

INTEGRATION BY CELL ALGORITHM FOR SLATER INTEGRALS IN A
SPLINE BASIS

By

Yanghui Qiu

Thesis

Submitted to the Faculty of the
Graduate School of Vanderbilt University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

in

Computer Science

August 1999

Nashville, Tennessee

Approved:

Date:

ACKNOWLEDGMENTS

I thank my advisor, Prof. Charlotte Froese Fischer, for her invaluable advice and guidance on my research, in both computational science and theoretical atomic physics, and for giving me the opportunity to exploit the fields of computational science and computer science.

I thank Prof. Fitzpatrick for his time in reading my thesis. I would like to express my appreciation to other faculty, staff, and graduate students in the Computer Science Department. I appreciate their help during this degree program. I particularly thank Mr. Andy Richter for his invaluable help in many computer related issues.

This work was supported by the Division of Chemical Sciences, Office of Basic Energy Sciences, Office of Energy Research, U.S. Department of Energy.

TABLE OF CONTENTS

	Page
ACKNOWLEDGMENTS	ii
LIST OF FIGURES	iv
LIST OF TABLES	v
 Chapter	
I. INTRODUCTION	1
II. SLATER INTEGRALS IN B-SPLINES	4
Definition of B-splines	4
Optimized general grid for atomic systems	5
Symmetry of the Slater matrix elements	6
Exchange symmetry of the Slater matrix elements	7
Scaling laws of the Slater matrix elements	7
III. INTEGRATION BY CELL ALGORITHM	9
Integration over the off-diagonal cells	10
Integration over the diagonal cells	11
Assembly of the cell integrals	13
An alternative to the diagonal-cell integration	14
IV. RESULTS AND DISCUSSIONS	17
V. CONCLUSION	26
 Appendix	
A. DERIVATION OF SCALING LAWS	27
B. CODE FOR EVALUATING THE CELL INTEGRALS AND ASSEMBLING SLATER MATRIX ELEMENTS	30
BIBLIOGRAPHY	51

LIST OF FIGURES

Figure	Page
2.1. The set of B-splines of order 5 in the region $[0, 1]$ on the knot sequence $t_1 = t_2 = \dots = t_5 = 0, t_i = t_{i-1} + 0.1$ for $i = 6, 7, \dots, 15$ and $t_{15} = t_{16} = \dots = t_{19} = 1$	5
3.1. Illustration of the area over which the integrand of the Slater matrix element $R^k(i, j; i', j')$ is non trivial. The area is a block of cells stretching from interval $i - k_s + 1$ to i' in the r_1 direction and from interval $j - k_s + 1$ to j' in the r_2 direction.	10
3.2. Graphical representation of the Gaussian points for $k_s = 8$. The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current two-dimensional cell integration.	14
3.3. Graphical representation of the Gaussian points for $k_s = 8$. The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current two-dimensional cell integration.	16
4.1. Flowchart of the Slater integral package.	18

LIST OF TABLES

Table		Page
4.1.	User and System time in seconds for setting up all of the Slater matrix elements for $k_s = 4, 5, 6, 8$ on a Sun workstation (CPU: UltraSPARC 143MHz; RAM: 64MB). k : the order of the Slater integrals; k_s : the order of the B-splines; h : the starting step size of the grid; t_1 : the time with the traditional method; t_2 : the time with the integration by cell method.	20
4.2.	Comparison of the accuracy of some F^k and G^k integrals for different order of B-splines k_s . The exact values of the Slater integrals are from Ref. [9]. Difference 1: the difference of the exact value and the one evaluated with the traditional method; Difference 2: the difference of the exact value and one evaluated with the integration by cell method. t_3 : the user and system time of assembling the Slater integral from the Slater matrix elements.	21

CHAPTER I

INTRODUCTION

Critical to the success of the many-electron atomic structure calculation is an accurate and efficient evaluation of the Slater integrals. The Slater integrals are integrations of a product of four atomic radial wavefunctions and a coupling weighting factor over the two dimensional configuration space, i.e.,

$$R^k(a, b; c, d) = \int_0^\infty \int_0^\infty \frac{r_{<}^k}{r_{>}^{k+1}} P_a(r_1) P_b(r_2) P_c(r_1) P_d(r_2) dr_1 dr_2 \quad (1.1)$$

where P_a, P_b, P_c , and P_d are the radial wavefunctions, k is an integer, and $r_{<}, r_{>}$ are the $\min(r_1, r_2)$ and $\max(r_1, r_2)$, respectively. The radial function of orbital a is approximated by

$$P_a(r) \approx \sum_i a_i B_i(r), \quad (1.2)$$

where $B_i(r)$ is the basis function and a_i the expansion coefficient.

Splines as basis functions in many-electron atomic systems were introduced over twenty years ago when Altenberger-Siczek and Gilbert [1] investigated the two-electron helium atom. Under the spline basis, the Slater integral becomes

$$R^k(a, b; c, d) = \sum_i \sum_j \sum_{i'} \sum_{j'} a_i b_j c_{i'} d_{j'} R^k(i, j; i', j'), \quad (1.3)$$

where the Slater matrix element $R^k(i, j; i', j')$ is

$$R^k(i, j; i', j') = \int_0^\infty \int_0^\infty \frac{r_{<}^k}{r_{>}^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2. \quad (1.4)$$

Because of the large number of numerical operations needed to be evaluated accurately, they finally concluded that B-splines were not suitable for atomic structure calculations.

However, there has recently been a renewed interest in the use of B-splines in complex atomic systems: Bottcher and Strayer [2] applied the B-splines for time-dependent problems, Johnson and co-workers for many-body perturbation theory [3, 4, 5], Fischer and co-workers [6, 7, 8, 9] for Hartree-Fock calculations and continuum problems, Chang and co-worker [10] for continuum problems, Hansen and co-workers for orthogonal operators and Rydberg series [11, 12], Decleva and co-workers [13, 14, 15] for multichannel continuum problems, and van der Hart for R-matrix theory [16]. More detailed discussions can be found in recent review by Sapirstein and Johnson [17].

One of the important developments was finding a method for evaluating the Slater matrix elements [8, 9]. In this approach, the Slater matrix elements $R^k(i, j; i', j')$ are rewritten as

$$R^k(i, j; i', j') = \int_0^\infty \frac{1}{r_1} B_i^{k_s}(r_1) Y_{jj'}^k(r_1) B_{i'}^{k_s}(r_1) dr_1 \quad (1.5)$$

where

$$Y_{jj'}^k(r_1) = r_1 \int_0^\infty B_j^{k_s}(r_2) \frac{r_2^k}{r_2^{k+1}} B_{j'}^{k_s}(r_2) dr_2. \quad (1.6)$$

The function $Y_{jj'}^k$ at a predefined point r is computed by solving the following differential equations for the given boundary conditions

$$\begin{aligned} \frac{d^2 Y_{jj'}^k(r)}{dr^2} &= \frac{k(k+1)}{r^2} Y_{jj'}^k(r) - \frac{2k+1}{r} B_j^{k_s}(r) B_{j'}^{k_s}(r) \\ Y_{jj'}^k(0) &= 0 \\ \frac{d}{dr} Y_{jj'}^k(r) &= -\frac{k}{r} Y_{jj'}^k(r) + B_j^{k_s}(r) B_{j'}^{k_s}(r) \quad \text{as } r \rightarrow +\infty \end{aligned} \quad (1.7)$$

and the Slater matrix element in Eq. (1.5) is obtained by a direct integration. The Slater integrals as a quadruple summation of the matrix elements are thus assembled.

The method of evaluating the Slater matrix elements by solving the differential equations was introduced by Hartree [18] when he was doing self-consistent calculations of atomic structure and was successful for general applications. However, when it is used in spline-based applications, the special characteristics of the B-splines are not considered and exploited. In this thesis, we have introduced and implemented an algorithm which is designed specifically for the spline-based applications. This method takes advantage of the piece-wise property and scaling invariance of the B-splines and implements the integration in Eq. (1.4) by cells. Since the system and user time spent on the evaluation of the Slater matrix elements is much more significant than that spent on assembling the Slater integrals from the matrix elements, significant improvements in both efficiency and accuracy in evaluating the Slater integrals are observed under the current algorithm.

CHAPTER II

SLATER INTEGRALS IN B-SPLINES

Definition of B-splines

Following de Boor [19], we divide the interval $[0, R]$ into segments. The endpoints of these segments are given by a knot sequence t_i , $i = 1, 2, \dots, n + k_s$. The B-splines of order k_s , $B_i^{k_s}(r)$, are a set of piecewise polynomials defined on the knot sequence recursively by the relations

$$B_i^1(r) = \begin{cases} 1, & t_i \leq r < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \quad (2.1)$$

and

$$B_i^{k_s}(r) = \frac{r - t_i}{t_{i+k_s-1} - t_i} B_i^{k_s-1}(r) + \frac{t_{i+k_s} - r}{t_{i+k_s} - t_{i+1}} B_{i+1}^{k_s-1}(r). \quad (2.2)$$

The function $B_i^{k_s}(r)$ is a piecewise polynomial of degree $k_s - 1$ inside the interval $t_i \leq r < t_{i+k_s}$ which vanishes outside this interval. The sum at any point r of all of the B-splines that do not vanish at the point is unity [20], i.e.,

$$\sum_i B_i^{k_s}(r) = 1. \quad (2.3)$$

The set of B-splines of order k_s on the knot sequence t_i forms a complete basis for piecewise polynomials of degree $k_s - 1$ on the interval spanned by the knot sequence. The knots defining our grid have k_s -fold multiplicity at the endpoints 0 and R , i. e. $t_1 = t_2 = \dots = t_{k_s} = 0$ and $t_{n+1} = t_{n+2} = \dots = t_{n+k_s} = R$. When multiple knots are encountered, limiting forms of the above recursive definition for the B-splines must be used. For $k_s > 1$, the B-splines generally vanish at their endpoints. However, at $r = 0$

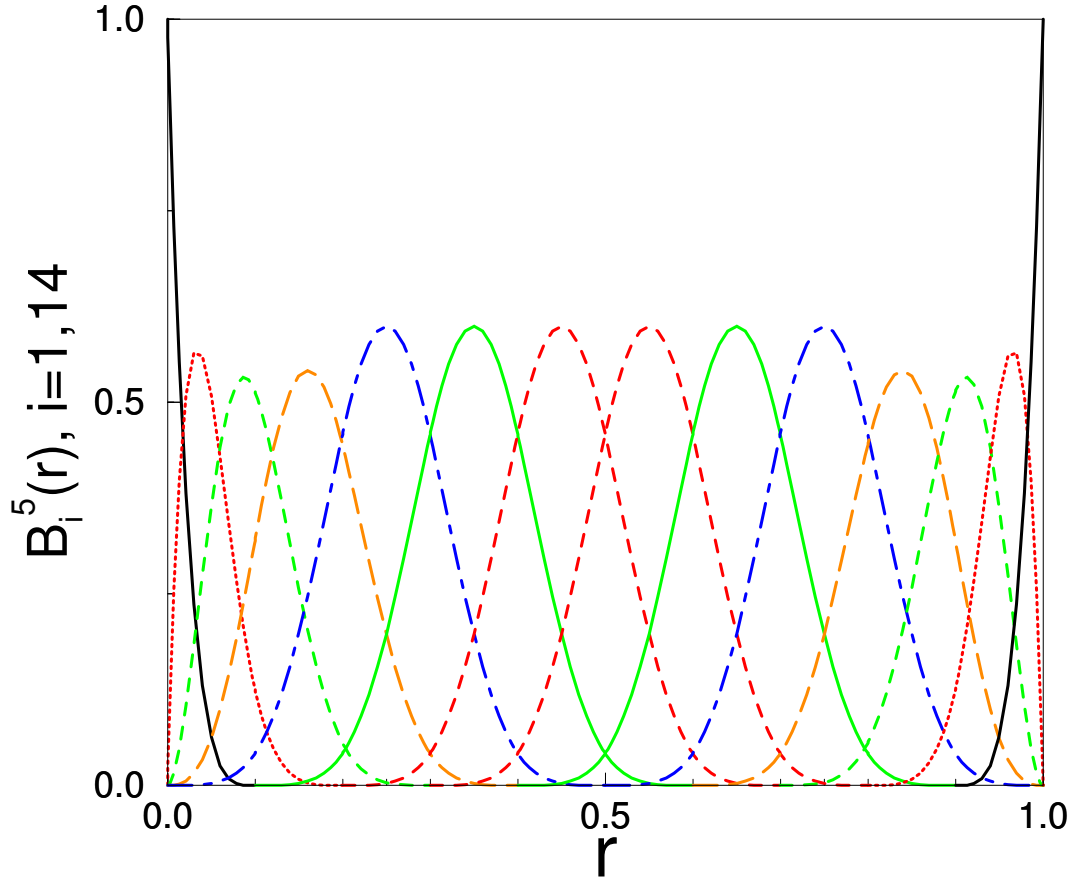


Figure 2.1: The set of B-splines of order 5 in the region $[0, 1]$ on the knot sequence $t_1 = t_2 = \dots = t_5 = 0, t_i = t_{i-1} + 0.1$ for $i = 6, 7, \dots, 15$ and $t_{15} = t_{16} = \dots = t_{19} = 1$.

the first B-spline is equal to 1 with all others vanishing and at $r = R$ the last B-spline shows the same behavior as the first B-spline at $r = 0$. The two end B-splines are generally related to the boundary conditions of the problem under investigation [21].

A set of B-splines of order 5 in the region $[0, 1]$ is shown in Fig. (2.1).

Optimized general grid for atomic systems

One of the advantages in spline basis is that the choice of grid may be tailored to the problem under investigation. A different type of grid can easily be used in different regions of the configuration space. In atomic physics calculation, an optimized general

grid for both bound and continuum states was introduced by Fischer [7, 21] and was subsequently applied successfully in the calculation of photoionization [13, 22]. In the grid definition, four different parameters are introduced. They are the step size $h = 2^{-m}$ with m an integer, the maximum step size h_{max} , the maximum range R , and the order of splines k_s . The grid points are defined through the array t_i , such that

$$\begin{aligned}
 t_i &= 0, & \text{for } i = 1, \dots, k_s \\
 t_{i+1} &= t_i + h, & \text{for } i = k_s, \dots, k_s + m \\
 t_{i+1} &= t_i(1 + h), & \text{for } 1 \leq t_{i+1} - t_i < h_{max} \\
 t_{i+1} &= t_i + h_{max}, & \text{for } t_i < ZR \\
 r_i &= t_i/Z, & \text{for any } i
 \end{aligned}$$

where r is the radial coordinate and Z is the nuclear charge. The rules of setting such a knot sequence can be found in Ref. [21]. Briefly speaking, since the Hamiltonians of atomic systems and the corresponding orbital wavefunctions scale approximately with respect to the nuclear charge Z , the knot sequence is set as $t = Zr$. For the continuum calculation, the logarithmic grid is shown to be a good choice for small r , except near the origin, where a constant grid can avoid the problem of singularity. Meanwhile, the logarithmic grid results in a very large step size for large r , which can be larger than the wavelength of the oscillatory continuum wavefunctions, therefore a constant grid in the large r region is used.

Symmetry of the Slater matrix elements

The Slater matrix elements have two symmetries, i.e., the matrix elements are invariant under index exchange and are scaling invariant under coordinate displacement within the exponential region.

Exchange symmetry of the Slater matrix elements

From the definition of the Slater matrix elements, it is trivial to show that the exchange indices of i and i' , j and j' , and $\{i, i'\}$ and $\{j, j'\}$ does not affect the result of the Slater matrix element, i.e.,

$$\begin{aligned}
 R^k(i, j; i', j') &= R^k(i', j; i, j') \\
 &= R^k(i, j'; i', j) \\
 &= R^k(j, i; j', i') \\
 &= R^k(j', i; j, i') \\
 &= R^k(j, i'; j', i) \\
 &= R^k(j', i'; j, i) \\
 &= R^k(i', j'; i, j)
 \end{aligned}$$

Therefore, only about 1/8 of the Slater matrix elements need to be evaluated.

Scaling laws of the Slater matrix elements

The grid with an exponentially increasing interval length results in several properties that can be exploited to avoid redundant calculations. Suppose the splines discussed lie entirely within the exponential region. Since the splines are normalized such that the sum of the values at any point in the range $[0, R]$ is equal to unity, a simple displacement invariance applies. Let the left-most knot defining $B_i^{k_s}(r)$ be t_i and let $r = t_i + s$, then,

$$B_i^{k_s}(t_i + s) = B_{i+1}^{k_s}((1 + h)(t_i + s)) = B_{i+1}^{k_s}(t_{i+1} + s(1 + h)). \quad (2.4)$$

From the displacement invariance of the splines, several scaling laws exist [23]. The ones relevant to the Slater matrix elements are as follows:

$$\langle B_{i+1}^{k_s}(r) | r^k | B_{j+1}^{k_s}(r) \rangle = (1+h)^{1+k} \langle B_i^{k_s}(r) | r^k | B_j^{k_s}(r) \rangle \quad (2.5)$$

$$R^k(i+1, j+1; i'+1, j'+1) = (1+h)R^k(i, j; i', j'). \quad (2.6)$$

However, the most valuable properties to the current integration by cell method is the scaling laws over individual cells, i.e.,

$$\int_{r_{iv+1}}^{r_{iv+2}} B_{i+1}^{k_s}(r) B_{j+1}^{k_s}(r) r^k dr = (1+h)^{1+k} \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r) B_j^{k_s}(r) r^k dr \quad (2.7)$$

and

$$\begin{aligned} & \int_{r_{iv+1}}^{r_{iv+2}} \int_{r_{jv+1}}^{r_{jv+2}} \frac{r_{>}^k}{r_{<}^{k+1}} B_{i+1}^{k_s}(r_1) B_{j+1}^{k_s}(r_2) B_{i'+1}^{k_s}(r_1) B_{j'+1}^{k_s}(r_2) dr_1 dr_2 \\ &= (1+h) \int_{r_{iv}}^{r_{iv+1}} \int_{r_{jv}}^{r_{jv+1}} \frac{r_{>}^k}{r_{<}^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2 \end{aligned} \quad (2.8)$$

A detailed proof of the scaling laws over a cell is given in Appendix A.

CHAPTER III

INTEGRATION BY CELL ALGORITHM

The distinguishing feature of the B-splines is their piecewise property. Every spline is a positive definite polynomial and stretches over a finite range in the configuration space. If k_s order B-splines are chosen, the spline $B_i^{k_s}$ is non trivial only in the range from knot i to knot $i + k_s$, i.e., the spline $B_i^{k_s}$ is non zero only in intervals $\text{MAX}(1, i - k_s + 1)$, $\text{MAX}(1, i - k_s + 1) + 1$, ..., $\text{MIN}(i, nv)$, where nv is the total number of intervals. This feature leads to an important constraint such that at any interval iv in the one-dimensional configuration space, the number of splines whose values are not zero is always k_s . In fact these non-zero ones are the spline $iv, iv + 1, \dots$, and $iv + k_s - 1$. Other splines at interval iv simply vanish. When this property is applied to the Slater matrix elements

$$R^k(i, j; i', j') = \int_0^\infty \int_0^\infty \frac{r_1^k}{r_1^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2, \quad (3.1)$$

$R^k(i, j; i', j') = 0$, if either $|i' - i| \geq k_s$ or $|j' - j| \geq k_s$. Moreover, the integration contributing to $R^k(i, j; i', j')$ extends over only the cells in which $B_i^{k_s}(r_1)$ and $B_{i'}^{k_s}(r_1)$, and $B_j^{k_s}(r_2)$ and $B_{j'}^{k_s}(r_2)$ overlap. Because of the symmetry of the Slater matrix, we can generally assume that $i \leq i', j \leq j'$, and $i \leq j$. The area over which the integrand contributes to the Slater matrix element $R^k(i, j; i', j')$ is illustrated in Fig. 3.1. The area is a block of cells stretching from interval $i - k_s + 1$ to i' in the r_1 direction and from interval $j - k_s + 1$ to j' in the r_2 direction. The integration by cell algorithm exploits this feature thoroughly. Since there are only k_s splines which are not zero

along the r_1 or r_2 direction in each cell, we can integrate all the non-trivial $\{i, i', j, j'\}$ combinations over each individual cell first. The Slater matrix elements are then obtained as a summation of the cell integrals.

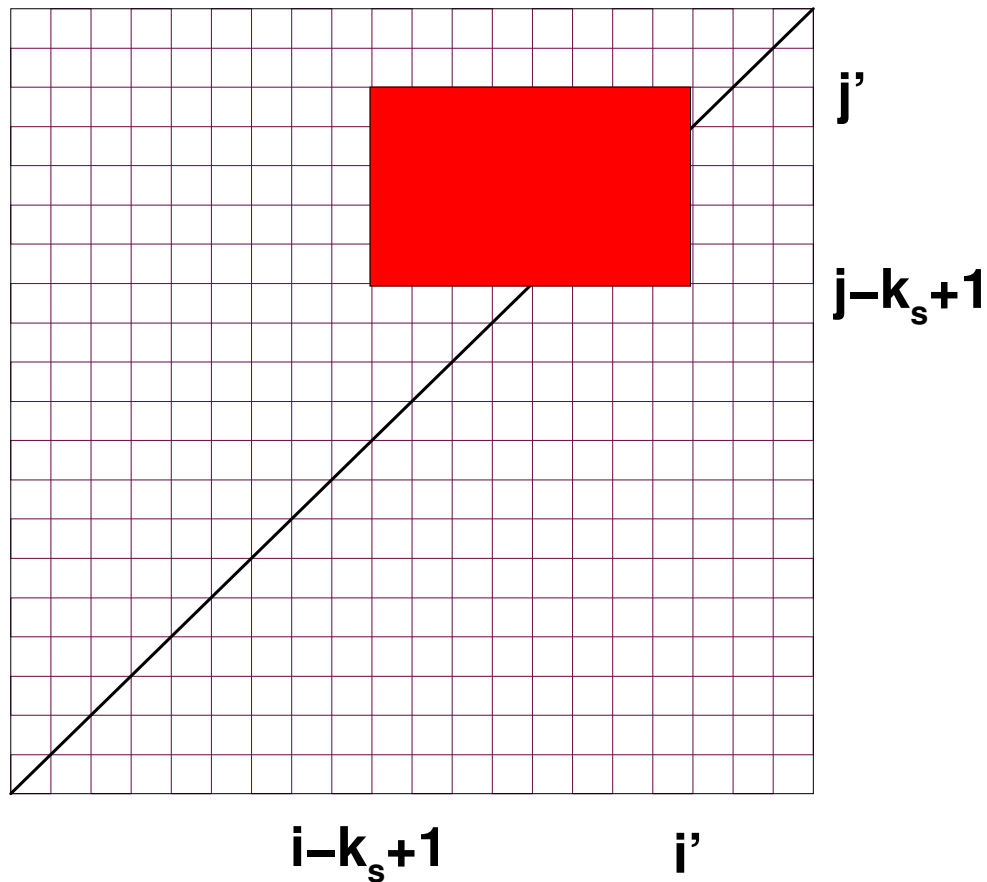


Figure 3.1: Illustration of the area over which the integrand of the Slater matrix element $R^k(i, j; i', j')$ is non trivial. The area is a block of cells stretching from interval $i - k_s + 1$ to i' in the r_1 direction and from interval $j - k_s + 1$ to j' in the r_2 direction.

Integration over the off-diagonal cells

Over the off-diagonal cells the integrand in the Slater matrix element is separable and the integration limits are not coupled. Therefore, the two-dimensional integral is

reduced to a product of two one-dimensional integrals. Assume $iv < jv$, then,

$$\begin{aligned}
R^k(i, j; i', j'; iv, jv) &= \int_{r_{jv}}^{r_{jv+1}} \int_{r_{iv}}^{r_{iv+1}} \frac{r_{>}^k}{r_{>}^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2 \\
&= \int_{r_{iv}}^{r_{iv+1}} r_1^k B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1 \int_{r_{jv}}^{r_{jv+1}} \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \\
&= r^k(i, i'; iv) \times r^{-(k+1)}(j, j'; jv),
\end{aligned} \tag{3.2}$$

where

$$r^k(i, i'; iv) = \int_{r_{iv}}^{r_{iv+1}} r_1^k B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1, \tag{3.3}$$

$$r^{-(k+1)}(j, j'; jv) = \int_{r_{jv}}^{r_{jv+1}} \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2. \tag{3.4}$$

Although there are $n_v(n_v - 1)$ off-diagonal cells, we only need to calculate and store the n_v one-dimensional moment arrays $r^k(i, i'; iv)$ and $r^{-(k+1)}(j, j'; jv)$ for later assembly. The arrays of $r^k(i, i'; iv)$ and $r^{-(k+1)}(i, i'; iv)$ with $i, i' = 1, \dots, k_s$ and $iv = 1, \dots, n_v$ can be evaluated effectively using Gaussian quadrature with k_s Gaussian points. Note that

$$r^k(i, i'; iv) = r^k(i', i; iv) \tag{3.5}$$

$$r^{-(k+1)}(i, i'; iv) = r^{-(k+1)}(i', i; iv), \tag{3.6}$$

only $r^k(i, i'; iv)$ and $r^{-(k+1)}(i, i'; jv)$ such that $i < i'$ need to be evaluated. Moreover, the evaluation can be further reduced by using the scaling law in Eq. (2.7) in the logarithmic grid region. The storage of $r^k(i, i'; iv)$ and $r^{-(k+1)}(i, i'; jv)$ ($i < i'$) takes $n_v k_s^2$ locations in memory.

Integration over the diagonal cells

Over the diagonal cells the two coordinates in the integrand (the integration limits) of the Slater matrix elements are coupled. Moreover, the integrand is not continuous

across the cell diagonal $r_1 = r_2$. Although there are only n_v cells, the diagonal cell integration is the most CPU intensive part in the evaluation of the Slater matrix element and it is also the major barrier towards achieving highly accurate matrix elements.

We separate the rectangular cell into two triangles so that the integrand inside each triangle is continuous. The integration over the rectangular cell is then a summation of integrations over the two triangles, which are symmetric with respect to index exchanges, i.e.,

$$\begin{aligned}
R^k(i, j; i', j'; iv) &= \int_{r_{iv}}^{r_{iv+1}} \int_{r_{iv}}^{r_{iv+1}} \frac{r_1^k}{r_1^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2 \\
&= \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1 \cdot \left\{ \frac{1}{r_1^{k+1}} \int_{r_{iv}}^{r_1} r_2^k B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \right. \\
&\quad \left. + r_1^k \int_{r_1}^{r_{iv+1}} \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \right\} \\
&= R_{\Delta}^k(i, j, i', j'; iv) + R_{\Delta}^k(j, i, j', i'; iv), \tag{3.7}
\end{aligned}$$

where

$$R_{\Delta}^k(i, j, i', j'; iv) = \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1 \frac{1}{r_1^{k+1}} \int_{r_{iv}}^{r_1} r_2^k B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2. \tag{3.8}$$

We use Gaussian quadrature again to do the integration over the triangular cell in Eq. (3.8). The key for an effective evaluation is to choose optimized Gaussian grid points such that the available B-spline values used in the one dimensional integration can be exploited and the new B-spline values needed to achieve the required accuracy are minimized. We know that n point Gaussian integration has $2n - 1$ degrees of accuracy. It is obvious that at least a k_s by k_s Gaussian grid with respect to both coordinates is needed to achieve the highest algebraic accuracy for the integrals in Eq. (3.8) for $k = 0$. Therefore, we apply the original Gaussian points used in

the evaluation of one-dimensional integrals, $r^k(i, i'; iv)$ and $r^{-(k+1)}(j, j'; jv)$, to the two-dimensional integration with respect to coordinate r_1 . In the two-dimensional integration regarding to coordinate r_2 , we also use k_s Gaussian points. A graphical representation of the chosen Gaussian points for $k_s = 8$ is shown in Fig. (3.2). The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current triangular cell integration. With this two dimensional grid, the difficulty of the cell integrals near the origin because of the singularity of the integrand is minimized and uniformly accurate results for all the Slater integrals can be obtained. The evaluation can be further reduced by using the scaling law in Eq. (2.8) in the logarithmic grid region. The storage of $R_{\Delta}^k(i, j, i', j'; iv)$ ($i < i', j < j', i < j$) takes $n_v k_s^4$ locations in memory. Practical calculation demonstrates the effectiveness of current choice of grid points (see later section).

Assembly of the cell integrals

Once all the cell integrals $r^k(i, i'; iv)$, $r^{-(k+1)}(j, j'; jv)$, and $R_{\Delta}^k(i, j, i', j'; iv)$ are prepared, the Slater matrix elements in Eq. (3.1) can be assembled straightforwardly. A FORTRAN 90 code for evaluating the cell integrals and assembling the Slater matrix elements from the cell integrals is attached in Appendix B. The effect of putting the pieces together is almost trivial for a computer. Since the Slater matrix elements need more space to store than the cell integrals, it is more efficient to store the cell integrals directly. When need for the Slater matrix elements arises, we can assemble it immediately.

Once all of the Slater matrix elements are assembled, the value of any Slater integral in Eq. (1.3) can be easily obtained.

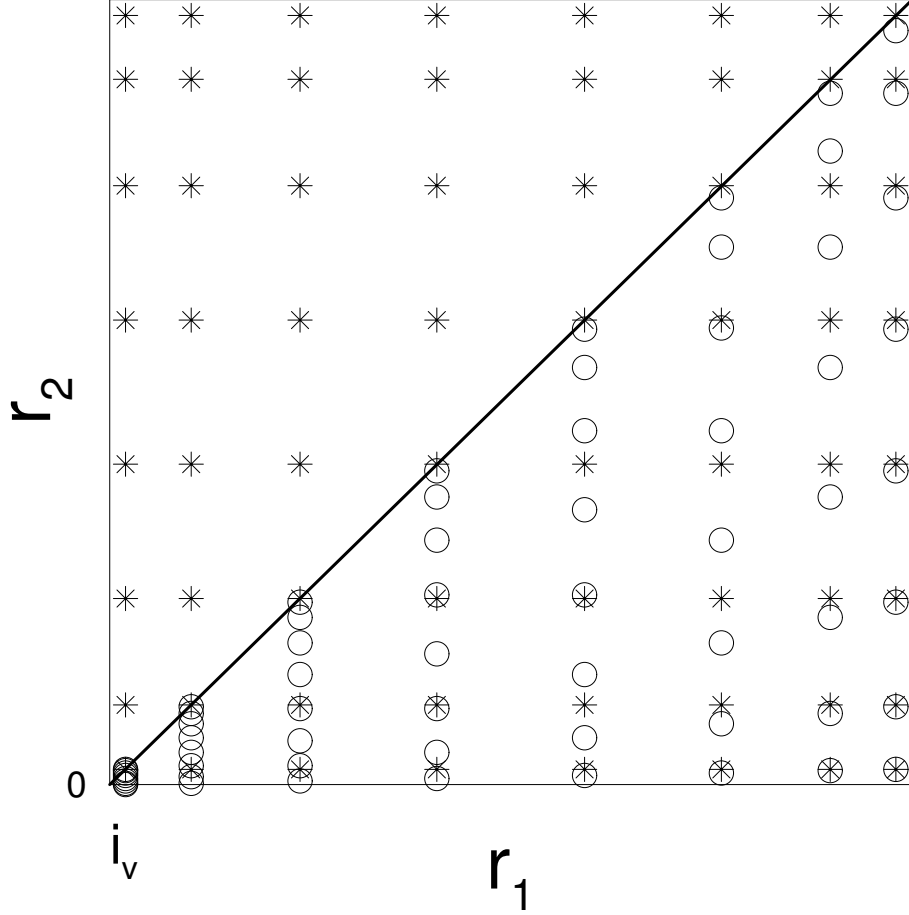


Figure 3.2: Graphical representation of the Gaussian points for $k_s = 8$. The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current two-dimensional cell integration.

An alternative to the diagonal-cell integration

The integration over a diagonal cell in Eq. (3.7) can be implemented in an alternative way. The cell integral can be rewritten as

$$\begin{aligned}
R^k(i, j; i', j'; iv) &= \int_{r_{iv}}^{r_{iv+1}} dr_1 r_1^k B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) \int_{r_{iv}}^{r_{iv+1}} dr_2 \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) \\
&+ \int_{r_{iv}}^{r_{iv+1}} dr_1 B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) \int_{r_{iv}}^{r_1} dr_2 B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) \left\{ \frac{r_2^k}{r_1^{k+1}} - \frac{r_1^k}{r_2^{k+1}} \right\}
\end{aligned} \tag{3.9}$$

where the first term in Eq. (3.9) is a product of two one-dimensional integrals which are evaluated during the off-diagonal cell integration in Eq. (3.2) and the second term is a coupled two-dimensional integral over a triangular cell. The second term has a nice property such that the integrand is zero along the hypotenuse of the cell and becomes most significant at the low-right corner of the cell (Fig. (3.3)). To exploit this property, we make a coordinate rotation

$$\begin{aligned} x &= \frac{r_1 + r_2}{\sqrt{2}} \\ y &= \frac{r_1 - r_2}{\sqrt{2}} \end{aligned} \quad (3.10)$$

where the new coordinates x and y are always positive in the integration area. The second term in Eq. (3.9) is then transformed as

$$\begin{aligned} R^k(i, j; i', j'; iv)|_{term2} &= \sqrt{2} \int_0^{\frac{r_{iv+1}-r_{iv}}{\sqrt{2}}} dy \int_{\sqrt{2}r_{iv}+y}^{\sqrt{2}r_{iv+1}-y} dx B_i^{k_s}\left(\frac{x+y}{\sqrt{2}}\right) B_{i'}^{k_s}\left(\frac{x+y}{\sqrt{2}}\right) \\ &\cdot B_j^{k_s}\left(\frac{x-y}{\sqrt{2}}\right) B_{j'}^{k_s}\left(\frac{x-y}{\sqrt{2}}\right) \left\{ \frac{(x-y)^k}{(x+y)^{k+1}} - \frac{(x+y)^k}{(x-y)^{k+1}} \right\}. \end{aligned} \quad (3.11)$$

We use Gaussian quadrature to do the integration over the triangular cell. An unevenly distributed Gaussian grid points can be used to adapt the characteristics of the integrand. In Fig. (3.3) a graphical representation of the adapted optimal Gaussian points is shown. The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current triangular cell integration. With this two dimensional grid, greater weight of integration is located at the low-right corner of the triangular cell where the integrand is most significant. This approach however has drawbacks. It does not pay special attention to the singularity of the integrand at the origin. Moreover, the B-spline itself as a function of one variable r_1 or r_2 becomes a function of two variables x and y , which leads to more computational effect during the integration.

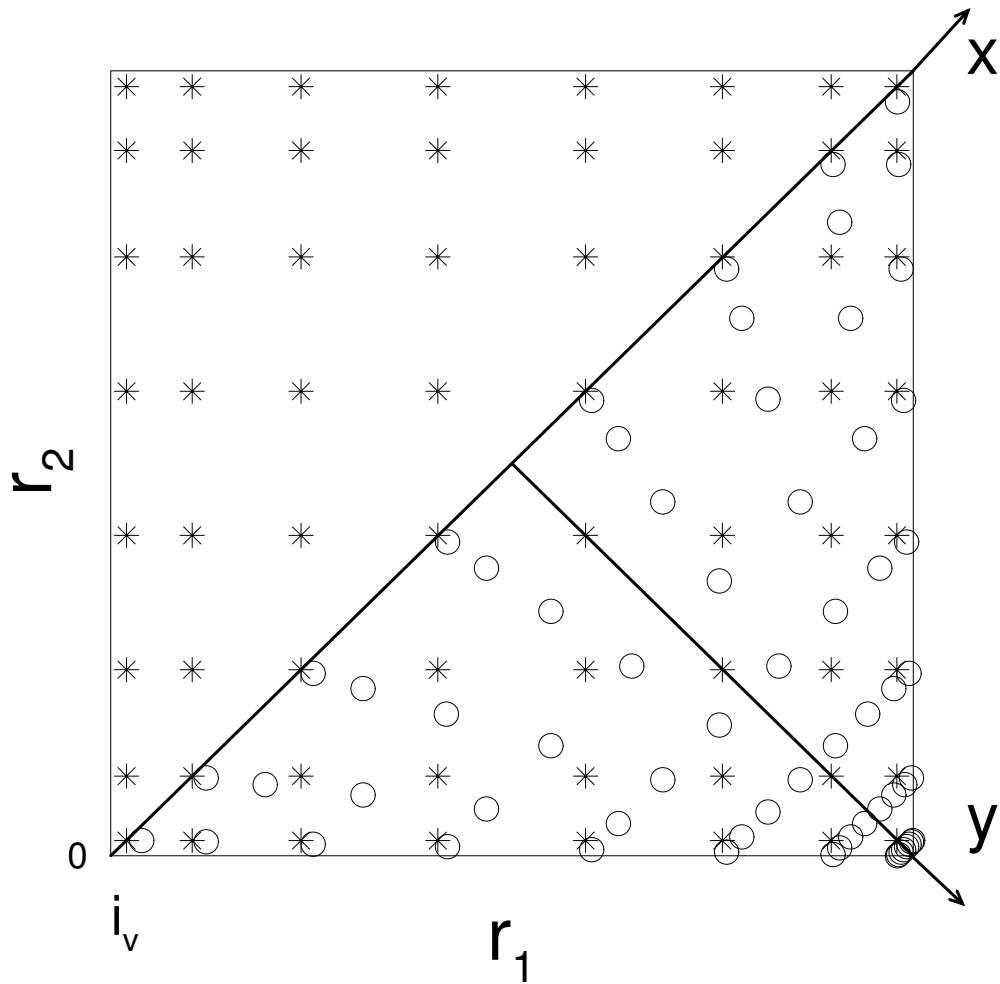


Figure 3.3: Graphical representation of the Gaussian points for $k_s = 8$. The star is the original Gaussian point used in the one-dimensional integration and the circle is the Gaussian point used in current two-dimensional cell integration.

CHAPTER IV

RESULTS AND DISCUSSIONS

We have implemented the two variants of the integration-by-cell algorithm for the Slater matrix elements in FORTRAN 90 and assembled the Slater integrals thereafter. The traditional algorithm based on solving the differential equation is also implemented for comparison. The Slater integral package we developed is divided into 4 modules. The first module, **define_grid**, gets input data for the grid and sets up the knots for splines according to the optimized grid scheme. The second module, **define_spline**, initializes the splines and their derivatives and evaluates the hydrogenic matrix elements (the one-electron Hamiltonian operator in spline basis). The first two modules are set up for general applications. The third module, **set_Slater_matrix_elements**, sets up the Slater matrix elements for a given k with both the integration-by-cell algorithm and the traditional one. The fourth module, **test_Slater_integrals**, determines a set of hydrogenic orbitals in terms of the spline basis and calculates a series of Slater integrals from the Slater matrix elements, all of the same k . The flowchart of the Slater integral package is shown in Fig. (4.1).

Extensive tests were performed for the evaluation of the Slater matrix elements and the Slater integrals for a variety of parameters but only a few sets are given here. The results are obtained using a grid with $Z = 1$, $h = 1/8$ and $1/4$, $R = 160$ *a.u.*, and $k_s = 4, 5, 6, 8$ where $nv = 52$ for $h = 1/8$ and $nv = 27$ for $h = 1/4$.

The calculations were performed on a Sun workstation (CPU: UltraSPARC 143MHz; RAM: 64MB) in double precision where the fractional part consists of 52 bits for an

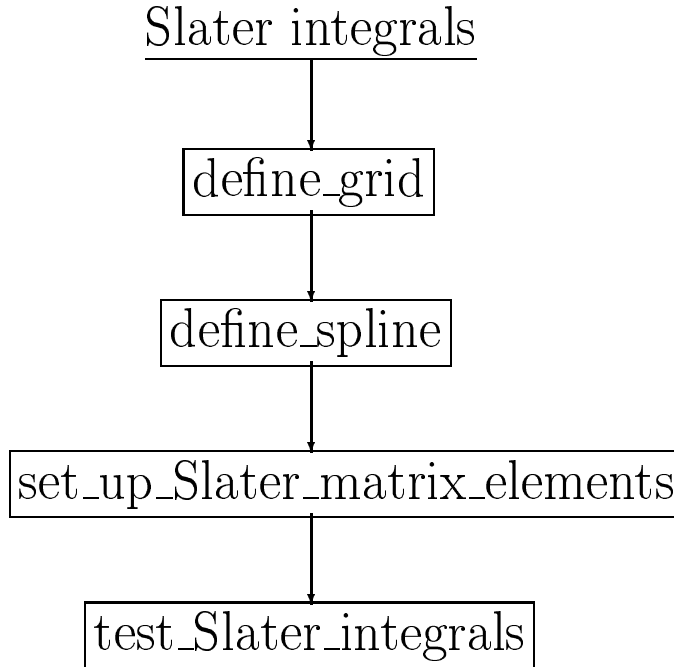


Figure 4.1: Flowchart of the Slater integral package.

accuracy of 15 significant digits. The user and system time is returned using the system function `DTIME()`. We find the two schemes of the integration-by-cell algorithm are almost equally efficient in terms of the user and system time. However, the results evaluated with the second scheme mentioned in Section (3.4) are much less accurate for the ns Slater integrals such as $F^0(1s, 1s)$ and $G^0(1s, 2s)$ where the radial wavefunctions of the integrands are localized near the origin. For example, for the above parameters and $k_s = 8$, the value of $F^0(1s, 1s)$ evaluated with the second scheme is about 3 orders of magnitude less accurate. The reason for this is obvious. The Slater matrix integrand has a singularity at the origin. The difficulty of the cell integrals caused by the singularity at the origin is minimized by the chosen Gaussian grid points which are densely populated near the origin in the first scheme of the integration-by-cell algorithm while it is not treated with care in the second scheme.

In the following, we only compare the results evaluated with the first scheme of the integration-by-cell algorithm and those with the traditional method of solving the differential equations. The comparison of the time in setting up the Slater matrix elements is shown in table (4.1) where t_1 is the time with the traditional method and t_2 is the time with the first scheme of the integration-by-cell algorithm including the time for evaluating the splines at the new points. We find that the integration by cell method is several times faster in evaluating the Slater matrix elements than the traditional method by solving the differential equations for all the choices of k_s and the two cases of h (1/8 and 1/4) . Since accuracy is also one of our major concerns, we compare the differences between the exact value and the results evaluated with the two methods in table (4.2), where Difference 1 is the deviation between the exact value and the Slater integral assembled from the Slater matrix elements which are evaluated with the traditional method and the input parameter $h = 1/8$, Difference 2 is the deviation between the exact value and the Slater integral assembled from the Slater matrix elements with the integration-by-cell method and the same input parameters as Difference 1, and t_3 is the user and system time of assembling the Slater integral from the Slater matrix elements. We see that all of the Slater integrals obtained by the integration by cell method are systematically at least as accurate as the one obtained by the traditional method. Moreover, unlike the traditional method where some of the Slater integrals, such as $F^0(4s, 4s)$, $F^2(4f, 4f)$, are significantly less accurate than others , the integration by cell method gives uniformly accurate results for all the Slater integrals. When k_s increases the difference between the exact value of the Slater integrals and the one evaluated with the integration by cell method decreases. When $k_s = 8$, both values converge.

Table 4.1: User and System time in seconds for setting up all of the Slater matrix elements for $k_s = 4, 5, 6, 8$ on a Sun workstation (CPU: UltraSPARC 143MHz; RAM: 64MB). k : the order of the Slater integrals; k_s : the order of the B-splines; h : the starting step size of the grid; t_1 : the time with the traditional method; t_2 : the time with the integration by cell method.

$h = 1/8$				$h = 1/4$			
k	t_1	t_2	t_1/t_2	k	t_1	t_2	t_1/t_2
$k_s = 4$				$k_s = 4$			
0	1.1(+00)	1.1(-01)	1.0(+01)	0	3.3(-01)	4.1(-02)	7.9(+00)
1	1.1(+00)	1.0(-01)	1.1(+01)	1	3.2(-01)	3.8(-02)	8.3(+00)
2	1.1(+00)	1.0(-01)	1.1(+01)	2	3.2(-01)	3.8(-02)	8.3(+00)
3	1.1(+00)	1.1(-01)	1.1(+01)	3	3.2(-01)	3.9(-02)	8.3(+00)
4	1.1(+00)	1.1(-01)	1.1(+01)	4	3.2(-01)	3.9(-02)	8.3(+00)
6	1.1(+00)	1.1(-01)	1.1(+01)	6	3.2(-01)	3.9(-02)	8.3(+00)
$k_s = 5$				$k_s = 5$			
0	2.4(+00)	2.4(-01)	9.9(+00)	0	6.4(-01)	9.5(-02)	6.7(+00)
1	2.4(+00)	2.3(-01)	1.0(+01)	1	6.3(-01)	9.1(-02)	7.0(+00)
2	2.4(+00)	2.3(-01)	1.0(+01)	2	6.4(-01)	9.1(-02)	7.0(+00)
3	2.4(+00)	2.3(-01)	1.0(+01)	3	6.3(-01)	9.2(-02)	6.9(+00)
4	2.4(+00)	2.3(-01)	1.0(+01)	4	6.3(-01)	9.1(-02)	7.0(+00)
6	2.4(+00)	2.3(-01)	1.0(+01)	6	6.4(-01)	9.1(-02)	7.0(+00)
$k_s = 6$				$k_s = 6$			
0	4.0(+00)	4.5(-01)	8.9(+00)	0	1.1(+00)	2.0(-01)	5.6(+00)
1	4.0(+00)	4.4(-01)	9.1(+00)	1	1.1(+00)	2.0(-01)	5.8(+00)
2	4.0(+00)	4.3(-01)	9.1(+00)	2	1.1(+00)	2.0(-01)	5.8(+00)
3	4.0(+00)	4.4(-01)	9.1(+00)	3	1.1(+00)	2.0(-01)	5.7(+00)
4	4.0(+00)	4.4(-01)	9.1(+00)	4	1.1(+00)	2.0(-01)	5.8(+00)
6	4.0(+00)	4.4(-01)	9.1(+00)	6	1.1(+00)	2.0(-01)	5.8(+00)
$k_s = 8$				$k_s = 8$			
0	1.0(+01)	1.5(+00)	6.9(+00)	0	3.0(+00)	7.4(-01)	4.0(+00)
1	1.0(+01)	1.5(+00)	6.9(+00)	1	3.0(+00)	7.4(-01)	4.0(+00)
2	1.0(+01)	1.5(+00)	6.9(+00)	2	3.0(+00)	7.4(-01)	4.0(+00)
3	1.0(+01)	1.5(+00)	6.9(+00)	3	3.0(+00)	7.4(-01)	4.0(+00)
4	1.0(+01)	1.5(+00)	6.9(+00)	4	3.1(+00)	7.4(-01)	4.1(+00)
6	1.0(+01)	1.5(+00)	6.9(+00)	6	3.0(+00)	7.4(-01)	4.0(+00)

It is important to note that once all of the Slater matrix elements are prepared the user and system time t_3 in calculating the Slater integrals is trivial compared to the time needed in evaluating the matrix elements. It is obvious from the tables (4.1, 4.2) that $t_3 \ll t_1$ and $t_3 \ll t_2$ for a given k_s . Therefore, the effectiveness in the evaluation of the Slater matrix elements demonstrated by the integration by cell method indeed significantly improves the evaluation of the Slater integrals.

Table 4.2: Comparison of the accuracy of some F^k and G^k integrals for different order of B-splines k_s . The exact values of the Slater integrals are from Ref. [9]. Difference 1: the difference of the exact value and the one evaluated with the traditional method; Difference 2: the difference of the exact value and one evaluated with the integration by cell method. t_3 : the user and system time of assembling the Slater integral from the Slater matrix elements.

Table 4.2a) $k_s = 4$.

F^k/G^k	Exact value	Difference 1	Difference 2	t_3 (Seconds)
$F^0(1s, 1s)$	5/8	-4.8(-09)	-3.0(-11)	2.3(-03)
$F^0(1s, 2s)$	17/81	-4.6(-10)	-2.4(-11)	4.1(-03)
$F^0(1s, 2p)$	59/243	1.6(-10)	-1.6(-11)	4.0(-03)
$F^0(2s, 2s)$	77/512	-1.3(-09)	-6.8(-11)	2.1(-03)
$F^0(2s, 2p)$	83/512	-6.8(-10)	-4.5(-11)	4.1(-03)
$F^0(2p, 2p)$	93/512	-6.6(-10)	-2.5(-11)	2.4(-03)
$F^0(4s, 4s)$	19541/524288	-3.8(-09)	-5.3(-10)	2.1(-03)
$F^0(4s, 4p)$	19943/524288	-3.2(-09)	-4.6(-10)	4.0(-03)
$F^0(4s, 4d)$	20693/524288	-1.3(-09)	-3.6(-10)	4.0(-03)
$F^0(4s, 4f)$	21743/524288	1.2(-10)	-2.8(-10)	4.1(-03)
$F^0(4p, 4p)$	20413/524288	-3.0(-09)	-3.9(-10)	2.2(-03)
$F^0(4p, 4d)$	21239/524288	-1.7(-09)	-2.9(-10)	4.0(-03)
$F^0(4p, 4f)$	22373/524288	-1.1(-12)	-2.1(-10)	3.9(-03)
$F^0(4d, 4d)$	22373/524288	-1.8(-09)	-1.9(-10)	2.2(-03)
$F^0(4d, 4f)$	23759/524288	-3.8(-10)	-1.2(-10)	3.9(-03)
$F^0(4f, 4f)$	26333/524288	-5.6(-10)	-4.3(-11)	2.2(-03)
$G^0(1s, 2s)$	16/729	-7.3(-10)	2.5(-12)	2.4(-03)
$G^0(2p, 3p)$	96768/9765625	-3.8(-10)	7.0(-12)	2.2(-03)
$G^0(2p, 4p)$	560/177147	-2.0(-10)	4.1(-12)	2.2(-03)
$G^1(1s, 2p)$	112/2187	-2.5(-10)	1.9(-12)	2.3(-03)
$G^1(2s, 2p)$	45/512	-9.9(-10)	-3.6(-11)	2.1(-03)
$G^1(2p, 3s)$	92016/9765625	-1.0(-09)	1.8(-11)	2.2(-03)
$G^1(2p, 3d)$	1824768/48828125	4.0(-11)	4.1(-12)	2.2(-03)
$G^1(2p, 4s)$	5168/1594323	-5.5(-10)	1.3(-11)	2.1(-03)
$G^1(2p, 4d)$	19120/1594323	-2.1(-10)	3.3(-12)	2.2(-03)
$F^2(4f, 4f)$	103275/3670016	-1.0(-09)	-3.7(-11)	2.4(-03)
$G^2(2p, 3p)$	110592/9765625	-8.9(-10)	2.3(-11)	2.2(-03)
$G^2(2p, 4p)$	2128/531441	-5.2(-10)	1.3(-11)	2.1(-03)
$G^2(2p, 4f)$	4784/1594323	-6.7(-11)	1.7(-12)	2.2(-03)
$G^3(2p, 3d)$	1064448/48828125	-3.6(-10)	1.0(-12)	2.2(-03)
$G^3(2p, 4d)$	3920/531441	-2.7(-10)	2.7(-12)	2.2(-03)
$F^4(4f, 4f)$	69003/3670016	-1.0(-09)	-2.8(-11)	2.3(-03)
$G^4(2p, 4f)$	1040/531441	-4.9(-11)	1.3(-12)	2.3(-03)
$F^6(4f, 4f)$	7293/524288	-6.9(-10)	-2.1(-11)	2.2(-03)

Table 4.2b) $k_s = 5$.

F^k/G^k	Exact value	Difference 1	Difference 2	t_3 (Seconds)
$F^0(1s, 1s)$	5/8	-2.0(-11)	-1.9(-13)	4.1(-03)
$F^0(1s, 2s)$	17/81	-2.3(-12)	-2.3(-13)	7.3(-03)
$F^0(1s, 2p)$	59/243	9.8(-13)	-1.6(-13)	7.0(-03)
$F^0(2s, 2s)$	77/512	-1.3(-11)	-8.9(-13)	4.0(-03)
$F^0(2s, 2p)$	83/512	-6.0(-12)	-5.5(-13)	6.9(-03)
$F^0(2p, 2p)$	93/512	-5.0(-12)	-2.9(-13)	3.8(-03)
$F^0(4s, 4s)$	19541/524288	-1.1(-10)	-1.3(-11)	3.7(-03)
$F^0(4s, 4p)$	19943/524288	-8.8(-11)	-1.1(-11)	6.7(-03)
$F^0(4s, 4d)$	20693/524288	-2.9(-11)	-8.2(-12)	6.9(-03)
$F^0(4s, 4f)$	21743/524288	5.4(-12)	-5.8(-12)	7.0(-03)
$F^0(4p, 4p)$	20413/524288	-8.0(-11)	-9.0(-12)	4.0(-03)
$F^0(4p, 4d)$	21239/524288	-3.8(-11)	-6.5(-12)	6.9(-03)
$F^0(4p, 4f)$	22373/524288	2.0(-12)	-4.2(-12)	6.8(-03)
$F^0(4d, 4d)$	22373/524288	-3.6(-11)	-4.1(-12)	3.9(-03)
$F^0(4d, 4f)$	23759/524288	-6.0(-12)	-2.1(-12)	6.7(-03)
$F^0(4f, 4f)$	26333/524288	-7.0(-12)	-7.0(-13)	3.9(-03)
$G^0(1s, 2s)$	16/729	-3.6(-12)	4.8(-14)	3.7(-03)
$G^0(2p, 3p)$	96768/9765625	-4.1(-12)	1.5(-13)	3.9(-03)
$G^0(2p, 4p)$	560/177147	-2.5(-12)	7.5(-14)	3.9(-03)
$G^1(1s, 2p)$	112/2187	-2.5(-12)	9.9(-15)	4.1(-03)
$G^1(2s, 2p)$	45/512	-2.0(-11)	-3.8(-13)	3.9(-03)
$G^1(2p, 3s)$	92016/9765625	-1.3(-11)	4.2(-13)	3.8(-03)
$G^1(2p, 3d)$	1824768/48828125	-7.8(-12)	3.5(-14)	3.8(-03)
$G^1(2p, 4s)$	5168/1594323	-8.0(-12)	2.4(-13)	3.9(-03)
$G^1(2p, 4d)$	19120/1594323	-3.8(-12)	-1.3(-15)	3.7(-03)
$F^2(4f, 4f)$	103275/3670016	-1.9(-11)	-4.0(-13)	4.1(-03)
$G^2(2p, 3p)$	110592/9765625	-1.1(-11)	4.0(-13)	3.9(-03)
$G^2(2p, 4p)$	2128/531441	-7.7(-12)	1.6(-13)	3.8(-03)
$G^2(2p, 4f)$	4784/1594323	-8.6(-13)	1.7(-14)	3.7(-03)
$G^3(2p, 3d)$	1064448/48828125	-4.4(-12)	-3.0(-14)	4.0(-03)
$G^3(2p, 4d)$	3920/531441	-3.9(-12)	-3.2(-14)	3.7(-03)
$F^4(4f, 4f)$	69003/3670016	-1.8(-11)	-2.0(-13)	4.1(-03)
$G^4(2p, 4f)$	1040/531441	-7.2(-13)	1.4(-14)	3.7(-03)
$F^6(4f, 4f)$	7293/524288	-1.3(-11)	-1.0(-13)	4.1(-03)

Table 4.2c) $k_s = 6$.

F^k/G^k	Exact value	Difference 1	Difference 2	t_3 (Seconds)
$F^0(1s, 1s)$	5/8	-1.1(-13)	-1.8(-15)	6.2(-03)
$F^0(1s, 2s)$	17/81	-2.2(-14)	-3.2(-15)	1.1(-02)
$F^0(1s, 2p)$	59/243	5.5(-15)	-2.2(-15)	1.1(-02)
$F^0(2s, 2s)$	77/512	-1.9(-13)	-1.4(-14)	5.8(-03)
$F^0(2s, 2p)$	83/512	-8.5(-14)	-7.8(-15)	1.1(-02)
$F^0(2p, 2p)$	93/512	-6.3(-14)	-3.7(-15)	5.8(-03)
$F^0(4s, 4s)$	19541/524288	-4.6(-12)	-4.0(-13)	5.6(-03)
$F^0(4s, 4p)$	19943/524288	-3.4(-12)	-3.3(-13)	1.1(-02)
$F^0(4s, 4d)$	20693/524288	-9.8(-13)	-2.4(-13)	1.1(-02)
$F^0(4s, 4f)$	21743/524288	2.2(-13)	-1.4(-13)	1.1(-02)
$F^0(4p, 4p)$	20413/524288	-3.0(-12)	-2.7(-13)	5.7(-03)
$F^0(4p, 4d)$	21239/524288	-1.3(-12)	-1.8(-13)	1.1(-02)
$F^0(4p, 4f)$	22373/524288	8.3(-14)	-1.0(-13)	1.1(-02)
$F^0(4d, 4d)$	22373/524288	-1.1(-12)	-1.1(-13)	5.7(-03)
$F^0(4d, 4f)$	23759/524288	-1.5(-13)	-4.7(-14)	1.1(-02)
$F^0(4f, 4f)$	26333/524288	-1.4(-13)	-1.3(-14)	5.9(-03)
$G^0(1s, 2s)$	16/729	-2.7(-14)	7.6(-16)	5.7(-03)
$G^0(2p, 3p)$	96768/9765625	-7.0(-14)	3.2(-15)	5.6(-03)
$G^0(2p, 4p)$	560/177147	-5.0(-14)	1.1(-15)	5.6(-03)
$G^1(1s, 2p)$	112/2187	-3.1(-13)	-1.9(-16)	6.3(-03)
$G^1(2s, 2p)$	45/512	-4.9(-12)	-3.2(-15)	5.9(-03)
$G^1(2p, 3s)$	92016/9765625	-4.1(-13)	9.9(-15)	5.6(-03)
$G^1(2p, 3d)$	1824768/48828125	-4.0(-12)	-5.3(-16)	5.7(-03)
$G^1(2p, 4s)$	5168/1594323	-2.0(-13)	3.9(-15)	5.8(-03)
$G^1(2p, 4d)$	19120/1594323	-5.7(-13)	-1.6(-15)	5.6(-03)
$F^2(4f, 4f)$	103275/3670016	-3.8(-12)	2.3(-15)	6.2(-03)
$G^2(2p, 3p)$	110592/9765625	-2.4(-13)	7.4(-15)	5.6(-03)
$G^2(2p, 4p)$	2128/531441	-1.7(-13)	6.4(-16)	5.6(-03)
$G^2(2p, 4f)$	4784/1594323	-3.2(-14)	3.9(-16)	5.6(-03)
$G^3(2p, 3d)$	1064448/48828125	-8.4(-14)	-2.2(-15)	6.3(-03)
$G^3(2p, 4d)$	3920/531441	-8.6(-14)	-2.5(-15)	5.6(-03)
$F^4(4f, 4f)$	69003/3670016	-4.6(-13)	8.8(-15)	6.3(-03)
$G^4(2p, 4f)$	1040/531441	-1.5(-14)	3.8(-16)	5.8(-03)
$F^6(4f, 4f)$	7293/524288	-3.6(-13)	1.0(-14)	6.2(-03)

Table 4.2d) $k_s = 8$.

F^k/G^k	Exact value	Difference 1	Difference 2	t_3 (Seconds)
$F^0(1s, 1s)$	5/8	-4.2(-15)	4.4(-16)	1.2(-02)
$F^0(1s, 2s)$	17/81	-1.4(-15)	1.4(-16)	2.1(-02)
$F^0(1s, 2p)$	59/243	-1.9(-15)	-1.4(-16)	2.1(-02)
$F^0(2s, 2s)$	77/512	-7.2(-16)	5.6(-17)	1.2(-02)
$F^0(2s, 2p)$	83/512	-4.7(-16)	1.9(-16)	2.1(-02)
$F^0(2p, 2p)$	93/512	-8.9(-16)	0.0(+00)	1.2(-02)
$F^0(4s, 4s)$	19541/524288	-1.6(-14)	-6.8(-16)	1.2(-02)
$F^0(4s, 4p)$	19943/524288	-1.1(-14)	-5.1(-16)	2.1(-02)
$F^0(4s, 4d)$	20693/524288	-2.8(-15)	-2.8(-16)	2.1(-02)
$F^0(4s, 4f)$	21743/524288	8.8(-16)	-6.2(-17)	2.1(-02)
$F^0(4p, 4p)$	20413/524288	-8.9(-15)	-3.5(-16)	1.2(-02)
$F^0(4p, 4d)$	21239/524288	-3.3(-15)	-1.7(-16)	2.1(-02)
$F^0(4p, 4f)$	22373/524288	4.0(-16)	-9.7(-17)	2.1(-02)
$F^0(4d, 4d)$	22373/524288	-2.2(-15)	-6.9(-17)	1.2(-02)
$F^0(4d, 4f)$	23759/524288	-8.3(-17)	3.5(-17)	2.1(-02)
$F^0(4f, 4f)$	26333/524288	6.2(-17)	2.8(-17)	1.2(-02)
$G^0(1s, 2s)$	16/729	-6.6(-17)	2.8(-17)	1.2(-02)
$G^0(2p, 3p)$	96768/9765625	-1.1(-16)	5.2(-18)	1.2(-02)
$G^0(2p, 4p)$	560/177147	-7.0(-17)	1.3(-18)	1.2(-02)
$G^1(1s, 2p)$	112/2187	-5.3(-15)	2.1(-17)	1.2(-02)
$G^1(2s, 2p)$	45/512	-8.2(-14)	0.0(+00)	1.2(-02)
$G^1(2p, 3s)$	92016/9765625	-3.0(-15)	2.1(-17)	1.2(-02)
$G^1(2p, 3d)$	1824768/48828125	-6.9(-14)	-2.1(-17)	1.2(-02)
$G^1(2p, 4s)$	5168/1594323	-8.5(-16)	0.0(+00)	1.2(-02)
$G^1(2p, 4d)$	19120/1594323	-9.0(-15)	1.7(-18)	1.2(-02)
$F^2(4f, 4f)$	103275/3670016	-6.3(-14)	3.1(-17)	1.2(-02)
$G^2(2p, 3p)$	110592/9765625	-5.3(-16)	0.0(+00)	1.2(-02)
$G^2(2p, 4p)$	2128/531441	-2.7(-16)	-7.8(-18)	1.2(-02)
$G^2(2p, 4f)$	4784/1594323	-2.9(-16)	-1.3(-18)	1.2(-02)
$G^3(2p, 3d)$	1064448/48828125	-9.7(-17)	-1.7(-17)	1.2(-02)
$G^3(2p, 4d)$	3920/531441	-1.1(-16)	-4.3(-18)	1.2(-02)
$F^4(4f, 4f)$	69003/3670016	-7.8(-16)	3.8(-17)	1.2(-02)
$G^4(2p, 4f)$	1040/531441	-1.9(-17)	-1.7(-18)	1.2(-02)
$F^6(4f, 4f)$	7293/524288	-7.4(-16)	3.5(-17)	1.2(-02)

CHAPTER V

CONCLUSION

We have developed an algorithm for evaluating Slater integrals in a spline basis (B-spline). The algorithm divides the two dimensional configuration space into a number of rectangular cells according to our chosen general grid and implements the two dimensional integration over each individual cells using Gaussian quadrature. Over the off-diagonal cells, the two dimensional cell-integrals are reduced to a product of two one-dimensional integrals. Over each diagonal cell, the rectangular cell integral is transformed into two triangular cell integrals to overcome the discontinuity of the integrand and the available B-spline values used in the one-dimensional integration are reused. Furthermore, the scaling invariance of the B-splines in the logarithmic region of the chosen grid is fully exploited. The values of the Slater matrix elements and the given Slater integrals are obtained by assembling the cell integrals. This algorithm significantly improves the efficiency and accuracy of the traditional method of solving differential equations and renders the B-spline methods much more effective when applied to multi-electron atomic systems.

Appendix A

DERIVATION OF SCALING LAWS

$$\int_{r_{iv+1}}^{r_{iv+2}} B_{i+1}^{k_s}(r) B_{j+1}^{k_s}(r) r^k dr = (1+h)^{1+k} \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r) B_j^{k_s}(r) r^k dr \quad (1.1)$$

$$\int_{r_{iv+1}}^{r_{iv+2}} B_{i+1}^{k_s}(r) B_{j+1}^{k_s}(r) r^k dr$$

Let $r - r_{iv+1} = t$

$$= \int_0^{(r_{iv+2}-r_{iv+1})} B_{i+1}^{k_s}(t + r_{iv+1}) B_{j+1}^{k_s}(t + r_{iv+1}) (t + r_{iv+1})^k dt$$

Let $t/(1+h) = x$

$$= (1+h) \int_0^{(r_{iv+2}-r_{iv+1})/(1+h)} B_{i+1}^{k_s}[(1+h)(x + r_{iv})] B_{j+1}^{k_s}[(1+h)(x + r_{iv})] (1+h)^k (x + r_{iv})^k dx$$

$$= (1+h)^{1+k} \int_0^{r_{iv+1}-r_{iv}} B_i^{k_s}(x + r_{iv}) B_j^{k_s}(x + r_{iv}) (x + r_{iv})^k dx$$

Let $x + r_{iv} = r$

$$= (1+h)^{1+k} \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r) B_j^{k_s}(r) r^k dr.$$

$$\begin{aligned} & \int_{r_{iv+1}}^{r_{iv+2}} \int_{r_{jv+1}}^{r_{jv+2}} \frac{r_{<}^k}{r_{>}^{k+1}} B_{i+1}^{k_s}(r_1) B_{j+1}^{k_s}(r_2) B_{i'+1}^{k_s}(r_1) B_{j'+1}^{k_s}(r_2) dr_1 dr_2 \\ &= (1+h) \int_{r_{iv}}^{r_{iv+1}} \int_{r_{jv}}^{r_{jv+1}} \frac{r_{<}^k}{r_{>}^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2 \end{aligned} \quad (1.2)$$

To show equation (1.2), there are two possibilities: $iv = jv$ and $iv \neq jv$.

1) $iv \neq jv$.

Since iv and jv are symmetric, we can assume $iv < jv$ generally. In this case, the two dimensional integration is completely separable.

$$\int_{r_{iv+1}}^{r_{iv+2}} \int_{r_{jv+1}}^{r_{jv+2}} \frac{r_{<}^k}{r_{>}^{k+1}} B_{i+1}^{k_s}(r_1) B_{j+1}^{k_s}(r_2) B_{i'+1}^{k_s}(r_1) B_{j'+1}^{k_s}(r_2) dr_1 dr_2$$

$$\begin{aligned}
&= \int_{r_{iv+1}}^{r_{iv+2}} r_1^k B_{i+1}^{k_s}(r_1) B_{i'+1}^{k_s}(r_1) dr_1 \int_{r_{jv+1}}^{r_{jv+2}} \frac{1}{r_2^{k+1}} B_{j+1}^{k_s}(r_2) B_{j'+1}^{k_s}(r_2) dr_2 \\
&= (1+h)^{1+k} \int_{r_{iv}}^{r_{iv+1}} r_1^k B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1 \cdot (1+h)^{-k} \int_{r_{jv}}^{r_{jv+1}} \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \\
&= (1+h) \int_{r_{iv}}^{r_{iv+1}} \int_{r_{jv}}^{r_{jv+1}} \frac{r_1^k}{r_2^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2
\end{aligned}$$

2) $iv = jv$.

$$\begin{aligned}
&\int_{r_{iv+1}}^{r_{iv+2}} \int_{r_{iv+1}}^{r_{iv+2}} \frac{r_1^k}{r_2^{k+1}} B_{i+1}^{k_s}(r_1) B_{j+1}^{k_s}(r_2) B_{i'+1}^{k_s}(r_1) B_{j'+1}^{k_s}(r_2) dr_1 dr_2 \\
&= \int_{r_{iv+1}}^{r_{iv+2}} B_{i+1}^{k_s}(r_1) B_{i'+1}^{k_s}(r_1) dr_1 \left\{ \frac{1}{r_1^{k+1}} \int_{r_{iv+1}}^{r_1} r_2^k B_{j+1}^{k_s}(r_2) B_{j'+1}^{k_s}(r_2) dr_2 \right. \\
&\quad \left. + r_1^k \int_{r_1}^{r_{iv+2}} \frac{1}{r_2^{k+1}} B_{j+1}^{k_s}(r_2) B_{j'+1}^{k_s}(r_2) dr_2 \right\}
\end{aligned}$$

Let $t_1 = r_1 - r_{iv+1}$, $t_2 = r_2 - r_{iv+1}$

$$\begin{aligned}
&= \int_0^{(r_{iv+2}-r_{iv+1})} B_{i+1}^{k_s}(t_1 + r_{iv+1}) B_{i'+1}^{k_s}(t_1 + r_{iv+1}) dt_1 \\
&\cdot \left\{ \frac{1}{(t_1 + r_{iv+1})^{k+1}} \int_0^{t_1} (t_2 + r_{iv+1})^k B_{j+1}^{k_s}(t_2 + r_{iv+1}) B_{j'+1}^{k_s}(t_2 + r_{iv+1}) dt_2 \right. \\
&\quad \left. + (t_1 + r_{iv+1})^k \int_{t_1}^{(r_{iv+2}-r_{iv+1})} \frac{1}{(t_2 + r_{iv+1})^{k+1}} B_{j+1}^{k_s}(t_2 + r_{iv+1}) B_{j'+1}^{k_s}(t_2 + r_{iv+1}) dt_2 \right\}
\end{aligned}$$

Let $r_{iv+1} = (1+h)r_{iv}$, $(1+h)x_1 = t_1$, $(1+h)x_2 = t_2$

$$\begin{aligned}
&= (1+h) \int_0^{(r_{iv+1}-r_{iv})} B_{i+1}^{k_s}[(1+h)(x_1 + r_{iv})] B_{i'+1}^{k_s}[(1+h)(x_1 + r_{iv})] dx_1 \\
&\cdot \left\{ \frac{1}{(x_1 + r_{iv})^{k+1}} \int_0^{x_1} (x_2 + r_{iv})^k B_{j+1}^{k_s}[(1+h)(x_2 + r_{iv})] B_{j'+1}^{k_s}[(1+h)(x_2 + r_{iv})] dx_2 \right. \\
&\quad \left. + (x_1 + r_{iv})^k \int_{x_1}^{(r_{iv+1}-r_{iv})} \frac{1}{(x_2 + r_{iv})^{k+1}} B_{j+1}^{k_s}[(1+h)(x_2 + r_{iv})] B_{j'+1}^{k_s}[(1+h)(x_2 + r_{iv})] dx_2 \right\} \\
&= (1+h) \int_0^{(r_{iv+1}-r_{iv})} B_i^{k_s}(x_1 + r_{iv}) B_{i'}^{k_s}(x_1 + r_{iv}) dx_1 \\
&\cdot \left\{ \frac{1}{(x_1 + r_{iv})^{k+1}} \int_0^{x_1} (x_2 + r_{iv})^k B_j^{k_s}(x_2 + r_{iv}) B_{j'}^{k_s}(x_2 + r_{iv}) dx_2 \right. \\
&\quad \left. + (x_1 + r_{iv})^k \int_{x_1}^{(r_{iv+1}-r_{iv})} \frac{1}{(x_2 + r_{iv})^{k+1}} B_j^{k_s}(x_2 + r_{iv}) B_{j'}^{k_s}(x_2 + r_{iv}) dx_2 \right\}
\end{aligned}$$

Let $x_1 + r_{iv} = r_1$, and $x_2 + r_{iv} = r_2$

$$= (1+h) \int_{r_{iv}}^{r_{iv+1}} B_i^{k_s}(r_1) B_{i'}^{k_s}(r_1) dr_1 \cdot \left\{ \frac{1}{r_1^{k+1}} \int_{r_{iv}}^{r_1} r_2^k B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \right.$$

$$\begin{aligned}
& + r_1^k \int_{r_1}^{r_{iv+1}} \frac{1}{r_2^{k+1}} B_j^{k_s}(r_2) B_{j'}^{k_s}(r_2) dr_2 \Big\} \\
= & (1+h) \int_{r_{iv}}^{r_{v+1}} \int_{r_{jv}}^{r_{jv+1}} \frac{r_{>}^k}{r_{<}^{k+1}} B_i^{k_s}(r_1) B_j^{k_s}(r_2) B_{i'}^{k_s}(r_1) B_{j'}^{k_s}(r_2) dr_1 dr_2
\end{aligned}$$

Appendix B

CODE FOR EVALUATING THE CELL INTEGRALS AND ASSEMBLING
SLATER MATRIX ELEMENTS

```
!=====
SUBROUTINE slaterMatrixElements(k,rk)
!=====
!
! Assembles Slater matrix elements of power k in the Spline basis
!
! SUBROUTINE called:
!   pmoments
!
! Calling sequence:
!
!   slaterMatrixElements
!   -----
!           |
!           pmoments
!           /   \
!   moment pdiag
!           ||
!           triang
```

```

!           |
!           gauss
!
!-----
!
!  on entry
!  -----
!      k      the power of the slater matrix elements
!
!  on exit
!  -----
!      rk      four-dimensional array of Slater matrix elements of power k in
!              the Spline basis
!
!-----
!
      USE spline_param

      IMPLICIT NONE

      INTEGER, INTENT(IN) :: k

      REAL(KIND=8), DIMENSION(ns,ns,ks,ks), INTENT(OUT) :: rk

! .. local variables

      INTEGER :: iv,ih,i, ihp,ip, jv,jh,j, jhp,jp

```

```

REAL(KIND=8), DIMENSION(ks,ks,nv) :: rkm, rkkm
REAL(KIND=8), DIMENSION(ks,ks,ks,ks,nv) :: rkt

CALL pmoments(k, rkm, rkkm, rkt)

rk=0.0d0

DO jv=1,nv
  DO jh=1,ks
    j = jv + jh - 1
    DO jhp=jh,ks
      jp = jhp - jh + 1

      DO iv=1,nv
        DO ih=1,ks
          i = iv + ih -1
          DO ihp=ih,ks
            ip = ihp - ih + 1

            IF( iv < jv ) THEN
              rk(i,j,ip,jp) = rk(i,j,ip,jp) + &
                rkm(ih,ihp,iv)*rkkm(jh,jhp,jv)
            ELSE IF( iv > jv ) THEN
              rk(i,j,ip,jp) = rk(i,j,ip,jp) + &
                rkm(jh,jhp,jv)*rkkm(ih,ihp,iv)
            
```

```

ELSE
    rk(i,j,ip,jp) = rk(i,j,ip,jp) +      &
        rkt(ih,jh,ihp,jhp,iv) + rkt(jh,ih,jhp,ihp,iv)
END IF
END DO
END DO
END DO
END DO
END DO
END DO
END DO

```

END SUBROUTINE slaterMatrixElements

```

!=====
SUBROUTINE pmoments(k, rkm, rkkm, rkt)
!=====
!
!   Computes Slater moments in the spline basis
!
!   SUBROUTINES called:
!       moment
!       pdiag
!

```

```

!-----
!
!   on entry
!   -----
!       k       the power of the slater matrix elements
!
!   on exit:
!   -----
!       rkm:    the moments defining  $\langle B_i | r^k | B_j \rangle$  over an interval
!       rkkm:   the moments defining  $\langle B_i | 1/r^{(k+1)} | B_j \rangle$  over an interval
!       rkt     the four-dimensional array of pieces
!               defining  $\langle B_i B_j | r^k / r^{(k+1)} | B_i' B_j' \rangle$ 
!               over a triangle
!
!-----
!
USE spline_param

IMPLICIT NONE

INTEGER, INTENT(IN) :: k

REAL(KIND=8), DIMENSION(ks,ks,nv), INTENT(OUT) :: rkm, rkkm

REAL(KIND=8), DIMENSION(ks,ks,ks,ks,nv), INTENT(OUT) :: rkt

CALL moment(k, rkm)

```

```
CALL moment(-(k+1), rkkm)
```

```
CALL pdiag(k, rkt)
```

```
END SUBROUTINE pmoments
```

```
!=====
SUBROUTINE moment(k, rkm)
!=====
!
! Computes moments in intervals
!
!-----
!
! on entry
! -----
!      k      the power of moments
!
! on exit
! -----
!      rkm     the moments defining  $\langle B_i | r^k | B_j \rangle$  over an interval
!
!-----
!
```

```

USE spline_param

USE spline_grid

IMPLICIT NONE

INTEGER, INTENT(IN) :: k

REAL(KIND=8), DIMENSION(ks,ks,nv), INTENT(OUT) :: rkm

! .. local variables

INTEGER :: iv, i, j, mlpks,mtmp

REAL(KIND=8) :: hp1

IF ( k == 0) THEN

! .. the first equal step region

mlpks = ml+ks

DO iv=1,(mlpks-1)

DO i=1,ks

DO j=i,ks

rkm(i,j,iv) = SUM(bsp(iv,:,i)*bsp(iv,:,j) *grw(iv,:))

END DO

END DO

END DO

! .. the log region --- using scaling law

hp1=h+1.0d0

```

```

mtmp = ml+me-ks+2
DO iv=mlpks,mtmp
  DO i=1,ks
    DO j=i,ks
      rkm(i,j,iv) = rkm(i,j,iv-1)*hp1
    END DO
  END DO
END DO

! .. the last equal step region
DO iv=(mtmp+1),nv
  DO i=1,ks
    DO j=i,ks
      rkm(i,j,iv) = SUM(bsp(iv,:,i)*bsp(iv,:,j) *grw(iv,:))
    END DO
  END DO
END DO

ELSE IF ( k > 0) THEN
  ! .. the first equal step region
  mlpks = ml+ks
  DO iv=1,(mlpks-1)
    DO i=1,ks
      DO j=i,ks
        rkm(i,j,iv) =      &

```

```

                SUM(bsp(iv,:,i)*bsp(iv,:,j) *gr(iv,):**k *grw(iv,:))
        END DO

    END DO

END DO

! .. the log region --- using scaling law
hp1=(h+1.d0)**(1+k)
mtmp = ml+me-ks+2
DO iv=mlpks,mtmp

    DO i=1,ks

        DO j=i,ks

            rkm(i,j,iv) = rkm(i,j,iv-1)*hp1

        END DO

    END DO

END DO

! .. the last equal step region
DO iv=(mtmp+1),nv

    DO i=1,ks

        DO j=i,ks

            rkm(i,j,iv) =      &

                SUM(bsp(iv,:,i)*bsp(iv,:,j) *gr(iv,):**k *grw(iv,:))

        END DO

    END DO

END DO

```

```

END DO

ELSE IF ( k < 0 ) THEN

! .. the first equal step region

mlpks = ml+ks

DO iv=1,(mlpks-1)

  DO i=1,ks

    DO j=i,ks

      rkm(i,j,iv) =      &
          SUM(bsp(iv,:,i)*bsp(iv,:,j) *grm(iv,):**(-k) *grw(iv,:))

    END DO

  END DO

END DO

! .. the log region --- using scaling law

IF(k/=-1) THEN

  hp1=(h+1.d0)**(1+k)

ELSE

  hp1=1.d0

END IF

mtmp = ml+me-ks+2

DO iv=mlpks,mtmp

  DO i=1,ks

    DO j=i,ks

```

```

        rkm(i,j,iv) = rkm(i,j,iv-1)*hp1
    END DO

    END DO

END DO

! .. the last equal step region
DO iv=(mtmp+1),nv
    DO i=1,ks
        DO j=i,ks
            rkm(i,j,iv) =      &
                SUM(bsp(iv,:,i)*bsp(iv,:,j) *grm(iv,):**(-k) *grw(iv,:))
        END DO
    END DO
END DO

END IF

END SUBROUTINE moment

```

```

!=====

```

```

SUBROUTINE pdiag(k, rkt)

```

```

!=====

```

```

!
```

```

! Computes the Slater matrix elements in the triangle cells

```

```

!
!   SUBROUTINES contained:
!
!       triang
!
!-----
!
!   on entry
!   -----
!
!       k           the power of moments
!
!   on exit
!   -----
!
!       rkt         the four-dimensional array of pieces
!
!                   defining  $\langle B_i B_j | r^k / r^{(k+1)} | B_i' B_j' \rangle$ 
!
!                   over a triangle
!
!-----
!
!
!   USE spline_param
!
!   USE spline_grid
!
!
!   IMPLICIT NONE
!
!   INTEGER, INTENT(IN) :: k
!
!   REAL(KIND=8), DIMENSION(ks,ks,ks,ks,nv), INTENT(OUT) :: rkt

```

```

! .. local variables

INTEGER :: iv,i,j,ip,jp, mlpks,mtmp

REAL(KIND=8) :: hp1

IF(k<0) STOP 'k<0'

! .. the first equal step region.

mlpks = ml+ks

DO iv=1,(mlpks-1)

    CALL triang

END DO

! .. the log region --- using scaling law.

hp1=h+1.d0

mtmp = ml+me-ks+2

DO iv=mlpks,mtmp

    DO j=1,ks

        DO jp=j,ks

            DO i=1,ks

                DO ip=i,ks

                    rkt(i,j,ip,jp,iv) = hp1 * rkt(i,j,ip,jp,iv-1)

                END DO

            END DO

        END DO

    END DO

END DO

```

```

        END DO
    END DO
END DO

```

```

! .. the last equal step region

```

```

DO iv=(mtmp+1),nv

```

```

    CALL triang

```

```

END DO

```

```

CONTAINS

```

```

!=====

```

```

    SUBROUTINE triang

```

```

!=====

```

```

!
!
!                                     iv+1 -- iv+1
! Returns the "slater matrix element" in low triangle cell |      | ,
!
!                                     iv  -- iv+1
! i. e.,
!
! /
! /      k
! |      r_<
! | dr_1 dr_2  --- bsp(iv,:,i)(r_1) bsp(iv,:,j)(r_2)

```

```

! |          k+1
! |          r_>
! /          bsp(iv, :, ip)(r_1) bsp(iv, :, jp)(r_2)
! /
! -----
!
! SUBROUTINES called:
!     gauss
!     qbsplvb
!
! -----
!
! On entry
! -----
!     k:     the indices of of the bsplines
!     iv:    the index of the integration region
!
! -----
!
! USE spline_param
! USE spline_grid

```

IMPLICIT NONE

```

! .. local variables

INTEGER :: i, j, ip, jp, m, i1, left, ks1

REAL(KIND=8) :: xbase

REAL(KIND=8), DIMENSION(ks,ks) :: gx,gw,gxk

REAL(KIND=8), DIMENSION(ks,ks,ks) :: bspTmp

REAL(KIND=8), DIMENSION(ks) :: grmk,x,w,biatx,fdr2

!

!           / r(iv,m)                                     k

! .. fdr2 = |           bsp(iv,:,j)(r2) bsp(iv,:,jp)(r2) r2  dr2

!           / r_iv

!

left=iv+ks-1

xbase=t(left)

IF(k/=0) THEN

    grmk(1:ks) = grm(iv,1:ks)**(k+1)

ELSE

    grmk(1:ks) = grm(iv,1:ks)

END IF

! .. setup the gaussian points

! .. ks1 = ks-2

! .. if(ks1<4) ks1=ks

ks1 = ks

```

```

CALL gauss(ks1,x,w)

DO m=1,ks
  DO i1=1,ks1
    ! .. the absolute coordinate at the new gaussian point
    gx(i1,m) = (gr(iv,m)-xbase)*x(i1) + xbase

    IF(k>1) THEN
      gxk(i1,m) = gx(i1,m)**k
    ELSE IF(k==1) THEN
      gxk(i1,m) = gx(i1,m)
    END IF

    ! .. Note: K=0 is not relevent

    ! .. the corresponding gaussian weights
    gw(i1,m) = (gr(iv,m)-xbase)*w(i1)

    ! .. the bspline values at the new gaussian points
    CALL qbsplvb(ks, gx(i1,m), left, biatx)
    bspTmp(i1,m,1:ks)= biatx(1:ks)
  END DO
END DO

DO i=1,ks

```

```

DO ip=i,ks
  DO j=1,ks
    DO jp=j,ks
      DO m=1,ks
        fdr2(m)= 0.0d0
        DO i1=1,ks1
          IF(k/=0) THEN
            fdr2(m)= fdr2(m) + gxk(i1,m) * bspTmp(i1,m,j) * &
              bspTmp(i1,m,jp) * gw(i1,m)
          ELSE
            fdr2(m)= fdr2(m) + bspTmp(i1,m,j)*bspTmp(i1,m,jp)*gw(i1,m)
          END IF
        END DO
      END DO
    END DO
    rkt(i,j,ip,jp,iv) = SUM( bsp(iv,:,i) * bsp(iv,:,ip) * &
      grw(iv,:) * grmk(:) * fdr2(:) )
  END DO
END DO
END DO
END DO
END SUBROUTINE triang

END SUBROUTINE pdiag

```

```

=====
MODULE spline_param
=====
! The SPLINE_PARAM module defines the spline parameters
! -----

IMPLICIT NONE

SAVE

! .. Here are the commonly used parameters

INTEGER :: ks, ns, nv, ml,me

REAL(KIND=8) :: h

END MODULE spline_param

```

```

=====
MODULE spline_grid
=====
! The spline_grid module defines the values of splines at the gaussian
! points defined by the intervals of a grid. Included in the module
! is the gaussian data for performing integrations on the grid.
! -----

IMPLICIT NONE

```

SAVE

! .. arrays for defining grid

REAL(KIND=8), DIMENSION(:), ALLOCATABLE:: t

! .. arrays for initializing spline values

REAL(KIND=8), DIMENSION(:,,:), ALLOCATABLE :: bs

REAL(KIND=8), DIMENSION(:, :, :), ALLOCATABLE:: bsp

REAL(KIND=8), DIMENSION(:, :, :, :), ALLOCATABLE:: bspd

! .. arrays for initializing gaussian data

REAL(KIND=8), DIMENSION(:, :), ALLOCATABLE:: gr, grm, grw

CONTAINS

!=====

 SUBROUTINE allocate_grid

!=====

! This program allocates space of the arrays for initializing spline

! values and gaussian data in MODULE spline_grid

!-----

 USE spline_param

 IMPLICIT NONE

 ALLOCATE(bs(ks,ns), bsp(nv+1,ks,ks), bspd(nv+1,ks,ks,2))

```
        ALLOCATE( gr(nv,ks),grm(nv,ks),grw(nv,ks) )  
    END SUBROUTINE allocate_grid  
  
END MODULE spline_grid
```

BIBLIOGRAPHY

- [1] A. Altenberger-Siczek and T. L. Gilbert, *J. Chem. Phys.* **64**, 432, (1976).
- [2] C. Bottcher and M. R. Strayer, *Ann. Phys. NY* **175**, 64, (1987).
- [3] W. R. Johnson and J. Sapirstein, *Phys. Rev. Lett.* **57**, 1126, (1986).
- [4] W. R. Johnson, M. Idrees, and J. Sapirstein, *Phys. Rev. A* **35**, 3218, (1987).
- [5] W. R. Johnson, S. A. Blundell, and J. Sapirstein, *Phys. Rev. A* **35**, 3218, (1988).
- [6] C. Fischer and M. Idrees, *Comput. Phys.* **3**, 53, (1989).
- [7] C. Fischer and M. Idrees, *J. Phys. B* **23**, 679, (1990).
- [8] C. Fischer and W. Guo, *J. Comput. Phys.* **90**, 486, (1990).
- [9] C. Fischer, W. Guo, and Z. Shen, *Int. J. Quantum. Chem.* **42**, 849, (1992).
- [10] T. N. Chang and X. Tang, *Phys. Rev. A* **44**, 232, (1991).
- [11] Y. T. Shen, M. Landtman, and J. E. Hansen, *J. Phys. B* **23**, L121, (1990).
- [12] H. van der Hart and J. E. Hansen, *J. Phys. B* **25**, 41, (1992).
- [13] P. Decleva, A. Lisini, and M. Venuti, *J. Phys. B* **27**, 4867, (1994).
- [14] M. Venuti, P. Decleva, and A. Lisini, *J. Phys. B* **29**, 5315, (1996).
- [15] M. Venuti and P. Decleva, *J. Phys. B* **30**, 4839, (1997).
- [16] H. van der Hart, *J. Phys. B* **30**, 453, (1997).
- [17] J. Sapirstein and W. R. Johnson, *J. Phys. B* **29**, 5213, (1996).
- [18] C. Fischer, *The Hartree-Fock Method for Atoms* (New York: John Wiley & Sons), 1977
- [19] C. deBoor, *A Practical Guide to Splines* (New York: Springer), 1978.
- [20] A. S. Umar, J. Wu, M. R. Strayer, and C. Bottcher *A Practical Guide to Splines* (New York: Springer) *J. Comput. Phys.* **93**, 426, (1991).
- [21] T. Brage, C. F. Fischer, and G. Miecznik *J. Phys. B* **25**, 5289, (1992).
- [22] J. H. Xi and C. Fischer, *Phys. Rev. A* **53**, 3169, (1996).
- [23] C. F. Fischer, *The Int. J. Supercom. Appl.* **5**, 5, (1991).

INTEGRATION BY CELL ALGORITHM FOR SLATER INTEGRALS IN A
SPLINE BASIS

YANGHUI QIU

Thesis under the direction of Professor Charlotte Froese Fischer

An algorithm for evaluating Slater integrals in a spline basis (B-spline) is introduced. Based on the piece-wise property of the B-splines, the algorithm divides the two dimensional configuration space into a number of rectangular cells according to an optimally chosen grid and implements the two dimensional integration over each individual cell using Gaussian quadrature. Over the off-diagonal cells, the integrands are separable so that each two dimensional cell-integral is reduced to a product of two one-dimensional integrals. Furthermore, the scaling invariance of the B-splines in the logarithmic region of the chosen grid is fully exploited such that only part of the cell integration needs to be implemented. The values of given Slater integrals are obtained by assembling the cell integrals. This algorithm significantly improves the efficiency and accuracy of the traditional method of solving differential equations and renders the B-spline method much more effective when applied to multi-electron atomic systems.

Approved_____ Date_____