GEN1INT Manual

Version 0.2.1

Bin Gao and Andreas J. Thorvaldsen

May 25, 2012

Centre for Theoretical and Computational Chemistry (CTCC)
Department of Chemistry
University of Tromsø, N–9037
Tromsø, Norway

Contents

Notation

The following notation conventions in Refs. [1–3] will be used throughout the manual: bold capital letters such as $\boldsymbol{R}_\kappa$ denote positions of nuclei (or centers). The vector from $\boldsymbol{R}_\lambda$ to $\boldsymbol{R}_\kappa$ is denoted by $\boldsymbol{R}_{\kappa\lambda} = \boldsymbol{R}_\kappa - \boldsymbol{R}_\lambda$. The capital letters X_κ, Y_κ, and Z_κ represent the Cartesian coordinates of a nucleus at position $\boldsymbol{R}_\kappa$, whereas R_κ denotes the norm of vector $\boldsymbol{R}_\kappa$. The position of an electron relative to a nucleus at position $\boldsymbol{R}_\kappa$ is given by $\boldsymbol{r}_\kappa = \boldsymbol{r} - \boldsymbol{R}_\kappa$. Small letters x_κ, y_κ and z_κ, and r_κ denote the three Cartesian coordinates of the electron relative to center $\boldsymbol{R}_\kappa$, and the norm of the vector $\boldsymbol{r}_\kappa$, respectively.

Moreover, the so-called **multi-index notation** [4] will be used extensively to simplify the expressions. For instance, the geometric derivatives with respect to a center $\boldsymbol{R}_\kappa$ are written as

$$\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{K}} = \left(\frac{\partial}{\partial X_\kappa}\right)^{K_X}\left(\frac{\partial}{\partial Y_\kappa}\right)^{K_Y}\left(\frac{\partial}{\partial Z_\kappa}\right)^{K_Z} = \frac{\partial^{|\boldsymbol{K}|}}{\partial X_\kappa^{K_X}\partial Y_\kappa^{K_Y}\partial Z_\kappa^{K_Z}}, \tag{1}$$

where the three-dimensional multi-index $\boldsymbol{K} = (K_X, K_Y, K_Z)^T$ is a vector of non-negtive integers and $|\boldsymbol{K}| = K_X + K_Y + K_Z$ is the norm (length) of the multi-index $\boldsymbol{K}$. More details on the multi-index notation could be found in Refs. [3, 4].

Therefore, the contracted rotational London atomic orbitals (LAO) [1, 5] used in GEN1INT could be written as

$$\begin{aligned}\omega_\kappa(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J}) &= \exp\left[-\tfrac{\mathrm{i}}{2}\boldsymbol{B}\cdot(\boldsymbol{R}_\kappa-\boldsymbol{G})\times\boldsymbol{r}_P + \mathrm{i}\mathbf{I}^{-1}\boldsymbol{J}\cdot(\boldsymbol{R}_\kappa-\boldsymbol{O})\times\boldsymbol{r}_P\right]\chi_\kappa(\boldsymbol{r}) \\ &= \exp\left[-\tfrac{\mathrm{i}}{2}\boldsymbol{B}\cdot\boldsymbol{R}_{\kappa G}\times\boldsymbol{r}_P + \mathrm{i}\boldsymbol{J}\cdot\mathbf{I}^{-T}(\boldsymbol{R}_{\kappa O}\times\boldsymbol{r}_P)\right]\chi_\kappa(\boldsymbol{r}),\end{aligned} \tag{2}$$

where $\boldsymbol{B}$ and $\boldsymbol{J}$ denote the external magnetic field and total rotational angular momentum, respectively. $\boldsymbol{G}$ is the gauge origin of the magnetic vector potential, $\boldsymbol{P}$ is the origin of the London phase factor, $\boldsymbol{O}$ is the center of mass of the system, and $\mathbf{I}^{-T}$ the transpose of the inverse of the inertia tensor $\mathbf{I}$. $\chi_\kappa(\boldsymbol{r})$ is the atomic orbital (AO) located at nucleus $\boldsymbol{R}_\kappa$

$$\chi_\kappa(\boldsymbol{r}) = \theta_\kappa(\boldsymbol{r}_\kappa)\rho_\kappa(r_\kappa), \tag{3}$$

where $\rho_\kappa(r_\kappa)$ is a contracted Gaussian[*]

$$\rho_\kappa(r_\kappa) = \sum_i w_{i\kappa}\exp(-a_{i\kappa}r_\kappa^2), \tag{4}$$

with $w_{i\kappa}$ and $a_{i\kappa}$ being the radial contraction coefficients (normalization constant included) and orbital exponents, respectively. The angular part $\theta_\kappa(\boldsymbol{r}_\kappa)$ of the AO is either a real solid-harmonic function $S_{l_\kappa m_\kappa}(\boldsymbol{r}_\kappa)$ or Cartesian Gaussian

$$G_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r}) = \boldsymbol{r}_\kappa^{\boldsymbol{l}_\kappa}\exp(-a_{i\kappa}r_\kappa^2), \tag{5}$$

which obey the following transformation [6]

$$S_{l_\kappa m_\kappa}(\boldsymbol{r}_\kappa)\sum_i w_{i\kappa}\exp\left(-a_{i\kappa}r_\kappa^2\right) = \sum_{|\boldsymbol{l}_\kappa|=l_\kappa} S_{\boldsymbol{l}_\kappa}^{l_\kappa m_\kappa}\sum_i w_{i\kappa}G_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r}). \tag{6}$$

[*]Each individual $\exp(-a_{i\kappa}r_\kappa^2)$ is named as primitive Gaussian.

Preface

Gen1Int is a Fortran 90 library (with Python interface) to evaluate the derivatives of one-electron integrals with respect to the geometry perturbation, external electric and magnetic fields, and total rotational angular momentum

1. at zero fields (for instance $\boldsymbol{B} = \boldsymbol{0}$ and $\boldsymbol{J} = \boldsymbol{0}$), and
2. using the contracted rotational London atomic orbitals (LAO) as in Eq. (2).

More explicitly, what we evaluate is

$$\prod^{N_g} \boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g} \left\{ \boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}-\boldsymbol{K}_0} \boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}-\boldsymbol{L}_0} \int \boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa} \left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1} \boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_1} \omega_\kappa^*(\boldsymbol{r}; \boldsymbol{B}, \boldsymbol{J}) \right] \hat{O}_{\ell_\beta}^{\boldsymbol{K}_0 \boldsymbol{L}_0} \boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda} \left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2} \boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_2} \omega_\lambda(\boldsymbol{r}; \boldsymbol{B}, \boldsymbol{J}) \right] \mathrm{d}\boldsymbol{r} \right\}_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}}, \tag{7}$$

where we have introduced the following generalized one-electron operator [1, 3]

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0 \boldsymbol{L}_0} = \begin{pmatrix} \boldsymbol{K} \\ \boldsymbol{K}_0 \end{pmatrix} \begin{pmatrix} \boldsymbol{L} \\ \boldsymbol{L}_0 \end{pmatrix} \left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_0} \boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_0} \hat{O}_{\ell_\beta} \left(\{\boldsymbol{r}_{C_\alpha}\}, \boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}; \boldsymbol{B}, \boldsymbol{J} \right) \right]_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}}. \tag{8}$$

The quantities $\boldsymbol{L}_\kappa$ and $\boldsymbol{L}_\lambda$, $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$, and $\boldsymbol{L}_1$ and $\boldsymbol{L}_2$ in Eq. (7) are the partial derivatives respectively on bra and ket, with respect to the geometry perturbation, external magnetic field, and total rotational angular momentum.

Notice that the number of centers in operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0 \boldsymbol{L}_0}$ usually satisfies $N_\alpha \leq 2$, so that the number of differentiated centers in the total geometric derivatives $\prod^{N_g} \boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$ should satisfy $N_g \leq 4$. The evaluation of geometric derivatives could be found in Section 4.3 and Ref. [1], while those of magnetic and total rotational angular momentum derivatives are in Section 4.4 and Refs. [1, 3].

In current version of Gen1Int, we have implemented the integral evaluation of the following different forms of operator

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0 \boldsymbol{L}_0} = \bar{C} \hat{f} \left(\{\boldsymbol{r}_{C_\alpha}\} \right) \boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}, \tag{9}$$

where

$$\hat{f}\left(\{\boldsymbol{r}_{C_\alpha}\}\right) = \begin{cases} \left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M} \boldsymbol{r}_M^{\boldsymbol{m}} \right) \left(\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} r_C^{-m_0} \right), & (m_0 = 1, 2), \\ \left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M} \boldsymbol{r}_M^{\boldsymbol{m}} \right) \left[\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} \delta(\boldsymbol{r}_C) \right], & \\ \boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M} \boldsymbol{r}_M^{\boldsymbol{m}}, & \\ \left(\boldsymbol{\partial}_{\boldsymbol{C}_1}^{\boldsymbol{L}_{C_1}} r_{C_1}^{-1} \right) \left(\boldsymbol{\partial}_{\boldsymbol{C}_2}^{\boldsymbol{L}_{C_2}} r_{C_2}^{-1} \right), & \\ \left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M} \boldsymbol{r}_M^{\boldsymbol{m}} \right) \left[\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} \frac{\mathrm{erf}\left(\sqrt{\varrho} r_C\right)}{r_C} \right], & \\ \text{Effective core potential}, & \\ \text{Model core potential (Version 1)}. & \end{cases} \tag{10}$$

The recurrence relations of evaluating these operators have been developed in Refs. [1–3]. In Section 4.6, we focus on the implementation of these recurrence relations from the point view of programmers.

It is well-known that the real solid-harmonic functions $S_{l_\kappa m_\kappa}(\boldsymbol{r}_\kappa)$[†] are not separable in the Cartesian directions, the integrals are therefore evaluated either using the separable Cartesian Gaussians or the following Hermite Gaussians

$$H_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r}) = (2a_{i\kappa})^{-|\boldsymbol{l}_\kappa|}\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}\exp(-a_{i\kappa}r_\kappa^2), \tag{11}$$

from which the real solid-harmonic Gaussians are obtained as [6]

$$S_{l_\kappa m_\kappa}(\boldsymbol{r}_\kappa)\sum_i w_{i\kappa}\exp\left(-a_{i\kappa}r_\kappa^2\right) = \sum_{|\boldsymbol{l}_\kappa|=l_\kappa} S_{\boldsymbol{l}_\kappa}^{l_\kappa m_\kappa}\sum_i w_{i\kappa}H_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r}), \tag{12}$$

with the same expansion coefficients $S_{\boldsymbol{l}_\kappa}^{l_\kappa m_\kappa}$ as in Eq. (6) [7]. Therefore, the integral evaluation in GEN1INT with real solid-harmonic Gaussians is first performed using either contracted Cartesian or Hermite Gaussians followed by the transformation (6) or (12). The implementation of this transformation is described in Section 4.8.3. In Section 4.8.5, we describe the implementation of transformation between Cartesian and Hermite Gausians, which is used when recovering the orbital quantum numbers of primitive Cartesian Gaussians from primitive Hermite Gaussians.

Another important issue related to basis sets is the normalization of contracted spherical and Cartesian Gaussians. Usually, this should be performed outside GEN1INT. However, we have implemented such functionalities as described in Sections 4.8.1 and 4.8.2.

Last but not least, most integrals need the evaluation of auxiliary functions, such as Boys function. We have addressed this problem in Section 4.9. Moreover, the evaluation of these functions may also affect the accuracy and stability of GEN1INT, and results some limitations as described in Chapter 7.

To sum up, the following chapters are recommended for basic usage of GEN1INT:

1. Chapter 1 “Installation”,

2. Chapter 2 “Python Interface”,

3. Chapter 4.3.1 “Sequence of Total Geometric Derivatives”,

4. Chapter 6.1 “Header Files in GEN1INT”, and

5. Chapter 7 “Limitations”.

In particular, we have described the results you get from GEN1INT in Section 2.2, such as the order of basis sets, operators and derivatives, which are better and necessary to know ;-)

Other sections in Chapter 4 “Framework of GEN1INT” and Chapter 3 “Use GEN1INT in Your Code” are more advanced topics, which might be suitable for those who want to use GEN1INT in their own code, or who would like to contribute to GEN1INT. All the GEN1INT subroutines are listed in Section 5. In Section 3.3, we describe one possible solution to use GEN1INT in parallel. However, such try is just an example, the parallelization is obviously performed outside GEN1INT, and requires the consideration of advanced users. A special topic of “mixed Spherical and Cartesian Gaussians” is described in Section 3.4 which might be useful in some case. If you are finally interested in the files and directories of GEN1INT, please refer to Chapter 6.

Finally, as regards details of the theoretical background, we refer to Refs. [1–3]. Enjoy ;-)

[†]In the following chapters, we often use the term “spherical Gaussian” for $\chi_\kappa(\boldsymbol{r}) = S_{l_\kappa m_\kappa}(\boldsymbol{r}_\kappa)\sum_i w_{i\kappa}\exp\left(-a_{i\kappa}r_\kappa^2\right)$ instead of “real solid-harmonic Gaussian”.

Acknowledgments

When Professor Kenneth Ruud provides us this project, we did not think it is such a big one (or, maybe we are not so efficient ;-)). It took us around one year to have a workable version only with geometric derivatives and few forms of operator. What is worse, it is not a standalone library and not so easy to implement in other quantum-chemistry codes except for DALTON (we start this project in DALTON ;-)).

After another year of work, especially after the DALTON meeting in Oslo, Jan. 11-12, 2010, we finally fixed the framework of GEN1INT: writing in Fortran 90 language with Python interface. So that all the object-oriented stuff is taken care by Python. We also avoided using advanced data types in Fortran 90 like type and pointer, only the allocatable array is used comparing with Fortran 77. This makes the wrapper work using Python or other languages be quite easy.

It is worthy of mentioning that the GEN1INT project can not be finished without the help and discussions from other persons. Firstly, we would like to express our great thankfulness to Professor Kenneth Ruud. Without the opportunity providing by him, without his helpful discussions and suggestions, and without his great patience, this work could not even be presented. Tusen takk ;-)

We also wish to thank another CTCC leader, Professor Trygve Helgaker, who has always provided insightful suggestions and discussions, and helped us prepare and polish the manuscripts related to GEN1INT.

During writing GEN1INT and preparing related scientific publications, we also received great help and discussions from, including but not exclusively, Dr. Radovan Bast (many things), Dr. Michal Repisky (especially, the discussion of recurrence relations), and Dr. Stefano Borini (the help of using Q5Cost in GEN1INT Version 0.1.0). We also appreciate the nice work environment in CTCC, the computational resource on Stallo, and more ..., and of course, your choice and use of GEN1INT, and contributions ... ;-)

Chapter 1

Installation

Before installing Gen1Int, you need to make sure the following programs are installed on your computer:

1. Git (for developers),
2. CMake or GNU Autotools (for generating Fortran library),
3. Fortran 90 compiler,
4. Python and NumPy (for Python interface).

The latest version of Gen1Int could be found at `http://repo.ctcc.no/projects/gen1int`. After you get a workable version, you may first need to check or modify the following header files for your own case[*]

1. `src/stdout.h` defines the IO of standard output, default is 6;
2. `src/xkind.h` defines the kind type parameter of real numbers, default is real(8);
3. `src/max_gen_order.h`, `src/boys_power.h` and `src/tab_boys.h`: generated by `tools/GenHeader.py`, contains respectively the maximum order (default is 50), minimum and maximum arguments, step size and number of steps, values of pretabulated Boys functions. These files will be used in `src/aux_boys_vec.F90` to improve the efficiency. The Boys functions will be calculated in run-time by subroutines in `src/aux_boys_vec.F90` if given argument and order is not found in the table.

Afterwards, you could start to compile Gen1Int.

1.1 CMake

For CMake users, let us assume that you want to compile the library in directory "`build`", which might be performed by the following steps:

```
mkdir build
cd build
ccmake ..
make
```

[*]For instance, you may run `tools/GenHeader.py` with your required maximum order and replace the header files `src/max_gen_order.h`, `src/boys_power.h` and `src/tab_boys.h`.

In the step "`ccmake ..`", you may need to change "`CMAKE_BUILD_TYPE`" (default is "`RelWithDebInfo`"), and other options as you may would like. As regards `CMAKE_BUILD_TYPE`, we would recommend you to change it as "`Release`", otherwise you may get huge dumping information which may only be useful for debugging. You could also add the option "`-DXTIME`" for compiler, so that GEN1INT will print the CPU elapsed time during the calculations. However this will also produce extremely large information, which might not be useful for ordinary use of GEN1INT.

You could also use

```
cmake -DCMAKE_BUILD_TYPE=Release ..
```

instead of "`ccmake ..`". You may also try

```
cmake -DDISABLE_F90_MODULE=1 ..
```

so that the Fortran 90 module of GEN1INT in `src/gen1int.F90` will not be compiled.

During the step of "`make`", you could also try "`make VERBOSE=1`" to get more information during compiling. If everything is OK, you will get the library named as "`libgen1int.a`" and an executable code "`test_gen1int`". We strongly recommend you run this code for test suite (more details about test suite could be found in Section 4.10), and make sure there is no error. Otherwise, please write to us (see the contact information in "`AUTHORS`") with the test log file `test_gen1int.html`, thank you!

1.2 GNU Autotools

We are lazy to provide you a fantastic "`configure`" file ;-), so please first check "`configure.ac`" to see if it needs any modification for your requirement. The procedure of compiling using GNU autotools might be

```
aclocal
autoheader
automake --add-missing
autoconf
./configure --with-debug --with-time --enable-fmodule
make
```

where the meaning of "`--with-debug --with-time`" can be found in Section 1.1. While "`--enable-fmodule`" will compile the Fortran 90 module of GEN1INT in `src/gen1int.F90`.

Similar to the case of CMake, the library is named as "`libgen1int.a`" and you will get an executable code "`test_gen1int`". Again, we strongly recommend you run this code for test suite (more details about test suite could be found in Section 4.10), and make sure there is no error. Otherwise, please write to us (see the contact information in "`AUTHORS`") with the test log file `test_gen1int.html`.

1.3 Python

We use `f2py` to wrapper GEN1INT. However, as far as our knowledges concerns, `f2py` does not do preprocess source codes. We therefore provide several simple functions in `setup.py` to perform such preprocessing in GEN1INT, and generate source codes named as "`src/py_xxxx.F90`" for `f2py`.

Additionally, there is a dictionary variable "`def_opts`" in `setup.py` which controls the compiling options

```
def_opts = {'DEBUG':False,'XTIME':False}
```

where the meaning can be found in Section 1.1, and you may modify it according to your case.

To summarize, what you may use is the following one-line command to install GEN1INT in Python

```
python setup.py install
```

or

```
python setup.py install --home=path_install
```

to install GEN1INT in directory "`path_install`".

Describe the test suite (in directory `test_py`) ...

1.4 Compiler and Flags

Some compilers with specific flags may not work correctly. For instance, we have problem to compile `src/basic/hgto_to_cgto.F90` using "ifort (IFORT) 11.1 20090511" on Stallo (`http://www.notur.no/hardware/stallo/`) with flags `-g` and `-O3` together. Changing the compiler, removing flag `-g`, or using flag `-O1` has solved the problem.

Chapter 2

Python Interface of Gen1Int

2.1 "Hello World" in Python

Describe the "Hello World" in Python, such as ...

```
import Gen1Int.ContrInt
import Gen1Int.Tools
```

2.2 What You Get from Gen1Int

The results obtained from Gen1Int is always a five-dimensional array containing the contracted integrals

```
contr_ints(num_gto_bra,num_contr_bra,num_gto_ket,num_contr_ket,num_opt)
```

where the first dimension either contains the angular parts of spherical Gaussians, or xyz powers of Cartesian Gaussians on bra center. The second dimension contains the sub-shells with the same azimuthal quantum number (but different principal quantum numbers) on bra center. Taking p sub-shells as an example, the second dimension is arranged in an ascending order according to the principal quantum numbers, such as $(2p, 3p, \ldots, 6p)$. The third and fourth dimensions contains the angular parts or xyz powers, sub-shells on ket center, which are arranged in the same way as those on bra center.

The angular parts of spherical Gaussians in `contr_ints` are arranged according to their magnetic quantum numbers, from $-l$ to $+l$ (l is the azimuthal quantum number), i.e., in an ascending order. As regards the xyz powers of Cartesian Gaussians, we use an ascending zy-major order in Gen1Int. For instance, the Cartesian Gaussians representing f sub-shell are arranged as[*]

$$\begin{array}{ccccccc} xxx & & xxy & & xyy & & yyy \\ & xxz & & xyz & & yyz & \\ & & xzz & & yzz & & \\ & & & zzz & & & \end{array}$$

which could be generated by the following loops in Python

```
for z in xrange(l+1):
    for y in xrange(l+1-z):
        return [l-y-z,y,z]
```

[*]The reason of this ordering is ...

Last, the fifth dimension `num_opt` in the contracted integrals contains all the xyz components of[†]

1. electronic derivatives ($\boldsymbol{\partial}_{\boldsymbol{n}}^{\boldsymbol{r}}$),
2. Cartesian multipole moment ($\boldsymbol{r}_M^{\boldsymbol{m}}$),
3. partial and total derivatives with respect to magnetic field ($\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1}$, $\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2}$ and $\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}}$),
4. partial and total derivatives with respect to total rotational angular momentum ($\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_1}$, $\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_2}$ and $\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}}$),
5. partial geometric derivatives with respect to centers on bra, ket and operator ($\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}$, $\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}$ and $\boldsymbol{\partial}_{\boldsymbol{C}_\alpha}^{\boldsymbol{L}_\alpha}$), and
6. total geometric derivatives ($\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$).

Therefore, the fifth dimension is arranged in the order of

```
num_elec, num_mom,
num_mag_bra, num_mag_ket, num_mag_total,
num_ram_bra, num_ram_ket, num_ram_total,
num_geo_bra, num_geo_ket, num_geo_opt, num_geo_total,
```

where `num_ram_xxx` is the number of xyz components of derivatives with respect to total rotational angular momentum (RAM). All the xyz components of the aforementioned operators and derivatives are arranged using the ascending zy-major order as that of xyz powers of Cartesian Gaussians.

As regards the total geometric derivatives ($\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$), the xyz components of the first differentiated center is the most consecutive part, followed by the second, third, ..., and the last differentiated center.

2.3 Tools in Gen1Int

As discussed in previous section, the angular parts of spherical Gaussians and xyz powers of Cartesian Gaussians in Gen1Int are arranged in order. If the order of your basis functions are different from them, you could reorder the contracted integrals by using the functions defined in `Gen1Int.Tools`, as shown in Table 2.1.

Table 2.1: Functions of reordering integrals in `Gen1Int.Tools`.

`reorder_sgtos`	Reorders the contracted real solid-harmonic Gaussians on bra or ket center.		
	In	`ang_ket`	orbital quantum number (or angular number) of ket center
		`num_sgto_ket`	number of basis functions on ket center (equals to 2`ang_ket`+1)
		`mag_ket`	magnetic numbers of basis functions on ket center

Continued on next page

[†]There is neither electronic derivatives nor Cartesian multipole moment for effective core potential and model core potential.

Table 2.1 – continued from previous page

		`dim_bra_sgto`	dimension of SGTOs on bra center
		`num_contr_ket`	number of contractions of ket center
		`num_opt`	number of operators
		`gen_ints`	contracted integrals from `Gen1Int.ContrInt`
	Out	`ro_ints`	reordered integrals according to given `mag_ket`
`reorder_sgto_ints`	Reorders the integrals of contracted real solid-harmonic Gaussians.		
	In	`ang_bra`	orbital quantum number (or angular number) of bra center
		`num_sgto_bra`	number of basis functions on bra center (equals to 2`ang_bra`+1)
		`mag_bra`	magnetic numbers of basis functions on bra center
		`ang_ket`	orbital quantum number (or angular number) of ket center
		`num_sgto_ket`	number of basis functions on ket center (equals to 2`ang_ket`+1)
		`mag_ket`	magnetic numbers of basis functions on ket center
		`num_contr_bra`	number of contractions of bra center
		`num_contr_ket`	number of contractions of ket center
		`num_opt`	number of operators
		`gen_ints`	contracted integrals from `Gen1Int.ContrInt`
	Out	`ro_ints`	reordered integrals according to given `mag_bra` and `mag_ket`
`reorder_cgtos`	Reorders the integrals of contracted Cartesian Gaussians on bra or ket center.		
	In	`ang_ket`	orbital quantum number (or angular number) of ket center
		`num_cgto_ket`	number of basis functions on ket center (equals to (`ang_ket`+1)(`ang_ket`+2)/2)
		`power_ket`	Cartesian powers of basis functions on ket center
		`dim_bra_cgto`	dimension of CGTOs on bra center
		`num_contr_ket`	number of contractions of ket center
		`num_opt`	number of operators
		`gen_ints`	contracted integrals from `Gen1Int.ContrInt`
	Out	`ro_ints`	reordered integrals according to given `power_ket`
`reorder_cgto_ints`	Reorders the integrals of contracted Cartesian Gaussians.		
	In	`ang_bra`	orbital quantum number (or angular number) of bra center
		`num_cgto_bra`	number of basis functions on bra center (equals to (`ang_bra`+1)(`ang_bra`+2)/2)
		`power_bra`	Cartesian powers of basis functions on bra center

Continued on next page

Table 2.1 – continued from previous page

		`ang_ket`	orbital quantum number (or angular number) of ket center
		`num_cgto_ket`	number of basis functions on ket center (equals to (`ang_ket`+1)(`ang_ket`+2)/2)
		`power_ket`	Cartesian powers of basis functions on ket center
		`num_contr_bra`	number of contractions of bra center
		`num_contr_ket`	number of contractions of ket center
		`num_opt`	number of operators
		`gen_ints`	contracted integrals from `Gen1Int.ContrInt`
	Out	`ro_ints`	reordered integrals according to given `power_bra` and `power_ket`

The Fortran 90 source codes of these reordering subroutines are in file `reorder_ints.F90`, which could be called by users in their own programs.

Describe other functions in `Gen1Int.Tools` ...

2.4 Pre-defined Property Integrals in Python Interface

To further facilitate the use of Gen1Int, we have implemented ?? property integrals in file `Gen1Int/PropInt.py`, which could be used by

```
import Gen1Int.PropInt
```

All the functions defined in `Gen1Int/PropInt.py` are given in Table 2.2 with detailed descriptions (***this table needs to be rewritten, sorry***) ...

The total electron density $\rho(\boldsymbol{r})$ can be written as

$$\rho(\boldsymbol{r}) = \sum_{\kappa\lambda} D_{\kappa\lambda}\chi_{\kappa}(\boldsymbol{r})\chi_{\lambda}(\boldsymbol{r}), \tag{2.1}$$

Table 2.2: Implemented one-electron property integrals in GEN1INT.

Keyword	Integrals	Labels	Description
`*1ELPOT`	$-\sum_K \left\langle \chi_\kappa \left\| \frac{Z_K}{r_K} \right\| \chi_\lambda \right\rangle$	`POTENERG`[‡]	One-electron potential energy integrals.
`*ANGLON`	$\langle \chi_\kappa \| \mathbf{L}_N \| \chi_\lambda \rangle$	`XANGLON_` `YANGLON_` `ZANGLON_`	Contribution to the one-electron contribution of the magnetic moment using London orbitals arising from the differentiation of London-orbital transformed Hamiltonian, see Ref. [8].
`*ANGMOM`	$\langle \chi_\kappa \| \mathbf{L}_O \| \chi_\lambda \rangle$	`XANGMOM_` `YANGMOM_` `ZANGMOM_`	Angular momentum around the molecular origin. This can be adjusted by changing the gauge origin through the use of the `.GAUGEO` keyword.
`*CARMOM`	$\left\langle \chi_\kappa \left\| x^i y^j z^k \right\| \chi_\lambda \right\rangle$	`CMiijjkk`	Cartesian multipole integrals, whose order is determined by the keyword `.IORCAR`. Labels `ii`, `jj`, and `kk` are determined by, such as `ii` $= \left(\frac{i}{10}\right) \times 10 + \mathrm{mod}\,(i, 10)$.
`*DARWIN`	$\frac{\pi\alpha^2}{2} \sum_K \langle \chi_\kappa \| \delta(\mathbf{r}_K) \| \chi_\lambda \rangle$	`DARWIN__`	One-electron Darwin integrals [9].
`*DIPLEN`	$\langle \chi_\kappa \| \mathbf{r} \| \chi_\lambda \rangle$	`XDIPLEN_` `YDIPLEN_` `ZDIPLEN_`	Dipole length integrals.
`*DIPVEL`	$\langle \chi_\kappa \| \nabla \| \chi_\lambda \rangle$	`XDIPVEL_` `YDIPVEL_` `ZDIPVEL_`	Dipole velocity integrals.
`*DPTOVL`	$\left\langle \chi_\kappa \left\| \frac{\partial^2}{\partial \mathbf{r}^2} \right\| \chi_\lambda \right\rangle$	`dd/dxdx_` `dd/dxdy_` `dd/dxdz_` `dd/dydy_` `dd/dydz_` `dd/dzdz_`	DPT (Direct Perturbation Theory) integrals: Small-component one-electron overlap integrals.

Continued on next page

[‡] `_S` indicates the integral matrices are symmetric, `_A` for antisymmetric, while `_N` for non-symmetric.

Table 2.2 – continued from previous page

Keyword	Integrals	Labels	Description
*DSUSLH	$\frac{1}{4}Q_{MN}\left\langle\chi_\kappa\left\|\mathbf{r}\mathbf{r}^T h\right\|\chi_\lambda\right\rangle Q_{MN}$	XXDSUSLH XYDSUSLH XZDSUSLH YYDSUSLH YZDSUSLH ZZDSUSLH	The contribution to diamagnetic magnetizability integrals from the differentiation of the London orbital phase-factors, see Ref. [8].
*DSUSNL	$\frac{1}{4}\left\langle\chi_\kappa\left\|r_N^2\mathbf{I}_{3\times3}-\mathbf{r}_N\mathbf{r}_N^T\right\|\chi_\lambda\right\rangle$	XXDSUSNL XYDSUSNL XZDSUSNL YYDSUSNL YZDSUSNL ZZDSUSNL	The contribution to the diamagnetic magnetizability integrals using London orbitals but with contributions from the differentiation of the Hamiltonian only, see Ref. [8].
*EFGCAR	$\left\langle\chi_\kappa\left\|\frac{3\mathbf{r}_K\mathbf{r}_K^T-\mathbf{r}_K^T\mathbf{r}_K\mathbf{I}_{3\times3}}{r_K^5}\right\|\chi_\lambda\right\rangle$	XYEFGabc	Cartesian electric field gradient integrals. Where X and Y are the Cartesian directions, abc the number of the symmetry independent center, and c that centers c'th symmetry-generated atom.
*FC	$\frac{4\pi g_e}{3}\left\langle\chi_\kappa\left\|\delta\left(\mathbf{r}_K\right)\right\|\chi_\lambda\right\rangle$[§]	FC_NAMab	Fermi-contact integrals, see Ref. [10]. Where NAM is the three first letters in the name of this atom, as given in the MOLECULE.INP file, and ab is the number of the symmetry-adapted nucleus.
*KINENE	$-\frac{1}{2}\left\langle\chi_\kappa\left\|\nabla^2\right\|\chi_\lambda\right\rangle$	KINENERG	Kinetic energy integrals.
*LONMOM	$\frac{1}{2}Q_{MN}\left\langle\chi_\kappa\left\|\mathbf{r}h\right\|\chi_\lambda\right\rangle$[¶]	XLONMOM_ YLONMOM_ ZLONMOM_	Contribution to the London magnetic moment from the differentiation with respect to magnetic field on the London orbital phase factors, see Ref. [8].
*MAGMOM	$\frac{1}{2}\left\langle\chi_\kappa\left\|\mathbf{L}_N+Q_{MN}\mathbf{r}h\right\|\chi_\lambda\right\rangle$	dh/dBX__ dh/dBY__ dh/dBZ__	One-electron contribution to the magnetic moment around the nuclei to which the atomic orbitals are attached. This is the London atomic orbital magnetic moment as defined in Eq. (35) of Ref. [11]. The integral is calculated as the sum of *LONMOM and *ANGLON.

Continued on next page

[§] K is the nucleus of interest.

[¶] Antisymmetric matrix $Q_{MN} = \begin{bmatrix} 0 & -Z_{MN} & Y_{MN} \\ Z_{MN} & 0 & -X_{MN} \\ -Y_{MN} & X_{MN} & 0 \end{bmatrix}$, while h is the one-electron Hamiltonian in absence of magnetic field (see *ONEHAMIL).

Table 2.2 – continued from previous page

Keyword	Integrals	Labels	Description
`*MASSVE`	$\frac{\alpha^2}{8}\left\langle\chi_\kappa\left\lvert\nabla^2\cdot\nabla^2\right\rvert\chi_\lambda\right\rangle$	`MASSVELO`	Mass-velocity integrals.
`*NELFLD`	$\left\langle\chi_\kappa\left\lvert\frac{\mathbf{r}_K}{r_K^3}\right\rvert\chi_\lambda\right\rangle^\dagger$	`NEF_abc_`	Nuclear electric field integrals. Where `abc` is the number of the symmetry-adapted nuclear coordinate.
`*NST`	$\frac{1}{2}\left\langle\chi_\kappa\left\lvert\frac{\mathbf{r}_N^T\mathbf{r}_K\mathbf{I}_{3\times3}-\mathbf{r}_N\mathbf{r}_K^T}{r_K^3}+Q_{MN}\frac{\mathbf{r}\mathbf{L}_K^T}{r_K^3}\right\rvert\chi_\lambda\right\rangle^\dagger$	`abc_NSTd`	Calculate the one-electron contribution to the diamagnetic nuclear shielding tensor integrals using London atomic orbitals, see Ref. [8]. It is calculated as the sum of `NSTLON` and `NSTNOL`. Where `abc` is the number of the symmetry-adapted nuclear magnetic moment coordinate, and `d` refers to the x, y, or z component of the magnetic field.
`*NSTCGO`	$\frac{1}{2}\left\langle\chi_\kappa\left\lvert\frac{\mathbf{r}_O^T\mathbf{r}_K\mathbf{I}_{3\times3}-\mathbf{r}_O\mathbf{r}_K^T}{r_K^3}\right\rvert\chi_\lambda\right\rangle^\dagger$	`abcNSCOd`	Calculate the diamagnetic nuclear shielding tensor integrals without using London atomic orbitals. Note that the gauge origin is controlled by the keyword `.GAUGEO`. Where `abc` is the number of the symmetry-adapted nuclear magnetic moment coordinate, and `d` refers to the x, y, or z component of the magnetic field. O is the gauge origin.
`*NSTLON`	$\frac{1}{2}Q_{MN}\left\langle\chi_\kappa\left\lvert\frac{\mathbf{r}\mathbf{L}_K^T}{r_K^3}\right\rvert\chi_\lambda\right\rangle^\dagger$	`abcNSLOd`	Calculate the contribution to the London orbital nuclear shielding tensor from the differentiation of the London orbital phase-factors, see Ref. [8]. Where `abc` is the number of the symmetry-adapted nuclear magnetic moment coordinate, and `d` refers to the x, y, or z component of the magnetic field.
`*NSTNOL`	$\frac{1}{2}\left\langle\chi_\kappa\left\lvert\frac{\mathbf{r}_N^T\mathbf{r}_K\mathbf{I}_{3\times3}-\mathbf{r}_N\mathbf{r}_K^T}{r_K^3}\right\rvert\chi_\lambda\right\rangle^\dagger$	`abcNSNLd`	Calculate the contribution to the nuclear shielding tensor when using London atomic orbitals from the differentiation of the Hamiltonian alone, see Ref. [8]. Where `abc` is the number of the symmetry-adapted nuclear magnetic moment coordinate, and `d` refers to the x, y, or z component of the magnetic field.

Continued on next page

Table 2.2 – continued from previous page

Keyword	Integrals	Labels	Description
*NUCPOT	$\left\langle \chi_\kappa \middle\| \frac{1}{r_K} \middle\| \chi_\lambda \right\rangle^\dagger$	POT.E_ab	Calculate the nuclear potential energy. Currently this keyword can only be used in calculations not employing symmetry. Where ab are the two first letters in the name of this nucleus. Thus note that in order to distinguish between integrals, the first two letters in an atom's name must be unique.
*ONEHAMIL	$-\left\langle \chi_\kappa \middle\| \sum_K \frac{Z_K}{r_K} + \frac{1}{2}\nabla^2 \middle\| \chi_\lambda \right\rangle$	ONEHAMIL	One-electron Hamiltonian.
*OVERLAP	$\langle \chi_\kappa \| \chi_\lambda \rangle$	OVERLAP_	Overlap integrals.
*PSO	$\left\langle \chi_\kappa \middle\| \frac{\mathbf{L}_K}{r_K^3} \middle\| \chi_\lambda \right\rangle^\dagger$	PSO_abc_	Paramagnetic spin-orbit integrals, see Ref. [10]. Where abc is the number of the symmetry-adapted nuclear magnetic moment coordinate.
*S1MAG	$\frac{1}{2}Q_{MN} \langle \chi_\kappa \|\mathbf{r}\| \chi_\lambda \rangle^\ddagger$	dS/dBX__ dS/dBY__ dS/dBZ__	Calculate the first derivative overlap matrix with respect to an external magnetic field by differentiation of the London phase factors, see Ref. [8].
*SECMOM	$\left\langle \chi_\kappa \middle\| \mathbf{r}\mathbf{r}^T \middle\| \chi_\lambda \right\rangle$	XXSECMOM XYSECMOM XZSECMOM YYSECMOM YZSECMOM ZZSECMOM	Second-moment integrals.
*SQHDOL	$\left\langle \frac{\partial \chi_\kappa}{\partial R_{ab}} \middle\| \chi_\lambda \right\rangle$	SQHDLabc	Square, non-symmetrized half differentiated overlap integrals with respect to geometric distortions, see Ref. [12]. Differentiation on the bra-vector. Where abc is the number of the symmetry-adapted coordinate being differentiated.
*SQHDOR	$\left\langle \chi_\kappa \middle\| \frac{\partial \chi_\lambda}{\partial R_{ab}} \right\rangle$	SQHDRabc	Square, non-symmetrized half-differentiated overlap integrals with respect to geometric distortions, see Ref. [12]. Differentiation on the ket-vector. Where abc is the number of the symmetry-adapted coordinate being differentiated.

Continued on next page

Table 2.2 – continued from previous page

Keyword	Integrals	Labels	Description
*THETA	$\frac{1}{2}\left\langle \chi_\kappa \left\| 3\mathbf{r}\mathbf{r}^T - r^2\mathbf{I}_{3\times 3} \right\| \chi_\lambda \right\rangle$	XXTHETA_	Traceless quadrupole moment integrals as defined by Buckingham [13].
		XYTHETA_	
		XZTHETA_	
		YYTHETA_	
		YZTHETA_	
		ZZTHETA_	
*THIRDM	$\left\langle \chi_\kappa \left\| \mathbf{r}^3 \right\| \chi_\lambda \right\rangle$	XXX_3MOM	Third-moment integrals.
		XXY_3MOM	
		XXZ_3MOM	
		XYY_3MOM	
		XYZ_3MOM	
		XZZ_3MOM	
		YYY_3MOM	
		YYZ_3MOM	
		YZZ_3MOM	
		ZZZ_3MOM	
*2NDMM	$\left\langle \chi_\kappa \left\| \mathbf{r}\mathbf{p}^T + \mathbf{p}\mathbf{r}^T \right\| \chi_\lambda \right\rangle$	XX2NDMM_	Second-moment integrals in momentum space.
		XY2NDMM_	
		XZ2NDMM_	
		YY2NDMM_	
		YZ2NDMM_	
		ZZ2NDMM_	
*3RDMM	$\left\langle \chi_\kappa \left\| \mathbf{r}\mathbf{r}\mathbf{p} + \mathbf{r}\mathbf{p}\mathbf{r} + \mathbf{p}\mathbf{r}\mathbf{r} \right\| \chi_\lambda \right\rangle$	XXX3RDMM	Third-moment integrals in momentum space.
		XXY3RDMM	
		XXZ3RDMM	
		XYY3RDMM	
		XYZ3RDMM	
		XZZ3RDMM	
		YYY3RDMM	
		YYZ3RDMM	
		YZZ3RDMM	
		ZZZ3RDMM	

The following equations will be merged into Table 2.2 ...

**2NDMM*

$$
\begin{aligned}
&\textit{XX2NDMM\_\_A:} && x\mathrm{d}x + \mathrm{d}xx = 2x\mathrm{d}x + 1, && (2.2)\\
&\textit{XY2NDMM\_\_A:} && x\mathrm{d}y + \mathrm{d}yx, && (2.3)\\
&\textit{XZ2NDMM\_\_A:} && x\mathrm{d}z + \mathrm{d}xz, && (2.4)\\
&\textit{YY2NDMM\_\_A:} && y\mathrm{d}y + \mathrm{d}yy = 2y\mathrm{d}y + 1, && (2.5)\\
&\textit{YZ2NDMM\_\_A:} && y\mathrm{d}z + \mathrm{d}yz, && (2.6)\\
&\textit{ZZ2NDMM\_\_A:} && z\mathrm{d}z + \mathrm{d}zz = 2z\mathrm{d}z + 1. && (2.7)
\end{aligned}
$$

**3RDMM*

$$
\begin{aligned}
&\textit{XXX3RDMM\_A:} && x^2\mathrm{d}x + x\mathrm{d}xx + \mathrm{d}xx^2 = 3x^2\mathrm{d}x + 3x, && (2.8)\\
&\textit{XXY3RDMM\_A:} && x^2\mathrm{d}y + x\mathrm{d}xy + \mathrm{d}xxy = x^2\mathrm{d}y + 2xy\mathrm{d}x + y, && (2.9)\\
&\textit{XXZ3RDMM\_A:} && x^2\mathrm{d}z + x\mathrm{d}xz + \mathrm{d}xxz = x^2\mathrm{d}z + 2xz\mathrm{d}x + z, && (2.10)\\
&\textit{XYY3RDMM\_A:} && xy\mathrm{d}y + x\mathrm{d}yy + \mathrm{d}xy^2 = y^2\mathrm{d}x + 2xy\mathrm{d}y + x, && (2.11)\\
&\textit{XYZ3RDMM\_A:} && xy\mathrm{d}z + x\mathrm{d}yz + \mathrm{d}xyz = xy\mathrm{d}z + xz\mathrm{d}y + yz\mathrm{d}x, && (2.12)\\
&\textit{XZZ3RDMM\_A:} && xz\mathrm{d}z + x\mathrm{d}zz + \mathrm{d}xz^2 = z^2\mathrm{d}x + 2xz\mathrm{d}z + x, && (2.13)\\
&\textit{YYY3RDMM\_A:} && y^2\mathrm{d}y + y\mathrm{d}yy + \mathrm{d}yy^2 = 3y^2\mathrm{d}y + 3y, && (2.14)\\
&\textit{YYZ3RDMM\_A:} && y^2\mathrm{d}z + y\mathrm{d}yz + \mathrm{d}yyz = y^2\mathrm{d}z + 2yz\mathrm{d}y + z, && (2.15)\\
&\textit{YZZ3RDMM\_A:} && yz\mathrm{d}z + y\mathrm{d}zz + \mathrm{d}yz^2 = z^2\mathrm{d}y + 2yz\mathrm{d}z + y, && (2.16)\\
&\textit{ZZZ3RDMM\_A:} && z^2\mathrm{d}z + z\mathrm{d}zz + \mathrm{d}zz^2 = 3z^2\mathrm{d}z + 3z. && (2.17)
\end{aligned}
$$

2.5 Memory Usage in Gen1Int

Temporary Integrals

1. ***loops over different AO sub-shells***
2. ***loops over the xyz components of AO sub-shells***
3. ***less temporary memory used during recurrence relations, for efficiency both in CPU time and memory***

We are going to find the maximum of

$$\frac{(l+1-n)(l+2-n)(l+3-n)-(m-n)(m+1-n)(m+2-n)}{6}\frac{(n+1)(n+2)}{2}. \qquad (2.18)$$

Revision 49fcc2a0 ID 49fcc2a09b41f415428fe7218927efb1e6ac2371

Our experience has shown that Gen1Int may work incorrectly compiled with some compilers together with specific flags. For instance, the subroutines in `src/hgto_to_cgto.F90` and `src/hgto_to_lcgto.F90` do give wrong results when compiled on Stallo cluster (at University of Tromsø) using Intel Fortran compilers version 10.1, 11.0, 11.1 and 11.1.072 with optimization flag either `-O2` or `-O3`. They however work with flag `-O1`, and that is the reason why we set default optimization flag as `-O1` in `CMakeLists.txt`, `src/Makefile.am` and `test_f90/Makefile.am` — *fixed!*

Chapter 3

Fortran Interface of Gen1Int

3.1 “Hello World” in Fortran 90

In this section, we will give an typical use of Gen1Int in Fortran 90 code. Take the calculation of Cartesian multipole moment integrals using contracted Cartesian Gaussians as an example, users may need the following two subroutines by setting `num_cents=0` (no total geometric derivatives)

```
! calculates the Cartesian multipole moment integrals using
! contracted Cartesian Gaussians
call contr_cgto_carmom(idx_bra, coord_bra, angular_bra, num_prim_bra, &
                       exponent_bra, num_contr_bra, contr_coef_bra,   &
                       idx_ket, coord_ket, angular_ket, num_prim_ket, &
                       exponent_ket, num_contr_ket, contr_coef_ket,   &
                       order_geo_bra, order_geo_ket,                  &
                       num_cents, idx_cent, order_cent,               &
                       idx_diporg, dipole_origin, scal_const,         &
                       order_geo_mom, order_mom, order_elec,          &
                       num_cart_bra, num_cart_ket, num_opt, gen_ints)
! reorders the integrals of contracted Cartesian Gaussians
call reorder_cgto_ints(ang_bra, num_cgto_bra, power_bra, &
                       ang_ket, num_cgto_ket, power_ket, &
                       num_contr_bra, num_contr_ket,     &
                       num_opt, gen_ints, contr_ints)
```

The results are returned in[*]

```
contr_ints(num_cart_bra,num_contr_bra,num_cart_ket,num_contr_ket, &
           num_elec,num_mom,num_geo_bra,num_geo_ket,num_geo_mom)
```

As regards spherical Gaussians, the following subroutines will be used

```
contr_sgto_carmom(...)
reorder_sgto_ints(...)
```

Other property integrals could also be calculated in a similar way, the related subroutines for contracted integrals are given in Section 5.1.

[*]Indeed, what is returned from Gen1Int is always a five-dimensional array; the xyz-components of both operators and derivatives are in the fifth dimension.

The total geometric derivatives are controlled by the arguments `num_cents`, `idx_cent` and `order_cent`. For instance

```
num_cents = 1; idx_cent = (/1/); order_cent = (/3/)
```

gives one-center third order total geometric derivatives on atom `1`, while

```
num_cents = 2; idx_cent = (/1,2/); order_cent = (/3,4/)
```

gives two-center seventh order total geometric derivatives on atoms `1` (third order) and `2` (fourth order). If you would like to use the sequence of total geometric derivatives as described in Section 4.3.1, you may first need to call `geom_total_tree_init` to initialize the "full arithmetic N-ary tree"

```
call geom_total_tree_init(num_atoms, order_geo, max_num_cent, &
                          num_paths, visit_height, idx_node,  &
                          wt_node, idx_cent, order_cent, num_geo_cent)
```

You also get the first path (saved in `wt_node(order_geo)`, `idx_cent` and `order_cent`) from `geom_total_tree_init`, which could be used when calling subroutines calculating the contracted integrals. For instance, the total geometric derivatives of Cartesian multipole moment integrals

```
num_cents = wt_node(order_geo)
! calculates the Cartesian multipole moment integrals using
! contracted Cartesian Gaussians
call contr_cgto_carmom(idx_bra, coord_bra, angular_bra, num_prim_bra,  &
                       exponent_bra, num_contr_bra, contr_coef_bra,    &
                       idx_ket, coord_ket, angular_ket, num_prim_ket,  &
                       exponent_ket, num_contr_ket, contr_coef_ket,    &
                       order_geo_bra, order_geo_ket, num_cents,        &
                       idx_cent(1:num_cents), order_cent(1:num_cents), &
                       idx_diporg, dipole_origin, scal_const,          &
                       order_geo_mom, order_mom, order_elec,           &
                       num_cart_bra, num_cart_ket, num_opt, gen_ints)
! reorders the integrals of contracted Cartesian Gaussians
call reorder_cgto_ints(ang_bra, num_cgto_bra, power_bra, &
                       ang_ket, num_cgto_ket, power_ket, &
                       num_contr_bra, num_contr_ket,     &
                       num_opt, gen_ints, contr_ints)
```

The results are returned in

```
contr_ints(num_cart_bra,num_contr_bra,num_cart_ket,num_contr_ket, &
           num_elec,num_mom,num_geo_bra,num_geo_ket,num_geo_mom,  &
           num_geo_total)
```

Other total geometric derivatives could be obtained by calling subroutine `geom_total_tree_search` `num_paths`−1 times (`num_paths` is already got from subroutine `geom_total_tree_init`) as follows

```
do ipath = 2, num_paths
    call geom_total_tree_search(num_atoms, order_geo, max_num_cent, &
                                visit_height, idx_node, wt_node,    &
                                idx_cent, order_cent, num_geo_cent)
end do
```

Likewise, the information of total geometric derivatives is saved in `wt_node(order_geo)`, `idx_cent` and `order_cent` each time, which could be used to call the corresponding subroutine to calculate the contracted integrals.

Finally, if users are interested in derivatives with respect to the magnetic field and/or total rotational angular momentum at zero fields with (rotational) London atomic orbitals, the subroutines `lcgto_zero_xxxx` and `lsgto_zero_xxxx` should be used for Cartesian and spherical Gaussians, respectively. Here `xxxx` is the name of specific operator as shown in Section 5.1.

Still taking the Cartesian multipole moment integrals as an example, the mixed derivatives with respect to the geometry perturbation, external magnetic field and total rotational angular momentum at zero fields with (rotational) London Cartesian Gaussians could be obtained by

```
! calculates the Cartesian multipole moment integrals using
! contracted (rotational) London Cartesian Gaussians
call lcgto_zero_carmom(idx_bra, coord_bra, angular_bra, num_prim_bra, &
                       exponent_bra, num_contr_bra, contr_coef_bra,   &
                       idx_ket, coord_ket, angular_ket, num_prim_ket, &
                       exponent_ket, num_contr_ket, contr_coef_ket,   &
                       order_mag_bra, order_mag_ket, order_mag_total, &
                       order_ram_bra, order_ram_ket, order_ram_total, &
                       order_geo_bra, order_geo_ket,                  &
                       num_cents, idx_cent, order_cent,               &
                       idx_diporg, dipole_origin, scal_const,         &
                       order_geo_mom, order_mom, order_elec,          &
                       num_cart_bra, num_cart_ket, num_opt, contr_ints)
```

The results will be returned in

```
contr_ints(num_cart_bra,num_contr_bra,num_cart_ket,num_contr_ket, &
           num_elec,num_mom,                                      &
           num_mag_bra,num_mag_ket,num_mag_total,                 &
           num_ram_bra,num_ram_ket,num_ram_total,                 &
           num_geo_bra,num_geo_ket,num_geo_mom,num_geo_total)
```

3.2 Using Fortran 90 Module

We have also provided a Fortran 90 module `src/gen1int.F90` to facilitate Fortran users, in which we have introduced two public types `one_prop_t` and `geom_tree_t` for different pre-defined property integrals and geometric derivatives, respectively.

3.2.1 Pre-defined Property Integrals in Fortran 90 Module

You may first define a variable containing the information of property integrals and other useful arguments (you do not need all of them) as

```
use gen1int
... ...
! information of property integrals
type(one_prop_t) prop_operator
! number of property integral matrices
```

```
  integer num_prop
  ! symmetry of property integral matrices (SYMM_INT_MAT, ANTI_INT_MAT or SQUARE_INT_MAT
  ! is respectively symmetric, anti-symmetric or square matrices)
  integer prop_sym
  ! coordinates of dipole origin
  real(REALK) dipole_origin(3)
  ! atomic centers of nuclei (<1 for non-atomic center)
  integer idx_nuclei(NUM_NUCLEI)
  ! coordinates of nuclei
  real(REALK) coord_nuclei(3,NUM_NUCLEI)
  ! charges of nuclei
  real(REALK) charge_nucle(NUM_NUCLEI)
  ! order of geometric derivatives with respect to the potential center
  integer order_geo_pot
  ! order of multipole integrals
  integer order_mom
  ! atomic centers of Gaussian charge potential origins (<1 for non-atomic center)
  integer idx_gauorg(NUM_GAUPOT)
  ! coordinates of Gaussian charge potential origins
  real(REALK) gaupot_origin(3,NUM_GAUPOT)
  ! charges of Gaussian charge potential
  real(REALK) gaupot_charge(NUM_GAUPOT)
  ! exponents used in the Gaussian broadening function of charges
  real(REALK) gaupot_expt(NUM_GAUPOT)
  ! coordinates of grid points used by overlap distribution
  real(REALK) grid_points(3,NUM_POINTS)
  ! logical unit number of the viewer
  integer io_viewer
  ... ...
```

where NUM_NUCLEI is the number of nuclei, NUM_GAUPOT is the number of Gaussian charge potential origins, and NUM_POINTS is the number of grid points. REALK is defined in src/xkind.h.

The information of the one-electron property integrals prop_operator needs to be initialized by calling a public subroutine OnePropCreate with one of 8 pre-defined property integrals in this module. More explicitly, you will get

1. overlap integrals

```
    call OnePropCreate(prop_name=INT_OVERLAP,  &
                       one_prop=prop_operator, &
                       info_prop=info_prop)
```

2. kinetic energy integrals

```
    call OnePropCreate(prop_name=INT_KIN_ENERGY, &
                       one_prop=prop_operator,   &
                       info_prop=info_prop)
```

3. one-electron potential energy integrals

```
call OnePropCreate(prop_name=INT_POT_ENERGY,    &
                   one_prop=prop_operator,       &
                   info_prop=info_prop,          &
                   idx_nuclei=idx_nuclei,        &
                   coord_nuclei=coord_nuclei,    &
                   charge_nuclei=charge_nuclei, &
                   order_geo_pot=order_geo_pot)
```

4. one-electron Hamiltonian

```
call OnePropCreate(prop_name=INT_ONE_HAMIL,     &
                   one_prop=prop_operator,       &
                   info_prop=info_prop,          &
                   idx_nuclei=idx_nuclei,        &
                   coord_nuclei=coord_nuclei,    &
                   charge_nuclei=charge_nuclei)
```

5. Cartesian multipole integrals

```
call OnePropCreate(prop_name=INT_CART_MULTIPOLE, &
                   one_prop=prop_operator,        &
                   info_prop=info_prop,           &
                   dipole_origin=dipole_origin,   &
                   order_mom=order_mom)
```

6. spherical multipole integrals ***(not work)***

```
call OnePropCreate(prop_name=INT_SPHER_MULTIPOLE, &
                   one_prop=prop_operator,         &
                   info_prop=info_prop,            &
                   dipole_origin=dipole_origin,    &
                   order_mom=order_mom)
```

7. Gaussian charge potential integrals

```
call OnePropCreate(prop_name=INT_GAUSSIAN_POT,  &
                   one_prop=prop_operator,       &
                   info_prop=info_prop,          &
                   idx_gauorg=idx_gauorg,        &
                   gaupot_origin=gaupot_origin, &
                   gaupot_charge=gaupot_charge, &
                   gaupot_expt=gaupot_expt,      &
                   order_geo_pot=order_geo_pot)
```

8. overlap distribution

```
call OnePropCreate(prop_name=INT_OVERLAP_DIST, &
                   one_prop=prop_operator,      &
                   info_prop=info_prop,         &
                   grid_points=grid_points)
```

The argument for keyword `prop_name` is 8 pre-defined character parameters in this module. The information of property integrals has been successfully initialized if `info_prop=0` (otherwise you may either give some wrong input argument or the memory for the information of property integrals was not successfully allocated).

Other subroutines related to set the information of property integrals include:

1. `OnePropSetPartialGeom`: sets the partial geometric derivatives.
2. `OnePropSetMag`: sets the magnetic derivatives.
3. `OnePropSetRAM`: sets the derivatives with respect to the total rotational angular momentum.
4. `OnePropSetGTO`: sets the type of GTOs.

Please refer to the comments in corresponding subroutine in `src/gen1int.F90` for more details.

Later on, you may also need

```
call OnePropGetNumProp(one_prop=prop_operator, num_prop=num_prop)
call OnePropGetSymmetry(one_prop=prop_operator, prop_sym=prop_sym)
```

to get the number of property integral matrices returned from Gen1Int, and their symmetry information: pre-defined integer parameters `SYMM_INT_MAT`, `ANTI_INT_MAT` and `SQUARE_INT_MAT` in this module, represent th symmetric, anti-symmetric and square matrices, respectively.

The information of property integrals could be printed in a readable way by using

```
call OnePropView(one_prop=prop_operator, io_viewer=io_viewer)
```

The important public subroutine is to evaluate the integrals of `prop_operator` you created, which could be done by

```
call OnePropGetIntegral(idx_bra, coord_bra, angular_bra, num_prim_bra, &
                        exponent_bra, num_contr_bra, contr_coef_bra,   &
                        idx_ket, coord_ket, angular_ket, num_prim_ket, &
                        exponent_ket, num_contr_ket, contr_coef_ket,   &
                        spher_gto, prop_operator, geom_tree,           &
                        num_gto_bra, num_gto_ket, num_opt, contr_ints, &
                        mag_num_bra, mag_num_ket, powers_bra, powers_ket)
```

where `idx_bra` to `contr_coef_bra` provide and information of basis sets on bra center, and `idx_ket` to `contr_coef_ket` on ket center, `prop_operator` is the property integrals created by `OnePropCreate`, `num_gto_bra` and `num_gto_ket` describe the number Gaussian type orbitals on bra and ket center (for instance, 3 for $2p_x$, $2p_y$ and $2p_z$), `num_opt` is the number of operators including different derivatives, `contr_ints` contains the calculated contracted integrals after `OnePropGetIntegral` is performed.

The left arguments are optional, `geom_tree` contains the information of total geometric derivatives, and will be discussed in Section 3.2.2. The argument `spher_gto` indicates if basis sets on bra and ket center are spherical or Cartesian GTOs. The arguments `mag_num_bra` to `powers_ket` are only needed if the arrangement of your GTOs is different from that in Gen1Int (see Section 2.2), then the integrals will be reordered before returning to you.

Last but not least, you may need

```
call OnePropDestroy(one_prop=prop_operator)
```

to free the space taken by the information of one-electron property integrals after all the evaluations being done.

3.2.2 Total Geometric Derivatives in Fortran 90 Module

As will be discussed in Section 4.3, we have used the “full arithmetic N-ary tree” to search all unique geometric derivatives, such information is contained in a type variable

```
type(geom_tree_t) geom_tree
```

and is initialed by

```
call GeomTreeCreate(num_atoms=num_atoms,                  &
                    order_geo=order_geo,                  &
                    max_num_cent=max_num_cent,            &
                    geom_tree=geom_tree,                  &
                    num_paths=num_paths,                  &
                    info_geom=info_geom,                  &
                    path_num_unique=path_num_unique,      &
                    path_num_redunt=path_num_redunt)
```

where other arguments except for `geom_tree` are integers. Input arguments `num_atoms` is the number of atoms, `order_geo` is the order of total geometric derivatives, `max_num_cent` is the maximum number of differentiated centers.

Output arguments `num_paths` is the total number of different paths, `info_geom` should be 0 if the N-ary tree was successfully created. `geom_tree` contains all the information of total geometric derivatives in further calculations (for instance when calling `OnePropGetIntegral` as in Section 3.2.1), it will contain the information of the first path if the N-ary tree was successfully created, and could be therefore used in `OnePropGetIntegral` to get the total geometric derivatives of property integrals on the first path of N-ary tree.

Other two output arguments are optional: `path_num_unique` is the number of unique geometric derivatives of the first path, and `path_num_redunt` is the number of redundant geometric derivatives of the first path.

Other paths could be retrieved in an iterative way by calling

```
! loops over other paths
do ipath = 2, num_paths
  call GeomTreeSearch(geom_tree=geom_tree,                &
                      path_num_unique=path_num_unique,    &
                      path_num_redunt=path_num_redunt)
end do
```

where the optional output arguments `path_num_unique` is the number of unique geometric derivatives of current path, and `path_num_redunt` is the number of redundant geometric derivatives of current path.

Similarly, the information of N-ary tree and its current path can also be printed in a readable manner by

```
call GeomTreeView(geom_tree=geom_tree, io_viewer=io_viewer)
```

Other subroutines related to N-ary tree are:

1. `GeomTreeGetNumAtoms`: gets the number of atoms for a given N-ary tree.
2. `GeomTreeGetOrder`: gets the order of total geometric derivatives for a given N-ary tree.

3. `GeomTreeGetMaxNumCenters`: gets the maximum number of differentiated centers for a given N-ary tree.

4. `GeomTreeGetNumPaths`: gets the total number of different allowed paths in N-ary tree.

5. `GeomTreeGetNumGeo`: gets the total number of unique total geometric derivatives in the N-ary tree.

6. `GeomPathGetIndex`: gets the index of current path.

7. `GeomPathGetNumCenters`: returns the number of differentiated centers of current path for given N-ary tree.

8. `GeomPathGetOffset`: returns the offset of unique derivatives of current path (or the number of total geometric derivatives in all previous paths), which could be used to put the integral matrices and expectation values into appropriate positions.

9. `GeomPathGetNumUnique`: gets the number of unique geometric derivatives of current path.

10. `GeomPathGetNumRedunt`: gets the number of redundant geometric derivatives of current path.

11. `GeomPathGetReduntList`: gets the list addresses of redundant total geometric derivatives for current path. The output argument `redunt_list` is a 2 by `path_num_redunt` integer array, in which `redunt_list(1,:)` is the address of unique total geometric derivatives getting from, for instance, `OnePropGetIntegral`; while `redunt_list(2,:)` is the corresponding address of redundant total geometric derivatives. This array could be used to get the redundant total geometric derivatives after `OnePropGetIntegral` being performed.

12. `GeomPathSetReduntExpt`: will put the unique geometric derivatives from `OnePropGetIntegral` into appropriate positions in an array of redundant total geometric derivatives.

Please refer to the comments in corresponding subroutine in `src/gen1int_geom.F90` for more details.

Last, please do not forget to free the space taken by the N-ary tree

```
call GeomTreeDestroy(geom_tree=geom_tree)
```

3.3 Parallelization of Gen1Int

Take a molecule with 12 atoms as an example, the number of third order one-center geometric derivatives is 120, and 2376 for the two-center geometric derivatives, 5940 the three-center's — ***8436*** third order geometric derivatives in total. Therefore, an efficient way of calculating the huge number of derivatives must be considered.

In this section, we will describe a possible scheme of parallelization, we will use MPI …

3.4 Mixed Spherical and Cartesian Gaussians

Notice that the transformation between Hermite, Cartesian and spherical Gaussians could be performed on either bra or ket only, it may therefore be possible to calculate the integrals of mixed Spherical and Cartesian Gaussians.

However, we are not sure if this functionality is useful in practice …

Chapter 4

Framework of Gen1Int

The following two well-established theorems [14] will be extensively used to develop the workable recurrence relations:

Theorem 1, reversing the order of integration "Let $f(x,y)$ be continuous function of constatnt sign defined for $a \leq x < \infty$, $c \leq y < \infty$, and let the integrals $J(y) := \int_a^\infty f(x,y)\mathrm{d}x$ and $J^*(x) := \int_c^\infty f(x,y)\mathrm{d}y$ regarded as functions of the corresponding parameter be, respectively, continuous for $c \leq y < \infty$ and $a \leq x < \infty$. Then if at least one of the iterated integrals $\int_c^\infty \mathrm{d}y \left(\int_a^\infty f(x,y)\mathrm{d}x\right)$ and $\int_a^\infty \mathrm{d}x \left(\int_c^\infty f(x,y)\mathrm{d}y\right)$ converges, the other integral also converges and their values coincide."

Theorem 2, the differentiation of integration "Let $f(x,y)$ and $\frac{\partial f(x,y)}{\partial y}$ be continuous for $a \leq x < \infty$, $c \leq y \leq d$, and let the integral $J(y) := \int_a^\infty f(x,y)\mathrm{d}x$ be convergent on $c \leq y \leq d$. Supose that the integral $\int_a^\infty \frac{\partial f(x,y)}{\partial y}\mathrm{d}x$ converges uniformly on the interval $c \leq y \leq d$. Then $J(y)$ is differentiable on $c \leq y \leq d$ and

$$\frac{\mathrm{d}J(y)}{\mathrm{d}y} = \int_a^\infty \frac{\partial f(x,y)}{\partial y}\mathrm{d}x.\text{"} \tag{4.1}$$

4.1 Theoretical Background of Gen1Int

Since there are partial derivatives on bra and ket, and might be electronic derivatives in the operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$, a straightforward way of evaluating Eq. (7) is to utilize the binomial theorem for the total derivatives with respect to the magnetic field and total rotational angular momentum [1]

$$(O_{\ell_\beta})_{\kappa\lambda} = \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}-\boldsymbol{K}_0}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}-\boldsymbol{L}_0}\binom{\boldsymbol{K}-\boldsymbol{K}_0}{\boldsymbol{K}'}\binom{\boldsymbol{L}-\boldsymbol{L}_0}{\boldsymbol{L}'}\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}\int\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1+\boldsymbol{K}'}\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_1+\boldsymbol{L}'}\omega_\kappa^*(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\right]_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}} \tag{4.2}$$
$$\times\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2+\boldsymbol{K}''}\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_2+\boldsymbol{L}''}\omega_\lambda(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\right]_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}}\mathrm{d}\boldsymbol{r},$$

$$= \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}-\boldsymbol{K}_0}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}-\boldsymbol{L}_0}\binom{\boldsymbol{K}-\boldsymbol{K}_0}{\boldsymbol{K}'}\binom{\boldsymbol{L}-\boldsymbol{L}_0}{\boldsymbol{L}'}\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g} \tag{4.3}$$
$$\times\left(\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_1|+|\boldsymbol{K}'|}(-\mathrm{i})^{|\boldsymbol{L}_1|+|\boldsymbol{L}'|}\left(-\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_2|+|\boldsymbol{K}''|}\mathrm{i}^{|\boldsymbol{L}_2|+|\boldsymbol{L}''|}$$
$$\times\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int(\boldsymbol{R}_{\kappa G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_1+\boldsymbol{K}'}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\kappa O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_1+\boldsymbol{L}'}\chi_\kappa(\boldsymbol{r})$$
$$\times\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}(\boldsymbol{R}_{\lambda G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_2+\boldsymbol{K}''}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\lambda O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_2+\boldsymbol{L}''}\chi_\lambda(\boldsymbol{r})\mathrm{d}\boldsymbol{r},$$

where $\boldsymbol{K}''$ and $\boldsymbol{L}''$ are defined as

$$\boldsymbol{K}'' = \boldsymbol{K}-\boldsymbol{K}_0-\boldsymbol{K}', \tag{4.4}$$
$$\boldsymbol{L}'' = \boldsymbol{L}-\boldsymbol{L}_0-\boldsymbol{L}'. \tag{4.5}$$

By introducing the following auxiliary integrals [1]

$$\{\boldsymbol{K}_1\boldsymbol{K}_2\boldsymbol{L}_1\boldsymbol{L}_2\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0} \tag{4.6}$$
$$=\left(\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_1|}(-\mathrm{i})^{|\boldsymbol{L}_1|}\left(-\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_2|}\mathrm{i}^{|\boldsymbol{L}_2|}\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int\boldsymbol{r}_P^{\boldsymbol{N}_1}(\boldsymbol{R}_{\kappa G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_1}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\kappa O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_1}\chi_\kappa(\boldsymbol{r})$$
$$\times\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\boldsymbol{r}_P^{\boldsymbol{N}_2}(\boldsymbol{R}_{\lambda G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_2}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\lambda O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_2}\chi_\lambda(\boldsymbol{r})\mathrm{d}\boldsymbol{r},$$

the integral $(O_{\ell_\beta})_{\kappa\lambda}$ could be further written as

$$(O_{\ell_\beta})_{\kappa\lambda} = \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}-\boldsymbol{K}_0}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}-\boldsymbol{L}_0}\binom{\boldsymbol{K}-\boldsymbol{K}_0}{\boldsymbol{K}'}\binom{\boldsymbol{L}-\boldsymbol{L}_0}{\boldsymbol{L}'} \tag{4.7}$$
$$\times\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}\{\boldsymbol{K}_1{+}\boldsymbol{K}',\boldsymbol{K}_2{+}\boldsymbol{K}'',\boldsymbol{L}_1{+}\boldsymbol{L}',\boldsymbol{L}_2{+}\boldsymbol{L}'',\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{0}\boldsymbol{0}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0},$$

where $\boldsymbol{l}_\kappa$ and $\boldsymbol{l}_\lambda$ are the orbital quantum numbers of bra and ket, respectively.

The total geometric derivative $\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$ in Eq. (4.7) could be transferred to the partial geometric derivatives on bra ($\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}$), ket ($\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}$), or the centers in operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$. This requires the exact form of the operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$, and the translational invariance may also be used to reduce the computational cost. See Ref. [1] and Section 4.3 for details.

The auxiliary integral $\{\boldsymbol{K}_1\boldsymbol{K}_2\boldsymbol{L}_1\boldsymbol{L}_2\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$ as defined in Eq. (4.6) could then be reduced to $\{\boldsymbol{0000}\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$ by using the recurrence relations for both contracted spherical and Cartesian Gaussians [1, 3][*]

$$\{\boldsymbol{K}_1{+}\boldsymbol{e}_\xi,\boldsymbol{L}_\kappa\boldsymbol{N}_1\}_{\boldsymbol{K}_0\boldsymbol{L}_0} = \tfrac{\mathrm{i}}{2}\big[(\boldsymbol{R}_{\kappa G})_{\xi+1}\{\boldsymbol{K}_1\boldsymbol{L}_\kappa,\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0} \tag{4.8}$$
$$-(\boldsymbol{R}_{\kappa G})_{\xi-1}\{\boldsymbol{K}_1\boldsymbol{L}_\kappa,\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$$
$$+(\boldsymbol{L}_\kappa)_{\xi+1}\{\boldsymbol{K}_1,\boldsymbol{L}_\kappa{-}\boldsymbol{e}_{\xi+1},\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$$
$$-(\boldsymbol{L}_\kappa)_{\xi-1}\{\boldsymbol{K}_1,\boldsymbol{L}_\kappa{-}\boldsymbol{e}_{\xi-1},\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\big],$$

[*]The derivatives with respect to total rotational angular momentum are evaluated in a coordinate system in which the coordinate axes are chosen as principal axes, so that the inertia tensor **I** is diagonal.

$$\begin{aligned}\{\boldsymbol{K}_2{+}\boldsymbol{e}_\xi,\boldsymbol{L}_\lambda\boldsymbol{N}_2\}_{\boldsymbol{K}_0\boldsymbol{L}_0}=-\tfrac{\mathrm{i}}{2}\big[&(\boldsymbol{R}_{\lambda G})_{\xi+1}\{\boldsymbol{K}_2\boldsymbol{L}_\lambda,\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{R}_{\lambda G})_{\xi-1}\{\boldsymbol{K}_2\boldsymbol{L}_\lambda,\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&+(\boldsymbol{L}_\lambda)_{\xi+1}\{\boldsymbol{K}_2,\boldsymbol{L}_\lambda{-}\boldsymbol{e}_{\xi+1},\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{L}_\lambda)_{\xi-1}\{\boldsymbol{K}_2,\boldsymbol{L}_\lambda{-}\boldsymbol{e}_{\xi-1},\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\big],\end{aligned}\tag{4.9}$$

$$\begin{aligned}\{\boldsymbol{L}_1{+}\boldsymbol{e}_\xi,\boldsymbol{L}_\kappa\boldsymbol{N}_1\}_{\boldsymbol{K}_0\boldsymbol{L}_0}=-\mathrm{i}\left(\mathbf{I}^{-T}\right)_{\xi\xi}\big[&(\boldsymbol{R}_{\kappa O})_{\xi+1}\{\boldsymbol{L}_1\boldsymbol{L}_\kappa,\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{R}_{\kappa O})_{\xi-1}\{\boldsymbol{L}_1\boldsymbol{L}_\kappa,\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&+(\boldsymbol{L}_\kappa)_{\xi+1}\{\boldsymbol{L}_1,\boldsymbol{L}_\kappa{-}\boldsymbol{e}_{\xi+1},\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{L}_\kappa)_{\xi-1}\{\boldsymbol{L}_1,\boldsymbol{L}_\kappa{-}\boldsymbol{e}_{\xi-1},\boldsymbol{N}_1{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\big],\end{aligned}\tag{4.10}$$

and

$$\begin{aligned}\{\boldsymbol{L}_2{+}\boldsymbol{e}_\xi,\boldsymbol{L}_\lambda\boldsymbol{N}_2\}_{\boldsymbol{K}_0\boldsymbol{L}_0}=\mathrm{i}\left(\mathbf{I}^{-T}\right)_{\xi\xi}\big[&(\boldsymbol{R}_{\lambda O})_{\xi+1}\{\boldsymbol{L}_2\boldsymbol{L}_\lambda,\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{R}_{\lambda O})_{\xi-1}\{\boldsymbol{L}_2\boldsymbol{L}_\lambda,\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&+(\boldsymbol{L}_\lambda)_{\xi+1}\{\boldsymbol{L}_2,\boldsymbol{L}_\lambda{-}\boldsymbol{e}_{\xi+1},\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi-1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\\&-(\boldsymbol{L}_\lambda)_{\xi-1}\{\boldsymbol{L}_2,\boldsymbol{L}_\lambda{-}\boldsymbol{e}_{\xi-1},\boldsymbol{N}_2{+}\boldsymbol{e}_{\xi+1}\}_{\boldsymbol{K}_0\boldsymbol{L}_0}\big].\end{aligned}\tag{4.11}$$

The transformation of $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$ to $\boldsymbol{L}_\kappa$, $\boldsymbol{l}_\kappa$, and $\boldsymbol{L}_\lambda$, $\boldsymbol{l}_\lambda$ however depends on the type of basis functions $\chi_\kappa(\boldsymbol{r})$ and $\chi_\lambda(\boldsymbol{r})$. As mentioned in Preface, after the transformation (6) or (12), the basis functions used in the integral evaluation are either contracted Cartesian or Hermite Gaussians. For Cartesian Gaussians, the recurrence relations by transferring $\boldsymbol{N}_1$ to $\boldsymbol{l}_\kappa$, and $\boldsymbol{N}_2$ to $\boldsymbol{l}_\lambda$ directly work on the contracted integrals [3]

$$\begin{aligned}\{\boldsymbol{L}_\kappa,\boldsymbol{N}_1{+}\boldsymbol{e}_\xi,\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}=&(\boldsymbol{R}_{\kappa P})_\xi\{\boldsymbol{L}_\kappa\boldsymbol{N}_1\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}+(\boldsymbol{L}_\kappa)_\xi\{\boldsymbol{L}_\kappa{-}\boldsymbol{e}_\xi,\boldsymbol{N}_1\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}\\&+\{\boldsymbol{L}_\kappa\boldsymbol{N}_1,\boldsymbol{l}_\kappa{+}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}},\end{aligned}\tag{4.12}$$

and

$$\begin{aligned}\{\boldsymbol{L}_\lambda,\boldsymbol{N}_2{+}\boldsymbol{e}_\xi,\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}=&(\boldsymbol{R}_{\lambda P})_\xi\{\boldsymbol{L}_\lambda\boldsymbol{N}_2\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}+(\boldsymbol{L}_\lambda)_\xi\{\boldsymbol{L}_\lambda{-}\boldsymbol{e}_\xi,\boldsymbol{N}_2\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}\\&+\{\boldsymbol{L}_\lambda\boldsymbol{N}_2,\boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}},\end{aligned}\tag{4.13}$$

which give the following contracted Cartesian integrals

$$\{\mathbf{0000}\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\mathbf{00}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}=\sum_{ij}w_{i\kappa}w_{j\lambda}\,\{\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}},\tag{4.14}$$

with the primitive Cartesian integrals defined as [1, 3]

$$\{\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}=\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int\boldsymbol{r}_\kappa^{\boldsymbol{l}_\kappa}\mathrm{e}^{-a_{i\kappa}r_\kappa^2}\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\boldsymbol{r}_\lambda^{\boldsymbol{l}_\lambda}\mathrm{e}^{-b_{j\lambda}r_\lambda^2}\mathrm{d}\boldsymbol{r}.\tag{4.15}$$

We could further transfer $\boldsymbol{l}_\kappa$ to $\boldsymbol{L}_\kappa$, and $\boldsymbol{l}_\lambda$ to $\boldsymbol{L}_\lambda$ using

$$\{\boldsymbol{L}_\kappa,\boldsymbol{l}_\kappa{+}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}=\frac{1}{2a_{i\kappa}}\left[\{\boldsymbol{L}_\kappa{+}\boldsymbol{e}_\xi,\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}+(\boldsymbol{l}_\kappa)_\xi\{\boldsymbol{L}_\kappa,\boldsymbol{l}_\kappa{-}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}\right],\tag{4.16}$$

$$\{\boldsymbol{L}_\lambda,\boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}=\frac{1}{2b_{j\lambda}}\left[\{\boldsymbol{L}_\lambda{+}\boldsymbol{e}_\xi,\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}+(\boldsymbol{l}_\lambda)_\xi\{\boldsymbol{L}_\lambda,\boldsymbol{l}_\lambda{-}\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Cart}}\right],\tag{4.17}$$

and arriving at

$$\{\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{00}\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Cart}} = \boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int \mathrm{e}^{-a_{i\kappa}r_\kappa^2}\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\mathrm{e}^{-b_{j\lambda}r_\lambda^2}\mathrm{d}\boldsymbol{r}. \tag{4.18}$$

As regards the contracted Hermite integrals, we first have

$$\begin{aligned}&\{\boldsymbol{0000L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\text{Herm}} \\ &\quad= \sum_{ij} w_{i\kappa}w_{j\lambda}\{\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} \\ &\quad= \sum_{ij} w_{i\kappa}w_{j\lambda}(2a_{i\kappa})^{|\boldsymbol{L}_\kappa|}(2b_{j\lambda})^{|\boldsymbol{L}_\lambda|}\{\boldsymbol{00N}_1\boldsymbol{N}_2,\boldsymbol{l}_\kappa+\boldsymbol{L}_\kappa,\boldsymbol{l}_\lambda+\boldsymbol{L}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}},\end{aligned} \tag{4.19}$$

by transferring $\boldsymbol{L}_\kappa$ to $\boldsymbol{l}_\kappa$, and $\boldsymbol{L}_\lambda$ to $\boldsymbol{l}_\lambda$, using the following simple relations [1, 3]

$$\{\boldsymbol{L}_\kappa\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} = (2a_{i\kappa})^{|\boldsymbol{L}_\kappa|}\{\boldsymbol{0},\boldsymbol{l}_\kappa+\boldsymbol{L}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}}, \tag{4.20}$$

$$\{\boldsymbol{L}_\lambda\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} = (2b_{j\lambda})^{|\boldsymbol{L}_\lambda|}\{\boldsymbol{0},\boldsymbol{l}_\lambda+\boldsymbol{L}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}}. \tag{4.21}$$

The transformation of $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$ also has to work on the primitive Hermite Gaussians [3]

$$\begin{aligned}\{\boldsymbol{N}_1+\boldsymbol{e}_\xi,\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} &= (\boldsymbol{R}_{\kappa P})_\xi\{\boldsymbol{N}_1\boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} + \frac{(\boldsymbol{l}_\kappa)_\xi}{2a_{i\kappa}}\{\boldsymbol{N}_1,\boldsymbol{l}_\kappa-\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} \\ &\quad+\{\boldsymbol{N}_1,\boldsymbol{l}_\kappa+\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}},\end{aligned} \tag{4.22}$$

$$\begin{aligned}\{\boldsymbol{N}_2+\boldsymbol{e}_\xi,\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} &= (\boldsymbol{R}_{\lambda P})_\xi\{\boldsymbol{N}_2\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} + \frac{(\boldsymbol{l}_\lambda)_\xi}{2b_{j\lambda}}\{\boldsymbol{N}_2,\boldsymbol{l}_\lambda-\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} \\ &\quad+\{\boldsymbol{N}_2,\boldsymbol{l}_\lambda+\boldsymbol{e}_\xi\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}},\end{aligned} \tag{4.23}$$

and arriving at [3]

$$\{\boldsymbol{0000l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} = \int\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\mathrm{e}^{-a_{i\kappa}r_\kappa^2}\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}}\mathrm{e}^{-b_{j\lambda}r_\lambda^2}\mathrm{d}\boldsymbol{r}. \tag{4.24}$$

Therefore, the final basic integral to be evaluated, for both primitive Cartesian and Hermite Gaussians, is [1, 3]

$$\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij} = \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}}\int\exp(-a_{i\kappa}r_\kappa^2)\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\exp(-b_{j\lambda}r_\lambda^2)\mathrm{d}\boldsymbol{r}, \tag{4.25}$$

and apparently

$$\begin{cases}\{\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{00}\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Cart}} = (2a_{i\kappa})^{|\boldsymbol{L}_\kappa|}(2b_{j\lambda})^{|\boldsymbol{L}_\lambda|}\left[\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}, \\ \{\boldsymbol{0000l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\text{Herm}} = \left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}.\end{cases} \tag{4.26}$$

The recurrence relations of evaluating $\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}$ depends on the knowledge of operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$. In our recent work [1–3], we have developed such recurrence relations for the different operators in Eq. (9). The implementation of these recurrence relations will be discussed in Section 4.6.

Based on the aforementioned analysis, we have divided GEN1INT into the following steps in order to get the integral (7):

1. **Geometric derivatives**

   (a) Generating the total geometric derivatives $\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$ in Eq. (7) according to the given N_g and number of atoms, avoiding repetition and omission, and arranging these geometric derivatives in a required sequence (Section 4.3.1);

   (b) For both contracted spherical and Cartesian Gaussians, transferring the above generated $\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$ to $\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}$, $\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}$, or geometric derivatives of the centers in operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$ (Section 4.3.2)[†];

2. **Magnetic and total rotational angular momentum derivatives**

   (a) For both contracted spherical and Cartesian Gaussians

      i. Performing the sum (4.7) and calling the next step to calculate individual $\{\boldsymbol{K}_1{+}\boldsymbol{K}', \boldsymbol{K}_2{+}\boldsymbol{K}'', \boldsymbol{L}_1{+}\boldsymbol{L}', \boldsymbol{L}_2{+}\boldsymbol{L}'', \boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{0}\boldsymbol{0}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$ (Section 4.4)[‡];

      ii. Recovering $\{\boldsymbol{K}_1\boldsymbol{K}_2\boldsymbol{L}_1\boldsymbol{L}_2\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{0}\boldsymbol{0}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$ from $\{\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0}$ (Section 4.4)[§];

   (b) For contracted spherical Gaussians[¶]

      i. Performing the transformation (12) to contracted Hermite Gaussians (Section 4.8.3);

      ii. For primitive Hermite Gaussians

         A. Using Eqs. (4.20) and (4.21) to recover the partial geometric derivatives on bra and ket (Section 4.8.4);

         B. Recovering $\{\boldsymbol{0}\boldsymbol{0}\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}}$ from $\{\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}}$ using Eqs. (4.22) and (4.23) (Section 4.4);

   (c) For contracted Cartesian Gaussians

      i. Recovering $\{\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{N}_1\boldsymbol{N}_2\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}$ from $\{\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}_\kappa\boldsymbol{L}_\lambda\boldsymbol{0}\boldsymbol{0}\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0,\mathrm{Cart}}$ by using Eqs. (4.12) and (4.13) (Section 4.4);

      ii. For primitive Cartesian Gaussians, using Eqs. (4.16) and (4.17) to recover the orbital quantum numbers and partial geometric derivatives on bra and ket (Section 4.8.5);

3. Evaluation of different $\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}$ for primitive Hermite Gaussians, quadrature is needed for diamagnetic spin-orbit coupling, and effective core potential integrals (Section 4.6);

4. Evaluation of auxiliary functions, including Boys function (Section 4.9.1), function $G_n(T)$ (Section 4.9.2), and scaled modified spherical Bessel function of the first kind (Section 4.9.3).

As regards the use of normal atomic orbitals in Eq. (3) instead LAOs, only steps (b)i and ii.A are needed for contracted spherical Gaussians, and step (c)ii is needed for contracted Cartesian Gaussians in step 2.

In Fig. 4.1, we have given an illustration of the framework of Gen1Int, and the relationships between the aforementioned different steps.

Last but not least, one of the most important part for a library is the test suite, we have prepared tests (Fortran 90 and Python) for all the subroutines in Gen1Int. Please see Section 4.10 for more details.

[†]This step needs the exact form of the operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$.

[‡]Like previous step, the sum (4.7) is implemented in each subroutine related to different form of operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$.

[§]This step and the following steps for magnetic and total rotational angular momentum derivatives are independent of the exact form of operator, and implemented as individual subroutines.

[¶]We could also use the transformation (6) to contracted Cartesian Gaussians, and go to next step.

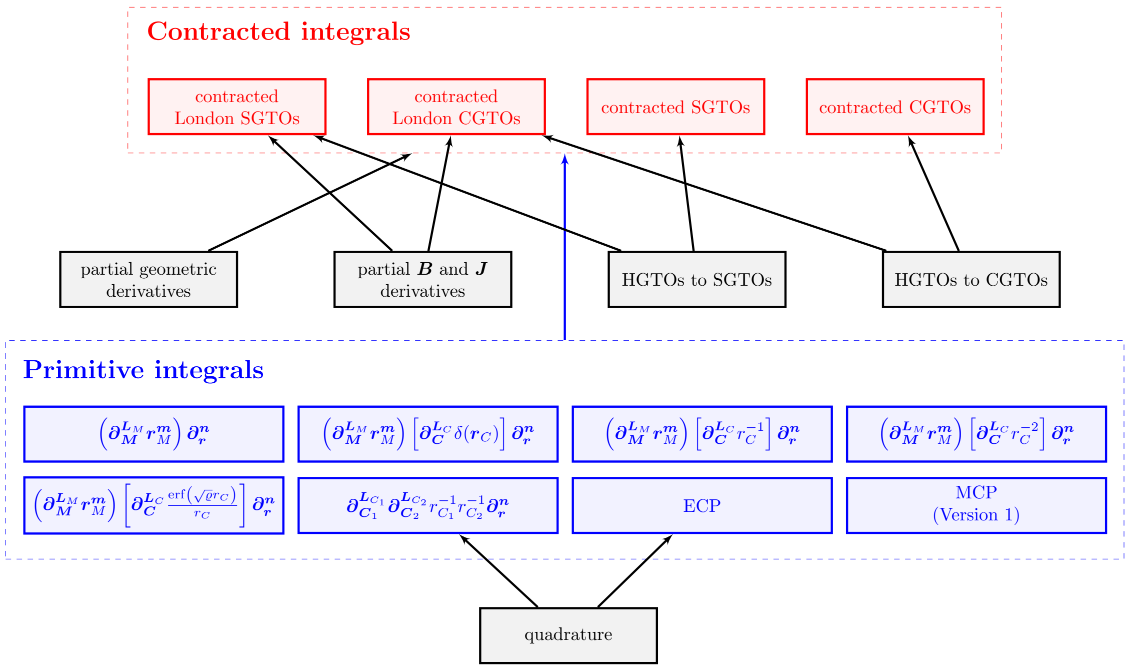

Contracted integrals
contracted London SGTOs
contracted London CGTOs
contracted SGTOs
contracted CGTOs
partial geometric derivatives
partial $\boldsymbol{B}$ and $\boldsymbol{J}$ derivatives
HGTOs to SGTOs
HGTOs to CGTOs
Primitive integrals
$\left(\partial_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
$\left(\partial_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left[\partial_{\boldsymbol{C}}^{\boldsymbol{L}_C}\delta(\boldsymbol{r}_C)\right]\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
$\left(\partial_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left[\partial_{\boldsymbol{C}}^{\boldsymbol{L}_C}r_C^{-1}\right]\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
$\left(\partial_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left[\partial_{\boldsymbol{C}}^{\boldsymbol{L}_C}r_C^{-2}\right]\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
$\left(\partial_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left[\partial_{\boldsymbol{C}}^{\boldsymbol{L}_C}\frac{\mathrm{erf}(\sqrt{\varrho}r_C)}{r_C}\right]\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
$\partial_{\boldsymbol{C}_1}^{\boldsymbol{L}_{C_1}}\partial_{\boldsymbol{C}_2}^{\boldsymbol{L}_{C_2}}r_{C_1}^{-1}r_{C_2}^{-1}\partial_{\boldsymbol{r}}^{\boldsymbol{n}}$
ECP
MCP (Version 1)
quadrature

Figure 4.1: Illustration of the framework of Gen1Int.

4.2 Data Structure in Gen1Int

As aforementioned, we have divided Gen1Int into the following steps to get the contacted integrals (7). Before discussing the individual step in detail, we first consider one of the most important left problem — the data structure in these steps. More explicitly, we need to design the ordering of the returned contracted integrals, i.e., the angular parts (or xyz powers), sub-shells, the xyz components of operators and different derivatives, which one is more consecutive in memory? The factors affecting our choice are:

1. complexity of the code by using the chosen data structure,
2. efficiency of the code by the data structure and,
3. complexity of further using the integrals by the data structure.

As regards the third factor, it is apparent that we usually need to either contract these integrals with density matrix, perform matrix operations with other integrals, or write them into file. Therefore, the angular parts (or xyz powers) and sub-shells should be the most consecutive ones. In Gen1Int, the final contracted integrals is hence always given in a five-dimensional array as

```
contr_ints(num_gto_bra,num_contr_bra,num_gto_ket,num_contr_ket,num_opt)
```

where the first and third dimensions are the angular parts (or xyz powers) on bra and ket centers, respectively. The second and fourth dimensions are respectively the sub-shells with the same azimuthal quantum number (but different principal quantum numbers) on bra and ket centers. The fifth dimension `num_opt` represents all the xyz components of different operators and derivatives. In order to give a reasonable arrangement of these xyz components, we consider the steps of evaluating contracted integrals, where the ranks with underlines/overline are those involved in current step[‖]:

1. Recovering the total geometric derivatives

$$\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}, \boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}-\boldsymbol{K}_0, \boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}-\boldsymbol{L}_0}_{\text{consecutive}}, \overline{\boldsymbol{L}_\kappa}, \overline{\boldsymbol{L}_\lambda}, \overline{\boldsymbol{L}_\alpha}, \overline{\boldsymbol{L}_{\boldsymbol{M}}}, \overline{\boldsymbol{L}_g}\right\}$$

   from

$$\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}, \boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}-\boldsymbol{K}_0, \boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}-\boldsymbol{L}_0}_{\text{consecutive}}, \underline{\boldsymbol{L}_\kappa}, \underline{\boldsymbol{L}_\lambda}, \underline{\boldsymbol{L}_\alpha}, \underline{\boldsymbol{L}_{\boldsymbol{M}}}\right\};$$

2. Recovering the geometric derivatives on dipole origin $\boldsymbol{r}_M$

$$\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}}_{\text{consecutive}}, \overline{\boldsymbol{m}}, \underbrace{\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}-\boldsymbol{K}_0, \boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}-\boldsymbol{L}_0, \boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha}_{\text{outer loop}}, \overline{\boldsymbol{L}_{\boldsymbol{M}}}\right\}$$

   from

$$\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}}_{\text{consecutive}}, \underline{\boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}}, \underbrace{\boldsymbol{K}_1, \boldsymbol{K}_2, \boldsymbol{K}-\boldsymbol{K}_0, \boldsymbol{L}_1, \boldsymbol{L}_2, \boldsymbol{L}-\boldsymbol{L}_0, \boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha}_{\text{outer loop}}\right\};$$

3. Recovering the total derivatives with respect to magnetic field and total rotational angular momentum by

$$\begin{aligned}&\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}}_{\text{consecutive}}, \overline{\boldsymbol{K}_1}, \overline{\boldsymbol{K}_2}, \overline{\boldsymbol{K}-\boldsymbol{K}_0}, \overline{\boldsymbol{L}_1}, \overline{\boldsymbol{L}_2}, \overline{\boldsymbol{L}-\boldsymbol{L}_0}, \underbrace{\boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha}_{\text{outer loop}}\right\}\\ &= \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}-\boldsymbol{K}_0}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}-\boldsymbol{L}_0}\binom{\boldsymbol{K}-\boldsymbol{K}_0}{\boldsymbol{K}'}\binom{\boldsymbol{L}-\boldsymbol{L}_0}{\boldsymbol{L}'}\\ &\times\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}}_{\text{consecutive}}, \underline{\boldsymbol{K}_1+\boldsymbol{K}'}, \underline{\boldsymbol{K}_2+\boldsymbol{K}''}, \underline{\boldsymbol{L}_1+\boldsymbol{L}'}, \underline{\boldsymbol{L}_2+\boldsymbol{L}''}, \underbrace{\boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha}_{\text{outer loop}}\right\};\end{aligned}$$

4. Recovering

$$\left\{\underbrace{\boldsymbol{l}_\kappa, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}}_{\text{consecutive}}, \overline{\boldsymbol{K}_1+\boldsymbol{K}'}, \overline{\boldsymbol{K}_2+\boldsymbol{K}''}, \overline{\boldsymbol{L}_1+\boldsymbol{L}'}, \overline{\boldsymbol{L}_2+\boldsymbol{L}''}, \overline{\boldsymbol{L}_\kappa}, \overline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha}_{\text{outer loop}}\right\}$$

[‖]We would like to mention that there are usually nested loops in the recurrence relations. It is said that it would be better make the number of iterations of outer loop be fewer, whilst that of inner loop be more. This depends on the cache, pipelining and predetermination of CPUs. However, the later two depend on specific CPUs. We therefore mainly focus on increasing the CPU caching, i.e., we choose the data structure with more consecutive data during each step.

from

$$\left\{\underbrace{\underline{\boldsymbol{l}_\kappa}, \mathrm{contr}_{\mathrm{bra}}, \boldsymbol{l}_\lambda, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}}_{\text{consecutive}}, \underline{\boldsymbol{N}_1}, \underline{\boldsymbol{N}_2}, \underline{\boldsymbol{L}_\kappa}, \underline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha}_{\text{outer loop}}\right\};$$

5. For contracted spherical Gaussians

   (a) Getting the contracted spherical Gaussian integrals from contracted Hermite Gaussian integrals by

$$\sum_{|\boldsymbol{l}_\kappa|=l_\kappa}\sum_{|\boldsymbol{l}_\lambda|=l_\lambda} S^{l_\kappa m_\kappa}_{\boldsymbol{l}_\kappa} S^{l_\lambda m_\lambda}_{\boldsymbol{l}_\lambda} \sum_{ij} w_{i\kappa} w_{j\lambda} \left\{\boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \boldsymbol{N}_1, \boldsymbol{N}_2, \boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha, i, j\right\}_{\mathrm{Herm}},$$

   (b) Recovering primtive Hermite Gaussian integrals

$$\left\{\overline{\boldsymbol{l}_\kappa}, \overline{\boldsymbol{l}_\lambda}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \boldsymbol{N}_1, \boldsymbol{N}_2, \overline{\boldsymbol{L}_\kappa}, \overline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\mathrm{Herm}}$$

   from

$$\left\{\overline{\boldsymbol{l}_\kappa{+}\boldsymbol{L}_\kappa}, \overline{\boldsymbol{l}_\lambda{+}\boldsymbol{L}_\lambda}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \boldsymbol{N}_1, \boldsymbol{N}_2, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\mathrm{Herm}},$$

   (c) Recovering

$$\left\{\overline{\boldsymbol{l}_\kappa}, \overline{\boldsymbol{l}_\lambda}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \overline{\boldsymbol{N}_1}, \overline{\boldsymbol{N}_2}, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\mathrm{Herm}}$$

   from

$$\left\{\underline{\boldsymbol{l}_\kappa}, \underline{\boldsymbol{l}_\lambda}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\mathrm{Herm}};$$

6. For contracted Cartesian Gaussians

   (a) Recovering

$$\left\{\overline{\boldsymbol{l}_\kappa}, \mathrm{contr}_{\mathrm{bra}}, \overline{\boldsymbol{l}_\lambda}, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \overline{\boldsymbol{N}_1}, \overline{\boldsymbol{N}_2}, \overline{\boldsymbol{L}_\kappa}, \overline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha}_{\text{outer loop}}\right\}$$

   from

$$\left\{\underline{\boldsymbol{l}_\kappa}, \mathrm{contr}_{\mathrm{bra}}, \underline{\boldsymbol{l}_\lambda}, \mathrm{contr}_{\mathrm{ket}}, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \underline{\boldsymbol{L}_\kappa}, \underline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha}_{\text{outer loop}}\right\}$$
$$= \sum_{ij} w_{i\kappa} w_{j\lambda} \left\{\boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda, \boldsymbol{n}, \boldsymbol{m}{-}\boldsymbol{L}_{\boldsymbol{M}}, \boldsymbol{L}_\kappa, \boldsymbol{L}_\lambda, \boldsymbol{L}_\alpha, i, j\right\}_{\mathrm{Cart}},$$

(b) Recovering primitive Cartesian Gaussian integrals

$$\left\{\overline{\boldsymbol{l}_\kappa}, \overline{\boldsymbol{l}_\lambda}, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}, \overline{\boldsymbol{L}_\kappa}, \overline{\boldsymbol{L}_\lambda}, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\text{Cart}}$$

from primitive Hermite Gaussian integrals

$$\left\{\underline{\boldsymbol{L}_\kappa}, \underline{\boldsymbol{L}_\lambda}, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_{\boldsymbol{M}}, \underbrace{\boldsymbol{L}_\alpha, i, j}_{\text{outer loop}}\right\}_{\text{Herm}} \quad ;$$

7. Recovering different primitive Hermite Gausssin integrals $\{\boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda, \boldsymbol{n}, \boldsymbol{m}-\boldsymbol{L}_M, \boldsymbol{L}_\alpha, i, j\}_{\text{Herm}}$.

Therefore, the fifth dimension is arranged in the order of

```
num_elec, num_mom,
num_mag_bra, num_mag_ket, num_mag_total,
num_ram_bra, num_ram_ket, num_ram_total,
num_geo_bra, num_geo_ket, num_geo_opt, num_geo_total,
```

As regards the total geometric derivatives, the xyz components of the first differentiated center is the most consecutive part, followed by the second, third, ..., and the last differentiated center.

4.3 Geometric Derivatives

As mentioned previously, there are two tasks for geometric derivatives:

1. Generating the total geometric derivatives $\prod^{N_g} \boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}$ according to the given N_g and number of atoms, avoiding repetition and omission, and arranging these geometric derivatives in a required sequence;

2. Transferring the above generated total geometric derivatives to partial geometric derivatives on centers bra, ket, and/or centers in operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$.

These questions will be addressed in the following two sections.

4.3.1 Sequence of Total Geometric Derivatives

We first consider the question of generating all possible total geometric derivatives in a required sequence. Let us consider the g-center L-th order total geometric derivatives (of an N-atom system)

$$\boldsymbol{\partial}_{\boldsymbol{R}_1}^{\boldsymbol{L}_1}\boldsymbol{\partial}_{\boldsymbol{R}_2}^{\boldsymbol{L}_2}\cdots\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}\int \omega_\kappa^*(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\hat{O}_{\ell_\beta}\left(\{\boldsymbol{r}_{C_\alpha}\}, \boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}};\boldsymbol{B},\boldsymbol{J}\right)\omega_\lambda(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\mathrm{d}\boldsymbol{r}, \tag{4.27}$$

where

$$\begin{cases} |\boldsymbol{L}_{g'}| \geq 1, & (1 \leq g' \leq g), \\ \sum_{g'=1}^{g} |\boldsymbol{L}_{g'}| = L. \end{cases} \tag{4.28}$$

The sequence of the total geometric derivatives represents by the sequence of the indices (denoted as $\overline{R_{g'}}$, $1 \leq g' \leq g$) of centers. We use an ascending order of the indices in GEN1INT

$$1 \leq \overline{R_1} \leq \overline{R_2} \leq \ldots \overline{R_g} \leq N. \tag{4.29}$$

In Ref. [1], we have developed a procedure to generate all the possible total geometric derivatives by using the conception "full arithmetic N-ary tree". Taking fourth order total geometric derivatives as an example, as shown in Fig. 4.2, the task of finding required total geometric derivatives could be done by the following steps:

1. For a given path $\{\overline{R_1}, \ldots, \overline{R_{v-1}}, \overline{R_v}, \overline{R_{v+1}}, \ldots, \overline{R_L}\}$ and "height" v[**], we replace $\overline{R_v}$ with its sibling $\overline{R_v}+1$. Here $\overline{R_v}$ should be the highest node which could be replaced, i.e., $\overline{R_{v+1}} = \cdots = \overline{R_L} = N$.

2. We then set $\overline{R_{v+1}} = \cdots = \overline{R_L} = \overline{R_v}+1$. If the number of different centers in path $\{\overline{R_1}, \ldots, \overline{R_{v-1}}, \overline{R_v}+1, \ldots, \overline{R_v}+1\}$ is less than or equal to $N_\alpha + 2$[††], we then choose this path[‡‡] and set $v = L$ ($\overline{R_v}+1 < N$) or $v = v-1$ ($\overline{R_v}+1 = N$); otherwise, we set $v = v-1$ and back to previous step to generate satisfied path.

This procedure could start from the leftmost path $\{\underbrace{11\ldots1}_{L}\}$ and $v = L$, and end at the path $\{\underbrace{NN\ldots N}_{L}\}$. All required total geometric derivatives could be generated, without repetition and omission.

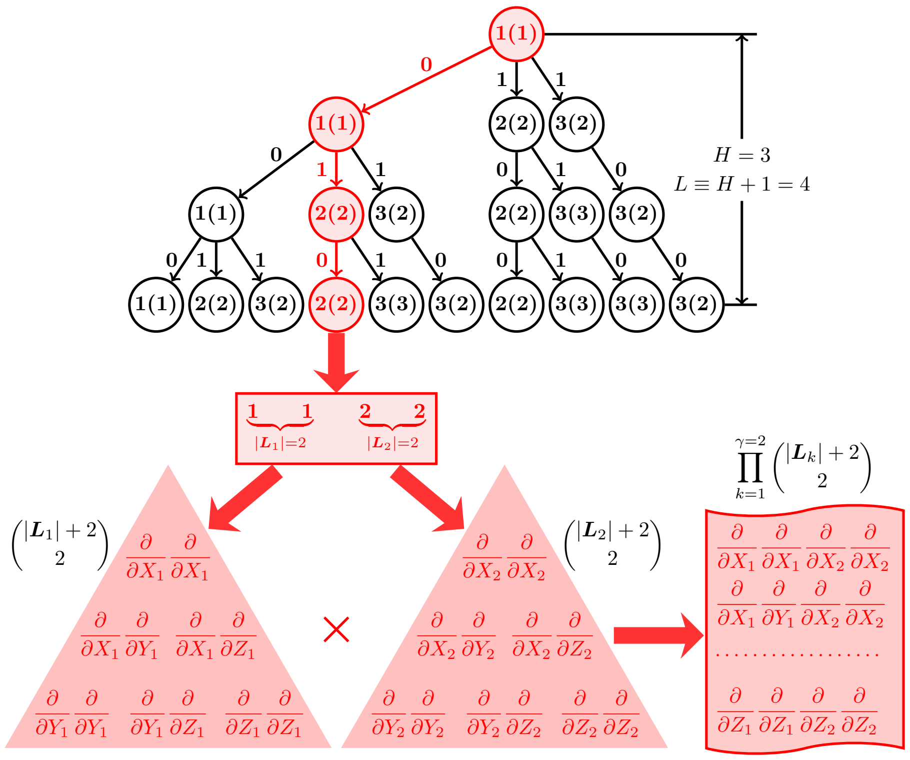


Figure 4.2: A typical "full arithmetic 3-ary tree" with height $H = 3$ started from the first atom. The numbers in the parentheses, and along with the arrows (paths between two nodes) are the weights of the corresponding node and path. The selected path from the "root" to the "leaf" node (denoted as red color) represents the generated differentiated geometric centers, while the red triangles and flag are the generated two center fourth order total geometric derivatives.

This procedure has been implemented in file `geom_total.F90` with the subroutines detailed in

[**] The height should be $v-1$.

[††] N_α is the number of centers in operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$.

[‡‡] Differentiated centers are in the reverse order of this path.

Table 4.1, where "**Public**" subroutines could be called by users in their own codes, and "**Private**" subroutines are usually not be called by the users.

Table 4.1: Subroutines of total geometric derivatives in GEN1INT.

Public			
geom_total_tree_init	Returns the total number of different paths, and generates the first path.		
	In	num_atoms	number of atoms (N)
		order_geo	order of total geometric derivatives (L)
		max_num_cent	maximum number of differentiated centers (g)
	Out	num_paths	total number of different paths
		visit_height	"height" of atom to visit (v)
		idx_node	indices of the selected atom nodes ($\{\overline{R_1}, \overline{R_2}, \ldots, \overline{R_L}\}$)
		wt_node	weights of the selected atom nodes
		idx_cent	indices of generated differentiated centers
		order_cent	order of derivatives of the differentiated centers
		num_geo_cent	number of all geometric derivatives for this generated path ($\prod_{g'=1}^{g}\binom{\|\boldsymbol{L}_{g'}\|+2}{2}$)
geom_total_tree_search	Searches for the next satisfied path from a given path, could be called recursively.		
	In	num_atoms	number of atoms (N)
		order_geo	order of total geometric derivatives (L)
		max_num_cent	maximum number of differentiated centers (g)
	InOut	visit_height	"height" of atom to visit (v)
		idx_node	indices of the selected atom nodes ($\{\overline{R_1}, \overline{R_2}, \ldots, \overline{R_L}\}$)
		wt_node	weights of the selected atom nodes
		idx_cent	indices of generated differentiated centers
		order_cent	order of derivatives of the differentiated centers
	Out	num_geo_cent	number of all geometric derivatives for this generated path ($\prod_{g'=1}^{g}\binom{\|\boldsymbol{L}_{g'}\|+2}{2}$)
Private			
geom_total_num_paths	Computes the total number of different paths.		
	In	num_atoms	number of atoms (N)
		order_geo	order of total geometric derivatives (L)
		max_num_cent	maximum number of differentiated centers (g)
	Out	num_paths	total number of different paths ($\sum_{g'=1}^{\min(N_g,L)}\binom{N}{g'}\binom{L-1}{g'-1}$)
geom_total_new_path	Generates a new path of differentiated centers from a given path, the first path will return for giving the last path.		
	In	num_atoms	number of atoms (N)
		order_geo	order of total geometric derivatives (L)

Continued on next page

Table 4.1 – continued from previous page

	InOut	`visit_height`	"height" of atom to visit (v)
		`idx_node`	indices of the selected atom nodes ($\{\overline{R_1},\overline{R_2},\ldots,\overline{R_L}\}$)
		`wt_node`	weights of the selected atom nodes

Last but not least, the number of redundant total geometric derivatives for a given path in Fig. 4.2 could be calculated as

$$3^L\prod_{g'=1}^{g}\binom{L-\sum_{n=1}^{g'-1}|\boldsymbol{L}_n|}{|\boldsymbol{L}_{g'}|},\tag{4.30}$$

as implemented in subroutine `geom_total_num_redunt`.

4.3.2 Partial Geometric Derivatives

After generating the differentiated centers $\{\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_g\}$ and their orders of total geometric derivatives $\{|\boldsymbol{L}_1|,|\boldsymbol{L}_2|,\cdots,|\boldsymbol{L}_g|\}$, the left problems are

1. transferring the generated total geometric derivatives to partial geometric derivatives on centers of bra, ket and/or operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$;
2. after calculations, retrieving the total geometric derivatives from these partial geometric derivatives.

Suppose the centers of bra, ket and/or operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$ are $\{\boldsymbol{R}_\kappa,\boldsymbol{R}_\lambda,\boldsymbol{R}_\alpha,\cdots\}$. In the first step, we could compare the indices of these centers with those of differentiated centers $\{\boldsymbol{R}_1,\boldsymbol{R}_2,\cdots,\boldsymbol{R}_g\}$ when $g\leq N_\alpha+2$. If there are identical centers in those of bra, ket and/or operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$, let us say there are N_β identical centers $\left\{\boldsymbol{R}_{\beta_1},\boldsymbol{R}_{\beta_2},\cdots,\boldsymbol{R}_{\beta_{N_\beta}}\right\}$, which are also identical with the differentiated center $\boldsymbol{R}_\beta$ in the total geometric derivatives, and order $|\boldsymbol{L}_\beta|$. We then have, according to the multinomial theorem (http://en.wikipedia.org/wiki/Multinomial_theorem), the following partial geometric derivatives

$$\sum_{\sum_{k=1}^{N_\beta}|\boldsymbol{l}_k|=|\boldsymbol{L}_\beta|}\binom{|\boldsymbol{L}_\beta|}{|\boldsymbol{l}_1|,|\boldsymbol{l}_2|,\cdots,|\boldsymbol{l}_{N_\beta}|}\prod_{k=1}^{N_\beta}\boldsymbol{\partial}_{\boldsymbol{R}_{\beta_k}}^{\boldsymbol{l}_k},\tag{4.31}$$

where

$$\binom{|\boldsymbol{L}_\beta|}{|\boldsymbol{l}_1|,|\boldsymbol{l}_2|,\cdots,|\boldsymbol{l}_{N_\beta}|}=\frac{|\boldsymbol{L}_\beta|!}{|\boldsymbol{l}_1|!|\boldsymbol{l}_2|!\cdots|\boldsymbol{l}_{N_\beta}|!}\tag{4.32}$$

$$=\binom{|\boldsymbol{l}_1|}{|\boldsymbol{l}_1|}\binom{|\boldsymbol{l}_1|+|\boldsymbol{l}_2|}{|\boldsymbol{l}_2|}\cdots\binom{|\boldsymbol{l}_1|+|\boldsymbol{l}_2|+\cdots+|\boldsymbol{l}_{N_\beta}|}{|\boldsymbol{l}_{N_\beta}|}\tag{4.33}$$

$$=\prod_{k=1}^{N_\beta}\binom{\sum_{k'=1}^{k}|\boldsymbol{l}_{k'}|}{|\boldsymbol{l}_k|},\tag{4.34}$$

is the multinomial coefficient. The sum of all multinomial coefficients is

$$\sum_{\sum_{k=1}^{N_\beta}|\boldsymbol{l}_k|=|\boldsymbol{L}_\beta|}\binom{|\boldsymbol{L}_\beta|}{|\boldsymbol{l}_1|,|\boldsymbol{l}_2|,\cdots,|\boldsymbol{l}_{N_\beta}|}=N_\beta^{|\boldsymbol{L}_\beta|},\tag{4.35}$$

and the number of multinomial coefficients (or the number of terms in multinomial sum) $\#_{|\boldsymbol{L}_\beta|,N_\beta}$ is

$$\#_{|\boldsymbol{L}_\beta|,N_\beta}=\binom{|\boldsymbol{L}_\beta|+N_\beta-1}{N_\beta-1}=\binom{|\boldsymbol{L}_\beta|+N_\beta-1}{|\boldsymbol{L}_\beta|}. \tag{4.36}$$

As pointed out in our recent work [1], sometimes this expansion is however not efficient, further simplifications could be made by using the translational invariance [15, 16]. In Ref. [1], we have given the possible partial geometric derivatives for operators with the number of centers $N_\alpha\le 2$ and in the case of $\overline{R_\kappa}=\overline{R_\lambda}$. These results are also shown here in Table 4.2.

Table 4.2: Possible partial geometric derivatives for $\overline{R_\kappa}=\overline{R_\lambda}$ and the operator with the number of centers $N_\alpha\le 2$.

	Identical centers	Possible partial geometric derivatives
$N_\alpha=0$	$\overline{R_\kappa}=\overline{R_\lambda}$	0
$N_\alpha=1$	$\overline{R_\kappa}=\overline{R_\lambda}=\overline{C_1}$	0
	$\overline{R_\kappa}=\overline{R_\lambda}\neq\overline{C_1}$	$(-1)^{\lvert\boldsymbol{L}_\kappa\rvert}\boldsymbol{\partial}_{\boldsymbol{C}_1}^{\boldsymbol{L}_{C_1}+\boldsymbol{L}_\kappa}$
$N_\alpha=2$	$\overline{R_\kappa}=\overline{R_\lambda}=\overline{C_1}=\overline{C_2}$	0
	$\overline{R_\kappa}=\overline{R_\lambda}=\overline{C_1}\neq\overline{C_2}$	$(-1)^{\lvert\boldsymbol{L}_\kappa\rvert}\boldsymbol{\partial}_{\boldsymbol{C}_2}^{\boldsymbol{L}_{C_2}+\boldsymbol{L}_\kappa}$
	$(\overline{R_\kappa}=\overline{R_\lambda})\neq(\overline{C_1}=\overline{C_2})$	$(-1)^{\lvert\boldsymbol{L}_\kappa\rvert}\sum_{\boldsymbol{l}_1=\boldsymbol{0}}^{\boldsymbol{L}_\kappa+\boldsymbol{L}_{C_1}}\binom{\boldsymbol{L}_\kappa+\boldsymbol{L}_{C_1}}{\boldsymbol{l}_1}\boldsymbol{\partial}_{\boldsymbol{C}_1}^{\boldsymbol{l}_1}\boldsymbol{\partial}_{\boldsymbol{C}_2}^{\boldsymbol{L}_\kappa+\boldsymbol{L}_{C_1}-\boldsymbol{l}_1}$
	$\overline{R_\kappa}=\overline{R_\lambda}\neq\overline{C_1}\neq\overline{C_2}$	$\sum_{\boldsymbol{l}_\kappa=\boldsymbol{0}}^{\boldsymbol{L}_\kappa}\binom{\boldsymbol{L}_\kappa}{\boldsymbol{l}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\kappa-\boldsymbol{l}_\kappa}\boldsymbol{\partial}_{\boldsymbol{C}_1}^{\boldsymbol{L}_{C_1}}\boldsymbol{\partial}_{\boldsymbol{C}_2}^{\boldsymbol{L}_{C_2}}$

Being aware of that most operators in one-electron integrals have two centers at maximum, we therefore only consider the operators with $N_\alpha\le 2$ centers in current version of GEN1INT by hand coding implementation of Table 4.2. The subroutines related to partial geometric derivatives are implemented in files `geom_part_zero.F90`, `geom_part_one.F90` and `geom_part_two.F90`.

Describe `shell_scatter.F90` and `shell_gather.F90` ... ***The procedure of scattering shells*** is given in Fig. 4.3, and implemented in file `shell_scatter.F90` ...

$$|\boldsymbol{m}|+|\boldsymbol{n}|+1<(|\boldsymbol{n}|+1)\frac{(|\boldsymbol{m}|+1)(|\boldsymbol{m}|+2)}{2} \tag{4.37}$$

4.4 Magnetic and Total Rotational Angular Momentum Derivatives

We assume that $\boldsymbol{K}_0$ and $\boldsymbol{L}_0$ in the operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$ could run all the xyz components in the triangle ...

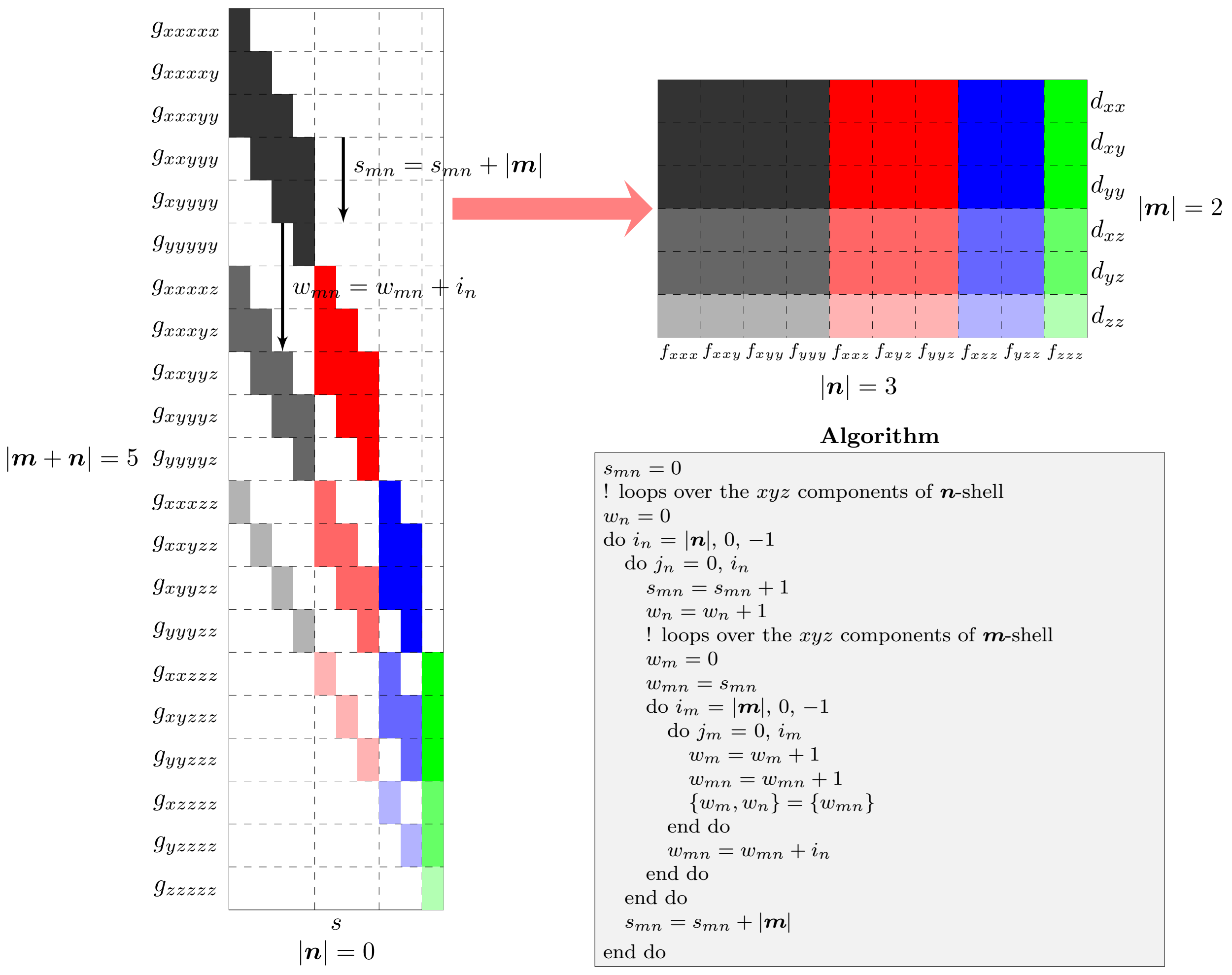
In order to evaluate

$$\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1}\omega_\kappa^*(\boldsymbol{r};\boldsymbol{B})\right]_{\boldsymbol{B}=\boldsymbol{0}}\hat{O}\left(\{\boldsymbol{r}_{C_\alpha}\},\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\right)\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2}\omega_\lambda(\boldsymbol{r};\boldsymbol{B})\right]_{\boldsymbol{B}=\boldsymbol{0}}\mathrm{d}\boldsymbol{r} \tag{4.38}$$

to any order $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$, we introduce the following auxiliary integral

$$\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\int\boldsymbol{r}_P^{\boldsymbol{N}_1}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1}\omega_\kappa^*(\boldsymbol{r};\boldsymbol{B})\right]_{\boldsymbol{B}=\boldsymbol{0}}\hat{O}\left(\{\boldsymbol{r}_{C_\alpha}\},\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\right)\boldsymbol{r}_P^{\boldsymbol{N}_2}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2}\omega_\lambda(\boldsymbol{r};\boldsymbol{B})\right]_{\boldsymbol{B}=\boldsymbol{0}}\mathrm{d}\boldsymbol{r}, \tag{4.39}$$

Describe how to perform this recurrence relations and sum ....



Algorithm

```
$s_{mn} = 0$
! loops over the $xyz$ components of $\boldsymbol{n}$-shell
$w_n = 0$
do $i_n = |\boldsymbol{n}|$, 0, $-1$
   do $j_n = 0$, $i_n$
      $s_{mn} = s_{mn} + 1$
      $w_n = w_n + 1$
      ! loops over the $xyz$ components of $\boldsymbol{m}$-shell
      $w_m = 0$
      $w_{mn} = s_{mn}$
      do $i_m = |\boldsymbol{m}|$, 0, $-1$
         do $j_m = 0$, $i_m$
            $w_m = w_m + 1$
            $w_{mn} = w_{mn} + 1$
            $\{w_m, w_n\} = \{w_{mn}\}$
         end do
         $w_{mn} = w_{mn} + i_n$
      end do
   end do
   $s_{mn} = s_{mn} + |\boldsymbol{m}|$
end do
```

Figure 4.3: Procedure of scattering shells.

4.5 Contracted Integrals

Describe subroutine `const_contr_ints` in file `const_contr_ints.F90` which performs the contractions (4.14) and (4.19) ...

Describe all the subroutines related to contracted integrals ...

4.6 One-electron Operators in Gen1Int

We first discuss the evaluation of basic integral $\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}$ with the operator $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}$ being the form $\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\left(\{\boldsymbol{r}_{C_\alpha}\},\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\right)=\bar{C}f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}$. The cases of effective core potential and model core potential (Version 1) could be obtained in the similar way. By substituting the explicit form of the operator into the basic integral (4.25), we get [1–3]

$$\begin{aligned}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}&=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda+\boldsymbol{n}}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|+|\boldsymbol{n}|}}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right]\\&\equiv\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda{+}\boldsymbol{n}\middle|f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\right]_{ij},\end{aligned}\tag{4.40}$$

where we have introduced the quantities

$$p_{ij}=a_{i\kappa}+b_{j\lambda},\quad u_{ij}=\frac{a_{i\kappa}b_{j\lambda}}{p_{ij}},\quad \boldsymbol{R}_\gamma=\frac{a_{i\kappa}\boldsymbol{R}_\kappa+b_{j\lambda}\boldsymbol{R}_\lambda}{p_{ij}}.\tag{4.41}$$

The recurrence relations of $\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda{+}\boldsymbol{n}\middle|f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\right]_{ij}$ for different $f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)$ have been discussed in Ref. [1–3]. We will, in the following sections, discuss the implementation of these recurrence relations.

4.6.1 Electronic Derivatives

The electronic derivatives could be recovered through

$$\bar{C}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\right]_{ij}\equiv\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda{+}\boldsymbol{n}\middle|f\left(\{\boldsymbol{r}_{C_\alpha}\}\right)\right]_{ij},\tag{4.42}$$

or in a more compact form

$$\{\boldsymbol{l}_\lambda,\boldsymbol{n}\}=\{\boldsymbol{l}_\lambda{+}\boldsymbol{n}\}.\tag{4.43}$$

The electronic derivatives could be recovered by calling the subroutine `shell_scatter` (see Section 4.3.2) ...

4.6.2 Cartesian Multipole Moments

The operator of Cartesian multipole moments can be written as follows

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}=\bar{C}\left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}.\tag{4.44}$$

The electronic derivatives have been discussed in previous section 4.6.1. We will, in current section, discuss in detail about the evaluation of the integrals of the operator $\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)$.

By gluing the subroutines together, we get the subroutine `prim_hgto_carmom` which returns the Cartesian multipole moment integrals of given primitive Hermite Gaussians on bra and ket centers. Describe `prim_hgto_carmom` in detail ...

Geometric Derivatives of Dipole Origin

The geometric derivatives of dipole origin $\boldsymbol{r}_M$ can be further written as

$$\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}} = \frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}, \tag{4.45}$$

where we have required that

$$\boldsymbol{m} \geq \boldsymbol{L}_M. \tag{4.46}$$

Therefore, the integrals of geometric derivatives of dipole origin $\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}$ in a multi-dimensional array $\{1{:}N_{\boldsymbol{L}_M}, 1{:}N_{\boldsymbol{m}}\}$, could be retrieved from the integrals of lower order Cartesian multipole moments $\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}$ in an array $\{1{:}N_{\boldsymbol{m}-\boldsymbol{L}_M}\}$, where

$$N_{\boldsymbol{L}_M} = \frac{(|\boldsymbol{L}_M|+1)(|\boldsymbol{L}_M|+2)}{2}, \tag{4.47}$$

$$N_{\boldsymbol{m}} = \frac{(|\boldsymbol{m}|+1)(|\boldsymbol{m}|+2)}{2}, \tag{4.48}$$

$$N_{\boldsymbol{m}-\boldsymbol{L}_M} = \frac{(|\boldsymbol{m}|-|\boldsymbol{L}_M|+1)(|\boldsymbol{m}|-|\boldsymbol{L}_M|+2)}{2}, \tag{4.49}$$

as illustrated in Table 4.3. The procedure of retrieving the integrals of geometric derivatives of dipole origin $\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}$ has been implemented in file `carmom_deriv.F90`.

Table 4.3: Retrieving integrals of $\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{3}}\boldsymbol{r}_M^{\boldsymbol{5}}$ through those of $\boldsymbol{r}_M^{\boldsymbol{2}}$.

	∂_{xxx}	∂_{xxy}	∂_{xyy}	∂_{yyy}	∂_{xxz}	∂_{xyz}	∂_{yyz}	∂_{xzz}	∂_{yzz}	∂_{zzz}
r_{xxxxx}	r_{xx}	0	0	0	0	0	0	0	0	0
r_{xxxxy}	r_{xy}	r_{xx}	0	0	0	0	0	0	0	0
r_{xxxyy}	r_{yy}	r_{xy}	r_{xx}	0	0	0	0	0	0	0
r_{xxyyy}	0	r_{yy}	r_{xy}	r_{xx}	0	0	0	0	0	0
r_{xyyyy}	0	0	r_{yy}	r_{xy}	0	0	0	0	0	0
r_{yyyyy}	0	0	0	r_{yy}	0	0	0	0	0	0
r_{xxxxz}	r_{xz}	0	0	0	r_{xx}	0	0	0	0	0
r_{xxxyz}	r_{yz}	r_{xz}	0	0	r_{xy}	r_{xx}	0	0	0	0
r_{xxyyz}	0	r_{yz}	r_{xz}	0	r_{yy}	r_{xy}	r_{xx}	0	0	0
r_{xyyyz}	0	0	r_{yz}	r_{xz}	0	r_{yy}	r_{xy}	0	0	0
r_{yyyyz}	0	0	0	r_{yz}	0	0	r_{yy}	0	0	0
r_{xxxzz}	r_{zz}	0	0	0	r_{xz}	0	0	r_{xx}	0	0
r_{xxyzz}	0	r_{zz}	0	0	r_{yz}	r_{xz}	0	r_{xy}	r_{xx}	0
r_{xyyzz}	0	0	r_{zz}	0	0	r_{yz}	r_{xz}	r_{yy}	r_{xy}	0
r_{yyyzz}	0	0	0	r_{zz}	0	0	r_{yz}	0	r_{yy}	0
r_{xxzzz}	0	0	0	0	r_{zz}	0	0	r_{xz}	0	r_{xx}
r_{xyzzz}	0	0	0	0	0	r_{zz}	0	r_{yz}	r_{xz}	r_{xy}
r_{yyzzz}	0	0	0	0	0	0	r_{zz}	0	r_{yz}	r_{yy}
r_{xzzzz}	0	0	0	0	0	0	0	r_{zz}	0	r_{xz}
r_{yzzzz}	0	0	0	0	0	0	0	0	r_{zz}	r_{yz}
r_{zzzzz}	0	0	0	0	0	0	0	0	0	r_{zz}

Recovering the Cartesian Multipole Moments

After removing the geometric derivatives on dipole origin, what left for us becomes

$$\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{m}\right]_{ij}=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\boldsymbol{r}_M^{\boldsymbol{m}}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right], \tag{4.50}$$

the Cartesian multipole moments could be recovered through the following recurrence relation [7]

$$\begin{aligned}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda,\boldsymbol{m}+\boldsymbol{e}_\xi\right]_{ij}=(\boldsymbol{R}_{\gamma M})_\xi\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{m}\right]_{ij}+\frac{1}{2p_{ij}}\Big\{&(\boldsymbol{l}_\kappa)_\xi\left[\boldsymbol{l}_\kappa-\boldsymbol{e}_\xi,\boldsymbol{l}_\lambda\boldsymbol{m}\right]_{ij}\\&+(\boldsymbol{l}_\lambda)_\xi\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda-\boldsymbol{e}_\xi,\boldsymbol{m}\right]_{ij}+m_\xi\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda,\boldsymbol{m}-\boldsymbol{e}_\xi\right]_{ij}\Big\},\end{aligned} \tag{4.51}$$

which has been implemented in file `carmom_moment.F90`.

Horizontal Recurrence Relation on Ket Center

The HGTOs on ket center could be recovered by the "horizontal" recurrence relation (HRR) [7]

$$\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda+\boldsymbol{e}_\xi,\boldsymbol{0}\right]_{ij}=-\frac{a_{i\kappa}}{b_{j\lambda}}\left[\boldsymbol{l}_\kappa+\boldsymbol{e}_\xi,\boldsymbol{l}_\lambda\boldsymbol{0}\right]_{ij}, \tag{4.52}$$

which has been implemented in file `carmom_hrr_ket.F90`.

Recovering the HGTOs on Bra Center

The HGTOs on bra center could be obtained through the following recurrence relation [7]

$$\left[\boldsymbol{l}_\kappa+\boldsymbol{e}_\xi,\boldsymbol{00}\right]_{ij}=(\boldsymbol{R}_{\gamma\kappa})_\xi\left[\boldsymbol{l}_\kappa\boldsymbol{00}\right]_{ij}-\frac{1}{2p_{ij}}\frac{(\boldsymbol{l}_\kappa)_\xi b_{j\lambda}}{a_{i\kappa}}\left[\boldsymbol{l}_\kappa-\boldsymbol{e}_\xi,\boldsymbol{00}\right]_{ij}, \tag{4.53}$$

starting from the integrals

$$\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\left[\boldsymbol{000}\right]_{ij}=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\left(\frac{\pi}{p_{ij}}\right)^{\frac{3}{2}}. \tag{4.54}$$

The recurrence relation (4.53) has been implemented in file `carmom_hbra.F90`.

4.6.3 δ-function

The operator of δ-function takes the form

$$\begin{aligned}\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}&=\bar{C}\left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left[\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}\delta(\boldsymbol{r}_C)\right]\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\\&=\frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}\left(\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}\delta(\boldsymbol{r}_C)\right)\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}},\end{aligned} \tag{4.55}$$

such that

$$\begin{aligned}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{L}_C\boldsymbol{L}_M\boldsymbol{m}\boldsymbol{n}\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij}&=\frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\\&\quad\times\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda+\boldsymbol{n}}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|+|\boldsymbol{n}|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}\delta(\boldsymbol{r}_C)\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right]\\&\equiv\frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda+\boldsymbol{n},\boldsymbol{L}_C,\boldsymbol{m}-\boldsymbol{L}_M\right]_{ij},\end{aligned} \tag{4.56}$$

where

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}'\right]_{ij} &= \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\kappa}_{\boldsymbol{R}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\lambda}_{\boldsymbol{R}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}^{\boldsymbol{L}'_C}_{\boldsymbol{C}}\left[\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\int \boldsymbol{r}^{\boldsymbol{m}'}_M\delta(\boldsymbol{r}_C)\mathrm{e}^{-p_{ij}r^2_\gamma}\mathrm{d}\boldsymbol{r}\right] \\ &= \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\kappa}_{\boldsymbol{R}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\lambda}_{\boldsymbol{R}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}^{\boldsymbol{L}'_C}_{\boldsymbol{C}}\left(\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\mathrm{e}^{-p_{ij}R^2_{C\gamma}}\boldsymbol{R}^{\boldsymbol{m}'}_{CM}\right).\end{aligned} \tag{4.57}$$

Therefore, $\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{L}_C\boldsymbol{L}_M\boldsymbol{m}\boldsymbol{n}\middle|\hat{O}^{\boldsymbol{K}_0\boldsymbol{L}_0}_{\ell_\beta}\right]_{ij}$ will be zero if centers $\boldsymbol{C}$ and $\boldsymbol{M}$ are the same.

Recovering the Cartesian Multipole Moments

The Cartesian multipole moments could be easily recovered (if there is any) via [2]

$$\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C,\boldsymbol{m}'{+}\boldsymbol{e}_\xi\right]_{ij} = (\boldsymbol{R}_{CM})_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}'\right]_{ij} + (\boldsymbol{L}'_C)_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C{-}\boldsymbol{e}_\xi,\boldsymbol{m}'\right]_{ij}, \tag{4.58}$$

by starting from $\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij}$, and implemented in file `delta_moment.F90`.

Recovering the HGTOs on Bra and Ket Centers

The recurrence relations of $\boldsymbol{l}'_\kappa$ and $\boldsymbol{l}'_\lambda$ take the same form [2]

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa{+}\boldsymbol{e}_\xi,\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} &= (\boldsymbol{R}_{C\kappa})_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} + (\boldsymbol{L}'_C)_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C{-}\boldsymbol{e}_\xi,\boldsymbol{0}\right]_{ij} \\ &\quad - \frac{(\boldsymbol{l}'_\kappa)_\xi}{2a_{i\kappa}}\left[\boldsymbol{l}'_\kappa{-}\boldsymbol{e}_\xi,\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij}, \end{aligned}\tag{4.59}$$

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa,\boldsymbol{l}'_\lambda{+}\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} &= (\boldsymbol{R}_{C\lambda})_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} + (\boldsymbol{L}'_C)_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C{-}\boldsymbol{e}_\xi,\boldsymbol{0}\right]_{ij} \\ &\quad - \frac{(\boldsymbol{l}'_\lambda)_\xi}{2b_{j\lambda}}\left[\boldsymbol{l}'_\kappa,\boldsymbol{l}'_\lambda{-}\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij}, \end{aligned}\tag{4.60}$$

and implemented in `delta_hket.F90`.

Recovering the Geometric Derivatives of δ-function

The recurrence relation of geometric derivatives of Dirac delta function could be easily obtained from

$$\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} = \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{\partial}^{\boldsymbol{L}'_C}_{\boldsymbol{C}}\left(\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\mathrm{e}^{-p_{ij}R^2_{C\gamma}}\right), \tag{4.61}$$

as

$$\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}'_C{+}\boldsymbol{e}_\xi,\boldsymbol{0}\right]_{ij} = -2p_{ij}\left\{(\boldsymbol{R}_{C\gamma})_\xi\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}'_C\boldsymbol{0}\right]_{ij} + (\boldsymbol{L}'_C)_\xi\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}'_C{-}\boldsymbol{e}_\xi,\boldsymbol{0}\right]_{ij}\right\}, \tag{4.62}$$

and

$$\left[\boldsymbol{0000}\right]_{ij} = \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\mathrm{e}^{-p_{ij}R^2_{C\gamma}}, \tag{4.63}$$

which is implemented in `delta_geom.F90`.

4.6.4 Nuclear Attraction Potential

The operator involved in nuclear attraction potential is

$$\begin{aligned}\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0} &= \bar{C}\left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left(\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}r_C^{-1}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}} \\ &= \frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}\left(\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}r_C^{-1}\right)\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}},\end{aligned} \tag{4.64}$$

such that

$$\begin{aligned}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{L}_C\boldsymbol{L}_M\boldsymbol{m}\boldsymbol{n}\middle|\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}\right]_{ij} &= \frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|} \\ &\times\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda+\boldsymbol{n}}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|+|\boldsymbol{n}|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}-\boldsymbol{L}_M}}{r_C}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right] \\ &\equiv\frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda{+}\boldsymbol{n},\boldsymbol{L}_C,\boldsymbol{m}{-}\boldsymbol{L}_M;0\right]_{ij},\end{aligned} \tag{4.65}$$

where

$$\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij} = \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}'_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}'_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}'}}{r_C}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right]. \tag{4.66}$$

Recovering the Cartesian Multipole Moments

The recrement in Cartesian multipole moments is

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C,\boldsymbol{m}'{+}\boldsymbol{e}_\xi;0\right]_{ij} &= \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}'_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}'_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}'+\boldsymbol{e}_\xi}}{r_C}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right] \\ &= \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}'_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}'_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C}\left[\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}'}(\boldsymbol{R}_{\gamma M}+\boldsymbol{r}_\gamma)_\xi}{r_C}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}\right] \\ &= \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}'_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}'_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C} \\ &\times\left[(\boldsymbol{R}_{\gamma M})_\xi\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}'}}{r_C}\mathrm{e}^{-p_{ij}r_\gamma^2}\mathrm{d}\boldsymbol{r}-\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\int\frac{\boldsymbol{r}_M^{\boldsymbol{m}'}}{r_C}\left(\frac{\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{e}_\xi}}{2p_{ij}}\mathrm{e}^{-p_{ij}r_\gamma^2}\right)\mathrm{d}\boldsymbol{r}\right],\end{aligned} \tag{4.67}$$

by taking $(\boldsymbol{R}_{\gamma M})_\xi$ outside and noticing that

$$\int f(x)g'(x)\mathrm{d}x = f(x)g(x)-\int f'(x)g(x)\mathrm{d}x, \tag{4.68}$$

we get the following recurrence relation to recover the Cartesian multipole moments

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C,\boldsymbol{m}'{+}\boldsymbol{e}_\xi;0\right]_{ij} &= (\boldsymbol{R}_{\gamma M})_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij}+\frac{1}{2p_{ij}}\Big\{(\boldsymbol{l}'_\kappa)_\xi\left[\boldsymbol{l}'_\kappa{-}\boldsymbol{e}_\xi,\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij} \\ &+(\boldsymbol{l}'_\lambda)_\xi\left[\boldsymbol{l}'_\kappa,\boldsymbol{l}'_\lambda{-}\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij}+m'_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C,\boldsymbol{m}'{-}\boldsymbol{e}_\xi;0\right]_{ij}-\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C{+}\boldsymbol{e}_\xi,\boldsymbol{m}';0\right]_{ij}\Big\},\end{aligned} \tag{4.69}$$

where we require a range of different orders HGTOs on bra and ket centers returned.

Recovering the HGTOs on Bra and Ket Centers

The recurrence relation of $\boldsymbol{l}'_\kappa$ is [2]

$$\begin{aligned}\left[\boldsymbol{l}'_\kappa+\boldsymbol{e}_\xi,\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}&=(\boldsymbol{R}_{\gamma\kappa})_\xi\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}-\frac{1}{2p_{ij}}\left\{\frac{(\boldsymbol{l}'_\kappa)_\xi b_{j\lambda}}{a_{i\kappa}}\left[\boldsymbol{l}'_\kappa-\boldsymbol{e}_\xi,\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}\right.\\&\left.-(\boldsymbol{l}'_\lambda)_\xi\left[\boldsymbol{l}'_\kappa,\boldsymbol{l}'_\lambda-\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}+\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C+\boldsymbol{e}_\xi,\boldsymbol{0};0\right]_{ij}\right\},\end{aligned}\tag{4.70}$$

and that of $\boldsymbol{l}'_\lambda$ [2]

$$\begin{aligned}\left[\boldsymbol{0},\boldsymbol{l}'_\lambda+\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}&=(\boldsymbol{R}_{\gamma\lambda})_\xi\left[\boldsymbol{0}\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}-\frac{1}{2p_{ij}}\left\{\frac{(\boldsymbol{l}'_\lambda)_\xi a_{i\kappa}}{b_{j\lambda}}\left[\boldsymbol{0},\boldsymbol{l}'_\lambda-\boldsymbol{e}_\xi,\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}\right.\\&\left.+\left[\boldsymbol{0}\boldsymbol{l}'_\lambda,\boldsymbol{L}'_C+\boldsymbol{e}_\xi,\boldsymbol{0};0\right]_{ij}\right\},\end{aligned}\tag{4.71}$$

which are implemented in file `nucpot_hbra.F90` and `nucpot_hket.F90`, respectively.

Recovering the Geometric Derivatives of Nuclear Potential Center

Finally, the geometric derivatives on operator center $\boldsymbol{C}$

$$\begin{aligned}\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij}&=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C}\left[\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\int\frac{\mathrm{e}^{-p_{ij}r^2_\gamma}}{r_C}\mathrm{d}\boldsymbol{r}\right]\\&=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\frac{2\pi}{p_{ij}}\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}'_C}F_0\left(p_{ij}R^2_{C\gamma}\right),\end{aligned}\tag{4.72}$$

and

$$\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{e}_\xi}F_{n_0}\left(p_{ij}R^2_{C\gamma}\right)=(\boldsymbol{R}_{C\gamma})_\xi(-2p_{ij})F_{n_0+1}\left(p_{ij}R^2_{C\gamma}\right),\tag{4.73}$$

so that

$$\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}'_C+\boldsymbol{e}_\xi,\boldsymbol{0};n_0\right]_{ij}=(\boldsymbol{R}_{C\gamma})_\xi\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}'_C\boldsymbol{0};n_0+1\right]_{ij}+(\boldsymbol{L}'_C)_\xi\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}'_C-\boldsymbol{e}_\xi,\boldsymbol{0};n_0+1\right]_{ij},\tag{4.74}$$

and

$$\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{0}\boldsymbol{0};n_0\right]_{ij}=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\frac{2\pi}{p_{ij}}(-2p_{ij})^{n_0}F_{n_0}\left(p_{ij}R^2_{C\gamma}\right),\tag{4.75}$$

where $F_{n_0}(T)$ is the n_0th order Boys function.

The recurrence relation (4.74) could be performed in a similar manner to those in Fig. 4.5 and 4.6, what we need to take into account is that recurrence relations for odd and even $|\boldsymbol{L}'_C|_{\min}$ are a bit different at $n_0+1=\frac{|\boldsymbol{L}'_C|_{\min}+\mod(|\boldsymbol{L}'_C|_{\min},2)}{2}$, as shown in Fig. 4.4. Such a recurrence relation (4.74) has been implemented in `nucpot_geom.F90`.

4.6.5 Inverse Square Distance Potential

$$\hat{O}^{\boldsymbol{K}_0\boldsymbol{L}_0}_{\ell_\beta}=\bar{C}\left(\boldsymbol{\partial}_{\boldsymbol{M}}^{\boldsymbol{L}_M}\boldsymbol{r}_M^{\boldsymbol{m}}\right)\left(\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C}r_C^{-2}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}}\tag{4.76}$$

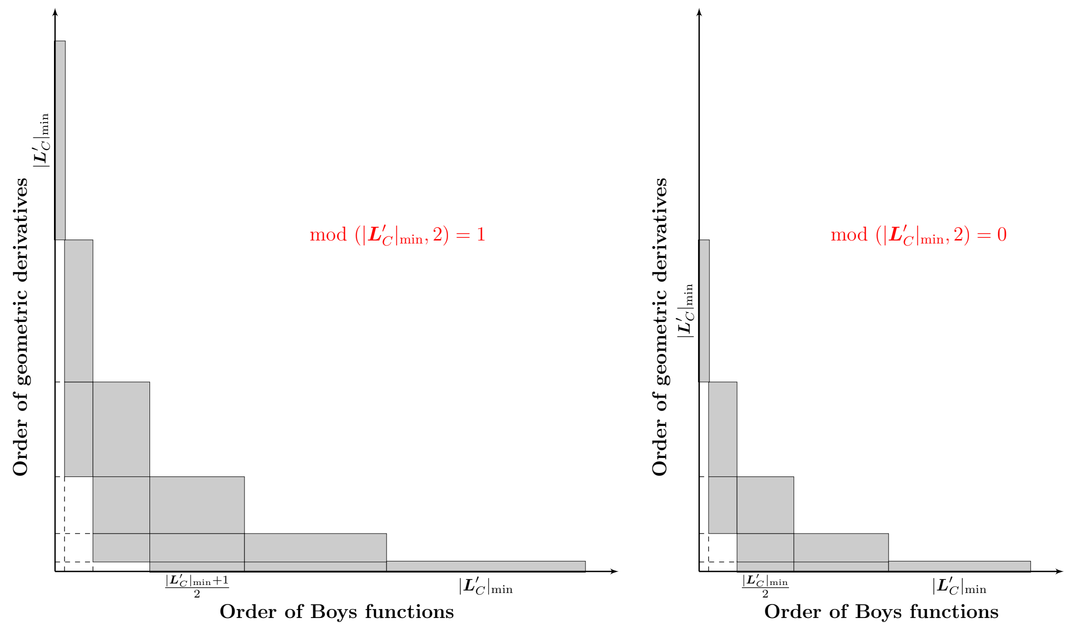

Order of geometric derivatives
$|\boldsymbol{L}'_C|_{\min}$
mod $(|\boldsymbol{L}'_C|_{\min}, 2) = 1$
$\frac{|\boldsymbol{L}'_C|_{\min}+1}{2}$
$|\boldsymbol{L}'_C|_{\min}$
Order of Boys functions
Order of geometric derivatives
$|\boldsymbol{L}'_C|_{\min}$
mod $(|\boldsymbol{L}'_C|_{\min}, 2) = 0$
$\frac{|\boldsymbol{L}'_C|_{\min}}{2}$
$|\boldsymbol{L}'_C|_{\min}$
Order of Boys functions

Figure 4.4: Procedure of recurrence relation (4.74) for odd and even $|\boldsymbol{L}'_C|_{\min}$.

4.6.6 Gaussian Charge Potential

The operator involved in Gaussian charge potential is

$$\begin{aligned}\hat{O}^{\boldsymbol{K}_0\boldsymbol{L}_0}_{\ell_\beta} &= \bar{C}\left(\boldsymbol{\partial}^{\boldsymbol{L}_M}_{\boldsymbol{M}}\boldsymbol{r}^{\boldsymbol{m}}_M\right)\left[\boldsymbol{\partial}^{\boldsymbol{L}_C}_{\boldsymbol{C}}\frac{\mathrm{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\right]\boldsymbol{\partial}^{\boldsymbol{n}}_{\boldsymbol{r}} \qquad (4.77)\\ &= \frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{r}^{\boldsymbol{m}-\boldsymbol{L}_M}_M\left[\boldsymbol{\partial}^{\boldsymbol{L}_C}_{\boldsymbol{C}}\frac{\mathrm{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\right]\frac{\boldsymbol{\partial}^{\boldsymbol{l}_\lambda}_{\boldsymbol{R}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}},\end{aligned}$$

such that

$$\begin{aligned}\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\boldsymbol{L}_C\boldsymbol{L}_M\boldsymbol{m}\boldsymbol{n}\middle|\hat{O}^{\boldsymbol{K}_0\boldsymbol{L}_0}_{\ell_\beta}\right]_{ij} &= \frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|} \qquad (4.78)\\ &\times\frac{\boldsymbol{\partial}^{\boldsymbol{l}_\kappa}_{\boldsymbol{R}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}^{\boldsymbol{l}_\lambda+\boldsymbol{n}}_{\boldsymbol{R}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|+|\boldsymbol{n}|}}\boldsymbol{\partial}^{\boldsymbol{L}_C}_{\boldsymbol{C}}\left[\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\int\frac{\boldsymbol{r}^{\boldsymbol{m}-\boldsymbol{L}_M}_M\mathrm{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\mathrm{e}^{-p_{ij}r^2_\gamma}\mathrm{d}\boldsymbol{r}\right]\\ &\equiv\frac{(-1)^{|\boldsymbol{L}_M|}\boldsymbol{m}!}{(\boldsymbol{m}-\boldsymbol{L}_M)!}\left[\boldsymbol{l}_\kappa,\boldsymbol{l}_\lambda{+}\boldsymbol{n},\boldsymbol{L}_C,\boldsymbol{m}{-}\boldsymbol{L}_M;0\right]_{ij},\end{aligned}$$

where

$$\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij} = \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\kappa}_{\boldsymbol{R}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}'_\kappa|}}\frac{\boldsymbol{\partial}^{\boldsymbol{l}'_\lambda}_{\boldsymbol{R}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}'_\lambda|}}\boldsymbol{\partial}^{\boldsymbol{L}'_C}_{\boldsymbol{C}}\left[\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\int\frac{\boldsymbol{r}^{\boldsymbol{m}'}_M\mathrm{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\mathrm{e}^{-p_{ij}r^2_\gamma}\mathrm{d}\boldsymbol{r}\right]. \qquad (4.79)$$

The evaluation of $\left[\boldsymbol{l}'_\kappa\boldsymbol{l}'_\lambda\boldsymbol{L}'_C\boldsymbol{m}';0\right]_{ij}$ could be performed as the case of nuclear attraction potential in Section 4.6.4, and we finally need to consider the geometric derivatives on operator center $\boldsymbol{C}$

$$\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}'_C\boldsymbol{0};0\right]_{ij} = \bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\boldsymbol{\partial}^{\boldsymbol{L}'_C}_{\boldsymbol{C}}\left[\mathrm{e}^{-u_{ij}R^2_{\kappa\lambda}}\int\frac{\mathrm{e}^{-p_{ij}r^2_\gamma}\mathrm{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\mathrm{d}\boldsymbol{r}\right]. \qquad (4.80)$$

By noticing that

$$\operatorname{erf}\left(\sqrt{\varrho}r_C\right)=\frac{2}{\sqrt{\pi}}\int_0^{\sqrt{\varrho}r_C}\mathrm{e}^{-t^2}\mathrm{d}t, \tag{4.81}$$

we get

$$\int\frac{\mathrm{e}^{-p_{ij}r_\gamma^2}\operatorname{erf}\left(\sqrt{\varrho}r_C\right)}{r_C}\mathrm{d}\boldsymbol{r}=\frac{2\pi}{\varrho}\tilde{F}_0\left(\frac{p_{ij}}{\varrho},p_{ij}R_{C\gamma}^2\right), \tag{4.82}$$

where we have introduced ($n_0\geq 0,\tau>0,T\geq 0$)

$$\begin{aligned}\tilde{F}_{n_0}(\tau,T)&=\int_0^1\frac{u^{2n_0}}{(\tau+u^2)^{n_0+\frac{3}{2}}}\mathrm{e}^{-T\frac{u^2}{\tau+u^2}}\mathrm{d}u \\ &=\frac{1}{\tau\sqrt{1+\tau}(1+\tau)^{n_0}}F_{n_0}\left(\frac{T}{1+\tau}\right)\end{aligned} \tag{4.83}$$

with $F_{n_0}(T)$ being the n_0th order Boys function.

We notice the same relation of $\tilde{F}_{n_0}(\tau,T)$ as that of Boys function

$$\boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{e}_\xi}\tilde{F}_{n_0}\left(\frac{p_{ij}}{\varrho},p_{ij}R_{C\gamma}^2\right)=(\boldsymbol{R}_{C\gamma})_\xi(-2p_{ij})\tilde{F}_{n_0+1}\left(\frac{p_{ij}}{\varrho},p_{ij}R_{C\gamma}^2\right), \tag{4.84}$$

so that

$$\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}_C'+\boldsymbol{e}_\xi,\boldsymbol{0};n_0\right]_{ij}=(\boldsymbol{R}_{C\gamma})_\xi\left[\boldsymbol{0}\boldsymbol{0}\boldsymbol{L}_C'\boldsymbol{0};n_0+1\right]_{ij}+(\boldsymbol{L}_C')_\xi\left[\boldsymbol{0}\boldsymbol{0},\boldsymbol{L}_C'-\boldsymbol{e}_\xi,\boldsymbol{0};n_0+1\right]_{ij}, \tag{4.85}$$

and

$$\begin{aligned}[\boldsymbol{0000};n_0]_{ij}&=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\frac{2\pi}{\varrho}(-2p_{ij})^{n_0}\tilde{F}_{n_0}\left(\frac{p_{ij}}{\varrho},p_{ij}R_{C\gamma}^2\right) \\ &=\bar{C}(-2b_{j\lambda})^{|\boldsymbol{n}|}\mathrm{e}^{-u_{ij}R_{\kappa\lambda}^2}\frac{2\pi}{p_{ij}}\sqrt{\frac{\varrho}{\varrho+p_{ij}}}\left(-2\frac{\varrho p_{ij}}{\varrho+p_{ij}}\right)^{n_0}F_{n_0}\left(\frac{\varrho p_{ij}}{\varrho+p_{ij}}R_{C\gamma}^2\right).\end{aligned} \tag{4.86}$$

4.6.7 Diamagnetic Spin-Orbit Coupling

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}=\bar{C}\left(\boldsymbol{\partial}_{\boldsymbol{C}_1}^{\boldsymbol{L}_{C1}}r_{C_1}^{-1}\right)\left(\boldsymbol{\partial}_{\boldsymbol{C}_2}^{\boldsymbol{L}_{C2}}r_{C_2}^{-1}\right)\boldsymbol{\partial}_{\boldsymbol{r}}^{\boldsymbol{n}} \tag{4.87}$$

4.6.8 Effective Core Potential

Effective core potential

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0\boldsymbol{L}_0}=U_L(r_C)+\sum_{l=0}^{L-1}\sum_{m=-l}^{l}|Y_{lm}\rangle[U_l(r_C)-U_L(r_C)]\langle Y_{lm}|, \tag{4.88}$$

where $Y_{lm}(\theta_C,\varphi_C)$ are real spherical harmonics centered on $\boldsymbol{C}$, $U_L(r_C)$ and $U_l(r_C)-U_L(r_C)$ ($l=0,\ldots,L-1$) are expressed as combinations of Gaussians

$$U_L(r_C)=\frac{N_{\text{core}}}{r_C}+\sum_k d_{kL}r_C^{n_{kL}}\mathrm{e}^{-c_{kL}r_C^2}, \tag{4.89}$$

$$U_l(r_C)-U_L(r_C)=\sum_k d_{kl}r_C^{n_{kl}}\mathrm{e}^{-c_{kl}r_C^2}. \tag{4.90}$$

The evaluation of integral over $\frac{N_{\text{core}}}{r_C}$ has been discussed in our recent work [2]. Therefore, the new types of integrals arising from the geometric derivative of integral (4.25) over the ECP operator (4.88) are [17?]

$$\chi_{\kappa\lambda} = 4\pi \int_0^{+\infty} \sum_k d_{kL} \langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_{kL}{+}2, 0\rangle \,\mathrm{d}r, \tag{4.91}$$

and

$$\gamma_{\kappa\lambda} = 4\pi \sum_{l=0}^{L-1} \sum_{m=-l}^{l} \sum_{\mu_1,\mu_2=-l}^{l} \left(\sum_{|\boldsymbol{l}_1|,|\boldsymbol{l}_2|=l} u_{\boldsymbol{l}_1}^{l\mu_1} u_{\boldsymbol{l}_2}^{l\mu_2} v_{\boldsymbol{l}_1}^{lm} v_{\boldsymbol{l}_2}^{lm} \right) \int_0^{+\infty} \sum_k d_{kl} \langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; ll, n_{kl}{+}2\rangle \,\mathrm{d}r, \tag{4.92}$$

where

$$\begin{aligned}
\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn'\rangle &= \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_\kappa)^{|\boldsymbol{l}_\kappa|}} \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_\lambda)^{|\boldsymbol{l}_\lambda|}} \boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} \left[\mathrm{e}^{-a_\kappa R_{C\kappa}^2 - b_\lambda R_{C\lambda}^2} r^{n'+n} \mathrm{e}^{-\alpha_{kL} r^2} \frac{M_n(R_S r)}{(R_S r)^n} \right], && (4.93)\\
\boldsymbol{R}_S &= -2(a_\kappa \boldsymbol{R}_{C\kappa} + b_\lambda \boldsymbol{R}_{C\lambda}), && (4.94)\\
\alpha_{kL} &= a_\kappa + b_\lambda + c_{kL}, && (4.95)
\end{aligned}$$

and

$$\begin{aligned}
\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_1 n_2 n'\rangle &= 4\pi \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_\kappa)^{|\boldsymbol{l}_\kappa|}} \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_\lambda)^{|\boldsymbol{l}_\lambda|}} \boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} \Big[\mathrm{e}^{-a_\kappa R_{C\kappa}^2 - b_\lambda R_{C\lambda}^2} Y_{n_1\mu_1}\left(\theta_{R_{S_\kappa}}, \varphi_{R_{S_\kappa}}\right) && (4.96)\\
&\quad \times Y_{n_2\mu_2}\left(\theta_{R_{S_\lambda}}, \varphi_{R_{S_\lambda}}\right) r^{n'} \mathrm{e}^{-\alpha_{kl} r^2} M_{n_1}(R_{S_\kappa} r) M_{n_2}(R_{S_\lambda} r) \Big], &&\\
\boldsymbol{R}_{S_\kappa} &= -2a_\kappa \boldsymbol{R}_{C\kappa}, && (4.97)\\
\boldsymbol{R}_{S_\lambda} &= -2b_\lambda \boldsymbol{R}_{C\lambda}, && (4.98)\\
\alpha_{kl} &= a_\kappa + b_\lambda + c_{kl}. && (4.99)
\end{aligned}$$

$u_{\boldsymbol{l}_1}^{l\mu_1}$, $u_{\boldsymbol{l}_2}^{l\mu_2}$, $v_{\boldsymbol{l}_1}^{lm}$ and $v_{\boldsymbol{l}_2}^{lm}$ in the integral $\gamma_{\kappa\lambda}$ are the transform coefficients between real spherical harmonics and unitary sphere polynomials, and could be evaluated analytically [17].

The function $M_n(x)$ in the integrands $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn'\rangle$ and $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_1 n_2 n'\rangle$ is a modified spherical Bessel function of the first kind

$$M_n(x) = x^n \left(\frac{1}{x} \frac{\mathrm{d}}{\mathrm{d}x} \right)^n \frac{\sinh x}{x}. \tag{4.100}$$

Notice the relationship between the real solid-harmonics $S_{lm}(r,\theta,\varphi)$ and real spherical harmonics [6]

$$S_{lm}(r,\theta,\varphi) = (-1)^m \sqrt{\frac{4\pi}{2l+1}} r^l Y_{lm}(\theta,\varphi), \tag{4.101}$$

the function $\mathrm{e}^{-a_\kappa R_{C\kappa}^2} Y_{n_1\mu_1}\left(\theta_{R_{S_\kappa}}, \varphi_{R_{S_\kappa}}\right)$ in the integrand $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_1 n_2 n'\rangle$, for instance, could be written as

$$\begin{aligned}
\mathrm{e}^{-a_\kappa R_{C\kappa}^2} Y_{n_1\mu_1}\left(\theta_{R_{S_\kappa}}, \varphi_{R_{S_\kappa}}\right) &= (-1)^{\mu_1} \sqrt{\frac{2n_1+1}{4\pi}} \frac{1}{R_{S_\kappa}^{n_1}} S_{n_1\mu_1}(\boldsymbol{R}_{S_\kappa}) \mathrm{e}^{-\frac{1}{4a_\kappa} R_{S_\kappa}^2} && (4.102)\\
&= (-1)^{n_1+\mu_1} \sqrt{\frac{2n_1+1}{4\pi}} \frac{1}{R_{S_\kappa}^{n_1}} \sum_{|\boldsymbol{n}_1'|=n_1} S_{\boldsymbol{n}_1'}^{n_1\mu_1} \left(\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{n}_1'} \mathrm{e}^{-a_\kappa R_{C\kappa}^2} \right), &&
\end{aligned}$$

where we have transformed the real solid-harmonics to the sum of Hermite Gaussians, and $S_{\boldsymbol{n}_1'}^{n_1\mu_1}$ are the the transformation coefficients between Cartesian and real solid-harmonic Gaussians [7]. Therefore, we could rewrite $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_1 n_2 n' \rangle$ as

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n_1 n_2 n' \right\rangle &= (-1)^{n_1+\mu_1+n_2+\mu_2}\sqrt{(2n_1+1)(2n_2+1)} \\ &\times \sum_{|\boldsymbol{n}_1'|=n_1}\sum_{|\boldsymbol{n}_2'|=n_2} S_{\boldsymbol{n}_1'}^{n_1\mu_1} S_{\boldsymbol{n}_2'}^{n_2\mu_2} \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \right\rangle,\end{aligned} \tag{4.103}$$

where

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \right\rangle &= \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_\kappa)^{|\boldsymbol{l}_\kappa|}} \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_\lambda)^{|\boldsymbol{l}_\lambda|}} \boldsymbol{\partial}_{\boldsymbol{C}}^{\boldsymbol{L}_C} \left[\left(\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{n}_1'} \mathrm{e}^{-a_\kappa R_{C\kappa}^2} \right) \left(\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{n}_2'} \mathrm{e}^{-b_\lambda R_{C\lambda}^2} \right) \right. \\ &\left. \times\, r^{n'+n_1+n_2} \mathrm{e}^{-\alpha_{kl} r^2} \frac{M_{n_1}(R_{S_\kappa} r)}{(R_{S_\kappa} r)^{n_1}} \frac{M_{n_2}(R_{S_\lambda} r)}{(R_{S_\lambda} r)^{n_2}} \right].\end{aligned} \tag{4.104}$$

We next discuss the evaluation of the integrands $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \rangle$ and $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \rangle$. Notice that [**?**]

$$\left(\frac{1}{x} \frac{\mathrm{d}}{\mathrm{d}x} \right)^m \frac{M_n(x)}{x^n} = \frac{M_{n+m}(x)}{x^{n+m}}, \tag{4.105}$$

we have, for instance

$$\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{e}_\xi} \left[\frac{M_n(R_S r)}{(R_S r)^n} \right] = a_\kappa r^2 (\boldsymbol{R}_S)_\xi \frac{M_{n+1}(R_S r)}{(R_S r)^{n+1}}. \tag{4.106}$$

The recurrence relations of $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \rangle$ and $\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \rangle$ could thus be obtained using the translational invariance [15, 16], Eq. (**??**), Eqs. (4.106) and (**??**)

$$\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C{+}\boldsymbol{e}_\xi; nn' \right\rangle = -2a_\kappa \left\langle \boldsymbol{l}_\kappa{+}\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \right\rangle - 2b_\lambda \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi, \boldsymbol{L}_C; nn' \right\rangle, \tag{4.107}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa{+}\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \right\rangle &= \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi, \boldsymbol{L}_C; nn' \right\rangle - (\boldsymbol{R}_{\kappa\lambda})_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \right\rangle \\ &- \frac{(\boldsymbol{l}_\kappa)_\xi}{2a_\kappa} \left\langle \boldsymbol{l}_\kappa{-}\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \right\rangle + \frac{(\boldsymbol{l}_\lambda)_\xi}{2b_\lambda} \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{-}\boldsymbol{e}_\xi, \boldsymbol{L}_C; nn' \right\rangle,\end{aligned} \tag{4.108}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi, \boldsymbol{L}_C; nn' \right\rangle &= (\boldsymbol{R}_{C\lambda})_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; nn' \right\rangle - \frac{(\boldsymbol{l}_\lambda)_\xi}{2b_\lambda} \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{-}\boldsymbol{e}_\xi, \boldsymbol{L}_C; nn' \right\rangle \\ &+ (\boldsymbol{L}_C)_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C{-}\boldsymbol{e}_\xi; nn' \right\rangle + \frac{(\boldsymbol{R}_S)_\xi}{2} \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C; n{+}1, n'{+}1 \right\rangle \\ &+ \frac{(\boldsymbol{l}_\kappa)_\xi}{2} \left\langle \boldsymbol{l}_\kappa{-}\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C; n{+}1, n'{+}1 \right\rangle + \frac{(\boldsymbol{l}_\lambda)_\xi}{2} \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{-}\boldsymbol{e}_\xi, \boldsymbol{L}_C; n{+}1, n'{+}1 \right\rangle \\ &- (\boldsymbol{L}_C)_\xi (a_\kappa + b_\lambda) \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C{-}\boldsymbol{e}_\xi; n{+}1, n'{+}1 \right\rangle,\end{aligned} \tag{4.109}$$

and

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C{+}\boldsymbol{e}_\xi, \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \right\rangle = &- 2a_\kappa \left\langle \boldsymbol{l}_\kappa{+}\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \right\rangle \\ &- 2b_\lambda \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda{+}\boldsymbol{e}_\xi, \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n' \right\rangle,\end{aligned} \tag{4.110}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa+\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle =& \frac{1}{2a_\kappa}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1'+\boldsymbol{e}_\xi, \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &+ \frac{(\boldsymbol{R}_{S_\kappa})_\xi}{2}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1+1, n_2, n'+1\right\rangle \\ &+ (\boldsymbol{l}_\kappa)_\xi \left\langle \boldsymbol{l}_\kappa-\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1+1, n_2, n'+1\right\rangle \\ &- 2a_\kappa (\boldsymbol{L}_C)_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C-\boldsymbol{e}_\xi, \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1+1, n_2, n'+1\right\rangle,\end{aligned} \tag{4.111}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda+\boldsymbol{e}_\xi, \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle =& \frac{1}{2b_\lambda}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1', \boldsymbol{n}_2'+\boldsymbol{e}_\xi; n_1 n_2 n'\right\rangle \\ &+ \frac{(\boldsymbol{R}_{S_\lambda})_\xi}{2}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1, n_2+1, n'+1\right\rangle \\ &+ (\boldsymbol{l}_\lambda)_\xi \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda-\boldsymbol{e}_\xi, \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1, n_2+1, n'+1\right\rangle \\ &- 2b_\lambda (\boldsymbol{L}_C)_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C-\boldsymbol{e}_\xi, \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1, n_2+1, n'+1\right\rangle,\end{aligned} \tag{4.112}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1'+\boldsymbol{e}_\xi, \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle =& 2a_\kappa (\boldsymbol{R}_{C\kappa})_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &- (\boldsymbol{l}_\kappa)_\xi \left\langle \boldsymbol{l}_\kappa-\boldsymbol{e}_\xi, \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &+ 2(\boldsymbol{L}_C)_\xi a_\kappa \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C-\boldsymbol{e}_\xi, \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &- 2(\boldsymbol{n}_1')_\xi a_\kappa \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1'-\boldsymbol{e}_\xi, \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle,\end{aligned} \tag{4.113}$$

$$\begin{aligned}\left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1', \boldsymbol{n}_2'+\boldsymbol{e}_\xi; n_1 n_2 n'\right\rangle =& 2b_\lambda (\boldsymbol{R}_{C\lambda})_\xi \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &- (\boldsymbol{l}_\lambda)_\xi \left\langle \boldsymbol{l}_\kappa, \boldsymbol{l}_\lambda-\boldsymbol{e}_\xi, \boldsymbol{L}_C \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &+ 2(\boldsymbol{L}_C)_\xi b_\lambda \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda, \boldsymbol{L}_C-\boldsymbol{e}_\xi, \boldsymbol{n}_1' \boldsymbol{n}_2'; n_1 n_2 n'\right\rangle \\ &- 2(\boldsymbol{n}_2')_\xi b_\lambda \left\langle \boldsymbol{l}_\kappa \boldsymbol{l}_\lambda \boldsymbol{L}_C, \boldsymbol{n}_1', \boldsymbol{n}_2'-\boldsymbol{e}_\xi; n_1 n_2 n'\right\rangle.\end{aligned} \tag{4.114}$$

The integrals $\chi_{\kappa\lambda}$ and $\gamma_{\kappa\lambda}$ could finally be evaluated using adaptive quadrature with the knowledge of the values of integrands

$$\begin{aligned}\left\langle \mathbf{000}; nn'\right\rangle &= \mathrm{e}^{-a_\kappa R_{C\kappa}^2 - b_\lambda R_{C\lambda}^2} r^{n'+n} \mathrm{e}^{-\alpha_{kL} r^2} \frac{M_n(R_S r)}{(R_S r)^n} \\ &= \frac{r^{n'}\left(R_S^{-1}\right)^n}{\mathrm{e}^{a_\kappa (r-R_{C\kappa})^2 + b_\lambda (r-R_{C\lambda})^2 + c_{kL} r^2} \mathrm{e}^{(R_{S_\kappa}+R_{S_\lambda}-R_S) r}} M_n(R_S r) \mathrm{e}^{-R_S r}\end{aligned} \tag{4.115}$$

and

$$\begin{aligned}\left\langle \mathbf{00000}; n_1 n_2 n'\right\rangle &= \mathrm{e}^{-a_\kappa R_{C\kappa}^2 - b_\lambda R_{C\lambda}^2} r^{n'+n_1+n_2} \mathrm{e}^{-\alpha_{kl} r^2} \frac{M_{n_1}(R_{S_\kappa} r)}{(R_{S_\kappa} r)^{n_1}} \frac{M_{n_2}(R_{S_\lambda} r)}{(R_{S_\lambda} r)^{n_2}} \\ &= \frac{r^{n'}\left(R_{S_\kappa}^{-1}\right)^{n_1}\left(R_{S_\lambda}^{-1}\right)^{n_2}}{\mathrm{e}^{a_\kappa (r-R_{C\kappa})^2 + b_\lambda (r-R_{C\lambda})^2 + c_{kl} r^2}} M_{n_1}(R_{S_\kappa} r) \mathrm{e}^{-R_{S_\kappa} r} M_{n_2}(R_{S_\lambda} r) \mathrm{e}^{-R_{S_\lambda} r}\end{aligned} \tag{4.116}$$

at quadrature points [17]. The evaluation of the scaled modified spherical Bessel function of the first kind $M_n(z)\mathrm{e}^{-z}$ has been discussed in Ref. [17].

4.6.9 Model Core Potential (Version 1)

Model core potential (Version 1)

$$\hat{O}_{\ell_\beta}^{\boldsymbol{K}_0 \boldsymbol{L}_0} = V_{\mathrm{mp}}(r_C) + \hat{\Omega}, \tag{4.117}$$

where

$$V_{\mathrm{mp}}(r_C) = -\frac{Z - N_{\mathrm{core}}}{r_C}\sum_l A_l r_C^{n_l}\mathrm{e}^{-\alpha_l r_C^2}, \tag{4.118}$$

and

$$\hat{\Omega} = -\sum_k f_k\epsilon_k \left|\varphi_k(\boldsymbol{r}_C)\right\rangle \left\langle\varphi_k(\boldsymbol{r}_C)\right|, \tag{4.119}$$

with $1 \leq f_k \leq 2$ being adjustable parameters, and ϵ_k the eigenvalue of the k'th core orbital. $\varphi_k(\boldsymbol{r}_C)$ is the k'th core orbital, represented by real solid-harmonic Gaussian functions.

4.6.10 Overlap Distribution

Starting from Eq. (4.2), we could get the total and/or partial derivatives (evaluated at zero fields) of overlap distribution of two contracted rotational LAOs (2) as

$$\begin{aligned}\Omega_{\kappa\lambda}(\boldsymbol{r}) &= \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}}\binom{\boldsymbol{K}}{\boldsymbol{K}'}\binom{\boldsymbol{L}}{\boldsymbol{L}'}\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}\Big\{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_1+\boldsymbol{K}'}\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_1+\boldsymbol{L}'}\omega_\kappa^*(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\right]_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}} \\ &\times\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\left[\boldsymbol{\partial}_{\boldsymbol{B}}^{\boldsymbol{K}_2+\boldsymbol{K}''}\boldsymbol{\partial}_{\boldsymbol{J}}^{\boldsymbol{L}_2+\boldsymbol{L}''}\omega_\lambda(\boldsymbol{r};\boldsymbol{B},\boldsymbol{J})\right]_{\boldsymbol{B},\boldsymbol{J}=\boldsymbol{0}}\Big\}, \\ &= \sum_{\boldsymbol{K}'=\boldsymbol{0}}^{\boldsymbol{K}}\sum_{\boldsymbol{L}'=\boldsymbol{0}}^{\boldsymbol{L}}\binom{\boldsymbol{K}}{\boldsymbol{K}'}\binom{\boldsymbol{L}}{\boldsymbol{L}'}\prod^{N_g}\boldsymbol{\partial}_{\boldsymbol{R}_g}^{\boldsymbol{L}_g}\left(\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_1|+|\boldsymbol{K}'|}(-\mathrm{i})^{|\boldsymbol{L}_1|+|\boldsymbol{L}'|}\left(-\tfrac{\mathrm{i}}{2}\right)^{|\boldsymbol{K}_2|+|\boldsymbol{K}''|}\mathrm{i}^{|\boldsymbol{L}_2|+|\boldsymbol{L}''|} \\ &\times\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{L}_\kappa}\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{L}_\lambda}\Big\{(\boldsymbol{R}_{\kappa G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_1+\boldsymbol{K}'}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\kappa O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_1+\boldsymbol{L}'}\chi_\kappa(\boldsymbol{r}) \\ &\times(\boldsymbol{R}_{\lambda G}\times\boldsymbol{r}_P)^{\boldsymbol{K}_2+\boldsymbol{K}''}\left[\mathbf{I}^{-T}(\boldsymbol{R}_{\lambda O}\times\boldsymbol{r}_P)\right]^{\boldsymbol{L}_2+\boldsymbol{L}''}\chi_\lambda(\boldsymbol{r})\Big\},\end{aligned} \tag{4.120}$$

which, by applying the recurrence relations (4.8)-(4.23), could be obtained from the product of two Hermite Gaussians

$$\left[\boldsymbol{l}_\kappa\boldsymbol{l}_\lambda\middle|\Omega\right]_{ij} = \frac{\boldsymbol{\partial}_{\boldsymbol{R}_\kappa}^{\boldsymbol{l}_\kappa}}{(2a_{i\kappa})^{|\boldsymbol{l}_\kappa|}}\frac{\boldsymbol{\partial}_{\boldsymbol{R}_\lambda}^{\boldsymbol{l}_\lambda}}{(2b_{j\lambda})^{|\boldsymbol{l}_\lambda|}}\left[\exp(-a_{i\kappa}r_\kappa^2)\exp(-b_{j\lambda}r_\lambda^2)\right] = H_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r})H_{j\lambda}^{\boldsymbol{l}_\lambda}(\boldsymbol{r}). \tag{4.121}$$

The recurrence relations of $H_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r})$ and $H_{j\lambda}^{\boldsymbol{l}_\lambda}(\boldsymbol{r})$ are trivial [7]

$$H_{i\kappa}^{\boldsymbol{l}_\kappa+\boldsymbol{e}_\xi}(\boldsymbol{r}) = (\boldsymbol{r}_\kappa)_\xi H_{i\kappa}^{\boldsymbol{l}_\kappa}(\boldsymbol{r}) - \frac{(\boldsymbol{l}_\kappa)_\xi}{2a_{i\kappa}}H_{i\kappa}^{\boldsymbol{l}_\kappa-\boldsymbol{e}_\xi}(\boldsymbol{r}), \tag{4.122}$$

$$H_{j\lambda}^{\boldsymbol{l}_\lambda+\boldsymbol{e}_\xi}(\boldsymbol{r}) = (\boldsymbol{r}_\lambda)_\xi H_{j\lambda}^{\boldsymbol{l}_\lambda}(\boldsymbol{r}) - \frac{(\boldsymbol{l}_\lambda)_\xi}{2b_{j\lambda}}H_{j\lambda}^{\boldsymbol{l}_\lambda-\boldsymbol{e}_\xi}(\boldsymbol{r}), \tag{4.123}$$

which are implemented in `prim_hgto_odist.F90` by starting from

$$H_{i\kappa}^{\boldsymbol{0}}(\boldsymbol{r})H_{j\lambda}^{\boldsymbol{0}}(\boldsymbol{r}) = \exp(-a_{i\kappa}r_\kappa^2)\exp(-b_{j\lambda}r_\lambda^2). \tag{4.124}$$

The value of individual HGTOs could also be easily evaluated from above recurrence relation, which is implemented in `prim_hgto_value.F90`.

4.7 Quadrature in Gen1Int

Quadrature is used by diamagnetic spin-orbit coupling, and effective core potential ...

4.8 Basis Sets in Gen1Int

4.8.1 Normalization of Contracted Spherical Gaussians

Change the notations later on and adds a table for the subroutine ...

$$\chi_\kappa(\boldsymbol{r}) = \theta_\kappa(\boldsymbol{r}_\kappa)\rho_\kappa(r_\kappa), \tag{4.125}$$

where $\rho_\kappa(r_\kappa)$ is a contracted Gaussian

$$\rho_\kappa(r_\kappa) = \sum_i w_{i\kappa}\exp(-a_{i\kappa}r_\kappa^2), \tag{4.126}$$

This section mainly follows the book [6]. We rewrite the spherical Gaussian as

$$\chi_{\alpha_{nl}lm} = R_{\alpha_{nl}l}(r)Y_{lm}(\theta,\phi), \tag{4.127}$$

$$R_{\alpha_{nl}l}(r) = N_{\alpha_{nl}l}(\sqrt{2\alpha_{nl}}r)^l\exp(-\alpha_{nl}r^2), \tag{4.128}$$

where $R_{\alpha_{nl}l}(r)$ is the radial function. Y_{lm} are spherical-harmonic angular functions, with l and m the total- and z-projected angular momentum quantum numbers, respectively

$$Y_{lm}(\theta,\phi) = \sqrt{\frac{2l+1}{4\pi}\frac{(l-m)!}{(l+m)!}}P_l^m(\cos\theta)\exp(im\phi) \tag{4.129}$$

which are orthonormal

$$\int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi}Y_{lm}Y_{l'm'}^*\sin\theta\mathrm{d}\theta\mathrm{d}\phi = \delta_{ll'}\delta_{mm'}, \tag{4.130}$$

The complex solid harmonics in Racah's normalization

$$C_{lm}(r,\theta,\phi) = \sqrt{\frac{4\pi}{2l+1}}r^lY_{lm}(\theta,\phi), \tag{4.131}$$

and

$$\int_{\theta=0}^{\pi}\int_{\phi=0}^{2\pi}C_{lm}C_{lm}^*\sin\theta\mathrm{d}\theta\mathrm{d}\phi = \frac{4\pi}{2l+1}r^{2l} \tag{4.132}$$

The following expression defines the real-valued solid harmonics

$$\begin{pmatrix} S_{lm} \\ S_{l,-m}\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}(-1)^m & 1 \\ -(-1)^m\mathrm{i} & \mathrm{i}\end{pmatrix}\begin{pmatrix} C_{lm} \\ C_{l,-m}\end{pmatrix}, \qquad m>0, \tag{4.133}$$

and for $m=0$

$$S_{l0} = C_{l0}. \tag{4.134}$$

Therefore, the normalization constant of real-valued solid harmonics is $\frac{1}{2}\sqrt{\frac{2l+1}{\pi}}$. The normalization constant $N_{\alpha_{nl}l}$ is

$$N_{\alpha_{nl}l} = \frac{2(2\alpha_{nl})^{3/4}}{\pi^{1/4}}\sqrt{\frac{2^l}{(2l+1)!!}}. \tag{4.135}$$

For contracted spherical Gaussians, we have

$$R_l^{\mathrm{contr}}(r) = \sum_{\alpha_{nl}}N_{\alpha_{nl}l}(\sqrt{2\alpha_{nl}}r)^lC(\alpha_{nl})\exp(-\alpha_{nl}r^2), \tag{4.136}$$

where the summation runs over the set of (primitive) Gaussian exponents α_{nl}, and $C(\alpha_{nl})$ is the corresponding contraction coefficient.

The radial overlap between two spherical Gaussians is

$$S_{\alpha_1\alpha_2 l} = \int_0^\infty R_{\alpha_1 l}(r)R_{\alpha_2 l}(r)r^2\mathrm{d}r = \left(\frac{\sqrt{4\alpha_1\alpha_2}}{\alpha_1+\alpha_2}\right)^{l+3/2}, \tag{4.137}$$

we then get the norm of $R_l^{\mathrm{contr}}(r)$ as

$$\sqrt{\int_0^\infty [R_l^{\mathrm{contr}}(r)]^2 r^2\mathrm{d}r} = \sqrt{\sum_{\alpha_i,\alpha_j} C(\alpha_i)C(\alpha_j)\left(\frac{\sqrt{4\alpha_i\alpha_j}}{\alpha_i+\alpha_j}\right)^{l+3/2}}. \tag{4.138}$$

Therefore, the normalized radical part is

$$\bar{R}_l^{\mathrm{contr}}(r) = \sum_{\alpha_{nl}} \frac{N_{\alpha_{nl}l}(\sqrt{2\alpha_{nl}}r)^l C(\alpha_{nl})}{\sqrt{\sum_{\alpha_i,\alpha_j} C(\alpha_i)C(\alpha_j)\left(\frac{\sqrt{4\alpha_i\alpha_j}}{\alpha_i+\alpha_j}\right)^{l+3/2}}}\exp(-\alpha_{nl}r^2). \tag{4.139}$$

In other words, the normalized contraction coefficient is

$$\bar{C}(\alpha_{nl}) = \frac{N_{\alpha_{nl}l}(\sqrt{2\alpha_{nl}})^l C(\alpha_{nl})}{\sqrt{\sum_{\alpha_i,\alpha_j} C(\alpha_i)C(\alpha_j)\left(\frac{\sqrt{4\alpha_i\alpha_j}}{\alpha_i+\alpha_j}\right)^{l+3/2}}} \tag{4.140}$$

$$= \left(\frac{2}{\pi}\right)^{1/4}\frac{(4\alpha_{nl})^{l/2+3/4}}{\sqrt{(2l+1)!!}}\frac{C(\alpha_{nl})}{\sqrt{\sum_{\alpha_i,\alpha_j} C(\alpha_i)C(\alpha_j)\left(\frac{\sqrt{4\alpha_i\alpha_j}}{\alpha_i+\alpha_j}\right)^{l+3/2}}}. \tag{4.141}$$

The normalization of contracted spherical Gaussians has been implemented in subroutine `norm_contr_sgto`.

4.8.2 Normalization of Contracted Cartesian Gaussians

Change the notations later on and adds a table for the subroutine ...

In subroutine `norm_contr_cgto`, we have provided the normalization of contracted Cartesian Gaussians following the same procedure in DALTON subroutine `NRMORB` (so that the diagonal elements of overlap matrix scales as $(2l+1)!!$). After that, the normalized contraction coefficient of contracted Cartesian Gaussians is

$$\bar{C}(\alpha_{nl}) = \left(\frac{1}{2\pi}\right)^{3/4}(4\alpha_{nl})^{l/2+3/4}\frac{C(\alpha_{nl})}{\sqrt{\sum_{\alpha_i,\alpha_j} C(\alpha_i)C(\alpha_j)\left(\frac{\sqrt{4\alpha_i\alpha_j}}{\alpha_i+\alpha_j}\right)^{l+3/2}}}, \tag{4.142}$$

seems to replace $N_{\alpha_{nl}l}(\sqrt{2\alpha_{nl}})^l$ with $(\frac{2\alpha_{nl}}{\pi})^{3/4}(4\alpha_{nl})^{l/2}$ in Eq. (4.140).

4.8.3 Transformation between Spherical and Hermite Gaussians

This section is not readable yet ...

Describe subroutine `hgto_to_sgto` ...

solid harmonics (or harmonic polynomials)

$$\mathcal{Y}_{lm}(r,\theta,\phi) = r^l Y_{lm}(\theta,\phi) \tag{4.143}$$

as the product of spherical harmonics with the monomials r^l.

Email from Andreas ...

```
They were no easy task, I remember. I think I eventually figured them out
by inspecting Mathematica's SphericalHarmonicY (t=theta, p=phi):

      Ylm(t,p) = sqrt[(2l+1)/4pi * (l-m)!/(l+m)!] Plm(cos(t)) exp(imp)

where the first factor is the normalization constant, Plm is the
"associated Legendre function", and the replacements cos(t) =>
z/sqrt(xx+yy+zz), exp(imp) => (x+iy)^m. I've attached the Mathematica
"notebook" I worked in back then (warning: very messy).
```

and

```
There *may* be some tric involved, yes, because the spherical harmonic
polynomials satisfy

   (d2/dx2 + d2/dy2 + d2/dz2) r^l Ylm(r/|r|) = 0,

thus that contribution to <a|E_kin|b> disappears (There will still be
contributions involing the radial factor exp(-a r^2)).
```

The transformation of kinetic energy integrals need special consideration ...

4.8.4 Recovering Partial Geometric Derivatives from Hermite Gaussians

Describe how to perform Eqs. (4.20) and (4.21) as ...

$$\{\boldsymbol{L}_\kappa \boldsymbol{l}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}} = (2a_{i\kappa})^{|\boldsymbol{L}_\kappa|}\{\boldsymbol{0},\boldsymbol{l}_\kappa+\boldsymbol{L}_\kappa\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}}, \tag{4.144}$$

$$\{\boldsymbol{L}_\lambda \boldsymbol{l}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}} = (2b_{j\lambda})^{|\boldsymbol{L}_\lambda|}\{\boldsymbol{0},\boldsymbol{l}_\lambda+\boldsymbol{L}_\lambda\}_{\boldsymbol{K}_0\boldsymbol{L}_0/ij,\mathrm{Herm}}. \tag{4.145}$$

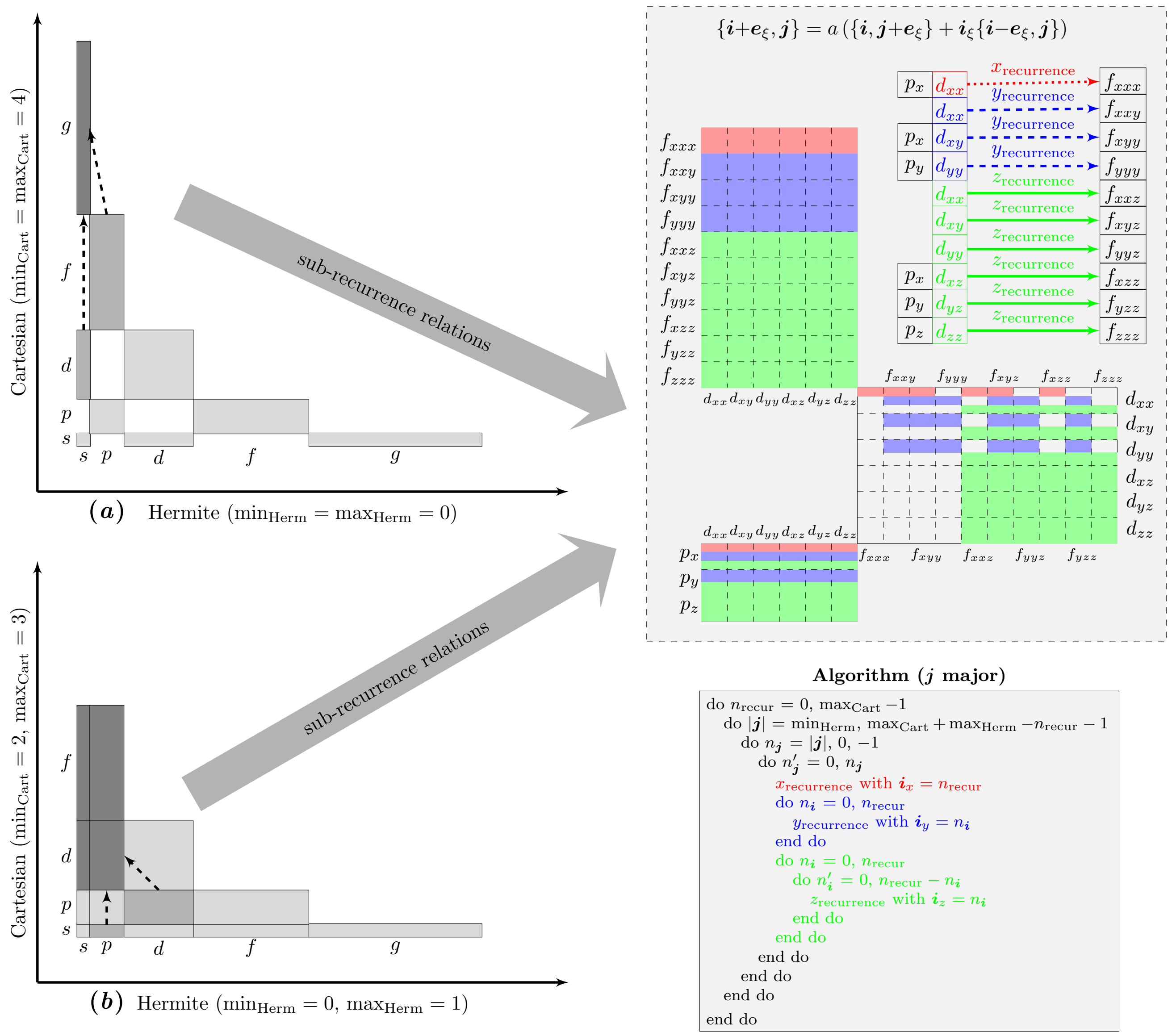
4.8.5 Transformation between Cartesian and Hermite Gaussians

The integrals using primitive Cartesian Gaussians (4.15) could be recovered using Eqs. (4.16) and (4.17), which could be written as the following general form

$$\{\boldsymbol{i}+\boldsymbol{e}_\xi,\boldsymbol{j}\} = a\left(\{\boldsymbol{i},\boldsymbol{j}+\boldsymbol{e}_\xi\} + \boldsymbol{i}_\xi\{\boldsymbol{i}-\boldsymbol{e}_\xi,\boldsymbol{j}\}\right). \tag{4.146}$$

The procedure of evaluating the above recurrence relation is given in Fig. 4.5, in which (a) describes the procedure of returning a specific order Cartesian and Hermite Gaussians (geometric derivative), while (b) depicts the procedure for a range of orders of Cartesian and Hermite Gaussians. These two different procedures are respectively used for non London atomic orbitals (non-LAOs) and LAOs, and implemented in files `hgto_to_cgto.F90` and `hgto_to_lcgto.F90`.

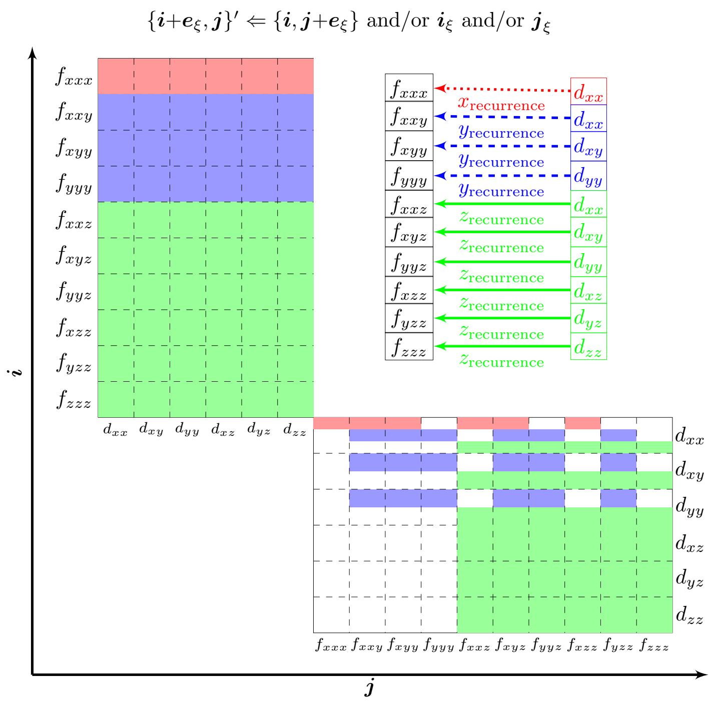
Furthermore, a more detailed version of $\{\boldsymbol{i},\boldsymbol{j}+\boldsymbol{e}_\xi\} \Rightarrow \{\boldsymbol{i}+\boldsymbol{e}_\xi,\boldsymbol{j}\}$ is given in Fig. 4.6.


$\{\boldsymbol{i}+\boldsymbol{e}_\xi,\boldsymbol{j}\} = a\left(\{\boldsymbol{i},\boldsymbol{j}+\boldsymbol{e}_\xi\} + \boldsymbol{i}_\xi\{\boldsymbol{i}-\boldsymbol{e}_\xi,\boldsymbol{j}\}\right)$
$x_{\text{recurrence}}$
$y_{\text{recurrence}}$
$z_{\text{recurrence}}$
sub-recurrence relations
sub-recurrence relations
Cartesian ($\min_{\text{Cart}} = \max_{\text{Cart}} = 4$)
$\boldsymbol{(a)}$ Hermite ($\min_{\text{Herm}} = \max_{\text{Herm}} = 0$)
Cartesian ($\min_{\text{Cart}} = 2$, $\max_{\text{Cart}} = 3$)
$\boldsymbol{(b)}$ Hermite ($\min_{\text{Herm}} = 0$, $\max_{\text{Herm}} = 1$)
s p d f g
f_{xxx} f_{xxy} f_{xyy} f_{yyy} f_{xxz} f_{xyz} f_{yyz} f_{xzz} f_{yzz} f_{zzz}
d_{xx} d_{xy} d_{yy} d_{xz} d_{yz} d_{zz}
p_x p_y p_z

Algorithm (j major)

```
do $n_{\text{recur}} = 0, \max_{\text{Cart}} -1$
   do $|\boldsymbol{j}| = \min_{\text{Herm}}, \max_{\text{Cart}} + \max_{\text{Herm}} - n_{\text{recur}} - 1$
      do $n_{\boldsymbol{j}} = |\boldsymbol{j}|, 0, -1$
         do $n'_{\boldsymbol{j}} = 0, n_{\boldsymbol{j}}$
            $x_{\text{recurrence}}$ with $\boldsymbol{i}_x = n_{\text{recur}}$
            do $n_{\boldsymbol{i}} = 0, n_{\text{recur}}$
               $y_{\text{recurrence}}$ with $\boldsymbol{i}_y = n_{\boldsymbol{i}}$
            end do
            do $n_{\boldsymbol{i}} = 0, n_{\text{recur}}$
               do $n'_{\boldsymbol{i}} = 0, n_{\text{recur}} - n_{\boldsymbol{i}}$
                  $z_{\text{recurrence}}$ with $\boldsymbol{i}_z = n_{\boldsymbol{i}}$
               end do
            end do
         end do
      end do
   end do
end do
```

Figure 4.5: Procedure of transformation between Cartesian and Hermite Gaussians.


$\{\boldsymbol{i}+\boldsymbol{e}_\xi, \boldsymbol{j}\}' \Leftarrow \{\boldsymbol{i}, \boldsymbol{j}+\boldsymbol{e}_\xi\}$ and/or $\boldsymbol{i}_\xi$ and/or $\boldsymbol{j}_\xi$
f_{xxx} f_{xxy} f_{xyy} f_{yyy} f_{xxz} f_{xyz} f_{yyz} f_{xzz} f_{yzz} f_{zzz}
d_{xx} d_{xy} d_{yy} d_{xz} d_{yz} d_{zz}
$x$recurrence
$y$recurrence
$z$recurrence
$\boldsymbol{i}$
$\boldsymbol{j}$

Algorithm ($\boldsymbol{i}$ major)

```
$w_{\boldsymbol{i}_\xi} = 1$
$w_{\boldsymbol{i}} = 1$
$w_{\boldsymbol{j}_\xi} = 0$
$w_{\boldsymbol{j}} = 0$
do $n_{\boldsymbol{j}} = |\boldsymbol{j}|, 0, -1$
  do $n'_{\boldsymbol{j}} = n_{\boldsymbol{j}}, 0, -1$
    $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
    $w_{\boldsymbol{j}} = w_{\boldsymbol{j}} + 1$
    $\{w_{\boldsymbol{j}_\xi}, w_{\boldsymbol{i}}\} \overset{x\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{j}}, w_{\boldsymbol{i}_\xi}\}'$ with $\boldsymbol{i}_x = |\boldsymbol{i}|$ and $\boldsymbol{j}_x = n'_{\boldsymbol{j}}$
  end do
  $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
end do
do $n_{\boldsymbol{i}} = 0, |\boldsymbol{i}|$
  $w_{\boldsymbol{i}_\xi} = w_{\boldsymbol{i}_\xi} + 1$
  $w_{\boldsymbol{j}_\xi} = 1$
  $w_{\boldsymbol{j}} = 0$
  do $n_{\boldsymbol{j}} = |\boldsymbol{j}|, 0, -1$
    do $n'_{\boldsymbol{j}} = 0, n_{\boldsymbol{j}}$
      $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
      $w_{\boldsymbol{j}} = w_{\boldsymbol{j}} + 1$
      $\{w_{\boldsymbol{j}_\xi}, w_{\boldsymbol{i}} + n_{\boldsymbol{i}}\} \overset{y\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{j}}, w_{\boldsymbol{i}_\xi}\}'$ with $\boldsymbol{i}_y = n_{\boldsymbol{i}}$ and $\boldsymbol{j}_y = n'_{\boldsymbol{j}}$
    end do
    $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
  end do
end do
$w_{\boldsymbol{i}} = w_{\boldsymbol{i}} - 1$
do $n_{\boldsymbol{i}} = 0, |\boldsymbol{i}|$
  do $n'_{\boldsymbol{i}} = 0, |\boldsymbol{i}| - n_{\boldsymbol{i}}$
    $w_{\boldsymbol{i}} = w_{\boldsymbol{i}} + 1$
    $w_{\boldsymbol{i}_\xi} = w_{\boldsymbol{i}_\xi} + 1$
    $w_{\boldsymbol{j}_\xi} = |\boldsymbol{j}| + 2$
    $w_{\boldsymbol{j}} = 0$
    do $n_{\boldsymbol{j}} = 0, |\boldsymbol{j}|$
      do $n'_{\boldsymbol{j}} = n_{\boldsymbol{j}}, |\boldsymbol{j}|$
        $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
        $w_{\boldsymbol{j}} = w_{\boldsymbol{j}} + 1$
        $\{w_{\boldsymbol{j}_\xi}, w_{\boldsymbol{i}}\} \overset{z\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{j}}, w_{\boldsymbol{i}_\xi}\}'$ with $\boldsymbol{i}_z = n_{\boldsymbol{i}}$, and $\boldsymbol{j}_z = n_{\boldsymbol{j}}$
      end do
    end do
  end do

end do
```

Algorithm ($\boldsymbol{j}$ major)

```
$w_{\boldsymbol{j}_\xi} = 0$
$w_{\boldsymbol{j}} = 0$
do $n_{\boldsymbol{j}} = |\boldsymbol{j}|, 0, -1$
  do $n'_{\boldsymbol{j}} = 0, n_{\boldsymbol{j}}$
    $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
    $w_{\boldsymbol{j}} = w_{\boldsymbol{j}} + 1$
    $w_{\boldsymbol{i}} = 1$
    $w_{\boldsymbol{i}_\xi} = 1$
    $\{w_{\boldsymbol{i}}, w_{\boldsymbol{j}_\xi}\} \overset{x\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{i}_\xi}, w_{\boldsymbol{j}}\}'$
      with $\boldsymbol{i}_x = |\boldsymbol{i}|$ and $\boldsymbol{j}_x = n_{\boldsymbol{j}} - n'_{\boldsymbol{j}}$
    do $n_{\boldsymbol{i}} = 0, |\boldsymbol{i}|$
      $w_{\boldsymbol{i}_\xi} = w_{\boldsymbol{i}_\xi} + 1$
      $\{w_{\boldsymbol{i}} + n_{\boldsymbol{i}}, w_{\boldsymbol{j}_\xi} + 1\} \overset{y\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{i}_\xi}, w_{\boldsymbol{j}}\}'$
        with $\boldsymbol{i}_y = n_{\boldsymbol{i}}$ and $\boldsymbol{j}_y = n'_{\boldsymbol{j}}$
    end do
    $w_{\boldsymbol{i}} = 0$
    do $n_{\boldsymbol{i}} = 0, |\boldsymbol{i}|$
      do $n'_{\boldsymbol{i}} = 0, |\boldsymbol{i}| - n_{\boldsymbol{i}}$
        $w_{\boldsymbol{i}} = w_{\boldsymbol{i}} + 1$
        $w_{\boldsymbol{i}_\xi} = w_{\boldsymbol{i}_\xi} + 1$
        $\{w_{\boldsymbol{i}}, w_{\boldsymbol{j}_\xi} + n_{\boldsymbol{j}} + 2\} \overset{z\text{recurrence}}{\Rightarrow} \{w_{\boldsymbol{i}_\xi}, w_{\boldsymbol{j}}\}'$
          with $\boldsymbol{i}_z = n_{\boldsymbol{i}}$ and $\boldsymbol{j}_z = |\boldsymbol{j}| - n_{\boldsymbol{j}}$
      end do
    end do
  end do
  $w_{\boldsymbol{j}_\xi} = w_{\boldsymbol{j}_\xi} + 1$
end do
```

Figure 4.6: Procedure of $\{\boldsymbol{i}, \boldsymbol{j}+\boldsymbol{e}_\xi\} \Rightarrow \{\boldsymbol{i}+\boldsymbol{e}_\xi, \boldsymbol{j}\}$.

4.9 Auxiliary Functions in Gen1Int

As shown in Section 4.6, the integrals of some operators needs the knowledge of special functions. We will therefore focus on the evaluation of auxiliary functions in Gen1Int. In particular, we will also discuss the limitations of current routines in Gen1Int, which may point out further development.

4.9.1 Boys function

Boys function is defined as

$$F_n(T)=\int_0^1\exp[-Tu^2]u^{2n}\mathrm{d}u, \tag{4.147}$$

which is used for nuclear attraction potential, Gaussian charge potential, diamagnetic spin-orbit coupling, and model core potential (Version 1) ...

Describe subroutines in `aux_boys_vec.F90` ...

4.9.2 Function $G_n(T)$

The function $G_n(T)$ ($n\geq 0$, $T\geq 0$) is defined as

$$G_n(T)=\int_0^1\exp[-T(1-u^2)]u^{2n}\mathrm{d}u, \tag{4.148}$$

which is used for inverse square distance potential ...

Describe subroutines in `aux_boys_gfun.F90` ...

4.9.3 Scaled Modified Spherical Bessel Function of the First Kind

The scaled modified spherical Bessel function of the first kind is defined as

$$\mathrm{e}^{-x}M_n(x)=\mathrm{e}^{-x}x^n\left(\frac{1}{x}\frac{\mathrm{d}}{\mathrm{d}x}\right)^n\frac{\sinh x}{x}, \tag{4.149}$$

where $M_n(x)$ is the so-called modified spherical Bessel function of the first kind. The scaled modified spherical Bessel function of the first kind is used when calculating the integrals of effective core potential. Its evaluation has been discussed in Ref. [17] ...

Describe subroutines in `aux_msphi_vec.F90` ...

4.10 Test Suite of Gen1Int

There are around 30,000 lines source code in Gen1Int, and most of them are designed for different recurrence relations. Therefore, a “complete” and “thorough” test suite of all the subroutines is vital for Gen1Int. The source codes of Fortran 90 test suite are in the directory `test_f90`, and those of Python test suite are in `tests`. The tests of recurrence relations are usually performed by comparing the results from Gen1Int and those from recursive functions in Fortran 90 and Python. Other tests are normally carried out by comparing with the predefined referenced results. Please see the comments of individual source code for further details.

Fortran 90 test suite will generate an HTML log file called `test_gen1int.html`, please check it carefully if there is any error (marked in red color). You could also report the errors together with the information of operating system and compilers to the authors.

We should however emphasize that we release Gen1Int WITHOUT ANY WARRANTY as claimed in the copyright page. We will nonetheless do our best to make Gen1Int be useful and the functionalities have been tested to the best of our ability.

Adding finite difference tests (order by order to 10?) using primitive and contracted GTOs (moving bra and operator center), with different situations: (1) bra, ket, operator centers are quite close, (2) bra center is far away from the other two, (3) ket center is far away from the other two, (4) operator center is far away from the other two, (5) all of them are far away from each other.

4.10.1 Testing Dashboard of Gen1Int

1. Create a file `cmake/Tests.cmake` in which `tools/runtest.sh` returns a 0 if the test passes and 1 (or something nonzero) if it fails;

2. Create `CTestConfig.cmake`;

3. Set up CDash and CTest in `CMakeLists.txt`;

4. Try `make test`, `make Experimental` or `make Nightly` (put it into `crontab`);

5. Check `http://repo.ctcc.no/CDash/index.php?project=Gen1Int`.

Chapter 5

Gen1Int Subroutines

In this section, we give the list of all public and private Gen1Int subroutines according to their categories. ***need to add more implemented subroutines ...***

5.1 Public Gen1Int Subroutines

"**Public**" subroutines are those that could be called by users in their own codes.

1. Utilities
   (a) `norm_contr_cgto`, see Table ;
   (b) `norm_contr_sgto`, see Table ;
   (c) `reorder_sgtos`, see Table 2.1;
   (d) `reorder_sgto_ints`, see Table 2.1;
   (e) `reorder_cgtos`, see Table 2.1;
   (f) `reorder_cgto_ints`, see Table 2.1;
   (g) `trace_sgto_ints`, see Table ;
   (h) `trace_cgto_ints`, see Table ;
   (i) `trace_gto_ints`, see Table ;
2. Geometric derivatives
   (a) `geom_total_tree_init`, see Table 4.1;
   (b) `geom_total_tree_search`, see Table 4.1;
3. Contracted integrals with Cartesian or spherical Gaussians
   (a) Cartesian multipole moments
      i. `contr_cgto_carmom`, see Table ;
      ii. `contr_sgto_carmom`, see Table ;
   (b) δ-function
      i. `contr_cgto_delta`, see Table ;
      ii. `contr_sgto_delta`, see Table ;

   (c) nuclear attraction potential
      i. `contr_cgto_nucpot`, see Table ;
      ii. `contr_sgto_nucpot`, see Table ;
   (d) inverse square distance potential
      i. `contr_cgto_isdpot`, see Table ;
      ii. `contr_sgto_isdpot`, see Table ;
   (e) Gaussian charge potential
      i. `contr_cgto_gaupot`, see Table ;
      ii. `contr_sgto_gaupot`, see Table ;
   (f) diamagnetic spin-orbit coupling
      i. `contr_cgto_dso`, see Table ;
      ii. `contr_sgto_dso`, see Table ;
   (g) effective core potential
      i. `contr_cgto_ecp_local`, see Table ;
      ii. `contr_cgto_ecp_non`, see Table ;
      iii. `contr_sgto_ecp_local`, see Table ;
      iv. `contr_sgto_ecp_non`, see Table ;
   (h) model core potential (Version 1)
      i. `contr_cgto_mcp1_pot`, see Table ;
      ii. `contr_cgto_mcp1_core`, see Table ;
      iii. `contr_sgto_mcp1_pot`, see Table ;
      iv. `contr_sgto_mcp1_core`, see Table ;

4. Contracted integrals with rotational London atomic orbitals
   (a) Cartesian multipole moments
      i. `lcgto_zero_carmom`, see Table ;
      ii. `lsgto_zero_carmom`, see Table ;
   (b) δ-function
      i. `lcgto_zero_delta`, see Table ;
      ii. `lsgto_zero_delta`, see Table ;
   (c) nuclear attraction potential
      i. `lcgto_zero_nucpot`, see Table ;
      ii. `lsgto_zero_nucpot`, see Table ;
   (d) inverse square distance potential
      i. `lcgto_zero_isdpot`, see Table ;
      ii. `lsgto_zero_isdpot`, see Table ;
   (e) Gaussian charge potential
      i. `lcgto_zero_gaupot`, see Table ;
      ii. `lsgto_zero_gaupot`, see Table ;
   (f) diamagnetic spin-orbit coupling

   i. `lcgto_zero_dso`, see Table ;
   ii. `lsgto_zero_dso`, see Table ;

(g) effective core potential

   i. `lcgto_zero_ecp_local`, see Table ;
   ii. `lcgto_zero_ecp_non`, see Table ;
   iii. `lsgto_zero_ecp_local`, see Table ;
   iv. `lsgto_zero_ecp_non`, see Table ;

(h) model core potential (Version 1)

   i. `lcgto_zero_mcp1_pot`, see Table ;
   ii. `lcgto_zero_mcp1_core`, see Table ;
   iii. `lsgto_zero_mcp1_pot`, see Table ;
   iv. `lsgto_zero_mcp1_core`, see Table ;

5.2 Private Gen1Int Subroutines

"**Private**" subroutines are usually not be called by the users, but used inside Gen1Int.

1. Utilities

   (a) subroutines `xtimer_set` and `xtimer_view` in file `xtimer.F90`, print the CPU elapsed time of individual Gen1Int subroutine[*], enabled by `-DXTIME` in compiler option;

   (b) subroutines `dump_gto_deriv`, `dump_gen_opt`, `dump_ecp`, `dump_mcp1_pot`, and `dump_mcp1_core` in file `dump_info.F90` print the information of basis sets, operators and derivatives during calculations, enabled by `-DDEBUG` in compiler option;

   (c) subroutines `dbinom_coeff` and `pascal_triangle`[†] in file `binom_coeff.F90` computes the binomial coefficient and Pascal's triangle, respectively;

   (d) subroutines `sort_gen_cents` and `sort_atom_cents` in file `sort_cents.F90` sort the given centers in descending order, `sort_gen_cents` accepts non-atomc centers (such as dipole origin) whose indices are greater than defined variable `MAX_IDX_NON` (= 0) in `max_idx_non.h`;

   (e) `shell_scatter`, see Section 4.3.2;

   (f) `shell_gather`, see Section 4.3.2;

   (g) subroutine `const_contr_ints` in file `const_contr_ints.F90` performs the contractions (4.14) and (4.19);

2. Basis sets transformations

   (a) `hgto_to_cgto`, see Section 4.8.5;

   (b) `hgto_to_cgto_p`, see Section 4.8.5;

   (c) `hgto_to_cgto_d`, see Section 4.8.5;

   (d) `sub_hgto_to_cgto`, see Section 4.8.5;

   (e) `hgto_to_lcgto`, see Section 4.8.5;

[*]The CPU elapsed time is got from the intrinsic function `cpu_time`.

[†]The numbers of Pascal's triangle are real type.

3. Geometric derivatives
   (a) `geom_total_num_paths`, see Table 4.1;
   (b) `geom_total_new_path`, see Table 4.1;
   (c) `geom_part_zero_param`, see Section 4.3.2;
   (d) `geom_part_zero_scatter`, see Section 4.3.2;
   (e) `geom_part_one_param`, see Section 4.3.2;
   (f) `geom_part_one_scatter`, see Section 4.3.2;
   (g) `geom_part_two_param`, see Section 4.3.2;
4. Geometric derivatives of dipole origin, `carmom_deriv`, see Section 4.6.2;
5. Cartesian multipole moment integrals
   (a) `contr_cgto_carmom_recurr`, see Table ;
   (b) `prim_hgto_carmom`, see Table ;
   (c) `carmom_hrr_ket`, see Section 4.6.2;
   (d) `sub_carmom_hrr_ket`, see Section 4.6.2;
   (e) `carmom_hbra`, see Section 4.6.2;
   (f) `carmom_moment`, see Section 4.6.2;
   (g) `carmom_moment_p`, see Section 4.6.2;
   (h) `sub_carmom_moment`, see Section 4.6.2;
6. δ-function
   (a) `xxxx`, see Table ;
   (b) `xxxx`, see Table ;
7. Nuclear attraction potential
   (a) `nucpot_geom`, see Section 4.6.4;
   (b) `sub_nucpot_geom_d`, see Section 4.6.4;
8. Inverse square distance potential
   (a) `xxxx`, see Table ;
   (b) `xxxx`, see Table ;
9. Gaussian charge potential
   (a) `xxxx`, see Table ;
   (b) `xxxx`, see Table ;
10. Diamagnetic spin-orbit coupling
   (a) `xxxx`, see Table ;
   (b) `xxxx`, see Table ;

11. Effective core potential
    (a) `xxxx`, see Table ;
    (b) `xxxx`, see Table ;
12. Model core potential (Version 1)
    (a) `xxxx`, see Table ;
    (b) `xxxx`, see Table ;
13. Quadrature
    (a) `xxxx`, see Table ;
    (b) `xxxx`, see Table ;
14. Auxiliary functions
    (a) Boys function
        i. `xxxx`, see Table ;
        ii. `xxxx`, see Table ;
    (b) Function $G_n(T)$
        i. `xxxx`, see Table ;
        ii. `xxxx`, see Table ;
    (c) Scaled Modified Spherical Bessel Function of the First Kind
        i. `xxxx`, see Table ;
        ii. `xxxx`, see Table ;

Chapter 6

Files and Directories of Gen1Int

We will list all the files and directories of Gen1Int, with a brief description ...

6.1 Header Files in Gen1Int

We will describe the header files in directory `src`, some of them may be modified to satisfy the requirement of users ...

1. `xkind.h`: kind type parameter of real numbers
2. `stdout.h`: IO unit of standard output
3. `pi.h`: constant π
4. `tag_cent.h`: marking the sequence of bra, ket and operator centers
5. `hgto_to_sgto.h`: parameters related to the transformation between HGTOs and SGTOs
6. `max_idx_non.h`: maximum index of non-atomic centers
7. `boys_power.h`: used by power series expansion of Boys function (generated by `tools/GenHeader.py`)
8. `max_gen_order.h`: maximum order for Gen1Int (generated by `tools/GenHeader.py`)
9. `tab_boys.h`: pretabulated Boys functions (generated by `tools/GenHeader.py`)

Chapter 7

Limitations of Gen1Int

The limitations of Gen1Int may come from the accuracy of auxiliary functions, the huge number of derivatives (lots of memory consumed), the accuracy due to large distance between basis set centers and operator center (will it be a problem?) ...

Gen1Int might not be efficient ...

We describe our future long- and short-term developments in file `TODO` ...

When C is far away from P, Eq. (4.75) are really some small numbers (unless we have large total exponent), positive and negative, one by one. Therefore, we might have two small number subtraction in Eq. (4.74) (remind that Boys function decays quickly, so multiplied by a large X_{CP} might still result in a small number), which as you know may be unstable.

Another thing is related to the evaluation of Boys function, for the time being, Gen1Int provides the Boys function with accuracy, at least, around 10^{-12} or 10^{-13} (for some arguments it might be even better). For your information, although we have encountered such problem, Gen1Int will stop with error message if the argument and order of Boys function can not be calculated with required accuracy.

Therefore, based on the above two points (small numbers in recurrence relations and the accuracy of Boys functions), Gen1Int might have problem to give accurate enough results when EFG center is far away from the product center.

Instead of first recovering derivatives on EFG center C, we could also first recover GTOs on bra and ket center, and get the derivatives on EFG center at last. However, this is more or less the same as what is implemented in Dalton now. And, we can say it is not stable from your tests.

Therefore, to summarize, although Gen1Int might have problem but its current solution is better than EFGINT in Dalton, and we have to trust it ;-)

P.S: Further improvement of Gen1Int could be:

(1) return zero integrals when bra and ket centers are far away;

(2) improve the accuracy of Boys functions, which could be done, and I also have some idea (for instance, increasing the accuracy of pretabulated table of Boys functions, improve the accuracy of modified asymptotic series expansion and upward recurrence relation–which is the key point);

(3) uses more stable recurrence relations or reduce the chance of two small or two big number operations, which I have no idea for the time being.

Bibliography

[1] Bin Gao and Kenneth Ruud. GEN1INT: An object-oriented library to evaluate one-electron integrals and their derivatives, 2011. In manuscript.

[2] Bin Gao, Andreas J. Thorvaldsen, and Kenneth Ruud. GEN1INT: a unified procedure for the evaluation of one-electron integrals over Gaussian basis functions and their geometric derivatives. *Int. J. Quantum Chem.*, 111(4):858–872, 2010.

[3] Bin Gao, Kenneth Ruud, and Trygve Helgaker. Evaluating one-electron integrals and their geometric derivatives II: magnetic properties, relativistic corrections, and pseudopotential, 2011. In manuscript.

[4] Xavier Saint Raymond. *Elementary introduction to the theory of pseudodifferential operators.* CRC Press, Boca Raton, Florida, 1991.

[5] Jürgen Gauss, Kenneth Ruud, and Trygve Helgaker. Perturbation-dependent atomic orbitals for the calculation of spin-rotation constants and rotational g tensors. *J. Chem. Phys.*, 105(7):2804–2812, 1996.

[6] Trygve Helgaker, Poul Jørgensen, and Jeppe Olsen. *Molecular Electronic-Structure Theory.* John Wiley & Sons Ltd, Chichester, 2000.

[7] Simen Reine, Erik Tellgren, and Trygve Helgaker. A unified scheme for the calculation of differentiated and undifferentiated molecular integrals over solid-harmonic Gaussians. *Phys. Chem. Chem. Phys.*, 9:4771–4779, 2007.

[8] Trygve Helgaker and Poul Jørgensen. An electronic Hamiltonian for origin independent calculations of magnetic properties. *J. Chem. Phys.*, 95(4):2595–2601, 1991.

[9] Sheela Kirpekar, Jens Oddershede, and Hans Jørgen Aagaard Jensen. Relativistic corrections to molecular dynamic dipole polarizabilities. *J. Chem. Phys.*, 103(8):2983–2990, 1995.

[10] Olav Vahtras, Hans Ågren, Poul Jørgensen, Hans Jørgen Aa. Jensen, Søren B. Padkjær, and Trygve Helgaker. Indirect nuclear spin–spin coupling constants from multiconfiguration linear response theory. *J. Chem. Phys.*, 96(8):6120–6125, 1992.

[11] Kenneth Ruud, Trygve Helgaker, Keld L. Bak, Poul Jørgensen, and Hans Jørgen Aa. Jensen. Hartree–Fock limit magnetizabilities from London orbitals. *J. Chem. Phys.*, 99(5):3847–3859, 1993.

[12] Keld Lars Bak, Poul Jørgensen, Hans Jørgen Aa. Jensen, Jeppe Olsen, and Trygve Helgaker. First-order nonadiabatic coupling matrix elements from multiconfigurational self-consistent-field response theory. *J. Chem. Phys.*, 97(10):7573–7584, 1992.

[13] A. D. Buckingham. Permanent and Induced Molecular Moments and Long-Range Intermolecular Forces. *Adv. Chem. Phys.*, 12:107, 1967.

[14] M. Aslam Chaudhry and Syed M. Zubair. *On a class of incomplete gamma functions with applications.* CRC Press, Boca Raton, 2002.

[15] Andrew Komornicki, Kazuhiro Ishida, Keiji Morokuma, Robert Ditchfield, and Morgan Conrad. Efficient determination and characterization of transition states using ab-initio methods. *Chem. Phys. Lett.*, 45(3):595–602, 1977.

[16] Luis R. Kahn. Relationships among derivatives of the integrals in the calculation of the gradient of the electronic energy with respect to the nuclear coordinates. *J. Chem. Phys.*, 75(8):3962–3966, 1981.

[17] Roberto Flores-Moreno, Rodrigo J. Alvarez-Mendez, Alberto Vela, and Andreas M. Köster. Half-Numerical Evaluation of Pseudopotential Integrals. *J. Comput. Chem.*, 27(9):1009–1019, 2005.

Index