat')
    open (2,file='CulH.dat')
```

```
      open (3,file='kinetic.dat')
      open (4,file='over.dat')
      open (5,file='oooo.dat')
      open (6,file='oooh.dat')
      open (7,file='oohh.dat')
      open (8,file='ohoh.dat')
      open (9,file='ohhh.dat')
      open (20,file='output.dat')
      open (21,file='term.dat')
      open (30,file='term0.dat')
      call InputSet
      call OutSet

      DO DISTANCE=1.D-1,6.D0,1.D-01

      call Input

**********************************************************************
*    Now all data is read
**********************************************************************

      do i=1,6
       do j=1,6
        CulO(i,j)=charge1*CulO(i,j)
        CulH(i,j)=charge2*CulH(i,j)
       enddo
      enddo

**********************************************************************
*     Determining input and iteration's independent data
**********************************************************************

      D(5,5) =1.d0/over(3,3)
      D(6,6) =1.d0/over(3,3)
```

```
      do l=1,4
       do n=1,4
        D(l,n)=0.d0
        do i=1,3
         D(l,n)=D(l,n)+C(l,i)*C(n,i)
        end do
       end do
      end do

      do k=1,6
       do m=1,6
        H(k,m)=Q(k,m) -CulO(k,m) -CulH(k,m)
       enddo
      enddo

      N=4
      IDIM=4

      do i=1,4
       do j=1,4
        overinv(i,j)=over(i,j)
        over1(i,j)  =over(i,j)
       enddo
      enddo

      call DINV(N,OVERINV,IDIM,IR,IFAIL)


*************************************************************************
*     At this point iterations begin
*************************************************************************

      DO iter=0,100

      do k=1,4
```

```
        do m=1,4
         y1=0.d0
         y2=0.d0

         do l=1,5
          do n=1,5
           y1=y1 + D(l,n)*ee(k,m,l,n)
           y2=y2 + D(l,n)*ee(k,n,l,m)
          enddo
         enddo
          F(k,m)  =H(k,m) + 2*y1 - y2  +
     *            D(6,6)*(ee(k,m,6,6)-ee(k,6,6,m)/2)
        enddo
       enddo

ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c     Now there are prepared two dimensions F and over
c     with them I want to solve the equation F*X=E*over*X
c     where X is  eigenvector and E is eigenvalue of F
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc


       do i=1,4
        do j=1,4
         G(i,j)  =0.d0

         do k=1,4
          G(i,j)=G(i,j)  + overinv(i,k)*F(k,j)
         enddo
         Trace(i,j)  =G(i,j)
        enddo
       enddo


       NM=4
```

```
N=4
call EISRG1 (NM,N,G,VR,VI,C,IERR,WORK)

do i=1,4
 dnorm(i)=0.d0
 do k=1,4
  do m=1,4
   dnorm(i)=dnorm(i)+C(k,i)*C(m,i)*over(k,m)
  enddo
 enddo
enddo


do i=1,4
 do l=1,4
  C(l,i)=C(l,i)/dsqrt(dnorm(i))
 enddo
enddo


do l=1,4
 do n=1,4
  D(l,n)  =0.d0

  do i=1,4
   IF(VR(i).LT.0.d0) D(l,n)=D(l,n)+ C(l,i)*C(n,i)
  end do

 enddo
enddo

END DO

CALL CHECK
```

```
************************************************************************
*     Energy calculation is presented here
************************************************************************

************************************************************************
*     Energy of p-orbitals:
************************************************************************

      E5=0.d0

      do l=1,6
       do n=1,6
        E5  =E5   + D(l,n)*(2*ee(5,5,l,n) - ee(5,n,l,5))
       end do
      end do

      E5=E5+H(5,5)      ! energy of 5-th orbital
      E5=E5/over(3,3)

************************************************************************
*     Total molecular energy:
************************************************************************

      E =0.d0
      E1=0.d0

      do i=1,4
       IF (VR(i).LT.0.d0) E=E+VR(i)
       ENERGY(i)=VR(i)
      enddo

      do k=1,4
       do m=1,4
        E1=E1+D(k,m)*H(k,m)
       end do
```

```
      end do

      Etot=E + E1 +3*E5/2 +D(5,5)*H(5,5)
      Etot =2*Etot
      write (30,500) distance,Etot

      call OutPut

      IF (distance.LE.3.8d0) THEN
       Pot=Etot +!-2*charge1*charge2/distance+
c     +                          149.07734d0
       write (21,500) distance,Pot
                             ELSE
       write (21,500) distance,Pot
      END IF

      END DO ! close DO statement on distance variable.

      call OutClose

  100 format (6(D20.12))
  200 format (D20.12,x,4(I1,x))
  300 format (4(D20.12))
  400 format (D20.12)
  500 format (F3.1,x,F10.5)

      END

**********************************************************************
*     this subroutine reads from units 1-9 values of various integrals
*     then prepares dimensions CulO,CulH,Q,over and ee with all
*     needed integrals.
**********************************************************************

      subroutine InputData
```

```
      implicit real*8 (a-h,o-z)

      common /INTEGRALS/ CulO(6,6),CulH(6,6),Q(6,6),
     *                   over(4,4),ee(6,6,6,6)


      ENTRY InputSet

*************************************************************************
*     From this point it reads and set oooo dimension
*************************************************************************

      do i=1,13
       read (5,200) f,lk,lm,ll,ln
       ee(lk,lm,ll,ln)=f
       ee(lm,lk,ll,ln)=f
       ee(lm,lk,ln,ll)=f
       ee(lk,lm,ln,ll)=f
       ee(ll,ln,lk,lm)=f
       ee(ll,ln,lm,lk)=f
       ee(ln,ll,lm,lk)=f
       ee(ln,ll,lk,lm)=f
      enddo

      do i=1,4
       read(5,200) f,lk,lm,ll,ln

       ee(5,lm,6,ln)=f
       ee(lm,5,ln,6)=f
       ee(6,ln,5,lm)=f
       ee(ln,6,lm,5)=f

       ee(6,lm,5,ln)=f
       ee(lm,6,ln,5)=f
       ee(5,ln,6,lm)=f
```

```
        ee(ln,5,lm,6)=f

       end do

       do i=1,4
        read(5,200) f,lk,lm,ll,ln
        ee(5,5,ll,ln)=f
        ee(5,5,ln,ll)=f
        ee(ll,ln,5,5)=f
        ee(ln,ll,5,5)=f

        ee(6,6,ll,ln)=f
        ee(6,6,ln,ll)=f
        ee(ll,ln,6,6)=f
        ee(ln,ll,6,6)=f
       end do

       read(5,200) f,lk,lm,ll,ln
       ee(5,6,6,5)=f
       ee(6,5,5,6)=f

       read(5,200) f,lk,lm,ll,ln
       ee(5,5,5,5)=f
       ee(6,6,6,6)=f

**********************************************************************

       ENTRY Input

**********************************************************************

       read (1,100) ((CulO(i,j),j=1,6),i=1,6)
       read (2,100) ((CulH(i,j),j=1,6),i=1,6)
       read (3,100) ((Q (i,j),j=1,6),i=1,6)
       read (4,300) ((over(i,j),j=1,4),i=1,4)
```

```
*********************************************************************
*     From this point it reads and set oooh dimension
*********************************************************************


      do i=1,18
       read (6,200) f,lk,lm,ll,ln
       ee(lk,lm,ll,ln)=f
       ee(lk,lm,ln,ll)=f
       ee(lm,lk,ln,ll)=f
       ee(lm,lk,ll,ln)=f

       ee(ll,ln,lk,lm)=f
       ee(ll,ln,lm,lk)=f
       ee(ln,ll,lk,lm)=f
       ee(ln,ll,lm,lk)=f
      enddo

      do i=1,3
       read(6,200) f,lk,lm,ll,ln
       ee(lk,6,4,5)=f
       ee(6,lk,5,4)=f
       ee(4,5,lk,6)=f
       ee(5,4,6,lk)=f

       ee(lk,5,4,6)=f
       ee(5,lk,6,4)=f
       ee(4,6,lk,5)=f
       ee(6,4,5,lk)=f
      end do

      do i=1,3
```

```
       read(6,200) f,lk,lm,ll,ln
       ee(5,5,lk,4)=f
       ee(5,5,4,lk)=f
       ee(lk,4,5,5)=f
       ee(4,lk,5,5)=f

       ee(6,6,lk,4)=f
       ee(6,6,4,lk)=f
       ee(lk,4,6,6)=f
       ee(4,lk,6,6)=f
      enddo

**********************************************************************
*     From this point it reads oohh:
**********************************************************************

      do i=1,6
       read(7,200) f,lk,lm,ll,ln
       ee(lk,lm,4,4)=f
       ee(lm,lk,4,4)=f
       ee(4,4,lk,lm)=f
       ee(4,4,lm,lk)=f
      end do

      read(7,200) f,lk,lm,ll,ln
      ee(5,5,4,4)=f
      ee(6,6,4,4)=f
      ee(4,4,5,5)=f
      ee(4,4,6,6)=f

**********************************************************************
*     From this point it reads ohoh:
**********************************************************************

      do i=1,6
```

```
       read(8,200) f,lk,lm,ll,ln
       ee(lk,4,ll,4)=f
       ee(4,lk,ll,4)=f
       ee(4,lk,4,ll)=f
       ee(lk,4,4,ll)=f

       ee(ll,4,lk,4)=f
       ee(ll,4,4,lk)=f
       ee(4,ll,4,lk)=f
       ee(4,ll,lk,4)=f
      end do

      read(8,200) f,lk,lm,ll,ln
      ee(5,4,6,4)=f
      ee(4,5,4,6)=f
      ee(6,4,5,4)=f
      ee(4,6,4,5)=f

*********************************************************************
*     From this point it reads ohhh:
*********************************************************************

      do i=1,3
       read(9,400) f
       ee(i,4,4,4)=f
       ee(4,i,4,4)=f
       ee(4,4,i,4)=f
       ee(4,4,4,i)=f
      enddo

      ee(4,4,4,4)=5.d0/8

c      do 1 i=1,6
c      do 1 j=1,6
c      do 1 l=1,6
```

```
c      do 1 k=1,6
c         IF (ee(i,j,k,l).NE.0.d0) write(*,200) ee(i,j,k,l),i,j,k,l
c    1 continue

  100 format (6(D20.12))
  200 format (D20.12,x,4(I1,x))
  300 format (4(D20.12))
  400 format (D20.12)
      RETURN
      END
    ***********************************************************************
*     This subrotine prints output information into file,connected to
*     unit 20
***********************************************************************

      subroutine out
      implicit real*8 (a-h,o-z)
      dimension VR(4)
      Common/distance/distance
      Common/outINFO/VR,E5,Etot
      ENTRY OutSet

      write (20,*) '_____________________________________________________
     _________________________________'
      write (20,100)
      write (20,*) '-----------------------------------------------------
     _________________________________'
      RETURN
      ENTRY Output

       Etot =Etot + 16/distance

***********************************************************************
*     This unit below sorts dimension VR in increasing order
***********************************************************************
```

```
    1 i=1
    2 IF (VR(i).LE.VR(i+1)) THEN
       i=i+1
      IF (i.EQ.4) GOTO 3
       GOTO 2

      ELSE
       t=VR(i)
       VR(i)=VR(i+1)
       VR(i+1)=t
       GOTO 1
      END IF


    3 write (20,101) distance, VR(1),VR(2),VR(3),VR(4),E5,Etot
      RETURN

      ENTRY OutClose
      write (20,*) '-----------------------------------------------------
     --------------------------------'




  100 format ('|', ' distance ',' | ','E_1s     ',' | ',
     &'E_2s     ',' | ','E_3s     ',' | ','E_4s     ',' | ','E_2p     ',
     &' | ','E_mol    ',' |')
  101 format (7('|',F10.5,x),'|')
  102 format ('_____________________________________________________
     &_________________________________')
      END
  *** ****************************************************************
*     This subroutine calculates difference between traces of input
*     and eigenvalues matrixes and check the orthogonality of
*     eigenvectors.All this information writes under unit 11 into
```

```
*     file check.dat
**********************************************************************

      subroutine check

      implicit real*8 (a-h,o-z)
      dimension delta(4,4)
      Common /CHECKING/Trace(4,4),VR(4),over(4,4),C(4,4), D(6,6)
      Common /distance/distance

      open(11,file='check.dat')

      trace1=0.d0
      trace2=0.d0

      do i=1,4
       trace1=trace1 + Trace(i,i)
       trace2=trace2 + VR(i)
      enddo
      differ  =trace1-trace2

      do i=1,4
       do j=1,4
        delta(i,j)  =0.d0
        do k=1,4
         do m=1,4
          delta(i,j)  =delta(i,j)    + C(k,i)*C(m,j)*over(k,m)
         enddo
        enddo
       enddo
      enddo



      write(11,100) distance
```

```
      write(11,110) differ
      write(11,120) ((delta(i,j),j=1,4),i=1,4)

  100 format ('at distance=',F10.5)
  110 format ('difference between traces=',D20.12)
  120 format (4(D20.12))

      END
*********************************************************************
*     This program finds energy spectrum for nuclears in molecular
*      term and also prints out corresponding wave function
*********************************************************************

      implicit real*8 (a-h,o-z)
      PARAMETER (NPM=10000)
      PARAMETER (O17=16.999133d0,H1=1.00782522d0,El=0.5110041d0,
     ,           A=931.4812d0)
      DIMENSION X(NPM),Y(NPM)
      DIMENSION SY(4,NPM)
      common/int_par/SY,X,Y,Ntot
      common /Moment/L
      common /mass/Rm

      open (10,file='term.dat')
      open (1,file='energy.dat')
      open (2,file='poten.dat')
      L=0

      Rm=A*O17*H1/El/(O17+H1) !reduced mass in electron mass unit


        i=0
    1   i=i+1
      IF(i.gt.npm) then
       print*," Data limits exceeded, you have to decrease lines= ",npm
```

```
       GOTO 2
      END IF
      read(10,*,end=2) x(i),Y(i)
      GOTO 1
    2 continue
      Ntot=i-1

      CALL CUBS3(X,Y,SY,Ntot)

*********************************************************************
*     Now data was read and spline's parameter setting was done
*********************************************************************

       do r=1.d-2,20.d0,1.d-2
P=poten(r)
write(2,200) r,P
       end do

*********************************************************************
*     Now the potential was written in file 'poten.dat' for checking
*     purposes
*********************************************************************

      do i=1,20
       call NUMEROV(E)

*********************************************************************
*     The subroutine NUMEROV finds energy of the bound states and
*     on exit containes the energy.
*********************************************************************

      E=E*2.d00*rm
      call WAVE(E)
      end do
```

```
*********************************************************************
*     The subroutine wave prints out wave function for given energy
*     in file 'wave.dat'
*********************************************************************

  100 format (F3.1,x,F10.5)
  200 format (2(F20.12,x))
      END
*********************************************************************
*     This is the molecular potential in Rydbergs
*********************************************************************

      double precision Function Poten(r)
      implicit real*8 (a-h,o-z)
      PARAMETER (NPM=10000)
      DIMENSION X(NPM),Y(NPM)
      DIMENSION sY(4,NPM)
      common/int_par/SY,X,Y,Ntot
      common /mass/Rm
      IF(r.ge.x(Ntot-2)) then
       Poten=0.d00
       return
      END IF

      gam=C3INT(sY,R,X,Ntot)
      Poten=2.d00*Rm*(gam + 16.d00/r)
      RETURN
      END
*********************************************************************
*     This subroutine prints out bound state wave function for given
*      energy E in file 'wave.dat'
*********************************************************************

      subroutine wave(e)
      implicit real*8 (a-h,o-z)
```

```
      external poten
      common /moment/L
      character*6 name
      dimension FF(2,500000),GG(2,50000)
      rmin=1.d-2
      step=1.d-3

      print *,'Enter a name of file'
      read *,name
      OPEN(14,File=name//'.dat')
      IF (e.ge.0.d0) then
       print*,' Energy GE 0'
       stop
      END IF

      cap=dsqrt(-E)
      call back(e,step,rb)
      rmax=20.d00/cap + rb
      rs=rb+ 5./cap
      al=step**2/12.d00
      psi0=Rmin**(l+1)

      IF(l.eq.0) THEN
       derpsi0=1.d00
                 ELSE
       derpsi0=(l+1)*Rmin**l
      END IF
*************************************************************************
*     Forward Run
*************************************************************************

      pmax=0.d0
      square=0.d0
      P0=psi0
      P1=P0+step*derpsi0
```

```
      f0=poten(Rmin)-E +l*(l+1)/rmin**2
      r1=rmin+step
      f1=poten(R1)-E +l*(l+1)/r1**2

      i=1

      do R=rmin+2.d00*step,rs,step
       f2=poten(R)-E +l*(l+1)/r**2
       a0=1.d00 - al*f0
       a1=2.d00+ 10.d00*al*f1
       a2=1.d00 - al*f2
       P2= (p1*a1-p0*a0)/a2
       f0=f1
       f1=f2
       p0=p1
       p1=p2
       FF(1,i)=r
       FF(2,i)=p2
       IF (dabs(p2).GE.pmax) pmax=dabs(p2)
i=i+1
      end do
      N1point=i-1

*********************************************************************
*     Backward Run
*********************************************************************

      gmax=    dexp(-cap*rmax)
      dergmax= - cap*gmax
      g2=gmax
      g1=g2-dergmax*step

      f2=poten(Rmax)-E +l*(l+1)/rmax**2
      r1=rmax-step
      f1=poten(R1)-E +l*(l+1)/r1**2
```

```
      i=1
      do r=rmax-2.d00*step,Rs,-step
       f0=poten(R)-E +l*(l+1)/r**2
       a2=1.d00 - al*f2
       a1=2.d00+ 10.d00*al*f1
       a0=1.d00 - al*f0
       G0= (G1*a1-G2*a2)/a0
       f2=f1
       f1=f0
       G2=G1
       G1=G0
       GG(1,i)=r
       GG(2,i)=g2
       i=i+1
      end do
      N2point=i-1

**********************************************************************
*     At this point we have two part of wave functions costracted
*     from zero to Rs and from Rmax to Rs. But, wave functions
*     themselves are not equal to each other at Rs.So we have to
*     multiplicate (for instance) Backward part for the nedeed factor.
**********************************************************************

      coef=GG(2,N2point)/FF(2,N1point)

      do i=1,N2point
       GG(2,i)=GG(2,i)/coef
      end do

      Ntot=N1point+N2point-1 !total number of points

      j=N2point+1
```

```
      do i=N1point+1,Ntot
        j=j-1
FF(2,i)=GG(2,j)
FF(1,i)=GG(1,j)
      end do

      do i=1,Ntot
       FF(2,i)=FF(2,i)/pmax
       square=square+FF(2,i)**2
      end do

      square=(square-( FF(2,1)**2+FF(2,Ntot)**2)/2.d00)*step
      square=dsqrt(square)

      do i=1,Ntot
       yx=FF(2,i)/square
       zx=yx
       IF (dabs(yx).lt.1.d-30) zx=0.d0
        write(14,200) FF(1,i),yx,zx
      end do

  200 format(3(1x,e20.10))

      return
      END
      implicit real*8 (a-h,o-z)
      PARAMETER (NPM=10000)
      DIMENSION X(NPM),Y(NPM)
      DIMENSION SY(4,NPM)
      common/int_par/SY,X,Y,Ntot
      common /Moment/L
      common /mass/Rm

      open (10,file='term.dat')
      open (14,file='wave.dat')
```

```
      Rm=931.4821d0*16.999133d0*1.00782522d0/
     /   0.5110041d0/(16.999133d0+1.0078252d0)


        i=0
    1   i=i+1
      IF(i.gt.npm) then
       print*," Data limits exceeded, you have to decrease lines= ",npm
       GOTO 2
      END IF
      read(10,*,end=2) x(i),Y(i)
      GOTO 1
    2 continue
      Ntot=i-1

      CALL CUBS3(X,Y,SY,Ntot)

**********************************************************************
*     Now data was read and spline's parameter setting was done
**********************************************************************

      L=0
      do E=1.d0,1000.d0,1.d0
       E1=Rm*E
       call wave(E1)
      end do
      END

**********************************************************************
*     This subroutine prints out scattering wave function for given
*      energy E in file 'scat.dat'
**********************************************************************

      subroutine wave(e)
```

```
      implicit real*8 (a-h,o-z)
      parameter (pi=3.1415926d0)
      external poten
      common /moment/L
      common /mass/Rm

      pimp=dsqrt(E)
      rmin=1.d-5
      step=5.d-6
      Rmax=50.d0
      al=step**2/12.d00
      psi0=Rmin**(l+1)

      IF(l.eq.0) THEN
       derpsi0=1.d00
                 ELSE
       derpsi0=(l+1)*Rmin**l
      END IF

*********************************************************************
*     Forward Run
*********************************************************************

      pmax=0.d0
      P0=psi0
      P1=P0+step*derpsi0
      f0=poten(Rmin)-E +l*(l+1)/rmin**2
      r1=rmin+step
      f1=poten(R1)-E +l*(l+1)/r1**2

      i=1
      r=rmin
      f2=poten(r)-E
      a0=1.d0-al*f0
      a1=2.d0+10.d0*al*f1
```

```
      a2=1.d0-al*f2
      y0=(p1*a1-p0*a0)/a2  !this is wave function at r=rmin

      do R=rmin+2.d00*step,Rmax,step
       f2=poten(R)-E +l*(l+1)/r**2
       a0=1.d00 - al*f0
       a1=2.d00+ 10.d00*al*f1
       a2=1.d00 - al*f2
       P2= (p1*a1-p0*a0)/a2
       f0=f1
       f1=f2
       p0=p1
       p1=p2
       IF (dabs(p2).GE.pmax.and.r.gt.5.d00) pmax=dabs(p2)
      end do

      y0=derpsi0/pmax

      factor1=dsqrt(2*pi*Rm*8*pimp)
      factor2=pi*Rm*8/pimp/2
      psiCol=dsqrt(factor1)*dexp(-factor2)
      ratio=psiCol/dsqrt(y0)
      ratio=ratio**2

      IF (pimp*rmin.GT.1.d-2) print *,'too much energy...'


      write (14,100) E,y0,psiCol,ratio
      print *,E,y0,psiCol,ratio


  100 format (4(E20.12,1x))
  200 format(3(1x,e20.10))

      return
```

```
END
```

Литература

[1] Ramaty R., Lingefelter R.E. Astrophysical Gamma Ray Emission Lines, Space Telecom. Sci. Instrum. Serie B, Cambrige Press

[2] Базь А.И., Зельдович Я.Б., Переломов А.Н. Рассеяние, реакции и распады в нерелятивистской квантовой механике, изд. "Наука Москва, 1971

[3] V.B. Belyaev, A.K. Motovilov, W.Sandhas On the possibility of fusion reactions in water molecules, JINR Rapid Commun. N6 [74] -95

[4] V.B. Belyaev, A.K. Motovilov, W.Sandhas Fusion reactions in molecules via nuclear threshold resonances, J. Phys. G. Nucl. Part. Phys. 22 (1996) pp.1111-1113

[5] Пеньков Ф.М. Ядерные Переходы из Молекулярных Резонансов, Препринт ОИЯИ, P4-96-267, 1996

[6] H. Picker The exited states in question are vibrational exited states in diatomic molecule (HD), Nukleonika, vol. 25 (1980), pp. 1491-1493

[7] Фудзинага С. Метод Молекулярных Орбиталей, Москва, "Мир 1983

[8] Слэтер Дж. Электронная Структура Молекул ,Изд. "Мир Москва, 1965

[9] Ландау Л.Д., Лифшиц Е.М. т.III Квантовая Механика (нерелятивистская теория), Москва, "Наука 1989

[10] Хартри Д. Расчеты Атомных Структур, Изд. Иностр. Лит., Москва,1960

[11] Roothaan C.C.J. Self Consistent Field Theory for Open Shells of Electronic Systems, Rev.Mod.Phys.,V.32,No 2, pp.179-185, April, 1960

[12] Жоголев Д.А., Волков В.Б. Методы ,алгоритмы и программы для квантовомеханичеких расчетов молекул, Киев "Наукова Думка 1976

[13] Roothaan C.C.J. Revs.Modern.Phys.,23,69

[14] Amos A.T. and Hall G.G. Single determinant wave function, Proc. of the Royal Soc.,Ser. A, No 1315, 10 October,Vol.263,pp.483-493,1961

[15] Freeman A.J. Configuration Interaction Study of the Electonic Structure of the OH Radical by the Atomic and Molecular Orbital Methods, Jour.Chem.Phys., Vol.28,No 2, pp.230-243, Febraury,1958

[16] Karo Arnold M. and Allen Leland C., LCAO Wave Functions for Hydrogen Fluoride with Hartree - Fock Atomic Orbitals, Jour. Chem. Phys. Vol.31, No 4, pp.968-977, October, 1959

[17] Ransil Bernard J. Studies in Molecular Srtucture. I. Scope and Summary of the Diatomic Molecule Program. Rev.Mod.Phys.,Vol.32, No 2, pp.968-977, April, 1960

[18] Федоренко Р.П. Введение в вычислительную физику, Изд. Москов. физ.-техн. института, 1994

[19] Jackson J.D. Phys.Rev 1957, V.106 p.330

[20] М. Абрамовиц и С. Стеган, Справочник по спецфункциям, Москва, "Наука 1979

[21] Бете Г. и Солпитер Э. Квантовая механика атомов с одним и двумя электронами, Гос. Изд. Физ. Мат. Лит.,Москва, 1960

[22] Bingel Werner A. United Atom Treatment of the behavior of Potential Energy Curves of Diatomic Molecules for Small R, pp.1250-1253, Jour.Chem.Phys., Vol.30,No 5 ,May, 1959

[23] Веселов М.Г., Лабзовский Л.Н. Теория Атома. Строение Электронных Оболочек, Москва, "Наука 1986

[24] Демидович Б.П. Марон И.А. Основы вычислительной математики, Москва, 1970

[25] Собельман И.И. Введение в теорию атомных спектров, Гос. Изд. физ. мат. Лит.,Москва, 1963