MOLINT 1.0: An Application Programming Interface Framework for the Computation of Molecular Integrals and Their First Derivatives for the Density-Fitted Methods

Uğur Bozkaya*

Department of Chemistry, Hacettepe University, Ankara 06800, Turkey

E-mail: ugur.bozkaya@hacettepe.edu.tr

*To whom correspondence should be addressed

Abstract

The efficient computation of molecular integrals and their derivatives is a crucial step in molecular property evaluation in modern quantum chemistry. As an integral tensor decomposition technique, the density-fitting (DF) approach becomes a popular tool to reduce the memory and disk requirements for the electron repulsion integrals. In this study, an application programming interface (API) framework, denoted MOLINT (MFW), for the computation of molecular integrals and their first derivatives, over contracted Gaussian functions, for the density-fitted methods is reported. The MFW is free software and it includes overlap, dipole, kinetic, potential, metric, and 3-index integrals, and their first derivatives. Furthermore, the MFW provides a smooth approach to build the Fock matrix and evaluate analytic gradients for the density-fitted methods. The MFW is a C++/FORTRAN hybrid code, which can take advantage of shared-memory parallel programming techniques. Our results demonstrate that the MFW is an efficient and user-friendly API for the computation of molecular integrals and their first derivatives.

1 Introduction

Computation of molecular integrals is a crucial step of any molecular property evaluation in modern quantum chemistry.[1–20] Especially, the computation of electron repulsion integrals (ERIs) requires a special attention. To overcome memory and disk problems associated with the conventional 4-index ERIs, tensor decomposition of ERIs has attracted significant interest.[21–38] The most popular ERI factorization technique is the density fitting (DF) approach.[21–28,35–38] In the DF technique, the four-dimensional ERI tensor is replaced with the three-dimensional tensors, expanding ERIs in terms of a predefined auxiliary basis set.

In addition to the energy, analytic gradients for electron correlation methods with the DF approach have been reported for MP2 with conventional HF reference (DF-MP2),[39–41] as well as with the DF-HF reference,[42] the second-order coupled cluster (CC2),[43,44] local MP2 (DF-LMP2),[45] complete active space second-order perturbation theory (DF-CASPT2),[46] orbital-optimized MP2, MP2.5, and MP3 (DF-OMP2, DF-OMP2.5, DF-OMP3),[47,48] local time-dependent coupled cluster response theory (DF-TD-LCC2),[49] the coupled-cluster singles and doubles and CCD methods (DF-CCSD and DF-CCD),[50] and recently for the coupled-cluster singles and doubles with perturbative triples [DF-CCSD(T)] method.[51]

One may use the existent 4-index integrals code to generate 3-index DF integrals. However, a better approach to develop new integral codes for 3-index integrals. In a 2004 study, Ahlrichs[52] presented an efficient algorithm to compute 3-center integrals based on the Obara-Saika (OS) scheme. In 2017 and 2018 studies, Samu and Kállay[53,54] implemented various approaches, such as OS, McMurchie and Davidson(MD), and Rys quadrature, for computations of 3-index ERIs and their first derivatives. Efficient implementations of 3-index ERIs are available in Libint library of Valeev,[16,55,56] which is available in Psi4,[57] in Mrcc program,[58] and in some commercial software.

In this research, an application programming interface (API) framework, denoted Molint (MFW), for the computation of molecular integrals and their first derivatives, over contracted Gaussian functions, for the density-fitted methods is reported. The MFW is an efficient and

"user-friendly" API framework, which provides a straightforward way to access molecular integrals and their derivatives. The MFW includes overlap, dipole, kinetic, potential, metric, and 3-index integrals as well as their first derivatives. Furthermore, the MFW provides a smooth approach to build the Fock matrix and compute the molecular gradients. The MFW is a C++/FORTRAN hybrid code, which can take advantage of shared-memory parallel programming techniques. The features and efficiency of the MFW are demonstrated in illustrative applications.

2 Spherical Gaussians

The general form of unnormalized primitive Gaussian type orbitals (GTOs) centered at R_A can be written as

$$G_{ijk}(a, |\mathbf{r} - \mathbf{R_A}|) = \big(x - A_x\big)^i \big(y - A_y\big)^j \big(z - A_z\big)^k e^{-a|r-R_A|^2}, \quad (1)$$

which may be further simplified to:

$$G_{ijk}(\mathbf{r}, a, \mathbf{A}) = x_A^i y_A^j z_A^k e^{-ar_A^2}. \quad (2)$$

The full set of GTOs with the same angular momentum number $l = i + j + k$ and the same exponent a are said to constitute a *shell*. The number of GTOs in a shell with angular momentum l is

$$N_l^c = \frac{(l+1)(l+2)}{2}. \quad (3)$$

A real valued spherical GTOs with quantum numbers l and m, with exponent a, centered at $\mathbf{A}$ is given by,

$$G_{lm}(\mathbf{r}, a, \mathbf{A}) = S_{lm}(x_A, y_A, z_A) e^{-ar_A^2}, \quad (4)$$

where $S_{lm}(\mathbf{r_A})$ is the real solid-harmonics,

$$S_{l,m}(r,\theta,\phi) = (-1)^m \sqrt{\frac{8\pi}{2l+1}} r^l Re[Y_l^m(\theta,\phi)], m > 0, \tag{5}$$

$$S_{l,-m}(r,\theta,\phi) = (-1)^m \sqrt{\frac{8\pi}{2l+1}} r^l Im[Y_l^m(\theta,\phi)], m > 0. \tag{6}$$

A shell of spherical GTOs contains all GTOs of the same a and l, but different $|m| \leq l$. The number of spherical GTOs in a shell with l and m is

$$N_l^s = 2l+1, \tag{7}$$

The relationship between spherical GTOs and Cartesian GTOs can be written as[1]

$$G_{lm}(\mathbf{r},a,\mathbf{A}) = N_{lm}^S \sum_{t=0}^{(l-|m|)/2} \sum_{u=0}^{t} \sum_{v=v_m}^{[|m|/2-v_m]+v_m} C_{tuv}^{lm} G_{i,j,k}(\mathbf{r},a,\mathbf{A}), \tag{8}$$

where

$$i = 2t + |m| - j, \tag{9}$$

$$j = 2(u+v), \tag{10}$$

$$k = l - i - j, \tag{11}$$

$$N_{lm}^S = \frac{1}{2^{|m|} l!} \sqrt{\frac{2(l+|m|)!(l-|m|)!}{2^{\delta_{0m}}}}, \tag{12}$$

$$C_{tuv}^{lm} = (-1)^{t+v-v_m} \frac{1}{4^t} \binom{l}{t} \binom{l-t}{|m|+t} \binom{t}{u} \binom{|m|}{2v}, \tag{13}$$

$$v_m = 0; m \geq 0, \tag{14}$$

$$v_m = 1/2; m < 0. \tag{15}$$

CGTOs can be written as

$$G_{\mu lm}(\mathbf{r}, a, \mathbf{A}) = \sum_p d_{\mu p} G_{lm}(\mathbf{r}, a_p, \mathbf{A}), \tag{16}$$

where $d_{\mu p}$ are the contraction coefficients.

3 Cartesian Gaussians

The primitive GTOs can be written as

$$G_{ijk}(\mathbf{r}, a, \mathbf{A}) = x_A^i y_A^j z_A^k e^{-ar_A^2}. \tag{17}$$

One of the important feature of the Cartesian GTOs is that they can be factorized as follows

$$G_{ijk}(\mathbf{r}, a, \mathbf{A}) = G_i(x, a, A_x) G_j(y, a, A_y) G_k(z, a, A_z), \tag{18}$$

where for example

$$G_i(x, a, A_x) = x_A^i e^{-ax_A^2}. \tag{19}$$

The normalization constant for a Gaussian is:

$$N_i = \left(\frac{2}{\pi}\right)^{1/4} \frac{2^i a^{(2i+1)/4}}{\sqrt{(2i-1)!!}}. \tag{20}$$

3.1 Normalization of Contracted Cartesian GTOs

The general form of unnormalized contracted Cartesian Gaussian type orbitals (CGTOs) centered at the R_A can be written as

$$\chi_\mu(\mathbf{r_A}) = \sum_p^{n_p} d_{p\mu} G_p(a_{p\mu}, \mathbf{r_A}), \tag{21}$$

where $d_{p\mu}$ are the contraction coefficients and n_p is the number of PGOs, and:

$$G_p(a_{p\mu}, \mathbf{r_A}) = x_A^i y_A^j z_A^k e^{-a_{p\mu} r_A^2}. \tag{22}$$

The contraction coefficients $d_{p\mu}$ already include normalization constants so that the resulting combination is properly normalized. Published contraction coefficients $\tilde{d}_{p\mu}$ are linear coefficients for normalized primitives, hence the normalization-including contraction coefficients $d_{p\mu}$ have to be computed from them. Hence, a normalized CGTO can be written as follows:

$$\chi_\mu(\mathbf{r_A}) = N \sum_p^{n_p} \tilde{d}_{p\mu} \tilde{G}_p, \tag{23}$$

where $\tilde{d}_{p\mu}$ are linear coefficients for normalized primitives $\tilde{G}_p$. Hence,

$$\chi_\mu(\mathbf{r_A}) = N \sum_p^{n_p} \tilde{d}_{p\mu} N_p G_p. \tag{24}$$

Now let us define (apply primitive normalization):

$$d_{p\mu} = N_p \tilde{d}_{p\mu}, \tag{25}$$

then,

$$\chi_\mu(\mathbf{r_A}) \;=\; N\sum_p^{n_p} d_{p\mu}G_p. \tag{26}$$

Finally, we employ normalization scheme of the LIBINT package.[16,55] In this scheme, Cartesian Gaussian functions in a shell of angular momentum l have the same normalization factor N, which is determined such that the Cartesian functions x^l, y^l, and z^l are normalized to unity.

$$\pi^{3/2}\frac{(2l-1)!!}{2^l}\sum_{p,q}^{n_p,n_q}\frac{d_{p\mu}d_{q\mu}}{\left(a_{p\mu}+b_{q\mu}\right)^{l+3/2}} \;=\; \frac{1}{N^2}, \tag{27}$$

the final coefficients are:

$$d'_{p\mu} \;=\; Nd_{p\mu}. \tag{28}$$

3.2 The Gaussian Product Rule

The famous Gaussian product rule can be written as:

$$e^{-ax_A^2}\,e^{-bx_B^2} \;=\; K_{AB}^x\,e^{-px_P^2}, \tag{29}$$

where

$$p \;=\; a+b, \tag{30}$$

$$\mu \;=\; \frac{ab}{a+b}, \tag{31}$$

$$X_{AB} \;=\; A_x - B_x, \tag{32}$$

$$P_x = \frac{aA_x + bB_x}{p}, \tag{33}$$

$$K^x_{AB} = e^{-\mu X^2_{AB}}. \tag{34}$$

3.3 Gaussian Overlap Distribution

Let us use the following short-hand notation

$$G_a(\mathbf{r}) = G_{ikm}(\mathbf{r}, a, \mathbf{A}), \tag{35}$$

$$G_b(\mathbf{r}) = G_{jln}(\mathbf{r}, b, \mathbf{B}). \tag{36}$$

The Gaussian overlap distribution can be written as

$$\Omega_{ab}(\mathbf{r}) = G_a(\mathbf{r})G_b(\mathbf{r}), \tag{37}$$

which can be factorized to:

$$\Omega_{ab}(\mathbf{r}) = \Omega^x_{ij}(x, a, b, A_x, B_x)\Omega^y_{kl}(y, a, b, A_y, B_y)\Omega^z_{mn}(z, a, b, A_z, B_z), \tag{38}$$

where the x-component can be written as

$$\Omega^x_{ij}(x, a, b, A_x, B_x) = K^x_{AB} x^i_A x^j_B e^{-p x^2_P}. \tag{39}$$

Gaussian Overlap Distribution has the following trivial properties

$$\Omega^x_{i+1,j} = x_A \Omega^x_{ij}, \tag{40}$$

$$\Omega^x_{i,j+1} = x_B \Omega^x_{ij}, \tag{41}$$

$$\Omega^x_{i,j+1} - \Omega^x_{i+1,j} = X_{AB}\Omega^x_{ij}. \tag{42}$$

4 Hermite Gaussians

The Hermite Gaussian with exponent p centered on $\mathbf{P}$ is defined as follows

$$\Lambda_{tuv}(\mathbf{r}, p, \mathbf{P}) = \Big(\frac{\partial}{\partial P_x}\Big)^t \Big(\frac{\partial}{\partial P_y}\Big)^u \Big(\frac{\partial}{\partial P_z}\Big)^v e^{-pr^2_P}, \tag{43}$$

where $\mathbf{r_P} = \mathbf{r} - \mathbf{P}$.

This functions are can be factorized as in the case of Cartesian GTOs

$$\Lambda_{tuv}(\mathbf{r}, p, \mathbf{P}) = \Lambda_t(x, p, P_x)\Lambda_u(y, p, P_y)\Lambda_v(x, p, P_z). \tag{44}$$

The x-component for example

$$\Lambda_t(x, p, P_x) = \Big(\frac{\partial}{\partial P_x}\Big)^t \Lambda_0(x, p, P_x), \tag{45}$$

where

$$\Lambda_0(x, p, P_x) = e^{-px^2_P}. \tag{46}$$

Hermite functions have the following recursion relation:

$$x_P\Lambda_t = \frac{1}{2p}\Lambda_{t+1} + t\Lambda_{t-1}. \tag{47}$$

5 Evaluation of The Boys Function

Before continue to the evaluation of molecular integrals let us consider the Boys function, which is central for the computation of potential and electron repulsion integrals.

$$F_n(x) = \int_0^1 t^{2n} e^{-xt^2}\, dt, \tag{48}$$

where $F_n(x)$ is the n-th order Boys function. Asymptotic values of the Boys function are

$$F_n(0) = \int_0^1 t^{2n}\, dt = \frac{1}{2n+1}, \tag{49}$$

and at large x:

$$F_n(x) \approx \frac{(2n-1)!!}{2^{n+1}}\sqrt{\frac{\pi}{x^{2n+1}}}. \tag{50}$$

The Boys function satisfies the upward recursion:

$$F_{n+1}(x) = \frac{(2n+1)F_n(x) - exp(-x)}{2x}, \tag{51}$$

and the downward recursion:

$$F_{n-1}(x) = \frac{2xF_n(x) + exp(-x)}{2n-1}. \tag{52}$$

The downward recursion is computationally more stable for small x values.

Saunders[59] suggest the following series expansion for the boys function:

$$F_n(x) = e^{-x}\sum_{k=0}^{\infty}\frac{(2x)^k}{(2n+1)(2n+3)\ldots(2n+2k+1)}, \tag{53}$$

which may be written as

$$F_n(x) = e^{-x} \sum_{k=0}^{\infty} \frac{(2x)^k}{g_{nk}}, \tag{54}$$

where

$$g_{nk} = \prod_{i=0}^{k} (2n + 2i + 1). \tag{55}$$

Saunders note that this series is unconditionally convergent. However, for large values of n, convergence can be slow.

A faster convergent modified Maclaurin series expansion is suggested by Gill, Head-Gordon, and People[9] as follows:

$$F_n(x) = \frac{1}{2} e^{-x} \frac{1}{n + 1/2} \left(1 + \frac{x}{n + 3/2} \left(1 + \frac{x^2}{n + 5/2} (1 + \dots) \right) \right), \tag{56}$$

The GHP method converges faster than Saunders' method.

For the small arguments, we can expand the Boys function as a Taylor series as follows,

$$F_n(x) = \sum_{k=0}^{\infty} \frac{F_{n+k}(x_0)(-\Delta x)^k}{k!}, \tag{57}$$

where Δx is generally chosen as 0.1 and $x_0 = x - \Delta x$. The first six terms of the Taylor series is a good approximation. Hence, we calculate $F_n(x_0), \dots, F_{n+k_{max}}(x_0)$ from the series expansion in Eq.(53)–Eq.(56) and we use these values to calculate $F_n(x)$. Further, we only need to compute the highest function $F_{n_{max}}(x)$ required and all other lower order functions can be calculated from the downward recursion relation

$$F_n(x) = \frac{2x F_{n+1}(x) + exp(-x)}{2n + 1}. \tag{58}$$

Final Strategy: Let us assume that we use a taylor expansion of order $k+1$ for the small argument.

1. Pretabulate $F_n(x)$ for $0 < x \leq 30$ and $n = n_{max} + k$ with the $\Delta x = 0.1$
2. Obtain all grid values for $n = 0 - n_{max} + k - 1$ from the downward recursion.
3. To compute $F_n(x)$ for an arbitrary x, find the closes point on the grid x_g, then use a $k+1$-term Taylor expansion Eq.(57), where $\Delta x = x - x_g$ should be used.
4. For $x > 30$ we may use the large argument.
5. For $x = 30$ we may use the Eq.(49).

6 McMurchie-Davidson (MD) Scheme

6.1 Overlap Distributions from Hermite Gaussians

Since the overlap distribution Ω_{ij} is a polynomial of degree $i+j$ in x_p, we can expand it as follows

$$\Omega_{ij}(x) \;=\; \sum_{t=0}^{i+j} E_t^{ij} \Lambda_t(x_P), \tag{59}$$

where expansion coefficients are constant, they are independent of the electronic coordinates. The expansion coefficients satisfy the following relation

$$E_t^{ij} \;=\; 0, \quad t < 0 \quad or \quad t > i+j. \tag{60}$$

Further, the Hermite expansion coefficients obey the following recursion relations,

$$E_t^{i+1,j} \;=\; X_{PA} E_t^{ij} + \frac{1}{2p}\big(i E_t^{i-1,j} + j E_t^{i,j-1} + E_{t-1}^{ij}\big), \tag{61}$$

$$E_t^{i,j+1} = X_{PB}E_t^{ij} + \frac{1}{2p}\big(iE_t^{i-1,j} + jE_t^{i,j-1} + E_{t-1}^{ij}\big), \tag{62}$$

with

$$E_0^{00} = K_{AB}^x, \tag{63}$$

$$E_t^{00} = 0, \quad t > 0. \tag{64}$$

6.2 Overlap Integrals

Here we consider overlap integrals over primitives. With the following short-hand notation:

$$G_a(\mathbf{r}) = G_{ikm}(\mathbf{r}, a, \mathbf{A}), \tag{65}$$

$$G_b(\mathbf{r}) = G_{jln}(\mathbf{r}, b, \mathbf{B}). \tag{66}$$

Overlap integrals are:

$$S_{ab} = \langle G_a | G_b \rangle, \tag{67}$$

which can be factorized as follows

$$S_{ab} = S_{ij}S_{kl}S_{mn}, \tag{68}$$

The final formula for the total integral becomes

$$S_{ab} = E_0^{ij}E_0^{kl}E_0^{mn}\Big(\frac{\pi}{p}\Big)^{3/2}. \tag{69}$$

6.3 Dipole Moment Integrals

Dipole moment integral can be written as

$$\mu_{ab} = \langle G_a|\mathbf{r}_C|G_b\rangle, \tag{70}$$

where **C** is the origin of the Cartesian dipole-moment, and it is x-component can be written as

$$\mu_{ij} = \langle G_i|x_C|G_j\rangle, \tag{71}$$

and the total integral can be written as:

$$\mu_{ab} = \mu_{ij}S_{kl}S_{mn} + S_{ij}\mu_{kl}S_{mn} + S_{ij}S_{kl}\mu_{mn}, \tag{72}$$

where

$$S_{ij} = E_0^{ij}\sqrt{\frac{\pi}{p}}, \tag{73}$$

and

$$\mu_{ij} = \left(E_0^{i+1,j} + X_{AC}E_0^{ij}\right)\sqrt{\frac{\pi}{p}}. \tag{74}$$

6.4 Kinetic Energy Integrals

The kinetic energy integral can be written as

$$T_{ab} = -\frac{1}{2}\langle G_a|\nabla^2|G_b\rangle, \tag{75}$$

it is x-component can be written as

$$T_{ij} = -\frac{1}{2}\langle G_i| \frac{\partial^2}{\partial x^2} |G_j\rangle, \tag{76}$$

and the total integral becomes:

$$T_{ab} = T_{ij}S_{kl}S_{mn} + S_{ij}T_{kl}S_{mn} + S_{ij}S_{kl}T_{mn}, \tag{77}$$

the x-components can be written as follows:

$$T_{ij} = -2b^2 S_{i,j+2} + b(2j+1)S_{ij} - \frac{1}{2}j(j-1)S_{i,j-2}. \tag{78}$$

6.5 Hermite Coulomb Integrals

Hermite Coulomb integrals are required for the evaluation of the potential energy and electron repulsion integrals.

$$R_{tuv}(p, R_{PC}) = \Big(\frac{\partial}{\partial P_x}\Big)^t \Big(\frac{\partial}{\partial P_y}\Big)^u \Big(\frac{\partial}{\partial P_z}\Big)^v F_0(pR_{PC}^2), \tag{79}$$

which can be obtained from the following auxiliary function

$$R^n_{tuv}(p, R_{PC}) = (-2p)^n \Big(\frac{\partial}{\partial P_x}\Big)^t \Big(\frac{\partial}{\partial P_y}\Big)^u \Big(\frac{\partial}{\partial P_z}\Big)^v F_n(pR_{PC}^2). \tag{80}$$

Hermite coulomb integrals satisfy the following recursion relations :

$$R^n_{t+1,u,v} = tR^{n+1}_{t-1,u,v} + X_{PC}R^{n+1}_{tuv}, \tag{81}$$

$$R^n_{t,u+1,v} = uR^{n+1}_{t,u-1,v} + Y_{PC}R^{n+1}_{tuv}, \tag{82}$$

$$R^{n}_{t,u,v+1} = vR^{n+1}_{t,u,v-1} + Z_{PC}R^{n+1}_{tuv}, \tag{83}$$

Starting from the highest

$$R^{n_{max}}_{000}(p, R_{PC}) = (-2p)^{n_{max}} F_{n_{max}}(pR^2_{PC}), \tag{84}$$

and using the recursion relations of Eqs.(81)–(83) we can generate all Hermite Coulomb integrals. The maximum value of n is

$$n_{max} = t + u + v. \tag{85}$$

6.6 Potential Energy Integrals

The potential energy integral can be written as

$$V_{ab} = -\sum_C \langle G_a | \frac{Z_C}{r_C} | G_b \rangle, \tag{86}$$

which can be written explicitly as follows:

$$V_{ab} = -\frac{2\pi}{p} \sum_C \sum_{tuv} Z_C E^{ab}_{tuv} R_{tuv}(p, R_{PC}). \tag{87}$$

Now we may write,

$$V_{ab} = \frac{2\pi}{p} \sum_{tuv} E^{ab}_{tuv} \Big[-\sum_C Z_C R_{tuv}(p, R_{PC}) \Big], \tag{88}$$

let us define

$$R'_{tuv} = -\sum_C Z_C R_{tuv}(p, R_{PC}), \tag{89}$$

hence, our final formula becomes:

$$V_{ab} = \frac{2\pi}{p} \sum_{tuv} E^{ab}_{tuv} R'_{tuv}. \tag{90}$$

Therefore, the contributions from all nuclei can be computed before the transformation.

6.7 Hermite Expansion for GTOs

Before proceed to the metric and 3-index ERIs, let us consider the Hermite expansion for a GTO. For the x–component it can be written as:

$$G_i = \sum_{t=0}^{i} E^i_t \Lambda_t(x_A), \tag{91}$$

Starting from the following Using derivative relations of GTO we obtain the following recursion relations:

$$E^{i+1}_t = (t+1)E^i_{t+1} + \frac{1}{2a} E^i_{t-1}, \tag{92}$$

and

$$E^{i+1}_t = \frac{1}{2a}\big(iE^{i-1}_t + E^i_{t-1}\big). \tag{93}$$

6.8 Metric Integrals

The metric integral is defined as follows,

$$J_{PQ} = \int\int \varphi_P(\mathbf{r_1}) \frac{1}{r_{12}} \varphi_Q(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}, \tag{94}$$

which may be written as (in terms of primitives):

$$(a|b) \;=\; \int\int \frac{G_a(\mathbf{r}_1)G_b(\mathbf{r}_2)}{r_{12}}\, d\mathbf{r}_1 d\mathbf{r}_2, \tag{95}$$

Using Hermite expansion and the Laplace transform of r_{12} we obtain the following final expression:

$$(a|b) \;=\; \frac{2\pi^{5/2}}{ab\sqrt{a+b}} \sum_{tuv} E^a_{tuv} \sum_{t'u'v'} (-1)^{t'+u'+v'} E^b_{t'u'v'} R_{t+t',u+u',v+v'}(\alpha, R_{AB}), \tag{96}$$

where

$$\alpha \;=\; \frac{ab}{a+b}. \tag{97}$$

6.8.1 The MD4 Algorithm for Metric Integrals

Let us define

$$F^b_{t'u'v'} \;=\; (-1)^{t'+u'+v'} E^b_{t'u'v'}, \tag{98}$$

and

$$\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}) \;=\; \frac{2\pi^{5/2}}{ab\sqrt{a+b}} R_{t+t',u+u',v+v'}(\alpha, R_{AB}). \tag{99}$$

Then, we may write

$$(a|b) \;=\; \sum_{tuv} E^a_{tuv} \sum_{t'u'v'} F^b_{t'u'v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}). \tag{100}$$

For efficiency we evaluate above equation in two steps as follows

$$(a|b) \;=\; \sum_{tuv} E^a_{tuv} g^b_{tuv}, \tag{101}$$

where

$$g^b_{tuv} = \sum_{t'u'v'} F^b_{t'u'v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}). \quad (102)$$

Further efficiency can be achieved by decomposing ERI and g^b_{tuv} into Cartesian components. Let us start with g^b_{tuv}

$$g^b_{tuv} = g^{i'k'm'}_{tuv}, \quad (103)$$

and

$$g^{i'k'm'}_{tuv} = \sum_{t'} F^{i'}_{t'} \sum_{u'} F^{k'}_{u'} \sum_{v'} F^{m'}_{v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}). \quad (104)$$

Then, we can write

$$(a|b) = g^{i'k'm'}_{ikm}, \quad (105)$$

$$g^{i'k'm'}_{ikm} = \sum_{t} E^i_t \sum_{u} E^k_u \sum_{v} E^m_v g^{i'k'm'}_{tuv}. \quad (106)$$

Following Helgaker et al, we call this algorithm as the MD4 algorithm.

6.9 3-Index Integrals

3-index TEI can be written as

$$(\mu\nu|Q) = \int\int \chi_\mu(\mathbf{r_1})\chi_\nu(\mathbf{r_1}) \frac{1}{r_{12}} \varphi_Q(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}, \quad (107)$$

3-index TEI may be written as (in terms of primitives):

$$(ab|C) \quad = \quad \int\int \frac{\Omega_{ab}(\mathbf{r}_1) G_C(\mathbf{r}_2)}{r_{12}} \, d\mathbf{r}_1 d\mathbf{r}_2, \tag{108}$$

where G_C is a primitive function of auxiliary basis set.

Applying the same machinery of the MD method we obtain:

$$(ab|C) \quad = \quad \frac{2\pi^{5/2}}{pq\sqrt{p+q}} \sum_{tuv} E^{ab}_{tuv} \sum_{t'u'v'} (-1)^{t'+u'+v'} E^{C}_{t'u'v'} R_{t+t',u+u',v+v'}(\alpha, R_{PC}), \tag{109}$$

where c is the exponent of G_C and

$$P_{x1} \quad = \quad \frac{aA_{x1} + bB_{x1}}{p}, \tag{110}$$

$$p \quad = \quad a + b, \tag{111}$$

$$\alpha \quad = \quad \frac{pc}{p+c}. \tag{112}$$

6.9.1 The MD4 Algorithm for 3-Index TEI

Let us define

$$F^{C}_{t'u'v'} \quad = \quad (-1)^{t'+u'+v'} E^{C}_{t'u'v'}, \tag{113}$$

and

$$\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}) \quad = \quad \frac{2\pi^{5/2}}{pc\sqrt{p+c}} R_{t+t',u+u',v+v'}(\alpha, R_{PC}), \tag{114}$$

Then, we may write

$$(ab|C) \;=\; \sum_{tuv} E^{ab}_{tuv} \sum_{t'u'v'} F^{C}_{t'u'v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}). \tag{115}$$

For efficiency we evaluate above equation in two steps as follows

$$(ab|C) \;=\; \sum_{tuv} E^{ab}_{tuv} g^{C}_{tuv}, \tag{116}$$

where

$$g^{C}_{tuv} \;=\; \sum_{t'u'v'} F^{Q}_{t'u'v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}). \tag{117}$$

Further efficiency can be achieved by decomposing ERI and g^{C}_{tuv} into Cartesian components. Let us start with g^{C}_{tuv}

$$g^{C}_{tuv} \;=\; g^{i'k'm'}_{tuv}, \tag{118}$$

and

$$g^{i'k'm'}_{tuv} \;=\; \sum_{t'} F^{i'}_{t'} \sum_{u'} F^{k'}_{u'} \sum_{v'} F^{m'}_{v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}). \tag{119}$$

Then, we can write

$$(ab|C) \;=\; g^{i'k'm'}_{ijklmn}, \tag{120}$$

$$g^{i'k'm'}_{ijklmn} \;=\; \sum_{t} E^{ij}_{t} \sum_{u} E^{kl}_{u} \sum_{v} E^{mn}_{v} g^{i'k'm'}_{tuv}. \tag{121}$$

7 Obra-Saika (OS) Scheme

7.1 3-Index Integrals

Let us remember

$$(ab|C) = \sum_{tuv} E^{ab}_{tuv} \sum_{t'u'v'} (-1)^{t'+u'+v'} E^{C}_{t'u'v'} \mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}), \tag{122}$$

where $p = a + b$ and c is the exponent of G_C and

$$\alpha = \frac{pc}{p+c}, \tag{123}$$

and

$$\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}) = \frac{2\pi^{5/2}}{pc\sqrt{p+c}} R_{t+t',u+u',v+v'}(\alpha, R_{PC}). \tag{124}$$

Now, let us introduce an auxiliary integral as follows:

$$\begin{aligned} \Theta^{N}_{i_x j_x k_x; i_y j_y k_y; i_z j_z k_z} &= (-2\alpha)^{-N} \sum_{tuv} E^{i_x j_x}_{t} E^{i_y j_y}_{u} E^{i_z j_z}_{v} \\ &\times \sum_{t'u'v'} (-1)^{t'+u'+v'} E^{k_x}_{t'} E^{k_y}_{u'} E^{k_z}_{v'} \mathcal{R}^{N}_{t+t',u+u',v+v'}(\alpha, R_{PC}), \end{aligned} \tag{125}$$

note that

$$\Theta^{0}_{i_x j_x k_x; i_y j_y k_y i_z j_z k_z} = g_{i_x j_x k_x; i_y j_y k_y; i_z j_z k_z}. \tag{126}$$

Now let us define

$$\mathcal{F}_N(\alpha, R_{PC}) = \frac{2\pi^{5/2}}{pc\sqrt{p+c}} F_N(\alpha R^2_{PC}), \tag{127}$$

and introduce an auxiliary function as follows:

$$\Theta^N_{ijk} = (-2\alpha)^{-N} \sum_{tuv} \sum_{t'u'v'} (-1)^{t'+u'+v'} E^{ij}_t E^k_{t'} E_{uvu'v'} \mathcal{R}^N_{t+t',u+u',v+v'}(\alpha, R_{PC}), \quad (128)$$

where

$$E_{uvu'v'} = E^{i_y j_y}_u E^{i_z j_z}_v E^{k_y}_{u'} E^{k_z}_{v'}, \quad (129)$$

and $N \leq l_1 + l_2 + l_3$.

7.1.1 The OS2 Algorithm for 3-Index TEI

In this section let us use an abstract notation for the auxiliary integral as follows:

$$\Theta^N_{abC} = (-2\alpha)^{-N} \sum_{tuv} E^{ab}_{tuv} \sum_{t'u'v'} (-1)^{t'+u'+v'} E^C_{t'u'v'} \mathcal{R}^N_{t+t',u+u',v+v'}(\alpha, R_{PC}), \quad (130)$$

note that

$$\Theta^0_{abC} = (ab|C). \quad (131)$$

In this notation angular momentum components of a G_a function are i_a, j_a, k_a for x, y, z, directions, respectively.

We may start with the following auxiliary integral:

$$\Theta^N_{000} = K^x_{AB} K^y_{AB} K^z_{AB} \mathcal{F}_N(\alpha R^2_{PC}), \quad (132)$$

where $N \leq l_1 + l_2 + l_3$

Then, we use Ahlrich's OS2 algorithm.[52] In the OS2 algorithm we start with Eq.(132) and apply vertical recurrence relation (VRR) to increment the angular momentum of the

first function on the bra side (given here for the x direction) as:

$$\Theta^N_{a+1_x,0,0} \;=\; X_{PA}\Theta^N_{a00} \;-\; \frac{\alpha}{p}X_{PC}\Theta^{N+1}_{a00} \;+\; \frac{i_a}{2p}\Big(\Theta^N_{a-1_x,0,0} - \frac{\alpha}{p}\Theta^{N+1}_{a-1_x,0,0}\Big), \qquad (133)$$

where $i_a \leq l_1 + l_2$.

When all $\Theta^N_{a+b,0,0}$ functions are computed, then we proceed to the ket side as the following VRR:

$$\Theta^N_{a,0,C+1_x} \;=\; \frac{\alpha}{c}X_{PC}\Theta^{N+1}_{a,0,C} \;+\; \frac{i_a\alpha}{2pc}\Theta^{N+1}_{a-1_x,0,C} \;+\; \frac{i_C}{2c}\Big(\Theta^N_{a,0,C-1_x} - \frac{\alpha}{c}\Theta^{N+1}_{a,0,C-1_x}\Big), \quad (134)$$

if auxiliary function is a solid harmonic then,

$$\Theta^N_{a,0,C+1_x} \;=\; \frac{\alpha}{c}X_{PC}\Theta^{N+1}_{a,0,C} \;+\; \frac{i_a}{2(p+c)}\Theta^{N+1}_{a-1_x,0,C}, \qquad (135)$$

where $i_C \leq l_3$.

Now we have all $\Theta^N_{a+b,0,C}$ functions. Hence, we can obtain all $(a+b,0|C)$ type integrals from $\Theta^0_{a+b,0,C}$. The next step is *horizontal recursion relation* (HRR) in the contracted basis:[8]

$$(a, b+1_x|C) \;=\; (a+1_x, b|C) \;+\; R_{AB}(ab|C). \qquad (136)$$

7.1.2 The OS1 Algorithm for 3-Index TEI

In the OS1 algorithm we start with Eq.(132) with $N \leq l_1 + l_2 + l_3$ and apply VRR to increment the angular momentum of the first function on the bra side (given here for the x direction) as:

$$\Theta^N_{a+1_x,0,0} \;=\; X_{PA}\Theta^N_{a00} \;-\; \frac{\alpha}{p}X_{PC}\Theta^{N+1}_{a00} \;+\; \frac{i_a}{2p}\Big(\Theta^N_{a-1_x,0,0} - \frac{\alpha}{p}\Theta^{N+1}_{a-1_x,0,0}\Big), \qquad (137)$$

where $i_a \leq l_1 + l_2 + l_3$.

When all $\Theta^N_{a+b+C,0,0}$ functions are computed, then we proceed to the ket side as the

following the *electron transfer recursion relation*,[10] ETRR:

$$\Theta^N_{a,0,C+1_x} \;=\; -\frac{b}{c}X_{AB}\Theta^N_{a0C} \;+\; \frac{i_a}{2c}\Theta^N_{a-1_x,0,C} \;+\; \frac{i_C}{2c}\Theta^N_{a,0,C-1_x} \;-\; \frac{p}{c}\Theta^N_{a+1_x,0,C}, \qquad (138)$$

if auxiliary function is a solid harmonic then,

$$\Theta^N_{a,0,C+1_x} \;=\; -\frac{b}{c}X_{AB}\Theta^N_{a0C} \;+\; \frac{i_a}{2c}\Theta^N_{a-1_x,0,C} \;-\; \frac{p}{c}\Theta^N_{a+1_x,0,C}. \qquad (139)$$

Further, since ETRR has the same order in both side of equations, it is enough to use $(N = 0)$ in the ETRR step:

$$(a,0|C+1_x) \;=\; -\frac{b}{c}X_{AB}(a,0|C) \;+\; \frac{i_a}{2c}(a-1_x,0|C) \;-\; \frac{p}{c}(a+1_x,0|C), \qquad (140)$$

It is also possible to setup a reverse ETRR as follows:

$$(a+1_x,0|C) \;=\; -\frac{b}{p}X_{AB}(a,0|C) \;+\; \frac{i_a}{2p}(a-1_x,0|C) \;-\; \frac{c}{p}(a,0|C+1_x). \qquad (141)$$

Now we have all $(a+b,0|C)$ type integrals. The next step is the HRR as in the case of OS2.

8 First Derivatives of Integrals

The firs derivative relation for a GTO can be written as

$$\frac{\partial G_i}{\partial A_x} \;=\; 2aG_{i+1} - iG_{i-1}, \qquad (142)$$

To proceed higher derivative let us introduce the notation

$$\frac{\partial^q G_i}{\partial A_x^q} \;=\; G_i^q, \qquad (143)$$

Then we have

$$G_i^{q+1} = 2aG_{i+1}^q - iG_{i-1}^q. \tag{144}$$

Further, it is also interesting to consider the first derivatives of the overlap distribution. Differentiating the overlap distribution we get the following equations:

$$\frac{\partial \Omega_{ij}^x}{\partial A_x} = 2a\Omega_{i+1,j}^x - i\Omega_{i-1,j}^x, \tag{145}$$

$$\frac{\partial \Omega_{ij}^x}{\partial B_x} = 2b\Omega_{i,j+1}^x - j\Omega_{i,j-1}^x. \tag{146}$$

Further, we may expand the derivative relations using the Hermite functions as follows:[60]

$$\frac{\partial \Omega_{ij}^x}{\partial A_x} = \sum_{t=0}^{i+j+1} \left(2aE_t^{i+1,j} - iE_t^{i-1,j}\right)\Lambda_t, \tag{147}$$

$$\frac{\partial \Omega_{ij}^x}{\partial B_x} = \sum_{t=0}^{i+j+1} \left(2bE_t^{i,j+1} - jE_t^{i,j-1}\right)\Lambda_t. \tag{148}$$

9 The MD Scheme for the First Derivatives

9.1 Overlap Derivatives

At first note that we have a *translational invarince* relation (TIR):

$$\frac{\partial S_{ab}}{\partial A_x} + \frac{\partial S_{ab}}{\partial B_x} = 0, \tag{149}$$

where A and B are the the centers of the first and the second GTO, respectively.

Derivatives of overlap with respect to the x coordinate of the nucleus N can be written

as:

$$\frac{\partial S_{ab}}{\partial N_x} \;=\; \frac{\partial S_{ab}}{\partial A_x}\delta_{AN} \;+\; \frac{\partial S_{ab}}{\partial B_x}\delta_{BN}, \tag{150}$$

using TIR we may write:

$$\frac{\partial S_{ab}}{\partial N_x} \;=\; \frac{\partial S_{ab}}{\partial A_x}\,\big(\delta_{AN} - \delta_{BN}\big). \tag{151}$$

Hence, we just need to differentiate with respect to one of centers only.

Using derivative properties of overlap distribution we obtain the following final formulas:

$$\frac{\partial S_{ab}}{\partial A_x} \;=\; \big(2aE_0^{i+1,j} - iE_0^{i-1,j}\big)E_0^{kl}E_0^{mn}\big(\frac{\pi}{p}\big)^{3/2}, \tag{152}$$

$$\frac{\partial S_{ab}}{\partial A_y} \;=\; \big(2aE_0^{k+1,l} - kE_0^{k-1,l}\big)E_0^{ij}E_0^{mn}\big(\frac{\pi}{p}\big)^{3/2}, \tag{153}$$

$$\frac{\partial S_{ab}}{\partial A_z} \;=\; \big(2aE_0^{m+1,n} - mE_0^{m-1,n}\big)E_0^{ij}E_0^{kl}\big(\frac{\pi}{p}\big)^{3/2}. \tag{154}$$

9.2 Dipole Derivatives

Let us express x-component of μ_{ab}^x more explicitly:

$$\mu_{ab}^x \;=\; \langle G_a|\mathbf{x}_C|G_b\rangle, \tag{155}$$

where $\mathbf{C}$ is the origin of the Cartesian dipole-moment, typically zero.

For the dipole derivatives, the translational invariance relation is:

$$\frac{\partial \mu_{ab}}{\partial A_x} \;+\; \frac{\partial \mu_{ab}}{\partial B_x} \;+\; \frac{\partial \mu_{ab}}{\partial C_x} \;=\; 0, \tag{156}$$

The last term is equal:

$$\frac{\partial \mu_{ab}}{\partial C_x} \;=\; -S_{ab}, \tag{157}$$

hence,

$$\frac{\partial \mu_{ab}}{\partial A_x} \;+\; \frac{\partial \mu_{ab}}{\partial B_x} \;-\; S_{ab} \;=\; 0, \tag{158}$$

If the dipole center is zero, then:

$$\frac{\partial \mu_{ab}}{\partial A_x} \;+\; \frac{\partial \mu_{ab}}{\partial B_x} \;=\; 0. \tag{159}$$

Derivatives of overlap with respect to the x coordinate of the nucleus N can be written as:

$$\frac{\partial \mu_{ab}}{\partial N_x} \;=\; \frac{\partial \mu_{ab}}{\partial A_x}\delta_{AN} \;+\; \frac{\partial \mu_{ab}}{\partial B_x}\delta_{BN} \;+\; \frac{\partial \mu_{ab}}{\partial C_x}\delta_{CN}, \tag{160}$$

If the dipole origin is zero, then:

$$\frac{\partial \mu_{ab}}{\partial N_x} \;=\; \frac{\partial \mu_{ab}}{\partial A_x}\big(\delta_{AN} - \delta_{BN}\big). \tag{161}$$

Using derivative relation of primitive Gaussians, we obtain derivatives for the x-component of the dipole integrals as follows:

$$\frac{\partial \mu^x_{ab}}{\partial A_x} \;=\; \big(2a\mu_{i+1,j} - i\mu_{i-1,j}\big)S_{kl}S_{mn}, \tag{162}$$

$$\frac{\partial \mu^x_{ab}}{\partial A_y} \;=\; \mu_{ij}\big(2aS_{k+1,l} - kS_{k-1,l}\big)S_{mn}, \tag{163}$$

$$\frac{\partial \mu^x_{ab}}{\partial A_z} = \mu_{ij} S_{kl} \big(2a S_{m+1,n} - m S_{m-1,n} \big). \tag{164}$$

Similarly, the derivatives for the y-component of the dipole integrals can be written as follows:

$$\frac{\partial \mu^y_{ab}}{\partial A_x} = \big(2a S_{i+1,j} - i S_{i-1,j} \big) \mu_{kl} S_{mn}, \tag{165}$$

$$\frac{\partial \mu^y_{ab}}{\partial A_y} = \big(2a \mu_{k+1,l} - k \mu_{k-1,l} \big) S_{ij} S_{mn}, \tag{166}$$

$$\frac{\partial \mu^y_{ab}}{\partial A_z} = S_{ij} \mu_{kl} \big(2a S_{m+1,n} - m S_{m-1,n} \big). \tag{167}$$

Finally, the derivatives for the z-component of the dipole integrals are:

$$\frac{\partial \mu^z_{ab}}{\partial A_x} = \big(2a S_{i+1,j} - i S_{i-1,j} \big) S_{kl} \mu_{mn}, \tag{168}$$

$$\frac{\partial \mu^z_{ab}}{\partial A_y} = \big(2a S_{k+1,l} - k S_{k-1,l} \big) S_{ij} \mu_{mn}, \tag{169}$$

$$\frac{\partial \mu^z_{ab}}{\partial A_z} = S_{ij} S_{kl} \big(2a \mu_{m+1,n} - m \mu_{m-1,n} \big). \tag{170}$$

9.3 Kinetic Energy Derivatives

Using the TIR, derivatives of kinetic energy integrals with respect to the x coordinate of the nucleus N can be written as:

$$\frac{\partial T_{ab}}{\partial N_x} = \frac{\partial T_{ab}}{\partial A_x} \big(\delta_{AN} - \delta_{BN} \big), \tag{171}$$

Using derivative properties of overlap distribution we obtain the following final formula for the derivative with respect to A_x:

$$\frac{\partial T_{ab}}{\partial A_x} = \big(2aT_{i+1,j} - iT_{i-1,j}\big)S_{kl}S_{mn} + \big(2aS_{i+1,j} - iS_{i-1,j}\big)\big(T_{kl}S_{mn} + S_{kl}T_{mn}\big). \quad (172)$$

Explicit equations for overlap (S_{ij}) and kinetic integrals (T_{ij}) were given in Eq.(73) and Eq.(78). One may write the derivatives with respect to A_y and A_z in a similar way.

9.4 Potential Energy Derivatives

Before starting to derivatives, let us define a more general potential function:

$$V_{ab,C}^{qrs} = \langle G_a|\Big(\frac{\partial}{\partial C_x}\Big)^q\Big(\frac{\partial}{\partial C_y}\Big)^r\Big(\frac{\partial}{\partial C_z}\Big)^s\frac{1}{r_C}|G_b\rangle, \quad (173)$$

It becomes:

$$V_{ab,C}^{qrs} = (-1)^{q+r+s}\,\frac{2\pi}{p}\sum_{tuv}E_{tuv}^{ab}R_{t+q,u+r,v+s}(p,R_{PC}), \quad (174)$$

Hence, derivatives with respect to the nucleus N can be written as:

$$\frac{\partial V_{ab}}{\partial N_x} = \frac{\partial V_{ab}}{\partial A_x}\,\delta_{AN} + \frac{\partial V_{ab}}{\partial B_x}\,\delta_{BN} - \Big[\sum_C Z_C V_{ab,C}^{100}\,\delta_{CN}\Big], \quad (175)$$

$$\frac{\partial V_{ab}}{\partial N_y} = \frac{\partial V_{ab}}{\partial A_y}\,\delta_{AN} + \frac{\partial V_{ab}}{\partial B_y}\,\delta_{BN} - \Big[\sum_C Z_C V_{ab,C}^{010}\,\delta_{CN}\Big], \quad (176)$$

$$\frac{\partial V_{ab}}{\partial N_z} = \frac{\partial V_{ab}}{\partial A_z}\,\delta_{AN} + \frac{\partial V_{ab}}{\partial B_z}\,\delta_{BN} - \Big[\sum_C Z_C V_{ab,C}^{001}\,\delta_{CN}\Big], \quad (177)$$

where

$$V_{ab,C}^{100} = -\frac{2\pi}{p}\sum_{tuv} E_{tuv}^{ab} R_{t+1,u,v}(p, R_{PC}), \tag{178}$$

$$V_{ab,C}^{010} = -\frac{2\pi}{p}\sum_{tuv} E_{tuv}^{ab} R_{t,u+1,v}(p, R_{PC}), \tag{179}$$

$$V_{ab,C}^{001} = -\frac{2\pi}{p}\sum_{tuv} E_{tuv}^{ab} R_{t,u,v+1}(p, R_{PC}). \tag{180}$$

The TIR for potential energy derivatives can be written as follows:[61]

$$\frac{\partial V_{ab,C}}{\partial A_x} + \frac{\partial V_{ab,C}}{\partial B_x} + V_{ab,C}^{100} = 0, \tag{181}$$

$$\frac{\partial V_{ab,C}}{\partial A_y} + \frac{\partial V_{ab,C}}{\partial B_y} + V_{ab,C}^{010} = 0, \tag{182}$$

$$\frac{\partial V_{ab,C}}{\partial A_z} + \frac{\partial V_{ab,C}}{\partial B_z} + V_{ab,C}^{001} = 0. \tag{183}$$

Therefore, the derivative with respect to N_x can be written as follows:

$$\frac{\partial V_{ab}}{\partial N_x} = \frac{\partial V_{ab}}{\partial A_x}\,\delta_{AN} + \frac{\partial V_{ab}}{\partial B_x}\,\delta_{BN} + Z_N\Big(\frac{\partial V_{ab,N}}{\partial A_x} + \frac{\partial V_{ab,N}}{\partial B_x}\Big). \tag{184}$$

Hence, the explicit equations become:

$$\frac{\partial V_{ab}}{\partial A_x} = \frac{2\pi}{p}\sum_{tuv}\Big(2aE_t^{i+1,j} - iE_t^{i-1,j}\Big)E_u^{kl}E_v^{mn}R'_{tuv}, \tag{185}$$

and

$$\frac{\partial V_{ab}}{\partial B_x} = \frac{2\pi}{p}\sum_{tuv}\Big(2bE_t^{i,j+1} - jE_t^{i,j-1}\Big)E_u^{kl}E_v^{mn}R'_{tuv}. \quad (186)$$

9.5 Metric Derivatives

At first note that we have the following TIR for metric integrals:

$$\Big(\frac{\partial G_a}{\partial A_x}|\frac{1}{r_{12}}|G_b\Big) + \Big(G_a|\frac{1}{r_{12}}|\frac{\partial G_b}{\partial B_x}\Big) = 0, \quad (187)$$

Hence, if both orbitals have the same center, then the first derivative is zero. The derivative with respect to the center N may be written as follows:

$$\frac{\partial(G_a|G_b)}{\partial N_x} = \Big(\frac{\partial G_a}{\partial A_x}|G_b\Big)\big(\delta_{AN} - \delta_{BN}\big). \quad (188)$$

Using derivative relation of primitive Gaussians, we obtain derivatives along the x-axis as follows:

$$\frac{\partial(a|b)}{\partial A_x} = \sum_{tuv}\big(2aE_t^{i+1} - iE_t^{i-1}\big)E_u^kE_v^m\sum_{t'u'v'}F^b_{t'u'v'}\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}), \quad (189)$$

$$\frac{\partial(a|b)}{\partial A_y} = \sum_{tuv}E_t^i\big(2aE_u^{k+1} - kE_u^{k-1}\big)E_v^m\sum_{t'u'v'}F^b_{t'u'v'}\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}), \quad (190)$$

$$\frac{\partial(a|b)}{\partial A_z} = \sum_{tuv}E_t^iE_u^k\big(2aE_v^{m+1} - mE_v^{m-1}\big)\sum_{t'u'v'}F^b_{t'u'v'}\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{AB}). \quad (191)$$

9.6 3-Index Derivatives

At first note that we have the following TIR for 3-index integrals:

$$\frac{\partial(ab|C)}{\partial A_x} + \frac{\partial(ab|C)}{\partial B_x} + \frac{\partial(ab|C)}{\partial C_x} = 0, \tag{192}$$

Hence, if all orbitals have the same center, then the first derivative is zero. The derivative with respect to the center N may be written as follows:

$$\frac{\partial(ab|C)}{\partial N_x} = \frac{\partial(ab|C)}{\partial A_x}\left(\delta_{AN} - \delta_{CN}\right) + \frac{\partial(ab|C)}{\partial B_x}\left(\delta_{BN} - \delta_{CN}\right). \tag{193}$$

Therefore, we need only derivatives with respect to the centers A and B.

Using derivative relation of primitive Gaussians, we obtain derivatives along the x-axis as follows:

$$\begin{aligned}\frac{\partial(ab|C)}{\partial A_x} &= \sum_{tuv}\left(2aE_t^{i+1,j} - iE_t^{i-1,j}\right)E_u^{kl}E_v^{mn} \\ &\times \sum_{t'u'v'} F^C_{t'u'v'}\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}),\end{aligned} \tag{194}$$

$$\begin{aligned}\frac{\partial(ab|C)}{\partial B_x} &= \sum_{tuv}\left(2bE_t^{i,j+1} - jE_t^{i,j-1}\right)E_u^{kl}E_v^{mn} \\ &\times \sum_{t'u'v'} F^C_{t'u'v'}\mathcal{R}_{t+t',u+u',v+v'}(\alpha, R_{PC}).\end{aligned} \tag{195}$$

Similar expressions can be readily written for the y and z components of the centers.

10 3-Index Integral Derivatives in the OS Scheme

Using the derivaive properties of GTOs we can write:

$$\frac{\partial(ab|C)}{\partial A_\delta} \;=\; 2a(a+1_\delta, b|C) \;-\; i_a(a-1_\delta, b|C), \tag{196}$$

and

$$\frac{\partial(ab|C)}{\partial B_\delta} \;=\; 2b(a, b+1_\delta, |C) \;-\; i_b(a, b-1_\delta|C), \tag{197}$$

where $\delta = x, y, z$. Hence, following prescription of Head-Gordon and Pople,[8] we can generate the all derivative equations from Eq.(196) and (197).

11 Density-Fitting

The AO basis integrals can be cast into the following form with the help of DF approach:

$$(\mu\nu|\lambda\sigma)_{DF} \;=\; \sum_{Q}^{N_{aux}} b^Q_{\mu\nu} b^Q_{\lambda\sigma}, \tag{198}$$

The DF tensors $b^Q_{\mu\nu}$ are defined as follows:

$$b^Q_{\mu\nu} \;=\; \sum_{P}^{N_{aux}} (\mu\nu|P)[\mathbf{J}^{-1/2}]_{PQ}, \tag{199}$$

where

$$(\mu\nu|P) \;=\; \int\int \chi_\mu(\mathbf{r_1})\chi_\nu(\mathbf{r_1})\frac{1}{r_{12}}\varphi_P(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}, \tag{200}$$

and

$$J_{PQ} \;=\; \int\int \varphi_P(\mathbf{r_1}) \frac{1}{r_{12}} \varphi_Q(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}, \tag{201}$$

where $\chi_\mu(\mathbf{r})$ and $\varphi_P(\mathbf{r})$ are the primary and auxiliary basis set members, respectively.

12 Integral Direct Fock

In the AO basis, the Fock matrix can be written as follows:

$$f_{\mu\nu} \;=\; h_{\mu\nu} \;+\; \sum_{\lambda,\sigma} D_{\lambda\sigma} \big[(\mu\nu|\lambda\sigma) \;-\; \frac{1}{2}(\mu\lambda|\nu\sigma) \big], \tag{202}$$

where $\boldsymbol{D}$ is the density matrix. For example, for the closed-shell case:

$$D_{\mu\nu} \;=\; 2 \sum_{i}^{occ} C_{\mu i} C_{\nu i}, \tag{203}$$

where $C_{\mu i}$ are the MO coefficients.

Two-electron part of the Fock matrix consists of so called J- and K-terms. The J-term can be written as:

$$J_{\mu\nu} \;=\; \sum_{\lambda,\sigma} D_{\lambda\sigma} (\mu\nu|\lambda\sigma). \tag{204}$$

Applying the DF technique we obtain the following final formula for $J_{\mu\nu}$:

$$J_{\mu\nu} \;=\; \sum_{P}^{N_{aux}} (\mu\nu|P) D'''_P, \tag{205}$$

where,

$$D'''_P \;=\; \sum_{R}^{N_{aux}} [\mathbf{J}^{-1/2}]_{PR} D''_R, \tag{206}$$

$$D''_R = \sum_Q^{N_{aux}} [\mathbf{J}^{-1/2}]_{RQ} D'_Q, \tag{207}$$

$$D'_Q = \sum_{\lambda,\sigma} D_{\lambda\sigma}(\lambda\sigma|Q). \tag{208}$$

Hence, we can compute the 3-index integrals on the fly and contract them with the density matrix to obtain $J_{\mu\nu}$.

The K-term can be written as:

$$K_{\mu\nu} = \sum_{\lambda,\sigma} D_{\lambda\sigma}(\mu\lambda|\nu\sigma), \tag{209}$$

We can compute the K-term in a *semi-direct* algorithm with three steps, as follows:

- Step-1:

$$(Q|\nu i) = \sum_\sigma C_{\sigma i}(Q|\nu\sigma) \quad cost = O * N^2 * N_{aux}, \tag{210}$$

- Step-2:

$$C^P_{\nu i} = \sum_Q^{N_{aux}} [\mathbf{J}^{-1/2}]_{PQ}(Q|\nu i) \quad cost = O * N * N^2_{aux}, \tag{211}$$

- Step-3:

$$K_{\mu\nu} = \sum_P^{N_{aux}} \sum_i^{occ} C^P_{\mu i} C^P_{\nu i} \quad cost = O * N^2 * N_{aux}. \tag{212}$$

The scaling of this algorithm is $O(N^4)$ and memory usage is $2 * O * N * N_{aux}$.

13 Analytic Gradients

For a general density-fitted wave function based method, the analytic gradient equation in the AO basis can be written as follows[42]

$$\frac{\mathrm{d}E}{\mathrm{d}x} = \sum_{\mu\nu} \gamma_{\mu\nu} h^x_{\mu\nu} - \sum_{\mu\nu} F_{\mu\nu} S^x_{\mu\nu} + \sum_{Q}^{N_{aux}} \sum_{\mu\nu} \Gamma^Q_{\mu\nu} (Q|\mu\nu)^x - \sum_{P,Q}^{N_{aux}} \Gamma_{PQ} \mathbf{J}^x_{PQ}. \quad (213)$$

where $\gamma_{\mu\nu}$ is the one-particle density matrix (OPDM), $F_{\mu\nu}$ is the generalized Fock matrix (GFM), Γ_{PQ} and $\Gamma^Q_{\mu\nu}$ are the 2- and 3-index two-particle density matrices (TPDM), $h^x_{\mu\nu}$, $S^x_{\mu\nu}$, $\mathbf{J}^x_{PQ}$, and $(Q|\mu\nu)^x$ are the core Hamiltonian, overlap, metric, and 3-index integral derivatives, respectively. All explicit equations for the PDMs and GFM were presented in our previous papers.[42,48,50,51]

14 MOLINT Framework

The MOLINT framework consists of MOLINT, TENSOR, MOLECULE, OPTIONS, MASTER, and BASIS libraries (Figure 1). Before illustrating the usage of the MOLINT framework, let us introduce the core libraries.

14.1 TENSOR Library

The TENSOR library, which is written in C++, consists of six classes. These are `Tensor1d`, `Tensor2d`, and `Tensor3d` for one-, two-, and three-dimensional arrays of `double`, respectively, and `Tensor1i`, `Tensor2i`, and `Tensor3i` for one-, two-, and three-dimensional arrays of `integer`, respectively. The TENSOR library includes numerous functions to handle, manipulate, and operate large arrays of pointers. As an example, we illustrate some prominent features of the `Tensor2d` class.

The `Tensor2d` class is developed to perform various tensor operations in a convenient way. It handles 2, 3, and 4 dimensional tensors, and store them like a matrix. Further,

the class take advantages of *shared pointers*. The memory allocation for a 2D tensor can be performed with the `Tensor2d` class as follows:

```
Tensor2d(std::string name, int d1, int d2);
```

where, `name`: The lable of tensor, `d1` and `d2` are the first and second dimensions of the tensor. An example of memory allocation is:

```
SharedTensor2d CmoA =
    SharedTensor2d(new Tensor2d("Alpha MO Coefficients", nao, nmo));
```

where, `CmoA` is the MO coefficient matrix, `nao` and `nmo` are the number of the AOs and MOs, respectively.

Similarly, 3- and 4-dimensional arrays can be allocated as follows:

```
K = SharedTensor2d(new Tensor2d("DFBCC B (Q|IA)", nQ, naoccA, navirA));
```

```
T = SharedTensor2d(new Tensor2d("T2 <Ij|Ab>", naoccA, naoccB, navirA, navirB));
```

Some important functions and example usages are:

- **Print:** `void print(type output);`

  Example: `T->print(output);` where `output` can be type of `std::string`, `const char*`, and `FILE*`.

- **Set value:** `void set(int i, int j, double value);`

  Example: `T->set(3, 5, 12.0);`

- **Set Tensor:** `void set(SharedTensor2d &A);`

  Example: `T->set(A);`

- **Add value:** `void add(int i, int j, double value);`

  Example: `T->add(3, 5, 12.0);`

- **Add Tensor:** `void add(SharedTensor2d &A);`

  Example: `T->add(A);`

- **Scale:** `void scale(double alpha);` where `alpha` is the scaling value.

  Example: `T->scale(2.0);`

- **AXPY:** The `axpy` function is used to add/subtract a SharedTensor via BLAS routine.

  `void axpy(const SharedTensor2d &A, double alpha);`

  Example: `T->axpy(2.0);`

- **Transpose:** `void trans(const SharedTensor2d &A);`

  Example: `T->trans(A);`

- **Diagonalize:** `void diagonalize(const SharedTensor2d &eigvectors,`

  `const SharedTensor1d &eigvalues, double cutoff, bool ascending);`

  Example: `FockA->diagonalize(UeigA, eigA, 1.0E-10, true);`

- **Linear equation solver for $\mathbf{Ax} = \mathbf{b}$:**

  `void cdgesv(const SharedTensor1d &b, int errcode);`

  Example: `A->cdgesv(b, errcode);`

- **Matrix multiplication of $\mathbf{C} = \alpha\mathbf{AB} + \beta\mathbf{C}$ type:**

  `void gemm(bool transa, bool transb, const SharedTensor2d &A,`

  `const SharedTensor2d &B, double alpha, double beta);`

  Example: `C->gemm(false, false, A, B, 1.0, 0.0);`

- **General contraction of $C_{m,n} = \alpha \sum_k A_{m,k}B_{k,n} + \beta C_{m,n}$ type:**

  `void contract(bool transa, bool transb, int m, int n, int k,`

  `const SharedTensor2d &A, const SharedTensor2d &B, double alpha, double beta);`

  Example: Let us try to evaluate the contraction of $x_{ij}^{ab} = \sum_e t_{ij}^{ae} F_{be}$, where i, j active occupied and a, b, e are active virtual indices:

```
X->contract(false, true, naoccA*navirA*naoccA, navirA, navirA, T, F, 1.0,
0.0);
```

- **Transformation of $\mathbf{C} = \mathbf{L}^T\mathbf{AL}$ type:**

  ```
  void transform(const SharedTensor2d &A, const SharedTensor2d &L);
  ```

  Example: `C->transform(A, L);`

- **Dot product of $value = \sum_{mnk\ldots} A_{mnk\ldots}B_{mnk\ldots}$ type:**

  ```
  double vector_dot(const SharedTensor2d &rhs);
  ```

  Example: `double value = A->vector_dot(B);`

- **Orthogonalization via modified Gram-Schmid algorithm:**

  ```
  void mgs();
  ```

  Example: `A->mgs();`

14.2 OPTIONS Library

The OPTIONS library (`liboptions`) read the input file, default is `input.inp`, which includes geometry and the MFW options. The MFW accepts the following user options:

- `nocom`: Do not shift the geometry to the center-of-mass. `Type`: Boolean, `Default`: false.
- `puream`: Do use pure angular momentum functions? `Type`: Boolean, `Default`: true.
- `print_level`: This options is a flag to control printing extra information. `Type`: integer, `Default`: 0.
- `geom_units`: Unit of the geometry. `Type`: string, `Possible values`: ang (default), bohr.
- `integral_cutoff`: Integral cutoff value. `Type`: double, `Default`: 1.0E-10.

- `boys_cutoff`: The cutoff value used in the evaluation of the Boys function. `Type`: double, `Default`: 1.0E-15.
- `boys_zero`: The computational zero used in the evaluation of the Boys function. `Type`: double, `Default`: 1.0E-6.
- `delta_x`: The step size in the Boys grid. `Type`: double, `Default`: 0.1.
- `x_max`: The maximum value of x for the Boys grid. `Type`: int, `Default`: 30.
- `boys_maxiter`: The maximum number of iterations in computing values of the Boys grid. `Type`: int, `Default`: 1000.
- `taylor_order`: The order of Taylor expansion in computing the Boys function. `Type`: int, `Default`: 6.
- `boys_method`: Method to compute the Boys grid. `Type`: string, `Possible values`: ghp (default), saunders.

```
$set_ints
nocom true
integral_cutoff 1.0e-10
boys_cutoff 1.0e-15
delta_x 0.1
x_max 30
taylor_order 6
print_level 0
boys_method ghp
$end_ints
```

14.3 MOLECULE Library

The MOLECULE library (`libmolecule`) read geometry, charge, and multiplicity information, and pass all necessary information to LIBMOLINT, which includes molecular integrals. An example input is given as follows:

```
$set_molecule
0 1
O 0.000000000000 0.000000000000 −0.065775570538
H 0.000000000000 −0.759061990794 0.521953018295
H 0.000000000000 0.759061990794 0.521953018295
$end_molecule
```

The current version of the MOLECULE library accepts Cartesian coordinates only. The `libmolecule` is directly called by the `libmolint`, hence; there is no need to call it when interfacing the MFW

14.4 MASTER Library

The MASTER library (`libmaster`) is built for the purpose of the communication within the MFW. In the MFW, users directly call `libmolint` only, and the constructor function creates the `libmaster` object, let us denote it `master`. Then, `master` gets all necessary options using the `liboptions` and store them. Similarly, the `master` get all necessary molecular info, such as coordinates, inter atomic distances, nuclear repulsion energy and gradients, and store them. Users can access the `master` object through the `libmolint`, and can get all information stored in it. Let us denote the object of the `libmolint` as `ints`. Then, users can access the `master` as follows:

```
SharedMaster master = ints->master();
```

The object `master` is reset in the destructor function of the `libmolint`.

14.5 BASIS Library

The BASIS library (`libbasis`) is a built-in library, which reads in all required basis set(s) information for a given molecule. The `libbasis` is directly called by in the `libmolint`, hence; there is no need to call it when interfacing the MFW. Further, `libbasis` includes common basis functions in a `share` directory in the GAUSSIAN format. One can readily add a new basis set as a `.gbs` file. The address of the `share` directory is defined as an environmental variable `MFWDATADIR`. Hence, users should export it, for example:

```
export MFWDATADIR=/Users/ugur/lib/molintfw/share
```

14.6 MOLINT Library

The usage and features of the core libraries were explained in the previous sections. In here, we focus on the `Molint` library (`libmolint`), which is capable of computing overlap, kinetic, potential, dipole, metric, and 3-index integrals, as well as their first derivatives (Figure 2).

Now, let us demonstrate how to use the MFW. The constructor function for the `libmolint` is described as follows:

```
Integrals(std::string file_name, std::string basis, std::string aux)
```

where `file_name` is the name of input file, `basis` and `aux` are the names of the primary and auxiliary basis sets, for which there should be `basis.gbs` and `aux.gbs` files, respectively. At first we need to construct the `libmolint` object as follows:

```
SharedIntegrals ints =
    SharedIntegrals(new Integrals("input.inp", "cc-pvdz", "cc-pvdz-ri"));
```

Basic information such as the number of atoms, atomic orbitals, auxiliary basis functions, and shells can be obtained from the object as follows:

```
natom = ints->natom();
```

```
nao = ints->nao();

naux = ints->naux();

nshell = ints->nshell();
```

14.6.1 Common Integrals

The overlap integrals can be obtained form the object as follows:

```
Sao->copy(ints->compute_overlap_ints());
```

Of course, the `Sao` array should be formed previously, such as follows:

```
auto Sao = SharedTensor2d(new Tensor2d("AO-basis Overlap Ints", nao, nao));
```

Similarly, the kinetic and potential integrals can be obtained as follows:

```
Tao->copy(ints->compute_kinetic_ints());

Vao->copy(ints->compute_potential_ints());
```

where `Tao` and `Vao` are two-dimensional tensors (Tensor2d).

The dipole integrals can be obtained as follows:

```
Mdm = ints->compute_dipole_ints();
```

where `Mdm` is a three-dimensional tensor (Tensor3d), which can be formed as follows:

```
auto Mdm = SharedTensor3d(new Tensor3d("Mu (x,y,z|mu,nu)", 3, nao, nao));
```

The metric integrals can be computed as follows:

```
ints->compute_metric_ints();

Jmetric->copy(ints->metric());
```

where `Jmetric` is a two-dimensional tensor of size `naux` $\times$ `naux`. One may obtain $[\mathbf{J}^{-1/2}]_{PQ}$ directly instead of metric integrals as follows:

```
Jmhalf->copy(ints->metric_mhalf());
```

The DF factors can be computed in two steps. In the first step one can compute three-index integrals as follows:

```
ints->compute_three_index_ints();
```

and in the second step the DF factors can be computed, according to Eq.(199), as follows:

```
ints->compute_df_factors();
```

Then, the DF factors can be obtained as follows:

```
Bdf->copy(ints->df_factors());
```

where `Bdf` is a two-dimensional tensor with row and column dimensions of `naux` and `nao*nao`, respectively. Similarly, the three-index integrals can be obtained as follows:

```
Qmn->copy(ints->three_index_tei());
```

However, in order to proceed in this way, one should compute metric integrals before the DF factors. Further, one may combine the computation of metric and DF factors, as follows:

```
ints->compute_df_integrals();
```

14.6.2 Density-Fitted Fock Matrix

So far, we have assume that there is enough memory to keep 3-index integrals in the core memory. However, for larger computations one may prefer an integral direct approach, where J- and K-terms of the Fock matrix can be computed. In our implementation we have a full direct approach for the J-term and a semi-direct approach for the K-term. The memory requirement of the semi-direct K (SDK) approach is $2 * O * N * N_{aux}$, which is significantly low (by $\frac{N}{2O}$) compare with $N^2 * N_{aux}$ for the incore case. In our JK algorithm we build the both terms simultaneously. For example, for the restricted HF (RHF) case:

```
ints->compute_dffock_JK(noccA, FaoA, CmoA, DaoA, Jmhalf);
```

where `noccA` is the number of α spin occupied orbitals, `FaoA` is the AO basis Fock matrix, `CmoA` is the MO coefficients matrix, `DaoA` is the density matrix, and `Jmhalf` is $\boldsymbol{J}^{-1/2}$.

For the unrestricted HF (UHF) case we need two call `libmolint` twice:

```
ints->compute_dffock_JK(noccA, FaoA, CmoA, Dtot, Jmhalf);

ints->compute_dffock_JK(noccB, FaoB, CmoB, Dtot, Jmhalf);
```

where `Dtot` is the total density matrix.

14.6.3 Integral Derivatives

The core-Hamiltonian term of analytic gradients in Eq.(213) $\left(\sum_{\mu\nu}\gamma_{\mu\nu}h^x_{\mu\nu}\right)$ can be computed as follows:

```
EgradH->copy(ints->compute_kinetic_grad(G1ao));

EgradH->add(ints->compute_potential_grad(G1ao));
```

where `G1ao` is the AO basis OPDM ($\gamma_{\mu\nu}$) and `EgradH` is a two-dimensional tensor, which can be allocated as follows:

```
SharedTensor2d EgradH =
    SharedTensor2d(new Tensor2d("-Core Hamiltonian Gradient:", natom, 3));
```

where `natom` is the total number of atoms.

The overlap gradient $\left(-\sum_{\mu\nu}F_{\mu\nu}S^x_{\mu\nu}\right)$ can be computed as follows:

```
EgradS->copy(ints->compute_overlap_grad(GFao));

EgradS->scale(-1.0);
```

where `GFao` is the AO basis GFM ($F_{\mu\nu}$) and `EgradS` is a two-dimensional tensor, which can be allocated as follows:

```
SharedTensor2d EgradS =
    SharedTensor2d(new Tensor2d("-Overlap Gradient:", natom, 3));
```

The two-electron part of analytic gradients should be computed in two steps since we employ different auxiliary basis sets for the reference and correlation energies. The details of this separation procedure were given in our previous studies.[42,47] The reference and separable parts (RefSep) of 2- and 3-index PDMs can be merged before the contraction. The RefSep contribution for the metric gradients $\left(-\sum_{P,Q}^{JKFIT}\Gamma_{PQ}^{(RefSep)}\mathbf{J}_{PQ}^{x}\right)$ can be computed as follows:

```
EgradM->copy(ints->compute_metric_grad(Gaux_ref));

EgradM->scale(-1.0);
```

where `Gaux_ref` is the RefSep part of the 2-index TPDM ($\Gamma_{PQ}^{(RefSep)}$) and `EgradM` is a two-dimensional tensor.

The RefSep contribution for the 3-index gradient $\left(\sum_{Q}^{JKFIT}\sum_{\mu\nu}\Gamma_{\mu\nu}^{Q}(Q|\mu\nu)^{x}\right)$ can be computed as follows:

```
Egrad3I->copy(ints->compute_3index_grad(gQao_ref));
```

where `gQao_ref` is the RefSep part of the 3-index TPDM ($\Gamma_{\mu\nu}^{Q(RefSep)}$) and `Egrad3I` is a two-dimensional tensor.

The correlation parts of the metric and 3-index gradients can be computed similarly. One needs to call the same function but with the correlation parts of 2- and 3-index TPDMs. Finally all gradient components are merged in to the final gradient.

15 Verification of the MOLINT Framework

The MFW code has been verified with respect to the PSI4[57] package for the computation of energies, dipole moments, and analytic gradients for various density-fitted methods, such as the Hartree-Fock (DF-HF), second-order perturbation theory (DF-MP2), coupled-cluster

singles and doubles (DF-CCSD), and the CCSD with perturbative triples [DF-CCSD(T)]. All comparisons were performed with my DFOCC module,[36,37,42,47,48,50,51,62,63] which is available in the PSI4[57] package, as well as it has an external version.

16 Illustrative Applications

Results from the DF-MP2 method were obtained for a set of alkanes for the assessment of computational cost for single-point energy and analytic gradient computations. For the alkanes set, Dunning's correlation-consistent polarized valence triple-, quadruple-, and quintuple-ζ basis sets (cc-pVTZ, cc-pVQZ, and cc-pV5Z) were employed with the frozen core approximation.[64,65] For the cc-pVXZ (X=T, Q, 5) primary basis sets, cc-pVXZ-JKFIT[26] and cc-pVXZ-RI[66] auxiliary basis sets were employed for reference and correlation energies, respectively. The DF-MP2 computations were performed with the external version of the DFOCC module.

We consider a set of alkanes (C_nH_{2n+2}, $n = 1 - 10$) to assess the efficiency of the MFW. The total computational (wall) time for single-point frozen-core energy and analytic gradient computations using the DF-MP2 method are presented in Tables 1–6. These computations were performed on an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer ($\sim$ 120 GB memory). In the MFW, 3-index integrals are parallelized using OPENMP API in the FORTRAN side, while all other integrals use only 1 core. The computations of overlap, kinetic, dipole, and potential integrals are already very fast; hence, there is no need to consider parallelism for them. However, the computational time for metric integrals also includes the cost of diagonalization and $\boldsymbol{J}^{-1/2}$ formation. Hence, our metric code is partially parallelized, through basic linear algebra subprogram routines (BLAS and LAPACK).

Tables 1–3 reports computational (wall) time for single-point frozen-core DF-MP2/cc-pVXZ (X=T,Q,5) energy computations with 1 and 8 cores. For the largest system considered, C_7H_{16} with cc-pV5Z/cc-pV5Z-RI, there are 1517 primary and 2807 auxiliary basis functions.

For this system, the computation of metric integrals and the formation of $\boldsymbol{J}^{-1/2}$ is performed just in 15 and 8 seconds, with 1 and 8 cores, respectively. Hence, the computations of metric and one-electron integrals are already very fast, and we need to focus on the cost of 3-index integrals, which are the time determining step in molecular integrals. The number of primary and auxiliary basis functions (N_{bf}/N_{aux}) in the largest systems considered are 608/1470 (cc-pVTZ/cc-pVTZ-RI), 1210/2530 (cc-pVQZ/cc-pVQZ-RI), and 1517/2807 (cc-pV5Z/cc-pV5Z-RI). For these systems wall time for serial (1 core) and parallel (8 cores) computations are 92.84 and 27.13 (cc-pVTZ), 630.15 and 166.49 (cc-pVQZ), and 1523.50 and 322.42 (cc-pV5Z) seconds. For the largest system considered, 1517 primary and 2807 auxiliary basis functions in the cc-pV5Z/cc-pV5Z-RI pair, the 3-index integrals are computed just in 5.4 minutes, with 8 cores, on our moderately fast server. Furthermore, for this system, our parallel code is 4.7-fold faster than the serial version. These results demonstrate that the MFW is satisfactorily efficient for the computation of 3-index integrals.

The MFW is capable of computing analytic gradients providing the AO basis PDMs and GFM. Tables 4–6 reports computational (wall) time for single-point frozen-core DF-MP2/cc-pVXZ (X=T,Q,5) analytic gradient computations with 2 cores. Our first derivative code has not been parallelized yet. However, in the case of 2 or more cores, we compute derivatives for two different centers simultaneously. For the largest system considered, C_5H_{12} with cc-pV5Z/cc-pV5Z-RI, there are 1115 primary and 2057 auxiliary basis functions. For this system, the computation of the metric term of the analytic gradient is performed just in 8 seconds. Hence, the computations of the metric and one-electron terms are already very fast, and we need to focus on the cost of the 3-index term, which is the time determining step in analytic gradient evaluation. The number of primary and auxiliary basis functions (N_{bf}/N_{aux}) in the C_5H_{12} molecule are 318/765 (cc-pVTZ/cc-pVTZ-RI), 635/1320 (cc-pVQZ/cc-pVQZ-RI), and 1115/2057 (cc-pV5Z/cc-pV5Z-RI). For these systems wall time, with 2 cores, are 50.18 (cc-pVTZ), 358.90 (cc-pVQZ), and 2285.34 (cc-pV5Z) seconds. These results demonstrate that the MFW is reasonably efficient for the evaluation of the analytic

gradients for the density-fitted methods.

17 Future Directions

The first version of MFW, MOLINT 1.0, includes the essential integrals and their first derivatives, as well as integral direct/semi-direct Fock and analytic gradients. In future releases, we intend to include more challenging features such as the second derivatives, R12 and/or F12 integrals, and gauge including atomic orbital (GIAO) integrals. Furthermore, a parallel API framework for the graphical process unit (GPU) is in our "to do" list. However, all these developments require a significant effort; hence, we believe that the MFW will cover these features in the next years.

18 Conclusions

In this research, an application programming interface (API) framework, denoted MOLINT (MFW), for the computation of molecular integrals and their first derivatives, over contracted Gaussian functions, for the density-fitted methods has been reported. The MFW includes overlap, dipole, kinetic, potential, metric, and 3-index integrals as well as their first derivatives. Furthermore, the MFW provides a smooth approach to build the Fock matrix and evaluate analytic gradients. The MFW is a C++/FORTRAN hybrid code, which can take advantage of shared-memory parallel programming techniques. Moreover, the MFW is free software, which can be obtained through an e-mail request from the developer (`http://www.bozkayalab.hacettepe.edu.tr/en/menu/molint-47`). The illustrative applications have demonstrated that the MFW is an efficient and user-friendly API for the computation of molecular integrals and their first derivatives. Consequently, considering both the computational efficiency and an easy-to-use structure of the MFW, we conclude that the MFW emerges as a very useful tool for computational quantum chemistry.

Acknowledgement

I thank Yavuz Alagöz and Betül Ermiş for their assistance to run test computations.

References

(1) Helgaker, T.; Jørgensen, P.; Olsen, J. *Molecular Electronic Structure Theory*, 1st ed.; John Wiley & Sons: New York, 2000; pp 336–426.

(2) Dupuis, M.; Rys, J.; King, H. F. Evaluation of Molecular Integrals Over Gaussian Basis Functions. *J. Chem. Phys.* **1976**, *65*, 111–116.

(3) Rys, J.; Dupuis, M.; King, H. F. Computation of Electron Repulsion Integrals Using the Rys Quadrature Method. *J. Comp. Chem.* **1983**, *4*, 154–157.

(4) Dupuis, M.; Marquez, A. The Rys Quadrature Revisited: A Novel Formulation for the Efficient Computation of Electron Repulsion Integrals Over Gaussian Functions. *J. Chem. Phys.* **2001**, *114*, 2067–2078.

(5) McMurchie, L. E.; Davidson, E. R. One- and Two-Electron Integrals Over Cartesian Gaussian Functions. *J. Comput. Phys.* **1978**, *26*, 218–231.

(6) Obara, S.; Saika, A. Efficient Recursive Computation of Molecular Integrals Over Cartesian Gaussian Functions. *J. Chem. Phys.* **1986**, *84*, 3963–3974.

(7) Obara, S.; Saika, A. General Recurrence Formulas for Molecular Integrals Over Cartesian Gaussian Functions. *J. Chem. Phys.* **1988**, *89*, 1540–1559.

(8) Head-Gordon, M.; Pople, J. A. A Method for Two-Electron Gaussian Integral and Integral Derivative Evaluation Using Recurrence Relations. *J. Chem. Phys.* **1988**, *89*, 5777–5786.

(9) Gill, P. M. W.; Head-Gordon, M.; Pople, J. A. Efficient Computation of Two-Electron-Repulsion Integrals and their nth-Order Derivatives Using Contracted Gaussian Basis Sets. *J. Phys. Chem.* **1990**, *94*, 5564–5572.

(10) Hamilton, T. P.; Schaefer, H. F. New Variations in Two-Electron Integral Evaluation in the Context of Direct SCF Procedures. *Chem. Phys.* **1991**, *150*, 163–171.

(11) Lindh, R.; Ryu, U.; Liu, B. The Reduced Multiplication Scheme of the Rys Quadrature and New Recurrence Relations for Auxiliary Function Based Two-Electron Integral Evaluation. *J. Chem. Phys.* **1991**, *95*, 5889–5897.

(12) Gill, P. M. W.; Pople, J. A. The Prism Algorithm for Two-Electron Integrals. *Int. J. Quant. Chem.* **1991**, *40*, 753–772.

(13) Ishida, K. General Formula Evaluation of the Electron-Repulsion Integrals and the First and Second Integral Derivatives Over Gaussian-Type Orbitals. *J. Chem. Phys.* **1991**, *95*, 5198–5205.

(14) Ishida, K. ACE Algorithm for the Rapid Evaluation of the Electron-Repulsion Integral Over Gaussian-Type Orbitals. *Int. J. Quant. Chem.* **1996**, *59*, 209–218.

(15) Ishida, K. Rigorous Formula for the Fast Calculation of the Electron Repulsion Integral Over the Solid Harmonic Gaussian-Type Orbitals. *J. Chem. Phys.* **1998**, *109*, 881–890.

(16) Kenny, J. P.; Janssen, C. L.; Valeev, E. F.; Windus, T. L. Components for Integral Evaluation in Quantum Chemistry. *J. Comp. Chem.* **2007**, *29*, 562–577.

(17) Flocke, N.; Lotrich, V. Efficient Electronic Integrals and Their Generalized Derivatives for Object Oriented Implementations of Electronic Structure Calculations. *J. Comp. Chem.* **2008**, *29*, 2722–2736.

(18) Gill, P. M. W.; Head-Gordon, M.; Pople, J. A. An Efficient Algorithm for the Generation

of Two-Electron Repulsion Integrals Over Gaussian Basis Functions. *Int. J. Quant. Chem.* **2009**, *36*, 269–280.

(19) Reine, S.; Helgaker, T.; Lindh, R. Multi-Electron Integrals. *WIREs Comput. Mol. Sci.* **2011**, *2*, 290–303.

(20) Pritchard, B. P.; Chow, E. Horizontal Vectorization of Electron Repulsion Integrals. *J. Comp. Chem.* **2016**, *37*, 2537–2546.

(21) Whitten, J. L. Coulombic Potential Energy Integrals and Approximations. *J. Chem. Phys.* **1973**, *58*, 4496–4501.

(22) Dunlap, B. I.; Connolly, J. W. D.; Sabin, J. R. On Some Approximations in Applications of Xα Theory. *J. Chem. Phys.* **1979**, *71*, 3396–3402.

(23) Feyereisen, M.; Fitzgerald, G.; Komornicki, A. Use of Approximate Integrals in Ab Initio Theory. An Application in MP2 Energy Calculations. *Chem. Phys. Lett.* **1993**, *208*, 359–363.

(24) Vahtras, O.; Almlöf, J.; Feyereisen, M. W. Integral Approximations for LCAO-SCF Calculations. *Chem. Phys. Lett.* **1993**, *213*, 514–518.

(25) Rendell, A. P.; Lee, T. J. Coupled-Cluster Theory Employing Approximate Integrals: An Approach to Avoid the Input/Output and Storage Bottlenecks. *J. Chem. Phys.* **1994**, *101*, 400–408.

(26) Weigend, F. A Fully Direct RI-HF Algorithm: Implementation, Optimised Auxiliary Basis Sets, Demonstration of Accuracy and Efficiency. *Phys. Chem. Chem. Phys.* **2002**, *4*, 4285–4291.

(27) Sodt, A.; Subotnik, J. E.; Head-Gordon, M. Linear Scaling Density Fitting. *J. Chem. Phys.* **2006**, *125*, 194109–194118.

(28) Werner, H.-J.; Manby, F. R.; Knowles, P. J. Fast Linear Scaling Second-Order Møller–Plesset Perturbation Theory (MP2) Using Local and Density Fitting Approximations. *J. Chem. Phys.* **2003**, *118*, 8149–8160.

(29) Schütz, M.; Manby, F. R. Linear Scaling Local Coupled Cluster Theory with Density Fitting. Part I: 4-External Integrals. *Phys. Chem. Chem. Phys.* **2003**, *5*, 3349–3358.

(30) Werner, H.-J.; Schütz, M. An Efficient Local Coupled Cluster Method for Accurate Thermochemistry of Large Systems. *J. Chem. Phys.* **2011**, *135*, 144116–144131.

(31) Beebe, N. H. F.; Linderberg, J. Simplifications in the Generation and Transformation of Two-Electron Integrals in Molecular Calculations. *Int. J. Quant. Chem.* **1977**, *12*, 683–705.

(32) Roeggen, I.; Wisloff-Nilssen, E. On the Beebe-Linderberg Two-Electron Integral Approximation. *Chem. Phys. Lett.* **1986**, *132*, 154–160.

(33) Koch, H.; de Meras, A. S.; Pedersen, T. B. Reduced Scaling in Electronic Structure Calculations Using Cholesky Decompositions. *J. Chem. Phys.* **2003**, *118*, 9481–9484.

(34) Aquilante, F.; Pedersen, T. B.; Lindh, R. Low-Cost Evaluation of the Exchange Fock Matrix from Cholesky and Density Fitting Representations of the Electron Repulsion Integrals. *J. Chem. Phys.* **2007**, *126*, 194106–194117.

(35) DePrince, A. E.; Sherrill, C. D. Accuracy and Efficiency of Coupled-Cluster Theory Using Density Fitting/Cholesky Decomposition, Frozen Natural Orbitals, and a T1-Transformed Hamiltonian. *J. Chem. Theory Comput.* **2013**, *9*, 2687–2696.

(36) Bozkaya, U. Orbital-Optimized MP3 and MP2.5 with Density-Fitting and Cholesky Decomposition Approximations. *J. Chem. Theory Comput.* **2016**, *12*, 1179–1188.

(37) Bozkaya, U. Orbital-Optimized Linearized Coupled-Cluster Doubles with Density-Fitting and Cholesky Decomposition Approximations: An Efficient Implementation. *Phys. Chem. Chem. Phys.* **2016**, *18*, 11362–11373.

(38) Bozkaya, U. Efficient Implementation of the Second-Order Quasidegenerate Perturbation Theory with Density-Fitting and Cholesky Decomposition Approximations: Is It Possible To Use Hartree–Fock Orbitals for a Multiconfigurational Perturbation Theory? *J. Chem. Theory Comput.* **2019**, *15*, 4415–4429.

(39) Weigend, F.; Häser, M. RI-MP2: First Derivatives and Global Consistency. *Theor. Chem. Acc.* **1997**, *97*, 331–340.

(40) Hättig, C.; Hellweg, A.; Köhn, A. Distributed Memory Parallel Implementation of Energies and Gradients for Second-Order Møller–Plesset Perturbation Theory with the Resolution-of-the-Identity Approximation. *Phys. Chem. Chem. Phys.* **2006**, *8*, 1159–1169.

(41) Distasio, R. A.; Steele, R. P.; Rhee, Y. M.; Shao, Y.; Head-Gordon, M. An Improved Algorithm for Analytical Gradient Evaluation in Resolution-of-the-Identity Second-Order Møller-Plesset Perturbation Theory: Application to Alanine Tetrapeptide Conformational Analysis. *J. Comp. Chem.* **2007**, *28*, 839–856.

(42) Bozkaya, U. Derivation of General Analytic Gradient Expressions for Density-Fitted Post-Hartree-Fock Methods: An Efficient Implementation for the Density-Fitted Second-Order Møller–Plesset Perturbation Theory. *J. Chem. Phys.* **2014**, *141*, 124108–124122.

(43) Hättig, C. Geometry Optimizations with the Coupled-Cluster Model CC2 Using the Resolution-of-the-Identity Approximation. *J. Chem. Phys.* **2003**, *118*, 7751.

(44) Köhn, A.; Hättig, C. Analytic Gradients for Excited States in the Coupled-Cluster

Model CC2 Employing the Resolution-of-the-Identity Approximation. *J. Chem. Phys.* **2003**, *119*, 5021.

(45) Schütz, M.; Werner, H.-J.; Lindh, R.; Manby, F. R. Analytical Energy Gradients for Local Second-Order Møller–Plesset Perturbation Theory Using Density Fitting Approximations. *J. Chem. Phys.* **2004**, *121*, 737.

(46) Györffy, W.; Shiozaki, T.; Knizia, G.; Werner, H.-J. Analytical Energy Gradients for Second-Order Multireference Perturbation Theory Using Density Fitting. *J. Chem. Phys.* **2013**, *138*, 104104.

(47) Bozkaya, U. Analytic Energy Gradients and Spin Multiplicities for Orbital-Optimized Second-Order Perturbation Theory with Density-Fitting Approximation: An Efficient Implementation. *J. Chem. Theory Comput.* **2014**, *10*, 4389–4399.

(48) Bozkaya, U. Analytic Energy Gradients for Orbital-Optimized MP3 and MP2.5 with the Density-Fitting Approximation: An Efficient Implementation. *J. Comp. Chem.* **2018**, *39*, 351–360.

(49) Ledermüller, L.; Schütz, M. *J. Chem. Phys.* **2014**, *140*, 164113.

(50) Bozkaya, U.; Sherrill, C. D. Analytic Energy Gradients for the Coupled-Cluster Singles and Doubles Method with the Density-Fitting Approximation. *J. Chem. Phys.* **2016**, *144*, 174103–174117.

(51) Bozkaya, U.; Sherrill, C. D. Analytic Energy Gradients for the Coupled-Cluster Singles and Doubles with Perturbative Triples Method with the Density-Fitting Approximation. *J. Chem. Phys.* **2017**, *147*, 044104–044114.

(52) Ahlrichs, R. Efficient Evaluation of Three-Center Two-Electron Integrals Over Gaussian Functions. *Phys. Chem. Chem. Phys.* **2004**, *6*, 5119–5121.

(53) Samu, G.; Kállay, M. Efficient Evaluation of Three-center Coulomb Integrals. *J. Chem. Phys.* **2017**, *146*, 204101.

(54) Samu, G.; Kállay, M. Efficient Evaluation of the Geometrical First Derivatives of Three-Center Coulomb Integrals. *J. Chem. Phys.* **2018**, *149*, 124101.

(55) E. F. Valeev, Libint: A Library For the Evaluation of Molecular Integrals of Many-Body Operators Over Gaussian Functions, http://libint.valeyev.net/. Last retrieved 13 August 2020.

(56) Valeev, E. F.; Janssen, C. L. Second-order Møller–Plesset theory with Linear R12 terms (MP2-R12) Revisited: Auxiliary Basis Set Method and Massively Parallel Implementation. *J. Chem. Phys.* **2004**, *121*, 1214–1227.

(57) Smith, D. G. A.; Burns, L. A.; Simmonett, A. C.; Parrish, R. M.; Schieber, M. C.; Galvelis, R.; Kraus, P.; Kruse, H.; Remigio, R. D.; Alenaizan, A.; James, A. M.; Lehtola, S.; Misiewicz, J. P.; Scheurer, M.; Shaw, R. A.; Schriber, J. B.; Xie, Y.; Glick, Z. L.; Sirianni, D. A.; O'Brien, J. S.; Waldrop, J. M.; Kumar, A.; Hohenstein, E. G.; Pritchard, B. P.; Brooks, B. R.; Schaefer, H. F.; Sokolov, A. Y.; Patkowski, K.; DePrince, A. E.; Bozkaya, U.; King, R. A.; Evangelista, F. A.; Turney, J. M.; Crawford, T. D.; Sherrill, C. D. Psi4 1.4: Open-source software for high-throughput quantum chemistry. *J. Chem. Phys.* **2020**, *152*, 184108.

(58) Kállay, M.; Nagy, P. R.; Mester, D.; Rolik, Z.; Samu, G.; Csontos, J.; Csóka, J.; Szabó, P. B.; Gyevi-Nagy, L.; Hégely, B.; Ladjánszki, I.; Szegedy, L.; Ladóczki, B.; Petrov, K.; Farkas, M.; Mezei, P. D.; Ganyecz, Á. The MRCC Program System: Accurate Quantum Chemistry From Water to Proteins. *J. Chem. Phys.* **2020**, *152*, 074107.

(59) Saunders, V. R. *Computational Techniques in Quantum Chemistry and Molecular Physics*; Springer Netherlands, 1975; pp 347–424.

(60) Helgaker, T.; Taylor, P. R. On the Evaluation of Derivatives of Gaussian Integrals. *Theor. Chem. Acc.* **1992**, *83*, 177–183.

(61) Komornicki, A.; Ishida, K.; Morokuma, K.; Ditchfield, R.; Conrad, M. Efficient Determination and Characterization of Transition States Using Ab-Initio Methods. *Chem. Phys. Lett.* **1977**, *45*, 595–602.

(62) Bozkaya, U. Orbital-Optimized Second-Order Perturbation Theory with Density-Fitting and Cholesky Decomposition Approximations: An Efficient Implementation. *J. Chem. Theory Comput.* **2014**, *10*, 2371–2378.

(63) Bozkaya, U. A Noniterative Asymmetric Triple Excitation Correction for the Density-Fitted Coupled-Cluster Singles and Doubles Method: Preliminary Applications. *J. Chem. Phys.* **2016**, *144*, 144108–144120.

(64) Dunning, T. H. Gaussian Basis Sets for Use in Correlated Molecular Calculations. I. The Atoms Boron Through Neon and Hydrogen. *J. Chem. Phys.* **1989**, *90*, 1007–1023.

(65) Woon, D. E.; Dunning, T. H. Gaussian Basis Sets for Use in Correlated Molecular Calculations. V. Core-Valence Basis Sets for Boron Through Neon. *J. Chem. Phys.* **1995**, *103*, 4572–4585.

(66) Weigend, F.; Köhn, A.; Hättig, C. Efficient Use of the Correlation Consistent Basis Sets in Resolution of the Identity MP2 Calculations. *J. Chem. Phys.* **2002**, *116*, 3175–3183.

Table 1: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pVTZ and cc-pVTZ-RI basis sets, wall time (in seconds) for the metric and 3-index integrals for DF-MP2 energy computations. All computations were performed on a single node (up to 8 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric (1 core)**	**Metric (8 cores)**	**3-Index (1 core)**	**3-Index (8 cores)**
CH_4	86	201	0.02	0.03	0.26	0.12
C_2H_6	144	342	0.07	0.08	1.24	0.41
C_3H_8	202	483	0.15	0.13	3.44	1.00
C_4H_9	260	624	0.26	0.21	7.32	2.16
C_5H_{10}	318	765	0.44	0.32	13.36	3.98
C_6H_{14}	376	906	0.64	0.45	22.05	6.51
C_7H_{16}	434	1047	0.91	0.60	33.73	9.65
C_8H_{18}	492	1188	1.24	0.78	48.98	14.06
C_9H_{20}	550	1329	1.67	0.99	68.43	19.60
$C_{10}H_{22}$	608	1470	2.15	1.22	92.84	27.13

Table 2: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pVQZ and cc-pVQZ-RI basis sets, wall time (in seconds) for the metric and 3-index integrals for DF-MP2 energy computations. All computations were performed on a single node (up to 8 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric (1 core)**	**Metric (8 cores)**	**3-Index (1 core)**	**3-Index (8 cores)**
CH_4	175	352	0.00	0.11	0.03	0.50
C_2H_6	290	594	0.28	0.23	8.67	2.33
C_3H_8	405	836	0.61	0.45	23.81	6.11
C_4H_9	520	1078	1.09	0.75	50.71	12.60
C_5H_{10}	635	1320	2.48	2.13	92.45	22.99
C_6H_{14}	750	1562	2.68	1.62	152.45	39.18
C_7H_{16}	865	1804	3.90	2.22	233.55	60.32
C_8H_{18}	980	2046	5.53	2.98	339.10	88.72
C_9H_{20}	1095	2288	7.49	3.76	467.37	119.32
$C_{10}H_{22}$	1210	2530	10.09	4.96	630.15	166.49

Table 3: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pV5Z and cc-pV5Z-RI basis sets, wall time (in seconds) for the metric and 3-index integrals for DF-MP2 energy computations. All computations were performed on a single node (up to 8 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric (1 core)**	**Metric (8 cores)**	**3-Index (1 core)**	**3-Index (8 cores)**
CH_4	311	557	0.47	0.31	12.34	2.90
C_2H_6	512	932	0.97	0.70	57.32	12.80
C_3H_8	713	1307	2.03	1.45	157.29	38.98
C_4H_9	914	1682	3.63	2.30	329.34	75.08
C_5H_{10}	1115	2057	6.19	3.56	601.72	132.32
C_6H_{14}	1316	2432	9.98	5.33	991.67	211.05
C_7H_{16}	1517	2807	15.25	8.05	1523.50	322.42

Table 4: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pVTZ and cc-pVTZ-RI basis sets, wall time (in seconds) for the metric and 3-index terms in DF-MP2 analytic gradient computations. All computations were performed on a single node (2 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric**	**3-Index**
CH_4	86	201	0.05	0.89
C_2H_6	144	342	0.17	4.55
C_3H_8	202	483	0.31	12.73
C_4H_9	260	624	0.49	27.68
C_5H_{10}	318	765	0.73	50.18

Table 5: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pVQZ and cc-pVQZ-RI basis sets, wall time (in seconds) for the metric and 3-index terms in DF-MP2 analytic gradient computations. All computations were performed on a single node (2 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric**	**3-Index**
CH_4	175	352	0.19	8.59
C_2H_6	290	594	0.52	32.78
C_3H_8	405	836	1.03	92.24
C_4H_9	520	1078	1.71	196.61
C_5H_{10}	635	1320	2.61	358.90

Table 6: The number of primary and auxiliary basis functions (N_{bf} and N_{aux}) in the cc-pV5Z and cc-pV5Z-RI basis sets, wall time (in seconds) for the metric and 3-index terms in DF-MP2 analytic gradient computations. All computations were performed on a single node (2 cores) Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10 GHz computer (memory $\sim$ 120 GB).

Molecule	$\boldsymbol{N_{bf}}$	$\boldsymbol{N_{aux}}$	**Metric**	**3-Index**
CH_4	311	557	0.54	45.85
C_2H_6	512	932	1.57	220.37
C_3H_8	713	1307	3.21	607.48
C_4H_9	914	1682	5.31	1257.25
C_5H_{10}	1115	2057	8.03	2285.34

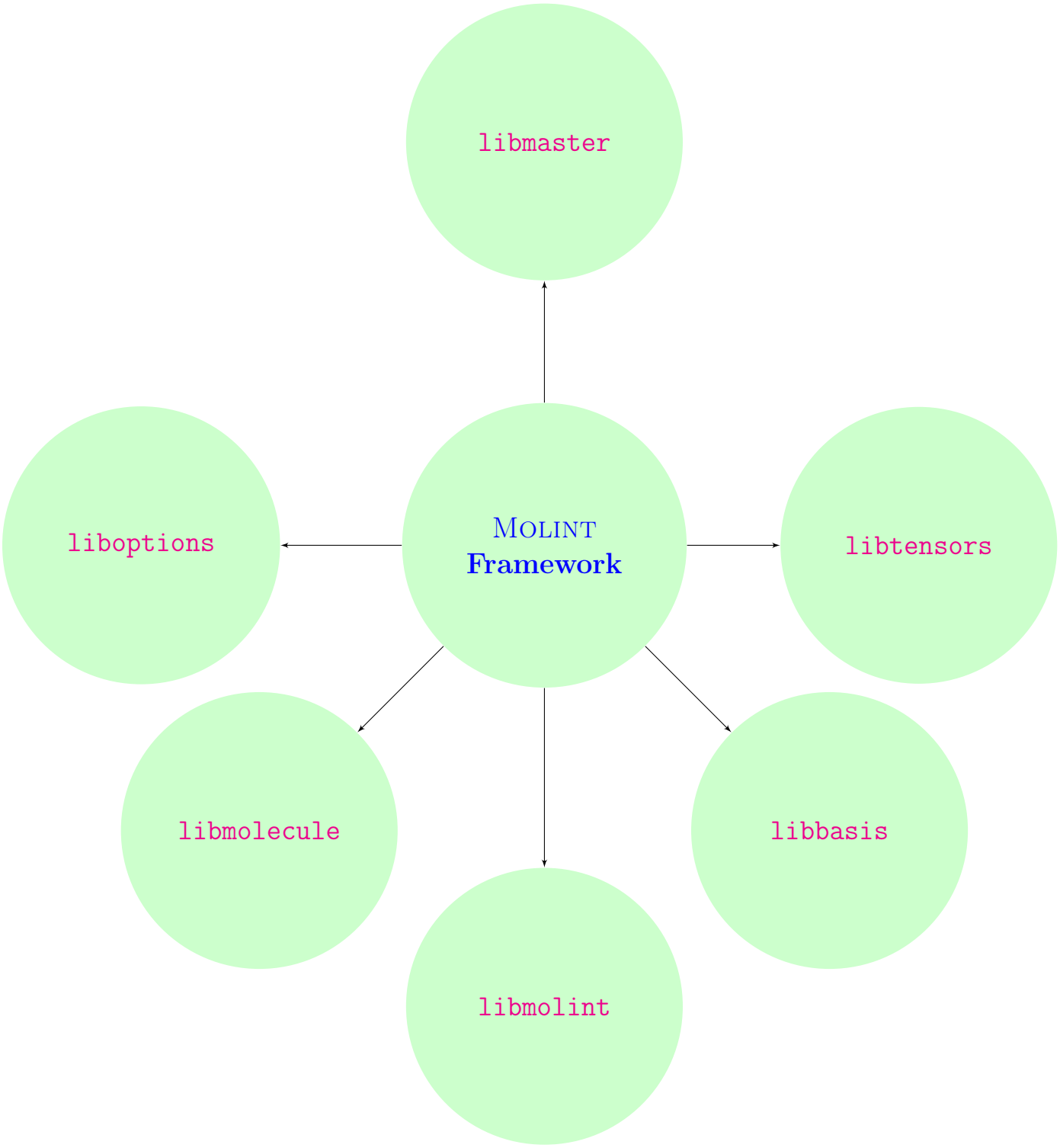


Figure 1: Components of the MOLINT framework.

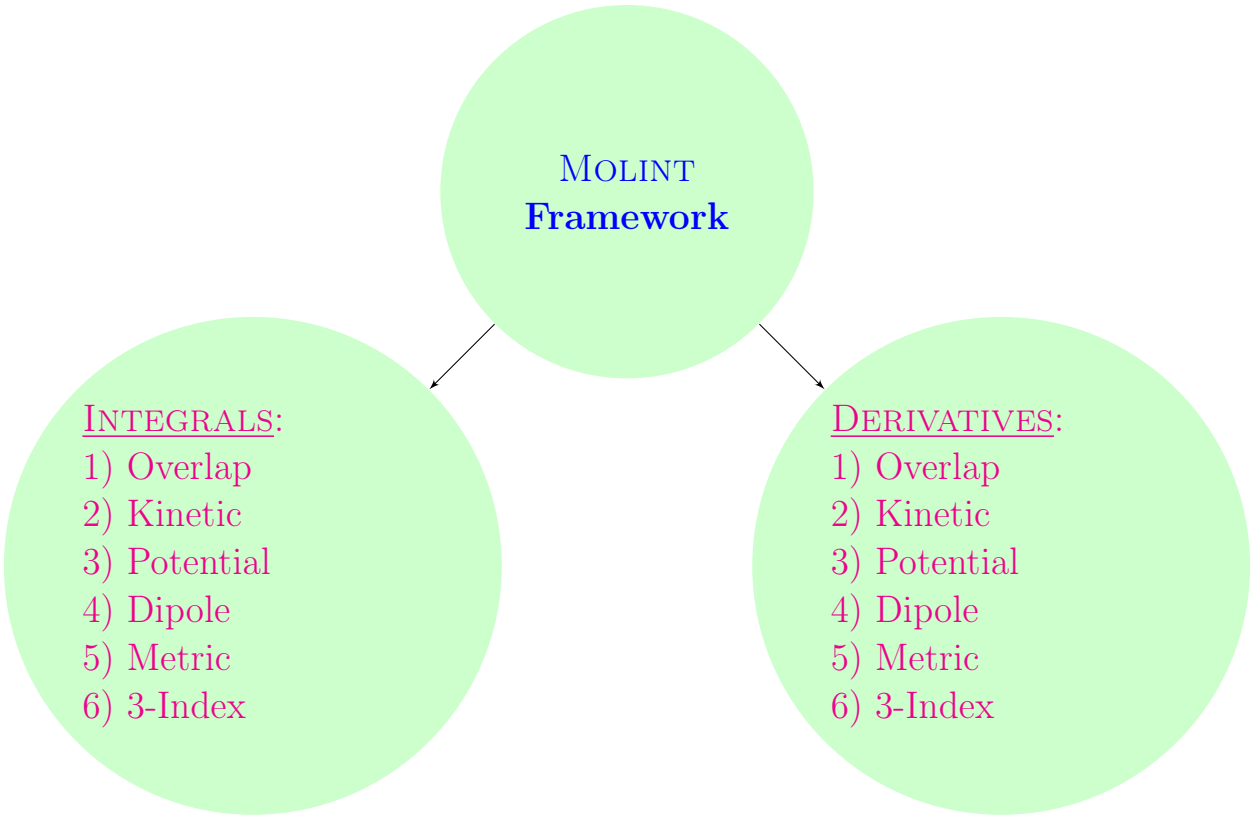
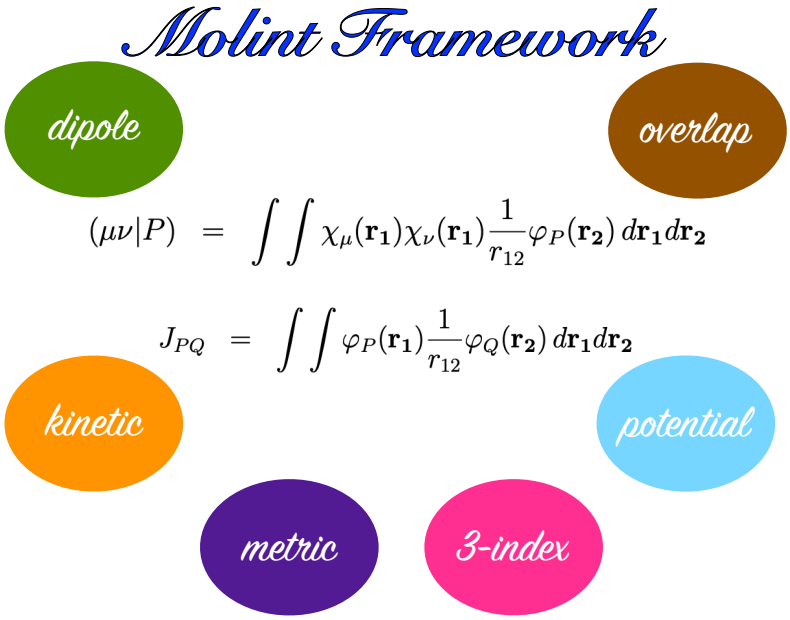


Figure 2: Capabilities of the MOLINT framework.

TOC Graphic


Molint Framework
dipole
overlap
$(\mu\nu|P) = \int\int \chi_\mu(\mathbf{r_1})\chi_\nu(\mathbf{r_1})\frac{1}{r_{12}}\varphi_P(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}$
$J_{PQ} = \int\int \varphi_P(\mathbf{r_1})\frac{1}{r_{12}}\varphi_Q(\mathbf{r_2})\, d\mathbf{r_1} d\mathbf{r_2}$
kinetic
potential
metric
3-index