Efficient evaluation of three-center Coulomb-integrals and their geometrical first derivatives

Ph.D. dissertation

Gyula Samu

Advisor: Prof. Mihály Kállay

Budapest University of Technology and Economics
Department of Physical Chemistry and Materials Science
MTA-BME Lendület Quantum Chemistry Research Group

George Oláh PhD School
Head of the School: Prof. László Nyulászi

2018

Contents

Acknowledgements

I would like to express my gratitude to my supervisor, Mihály Kállay. This work could not have been completed without his guidance and support. I am very grateful that he decided to accept me as his student despite my little experience in theoretical chemistry. The help received from my colleagues, especially Péter Nagy and Bence Hégely, is also greatly appreciated.

I am also happy to thank my parents, who continued to support me during my education, and without whom I could not pursue my doctoral studies.

Abbreviations

AF	auxiliary function
AO	atomic orbital
CPU	central processing unit
DF	density fitting
ED	extended domain
ERI	electron repulsion integral
ETR	electron transfer relation
FLOP	floating point operation
GHP	Gill–Head-Gordon–Pople
HF	Hartree–Fock
HGP	Head-Gordon–Pople
HRR	horizontal recurrence relation
LFD	local fitting domain
MD	McMurchie–Davidson
MO	molecular orbital
OS	Obara–Saika
PAO	projected atomic orbital
PCD	PAO center domain
RAM	random access memory
SCF	self consistent field
VRR	vertical recurrence relation

Symbols

$\boldsymbol{s}$,$\boldsymbol{p}$,$\boldsymbol{d}$,$\boldsymbol{f}$,$\boldsymbol{g}$,$\boldsymbol{h}$,$\boldsymbol{i}$,$\boldsymbol{k}$,$\boldsymbol{l}$	function with an angular momentum of value 0,1,2,3,4,5,6,7,8
$A, B, C \ldots$	indices of points on which the functions are centered
$\chi_A, \chi_B, \chi_C \cdots$	AO centered on $A, B, C \ldots$
G_{IJK}	Cartesian Gaussians
I, J, K	x, y, z components of Cartesian Gaussians
$\tilde{H}_{\tilde{I}\tilde{J}\tilde{K}}$	Hermite Gaussians
$\tilde{I}, \tilde{J}, \tilde{K}$	x, y, z components of Hermite Gaussians
$\bar{H}_{\bar{I}\bar{J}\bar{K}}$	unscaled Hermite Gaussians
$\bar{I}, \bar{J}, \bar{K}$	x, y, z components of unscaled Hermite Gaussians
$G_{L,m}$	Solid harmonic Gaussians
L	total angular momentum of a function
m	Solid harmonic component
$\boldsymbol{L}$	angular momentum vector of a function
$a, b, c, \ldots$	primitive Gaussian exponents
$\mathbf{r_i}$	absolute position vector of the **i**th electron
$\mathbf{r_A}$	position vector of an electron relative to center A
x_A, y_A, z_A	Cartesian components of the $\mathbf{r_A}$ vector
r_A	magnitude of the $\mathbf{r_A}$ vector
$\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots$	position vector of the function centers
$\mathbf{R}_{\text{AB}}$	vector pointing from $\mathbf{B}$ to $\mathbf{A}$
$X_{\text{AB}}, Y_{\text{AB}}, Z_{\text{AB}}$	Cartesian components of the $\mathbf{R}_{\text{AB}}$ vector
R_{AB}	magnitude of the $\mathbf{R}_{\text{AB}}$ vector
Γ	coefficient of the solid harmonic transformation
d	coefficient of the primitive contraction
$\psi, \tilde{\psi}$	localized MO, pseudocanonicalized localized MO
$\phi, \tilde{\phi}$	PAO, pseudocanonicalized PAO
$\bar{\psi}$, $\bar{\phi}$	occupied and virtual LNOs
Λ^x_{ab}	x-dependent part of the overlap of $\boldsymbol{s}$-type Gaussian functions with exponents a and b

Introduction

Quantum chemistry continues to be an emerging field of science, which serves as a tool of theoretical validation for hypotheses based on experimental results, and also as a group of methods for the prediction of chemical phenomena. A significant portion of the efforts in quantum chemical method development aims to reduce the computational costs of the simulations, allowing the modeling of increasingly more complex systems. As a result of the development of both the quantum chemical algorithms and commercial computer hardware, calculations for systems containing hundreds of atoms are now in the reach of highly sophisticated theoretical methods [1–10].

In quantum mechanical calculations the quantities of interest are computed as the eigenvalues of operators, which, in practice, have a matrix representation over a finite basis set. The elements of these matrices are integrals over the functions in the basis. These integrals are usually further expanded in integrals over more elementary functions. The evaluation of such integrals has always been an important factor in the computational expense of quantum chemical calculations. As the simulations challenged larger problems, it became apparent that the necessary integrals are too numerous to be stored even on disk. This realization led to the advent of integral direct methods, in which the integrals are recalculated each time they need to be processed, e.g., in each cycle of a direct self-consistent field (SCF) procedure [11–13] or for the overlapping domains in a local correlation calculation [1]. Such approaches introduce a high level of redundancy into the integral computation, and the feasibility of these schemes depends on the efficiency of the integral evaluation process.

It has been noticed early [14] that multicenter integrals can be efficiently approximated by expanding the products of functions in an auxiliary basis. For four-center electron repulsion integrals (ERIs, also called Coulomb-integrals), central to most quantum chemical methods, the most useful fitting procedure [15] results in the expansion of these ERIs in products of three- and two-center Coulomb-integrals. The advantage is twofold: the number of these ERIs is smaller than those of the four-center ones, and their evaluation is considerably simpler. Hence, the method called density fitting (DF) became an established approach to reduce the computational cost of a wide variety of methods [13, 16–30]. Nevertheless, the computation of three-center ERIs is still a time-consuming step of the simulations. While the calculation of four-center ERIs has a diverse literature, the efficient evaluation of three-center ones received significantly less attention. The same is true for the rapid calculation of the first geometrical derivatives of these ERIs, required for gradient computations with methods that apply DF.

The aim of my work was to adapt the well-known approaches and to develop new ones for the computation of these quantities. I estimated the number of floating point operations (FLOPs) required by each algorithm for the evaluation of the ERIs and ERI derivatives of a simple model system. Based on these preliminary results I utilized automated code generation techniques to efficiently implement the most promising schemes and to determine their relative runtime performances. I also compared various methods for the prescreening of these quantities and analyzed how the different algorithmic considerations affect the use of the cache memory of the central processing unit (CPU). These optimization steps resulted in an efficient implementation of three-center ERI and ERI derivative computation. The swift integral evaluation allowed our group to apply our local correlation methods to systems of groundbreaking size [1, 2].

Chapter 1 presents the theoretical background, that is, a general discussion of four-center ERIs and their derivatives, as well as a discussion of the most widely used algorithms to evaluate them. During the introduction of these methods certain techniques to derive recurrence relations, which will be applied in later chapters, will also be presented. Chapter 2 details the process of the optimization of three-center ERI evaluation. This will include the adaptation of the established methods to the three-center case, considering new combinations of the algorithms and also technical details of the computation specific to the present case, estimation of the FLOP requirements, details of the implementation, and the discussion of the results of performance tests and benchmark calculations. Chapter 3, which presents the optimization of the computation of geometrical first-order three-center ERI derivatives, has a similar structure. Here a new type of ERIs will also be considered with the aim of utilizing the advantageous properties of previous approaches for derivative calculations. Finally, Chapter 4 discusses the role of the evaluation of three-center ERIs in the local correlation schemes recently developed in our group [1, 2] and demonstrates the effect of the developments discussed in Chapter 2 by showcasing a few large-scale applications [2] of our Hartree–Fock, Møller–Plesset, and coupled-cluster implementations utilizing density fitting and local approximations.

Chapter 1

Theoretical background

Much of the work presented in this dissertation is based on the methods developed for the calculation of four-center ERIs (and their first derivatives), which we wish to approximate by DF. This chapter introduces the basic concepts and notations through the four-center integrals, as well as four widely used approaches to analytically evaluate them: the McMurchie–Davidson (MD) [31], Gill–Head-Gordon–Pople (GHP) [32], Obara–Saika (OS) [33], and Rys polynomial [34] methods. Historically the Rys method was the first to be developed out of these four schemes, followed by MD, OS, and GHP. However, the MD approach provides a relatively simple framework for the description and evaluation of the integrals and integral derivatives, from which the recursive equations of the other methods can be derived [35], and I will also utilize the concepts of MD in the later derivations. Integral prescreening, essential for the treatment of large systems, is also discussed, as well as the DF approximation of four-center ERIs. The end of the Chapter includes a short overview on the memory hierarchy of modern computers, which will be necessary for the interpretation of some of the results of Chapters 2 and 3.

It should be noted that the overview of methods for the evaluation of ERIs is not comprehensive. Methods not detailed in the present work include the original algorithm of Boys [36], in which ERIs are expanded in a set of auxiliary integrals related by differentiation; the axis rotation method of Hehre and Pople [37], efficient for integrals involving only $\boldsymbol{s}$ and $\boldsymbol{p}$ type functions; the accompanying coordinate expansion and transferred recurrence relation method [38, 39], well suited for the computation of highly contracted ERIs. There also exist approaches that avoid the exact computation of Coulomb-integrals in certain cases and are also not discussed here. The so-called J-engine [40, 41] and the related schemes [42] exploit the structure of the Coulomb term in a direct SCF calculation and only partially calculate the ERIs or ERI derivatives, and, by reverse operations, transform the density matrix into a form appropriate for the contraction with intermediate (differentiated) ERIs rather than with those over the atomic orbitals (AOs). Also, for charge distributions with sufficiently small overlap, the exact calculation of ERIs and their derivatives is not necessary. Here significant speedups can be achieved by the application of approximations based on multipole or asymptotic expansions [43–45].

1.1 Four-center Coulomb-integrals and their geometrical first derivatives

We will be concerned with the evaluation of ERIs over AOs, which are used to expand integrals over molecular orbitals (MOs). The four-center ERIs over AOs are defined as

$$(\chi_A\chi_B|\chi_C\chi_D) = \int\int \frac{\chi_A(\mathbf{r_1})\chi_B(\mathbf{r_1})\chi_C(\mathbf{r_2})\chi_D(\mathbf{r_2})}{|\mathbf{r_1}-\mathbf{r_2}|}\mathrm{d}\mathbf{r_1}\,\mathrm{d}\mathbf{r_2}\ , \tag{1.1.1}$$

where χ_A denotes the AO centered on nucleus A and $\mathbf{r_i}$ is the position vector of the $\mathbf{i}$th electron, and the integrations are over the full space for both electrons. The product $\chi_A\chi_B$ defines a probability distribution for the first electron, which can also be viewed as a charge distribution. Therefore, Eq. (1.1.1) gives the electrostatic repulsion between the two electrons in atomic units.

ERIs over AOs are computed as a linear combination of integrals over primitive functions. These are most commonly chosen to be unnormalized Cartesian Gaussians, first proposed by Boys [36], defined as

$$G_{IJK}(\mathbf{r_1}, a, \mathbf{A}) = x_\mathrm{A}^I y_\mathrm{A}^J z_\mathrm{A}^K \exp(-ar_\mathrm{A}^2)\ , \tag{1.1.2}$$

where $\mathbf{A}$ denotes the position on which the function is centered, a is a constant Gaussian exponent, and r_A is the magnitude of the vector $\mathbf{r_A} = \mathbf{r} - \mathbf{A}$ with x_A being the x component of $\mathbf{r_A}$. $L = I + J + K$ will be called the angular momentum of G_{IJK}, and the vector $\boldsymbol{L} = (I, J, K)$ will be referred to as its angular momentum vector. I, J, and K are zero or positive integers. Functions with the same center, exponent, and angular momentum constitute a shell with $(L + 1)(L + 2)/2$ components. The primitive Gaussians are separable in the three Cartesian directions, that is, $G_{IJK} = G_I G_J G_K$, where, for instance, $G_I = x_\mathrm{A}^I \exp(-ax_\mathrm{A}^2)$. They also obey the recurrence relations (given here for the x direction only)

$$x_\mathrm{A} G_I = G_{I+1} \tag{1.1.3}$$

and

$$\frac{\partial G_I}{\partial A_x} = 2aG_{I+1} - IG_{I-1} \tag{1.1.4}$$

where A_x is the x component of $\mathbf{A}$. The main reason why Gaussian functions are preferred for integral evaluation is the Gaussian product rule [36], which states that the product of two Gaussians is another Gaussian multiplied by a constant, that is,

$$\exp(-ax_\mathrm{A}^2)\exp(-bx_\mathrm{B}^2) = \kappa_{ab}^x \exp(-px_\mathrm{P}^2) \equiv \Lambda_{ab}^x \tag{1.1.5}$$

with

$$\kappa_{ab}^{x} = \exp(-\mu X_{\text{AB}}^2) \qquad p = a + b$$

$$\mu = \frac{ab}{p} \qquad \mathbf{P} = \frac{a\mathbf{A} + b\mathbf{B}}{p} \ ,$$

where X_{AB} is the x component of the vector $\mathbf{R}_{\text{AB}} = \mathbf{A} - \mathbf{B}$.

Cartesian Gaussian ERIs have to undergo two linear transformations for us to get the integrals over AOs. The Cartesian monomials multiplying the exponential part are combined into real solid harmonics, resulting in solid harmonic Gaussians defined as

$$G_{Lm} = \sum_{I+J+K=L} \Gamma_{\ IJK}^{\ Lm} \, G_{IJK} \ . \qquad (1.1.6)$$

A shell of solid harmonic Gaussians consists of functions with $0 \leq |m| \leq L$, having $2L + 1$ components. The $\Gamma_{\ IJK}^{\ Lm}$ coefficients in Eq. (1.1.6) only depend on the angular momentum vector and the value of L and m [35]. In the second transformation, referred to as primitive contraction, we combine the exponential part of the primitive functions. If Eq. (1.1.6) has already been applied, this is performed as

$$\chi_A(\mathbf{r}, \mathbf{A}, L, m) = \sum_a G_{Lm}(\mathbf{r}, a, \mathbf{A}) d_{a\chi_A} \ , \qquad (1.1.7)$$

where the contraction coefficients $d_{a\chi_A}$ also include the norm of the solid harmonic Gaussian function and are the same for a given shell. Note that the primitive contraction can be executed before the solid harmonic transformation as well. At the evaluation of ERIs and their derivatives both of these operations reduce the number of intermediates that have to be processed by further computational steps.

It is also possible to apply Hermite Gaussian primitive functions, which will be defined as [46]

$$\tilde{H}_{\tilde{I}\tilde{J}\tilde{K}}(\mathbf{r}, a, \mathbf{A}) = \frac{\partial^{\tilde{L}} \exp(-ar_{\text{A}}^2)}{(2a)^{\tilde{L}} \partial A_x^{\tilde{I}} \partial A_y^{\tilde{J}} \partial A_z^{\tilde{K}}} \qquad (1.1.8)$$

and tildes over the angular momentum, the angular momentum vector, and its components will be applied to distinguish them from the corresponding Cartesian quantities. This distinction will not be applied when the angular momentum is zero. The Hermite Gaussians are also separable in the Cartesian directions, that is, $\tilde{H}_{\tilde{I}\tilde{J}\tilde{K}} = \tilde{H}_{\tilde{I}} \tilde{H}_{\tilde{J}} \tilde{H}_{\tilde{K}}$, where $\tilde{H}_{\tilde{I}} = \partial^{\tilde{I}} \exp(-ax_{\text{A}}^2)/(2a\partial A_x)^{\tilde{I}}$. The recursions corresponding to Eqs. (1.1.3) and (1.1.4) are

$$x_{\text{A}} \tilde{H}_{\tilde{I}} = \tilde{H}_{\tilde{I}+\tilde{1}} + \frac{\tilde{I}}{2a} \tilde{H}_{\tilde{I}-\tilde{1}} \qquad (1.1.9)$$

and

$$\frac{\partial \tilde{H}_{\tilde{I}}}{\partial A_x} = 2a \tilde{H}_{\tilde{I}+\tilde{1}} \ , \qquad (1.1.10)$$

respectively. It can be proven [46] that the Hermite Gaussians can also be transformed into solid harmonic ones by the same coefficients that are used for the Cartesian Gaussians. In some cases, we will utilize the unscaled Hermite Gaussians defined as [35]

$$H_{\bar{I}\bar{J}\bar{K}}(\mathbf{r}, a, \mathbf{A}) = \frac{\partial^{\bar{L}} \exp(-ar_{\mathrm{A}}^2)}{\partial A_x^{\bar{I}} \partial A_y^{\bar{J}} \partial A_z^{\bar{K}}} \ , \tag{1.1.11}$$

which obey the recurrences

$$x_{\mathrm{A}} H_{\bar{I}} = \frac{1}{2a} H_{\bar{I}+\bar{1}} + \bar{I} H_{\bar{I}-\bar{1}} \tag{1.1.12}$$

and

$$\frac{\partial H_{\bar{I}}}{\partial A_x} = H_{\bar{I}+\bar{1}} \ . \tag{1.1.13}$$

We will also use the relation

$$x_{\mathrm{B}} = x_{\mathrm{A}} + X_{\mathrm{AB}} \ . \tag{1.1.14}$$

For brevity, the notation

$$(\boldsymbol{L}_a \boldsymbol{L}_b | \boldsymbol{L}_c \boldsymbol{L}_d) = (G_{I_a J_a K_a}[\mathbf{r_1}, a, \mathbf{A}] \ G_{I_b J_b K_b}[\mathbf{r_1}, b, \mathbf{B}] \mid G_{I_c J_c K_c}[\mathbf{r_2}, c, \mathbf{C}] \ G_{I_d J_d K_d}[\mathbf{r_2}, d, \mathbf{D}]) \tag{1.1.15}$$

will be used for primitive Cartesian Gaussian ERIs, and Hermite Gaussian ones will be referred to similarly. I_a, J_a, and K_a denote the x, y, and z components of the Gaussian with exponent a. The term class will refer to primitive or solid harmonic integrals with the same angular momenta, centers, and exponents, e.g., the primitive class $(\mathbf{11}|\mathbf{11})$ contains 81 primitive integrals. For contracted integrals, a class will mean a group of ERIs over the same contracted functions and with the same angular momenta and centers. A shell quartet will refer to all the four-center integrals belonging to the same centers and angular momenta. Shell triplet will bear the same meaning for three-center ERIs.

The primitive integral where the total angular momentum is zero is of central importance for the evaluation of ERIs. For both types of Gaussians, the value of this integral is given explicitly as [35]

$$(\mathbf{00}|\mathbf{00}) = \theta_{pq} \kappa_{ab} \kappa_{cd} F_0(\alpha R_{\mathrm{PQ}}^2) \ , \tag{1.1.16}$$

with

$$q = c + d \qquad \mathbf{Q} = \frac{c\mathbf{C} + d\mathbf{D}}{q}$$

$$\alpha = \frac{pq}{p+q} \qquad \theta_{pq} = \frac{2\pi^{5/2}}{pq\sqrt{p+q}}$$

$$\kappa_{ab} = \kappa_{ab}^x \kappa_{ab}^y \kappa_{ab}^z \ ,$$

R_{PQ} being the magnitude of the vector $\mathbf{R}_{\mathrm{PQ}}$, and F_n being the Boys function of order n, defined as

$$F_n(x) = \int_0^1 t^{2n} \exp(-xt^2)\mathrm{d}t \ . \tag{1.1.17}$$

Note that

$$\frac{\partial F_n(x)}{\partial x} = -F_{n+1}(x) \ . \tag{1.1.18}$$

Integrals $(\mathbf{00}|\mathbf{00})^{(n)}$ calculated similarly to Eq. (1.1.16) with the order of the Boys function not necessarily being zero serve as the starting values for the OS, MD, and GHP schemes for the evaluation of the primitive integrals.

When differentiating the ERIs with respect to a coordinate of a function center we invoke a consequence of the Leibniz integral rule, that is, if the integration limits do not depend on the variable of differentiation the differential operator can be moved inside the integral. Hence, differentiating Eq. (1.1.15) with respect to the x coordinate of an arbitrary center $\mathbf{S}$ we get

$$\begin{aligned}\frac{\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)}{\partial S_x} &= \Big(\frac{\partial \boldsymbol{L}_a}{\partial S_x}\boldsymbol{L}_b\Big|\boldsymbol{L}_c\boldsymbol{L}_d\Big) + \Big(\boldsymbol{L}_a\frac{\partial \boldsymbol{L}_b}{\partial S_x}\Big|\boldsymbol{L}_c\boldsymbol{L}_d\Big) \\ &+ \Big(\boldsymbol{L}_a\boldsymbol{L}_b\Big|\frac{\partial \boldsymbol{L}_c}{\partial S_x}\boldsymbol{L}_d\Big) + \Big(\boldsymbol{L}_a\boldsymbol{L}_b\Big|\boldsymbol{L}_c\frac{\partial \boldsymbol{L}_d}{\partial S_x}\Big) \ .\end{aligned} \tag{1.1.19}$$

The ERIs on the right-hand side of Eq. (1.1.19) are over differentiated functions which are defined by Eq. (1.1.4) or (1.1.10) depending on the Gaussian type. For example, the first term is evaluated as

$$\Big(\frac{\partial \boldsymbol{L}_a}{\partial S_x}\boldsymbol{L}_b\Big|\boldsymbol{L}_c\boldsymbol{L}_d\Big) = \Big\{2a([\boldsymbol{L}_a+\mathbf{1}_x]\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d) - I_a([\boldsymbol{L}_a-\mathbf{1}_x]\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)\Big\}\ \delta_{A_x,S_x} \tag{1.1.20}$$

using Cartesian Gaussians, and as

$$\Big(\frac{\partial \tilde{\boldsymbol{L}}_a}{\partial S_x}\tilde{\boldsymbol{L}}_b\Big|\tilde{\boldsymbol{L}}_c\tilde{\boldsymbol{L}}_d\Big) = 2a([\tilde{\boldsymbol{L}}_a+\tilde{\mathbf{1}}_x]\tilde{\boldsymbol{L}}_b|\tilde{\boldsymbol{L}}_c\tilde{\boldsymbol{L}}_d)\ \delta_{A_x,S_x} \tag{1.1.21}$$

utilizing Hermite Gaussians, where $\boldsymbol{L}_a+\mathbf{1}_x$ means that the x component of the angular momentum vector $\boldsymbol{L}_a$ has been increased by one, and δ is the Kronecker delta. Thus a straightforward approach to compute the differentiated integrals is to calculate the integrals required by Eq. (1.1.4) or (1.1.10), then to construct the derivatives. If the final ERI derivatives are required in the solid harmonic Gaussian basis, then, using Cartesian Gaussians, we have to apply the

$$\frac{\partial G_{Lm}(\mathbf{r},a,\mathbf{A})}{\partial A_x} = \sum_{I+J+K=L} C_{IJK}^{Lm}\frac{\partial G_{IJK}(\mathbf{r},a,\mathbf{A})}{\partial A_x} \tag{1.1.22}$$

equation to the integrals over differentiated Cartesian Gaussians given by Eq. (1.1.20). On the other hand, the differentiation and the solid harmonic transformation can be merged for Hermite Gaussians as

$$\frac{\partial G_{Lm}(\mathbf{r},a,\mathbf{A})}{\partial A_x} = \sum_{\tilde{I}+\tilde{J}+\tilde{K}=\tilde{L}} C^{\tilde{L}m}_{\tilde{I}\tilde{J}\tilde{K}} 2a\tilde{H}_{\tilde{I}+\tilde{1}\tilde{J}\tilde{K}}(\mathbf{r},a,\mathbf{A}) \ , \tag{1.1.23}$$

where $C^{\tilde{L}m}_{\tilde{I}\tilde{J}\tilde{K}} = C^{Lm}_{IJK}$ [46]. It is clear that the Hermite Gaussians are better suited for the computation of integral derivatives, however, as it will be demonstrated, the evaluation of the Hermite ERIs is more expensive.

The Coulomb-integrals are invariant to certain translations and rotations, which have been investigated by several authors [47–53]. This work will only be concerned with the translational invariance for reasons discussed later. This property means that, if every center is dislocated in the same direction with the same magnitude, the value of the integral does not change since we are integrating over the full space and the overlap of the functions remains the same. It can be proven from the Taylor series expansion of the translated ERI [50] that this can be expressed as

$$\Big(\frac{\partial \boldsymbol{L}_a}{\partial A_x}\boldsymbol{L}_b\Big|\boldsymbol{L}_c\boldsymbol{L}_d\Big) + \Big(\boldsymbol{L}_a\frac{\partial \boldsymbol{L}_b}{\partial B_x}\Big|\boldsymbol{L}_c\boldsymbol{L}_d\Big) + \Big(\boldsymbol{L}_a\boldsymbol{L}_b\Big|\frac{\partial \boldsymbol{L}_c}{\partial C_x}\boldsymbol{L}_d\Big) + \Big(\boldsymbol{L}_a\boldsymbol{L}_b\Big|\boldsymbol{L}_c\frac{\partial \boldsymbol{L}_d}{\partial D_x}\Big) = 0 \ . \tag{1.1.24}$$

Eq. (1.1.24) has two main consequences. First, if every function has a common center, the value of the derivative with respect to the coordinates of this center is zero. Second, when we wish to evaluate the derivatives of an integral with respect to all four centers, it suffices to only compute, for example, the A_x, B_x, and C_x derivatives and the one with respect to D_x can be given in a more simple manner by rearranging Eq. (1.1.24). Such an evaluation of derivatives also reduces the redundancy of the calculation since the recursive methods that are used to compute the ERIs required for their derivatives can share recursion intermediates. Proceeding in such a way is not always the preferred choice, for example, when we wish to produce derivatives with respect to a couple of degrees of freedom at a time for a coarse-grained parallelization scheme where the derivatives with respect to the coordinates of a center are evaluated on a separate node, or when the solution of response equations is necessary for each degree of freedom, as is the case for the calculation of second derivatives, and we wish to avoid the storage and sorting of the integral derivatives.

1.2 McMurchie–Davidson method

The strategy of the MD method [31] was originally to expand ERIs over Cartesian overlap distributions stemming from the $G_{I_aJ_aK_a}G_{I_bJ_bK_b}$ and $G_{I_cJ_cK_c}G_{I_dJ_dK_d}$ products in Eq. (1.1.15) into integrals over unscaled Hermite Gaussian functions, defined by Eq. (1.1.11), centered on $\mathbf{P}$ and $\mathbf{Q}$. To rationalize this approach, consider the application of the Gaussian product rule, Eq. (1.1.5), to the x-dependent part of the product of the first two functions in a four-center ERI:

$$G_{I_a}G_{I_b} = x_{\mathrm{A}}^{I_a}x_{\mathrm{B}}^{I_b}\kappa_{ab}^{x}\exp(-px_{\mathrm{P}}^2) = (x_{\mathrm{P}}+X_{\mathrm{PA}})^{I_a}(x_{\mathrm{P}}+X_{\mathrm{PB}})^{I_b}\kappa_{ab}^{x}\exp(-px_{\mathrm{P}}^2) = \sum_{\bar{I}_p=0}^{I_a+I_b} E_{\bar{I}_p}^{I_a,I_b} H_{\bar{I}_p} \ . \tag{1.2.1}$$

Here the relations

$$x_{\mathrm{A}} = x_{\mathrm{P}} + X_{\mathrm{PA}} \ , \ x_{\mathrm{B}} = x_{\mathrm{P}} + X_{\mathrm{PB}} \tag{1.2.2}$$

were exploited, and expressions for the E coefficients will be derived later. The polynomial in x_{P} resulting from the product $(x_{\mathrm{P}}+X_{\mathrm{PA}})^{I_a}(x_{\mathrm{P}}+X_{\mathrm{PB}})^{I_b}$ can be expanded in the polynomials appearing in $H_{\bar{i}_p}$ for $0 \le \bar{i}_p \le I_a + I_b$. Here and from now on, lower case letters for the angular momentum vector and its components (e.g., $\bar{\boldsymbol{l}}_p$ and $\bar{i}_p$) will refer to functions appearing in ERIs that are intermediates of the calculations of the target ERIs, for which upper case letters (such as $\boldsymbol{L}_a$ and I_a) will be used. Also taking into consideration the other Cartesian directions and the overlap distribution in the ket side, the four-center ERI can be written as

$$\begin{aligned}(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d) &= \sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b} \sum_{\bar{j}_p=0}^{J_a+J_b} E_{\bar{j}_p}^{J_a,J_b} \sum_{\bar{k}_p=0}^{K_a+K_b} E_{\bar{k}_p}^{K_a,K_b} \\ &\times \sum_{\bar{i}_q=0}^{I_c+I_d} E_{\bar{i}_q}^{I_c,I_d} \sum_{\bar{j}_q=0}^{J_c+J_d} E_{\bar{j}_q}^{J_c,J_d} \sum_{\bar{k}_q=0}^{K_c+K_d} E_{\bar{k}_q}^{K_c,K_d} (\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q) \ .\end{aligned} \tag{1.2.3}$$

It is possible to write the two-center unscaled Hermite ERIs appearing on the right-hand side of Eq. (1.2.3) in a simple form [31] as

$$\begin{aligned}(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q) &= \frac{\partial^{\bar{l}_p+\bar{l}_q}(\mathbf{0}|\mathbf{0})}{\partial P_x^{\bar{i}_p}\partial P_y^{\bar{j}_p}\partial P_z^{\bar{k}_p}\partial Q_x^{\bar{i}_q}\partial Q_y^{\bar{j}_q}\partial Q_z^{\bar{k}_q}} \\ &= (-1)^{\bar{l}_q}\frac{\partial^{\bar{l}_p+\bar{l}_q}(\mathbf{0})}{\partial P_x^{\bar{i}_p+\bar{i}_q}\partial P_y^{\bar{j}_p+\bar{j}_q}\partial P_z^{\bar{k}_p+\bar{k}_q}} = (-1)^{\bar{l}_q}(\bar{\boldsymbol{l}}_u) \ ,\end{aligned} \tag{1.2.4}$$

where we first moved the differential operators out of the integral, then, exploiting the translational invariance for two-center integrals [that is, $-\partial/\partial P_x(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q) = \partial/\partial Q_x(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q)$], we recast $(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q)$ into the effectively one-center quantities $(\bar{\boldsymbol{l}}_u)$ with $\bar{\boldsymbol{l}}_u = \bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q$. These are practically derivatives of the Boys function, and can be computed by the recurrence [31]

$$(\bar{\boldsymbol{l}}_u + \mathbf{1}_x)^{(n)} = X_{\mathrm{PQ}}(\bar{\boldsymbol{l}}_u)^{(n+1)} + \bar{i}_u(\bar{\boldsymbol{l}}_u - \mathbf{1}_x)^{(n+1)} \tag{1.2.5}$$

from the starting values

$$(\mathbf{0})^{(n)} = (-2\alpha)^n\theta_{pq}F_n(\alpha R_{\mathrm{PQ}}^2) \ . \tag{1.2.6}$$

In the following superscript (n) will only be kept when n is not equal to zero.

The E expansion coefficients in Eq. (1.2.3) can be computed by recursion [31, 35]. These recurrence relations are of great aid when setting up other ERI evaluation schemes

[35], and I will later derive such coefficients for distributions arising from the product of Hermite Gaussians, and from multiplying Cartesian and Hermite Gaussians. Therefore it is pertinent to show here how these quantities for the expansion of Cartesian distributions are derived [35]. We are looking for the coefficients that satisfy

$$G_{i_a} G_{i_b} = \sum_{\bar{i}_p=0}^{i_a+i_b} E_{\bar{i}_p}^{i_a,i_b} H_{\bar{i}_p} \tag{1.2.7}$$

Incrementing i_a by one and applying Eqs. (1.1.3) and (1.2.2) we get

$$G_{i_a+1} G_{i_b} = x_{\mathrm{A}} G_{i_a} G_{i_b} = (x_{\mathrm{P}} + X_{\mathrm{PA}}) G_{i_a} G_{i_b} = \sum_{\bar{i}_p=0}^{i_a+i_b+1} E_{\bar{i}_p}^{i_a+1,i_b} H_{\bar{i}_p} \ . \tag{1.2.8}$$

To obtain a recurrence between the E coefficients we rewrite the expression after the second equation sign according to Eq. (1.2.7), then collect the E values associated with $H_{\bar{i}_p}$. For $x_{\mathrm{P}} G_{i_a} G_{i_b}$ we write

$$\begin{aligned} x_{\mathrm{P}} G_{i_a} G_{i_b} = \sum_{\bar{i}_p=0}^{i_a+i_b} E_{\bar{i}_p}^{i_a,i_b} x_{\mathrm{P}} H_{\bar{i}_p} &= \sum_{\bar{i}_p=0}^{i_a+i_b} E_{\bar{i}_p}^{i_a,i_b} \Big\{ \frac{1}{2p} H_{\bar{i}_p+\bar{1}} + \bar{i}_p H_{\bar{i}_p-\bar{1}} \Big\} \\ &= \sum_{\bar{i}_p=0}^{i_a+i_b+1} H_{\bar{i}_p} \Big\{ \frac{1}{2p} E_{\bar{i}_p-\bar{1}}^{i_a,i_b} + (\bar{i}_p+\bar{1}) E_{\bar{i}_p+\bar{1}}^{i_a,i_b} \Big\} \ , \end{aligned} \tag{1.2.9}$$

noting that

$$E_{\bar{i}_p}^{i_a,i_b} = 0, \quad \bar{i}_p < 0 \text{ or } \bar{i}_p > i_a + i_b \ . \tag{1.2.10}$$

Substituting Eq. (1.2.9) into Eq. (1.2.8), expanding $X_{\mathrm{PA}} G_{i_a} G_{i_b}$ according to Eq. (1.2.7), then collecting the terms belonging to the unscaled Hermite Gaussian of the same order results in the relation

$$E_{\bar{i}_p}^{i_a+1,i_b} = \frac{1}{2p} E_{\bar{i}_p-\bar{1}}^{i_a,i_b} + X_{\mathrm{PA}} E_{\bar{i}_p}^{i_a,i_b} + (\bar{i}_p+\bar{1}) E_{\bar{i}_p+\bar{1}}^{i_a,i_b} \ . \tag{1.2.11}$$

It can be seen in the same manner that the recurrence to increment i_b is

$$E_{\bar{i}_p}^{i_a,i_b+1} = \frac{1}{2p} E_{\bar{i}_p-\bar{1}}^{i_a,i_b} + X_{\mathrm{PB}} E_{\bar{i}_p}^{i_a,i_b} + (\bar{i}_p+\bar{1}) E_{\bar{i}_p+\bar{1}}^{i_a,i_b} \ . \tag{1.2.12}$$

Eqs. (1.2.11) and (1.2.12) use the starting value

$$E_0^{0,0} = \kappa_{ab}^x \ . \tag{1.2.13}$$

Inspecting Eqs. (1.2.11) to (1.2.13) and exploiting that, from the definition of $\mathbf{P}$ and p, we have

$$X_{\mathrm{PA}} = -\frac{b}{p} X_{\mathrm{AB}} \ , \ X_{\mathrm{PB}} = \frac{a}{p} X_{\mathrm{AB}} \tag{1.2.14}$$

and also

$$\frac{\partial}{\partial P_x} = \frac{\partial}{\partial A_x} + \frac{\partial}{\partial B_x} \ , \tag{1.2.15}$$

we can show that the expansion coefficients are independent of P_x. This observation allows us to obtain other recurrences. Differentiating both sides of Eq. (1.2.7) with respect to P_x by utilizing Eqs. (1.2.15), (1.1.4), and (1.1.13) we arrive at

$$2aG_{i_a+1}G_{i_b} - i_a G_{i_a-1}G_{i_b} + 2bG_{i_a}G_{i_b+1} - i_b G_{i_a}G_{i_b-1} = \sum_{\bar{i}_p=0}^{i_a+i_b} E_{\bar{i}_p}^{i_a,i_b} H_{\bar{i}_p+\bar{1}} \ . \qquad (1.2.16)$$

Substituting Eq. (1.2.7) on the left-hand side and collecting the terms associated with the same unscaled Hermite Gaussians we get

$$2aE_{\bar{i}_p}^{i_a+1,i_b} - i_a E_{\bar{i}_p}^{i_a-1,i_b} + 2bE_{\bar{i}_p}^{i_a,i_b+1} - i_b E_{\bar{i}_p}^{i_a,i_b-1} = E_{\bar{i}_p-\bar{1}}^{i_a,i_b} \ . \qquad (1.2.17)$$

Next, we insert Eqs. (1.2.11) and (1.2.12) into the first and third terms of Eq. (1.2.17), respectively, then apply Eq. (1.2.14) and $p = a + b$, and after canceling and rearranging terms we obtain

$$2p(\bar{i}_p + \bar{1})E_{\bar{i}_p+\bar{1}}^{i_a,i_b} = i_a E_{\bar{i}_p}^{i_a-1,i_b} + i_b E_{\bar{i}_p}^{i_a,i_b-1} \ . \qquad (1.2.18)$$

We derive our final recurrence by substituting Eq. (1.2.18) into the third term on the right-hand side of Eq. (1.2.11) to get

$$E_{\bar{i}_p}^{i_a+1,i_b} = X_{\mathrm{PA}} E_{\bar{i}_p}^{i_a,i_b} + \frac{1}{2p}\left(E_{\bar{i}_p-\bar{1}}^{i_a,i_b} + i_a E_{\bar{i}_p}^{i_a-1,i_b} + i_b E_{\bar{i}_p}^{i_a,i_b-1}\right) \ . \qquad (1.2.19)$$

Eq. (1.2.19) will be used in the derivation of the Obara–Saika and Rys methods, and recurrences analogous to it will be derived in subsection 3.1.4 to extend the Rys scheme to ERIs containing Hermite Gaussian functions.

The full set of expansion coefficients required for Eq. (1.2.3) can be computed by Eqs. (1.2.11) and (1.2.12). It is possible to develop more efficient recursions [35] to produce these quantities, however, their computation is of negligible cost compared to the assembly of the two-center unscaled Hermite ERIs, especially when high angular momenta are involved. Before considering an alternative scheme to calculate a target ERI from the two-center ones it is probably helpful to illustrate how an ERI is evaluated by the method described so far. From Eq. (1.2.3) the integral $(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z)$ can be written as

$$(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z) = \kappa_{ab}^{y}\kappa_{ab}^{z}\kappa_{cd}^{x} \sum_{\bar{i}_p=0}^{2} E_{\bar{i}_p}^{1,1} \sum_{\bar{j}_q=0}^{1} E_{\bar{j}_q}^{1,0} \sum_{\bar{k}_q=0}^{1} E_{\bar{k}_q}^{0,1} (H_{\bar{i}_p00}|H_{0\bar{j}_q\bar{k}_q}) \ , \qquad (1.2.20)$$

where, for illustrative purposes, the more compact angular momentum vector notation was not applied, and Eq (1.2.13) was substituted for the $E_0^{0,0}$ values. The expansion coefficients are calculated from Eqs. (1.2.10) to (1.2.13). For $H_{\bar{i}_p}$ we need the $E_{\bar{i}_p}^{i_a,i_b}$ quantities $E_0^{1,1}$, $E_{\bar{1}}^{1,1}$, and $E_{\bar{2}}^{1,1}$. With the conditions given by Eq. (1.2.10) and from the fact that i_a and i_b are nonnegative we get

$$\begin{aligned}
E_{\bar{2}}^{1,1} &= \frac{1}{2p} E_{\bar{1}}^{1,0} \ , \\
E_{\bar{1}}^{1,0} &= \frac{1}{2p} \kappa_{ab}^{x} \ , \\
E_{\bar{1}}^{1,1} &= \frac{1}{2p} E_{0}^{1,0} + X_{\mathrm{PB}} E_{\bar{1}}^{1,0} \ , \\
E_{0}^{1,0} &= X_{\mathrm{PA}} \kappa_{ab}^{x} \ , \\
E_{0}^{1,1} &= X_{\mathrm{PB}} E_{0}^{1,0} + E_{\bar{1}}^{1,0} \ . \qquad (1.2.21)
\end{aligned}$$

According to Eq. (1.2.4) we need one-center ERIs with $l_u = 0, \ldots, 4$, computed by Eq. (1.2.5). The calculation is performed for $(\bar{\boldsymbol{g}}_{x^2yz})$ by the following relations

$$\begin{aligned}
(\bar{\boldsymbol{g}}_{x^2yz}) &= X_{\mathrm{PQ}} (\bar{\boldsymbol{f}}_{xyz})^{(1)} + (\bar{\boldsymbol{d}}_{yz})^{(1)} \ , \\
(\bar{\boldsymbol{f}}_{xyz})^{(1)} &= X_{\mathrm{PQ}} (\bar{\boldsymbol{d}}_{yz})^{(2)} \ , \\
(\bar{\boldsymbol{d}}_{yz})^{(1)} &= Y_{\mathrm{PQ}} (\bar{\boldsymbol{p}}_{z})^{(2)} \ , \\
(\bar{\boldsymbol{d}}_{yz})^{(2)} &= Y_{\mathrm{PQ}} (\bar{\boldsymbol{p}}_{z})^{(3)} \ , \\
(\bar{\boldsymbol{p}}_{z})^{(2)} &= Z_{\mathrm{PQ}} (\bar{\boldsymbol{s}})^{(3)} \ , \\
(\bar{\boldsymbol{p}}_{z})^{(3)} &= Z_{\mathrm{PQ}} (\bar{\boldsymbol{s}})^{(4)} \ , \qquad (1.2.22)
\end{aligned}$$

where $(\bar{\boldsymbol{s}})^{(n)} \equiv (\mathbf{0})^{(n)}$. Note that the recursion given by Eq. (1.2.5) is overdefined, meaning that $(\bar{\boldsymbol{g}}_{x^2yz})$ could be built up by other paths, that is, the components of the angular momentum vector could be incremented in various orders, using other intermediate ERIs. This will be the case with most of the recurrence relations discussed in this work.

In the case of higher angular momenta it is more beneficial to relate the $(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q)$ integrals to the target ones by recursion [32]. For this purpose consider the auxiliary function

$$\Omega_{i_a,i_b}^{\bar{i}_p} = x_{\mathrm{A}}^{i_a} x_{\mathrm{B}}^{i_b} \Big(\frac{\partial}{\partial P_x} \Big)^{\bar{i}_p} \Lambda_{ab}^{x} \ , \qquad (1.2.23)$$

which is the x-dependent part of the function $\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p} = \Omega_{i_a,i_b}^{\bar{i}_p} \Omega_{j_a,j_b}^{\bar{j}_p} \Omega_{k_a,k_b}^{\bar{k}_p}$. It is clear that $\Omega_{0,0}^{\bar{i}_p}$ is equal to $\kappa_{ab}^{x} H_{\bar{i}_p}$, and Ω_{i_a,i_b}^{0} is identical to the Cartesian Gaussian product $G_{i_a} G_{i_b}$. To arrive at a suitable recurrence we increment i_a in Eq. (1.2.23), then utilize Eqs. (1.2.2) and (1.1.12) to obtain

$$\Omega_{i_a+1,i_b}^{\bar{i}_p} = x_{\mathrm{A}} \Omega_{i_a,i_b}^{\bar{i}_p} = (x_{\mathrm{P}} + X_{\mathrm{PA}}) \Omega_{i_a,i_b}^{\bar{i}_p} = \bar{i}_p \Omega_{i_a,i_b}^{\bar{i}_p - \bar{1}} + X_{\mathrm{PA}} \Omega_{i_a,i_b}^{\bar{i}_p} + \frac{1}{2p} \Omega_{i_a,i_b}^{\bar{i}_p + \bar{1}} \ . \qquad (1.2.24)$$

Similarly, when we increment i_b we find

$$\Omega_{i_a,i_b+1}^{\bar{i}_p} = \bar{i}_p \Omega_{i_a,i_b}^{\bar{i}_p - \bar{1}} + X_{\mathrm{PB}} \Omega_{i_a,i_b}^{\bar{i}_p} + \frac{1}{2p} \Omega_{i_a,i_b}^{\bar{i}_p + \bar{1}} \ . \qquad (1.2.25)$$

The equations for the other Cartesian directions and for the buildup of ket-side distribution are analogous to Eqs. (1.2.24) and (1.2.25). With these we can relate the two-center unscaled Hermite ERIs to the four-center Cartesian ones, given that we multiply the quantities defined by Eq. (1.2.6) by $\kappa_{ab}\kappa_{cd}$ before using Eq. (1.2.5).

There exists a more efficient alternative to build up $\boldsymbol{L}_b$ (and similarly, $\boldsymbol{L}_d$) than to apply Eq. (1.2.25). If we increment i_b by one in the product of two Cartesian Gaussians and then use Eqs. (1.1.14) and (1.1.3), we get

$$G_{i_a}G_{i_b+1} = x_{\mathrm{B}}G_{i_a}G_{i_b} = (x_{\mathrm{A}} + X_{\mathrm{AB}})G_{i_a}G_{i_b} = G_{i_a+1}G_{i_b} + X_{\mathrm{AB}}G_{i_a}G_{i_b} \tag{1.2.26}$$

which is usually referred to as the Head-Gordon–Pople (HGP) equation or the horizontal recurrence relation (HRR). The HRR was originally developed in the context of the OS scheme [54], but it can be applied within MD as well after we built up the necessary $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c\boldsymbol{0})$ ERIs via Eq. (1.2.3) or (1.2.24). Note that Eq. (1.2.26) does not depend on any primitive Gaussian exponents, hence it can be applied after the integrals have been contracted according to Eq. (1.1.7), meaning that this recurrence can be performed over the smaller contracted basis. This property is usually considered to be a major advantage of the HRR, however, it will be pointed out later that for three-center ERIs the application of Eq. (1.2.26) at the contracted stage is not necessarily a beneficial strategy.

Let us again consider an example. The $(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z) \equiv (\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_x,\boldsymbol{p}_x}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_y,\boldsymbol{p}_z})$ class is evaluated with the application of Eqs. (1.2.24) and (1.2.26) as the following. The functions centered on $\mathbf{B}$ and $\mathbf{D}$ are built up by Eq. (1.2.26) as

$$\begin{aligned}
(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z) &= (\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{p}_z) + X_{\mathrm{AB}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{p}_z)\ , \\
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{p}_z) &= (\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{d}_{yz}\boldsymbol{s}) + Z_{\mathrm{CD}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{s})\ , \\
(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{p}_z) &= (\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{d}_{yz}\boldsymbol{s}) + Z_{\mathrm{CD}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{s})\ .
\end{aligned} \tag{1.2.27}$$

For the evaluation of the $(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_y\boldsymbol{s}) \equiv (\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_y,\boldsymbol{s}})$ class, for example, we apply Eq. (1.2.24):

$$\begin{aligned}
(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_y,\boldsymbol{s}}) &= Y_{\mathrm{QC}}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}) + \frac{1}{2q}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}})\ , \\
(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) &= X_{\mathrm{PA}}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_x,\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) + \frac{1}{2p}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{p}_x,\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}})\ , \\
(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_x,\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) &= X_{\mathrm{PA}}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) + \frac{1}{2p}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}})\ , \\
(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{p}_x,\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) &= (\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) + X_{\mathrm{PA}}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}}) + \frac{1}{2p}(\Omega^{\bar{\boldsymbol{d}}_{x^2}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{p}}_y}_{\boldsymbol{s},\boldsymbol{s}})\ ,
\end{aligned} \tag{1.2.28}$$

and similarly for $(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})$. The auxiliary integrals with only Hermite indices are produced from the one-center ERIs according to Eq. (1.2.4).

Concerning the ERI derivatives we can consider different options than to directly evaluate the ERIs necessary for Eq. (1.1.20). Ref. 46 details how to calculate the expansion coefficients which can be used to transform two-center Hermite ERIs into four-center Hermite integrals with Eq. (1.2.3). Note that according to Eq. (1.1.8) the $(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q)$

integrals have to be multiplied by $2p^{-\tilde{l}_p}2q^{-\tilde{l}_q}$ to obtain the $(\tilde{\boldsymbol{l}}_p|\tilde{\boldsymbol{l}}_q)$ ERIs required for this purpose. This way the integral derivatives can be computed by the less expensive Eq. (1.1.21). A recursion similar to Eq. (1.2.24) for the computation of Hermite ERIs has not been presented in the literature before. I will derive a suitable equation in a later chapter. Finally, the approach of Helgaker and Taylor [55] to compute Cartesian ERI derivatives with the MD scheme should be mentioned. The order of the polynomial in x_{P} that we aim to expand in Eq. (1.2.1) is higher by one in the case of the derivative integrals since these involve a Gaussian with incremented angular momentum. This means that the range of summation for the corresponding expansion coefficient in Eq. (1.2.3) is increased by one. Alternatively, one can exploit that [35]

$$\frac{\partial}{\partial A_x} = \frac{a}{p}\frac{\partial}{\partial P_x} + \frac{\partial}{\partial X_{\mathrm{AB}}} \ , \tag{1.2.29}$$

and that the unscaled Hermite ERIs centered on $\mathbf{P}$ are independent of X_{AB} [55]. As it was mentioned before the E coefficients are independent of P_x, so we can write the relations

$$\frac{\partial G_{i_a}G_{i_b}}{\partial P_x} = \sum_{\bar{i}_p=0}^{i_a+i_b} E_{\bar{i}_p}^{i_a,i_b} H_{\bar{i}_p+\bar{1}} \qquad \frac{\partial G_{i_a}G_{i_b}}{\partial X_{\mathrm{AB}}} = \sum_{\bar{i}_p=0}^{i_a+i_b} \partial E_{\bar{i}_p}^{i_a,i_b} H_{\bar{i}_p} \tag{1.2.30}$$

with $\partial E_{\bar{i}_p}^{i_a,i_b} \equiv \partial E_{\bar{i}_p}^{i_a,i_b}/\partial X_{\mathrm{AB}}$. The recursive evaluation of the differentiated coefficients is discussed in Ref. 55. Based on Eq. (1.2.3) it follows that

$$\frac{\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)}{\partial P_x} = \sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b} \dots \ (\bar{\boldsymbol{l}}_p + \bar{\mathbf{1}}_x + \bar{\boldsymbol{l}}_q) \tag{1.2.31}$$

and

$$\frac{\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)}{\partial X_{\mathrm{AB}}} = \sum_{\bar{i}_p=0}^{I_a+I_b} \partial E_{\bar{i}_p}^{I_a,I_b} \dots \ (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q) \ . \tag{1.2.32}$$

where the summations over the other five expansion coefficients in Eq. (1.2.3) have not been displayed. We recover the A_x derivatives from Eq. (1.2.29), while the B_x derivatives are computed from Eq. (1.2.15). Since $\partial/\partial Q_x = \partial/\partial C_x + \partial/\partial D_x$ analogously to Eq. (1.2.15), it follows from Eq. (1.1.24) that $-\partial/\partial P_x = \partial/\partial Q_x$, so one of the derivatives required to apply the above scheme for the ket side derivatives is trivial when we compute all of the derivatives of an ERI in a common algorithm.

1.3 Gill–Head-Gordon–Pople method

The essence of the GHP scheme [32] is to modify Eqs. (1.2.24) and (1.2.25) so that they can be applied to integrals over the contracted functions. The approach utilizes the auxiliary functions

$$\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} = \frac{(2a)^{\lambda}(2b)^{\beta}}{(2p)^{\zeta}}\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b} \ , \tag{1.3.1}$$

where λ, β, and ζ are integers. With these functions, and also using Eq. (1.2.14), Eqs. (1.2.24) and (1.2.25) can be recast as the contracted transfer equations

$$\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a+\boldsymbol{1}_x,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} = \bar{i}_p\Omega^{\bar{\boldsymbol{l}}_p-\bar{\boldsymbol{1}}_x}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} - X_{\mathrm{AB}}\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta+1,\zeta+1} + \Omega^{\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{1}}_x}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta+1} \tag{1.3.2}$$

and

$$\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b+\boldsymbol{1}_x}|_{\lambda,\beta,\zeta} = \bar{i}_p\Omega^{\bar{\boldsymbol{l}}_p-\bar{\boldsymbol{1}}_x}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} + X_{\mathrm{AB}}\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda+1,\beta,\zeta+1} + \Omega^{\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{1}}_x}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta+1} \ , \tag{1.3.3}$$

respectively. For example, in the third term on the right-hand side in both Eq. (1.3.2) and Eq. (1.3.3) the ζ index is increased by one, because in Eqs. (1.2.24) and (1.2.25) these terms are multiplied by $1/(2p)$, corresponding to an increment in ζ in Eq. (1.3.1). The auxiliary functions and recurrences for the buildup of $\boldsymbol{L}_c\boldsymbol{L}_d$ are analogous to the equations above, with the ket-side scalings denoted as λ', β', and ζ'. Eqs. (1.3.2) and (1.3.3) do not make any explicit reference to primitive Gaussian exponents, hence these recurrences can be performed at the contracted stage, given that the required λ, β, ζ (and λ', β', ζ') -scaled two-center ERIs have been transformed into the contracted basis. The strategy of the GHP scheme for four-center ERIs is thus the following. First, the necessary $(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{0},\boldsymbol{0}}|\Omega^{\bar{\boldsymbol{l}}_q}_{\boldsymbol{0},\boldsymbol{0}}) \equiv (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q)$ integral classes are computed. Then, all the scaled versions of these integrals [denoted as ${}_{\lambda,\beta,\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{0},\boldsymbol{0}}|\Omega^{\bar{\boldsymbol{l}}_q}_{\boldsymbol{0},\boldsymbol{0}})_{\lambda',\beta',\zeta'}$] required to compute the ${}_{0,0,0}(\Omega^{\boldsymbol{0}}_{\boldsymbol{L}_a,\boldsymbol{L}_b}|\Omega^{\boldsymbol{0}}_{\boldsymbol{L}_c,\boldsymbol{L}_d})_{0,0,0} \equiv (\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)$ classes are produced. After the scaled integrals have been contracted by Eq. (1.1.7), Eqs. (1.3.2) and (1.3.3) are applied to obtain the target ERIs. It is usually more efficient [32] to only utilize Eq. (1.3.2) and its analogue to create $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c\boldsymbol{0})$ type intermediates, then to exploit Eq. (1.2.26). It is not trivial to say when it is beneficial to apply the GHP algorithm. While the contracted basis is usually smaller than the primitive Gaussian basis, the number of the required scaled classes increases with the target angular momenta. Since the number of primitives an AO is composed from generally decreases with the increasing angular momentum, the GHP scheme is usually successful for ERIs containing $\boldsymbol{s}$ or $\boldsymbol{p}$ functions.

To illustrate the algorithm consider the evaluation of the ${}_{0,0,0}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})_{0,0,0}$ class of ERIs. Since Eqs. (1.3.2) and (1.3.3) only depend on the coordinates of one electron, the reference to the ket side of the integrals is omitted from the relations. The necessary calculational steps are

$$\begin{aligned}
{}_{0,0,0}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{d}_{x^2},\boldsymbol{s}}| &= -X_{\mathrm{AB}}\ {}_{0,1,1}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_x,\boldsymbol{s}}| + {}_{0,0,1}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{p}_x,\boldsymbol{s}}| \ , \\
{}_{0,1,1}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{p}_x,\boldsymbol{s}}| &= -X_{\mathrm{AB}}\ {}_{0,2,2}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}| + {}_{0,1,2}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{s},\boldsymbol{s}}| \ , \\
{}_{0,0,1}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{p}_x,\boldsymbol{s}}| &= {}_{0,0,1}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}| - X_{\mathrm{AB}}\ {}_{0,1,2}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{s},\boldsymbol{s}}| + {}_{0,0,2}(\Omega^{\bar{\boldsymbol{d}}_{x^2}}_{\boldsymbol{s},\boldsymbol{s}}| \ .
\end{aligned} \tag{1.3.4}$$

Note that the function $\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}$ has two different scalings associated with it, ${}_{0,0,1}$ and ${}_{0,2,2}$. Thus we need to produce all of the ${}_{0,0,1}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})_{0,0,0}$, ${}_{0,2,2}(\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})_{0,0,0}$, ${}_{0,1,2}(\Omega^{\bar{\boldsymbol{p}}_x}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})_{0,0,0}$, and ${}_{0,0,2}(\Omega^{\bar{\boldsymbol{d}}_{x^2}}_{\boldsymbol{s},\boldsymbol{s}}|\Omega^{\bar{\boldsymbol{s}}}_{\boldsymbol{s},\boldsymbol{s}})_{0,0,0}$ scaled primitive classes, transform these into the contracted basis via Eq. (1.1.7), then perform the operations in Eq. (1.3.4).

When the method is applied to derivatives of Cartesian ERIs the necessary scaled classes become more numerous. For example, according to Eq. (1.1.20), the computation of $([\partial \boldsymbol{L}_a/\partial A_x]\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)$ requires the evaluation of the classes ${}_{1,0,0}(\Omega^{\mathbf{0}}_{\boldsymbol{L}_a+\mathbf{1}_x,\boldsymbol{L}_b}|\Omega^{\mathbf{0}}_{\boldsymbol{L}_c,\boldsymbol{L}_d})_{0,0,0}$ and ${}_{0,0,0}(\Omega^{\mathbf{0}}_{\boldsymbol{L}_a-\mathbf{1}_x,\boldsymbol{L}_b}|\Omega^{\mathbf{0}}_{\boldsymbol{L}_c,\boldsymbol{L}_d})_{0,0,0}$, each of which requires different scaled classes for Eq. (1.3.2). According to Eq. (1.1.21) utilizing Hermite Gaussians would require only one class to calculate the derivative ERIs. The recursions corresponding to Eqs. (1.3.2) and (1.3.3) applicable to ERIs over Hermite Gaussians or both Hermite and Cartesian Gaussians will be derived in a later chapter. It should be mentioned that the GHP scheme was developed further to treat arbitrary order geometrical derivatives as well [56]. This was achieved by introducing the auxiliary functions

$$ {}_{i'_a,i'_b}\Omega^{\bar{i}_p}_{i_a,i_b}|_{\lambda,\beta,\zeta} = \frac{\partial^{i'_a}}{\partial A_x^{i'_a}}\frac{\partial^{i'_b}}{\partial B_x^{i'_b}}\Omega^{\bar{i}_p}_{i_a,i_b}|_{\lambda,\beta,\zeta} \ , \tag{1.3.5} $$

and developing recursions for $i_a \to i'_a$, $i_a \to i_b$, $\bar{i}_p \to i_a$, $\bar{i}_p \to i'_a$, and $i'_a \to i'_b$ translations between such functions. These relations offer new pathways for the evaluation of the derivative integrals, however, such approaches are mostly interesting for higher-order derivatives, and these routes will not be investigated in the present work.

1.4 Obara–Saika method

The OS scheme [33, 57], related to the earlier method of Schlegel [58] for the evaluation of the derivatives of ERIs over $\boldsymbol{s}$- and $\boldsymbol{p}$-type Gaussians, applies relations between the auxiliary integrals $(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}$ to reach the target ERIs with $n = 0$. The presentation of the method will follow Ref. 35 rather than the original derivation. We will thus write the auxiliary ERIs in accordance with Eqs. (1.2.3) and (1.2.4) as

$$ (\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} = T_n \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E^{i_a,i_b}_{\bar{i}_p} E^{i_c,i_d}_{\bar{i}_q} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q)^{(n)} \ , \tag{1.4.1} $$

where

$$ T_n = \theta_{pq}(-2\alpha)^{-n} \tag{1.4.2} $$

and

$$ \sum_{\bar{j},\bar{k}} E_{\bar{j},\bar{k}} = \sum_{\bar{j}_p=0}^{j_a+j_b} \sum_{\bar{k}_p=0}^{k_a+k_b} \sum_{\bar{j}_q=0}^{j_c+j_d} \sum_{\bar{k}_q=0}^{k_c+k_d} E^{j_a,j_b}_{\bar{j}_p} E^{k_a,k_b}_{\bar{k}_p} E^{j_c,j_d}_{\bar{j}_q} E^{k_c,k_d}_{\bar{k}_q} \ . \tag{1.4.3} $$

Eq. (1.4.1) emphasizes the parts of the expression that depend on the x direction, since the recursion to increment the x component of the angular momentum vectors will be independent from the other directions. To arrive at the desired equation we first increment one of the indices, e.g., i_a, and get

$$([\boldsymbol{l}_a + \mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} = T_n \sum_{\bar{i}_p=0}^{i_a+i_b+1} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E_{\bar{i}_p}^{i_a+1,i_b} E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q)^{(n)} \;, \qquad (1.4.4)$$

into which we insert Eq. (1.2.19), and utilizing Eqs. (1.4.1) and (1.2.10) we obtain

$$\begin{aligned} ([\boldsymbol{l}_a + \mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} \\ &+ \frac{1}{2p}\Big(i_a([\boldsymbol{l}_a - \mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} + i_b(\boldsymbol{l}_a[\boldsymbol{l}_b - \mathbf{1}_x]|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}\Big) + \frac{1}{2p}U_n \qquad (1.4.5) \end{aligned}$$

where

$$U_n = T_n \sum_{\bar{i}_p=0}^{i_a+i_b+1} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E_{\bar{i}_p-1}^{i_a,i_b} E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q)^{(n)} \;. \qquad (1.4.6)$$

From Eq. (1.2.10) it follows that the terms belonging to $\bar{i}_p = 0$ in U_n are zero. Thus the expression remains the same if we substitute $\bar{i}_p + 1$ in the place of $\bar{i}_p$ and let $\bar{i}_p$ run from 0 to $i_a + i_b$. This way we get

$$U_n = T_n \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E_{\bar{i}_p}^{i_a,i_b} E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q + \bar{\mathbf{1}}_x)^{(n)} \;. \qquad (1.4.7)$$

Next we exploit Eqs. (1.2.5) and (1.4.2), and write

$$\begin{aligned} U_n &= -2\alpha X_{\mathrm{PQ}} T_{n+1} \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E_{\bar{i}_p}^{i_a,i_b} E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q)^{(n+1)} \\ &- 2\alpha T_{n+1} \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} (\bar{i}_p + \bar{i}_q) E_{\bar{i}_p}^{i_a,i_b} E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q - \mathbf{1}_x)^{(n+1)} \\ &= -2\alpha X_{\mathrm{PQ}} (\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)} \\ &- \frac{\alpha}{p} T_{n+1} \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} (2p\bar{i}_p E_{\bar{i}_p}^{i_a,i_b}) E_{\bar{i}_q}^{i_c,i_d} E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q - \mathbf{1}_x)^{(n+1)} \\ &- \frac{\alpha}{q} T_{n+1} \sum_{\bar{i}_p=0}^{i_a+i_b} \sum_{\bar{i}_q=0}^{i_c+i_d} \sum_{\bar{j},\bar{k}} (-1)^{\bar{l}_q} E_{\bar{i}_p}^{i_a,i_b} (2q\bar{i}_q E_{\bar{i}_q}^{i_c,i_d}) E_{\bar{j},\bar{k}} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_q - \mathbf{1}_x)^{(n+1)} \;, \qquad (1.4.8) \end{aligned}$$

where after the second equation sign we applied Eq. (1.4.1) for the first term, and factored out $1/(2p)$ and $1/(2q)$ in the second and third terms, respectively. This latter rearrangement was performed so that we could invoke Eq. (1.2.18) and arrive at

$$\begin{aligned}
U_n &= -2\alpha X_{\mathrm{PQ}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)} \\
&- i_a\frac{\alpha}{p}T_{n+1}\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}\sum_{\bar{j},\bar{k}}(-1)^{\bar{l}_q}E^{i_a-1,i_b}_{\bar{i}_p-\bar{1}}E^{i_c,i_d}_{\bar{i}_q}E_{\bar{j},\bar{k}}(\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{l}}_q-\mathbf{1}_x)^{(n+1)} \\
&- i_b\frac{\alpha}{p}T_{n+1}\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}\sum_{\bar{j},\bar{k}}(-1)^{\bar{l}_q}E^{i_a,i_b-1}_{\bar{i}_p-\bar{1}}E^{i_c,i_d}_{\bar{i}_q}E_{\bar{j},\bar{k}}(\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{l}}_q-\mathbf{1}_x)^{(n+1)} \\
&+ i_c\frac{\alpha}{q}T_{n+1}\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}\sum_{\bar{j},\bar{k}}(-1)^{\bar{l}_q-\bar{1}}E^{i_a,i_b}_{\bar{i}_p}E^{i_c-1,i_d}_{\bar{i}_q-\bar{1}}E_{\bar{j},\bar{k}}(\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{l}}_q-\mathbf{1}_x)^{(n+1)} \\
&+ i_d\frac{\alpha}{q}T_{n+1}\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}\sum_{\bar{j},\bar{k}}(-1)^{\bar{l}_q-\bar{1}}E^{i_a,i_b}_{\bar{i}_p}E^{i_c,i_d-1}_{\bar{i}_q-\bar{1}}E_{\bar{j},\bar{k}}(\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{l}}_q-\mathbf{1}_x)^{(n+1)}\ . \qquad (1.4.9)
\end{aligned}$$

Utilizing Eq. (1.4.1) U_n becomes

$$\begin{aligned}
U_n &= -2\alpha X_{\mathrm{PQ}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)} \\
&- \frac{\alpha}{p}\Big(i_a([\boldsymbol{l}_a-\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}+i_b(\boldsymbol{l}_a[\boldsymbol{l}_b-\mathbf{1}_x]|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}\Big) \\
&+ \frac{\alpha}{q}\Big(i_c(\boldsymbol{l}_a\boldsymbol{l}_b|[\boldsymbol{l}_c-\mathbf{1}_x]\boldsymbol{l}_d)^{(n+1)}+i_d(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c[\boldsymbol{l}_d-\mathbf{1}_x])^{(n+1)}\Big)\ , \qquad (1.4.10)
\end{aligned}$$

which is inserted into Eq. (1.4.5) to finally obtain

$$\begin{aligned}
([\boldsymbol{l}_a+\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}-\frac{\alpha}{p}X_{\mathrm{PQ}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)} \\
&+ \frac{i_a}{2p}\Big(([\boldsymbol{l}_a-\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}-\frac{\alpha}{p}([\boldsymbol{l}_a-\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}\Big) \\
&+ \frac{i_b}{2p}\Big((\boldsymbol{l}_a[\boldsymbol{l}_b-\mathbf{1}_x]|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}-\frac{\alpha}{p}(\boldsymbol{l}_a[\boldsymbol{l}_b-\mathbf{1}_x]|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}\Big) \\
&+ \frac{\alpha}{2pq}\Big(i_c(\boldsymbol{l}_a\boldsymbol{l}_b|[\boldsymbol{l}_c-\mathbf{1}_x]\boldsymbol{l}_d)^{(n+1)}+i_d(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c[\boldsymbol{l}_d-\mathbf{1}_x])^{(n+1)}\Big)\ . \qquad (1.4.11)
\end{aligned}$$

Eq. (1.4.11), often referred to as the vertical recurrence relation (VRR), has the remarkable feature that every required intermediate ERI has lower angular momenta than the target integral on the left-hand side. The analogous recursion to increment i_b can be derived by substituting Eq. (1.4.11) into Eq. (1.2.26) and using $X_{\mathrm{PB}} = X_{\mathrm{PA}} + X_{\mathrm{AB}}$. The corresponding relation to build up i_c is

$$\begin{aligned}
(\boldsymbol{l}_a\boldsymbol{l}_b|[\boldsymbol{l}_c+\mathbf{1}_x]\boldsymbol{l}_d)^{(n)} &= X_{\mathrm{QC}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n)}+\frac{\alpha}{q}X_{\mathrm{PQ}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)} \\
&+ \frac{i_c}{2q}\Big((\boldsymbol{l}_a\boldsymbol{l}_b|[\boldsymbol{l}_c-\mathbf{1}_x]\boldsymbol{l}_d)^{(n)}-\frac{\alpha}{q}(\boldsymbol{l}_a\boldsymbol{l}_b|[\boldsymbol{l}_c-\mathbf{1}_x]\boldsymbol{l}_d)^{(n+1)}\Big) \\
&+ \frac{i_d}{2q}\Big((\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c[\boldsymbol{l}_d-\mathbf{1}_x])^{(n)}-\frac{\alpha}{q}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c[\boldsymbol{l}_d-\mathbf{1}_x])^{(n+1)}\Big) \\
&+ \frac{\alpha}{2pq}\Big(i_a([\boldsymbol{l}_a-\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}+i_b(\boldsymbol{l}_a[\boldsymbol{l}_b-\mathbf{1}_x]|\boldsymbol{l}_c\boldsymbol{l}_d)^{(n+1)}\Big)\ . \qquad (1.4.12)
\end{aligned}$$

The eight-term OS recursions get simplified in the case when some of the angular momenta are zero. The recurrence where only $\boldsymbol{l}_a$ is built up,

$$\begin{aligned}([\boldsymbol{l}_a+\mathbf{1}_x]\mathbf{0}|\mathbf{00})^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{00})^{(n)} - \frac{\alpha}{p}X_{\mathrm{PQ}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{00})^{(n+1)} \\ &+ \frac{i_a}{2p}\Big(([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\mathbf{00})^{(n)} - \frac{\alpha}{p}([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\mathbf{00})^{(n+1)}\Big)\ , \qquad (1.4.13)\end{aligned}$$

will be applied later after modification for the three-center ERIs. For four-center integrals with high angular momenta it is more advantageous [35] to work with Eq. (1.2.26) starting from the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0})$ intermediates. These can be produced from $(\boldsymbol{l}_a\mathbf{0}|\mathbf{00})$ type ERIs via the electron transfer relation (ETR) [59, 60], which is derived from the translational invariance, Eq. (1.1.24):

$$\frac{\partial(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0})}{\partial A_x} + \frac{\partial(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0})}{\partial B_x} + \frac{\partial(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0})}{\partial C_x} + \frac{\partial(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0})}{\partial D_x} = 0\ . \qquad (1.4.14)$$

After using Eq. (1.1.20) we get

$$\begin{aligned}2a([\boldsymbol{l}_a+\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) &- i_a([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) + 2b(\boldsymbol{l}_a\mathbf{1}_x|\boldsymbol{l}_c\mathbf{0}) \\ &+ 2c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c+\mathbf{1}_x]\mathbf{0}) - i_c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c-\mathbf{1}_x]\mathbf{0}) + 2d(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{1}_x) = 0\ , \qquad (1.4.15)\end{aligned}$$

where we can invoke Eq. (1.2.26) and $p = a + b,\ q = c + d$ to write

$$\begin{aligned}2p([\boldsymbol{l}_a+\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) &- i_a([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) + 2bX_{\mathrm{AB}}(\boldsymbol{l}_a\mathbf{0}_x|\boldsymbol{l}_c\mathbf{0}) + 2q(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c+\mathbf{1}_x]\mathbf{0}) \\ &- i_c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c-\mathbf{1}_x]\mathbf{0}) + 2dX_{\mathrm{CD}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0}_x) = 0\ . \qquad (1.4.16)\end{aligned}$$

The ETR is obtained by rearranging Eq. (1.4.16) as

$$\begin{aligned}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c+\mathbf{1}_x]\mathbf{0}) &= -\frac{p}{q}([\boldsymbol{l}_a+\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) - \frac{bX_{\mathrm{AB}}+dX_{\mathrm{CD}}}{q}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) \\ &+ \frac{1}{2q}\Big(i_a([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c\mathbf{0}) + i_c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c-\mathbf{1}_x]\mathbf{0})\Big)\ . \qquad (1.4.17)\end{aligned}$$

Based on computer experiments, Ishida [61] noted that the addition in the second term of the right-hand side of Eq. (1.4.17) can lead to numerical instability (loss of significant figures), and he also discussed approaches to avoid this problem. As we will see, this issue does not need to be handled in the case of three-center ERIs.

The application of the VRR and the ETR can be illustrated by the evaluation of the $(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{d}_{x^2}\boldsymbol{s})$ integral. The ETR is utilized as

$$\begin{aligned}
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{d}_{x^2}\boldsymbol{s}) &= -\frac{p}{q}(\boldsymbol{f}_{x^3}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) - \frac{bX_{\mathrm{AB}}+dX_{\mathrm{cd}}}{q}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) \\
&+ \frac{1}{2q}\Big(2(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) + (\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})\Big) , \\
(\boldsymbol{f}_{x^3}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) &= -\frac{p}{q}(\boldsymbol{g}_{x^4}\boldsymbol{s}|\boldsymbol{ss}) - \frac{bX_{\mathrm{AB}}+dX_{\mathrm{cd}}}{q}(\boldsymbol{f}_{x^3}\boldsymbol{s}|\boldsymbol{ss}) \\
&+ \frac{3}{2q}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) , \\
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) &= -\frac{p}{q}(\boldsymbol{f}_{x^3}\boldsymbol{s}|\boldsymbol{ss}) - \frac{bX_{\mathrm{AB}}+dX_{\mathrm{cd}}}{q}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) \\
&+ \frac{2}{2q}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss}) , \\
(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) &= -\frac{p}{q}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) - \frac{bX_{\mathrm{AB}}+dX_{\mathrm{cd}}}{q}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss}) \\
&+ \frac{1}{2q}(\boldsymbol{ss}|\boldsymbol{ss}) , \qquad (1.4.18)
\end{aligned}$$

while, to calculate the $(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})$ ERI for example, the required VRR steps are

$$\begin{aligned}
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) &= X_{\mathrm{PA}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss}) - \frac{\alpha}{p}X_{\mathrm{PQ}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(1)} \\
&+ \frac{1}{2p}\Big((\boldsymbol{ss}|\boldsymbol{ss}) - \frac{\alpha}{p}(\boldsymbol{ss}|\boldsymbol{ss})^{(1)}\Big) , \\
(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss}) &= X_{\mathrm{PA}}(\boldsymbol{ss}|\boldsymbol{ss}) - \frac{\alpha}{p}X_{\mathrm{PQ}}(\boldsymbol{ss}|\boldsymbol{ss})^{(1)} , \\
(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(1)} &= X_{\mathrm{PA}}(\boldsymbol{ss}|\boldsymbol{ss})^{(1)} - \frac{\alpha}{p}X_{\mathrm{PQ}}(\boldsymbol{ss}|\boldsymbol{ss})^{(2)} . \qquad (1.4.19)
\end{aligned}$$

Alternatively, if we only employed the VRR to evaluate $(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{d}_{x^2}\boldsymbol{s})$, the buildup of $\boldsymbol{L}_c$ would use the equations

$$\begin{aligned}
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{d}_{x^2}\boldsymbol{s}) &= X_{\mathrm{QC}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) + \frac{\alpha}{q}X_{\mathrm{PQ}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s})^{(1)} \\
&+ \frac{1}{2q}\Big((\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) - \frac{\alpha}{q}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})^{(1)}\Big) \\
&+ \frac{2\alpha}{2pq}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s})^{(1)} , \\
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s}) &= X_{\mathrm{QC}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss}) + \frac{\alpha}{q}X_{\mathrm{PQ}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})^{(1)} \\
&+ \frac{2\alpha}{2pq}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(1)} ,
\end{aligned}$$

$$\begin{aligned}
(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s})^{(1)} &= X_{\mathrm{QC}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})^{(1)} + \frac{\alpha}{q}X_{\mathrm{PQ}}(\boldsymbol{d}_{x^2}\boldsymbol{s}|\boldsymbol{ss})^{(2)} \\
&+ \frac{2\alpha}{2pq}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(2)} , \\
(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{p}_x\boldsymbol{s})^{(1)} &= X_{\mathrm{QC}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(1)} + \frac{\alpha}{q}X_{\mathrm{PQ}}(\boldsymbol{p}_x\boldsymbol{s}|\boldsymbol{ss})^{(2)} \\
&+ \frac{\alpha}{2pq}(\boldsymbol{ss}|\boldsymbol{ss})^{(2)} . \qquad (1.4.20)
\end{aligned}$$

The OS method can be developed for ERIs over Hermite Gaussians as well [46], advantageous for the evaluation of the derivatives of the integrals. If we substitute the Cartesian Gaussians in Eq. (1.1.15) by Hermite ones, move the derivative operators and the variables independent from the electron coordinates outside of the integrals, and apply Eq. (1.1.16) we get

$$(\tilde{\boldsymbol{L}}_a\tilde{\boldsymbol{L}}_b|\tilde{\boldsymbol{L}}_c\tilde{\boldsymbol{L}}_d) = \theta_{pq}\mathcal{D}_{\tilde{I}_a}\mathcal{D}_{\tilde{I}_b}\ldots\mathcal{D}_{\tilde{K}_c}\mathcal{D}_{\tilde{K}_d}\kappa_{ab}\kappa_{cd}F_0(\alpha R^2_{\mathrm{PQ}}) \qquad (1.4.21)$$

where, for example,

$$\mathcal{D}_{\tilde{I}_a} = \frac{\partial^{\tilde{I}_a}}{(2a\partial A_x)^{\tilde{I}_a}} . \qquad (1.4.22)$$

Similarly to the Cartesian case, the recursions to increment the three components of the angular momentum vectors will be independent, therefore, for ease of notation, consider auxiliary integrals with only the x components of the angular momenta not being zero, defined as

$$(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)}_x = \theta_{pq}\mathcal{D}_{\tilde{i}_a}\mathcal{D}_{\tilde{i}_b}\mathcal{D}_{\tilde{i}_c}\mathcal{D}_{\tilde{i}_d}\kappa_{ab}\kappa_{cd}F_n(\alpha R^2_{\mathrm{PQ}}) . \qquad (1.4.23)$$

Incrementing $\tilde{i}_a$ by one and performing a differentiation with respect to $\mathbf{A}$ we get

$$\begin{aligned}
([\tilde{\boldsymbol{l}}_a + \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)}_x &= \theta_{pq}\mathcal{D}_{\tilde{i}_a}\mathcal{D}_{\tilde{i}_b}\mathcal{D}_{\tilde{i}_c}\mathcal{D}_{\tilde{i}_d} \\
&\quad \Big[-\frac{2\mu}{2a}X_{\mathrm{AB}}F_n(\alpha R^2_{\mathrm{PQ}}) - 2\alpha X_{\mathrm{PQ}}\frac{a}{2ap}F_{n+1}(\alpha R^2_{\mathrm{PQ}})\Big]\kappa_{ab}\kappa_{cd} \\
&= \theta_{pq}\mathcal{D}_{\tilde{i}_a}\mathcal{D}_{\tilde{i}_b}\mathcal{D}_{\tilde{i}_c}\mathcal{D}_{\tilde{i}_d} \\
&\quad \Big[X_{\mathrm{PA}}F_n(\alpha R^2_{\mathrm{PQ}}) - X_{\mathrm{PQ}}\frac{\alpha}{p}F_{n+1}(\alpha R^2_{\mathrm{PQ}})\Big]\kappa_{ab}\kappa_{cd} , \qquad (1.4.24)
\end{aligned}$$

where the first and second terms in the bracket come from $\partial/(2a\partial A_x)$ acting on κ_{ab} and $F_n(\alpha R^2_{\mathrm{PQ}})$ [see Eq. (1.1.18)], respectively. In the first term Eq. (1.2.14) was also utilized. For us to be able to invoke Eq. (1.4.23) and derive a recurrence that connects ERIs with higher angular momenta to those with lower ones we need to move the $\mathcal{D}$ operators beyond X_{PA} and X_{PQ}. To achieve this we need the commutator

$$\Big[X_{\mathrm{PA}}, \mathcal{D}_{\tilde{i}_a}\Big] = X_{\mathrm{PA}}\mathcal{D}_{\tilde{i}_a} - \mathcal{D}_{\tilde{i}_a}X_{\mathrm{PA}} \qquad (1.4.25)$$

and its analogue for X_{PQ}. It can be seen by induction that these commutators are

$$\Big[X_{\mathrm{PA}}, \mathcal{D}_{\tilde{i}_a}\Big] = \frac{\tilde{i}_a}{2p}\frac{b}{a}\mathcal{D}_{\tilde{i}_a-\tilde{1}} \qquad \Big[X_{\mathrm{PQ}}, \mathcal{D}_{\tilde{i}_a}\Big] = -\frac{\tilde{i}_a}{2p}\mathcal{D}_{\tilde{i}_a-\tilde{1}} \tag{1.4.26}$$

which are inserted into Eq. (1.4.24) to give

$$\begin{aligned}
([\tilde{\boldsymbol{l}}_a + \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)}_x &= \theta_{pq}\mathcal{D}_{\tilde{i}_b}\mathcal{D}_{\tilde{i}_c}\mathcal{D}_{\tilde{i}_d} \\
&\quad \Big[\Big(X_{\mathrm{PA}}\mathcal{D}_{\tilde{i}_a} - \frac{\tilde{i}_a}{2p}\frac{b}{a}\mathcal{D}_{\tilde{i}_a-\tilde{1}}\Big)F_n(\alpha R^2_{\mathrm{PQ}}) \\
&- \frac{\alpha}{p}\Big(X_{\mathrm{PQ}}\mathcal{D}_{\tilde{i}_a} + \frac{\tilde{i}_a}{2p}\mathcal{D}_{\tilde{i}_a-\tilde{1}}\Big)F_{n+1}(\alpha R^2_{\mathrm{PQ}})\Big]\kappa_{ab}\kappa_{cd} \\
&= \theta_{pq}\mathcal{D}_{\tilde{i}_b}\mathcal{D}_{\tilde{i}_c}\mathcal{D}_{\tilde{i}_d} \\
&\quad \Big(X_{\mathrm{PA}}\mathcal{D}_{\tilde{i}_a}F_n(\alpha R^2_{\mathrm{PQ}}) - \frac{\alpha}{p}X_{\mathrm{PQ}}\mathcal{D}_{\tilde{i}_a}F_{n+1}(\alpha R^2_{\mathrm{PQ}})\Big)\kappa_{ab}\kappa_{cd} \\
&+ \frac{\tilde{i}_a}{2p}\Big(-\frac{b}{a}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)}_x - \frac{\alpha}{p}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n+1)}_x\Big)\ .
\end{aligned} \tag{1.4.27}$$

In the last two terms it was utilized that $(\tilde{i}_a/2p)(b/a)$ and $(\alpha/p)(\tilde{i}_a/2p)$ commute with the $\mathcal{D}$ operators, and thus Eq. (1.4.23) can be substituted into the terms containing these factors. Using the commutators of $\mathcal{D}_{\tilde{i}_b}$, $\mathcal{D}_{\tilde{i}_c}$, and $\mathcal{D}_{\tilde{i}_d}$ with X_{PA}, X_{QC}, and X_{PQ} the other $\mathcal{D}$ operators can also be moved after the nuclear coordinate dependent factors. Since the y- and z-dependent differentiations commute with all the factors that appear before the auxiliary integrals, we can return to the ERIs with general angular momentum vectors to obtain the VRR for Hermite Gaussian ERIs:

$$\begin{aligned}
([\tilde{\boldsymbol{l}}_a + \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)} &= X_{\mathrm{PA}}(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)} - \frac{\alpha}{p}X_{\mathrm{PQ}}(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n+1)} \\
&+ \frac{\tilde{i}_a}{2p}\Big(-\frac{b}{a}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)} - \frac{\alpha}{p}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n+1)}\Big) \\
&+ \frac{\tilde{i}_b}{2p}\Big((\tilde{\boldsymbol{l}}_a[\tilde{\boldsymbol{l}}_b - \tilde{\mathbf{1}}_x]|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n)} - \frac{\alpha}{p}(\tilde{\boldsymbol{l}}_a[\tilde{\boldsymbol{l}}_b - \tilde{\mathbf{1}}_x]|\tilde{\boldsymbol{l}}_c\tilde{\boldsymbol{l}}_d)^{(n+1)}\Big) \\
&+ \frac{\alpha}{2pq}\Big(\tilde{i}_c(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|[\tilde{\boldsymbol{l}}_c - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_d)^{(n+1)} + \tilde{i}_d(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{l}}_c[\tilde{\boldsymbol{l}}_d - \tilde{\mathbf{1}}_x])^{(n+1)}\Big)\ .
\end{aligned} \tag{1.4.28}$$

The resulting equation is very similar to the one applicable to Cartesian Gaussians, Eq. (1.4.11). Note that Eq. (1.4.28) is not the same as the one presented in Ref. 46 since the auxiliary integrals defined in Eq. (1.4.23) are slightly different. It is adequate to mention here that the recurrence for the $\tilde{i}_a \to \tilde{i}_b$ translation is not the same as for $i_a \to i_b$ given by Eq. (1.2.26). The corresponding equation can be derived using Eqs. (1.1.9) and (1.1.14) as

$$\begin{aligned}
\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b+\tilde{1}} &= x_{\mathrm{B}}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_b}{2b}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b-\tilde{1}} = (x_{\mathrm{A}} + X_{\mathrm{AB}})\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_b}{2b}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b-\tilde{1}} \\
&= \tilde{H}_{\tilde{i}_a+\tilde{1}}\tilde{H}_{\tilde{i}_b} + X_{\mathrm{AB}}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b} + \frac{\tilde{i}_a}{2a}\tilde{H}_{\tilde{i}_a-\tilde{1}}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_b}{2b}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b-\tilde{1}}
\end{aligned} \tag{1.4.29}$$

The ETR for Hermite Gaussian integrals, differing from Eq. (1.4.17) only in a multiplication by $-b/a$ and $-d/c$ for the third and fourth terms, respectively, can be derived in a manner similar to the one for Cartesian ERIs [46].

The use of the VRR and ETR for Hermite Gaussian ERIs is analogous to the examples shown in Eqs. (1.4.18), (1.4.19), and (1.4.20). To illustrate the application of Eq. (1.4.29) consider the buildup of $\tilde{\boldsymbol{L}}_b$ for the class $(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2})$ as

$$\begin{aligned}
(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) &= (\tilde{\boldsymbol{f}}_{x^3}\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) + X_{\mathrm{AB}}(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) \\
&+ \frac{2}{2a}(\tilde{\boldsymbol{p}}_x\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) - \frac{1}{2b}(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2})\ , \\
(\tilde{\boldsymbol{f}}_{x^3}\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) &= (\tilde{\boldsymbol{g}}_{x^4}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) + X_{\mathrm{AB}}(\tilde{\boldsymbol{f}}_{x^3}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) \\
&+ \frac{3}{2a}(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2})\ , \\
(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) &= (\tilde{\boldsymbol{f}}_{x^3}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) + X_{\mathrm{AB}}(\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) \\
&+ \frac{2}{2a}(\tilde{\boldsymbol{p}}_x\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2})\ , \\
(\tilde{\boldsymbol{p}}_x\tilde{\boldsymbol{p}}_x|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) &= (\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) + X_{\mathrm{AB}}(\tilde{\boldsymbol{p}}_x\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2}) \\
&+ \frac{1}{2a}(\tilde{\boldsymbol{s}}\tilde{\boldsymbol{s}}|\tilde{\boldsymbol{d}}_{x^2}\tilde{\boldsymbol{d}}_{x^2})\ .
\end{aligned} \tag{1.4.30}$$

Comparing Eqs. (1.2.26) and (1.4.29), and also Eqs. (1.1.20) and (1.1.21) it is not trivial to predict whether Cartesian or Hermite Gaussians are the basis functions of choice for the evaluation of the first derivatives of ERIs with the OS scheme.

1.5 Rys polynomial method

The algorithms discussed before are all based on calculating scaled Boys functions of various order and using them as starting values for a recursive procedure. Inspecting these methods and utilizing Eq. (1.1.17) it is evident that the target integral can be expressed as

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d) = \sum_{n=0}^{L_a+L_b+L_c+L_d} Z_n F_n(\alpha R_{\mathrm{PQ}}^2) = \int_0^1 \sum_{n=0}^{L_a+L_b+L_c+L_d} Z_n t^{2n}\exp(-\alpha R_{\mathrm{PQ}}^2 t^2)\mathrm{d}t\ , \tag{1.5.1}$$

where the values of the coefficients Z_n can be obtained, for example, by backtracking the OS recursions until the integral is only expanded in Boys functions. Eq. (1.5.1) is an integral over a polynomial $f(t^2) = \sum_{n=0}^{L_a+L_b+L_c+L_d} Z_n t^{2n}$ multiplied by a weight function $W(T,t^2) = \exp(-Tt^2)$ with $T = \alpha R_{\mathrm{PQ}}^2$. According to the theory of Gauss–Rys quadrature [34, 35, 62], such integrals can be exactly evaluated as

$$\int_0^1 f(t^2)W(T,t^2)\mathrm{d}t = \sum_{r=1}^{N_{rts}} f(t_r^2)w_r \tag{1.5.2}$$

with N_{rts} being an integer satisfying $N_{rts} > \lceil (L_a+L_b+L_c+L_d)/2 \rfloor$ where $\lceil x \rfloor$ is the integer part of x, and t_r is the rth positive root of the $2N_{rts}$th Rys polynomial in t.

These polynomials are defined to be orthonormal on the interval [0,1] with the weight function $W(T,t^2)$. w_r is the T-dependent weight factor of the quadrature associated with quadrature point t_r. The calculation of the roots of the appropriate Rys polynomials and the weight factors will not be discussed here; several approaches [34, 62–67] can be found in the literature. The computational expense of these calculations is of the same magnitude as that of the Boys function evaluations for the other methods.

The polynomial $f(t^2)$ appearing in Eq. (1.5.2) can be written as products of two-dimensional (2D) integrals associated with the three Cartesian directions, which are also polynomials in t^2 [35, 68], and which are computable by a recursive method. One way to show this and to derive the desired recurrence relations [35] is to utilize that, from Eqs. (1.2.3) and (1.2.4), the final four-center ERIs can be computed from one-center integrals over unscaled Hermite Gaussians. Invoking Eq. (1.1.17) it is possible to express these latter integrals as

$$\begin{aligned}(\bar{\boldsymbol{l}}_u) &= \int_0^1 \Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_u}\Big(\frac{\partial}{\partial P_y}\Big)^{\bar{j}_u}\Big(\frac{\partial}{\partial P_z}\Big)^{\bar{k}_u} W(T,t^2)\mathrm{d}t \\ &= \int_0^1 \mathcal{H}_{\bar{i}_u}(t^2)\mathcal{H}_{\bar{j}_u}(t^2)\mathcal{H}_{\bar{k}_u}(t^2)W(T,t^2)\mathrm{d}t \ , \end{aligned} \tag{1.5.3}$$

where the auxiliary polynomials

$$\mathcal{H}_{\bar{i}_u}(t^2) = \exp(\alpha X_{\mathrm{PQ}}^2 t^2)\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_u} \exp(-\alpha X_{\mathrm{PQ}}^2 t^2) \tag{1.5.4}$$

were applied. The integral in (1.5.3) is in a suitable form for the application of the Gauss–Rys quadrature. From Eqs. (1.2.4), (1.2.3), (1.5.2), and (1.5.3) we can write our target ERI as

$$\begin{aligned}(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d) &= \int_0^1 \Theta_x^{I_a,I_b,I_c,I_d}(t^2)\Theta_y^{J_a,J_b,J_c,J_d}(t^2)\Theta_z^{K_a,K_b,K_c,K_d}(t^2)W(T,t^2)\mathrm{d}t \\ &= \sum_{r=1}^{N_{rts}} \Theta_x^{I_a,I_b,I_c,I_d}(t_r^2)\Theta_y^{J_a,J_b,J_c,J_d}(t_r^2)\Theta_z^{K_a,K_b,K_c,K_d}(t_r^2)w_r \end{aligned} \tag{1.5.5}$$

with

$$\Theta_x^{I_a,I_b,I_c,I_d}(t^2) = \sum_{\bar{i}_p=0}^{I_a+I_b}\sum_{\bar{i}_q=0}^{I_c+I_d}(-1)^{\bar{i}_q}E_{\bar{i}_p}^{I_a,I_b}E_{\bar{i}_q}^{I_c,I_d}\mathcal{H}_{\bar{i}_p+\bar{i}_q}(t^2) \ . \tag{1.5.6}$$

The quantities defined by Eq. (1.5.6) will be referred to as 2D integrals since they can be shown [68] to be scaled versions of the integral

$$\begin{aligned}\Xi_x^{I_a,I_b,I_c,I_d}(t^2) &= \int\int x_{\mathrm{A}}^{I_a}x_{\mathrm{B}}^{I_b}x_{\mathrm{C}}^{I_c}x_{\mathrm{D}}^{I_d} \\ &\times \ \exp[-\mu X_{\mathrm{AB}} - \nu X_{\mathrm{CD}} - p x_{\mathrm{P}}^2 - q x_{\mathrm{Q}}^2 - t^2|x_1-x_2|^2]\ \mathrm{d}x_1\ \mathrm{d}x_2 \ . \end{aligned} \tag{1.5.7}$$

To obtain the equation which builds up $\Theta_x^{I_a,I_b,I_c,I_d}(t^2)$ we also need the recurrence connecting the auxiliary polynomials of different order. Incrementing $\bar{i}_u$ in Eq. (1.5.4) we get

$$\begin{aligned}\mathcal{H}_{\bar{i}_u+\bar{1}}(t^2) &= \exp(\alpha X_{\mathrm{PQ}}^2 t^2)\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_u+\bar{1}}\exp(-\alpha X_{\mathrm{PQ}}^2 t^2)\\ &= -2\alpha t^2 \exp(\alpha X_{\mathrm{PQ}}^2 t^2)\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_u} X_{\mathrm{PQ}}\exp(-\alpha X_{\mathrm{PQ}}^2 t^2)\\ &= -2\alpha t^2[X_{\mathrm{PQ}}\mathcal{H}_{\bar{i}_u}(t^2) + \bar{i}_u\mathcal{H}_{\bar{i}_u-\bar{1}}(t^2)]\ , \qquad (1.5.8)\end{aligned}$$

where the commutator $[(\partial/\partial P_x)^{\bar{i}_u}, X_{\mathrm{PQ}}] = \bar{i}_u(\partial/\partial P_x)^{\bar{i}_u-\bar{1}}$ was exploited. With Eqs. (1.2.19) and (1.5.8) the recursion for the 2D integrals is derived in an almost analogous manner to the VRR for the ERIs, Eq. (1.4.11). From Eq. (1.5.6) we can write

$$\Theta_x^{i_a+1,i_b,i_c,i_d}(t^2) = \sum_{\bar{i}_p=0}^{i_a+i_b+1}\sum_{\bar{i}_q=0}^{i_c+i_d}(-1)^{\bar{i}_q}E_{\bar{i}_p}^{i_a+1,i_b}E_{\bar{i}_q}^{i_c,i_d}\mathcal{H}_{\bar{i}_p+\bar{i}_q}(t^2)\ . \qquad (1.5.9)$$

Substituting Eq. (1.2.19) and rewriting $\bar{i}_p$ to $\bar{i}_p+1$ in the last term similarly as it was done for Eq. (1.4.7) gives us

$$\begin{aligned}\Theta_x^{i_a+1,i_b,i_c,i_d}(t^2) &= X_{\mathrm{PA}}\Theta_x^{i_a,i_b,i_c,i_d}(t^2) + \frac{1}{2p}[i_a\Theta_x^{i_a-1,i_b,i_c,i_d}(t^2) + i_b\Theta_x^{i_a,i_b-1,i_c,i_d}(t^2)]\\ &+ \frac{1}{2p}\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}(-1)^{\bar{i}_q}E_{\bar{i}_p}^{i_a,i_b}E_{\bar{i}_q}^{i_c,i_d}\mathcal{H}_{\bar{i}_p+\bar{i}_q+\bar{1}}(t^2)\ , \qquad (1.5.10)\end{aligned}$$

into which we can insert Eq. (1.5.8) to obtain

$$\begin{aligned}\Theta_x^{i_a+1,i_b,i_c,i_d}(t^2) &= X_{\mathrm{PA}}\Theta_x^{i_a,i_b,i_c,i_d}(t^2) + \frac{1}{2p}[i_a\Theta_x^{i_a-1,i_b,i_c,i_d}(t^2) + i_b\Theta_x^{i_a,i_b-1,i_c,i_d}(t^2)]\\ &- \frac{\alpha}{p}t^2 X_{\mathrm{PQ}}\Theta_x^{i_a,i_b,i_c,i_d}(t^2)\\ &- \frac{\alpha}{p}t^2\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}(-1)^{\bar{i}_q}(\bar{i}_pE_{\bar{i}_p}^{i_a,i_b})E_{\bar{i}_q}^{i_c,i_d}\mathcal{H}_{\bar{i}_p+\bar{i}_q-\bar{1}}(t^2)\\ &- \frac{\alpha}{p}t^2\sum_{\bar{i}_p=0}^{i_a+i_b}\sum_{\bar{i}_q=0}^{i_c+i_d}(-1)^{\bar{i}_q}E_{\bar{i}_p}^{i_a,i_b}(\bar{i}_qE_{\bar{i}_q}^{i_c,i_d})\mathcal{H}_{\bar{i}_p+\bar{i}_q-\bar{1}}(t^2)\ . \qquad (1.5.11)\end{aligned}$$

Finally, after using Eq. (1.2.18) and collecting terms we arrive at

$$\begin{aligned}\Theta_x^{i_a+1,i_b,i_c,i_d}(t^2) &= \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}t^2X_{\mathrm{PQ}}\Big)\Theta_x^{i_a,i_b,i_c,i_d}(t^2)\\ &+ \frac{1}{2p}\Big(1-\frac{\alpha}{p}t^2\Big)[i_a\Theta_x^{i_a-1,i_b,i_c,i_d}(t^2) + i_b\Theta_x^{i_a,i_b-1,i_c,i_d}(t^2)]\\ &+ \frac{\alpha}{2pq}t^2[i_c\Theta_x^{i_a,i_b,i_c-1,i_d}(t^2) + i_d\Theta_x^{i_a,i_b,i_c,i_d-1}(t^2)]\ . \qquad (1.5.12)\end{aligned}$$

Note that Eq. (1.5.12) has a similar form to Eq. (1.4.11) with the terms multiplied by t^2 being in the place of the auxiliary ERIs with incremented n index. The starting values for the recursion defined by Eq. (1.5.12) are

$$\Theta_x^{0,0,0,0} = \kappa_{ab}^x \kappa_{cd}^x \tag{1.5.13}$$

for every t^2. After working through the same procedure to increment i_b we obtain an analogous recursion with X_{PB} instead of X_{PA}. Subtracting this from Eq. (1.5.12) we get

$$\Theta_x^{i_a,i_b+1,i_c,i_d}(t^2) = \Theta_x^{i_a+1,i_b,i_c,i_d}(t^2) + X_{\mathrm{AB}}\Theta_x^{i_a,i_b,i_c,i_d}(t^2) \ , \tag{1.5.14}$$

resembling Eq. (1.2.26). The recursion to build up i_c is similar to Eq. (1.5.12) in the same way as Eq. (1.4.11) is analogous to Eq. (1.4.12).

The application of Eqs. (1.5.5), (1.5.12), and (1.5.14) can be demonstrated on the evaluation of the ERI $(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z)$. At least 3 quadrature points are required for an integral with a total angular momentum of 4, so the quadrature has the form of

$$(\boldsymbol{p}_x\boldsymbol{p}_x|\boldsymbol{p}_y\boldsymbol{p}_z) = \sum_{r=1}^{3} \Theta_x^{1,1,0,0}(t_r^2)\Theta_y^{0,0,1,0}(t_r^2)\Theta_z^{0,0,0,1}(t_r^2) w_r \ , \tag{1.5.15}$$

and the $\Theta_x^{1,1,0,0}(t_1^2)$ 2D integral, for example, is computed as

$$\begin{aligned}
\Theta_x^{1,1,0,0}(t_1^2) &= \Theta_x^{2,0,0,0}(t_1^2) + X_{\mathrm{AB}}\Theta_x^{1,0,0,0}(t_1^2) \ , \\
\Theta_x^{2,0,0,0}(t_1^2) &= \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}t_1^2\Big)\Theta_x^{1,0,0,0}(t_1^2) + \frac{1}{2p}\Big(1 - \frac{\alpha}{p}t_1^2\Big)\Theta_x^{0,0,0,0} \ , \\
\Theta_x^{1,0,0,0}(t_1^2) &= \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}t_1^2\Big)\Theta_x^{0,0,0,0} \ .
\end{aligned} \tag{1.5.16}$$

Concerning the efficient application of the Rys scheme to ERIs two aspects should be mentioned. First, one of the multiplications inside the sum in Eq. (1.5.5) can be spared if the w_r weight factor multiplies the starting 2D integral for one of the directions before the recursion takes place [68]. Second, it can be utilized that, when Eq. (1.2.26) is exploited, a range of $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c\boldsymbol{0})$ type intermediates has to be calculated. For example, to evaluate the $(\boldsymbol{dp}|\boldsymbol{ss})$ class by Eq. (1.2.26) we need both the $(\boldsymbol{f}_{xyz}\boldsymbol{s}|\boldsymbol{ss})$ and $(\boldsymbol{d}_{xy}\boldsymbol{s}|\boldsymbol{ss})$ ERIs. The products $\Theta_x^{1,0,0,0}(t_r^2)\Theta_y^{1,0,0,0}(t_r^2)$ can be thus reused during the evaluation of these integrals by Eq. (1.5.5). Here it is utilized that the number of the required quadrature points to calculate an ERI is at least $\lceil (l_a + l_b + l_c + l_d)/2 \rfloor + 1$, but Rys polynomials of higher order can also be used [35], that is, the $(\boldsymbol{f}_{xyz}\boldsymbol{s}|\boldsymbol{ss})$ and $(\boldsymbol{d}_{xy}\boldsymbol{s}|\boldsymbol{ss})$ ERIs can be computed on the same quadrature. This strategy is referred to as the reduced multiplication scheme [60].

For the differentiated ERIs, techniques have been developed to directly evaluate the 2D integral derivatives, from which the ERI derivatives can be computed by Eq. (1.5.5). One approach [64] utilizes that the only nuclear coordinate dependent parts of the 2D integrals are the G_I, G_J, or G_K Cartesian Gaussians (note that in Eq. (1.5.7) the Gaussian product rule was already applied), hence the differentiation rule they obey is completely analogous to Eq. (1.1.4). It has also been noted [69] that the derivatives of the 2D integrals can be expressed without reference to integrals over functions of higher angular momenta, though the computational cost of the required relation is the same as evaluating the 2D integral with the incremented angular momentum by Eq. (1.5.12). Recurrences to compute 2D integrals over Hermite Gaussian basis functions have not yet been presented in the literature. Such equations will be derived in a later chapter.

1.6 Integral prescreening

Formally, the computation of the $(\chi_A\chi_B|\chi_C\chi_D)$ ERIs (and their derivatives) has an $\mathcal{O}(N^4)$ scaling, where N is the number of AOs. It was noticed long ago [70, 71], however, that a huge portion of these integrals, and most of them for large systems, can be discarded due to their small magnitude. For systems of sufficient size [72] the scaling becomes close to $\mathcal{O}(N^2)$ when only significant ERIs are considered. There are two reasons why an integral might have a magnitude that is not worth including in the calculation [73]. First, if the overlap between χ_A and χ_B (or χ_C and χ_D) is small enough, the value of the ERI can get close to zero. Second, even if both the overlaps in the bra and the ket are significant, the interaction energy between them will be small if the two charge distributions fall far away from each other.

With the technique of integral prescreening certain estimates or exact upper bounds are applied to decide if an integral (or, more efficiently, the largest ERI in a batch of integrals) will have a higher magnitude than a user defined threshold. The first rigorous upper bound for ERIs was presented by Ahlrichs [74] for integrals that are over $\boldsymbol{s}$-type Gaussian functions. His approach was also useful for the screening of integrals involving $\boldsymbol{p}$- or $\boldsymbol{d}$-type functions [75], even though for these cases the bound is not exact. The most widely used upper bound is based on the realization that the operation between the charge distributions in the bra and the ket defined by the Coulomb-integral is formally a scalar product, and therefore obeys the Schwarz inequality [12, 15, 35, 76]. This implies the relation

$$|(\Psi_t\Psi_u|\Psi_v\Psi_w)| \leq Q_{tu}\, Q_{vw} \ , \tag{1.6.1}$$

where Ψ is an arbitrary function, and

$$Q_{tu} = \sqrt{(\Psi_t\Psi_u|\Psi_t\Psi_u)} \ . \tag{1.6.2}$$

Eq. (1.6.1) has the merits that it can provide bounds for ERIs over both AOs and primitive Gaussians, and that the calculation of Q_{tu} scales as $\mathcal{O}(N^2)$. The inequality in Eq. (1.6.1) becomes an identity if $t = v$ and $u = w$. It has been noted that for other cases the Schwarz bound can significantly overestimate the integral in question, and tighter bounds have been proposed [77, 78]. Yet, the application of the Schwarz bound remains a standard approach [35, 73], at least for shell quartets over AOs, because the Q_{tu} quantities can be computed in the same way as the final ERIs. The Schwarz bound does not exploit the fact that the electron-electron interaction weakens with the growth of the distance between the centers of the charge distributions. Approaches that account for this effect were previously based on the multipole expansion of the Coulomb operator [79–81], but later the attention in this field has shifted towards the empirical modification of the Schwarz bound, referred to as the QQR estimator [82, 83].

There exist other non-rigorous estimates that are much simpler to calculate. Almlöf and co-workers argued [11] that the magnitude of an ERI over Cartesian Gaussians is largely determined by a factor that multiplies every integral in a given class, written as

$$M_{L_a,L_b,L_c,L_d} = S_{L_a,L_b} S_{L_c,L_d} \frac{pq}{\pi(p+q)} \ , \tag{1.6.3}$$

where S_{L_a,L_b} and S_{L_c,L_d} are the radial overlaps of the functions in the bra and the ket, respectively. Another useful estimate is based on the assumption that

$$|(\mathbf{00}|\mathbf{00})| \geq |(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c\boldsymbol{L}_d)| \tag{1.6.4}$$

for every member of a class of Cartesian Gaussian ERIs [64]. It should be noted that Ref. 64 states that the inequality in Eq. (1.6.4) defines a rigorous upper bound. It can be demonstrated by simple numerical experiments that this claim is not true.

The screening of the integral derivatives was discussed by Ahlrichs and co-workers in the context of the Schwarz inequality [84]. An upper bound for first derivatives is given as

$$\left|\left(\frac{\partial \Psi_t}{\partial A_x}\Psi_u|\Psi_v\Psi_w\right)\right| \leq Q_{\partial tu}\, Q_{vw} \ , \tag{1.6.5}$$

where

$$Q_{\partial tu} = \sqrt{\left(\frac{\partial \Psi_t}{\partial A_x}\Psi_u|\frac{\partial \Psi_t}{\partial A_x}\Psi_u\right)} \ . \tag{1.6.6}$$

Prescreening for the derivative ERIs via the Schwarz bound is a less trivial matter than for the undifferentiated integrals since Eq. (1.6.6) requires the computation of the relatively expensive ERI second-derivatives. Non-rigorous estimates such as the ones given by Eqs. (1.6.3) and (1.6.4) are easily adapted for differentiated ERIs by the application of Eq. (1.1.4).

1.7 Density fitting

Another important cost reduction technique for the computation of four-center ERIs is the DF approximation. The core idea of the method is to fit orbital products to a basis of auxiliary functions (AFs) as

$$\chi_p\chi_q \approx \widetilde{\chi_p\chi_q} = \sum_P C_{pq}^P \rho_P \ , \tag{1.7.1}$$

where C_{pq}^{P} is an expansion coefficient and ρ_P is a function of the auxiliary basis [note that the subscript of the AOs in Eq. (1.7.1) is just a general index, and it does not refer to the center of the function as before]. This type of expansion of function products was probably first used by Boys and Shavitt in a calculation of the potential energy surface of H_3 [14, 25]. The most widely used procedure to obtain the fitting coefficients is to use the Coulomb-metric, that is, to minimize the Coulomb energy of the residual function $\chi_p\chi_q - \widetilde{\chi_p\chi_q}$ [15, 85]. This means finding the coefficients with which the ERI

$$\Delta_{pq} = (\chi_p\chi_q - \widetilde{\chi_p\chi_q}|\chi_p\chi_q - \widetilde{\chi_p\chi_q}) \tag{1.7.2}$$

has its minimal value. Such a method also minimizes the discrepancy between the electric fields of the original and the fitted orbital products [85]. Inserting Eq. (1.7.1) into Eq. (1.7.2) and setting $\partial\Delta_{pq}/\partial C_{pq}^{P} = 0$ we can write

$$(\chi_p\chi_q|\rho_R) = \sum_P C_{pq}^{P} V_{PR} \ , \tag{1.7.3}$$

where $\mathbf{V}$ is the matrix containing the $(\rho_P|\rho_R)$ type two-center ERIs. The coefficients can then be expressed as

$$C_{pq}^{P} = \sum_R (\chi_p\chi_q|\rho_R) V_{RP}^{-1} \ . \tag{1.7.4}$$

Thus the approximate ERI is written as

$$(\widetilde{\chi_p\chi_q}|\widetilde{\chi_r\chi_s}) = \sum_{P,R} C_{pq}^{P} V_{PR} C_{rs}^{R} = \sum_{P,R} (\chi_p\chi_q|\rho_P) V_{PR}^{-1} (\chi_r\chi_s|\rho_R) \ . \tag{1.7.5}$$

The corresponding formula for the ERI derivatives is obtained by straightforward differentiation of Eq. (1.7.5):

$$\begin{aligned} \frac{\partial}{\partial S_x}(\chi_p\chi_q|\chi_r\chi_s) &\approx \sum_{P,R} \left[\frac{\partial}{\partial S_x}(\chi_p\chi_q|\rho_P)\right] V_{PR}^{-1} (\chi_r\chi_s|\rho_R) \\ &+ \sum_{P,R} (\chi_p\chi_q|\rho_P) V_{PR}^{-1} \left[\frac{\partial}{\partial S_x}(\chi_r\chi_s|\rho_R)\right] \\ &+ \sum_{P,R} (\chi_p\chi_q|\rho_P) \left[\frac{\partial}{\partial S_x} V_{PR}^{-1}\right] (\chi_r\chi_s|\rho_R) \ . \end{aligned} \tag{1.7.6}$$

The evaluation of the three-center Coulomb-integrals is considerably simpler than that of their four-center counterparts. In turn, the three-center ERI list has to be evaluated more than once for the efficient application of Eq. (1.7.5) in practical cases (e.g., a Hartree–Fock calculation, see Chapter 4). It should be noted that other C_{pq}^{P} coefficients can also be determined by the minimization of other types of residual integrals instead of Eq. (1.7.2). For example, minimizing the self overlap integral of the $\chi_p\chi_q - \widetilde{\chi_p\chi_q}$ residual function [86] yields an expression with three-center overlap integrals and two-center ERIs. However, the resulting formula is less accurate [87] and thus requires the use of larger auxiliary basis sets. When the Coulomb-metric is applied, the rate-determining step is the computation of three-center quantities. It has been established that the error introduced by the DF approximation is usually negligible [13, 25, 88, 89] if an appropriate auxiliary basis is employed, such as the basis sets developed by Weigend *et al.* [88, 90, 91] and also by Hellweg and Rappoport [92].

The most notable result concerning the efficient evaluation of three-center ERIs belongs to Ahlrichs [93], who simplified the OS recursion

$$\begin{aligned}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c+\mathbf{1}_x)^{(n)} &= \frac{\alpha}{c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} + \frac{\alpha}{2pc}i_a([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} \\ &+ \frac{i_c}{2c}\Big((\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c-\mathbf{1}_x)^{(n)} - \frac{\alpha}{c}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c-\mathbf{1}_x])^{(n+1)}\Big)\,. \qquad (1.7.7)\end{aligned}$$

Eq. (1.7.7) is just the straightforward adaptation of Eq. (1.4.12) to $\mathbf{Q}=\mathbf{C}$, $q=c$, and $i_b=i_d=0$. Based on the asymptotic decay of the three-center ERIs with $\mathbf{R}_{\mathrm{PC}}$ he showed that, in the case when the angular part of the AF is a solid harmonic, the OS recurrence to build up $\boldsymbol{l}_c$ becomes

$$(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c+\mathbf{1}_x)^{(n)} = \frac{\alpha}{c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} + \frac{\alpha}{2pc}i_a([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)}\,. \qquad (1.7.8)$$

Eq. (1.7.8) is an improvement over Eq. (1.7.7) in two ways: it contains fewer terms, and the n auxiliary index becomes redundant since it decreases every time $\boldsymbol{l}_c$ is increased. This means that only intermediate ERIs with $n=L_c$ are required for the application of Eq. (1.7.8) to the buildup of the final angular momentum on the AF.

1.8 Memory hierarchy of computers

The efficiency of computer algorithms is mainly determined by two factors. One is the operation count, which gives the amount of elementary operations the processor has to perform. The other is the latency, that is, how much time the processor has to wait until the data and the instructions necessary for the execution of a statement become available to it. The amount of latency depends on the type of the storage device which the information is fetched from. Different types of memories (static random access memory (SRAM), dynamic RAM (DRAM), flash memory, hard disk, etc.) can be accessed at different speeds, however, as the latency becomes smaller so does the storage capacity, while the cost of the device grows. Modern computers utilize a range of storage devices with different latencies and capacities, and this construct is usually referred to as the memory hierarchy. The information the CPU immediately uses resides in the SRAM-based register memory, which has almost no latency but can only store a few bytes of information. Modern processors feature a register file, which is an array of register memories, having about 1 kilobyte of total capacity. At the beginning of the execution of

Table 1.1: Typical storage capacities and latencies of the different members of the memory hierarchy up to the main memory [94]. The "cycles" unit refers to the CPU clock cycles.

Memory	Capacity (kilobytes)	Access time (cycles)
Register file	1	1
Level 1 cache	32	2-4
Level 2 cache	256	10
Level 3 cache	10^4	40
Main memory	10^7	200

a statement, the CPU first inquires for data and instructions from the CPU cache memory. This is also an SRAM-based storage device and contains copies of the information stored in the main memory. It is divided into separate levels, with three levels being usual nowadays. A higher level cache is physically further from the CPU, has higher latency, but higher capacity as well. Usually there is a separate first level CPU cache for both data and instructions, while higher levels are shared in this aspect. If the necessary information has a copy in the first level cache, we have a first level cache hit, and the data and instructions are copied into the register. When we have a first level cache miss instead, the CPU attempts to read the information from the next level of CPU cache. If it is there, then a cache line, meaning the requested data or instruction and also additional information stored at physically adjacent bytes, is copied into the first level cache. This cache line is usually 64 bytes, which can contain 8 double precision floating point variables. If the first level of the cache is full, then the bytes belonging to addresses that were referenced least recently are freed, meaning that their content is copied into the next level of cache, and the new cache line takes their place. If the next level of cache is also full, then its least recently referenced bytes are moved to the next level of cache, and so on. The next (and for our purposes, the last) level of the memory hierarchy is the main memory, consisting of DRAM storage. The cache line system described above allows for the compiler (and to an extent, the programmer) to exploit the so-called "locality of reference" of programs. This means that if there is an access to a memory address at a given time, it is very probable that a nearby location will be referenced in the near future. This can be facilitated, for example, by the use of dense arrays which our program accesses sequentially, and also by reusing memory addresses with the use of buffer arrays. Table 1.1 depicts the typical latencies and capacities of the members of the hierarchy. The difference between the speeds of the caches and the main memory makes it evident that the efficient use of the CPU cache is crucial in high-performance computing.

Chapter 2

Optimization of three-center integrals

This chapter is mainly based on Ref. 95 and it discusses the efficient evaluation of three-center ERIs. The computation of these integrals, required for Eq. (1.7.5), is optimized in two steps: first a FLOP counting is performed for the considered algorithms, then efficient implementations of the most promising schemes are compared based on their runtime performances for certain model systems. After the adaptation of the methods discussed in Chapter 1 to three-center Coulomb-integrals algorithmic considerations are presented, concerning the beneficial order of operations, optimal CPU cache memory usage, and prescreening options. This is followed by a discussion on the results of the FLOP counts, then the process of the implementation is detailed, and finally a recommendation for the optimal algorithms is presented based on the performance tests. Note that three-center ERIs can be viewed as four-center ones where the fourth function has an exponent equal to zero. Hence, in the following when quantities depending on q (such as α) are used, it is implied that $q = c$.

In this chapter (and also in the following one) index ranges will be presented for each recursion that show which intermediates have to be evaluated by the recurrence in question. These index ranges were determined on the basis of a method outlined in the Appendix. Even though the presentation suggests that the lower limits of the ranges can take negative values, for brevity we will not use the notation $\max(0, l)$ for these cases but assume that the angular momenta are greater than or equal to zero.

2.1 Three-center ERI evaluation algorithms

2.1.1 Obara–Saika method

An efficient application of the OS scheme to three-center ERIs was presented by Ahlrichs [93]. This approach will be referred to as OS1. The first step here is to evaluate the required auxiliary integrals

$$(\mathbf{00}|\mathbf{0})^{(n)} = \theta_{pc}\kappa_{ab}F_n(\alpha R_{\mathrm{PC}}^2) \tag{2.1.1}$$

for $L_c \leq n \leq L_a + L_b + L_c$. Then the three-center version of Eq. (1.4.13),

$$\begin{aligned}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)} - \frac{\alpha}{p}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n+1)} \\ &+ \frac{i_a}{2p}\Big(([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} - \frac{\alpha}{p}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n+1)}\Big) ,\end{aligned} \tag{2.1.2}$$

is used to increment the angular momentum of the first function on the bra side. With Eq. (2.1.2), the classes $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(L_c)}$ are calculated for $L_a - L_c \leq l_a \leq L_a + L_b$. Next, in the case where solid harmonic basis functions are supposed on the ket side, $\boldsymbol{l}_c$ can be built up by the two-term VRR [93], Eq. (1.7.8). This relation is used to produce $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)^{(0)}$ classes for $L_a \leq l_a \leq L_a + L_b$. The last step is to increment $\boldsymbol{l}_b$ using Eq. (1.2.26):

$$(\boldsymbol{l}_a[\boldsymbol{l}_b + \mathbf{1}_x]|\boldsymbol{L}_c) = ([\boldsymbol{l}_a + \mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{L}_c) + X_{\mathrm{AB}}(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{L}_c) . \tag{2.1.3}$$

Besides the above algorithm there are at least three other possibilities to get the target integrals with OS-type recursions. The first one, labeled as OS2, evaluates the same auxiliary integrals with Eq. (2.1.1) as in OS1, then applies the VRR to the ket side first as

$$(\mathbf{00}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} = \frac{\alpha}{c}X_{\mathrm{PC}}(\mathbf{00}|\boldsymbol{l}_c)^{(n+1)} \tag{2.1.4}$$

to construct the classes $(\mathbf{00}|\boldsymbol{l}_c)^{(n)}$ for $L_c - L_a - L_b \leq l_c \leq L_c$ and $L_c - l_c \leq n \leq L_a + L_b$. This is followed by building up the angular momentum of the first function on the bra side as

$$\begin{aligned}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n)} - \frac{\alpha}{p}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} \\ &+ \frac{i_a}{2p}\Big(([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n)} - \frac{\alpha}{p}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)}\Big) \\ &+ \frac{\alpha}{2pc}i_c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n+1)} ,\end{aligned} \tag{2.1.5}$$

to compute $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ for $L_a \leq l_a \leq L_a + L_b$, and finally the algorithm is finished with Eq. (2.1.3).

Apart from the VRR, another way to build up $\boldsymbol{l}_a$ or $\boldsymbol{l}_c$ is to use the ETR. For three-center ERIs, the ETR has the form of

$$
\begin{aligned}
([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) &= -\frac{b}{p}X_{\mathrm{AB}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c) + \frac{i_a}{2p}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \\
&- \frac{c}{p}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x]) + \frac{i_c}{2p}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x]) \qquad (2.1.6)
\end{aligned}
$$

for the $\boldsymbol{l}_c \rightarrow \boldsymbol{l}_a$ conversion, and

$$
\begin{aligned}
(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x]) &= -\frac{b}{c}X_{\mathrm{AB}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c) + \frac{i_a}{2c}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \\
&- \frac{p}{c}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \qquad (2.1.7)
\end{aligned}
$$

for the $\boldsymbol{l}_a \rightarrow \boldsymbol{l}_c$ transfer. Note that, in principle, Eq. (2.1.7) also contains a fourth term on the right-hand side, $i_c/2c(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x])$, but this term is canceled for the same reasons as discussed by Ahlrichs for the VRR [93] when transforming to the solid harmonic basis. This cancellation also takes place for the third and fourth terms in Eqs. (2.1.2) and (2.1.5), respectively, and the second term in Eq. (2.1.6), but only in the case when $\boldsymbol{L}_b = \mathbf{0}$ [95]. It should be noted that the numerical instability in the ETR associated with the addition $pX_{\mathrm{PA}}/(c+d) + X_{\mathrm{QC}}$ [61] does not appear here. This is because, in the absence of the fourth center, Eq. (2.1.3) only has to be applied to the bra side, reducing the aforementioned sum to $p/cX_{\mathrm{PA}} = -b/cX_{\mathrm{AB}}$. If we wish to build up the integrals necessary for Eq. (2.1.3) with Eq. (2.1.6), we cannot use Eq. (2.1.4) for the construction of the $(\mathbf{00}|\boldsymbol{l}_c)$ type classes, instead we have to employ the full vertical recurrence [93],

$$
\begin{aligned}
(\mathbf{00}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} &= -\frac{\alpha}{c}X_{\mathrm{PC}}(\mathbf{00}|\boldsymbol{l}_c)^{(n+1)} \\
&+ \frac{i_c}{2c}\Big((\mathbf{00}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n)} - \frac{\alpha}{c}(\mathbf{00}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n+1)}\Big)\ , \qquad (2.1.8)
\end{aligned}
$$

for the ket side. The terms corresponding to the ones in the big parentheses in Eq. (2.1.8) vanish in Eqs. (1.7.8) and (2.1.4) during the solid harmonic transformation [93] of the ket side; however, with Eq. (2.1.6) terms belonging to angular momenta other than l_c get built into the integrals to be transformed, and these will not cancel. The scheme where Eq. (2.1.8) is employed first to build up $\boldsymbol{l}_c$ and then Eq. (2.1.6) for $\boldsymbol{l}_a$ will be referred to as OS3. In this route, Eq. (2.1.1) is used first to calculate the $(\mathbf{00}|\mathbf{0})^{(n)}$ integrals for $[L_c \bmod 2] \le n \le L_a + L_b + L_c$, then Eq. (2.1.8) for the classes $(\mathbf{00}|\boldsymbol{l}_c)$ with $L_c - L_a - L_b \le l_c \le L_a + L_b + L_c$, thereafter Eq. (2.1.6) is applied to get the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes for $L_a \le l_a \le L_a + L_b$. Finally, in the algorithm denoted as OS4, $\boldsymbol{l}_a$ is built up by Eq. (2.1.2), and $\boldsymbol{l}_c$ is incremented by the ETR, Eq. (2.1.7). Here the necessary $(\mathbf{00}|\mathbf{0})^{(n)}$ integrals are in the range $0 \le n \le L_a + L_b + L_c$ and are used to calculate the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})$ classes for $L_a + L_b - L_c \le l_a \le L_a + L_b + L_c$.

2.1.2 McMurchie–Davidson method

In the three-center case it is beneficial to define the two-center integrals necessary for the MD scheme as [31, 46]

$$\begin{aligned}(\bar{\boldsymbol{l}}_p|\tilde{\boldsymbol{l}}_c) &= \theta_{pc}(2c)^{-\tilde{l}_c}\frac{\partial^{\bar{l}_p+\tilde{l}_c}F_0(\alpha R_{\mathrm{PC}}^2)}{\partial P_x^{\bar{i}_p}\partial P_y^{\bar{j}_p}\partial P_z^{\bar{k}_p}\partial C_x^{\tilde{i}_c}\partial C_y^{\tilde{j}_c}\partial C_z^{\tilde{k}_c}}\\ &= \theta_{pc}(-2c)^{-\tilde{l}_c}\frac{\partial^{\bar{l}_p+\tilde{l}_c}F_0(\alpha R_{\mathrm{PC}}^2)}{\partial P_x^{\bar{i}_p+\tilde{i}_c}\partial P_y^{\bar{j}_p+\tilde{j}_c}\partial P_z^{\bar{k}_p+\tilde{k}_c}} = (-2c)^{-\tilde{l}_c}(\bar{\boldsymbol{l}}_u) \qquad (2.1.9)\end{aligned}$$

with $\bar{\boldsymbol{l}}_u = \bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{l}}_c$. The scaling with $(2c)^{-\tilde{l}_c}$ is applied since in the ket a Hermite Gaussian is used defined by Eq. (1.1.8), which will allow us to transform the ket side into the solid harmonic Gaussian basis without the transformation into Cartesian Gaussians first [note that this is not necessary for $(\bar{\boldsymbol{l}}_p|$]. The one-center integrals on the rightmost of Eq. (2.1.9) can be computed by Eq. (1.2.5) with

$$(\bar{\mathbf{0}})^{(n)} = (-2\alpha)^n\kappa_{ab}\theta_{pc}(-2c)^{-\tilde{L}_c}F_n(\alpha R_{\mathrm{PC}}^2)\ . \qquad (2.1.10)$$

From the one-center integrals three-center ERIs with two Cartesian Gaussians in the bra and a Hermite Gaussian in the ket are evaluated as [46]

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c) = \sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b}\sum_{\bar{j}_p=0}^{J_a+J_b} E_{\bar{j}_p}^{J_a,J_b}\sum_{\bar{k}_p=0}^{K_a+K_b} E_{\bar{k}_p}^{K_a,K_b}(\bar{\boldsymbol{l}}_p+\tilde{\boldsymbol{L}}_c)\ . \qquad (2.1.11)$$

Eq. (2.1.11) is considerably simpler than Eq. (1.2.3) because no transformation is necessary for the ket side. The E coefficients are computed efficiently by the recurrences [35]

$$E_{\bar{0}}^{i_a+1,0} = X_{\mathrm{PA}}E_{\bar{0}}^{i_a,0} + E_{\bar{1}}^{i_a,0}\ , \qquad (2.1.12)$$

$$E_{\bar{0}}^{i_a,i_b+1} = X_{\mathrm{PB}}E_{\bar{0}}^{i_a,i_b} + E_{\bar{1}}^{i_a,i_b}\ , \qquad (2.1.13)$$

and Eq. (1.2.18) with $E_{\bar{0}}^{0,0} = 1$. Using this starting value instead of the one defined by Eq. (1.2.13) is possible because κ_{ab} is included in Eq. (2.1.10). This results in fewer FLOPs with a code generated implementation of the coefficients, and only one exponential computation is necessary for κ_{ab} contrary to calculating κ_{ab}^x, κ_{ab}^y, and κ_{ab}^z separately.

The expansion defined by Eq. (2.1.11) can be applied to produce various types of three-center ERIs. In the MD1 algorithm, for example, we get the $(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$ classes directly. First the expansion coefficients are computed; e.g., in the x direction $E_{\bar{i}_p}^{i_a,i_b}$ values are needed for $0 \leq i_a \leq L_a$, $0 \leq i_b \leq L_b$, and $0 \leq \bar{i}_p \leq i_a + i_b$. This is followed by the calculation of the $(\bar{\mathbf{0}})^{(n)}$ integrals for $\lceil\tilde{L}_c/2\rfloor + [\bar{L}_c \bmod 2] \leq n \leq L_a + L_b + \tilde{L}_c$. The one-center integrals $(\bar{\boldsymbol{l}}_u)$ for $\tilde{L}_c \leq \bar{l}_u \leq L_a + L_b + \tilde{L}_c$ are built up by Eq. (1.2.5), from which the target integrals are readily assembled by Eq. (2.1.11). The work done in this assembly step can be reduced by performing it at an earlier stage to construct intermediate classes.

In the MD2 scheme, the $(\boldsymbol{l}_a\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ classes for $L_a \leq l_a \leq L_a + L_b$ are evaluated with Eq. (2.1.11). Here the necessary expansion coefficients are in the range of $0 \leq i_a \leq L_a + L_b$, $i_b = 0$, and $0 \leq \bar{i}_p \leq i_a$, and the required one-center integrals are the same as in MD1. After the assembly, the final integrals are computed by Eq. (2.1.3). A third option (MD3) is to obtain the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(\tilde{L}_c)}$ type intermediates for $L_a - \tilde{L}_c \leq l_a \leq L_a + L_b$ with Eq. (2.1.11), then to build up $\tilde{\boldsymbol{l}}_c$ with Eq. (1.7.8), and to finish with Eq. (2.1.3). Here the $(-1)^{\tilde{L}_c}$ scaling factor is absent from Eq. (2.1.10), and the required $(\bar{\mathbf{0}})^{(n)}$ values are in the range of $\tilde{L}_c \leq n \leq L_a + L_b + \tilde{L}_c$ and used for calculating the $(\bar{\boldsymbol{l}}_u)^{(\tilde{L}_c)}$ integrals for $0 \leq l_u \leq L_a + L_b$. The index range for the expansion coefficients is the same as in the MD2 scheme.

An alternative method to transform $\bar{\boldsymbol{l}}_p$ into $\boldsymbol{l}_a\boldsymbol{l}_b$ is to use the recurrences Eqs. (1.2.24) and (1.2.25). Relying on these relations, one can start from the two-center Hermite integrals $(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$, which are, by Eq. (2.1.9), practically scaled one-center $(\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ integrals, and, through hybrid intermediates, convert these into the target $(\Omega^{\bar{\mathbf{0}}}_{\boldsymbol{L}_a,\boldsymbol{L}_b}|\tilde{\boldsymbol{L}}_c) = (\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$ classes with a purely Cartesian bra side. In the MD4, MD5, and MD6 schemes, we proceed the same way as in the MD1, MD2, and MD3 cases, respectively, with the difference that the calculation of the expansion coefficients is omitted, and instead of Eq. (2.1.11) Eqs. (1.2.24) and (1.2.25) are applied for the transformation of the bra side. In each case, the required $(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ classes are the same as the ones necessary for Eq. (2.1.11).

2.1.3 Gill–Head-Gordon–Pople method

This approach is very similar to the MD5 scheme. The difference lies in the utilization of functions similar to those given by Eq. (1.3.1) for the bra side of the integrals. Since $\boldsymbol{l}_b$ will be incremented by Eq. (2.1.3), slightly simpler auxiliary functions defined as

$$_{\beta,\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|\tilde{\boldsymbol{L}}_c) = \frac{(2b)^\beta}{(2p)^\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\boldsymbol{l}_b}|\tilde{\boldsymbol{L}}_c) \tag{2.1.14}$$

will be used, because Eq. (1.3.2) only manipulates the β and ζ indices. Using the GHP scheme for three-center ERIs first the required $(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c) = (\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ Hermite integrals for $0 \leq \bar{l}_p \leq L_a + L_b$ are calculated. Then, all the scaled classes of these integrals necessary to the computation of the $_{0,0}(\Omega^{\bar{\mathbf{0}}}_{\boldsymbol{l}_a,\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ classes with Eq. (1.3.2) for $L_a \leq l_a \leq L_a + L_b$ are evaluated. For each of these classes we need to start from the $_{\beta,\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ scaled Hermite intermediates for $0 \leq \bar{l}_p \leq l_a$. To determine the β, ζ-scaled classes needed for each $(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ that will be used for the calculation of a given $_{0,0}(\Omega^{\bar{\mathbf{0}}}_{\boldsymbol{l}_a,\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ we have to trace back the recursion defined by Eq. (1.3.2). As each recursion step increments $\boldsymbol{l}_a$ by $\mathbf{1}_\sigma$ where σ is one of the Cartesian directions, there are an l_a number of steps. By analyzing the positions where $_{\beta,\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\mathbf{0},\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ and the intermediates connected to it can appear in Eq.

(1.3.2) during the recursion, it can be seen that such intermediates have to be the third term at least $\bar{l}_p$ times to reduce $\bar{\boldsymbol{l}}_p$ to $\bar{\mathbf{0}}$. In the additional $l_a - \bar{l}_p$ steps, these intermediates have to appear at the first and the third positions equal times if $\bar{\boldsymbol{l}}_p$ is to stay equal to $\bar{\mathbf{0}}$, and in the remaining steps they have to be the second term. From this it follows that for each $l_a, \bar{l}_p$ pair there are $\lceil (l_a - \bar{l}_p)/2 \rfloor + 1$ different scalings to consider. The β and ζ for these can be obtained by looking at how the changes in these values depend on the position the intermediates take in Eq. (1.3.2). The scaling indices are determined by how many times the connected intermediates take the second or third position. For example, in the case they take the third place $\bar{l}_p$ times and the second position in the remaining $l_a - \bar{l}_p$ steps, the values of β and ζ are $l_a - \bar{l}_p$ and l_a, respectively. Another example is when the intermediate takes the second position in two fewer steps in the recursion, and both the first and the third places are taken one more time than in the former example, making the scaling indices $\beta = l_a - \bar{l}_p - 2$ and $\zeta = l_a - 1$. Let us denote the scaled class in the first example as class 1 and that in the second example as class 2. In general, class n can be defined for the scaling indices $\beta = l_a - \bar{l}_p - 2(n-1)$ and $\zeta = l_a - (n-1)$. After these classes for $1 \leq n \leq \lceil (l_a - \bar{l}_p)/2 \rfloor + 1$ have been calculated for all the primitive classes, the scaled one-center integrals are transformed to the contracted basis by Eq. (1.1.7). When using segmented basis sets, the multiplication work in this contraction step can be reduced to simply multiplying Eq. (2.1.10) with the appropriate $d_{a\chi_A}$ coefficients [32]. Following the contraction Eq. (1.3.2) is applied, and lastly Eq. (2.1.3) is used to build up $\boldsymbol{l}_b$.

2.1.4 Rys polynomial method

From the discussion in Sect. 1.5 it follows that three-center ERIs over Cartesian Gaussians can be written as

$$\begin{aligned}(\boldsymbol{L}_a \boldsymbol{L}_b | \boldsymbol{L}_c) &= \int_0^1 \Theta_x^{I_a,I_b,I_c}(t^2)\Theta_y^{J_a,J_b,J_c}(t^2)\Theta_z^{K_a,K_b,K_c}(t^2)W(T,t^2)\mathrm{d}t \\ &= \sum_{r=1}^{N_{rts}} \Theta_x^{I_a,I_b,I_c}(t_r^2)\Theta_y^{J_a,J_b,J_c}(t_r^2)\Theta_z^{K_a,K_b,K_c}(t_r^2)w_r \qquad (2.1.15)\end{aligned}$$

with $N_{rts} > \lceil (L_a + L_b + L_c)/2 \rfloor$ and

$$\Theta_x^{I_a,I_b,I_c}(t^2) = \sum_{\bar{i}_p=0}^{I_a+I_b} \sum_{\bar{i}_q=0}^{I_c} (-1)^{\bar{i}_q} E_{\bar{i}_p}^{I_a,I_b} E_{\bar{i}_q}^{I_c,0} \mathcal{H}_{\bar{i}_p+\bar{i}_q}(t^2) \ . \qquad (2.1.16)$$

Following an analogous derivation to the one for the 2D integrals suitable for four-center ERI evaluation the recursions

$$\begin{aligned}\Theta_x^{i_a+1,i_b,i_c}(t^2) &= \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}t^2 X_{\mathrm{PC}}\Big)\Theta_x^{i_a,i_b,i_c}(t^2) \\ &+ \frac{1}{2p}\Big(1-\frac{\alpha}{p}t^2\Big)[i_a\Theta_x^{i_a-1,i_b,i_c}(t^2) + i_b\Theta_x^{i_a,i_b-1,i_c}(t^2)] \\ &+ \frac{\alpha}{2pc}t^2 i_c\Theta_x^{i_a,i_b,i_c-1}(t^2)\ , \end{aligned} \tag{2.1.17}$$

and

$$\begin{aligned}\Theta_x^{i_a,i_b,i_c+1}(t^2) &= \frac{\alpha}{c}t^2 X_{\mathrm{PC}}\Theta_x^{i_a,i_b,i_c}(t^2) \\ &+ \frac{1}{2c}\Big(1-\frac{\alpha}{c}t^2\Big)i_c\Theta_x^{i_a,i_b,i_c-1}(t^2) \\ &+ \frac{\alpha}{2pc}t^2[i_a\Theta_x^{i_a-1,i_b,i_c}(t^2) + i_b\Theta_x^{i_a,i_b-1,i_c}(t^2)]\ , \end{aligned} \tag{2.1.18}$$

with

$$\Theta_x^{0,0,0,0} = \kappa_{ab}^x \tag{2.1.19}$$

are obtained. Eqs. (2.1.9) and (1.5.3) imply that if we multiplied Eq. (2.1.15) with $(-2c)^{\tilde{L}_c}$, we would obtain the quadrature formula for $(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$, given that we define the 2D integrals as

$$\Theta_x^{I_a,I_b,I_c}(t^2) = \sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b}\mathcal{H}_{\bar{i}_p+I_c}(t^2)\ . \tag{2.1.20}$$

Although Eq. (2.1.20) is much simpler than Eq. (2.1.16), incrementing the I_a index and substituting Eq. (1.2.19) results in the same recurrence as Eq. (2.1.17), so this approach does not offer any benefits. An efficient way to apply Eq. (2.1.17) is to first build up i_a as

$$\Theta_x^{i_a+1,0,0}(t^2) = \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}X_{\mathrm{PC}}t^2\Big)\Theta_x^{i_a,0,0}(t^2) + \frac{i_a}{2p}\Big(1-\frac{\alpha}{p}t^2\Big)\Theta_x^{i_a-1,0,0}(t^2)\ , \tag{2.1.21}$$

then i_c with the equation

$$\begin{aligned}\Theta_x^{i_a,0,i_c+1}(t^2) &= \frac{\alpha}{c}X_{\mathrm{PC}}t^2\Theta_x^{i_a,0,i_c}(t^2) + \frac{\alpha}{2pc}t^2 i_a\Theta_x^{i_a-1,0,i_c}(t^2) \\ &+ \frac{1}{2c}\Big(1-\frac{\alpha}{c}t^2\Big)i_c\Theta_x^{i_a,0,i_c-1}(t^2)\ , \end{aligned} \tag{2.1.22}$$

and finally i_b using Eq. (1.5.14). Two observations should be made before proceeding. First, in Sect. 1.5 it was mentioned that the factors multiplying the Boys functions in Eq. (1.5.1) can be given by using the OS recursions [Eqs. (1.4.11) and (1.4.12)] to expand the ERI in only Boys functions. These recurrences are valid for arbitrary n and the factors multiplying the ERIs do not depend on n. It follows that the auxiliary integrals of the

OS scheme for three-center ERIs can be written as

$$(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c)^{(n)} = \sum_{m=0}^{l_a+l_b+l_c} Z_m F_{m+n}(\alpha R_{\text{PC}}^2) = \int_0^1 \sum_{m=0}^{l_a+l_b+l_c} Z_m t^{2(m+n)} \exp(-\alpha R_{\text{PC}}^2 t^2)\mathrm{d}t \ . \quad (2.1.23)$$

This means the polynomial multiplying the weight factor inside the integral is the same for arbitrary n except for a multiplication by a t^{2n} term. The more general quadrature formula is written as

$$(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c)^{(n)} = \sum_{r=1}^{N_{rts}} \Theta_x^{i_a,i_b,i_c}(t_r^2)\Theta_y^{j_a,j_b,j_c}(t_r^2)\Theta_z^{k_a,k_b,k_c}(t_r^2) w_r t_r^{2n} \ . \quad (2.1.24)$$

Second, the third term in Eq. (2.1.22) can be omitted if $\boldsymbol{L}_c$ is going to be transformed into the solid harmonic Gaussian basis. To see this, notice from backtracking the recursion defined by Eq. (1.7.8) that an integral $(\boldsymbol{l}_a^{\#}\boldsymbol{0}|\boldsymbol{l}_c^{\#})^{(m+n)}$ contributes to $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)}$ only if

$$l_c^{\#} = l_c - n \quad (2.1.25)$$

since each recursion step decreases n and increases $l_c^{\#}$ by one. Next, express $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)}$ as

$$(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)} = \sum_{r=1}^{N_{rts}} \Theta_x^{i_a,0,i_c}(t_r^2)\Theta_y^{j_a,0,j_c}(t_r^2)\Theta_z^{k_a,0,k_c}(t_r^2) w_r t_r^{2m} \ . \quad (2.1.26)$$

Substituting Eq. (2.1.22) into Eq. (2.1.26) results in

$$\begin{aligned}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)} &= \sum_{r=1}^{N_{rts}} t_r^{2m} w_r \\ &\left[\frac{\alpha}{c}X_{\text{PC}}t_r^2\Theta_x^{i_a,0,i_c-1}(t_r^2) + \frac{\alpha}{2pc}t_r^2 i_a\Theta_x^{i_a-1,0,i_c-1}(t_r^2) + \frac{i_c}{2c}(1-\frac{\alpha}{c}t_r^2)\Theta_x^{i_a,0,i_c-2}(t_r^2)\right] \\ &\left[\frac{\alpha}{c}Y_{\text{PC}}t_r^2\Theta_y^{j_a,0,j_c-1}(t_r^2) + \frac{\alpha}{2pc}t_r^2 j_a\Theta_y^{j_a-1,0,j_c-1}(t_r^2) + \frac{j_c}{2c}(1-\frac{\alpha}{c}t_r^2)\Theta_y^{j_a,0,j_c-2}(t_r^2)\right] \\ &\left[\frac{\alpha}{c}Z_{\text{PC}}t_r^2\Theta_z^{k_a,0,k_c-1}(t_r^2) + \frac{\alpha}{2pc}t_r^2 k_a\Theta_z^{k_a-1,0,k_c-1}(t_r^2) + \frac{k_c}{2c}(1-\frac{\alpha}{c}t_r^2)\Theta_z^{k_a,0,k_c-2}(t_r^2)\right] \ . \end{aligned} \quad (2.1.27)$$

Each of the terms arising by performing the multiplications amongst the brackets can contribute to an integral determined by the indices of the 2D integrals. For example, the term arising from multiplying the first terms of the brackets contributes to a scaled version of $(\boldsymbol{l}_a^{\#}\boldsymbol{0}|\boldsymbol{l}_c^{\#})^{(m+3)}$ with $\boldsymbol{l}_a^{\#} = (i_a, j_a, k_a)$ and $\boldsymbol{l}_c^{\#} = (i_c-1, j_c-1, k_c-1)$ through Eq. (2.1.26), which is used in the expansion of $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)}$ by Eq. (1.7.8) if we go three steps back in the recursion. The terms containing the third 2D integral from one or more brackets in Eq. (2.1.27) are used to build the $(\boldsymbol{l}_a^{\#}\boldsymbol{0}|\boldsymbol{l}_c^{\#})^{(m+n)}$ classes with Eq. (2.1.26) where $0 \le n \le 3$ (because the third 2D integral can be multiplied by a quantity that does or does not contain t_r^2), $l_a - 2 \le l_a^{\#} \le l_a$ (because the product can contain a maximum of two of the second 2D integrals which each reduce $l_a^{\#}$ by one), and $l_c - 6 \le l_c^{\#} \le l_c - 4$ (because the first two 2D integrals reduce $l_c^{\#}$ by one, while the third does so by two).

Since none of these satisfy Eq. (2.1.25), the contributions containing the third terms in the brackets in Eq. (2.1.27) will be canceled during the solid harmonic transformation and can be taken to be zero, which means that Eq. (2.1.22) reduces to

$$\Theta_x^{i_a,0,i_c+1}(t^2) = \frac{\alpha}{c} X_{\mathrm{PC}} t^2 \Theta_x^{i_a,0,i_c}(t^2) + \frac{\alpha}{2pc} t^2 i_a \Theta_x^{i_a-1,0,i_c}(t^2) \ . \tag{2.1.28}$$

The same reasoning applies to the second term in Eq. (2.1.21) in the case of $\boldsymbol{L}_b = \boldsymbol{0}$, when the third and fourth terms in Eq. (2.1.2) vanish.

Three possibilities for the three-center ERIs will be investigated here. In the RYS1 algorithm, the $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ integrals are evaluated directly by integrating on the quadrature. For this purpose, we have to compute $\Theta_x^{i_a,i_b,i_c}(t_r^2)$ for $1 \leq i_a \leq L_a$, $1 \leq i_b \leq L_b$, and $1 \leq i_c \leq L_c$ for the N_{rts} roots and for the three directions. In the RYS2 scheme, $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{L}_c)$ classes are calculated on the quadrature for $L_a \leq l_a \leq L_a + L_b$, then the OS-type HRR, Eq. (2.1.3) is applied. The indices of the necessary 2D integrals here are in the range of $1 \leq i_a \leq L_a + L_b$, $i_b = 0$, and $1 \leq i_c \leq L_c$. Finally, in the RYS3 algorithm the needed 2D integrals are $\Theta_x^{i_a,0,0}(t_r^2)$ for $1 \leq i_a \leq L_a + L_b$ for all the roots and directions, the $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0})^{(L_c)}$ classes are constructed for $L_a - L_c \leq l_a \leq L_a + L_b$ on the quadrature, and the target integrals are built up via Eqs. (1.7.8) and (2.1.3). To make the computations more efficient, the starting values $\Theta_x^{0,0,0} = \Theta_y^{0,0,0} = 1$ will be used for each root, as well as $\Theta_z^{0,0,0}(t_r^2) = \kappa_{ab} w_r$. In the case when ERIs with $n \neq 0$ are evaluated, the necessary t_r^{2n} factors can also be built into the $\Theta_z^{0,0,0}(t_r^2)$ values. The working equation for the quadrature is thus

$$(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c)^{(n)} = \sum_{r=1}^{N_{rts}} \Theta_x^{i_a,i_b,i_c}(t_r^2)\Theta_y^{j_a,j_b,j_c}(t_r^2)\Theta_z^{k_a,k_b,k_c}(t_r^2) \ . \tag{2.1.29}$$

2.1.5 Algorithmic considerations

Since its introduction the HRR equation, Eq. (2.1.3), has been a standard tool for evaluating molecular integrals over Gaussian functions. In addition to being a simple two-term recurrence relation, it is also independent of the basis set exponents, making it possible to apply it to contracted integrals instead of primitive ones, which (usually) means a smaller number of integrals to be treated. The same is true for the transformation to the solid harmonic Gaussian basis, and it has also been proposed that these two operations for one side (bra or ket) can be efficiently combined into a single matrix multiplication [64]. On the other hand, if we choose to use Eqs. (2.1.3) and (1.1.6) at the contracted level, we have to first contract the components of the classes $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{L}_c)$ for $L_a \leq l_a \leq L_a + L_b$, which consist of $[(L_a + L_b + 1)(L_a + L_b + 2)(L_a + L_b + 3)/6 - 1 - L_a(L_a + 1)/2](L_c + 1)(L_c + 2)/2$ integrals for every final class of $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$. If we perform the HRR and the solid harmonic transformation at the primitive level instead, this number becomes $(2L_a + 1)(2L_b + 1)(2L_c + 1)$, which is smaller in all the cases. This does not only affect the operation count of the contraction step, but the memory use of the code as well. For example, if we apply the nested loop structure shown in Algorithm 1, the arrays storing the partially and fully contracted integrals will be the largest ones used in the process of evaluating all $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ ERIs for three given centers. This means that we can expect the most data cache-miss events (meaning that the copy of the data stored at a referenced memory address cannot be found in the cache memory of the CPU) to happen at this stage of the algorithm. Since the fetching of data from main memory is about a magnitude slower than from the cache (two magnitudes if the data reside in the first level of the cache), such misses can have a considerable effect on the performance of the code, and fewer misses are expected for a smaller array. Thus it can be seen that it is not a trivial decision where Eqs. (2.1.3) and (1.1.6) should be applied. The schemes where the HRR and the solid harmonic transformation are done at the primitive level will be denoted as IN, while the ones where these two steps are performed at the contracted level will be labeled as OUT.

Our contraction procedure distinguishes between contracted and uncontracted functions for all three centers, especially because there can be significant number of uncontracted functions in generally contracted basis sets, e.g., in the cc-pVXZ bases [96, 97]. For example, in the cc-pVTZ basis for elements Li to Ne all the d and f functions are uncontracted, and out of the four s and three p functions only two and one are contracted, respectively, and all the functions in the corresponding fitting basis [88], cc-pVTZ-RI, are uncontracted. For the integrals that are evaluated over primitives which contribute to an uncontracted function the quantity $\theta_{pc}\kappa_{ab}$ is multiplied by the norm factor of the function which is otherwise absorbed into the contraction coefficients, and the integrals are written directly into the array that stores the contracted integrals, therefore both the floating-point and memory operations for the contraction are saved. In the case these primitives also contribute to other, contracted functions, the coefficients of the affected primitives for these contracted functions in Eq. (1.1.7) are divided by the above mentioned norm. Further notes on the efficient treatment of integral contraction will be discussed in Sect. 2.3.

The sizes of the arrays for integral contraction can be further reduced when the auxiliary basis set used for the density fitting approximation is uncontracted even if the functions on centers **A** and **B** are contracted. If we change the order of loops from a, b, c to c, a, b as it is shown in Algorithm 2, the sizes of the arrays for the contraction of the first and second functions reduce by a factor of the number of the contracted functions on the third center. Here the loop over the exponents of the ket side is also the loop over the contracted functions on **C**, and all calculations are performed inside this loop. This scheme, however, has the disadvantage that the a- and b-dependent quantities have to be precalculated in a separate loop to avoid unnecessary recalculations. Schemes with the *a*,*b*,*c* primitive loop structure will be referred to as *abc*, while the ones with *c*,*a*,*b* order will be denoted by *cab*.

Finally, from the recursive formulas for the calculation of six-dimensional integrals given in Sects. 2.1.1 to 2.1.3 it is evident that an integral can be constructed in numerous ways by such recursions, depending on which of the x,y,z components of the angular momentum is raised in the various recursion steps. A well-known consequence of this is that not all components of the intermediate classes have to be calculated, and that different paths in the recursion have different operation counts [98, 99]. In the present algorithms, the related tree-search problems were treated utilizing the ideas of Ryu and co-workers [99].

2.1.6 Prescreening of three-center integrals

Another aspect that can have a strong effect on the performance is the prescreening of integrals which are lower in absolute value than a user-defined threshold, hereafter denoted by ε. In the present work, as usual, the entire shell triplets are prescreened invoking the Schwarz inequality [12], and the distance-dependent estimator of Valeev and co-workers [100] is also employed. In addition, the screening of the primitive integrals is also implemented. For the latter, the threshold is also tied to ε by dividing it by the maximal level of contraction, that is, the product of the number of primitive functions on each center. Exceptions from this rule are integrals that contain a primitive (centered on, for example, $\mathbf{A}$) which contributes to only one contracted function χ_A. Then, ε is not divided by the number of primitives on $\mathbf{A}$, but rather the level of contraction for χ_A, making the threshold for primitive prescreening higher. For the estimation of the magnitude of the primitive integrals the value of the ERI $(\mathbf{00}|\mathbf{0})$ will be used. Instead of directly applying Eq. (2.1.1), the upper bound for the zeroth-order Boys function [35] can be used, from which we get

$$(\mathbf{00}|\mathbf{0}) = \theta_{pc}\kappa_{ab}F_0(\alpha R_{\text{PC}}^2) \leq \theta_{pc}\kappa_{ab}\text{min}\Big(1, \sqrt{\frac{\pi}{4\alpha R_{\text{PC}}^2}}\Big) \ . \tag{2.1.30}$$

The minimum criterion appears since the approximation used in Eq. (2.1.30) is only accurate for high values of αR_{PC}^2 (greater than about 74), and for smaller arguments it can give results greater than 1, which is the highest value the zeroth-order Boys function can take (when $\alpha R_{\text{PC}}^2 = 0$). In actual calculations it is more beneficial to use the square of the rightmost side of Eq. (2.1.30) for screening, so the expensive square root calculation only have to be done for classes with large αR_{PC}^2 that survive the prescreening. In this method (algorithm pPRE1), the estimate for $|(\mathbf{00}|\mathbf{0})|^2$ is compared to ε^2, and if the former value is greater, the class is evaluated.

Algorithm 1 *abc* primitive loop order

```
Loop over $a$
   Loop over $b$
     Algorithm pPRE2:  estimate $(\mathbf{00}|\mathbf{0})$ for the smallest $c$
      Loop over $c$
        Algorithm pPRE1:  estimate $(\mathbf{00}|\mathbf{0})$
        Algorithm OUT: Build up $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ for $L_a \le l_a \le L_a + L_b$ in the Cartesian
           Gaussian basis
        Algorithm IN: Build up $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ in the solid harmonic Gaussian basis
      End loop
     Contract the third function for all classes with exponents $a$ and $b$
   End loop
  Contract the second function for all classes with exponent $a$
End loop
Contract the first function for all classes
Loop over $\chi_A$(executed only in the case of algorithm OUT)
   Loop over $\chi_B$
     Algorithm cPRE2:  Look up the integral of highest absolute value in
        the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for the contracted $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$
        class with the smallest $c$
     Algorithm cPRE3:  Estimate the integral of highest absolute value in
        the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for the contracted $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$
        class with the smallest $c$
      Loop over $\chi_C$
        Algorithm cPRE1:  Look up the integral of highest absolute value in
           the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for the contracted $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$
           class
        Algorithm OUT: perform HRR to get $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$, perform solid harmonic
           transformation
      End loop
   End loop
End loop
```

In practice, I found that it can be more efficient to screen a batch of primitive exponent triplets than each individual one. Here it is utilized that the value of the right-hand side of Eq. (2.1.30) increases with the decrement of the Gaussian exponent c for the ket side. This can be seen by noting that $\partial\alpha/\partial c = p^2/(p+c)^2$ is always a positive number. Hence, we only need to estimate the $(\mathbf{00}|\mathbf{0})$ integral with the smallest c in an *abc* scheme before the innermost loop (algorithm pPRE2). One could proceed the same way in a *cab* scheme estimating the integral with the smallest b before the loop over b, but I found this choice to be inefficient, as it will be discussed in Sect. 2.4.

The primitive prescreening described above does not reduce the work of the HRR

Algorithm 2 *cab* primitive loop order

```
Loop over $a$
   Loop over $b$
     Calculate the quantities depending on functions in the bra
   End loop
End loop
Loop over $c$
   Loop over $a$
     Algorithm pPRE2:  estimate $(\mathbf{00}|\mathbf{0})$ for the smallest $b$
      Loop over $b$
        Algorithm pPRE1:  estimate $(\mathbf{00}|\mathbf{0})$
        Algorithm OUT: Build up $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ for $L_a \leq l_a \leq L_a + L_b$ in the Cartesian
           Gaussian basis
        Algorithm IN: Build up $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ in the solid harmonic Gaussian basis
      End loop
     Contract the second function for all classes with exponents $c$ and $a$
   End loop
  Contract the first function for all classes with exponent $c$
   Loop over $\chi_A$(executed only in the case of algorithm OUT)
      Loop over $\chi_B$
        Algorithm cPRE1:  Look up the integral of highest absolute value in
           the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for the contracted $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$
           class
        Algorithm cPRE4:  Estimate the integral of highest absolute value
           in the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for the contracted
           $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ class
        Algorithm OUT: perform HRR to get $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$, perform solid harmonic
           transformation
      End loop
   End loop
End loop
```

and the solid harmonic transformation steps if these are performed at the contracted level (algorithm OUT). The simplest option in this case is, for each combination of the contracted functions, to check if the largest value out of the contracted $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes needed for a class of $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ is greater than the threshold before applying Eqs. (2.1.3) and (1.1.6) to get the given class (algorithm cPRE1). We can also chose to screen a bigger batch of contracted classes instead by performing the search for the integral of highest absolute value before the loop over χ_C in an *abc* scheme or χ_B in a *cab* scheme. This is advantageous when the fitting basis is uncontracted and an *abc* scheme is applied (see Algorithm 1). In these cases, there will be an assumption that the integrals involving the most diffuse functions (that is, the smallest c) on the ket side will have higher absolute values than those containing higher c exponents, and therefore screening for the classes with the smallest c is enough to see if any of the integrals in the batch will reach the threshold (algorithm cPRE2). Like the pPRE1 and pPRE2 methods, this is not a rigorous screening, but its preciseness will be demonstrated in Sect. 2.4. An alternative method is to estimate the integral with the highest absolute value out of the screened batch. For this purpose, the estimates of the $(\mathbf{00}|\mathbf{0})$ integrals made by Eq. (2.1.30) are saved. Then, an estimated upper bound for the integral of highest value of a contracted class is gained by taking the $(\mathbf{00}|\mathbf{0})$ estimate calculated from the smallest a, b, and c exponents which contribute to the contracted functions in question and multiplying it by both the degree of contraction (product of the number of primitives for the three functions) and the maximal contraction coefficient used for each contracted function. This estimation can also be done before the loop over χ_C for the class with the smallest c (algorithm cPRE3) in an *abc* scheme (see Algorithm 1) when the fitting basis is uncontracted. With a *cab* loop order (Algorithm 2) wit cannot be assumed which contracted class contains the integrals of highest absolute value, therefore the estimation is performed for each class inside the loop over χ_B (algorithm cPRE4).

2.2 Floating point operation counts

The FLOP requirements of the discussed schemes were estimated by a program developed for this purpose. The considered operations include the calculation of the primitive integrals and the transformation into the solid harmonic Gaussian and contracted bases. Estimations for the evaluation of Boys functions and the roots and weights for the Rys quadratures are omitted because the computational requirements of both steps depend heavily on the actual values of αR_{PC}^2. Nevertheless, I found that the computation time spent on the two operations is rather similar, thus the neglect of their FLOP counts is not expected to influence the conclusions. Prescreening of the integrals is also not taken into account since this is also strongly system-dependent. The program counts the FLOP requirements of the schemes according to the equations given in Sect. 2.1 supposing that reusable compound quantities, such as $(\alpha/p)X_{\mathrm{PC}}$ in Eq. (2.1.2), are precalculated and treated as single variables. The sparsity of the transformation matrices for the solid harmonic Gaussian transformation and the primitive contraction is taken into consideration. The *abc* primitive loop structure was used and the solid harmonic transformation and the HRR was performed at the contracted level since this is the most conventional approach, but this does not change the theoretical order of efficiency for the investigated schemes. In the calculations presented in the following, a model system of three carbon atoms were chosen, and the number of FLOPs needed to evaluate all the ERIs over three separate centers was estimated for Dunning's [96] correlation consistent cc-pVXZ (X=D,T,Q,5) basis sets (XZ for short) for the bra side and the corresponding auxiliary basis sets of Weigend [88] (cc-pVXZ-RI) for the ket side.

The overall FLOP counts for all the shell triplets for the various algorithms are presented in Table 2.1, and for the TZ basis set the FLOP counts are depicted on the part a) of Fig. 2.1. Figures that show the theoretical performance of the other algorithms relative to the OS1 scheme can be found in the Supplementary Material of the related publication [101]. It can be seen that out of the OS-based schemes the OS1 algorithm shows the best theoretical performance. In the OS2 and OS3 schemes the more expensive recursion for $\boldsymbol{l}_a$ takes place after the build-up of $\boldsymbol{l}_c$, which makes these algorithms perform progressively worse with basis sets of higher cardinal number compared to OS1. In the OS4 route, the extra work introduced on the bra side with the use of the ETR becomes less and less significant with higher angular momenta in the bra, making the relative performance of OS4 better with bigger bases. Nevertheless, the OS1 scheme provides the lowest FLOP counts for each shell triplet. For the MD-based algorithms, the introduction of both the HRR for the bra (MD2 and MD5) and the VRR for the ket side (MD3 and MD6) improves the performance with respect to the MD1 and MD4 schemes, and increasingly so with the growth of $\boldsymbol{L}_b$ and $\boldsymbol{L}_c$, respectively. None of the MD routes perform better then the OS1 for any shell triplets except for the $(\boldsymbol{ss}|\boldsymbol{p})$, where the MD1, MD2, MD4, and MD5 schemes are slightly cheaper since the additional calculation of α/c from Eq. (1.7.8) is not necessary. Looking at the best performing MD3 and MD6 schemes, we see that the use of Eqs. (1.2.24) and (1.2.25) is preferred to the assembly of Eq. (2.1.11), except when $\boldsymbol{L}_b = \boldsymbol{0}$. The GHP scheme performs better than the OS1 when the bra side is $(\boldsymbol{ps}|$ since the extra contraction work for the scaled Hermite classes needed for Eq. (1.3.2) is negligible in these cases [except for very high angular momenta in the ket, see for example the $(\boldsymbol{ps}|\boldsymbol{i})$ shell triplet] and the $\boldsymbol{s}$ and $\boldsymbol{p}$ shells are contracted in all the investigated basis sets. The $(\boldsymbol{ss}|\boldsymbol{p})$ shell triplet also performs better, for the same reason as with the MD schemes. For higher angular momenta Eq. (1.3.2) becomes inefficient, hence the GHP scheme is only competitive for the DZ basis. As in the MD cases, the HRR for RYS2 and the ket-side VRR for RYS3 improve the FLOP counts. The RYS2 and RYS3 algorithms outmatch the OS1 in most of the cases when $\boldsymbol{L}_c = \boldsymbol{0}$. For example, the OS1 scheme is better for $(\boldsymbol{ds}|\boldsymbol{s})$, but not for $(\boldsymbol{dp}|\boldsymbol{s})$. This is because the two-point quadrature is more costly than Eq. (2.1.2) for the former case, but it is cheaper for the latter. The RYS1 is the worst performing one of the Rys-based algorithms, but it is still superior to OS1 for particular shell triplets, for example, for $(\boldsymbol{fd}|\boldsymbol{s})$. The RYS3 scheme can be better than the OS1 for $\boldsymbol{p}$ kets if the change from $\boldsymbol{s}$ to $\boldsymbol{p}$ does not increase the number of quadrature points. However, since from Eq. (1.7.8) the $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0})^{(L_c)}$ integral classes that have to be calculated with quadrature for RYS3 are in the range of $L_a - L_c \leq l_a \leq L_a + L_b$, the growth of $\boldsymbol{L}_c$ also increases the work in the quadrature

TABLE 2.1: FLOP counts for the various algorithms with the cc-pVXZ basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D $(\boldsymbol{dd}\|\boldsymbol{f})$	T $(\boldsymbol{ff}\|\boldsymbol{g})$	Q $(\boldsymbol{gg}\|\boldsymbol{h})$	5 $(\boldsymbol{hh}\|\boldsymbol{i})$
OS1	445777	2231707	14074904	71407908
OS2	545297	2967883	19981747	106671377
OS3	632210	3465805	22599746	116871757
OS4	754037	3587118	21812481	106908381
MD1	599215	3801560	30617263	198278829
MD2	555359	3165292	22249286	125117638
MD3	474978	2473785	15766358	80931165
MD4	616235	3824184	29178035	173497467
MD5	570267	3272596	22532400	121220098
MD6	470050	2420151	15243362	77170230
GHP	499430	3188703	25032932	152491888
RYS1	622518	3181603	20929684	112060719
RYS2	585778	2902749	18203750	92155512
RYS3	467187	2308256	14413073	72659045

step, so this is only the case for higher angular momentum bras. All in all, there is only a small difference between the overall estimates for the best performing OS1 and RYS3 algorithms. Because of this, and also because the FLOPs counts of the Boys functions and the roots and weights of the Rys quadratures are not estimated, these two schemes were implemented efficiently using automated code generation and wall time measurements were carried out, as will be discussed in Sects. 2.3 and 2.4, to decide which of the two is the most efficient scheme. The GHP algorithm for the $(\boldsymbol{ps}|\boldsymbol{s})$ - $(\boldsymbol{ps}|\boldsymbol{g})$ integrals has also been implemented "by hand", because the FLOP counts with this scheme are the lowest for these triplets.

The FLOP counts for the four different possible combinations of the IN-OUT and *abc*-*cab* schemes for the OS1 algorithm are shown in Table 2.2, and on the part b) of Fig. 2.1. The conclusions are also true for the RYS3 algorithm since the OS1 and RYS3 schemes do not differ in any part that is affected by varying these four algorithmic approaches. The estimates for the *abc* and *cab* cases are essentially the same, the small difference comes from the fact that for the *abc* schemes the additional costs of the pPRE2 type primitive prescreening are also counted, because additional calculations are needed here before the loop over c. The differences between the IN and OUT algorithms are more significant, and as expected, performing the HRR and the solid harmonic transformation at the contracted level is theoretically more efficient in every case when at least one of the functions is contracted. The difference becomes less pronounced with higher basis sets, because the $\boldsymbol{d}$ and higher shells are uncontracted in the investigated bases. These results,

Table 2.2: FLOP counts for the four different OS1 algorithms with the cc-pVXZ basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D $(\boldsymbol{dd}\vert\boldsymbol{f})$	T $(\boldsymbol{ff}\vert\boldsymbol{g})$	Q $(\boldsymbol{gg}\vert\boldsymbol{h})$	5 $(\boldsymbol{hh}\vert\boldsymbol{i})$
IN-*abc*	566748	2664883	15919037	79233985
IN-*cab*	565054	2662374	15916043	79232960
OUT-*abc*	445777	2231707	14074904	71407908
OUT-*cab*	443201	2227609	14069600	71404912

however, do not provide information about the difference in performance that could arise from the different memory layouts and prescreening strategies of the schemes. Hence, to assess the wall time performances as well as cache-miss rates these four variations have also been efficiently implemented for both the OS1 and RYS3 algorithms, and the *abc* and *cab* versions of the GHP schemes were also programmed.

Finally, the above described counter program was modified to estimate the memory required by the various algorithms. The memory demand mainly consists of two parts. Arrays are required to store the intermediate quantities necessary to compute the primitive ERIs, and to hold the intermediates of the primitive contraction (see Sect. 2.3). As the angular momenta increase, the memory demand of the recursion intermediates becomes the only important contribution since the number of primitives and AOs decrease. Table 2.3 presents the required memory (not counting the storage of the final ERIs) for the shell triplet with the highest memory demand for each algorithm for the cases where each center is a carbon atom using the cc-pV5Z and cc-pV5Z-RI basis sets, and when the centers are gold atoms applying the cc-pV5Z-PP and cc-pV5Z-PP-RI bases. For most schemes the highest memory demand belongs to the shell triplet with the highest angular momenta. The algorithms with the smallest memory footprint are the non-recursive methods, that is, the MD1 and RYS1 schemes. Since the primitive ERI evaluation does not require much memory for these methods, the shell triplets with the highest memory requirement can have lower angular momenta as well, as it is the case for the ERIs with gold centers in Table 2.3. This is only the case with the *abc* schemes, however, since, with the *cab* route, the memory demand of the arrays used for contraction is reduced.

2.3 Implementation

The four combinations of the IN-OUT and *abc*-*cab* schemes for the OS1 and RYS3 algorithms together with the prescreening approaches discussed in Sect. 2.1.5 have been implemented in the Mrcc program suite [102] by the means of automated code generation. The *abc* and *cab* variants of the GHP algorithm for the $(\boldsymbol{ps}|\boldsymbol{s})$ - $(\boldsymbol{ps}|\boldsymbol{g})$ triplets have

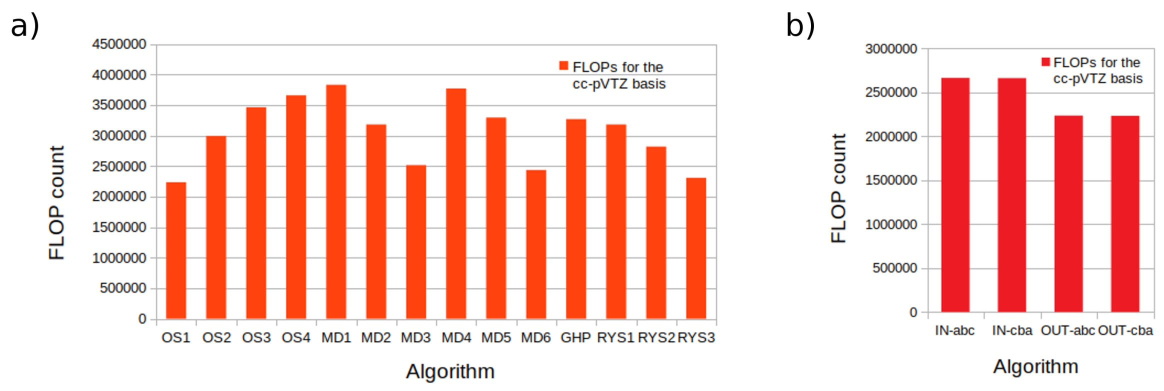

FIGURE 2.1: Theoretical FLOP counts for the various algorithms with the cc-pVTZ basis. The illustrated data is from a) Table 2.1 b) Table 2.2.

TABLE 2.3: Memory demand in kilobytes for the various algorithms for ERIs of three carbon atoms with the cc-pV5Z basis, and for ones of three gold atoms with the cc-pV5Z-PP basis. The shell triplet with the highest memory demand is shown. The algorithms assume the IN-*abc* route, and the *abc* scheme for OS1 OUT.

	C		Au	
OS1	$(\boldsymbol{hh}\|\boldsymbol{i})$	669.22	$(\boldsymbol{ii}\|\boldsymbol{l})$	1,966.91
OS2	$(\boldsymbol{hh}\|\boldsymbol{i})$	922.81	$(\boldsymbol{ii}\|\boldsymbol{l})$	2,865.56
OS3	$(\boldsymbol{hh}\|\boldsymbol{i})$	956.58	$(\boldsymbol{ii}\|\boldsymbol{l})$	2,753.92
OS4	$(\boldsymbol{hh}\|\boldsymbol{i})$	784.15	$(\boldsymbol{ii}\|\boldsymbol{l})$	2,353.88
OS1 OUT	$(\boldsymbol{hh}\|\boldsymbol{i})$	797.07	$(\boldsymbol{id}\|\boldsymbol{g})$	2,848.41
GHP	$(\boldsymbol{hh}\|\boldsymbol{i})$	6,619.07	$(\boldsymbol{ii}\|\boldsymbol{l})$	29,373.72
MD1	$(\boldsymbol{hh}\|\boldsymbol{i})$	420.86	$(\boldsymbol{dd}\|\boldsymbol{g})$	1,350.91
MD2	$(\boldsymbol{hh}\|\boldsymbol{i})$	552.66	$(\boldsymbol{ii}\|\boldsymbol{l})$	1,542.38
MD3	$(\boldsymbol{hh}\|\boldsymbol{i})$	672.98	$(\boldsymbol{ii}\|\boldsymbol{l})$	1,972.58
MD4	$(\boldsymbol{hh}\|\boldsymbol{i})$	1,195.10	$(\boldsymbol{ii}\|\boldsymbol{l})$	3,646.70
MD5	$(\boldsymbol{hh}\|\boldsymbol{i})$	1,046.47	$(\boldsymbol{ii}\|\boldsymbol{l})$	2,773.34
MD6	$(\boldsymbol{hh}\|\boldsymbol{i})$	678.63	$(\boldsymbol{ii}\|\boldsymbol{l})$	2,030.52
RYS1	$(\boldsymbol{hh}\|\boldsymbol{i})$	392.84	$(\boldsymbol{dd}\|\boldsymbol{g})$	1,348.88
RYS2	$(\boldsymbol{hh}\|\boldsymbol{i})$	501.73	$(\boldsymbol{ii}\|\boldsymbol{l})$	1,430.90
RYS3	$(\boldsymbol{hh}\|\boldsymbol{i})$	665.51	$(\boldsymbol{ii}\|\boldsymbol{l})$	1,959.06

been implemented in the conventional way. An individual FORTRAN 95 subroutine was created for every shell triplet up to $(\boldsymbol{hh}|\boldsymbol{i})$. The subroutines contain the loops over the primitive and the contracted Gaussians, the calculation of the necessary exponent- and center-dependent quantities, the evaluation of Boys functions (or the roots and weights for the Rys quadrature), the recursive build-up of angular momenta (or the quadrature for $\boldsymbol{l}_a$), and the transformations to the solid harmonic and contracted bases. The code generation based implementation is particularly useful for the exploitation of the fact that not all the intermediate integrals are needed for a given class when using the 6D recurrences Eqs. (2.1.2) to (2.1.3) and the 3D recurrence of Eq. (1.2.5), and the statements for calculating the unnecessary integrals are simply omitted from the code. For the 2D recursions of the RYS3 scheme this does not apply since the recursions for the x, y, and z directions are performed separately and all the components are needed in the recursion defined by Eq. (2.1.21). The calculation of the 2D integrals is vectorized for the roots of the Rys polynomials, and the quadrature for the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(L_c)}$ classes have been implemented utilizing the reduced multiplication scheme of Lindh and co-workers. [60] All the intermediate and target integrals are stored in one-index arrays. This is also a notable advantage of the generated codes since general implementations typically apply sparse multi-index arrays, which are easier to implement, but demand more memory, and are usually accessed in a less local manner, resulting in lower efficiency in the cache usage. The build-up of angular momenta and the solid harmonic transformation is performed for one class at a time, which means that the arrays for storing the intermediates of these tasks are of fixed length and the indices can be explicitly generated, eliminating the integer and memory operations for the calculation of indices.

A significant amount of vectorization can be achieved for the HRR and the solid harmonic transformation provided that the data are stored in the appropriate order. The HRR can be trivially vectorized for the components of $\boldsymbol{L}_c$ since Eq. (2.1.3) does not depend on the function in the ket. Systematic vectorization for the components of $\boldsymbol{l}_a$ is also possible if the component of $\boldsymbol{l}_b$ is the slowest changing property in the array. If the ordering of Cartesian components is as it is shown in Fig. 2.2, then the components of $\boldsymbol{l}_a$ can only be partially vectorized if z or y is raised in the angular momentum of $\boldsymbol{l}_b$, and fully if x is incremented, therefore, whenever it is possible, x should be raised by the HRR. For the GHP algorithm with a $(\boldsymbol{ps}|$ bra, where the target integrals are calculated directly from the one-center ones, Eq. (1.3.2) was vectorized in the same manner for the components of $\boldsymbol{L}_c$. For the solid harmonic transformation of one of the functions, the loops over all the (Cartesian or solid harmonic) components of the other two functions can only be vectorized if the components of the transformed function change most slowly. I found it to be efficient to rearrange the ordering of integrals before these highly vectorizable tasks. The sparsity of the solid harmonic transformation is fully exploited in the implementation, and the values of the coefficients in Eq. (1.1.6) are explicitly generated into the code. The approach where the HRR and the solid harmonic transformation for the bra are treated as one matrix multiplication by precalculating the combined transformation matrix [64], storing it in compressed sparse column format for a given bra, and reusing this matrix with a sparse matrix multiplication routine for the transformation of integrals, has also been considered. The experience was that performing the HRR separately step by step for each $\boldsymbol{l}_b$ with the vectorization scheme described above and exploiting that some components are unnecessary for the recursion is a more beneficial strategy. It should also be mentioned that the solid harmonic transformation of the ket side is always performed before the HRR since this makes the latter step less expensive.

The contraction of primitives can be treated in a vectorized manner without the rearrangement of data. For generally contracted functions, the multiplication with the coefficients in Eq. (1.1.7) is vectorized for all the necessary classes , e.g., for the construction of the integrals over all components of one of the χ_B functions in an *abc* scheme $N_{\chi_C} N_S$ number of integrals are treated simultaneously, where N_{χ_C} is the number of contracted functions centered on $\mathbf{C}$ and N_S is the number of integrals in the class (for algorithm IN) or in the necessary $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ classes (for algorithm OUT). For example, for a $(\boldsymbol{dd}|\boldsymbol{d})$ class $N_S = 5\times5\times5 = 125$ for algorithm IN and $N_S = 6\times1\times6+10\times1\times6+15\times1\times6 = 186$ for algorithm OUT because here we need the $(\boldsymbol{ds}|\boldsymbol{d})$, $(\boldsymbol{fs}|\boldsymbol{d})$, and $(\boldsymbol{gs}|\boldsymbol{d})$ classes for the HRR. It is also noteworthy that, at the contraction of the functions centered on $\mathbf{B}$, instead of performing the summation of Eq. (1.1.7) in the $N_a N_{\chi_B} N_{\chi_C} N_S$ long array used to store these partially contracted integrals (where N_a is the number of primitives centered on $\mathbf{A}$) it is more cache-friendly to do the summation in a buffer array of size $N_{\chi_C} N_S$, then to copy the data into the array that will be used for the contraction of primitives centered on $\mathbf{A}$.

The automated generation of the OS1 and RYS3 based subroutines was performed by a code generator software developed in FORTRAN 95. The main task of this program is to determine a recursion path for the target ERIs which does not use every possible component of every intermediate class. This is performed by algorithms similar to the schematic one presented in Algorithm 3, which illustrates how the backtracking of the recursions take place on the example of the HRR, Eq. (2.1.3). Take for example the buildup of a $(\boldsymbol{d}_{x^2}\boldsymbol{d}_{xy}|$ target bra side. If this is part of the target class, then this bra side is necessary, therefore it is marked in an array used for this purpose. We have to decide which spatial component we want to increase to build up the $\boldsymbol{d}_{xy}$ function. For the reasons discussed above this is chosen to be the x component if it is greater than 0 in $\boldsymbol{l}_b$. If not, we chose the y component if it is not 0, else the z component will be incremented. For Eq. (2.1.2), and other recursions where the target ERI depends on angular momentum lower by 2 compared to the target one, it is first checked if one of the components of the angular momentum vector is 1, since fewer number of operations are necessary in this case if we choose to increment this component. At the end, the bra sides necessary to build up $(\boldsymbol{d}_{x^2}\boldsymbol{d}_{xy}|$ [that is, $(\boldsymbol{f}_{x^3}\boldsymbol{p}_y|$ and $(\boldsymbol{d}_{x^2}\boldsymbol{p}_y|$] are marked as necessary, and the loops continue. After Algorithm 3 is finished, another subroutine is called, which goes through the loops in reverse order, and prints a line of code that uses Eq. (2.1.3) to build up a given bra side if it was marked as required.

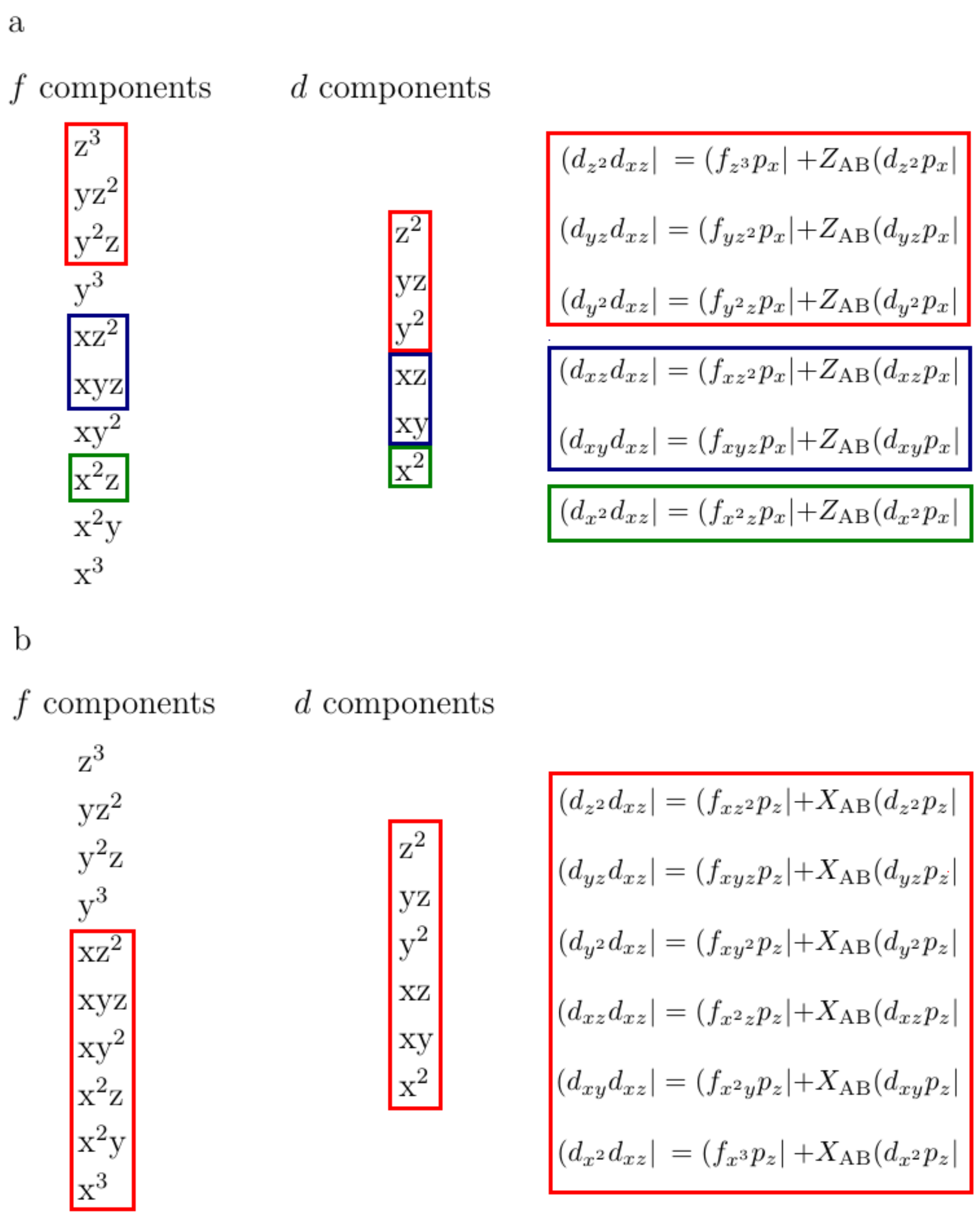
a
f components
d components
z^3
yz^2
y^{2}z
y^3
xz^2
xyz
xy^2
x^{2}z
x^{2}y
x^3
z^2
yz
y^2
xz
xy
x^2
$(d_{z^2}d_{xz}| = (f_{z^3}p_x| + Z_{\mathrm{AB}}(d_{z^2}p_x|$
$(d_{yz}d_{xz}| = (f_{yz^2}p_x| + Z_{\mathrm{AB}}(d_{yz}p_x|$
$(d_{y^2}d_{xz}| = (f_{y^2z}p_x| + Z_{\mathrm{AB}}(d_{y^2}p_x|$
$(d_{xz}d_{xz}| = (f_{xz^2}p_x| + Z_{\mathrm{AB}}(d_{xz}p_x|$
$(d_{xy}d_{xz}| = (f_{xyz}p_x| + Z_{\mathrm{AB}}(d_{xy}p_x|$
$(d_{x^2}d_{xz}| = (f_{x^2z}p_x| + Z_{\mathrm{AB}}(d_{x^2}p_x|$
b
f components
d components
z^3
yz^2
y^{2}z
y^3
xz^2
xyz
xy^2
x^{2}z
x^{2}y
x^3
z^2
yz
y^2
xz
xy
x^2
$(d_{z^2}d_{xz}| = (f_{xz^2}p_z| + X_{\mathrm{AB}}(d_{z^2}p_z|$
$(d_{yz}d_{xz}| = (f_{xyz}p_z| + X_{\mathrm{AB}}(d_{yz}p_z|$
$(d_{y^2}d_{xz}| = (f_{xy^2}p_z| + X_{\mathrm{AB}}(d_{y^2}p_z|$
$(d_{xz}d_{xz}| = (f_{x^2z}p_z| + X_{\mathrm{AB}}(d_{xz}p_z|$
$(d_{xy}d_{xz}| = (f_{x^2y}p_z| + X_{\mathrm{AB}}(d_{xy}p_z|$
$(d_{x^2}d_{xz}| = (f_{x^3}p_z| + X_{\mathrm{AB}}(d_{x^2}p_z|$

FIGURE 2.2: Two possible ways of calculating integrals with a $(\boldsymbol{dd}\,|$ bra side and $\boldsymbol{l}_b = (1, 0, 1)$ by a) incrementing z and b) incrementing x in $\boldsymbol{l}_b$. The indices for the Cartesian components increase as we proceed from top to bottom in the columns for the f and d shells above. The operations which can be vectorized are highlighted by boxes of various colors. In the present implementation, incrementing x is always better suited for vectorization. The ket side of the integrals are not shown since the HRR equation is invariant to the function in the ket.

Algorithm 3 Overview of the algorithm backtracking the HRR

```
Loop for $l_b$ from $L_b$ to 1
    Loop for possible $\boldsymbol{l}_b$ vectors
        Loop for $l_a$ from $L_a + l_b$ to $L_a$
            Loop for possible $\boldsymbol{l}_a$ vectors
                If $(\boldsymbol{l}_a \boldsymbol{l}_b|$ is necessary, then
                    Decide which component (x, y, or z) will
                      be increased to build up $(\boldsymbol{l}_a \boldsymbol{l}_b|$
                      (denoted as $\sigma$ below)
                    Mark $([\boldsymbol{l}_a{+}\mathbf{1}_\sigma][\boldsymbol{l}_b{-}\mathbf{1}_\sigma]|$ as necessary
                    Mark $(\boldsymbol{l}_a[\boldsymbol{l}_b{-}\mathbf{1}_\sigma]|$ as necessary
```

The generated subroutines are stored in different files. The file `dfint_triplets.f` contains driver subroutines that evaluate the ERIs over AOs for each shell triplet. There are two such routines for each ERI class, one that assumes a contracted DF basis set for the atom in the ket side, and another one that is optimized for uncontracted auxiliary bases and skips the contraction step entirely, reducing the memory requirement and the memory operation count of the code. Depending on the angular momenta involved, these driver subroutines might call other generated subroutines. The file `intsub.f` contains subroutines that calculate the final solid harmonic primitive integrals for IN routes, and the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ intermediates for the OUT schemes. These subroutines are not called when $L_a + L_b = 0$, $L_a + L_b \leq 4$ and $L_c = 0$, or for $(\boldsymbol{ps}|$, $(\boldsymbol{pp}|$, $(\boldsymbol{ds}|$ bras with $L_c = 1, 2$, because according to my experiences it is more efficient to generate the necessary instructions directly into the driver subroutine rather than calling another subroutine that contains them in these low angular momenta cases. The file `brasub.f` contains subroutines that produce $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{L_c}$ type intermediates either by the OS VRR, Eq. (2.1.2), or by Rys quadrature. These subroutines are called when $L_a + L_b > 4$. The final source file, `hrrspher.f`, contains subroutines performing the solid harmonic transformation and the HRR, Eq. (2.1.3), which are called by the driver subroutines of `dfint_triplets.f` in the case of OUT schemes. The call list of the driver subroutines that calls the appropriate subroutine for a given shell triplet is also code generated.

The implementation of ERIs also utilizes a coarse-grained OpenMP parallelization for the innermost atomic loop. To demonstrate the efficiency of the generated implementation, a subroutine that uses the OS1 scheme for arbitrary angular momenta has also been coded. Here, the recursions of Eqs. (2.1.2) and (1.7.8) are performed by general loops, and the intermediates are stored in a two-index array. The HRR and the solid harmonic transformation steps are done at the contracted level with a sparse matrix multiplication routine, which is applied to the solid harmonic transformation of the ket and the combined HRR and solid harmonic transformation of the bra [64] as described above.

2.4 Performance tests

In this section the wall time performances of the implemented algorithms measured using a single core of a 2-core 3.00 GHz Intel Xeon E3110 CPU are presented. The generated subroutines were compiled with the Intel Fortran compiler using the highest level of optimization. Measurements were carried out for penicillin [103] (PEN) and two DNA systems with one (DNA_1) and two (DNA_2) adenine-thymine base pairs [104]. The threshold ε for contracted integrals was set to 10^{-10} E_h in all of the calculations. Only the timings for DNA_2 with the cc-pVTZ basis set are presented here. The results for the other measurements, which show that the conclusions gained hold for all the investigated systems, can be found in the Supplementary Material of Ref. 95 [101]. Cache simulations were performed for hydrogen peroxide (R_{OO} = 2.7514 bohr, R_{HO} = 1.8274 bohr, $\sphericalangle_{\mathrm{HOO}} = 102.32°$, dihedral angle = 115.89°) with the Valgrind program package [105] supposing a three-level CPU cache structure which is common these days: 64 kB of level 1 (L1, 32 kB for both data and instructions), 256 kB of level 2, and 4MB of level 3 (last level, LL) cache. In the simulations, an L1 miss means that the data or instructions have not been found in the first level, while an LL miss indicates that no copy of the requested information can be found in the cache at all. Note that the number of L1 misses also contains the LL misses.

Fig. 2.3 shows the difference between the pPRE1 and pPRE2 primitive prescreening schemes in the case of the IN-*abc* algorithm. The pPRE2 method saves entering the loop over c and the prescreening for each c at the price that classes containing integrals of insignificant absolute values that would be screened out with the pPRE1 scheme are also computed. With the *abc* loop order, the pPRE2 approach is clearly more efficient. The difference between the performance of the two prescreening schemes, as well as the significance of primitive prescreening, shrinks with the decrease in the number of primitive functions. On the other hand, from Fig. 2.4 we see that the pPRE1 prescreening is more economical in the case of a *cab* scheme since the Schwarz screening already throws out most of the shell pairs where no b gives a significant contribution. The figures presenting the timings for the various cPRE algorithms can be found in the Supplementary Material of Ref. 95 [101]. The cPRE type of screening has less effect, and for triplets that do not require either the HRR or the solid harmonic transformation, it merely saves the writing of integrals into their final storing array. As the former two tasks become more significant, the cPRE screening gets more beneficial, especially with higher basis sets, where there

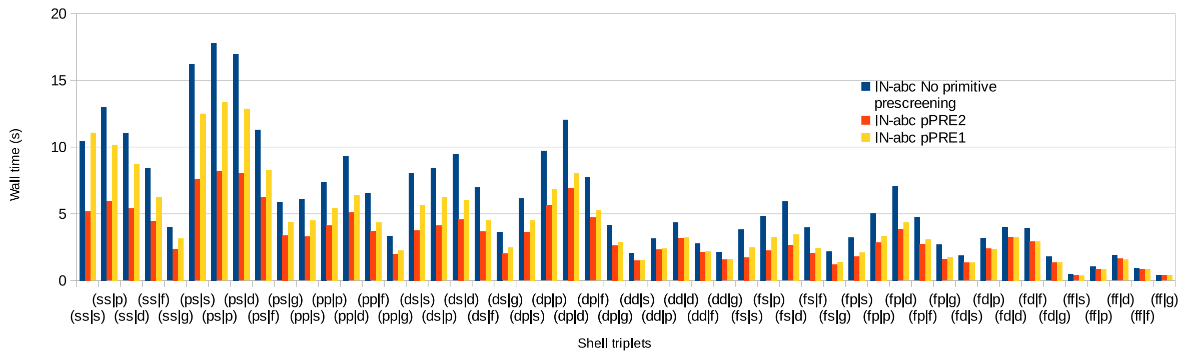

FIGURE 2.3: Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA$_2$ molecule with the cc-pVTZ basis set applying the OS1-IN-*abc* algorithm with various prescreening strategies

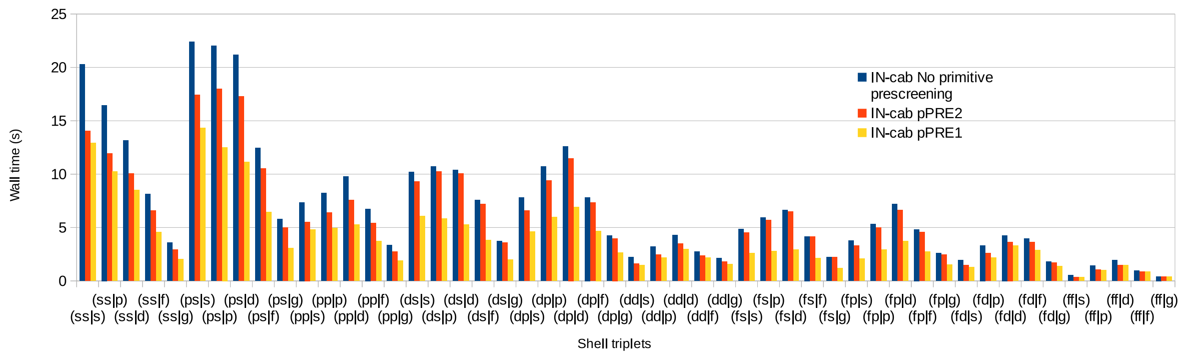

FIGURE 2.4: Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA$_2$ molecule with the cc-pVTZ basis set applying the OS1-IN-*cab* algorithm with various prescreening strategies

are more contracted functions for higher angular momenta. For the OUT-*abc* scheme the lookup of the integrals of highest absolute value (cPRE1 and cPRE2) is preferred over the estimation of this quantity (cPRE3). The cPRE1 and cPRE2 schemes have very similar performance, with cPRE2 being slightly more efficient. The same tendencies can be observed with the OUT-*cab* algorithm, where cPRE1 is the more efficient method. It can be concluded that for the *abc* primitive loop order the pPRE2 and cPRE2 are the prescreening schemes of choice, while for the *cab* algorithms the pPRE1 and cPRE1 screenings are preferred.

The accuracy of the pPRE2 and cPRE2 schemes have also been measured. Table 2.4 presents the total number of primitive integrals, the number of integrals that survive the pPRE2 type prescreening, and the number of integrals for the three test systems which are greater than the tolerance for primitive integrals when the screening value defined by the rightmost side of Eq. (2.1.30) would not indicate so. We can see that the method is very efficient, eliminating 49-64% of the required primitive integrals. Relative to the total number of integrals, only 0.12-0.32% are wrongly discarded. However, the maximum absolute value of these reaches 10^{-6} and 10^{-5} E_h in particular cases for the TZ and QZ basis sets, respectively. Hence, the effect of neglecting these integrals on the DF-HF energy was investigated. DF direct-SCF calculations were performed for the systems in Table 2.4 using $\varepsilon = 10^{-10}$ E_h, an SCF convergence threshold of 10^{-6} E_h, and local fitting domains for the exchange contribution with a tolerance of 1 E_h [1]. Table 2.5 shows the energies acquired with the pPRE2 scheme in comparison to those obtained with a code where only the primitive integrals not reaching the primitive tolerance were set to zero. We see that the differences between the two values reach the μE_h level only for the biggest systems. The precision of the cPRE2 type screening is presented in Table 2.6. The conclusions are similar to those for the primitive integrals. Finally, Table 2.7 compares the efficiency of the integral screening for the cc-pVXZ basis sets with the aug-cc-pVXZ bases for the DNA$_2$ molecule. The diffuse basis sets contain an extra very diffuse AO for each angular momentum. The table shows the effect of the screening on both the shell triplet and the integral class levels. The shell triplet level screening applies the Schwarz inequality and the approach of Ref. 100, while the integral class level screening applies the pPRE2 scheme. The results show that at the shell triplet level the screening is a lot less efficient with the aug-cc-pVXZ basis sets. This is not surprising since the diffuse AOs make it more probable that at least one integral of a shell triplet will be significant. The application of the pPRE2 scheme reduces the gap regarding the prescreening efficiency to 25-15 % for the DZ-QZ basis sets.

TABLE 2.4: Efficiency of the pPRE2 prescreening scheme with the cc-pVXZ basis sets. The values refer to the number of primitive integrals for the test system (see text).

	X					
	T			Q		
	Penicillin	DNA$_1$	DNA$_2$	Penicillin	DNA$_1$	DNA$_2$
All	1631398360	3851964314	32718325856	6919713714	16338904803	132719112875
Kept	834239122	1853159796	12055149629	3494103638	7780344806	48277844649
Wrongly discarded	2482618	5226489	39795743	22176135	47436295	313992101

TABLE 2.5: DF-HF energies (E_h) obtained with the pPRE2 type and an exact prescreening (see text).

	T			Q		
	Penicillin	DNA$_1$	DNA$_2$	Penicillin	DNA$_1$	DNA$_2$
pPRE2	-1497.470829	-1753.706235	-4487.551555	-1497.549519	-1753.827809	-4487.837089
Exact	-1497.470829	-1753.706235	-4487.551555	-1497.549519	-1753.827810	-4487.837098

The wall times measured for the shell triplets with the four variants of the OS1 algorithm, using the most efficient prescreening methods, are shown in Fig. 2.5. For triplets containing small angular momenta, the *cab* schemes are inefficient, even without primitive prescreening (see also Figs. 2.3 and 2.4). The reason for this is that the arrays that become smaller with a *cab* algorithm are already too short in these cases. For example, the length of the buffer array used for the contraction of functions centered on **B**

TABLE 2.6: Efficiency of the cPRE2 prescreening scheme with the cc-pVXZ basis sets. The values refer to the number of contracted integrals for the test system (see text).

	X					
	T			Q		
	Penicillin	DNA_1	DNA_2	Penicillin	DNA_1	DNA_2
All	767220662	1807820417	14321566043	3726010771	8773887044	66791086714
Kept	641203489	1470663279	10543869024	2771676289	6244416232	41002457733
Wrongly discarded	1586520	3792300	80023753	10942601	24712632	467635737

TABLE 2.7: Efficiency of the shell triplet level screening and the primitive level screening (using the pPRE2 scheme) of the three-center ERI evaluation for the DNA_2 molecule with the cc-pVXZ and the aug-cc-pVXZ (X=D,T,Q) basis sets. The values give the wall time expressed as percentage of the screening-free calculation. "Shell triplet" refers to only shell triplet level screening (see text).

Basis	DZ	aug-DZ	TZ	aug-TZ	QZ	aug-QZ
Shell triplet	49.49	83.57	45.88	77.12	39.81	68.10
pPRE2	25.76	48.59	25.21	46.77	21.70	38.69

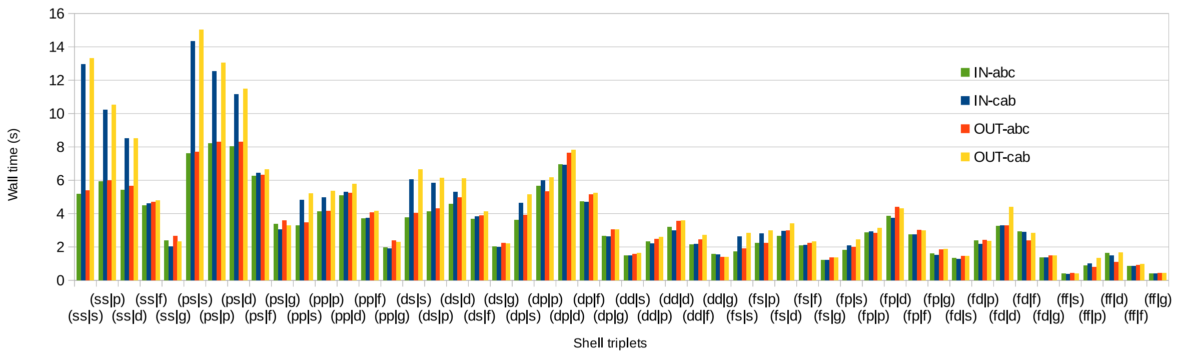

FIGURE 2.5: Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA_2 molecule with the cc-pVTZ basis set applying the four OS1 algorithms with the most efficient prescreening strategies

for $(\boldsymbol{ss}|\boldsymbol{s})$ is N_{χ_C} and 1 using an *abc* and a *cab* scheme, respectively. Here, applying the *cab* loop order ruins the vectorization for the primitive contraction. This effect loses its importance with the growth of $\boldsymbol{L}_c$ since N_S becomes bigger and N_{χ_C} becomes smaller. The difference between the *abc* and *cab* schemes grows when using basis sets of higher cardinal number, because of the higher number of contracted functions. The IN algorithms generally perform better than the OUT ones. One of the reasons is the apparent superiority of the pPRE-type screening, which lessens the amount of work for the HRR and solid harmonic transformation steps using the IN schemes. I must note, however, that only the $\boldsymbol{s}$ and $\boldsymbol{p}$ shells are contracted in the considered basis sets, making the OUT route theoretically more efficient only in shell triplets containing at least one such shell.

The timings can be better interpreted inspecting the results of the cache performance simulations. The cumulated results for all the shell triplets in the TZ basis are

Table 2.8: Cache performance simulation results for H_2O_2 with the cc-pVTZ basis set

Event	Algorithm			
	IN-*abc*	IN-*cab*	OUT-*abc*	OUT-*cab*
L1 instruction fetch miss	687995	720295	665954	820972
LL instruction fetch miss	576727	582575	641900	666989
L1 data read miss	219741	192668	252112	202708
LL data read miss	199655	191036	200703	199053
L1 data write miss	484132	385047	552062	407074
LL data write miss	482194	383336	544077	404681

presented in Table 2.8, while the results with the other basis sets can be found in the Supplementary Material of Ref. 95 [101]. We see that the number of level 1 instruction fetch misses (L1Is) is lower for the OUT-*abc* scheme than for the IN-*abc*, but a higher percentage of these are also last level misses. This is because with an IN algorithm the calculation of primitive integrals and the conversion into the solid harmonic Gaussian basis are done continuously step by step inside the primitive loops, while in the OUT case this procedure is divided into two parts with two separate loop structures, making it more friendly for the instruction cache for higher angular momenta, where the generated codes are lengthy. This effect is more pronounced with basis sets of higher cardinal number, where the angular momenta are higher and the loops over primitive and contracted functions perform more cycles. With the QZ and 5Z bases we can observe the same for OUT-*cab*: the number of L1Is is smaller than for the IN schemes, but higher than for the OUT-*abc* since all the calculations take place in the loop over c, making the reuse of instructions less temporally local (that is, the same tasks are not performed as frequently as they would be with the loop over c being the innermost one). For this reason, the *abc* schemes are always more friendly to the instruction cache. This aspect of the performance is the reason why the OUT schemes are sometimes more efficient for shell triplets we would not expect theoretically, for example, for the $(\boldsymbol{fd}|\boldsymbol{f})$ and $(\boldsymbol{ff}|\boldsymbol{d})$ cases with the TZ basis, and also explains why the performance of this approach improves with basis sets including functions of higher angular momentum. As anticipated from the sizes of the arrays used for the primitive contraction, the IN algorithms produce fewer data misses of both the read and write kind, and the *cab* loop order is beneficial in this aspect. This difference also grows with the cardinal number of the basis sets, and is more significant for write misses since the read operations are usually carried out from arrays that have been written in a previous calculation step.

Fig. 2.6 compares the efficiency of the OS1 and RYS3 schemes. For each shell triplet, the selected algorithmic approach was the one that best performed according to

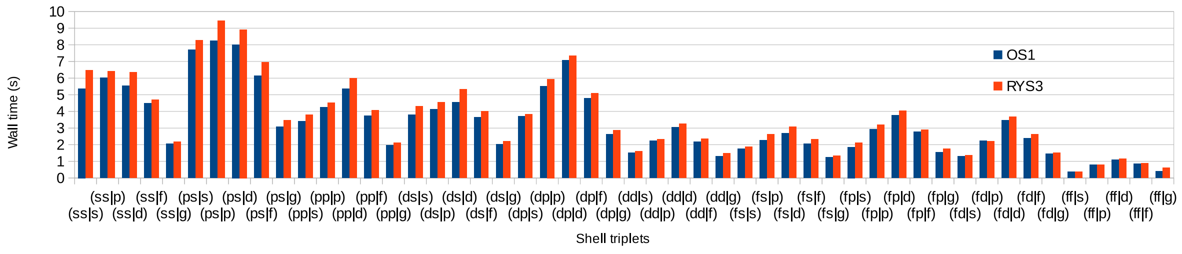

FIGURE 2.6: Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA$_2$ molecule with the cc-pVTZ basis set applying the most efficient OS1 and RYS3 algorithms

Fig. 2.5, keeping in mind that the most efficient combination of the IN-OUT and *abc*-*cab* approaches for the OS1 scheme is also the most efficient one for the RYS3 since the OS1 and RYS3 schemes do not differ in any part that depends on using the IN-OUT or *abc*-*cab* approaches. While the performances fall close, the OS1 scheme is superior in almost every case. The differences are more pronounced for the shell triplets with small angular momenta in the bra. The advantage of using OS1 becomes larger for the shell triplets where the number of Rys quadrature points is over 5. In these cases, the roots and weights are calculated by applying Wheeler's algorithm [66] and Golub's matrix method [67], while otherwise the less expensive schemes proposed by King and Dupuis [34] are employed. The disagreement between the timings and the FLOP estimates must come from the task that is not estimated by the operation counts, that is, the evaluation of Boys functions and the roots and weights for the Rys quadrature. In some cases, the RYS3 scheme is still slightly more efficient, e.g., for the $(\boldsymbol{fd}|\boldsymbol{p})$ and $(\boldsymbol{gd}|\boldsymbol{p})$ shell triplets. The GHP scheme is competitive for the implemented cases (see Sect. 2.3) with the 5Z basis, where the degree of contraction is the highest. For smaller basis sets, for the $(\boldsymbol{ps}|\boldsymbol{p})$ triplet GHP performs slightly better than OS1 since here the number of integrals to be contracted, that is, the number of integrals included in the scaled classes $(\Omega^{\bar{\mathbf{0}}}_{\mathbf{0},\mathbf{0}}|\bar{\mathbf{1}})_{1,1}$ and $(\Omega^{\bar{\mathbf{1}}}_{\mathbf{0},\mathbf{0}}|\bar{\mathbf{1}})_{0,1}$ needed for Eq. (1.3.2) is the same as the number of integrals to be contracted in the OS1 scheme, and all of the functions are contracted. The application of the *cab* loop order on the $(\boldsymbol{ps}|\boldsymbol{g})$ and $(\boldsymbol{ps}|\boldsymbol{f})$ triplets makes the GHP algorithm perform better for these cases than the other ones with the TZ and the QZ basis, respectively.

Fig. 2.7 presents the number of AO ERIs for the DNA$_2$ molecule and also the average wall time spent on the evaluation of one AO integral using the OS1-IN-*abc* algorithm without screening. As the angular momenta increase, the computational expense grows for the recurrence relations and reduces for the contraction of primitives. This explains why the average time necessary for one ERI is in the same order of magnitude for most shell triplets. The ERIs with an $(\boldsymbol{ss}|$ bra side are an exception from this since these integrals are computed over a larger number of primitive functions. For example, the

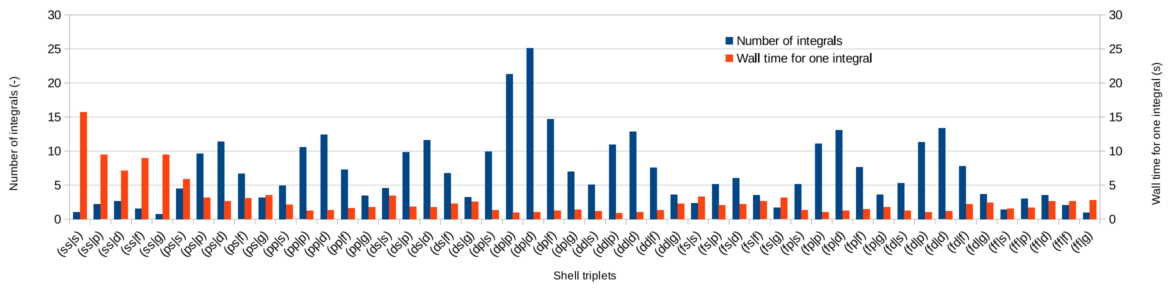

Figure 2.7: Number of AO ERIs and the average wall time spent on one AO ERI for the DNA$_2$ molecule with the cc-pVTZ basis set. The number of integrals is divided by 10^8, while the average time in seconds is multiplied by 10^8.

primitive basis for carbon consists of 10 Gaussians for the $\boldsymbol{s}$ angular momentum, while 5 for $\boldsymbol{p}$. Note, however, that according to Fig. 2.3 the primitive level prescreening is very effective for these shell triplets. Comparing the number of integrals per shell triplet with the cumulative timings on Fig. 2.3 it can be seen that, without screening, the highest amount of time is not spent on the shell triplets with the highest amount of ERIs but lower angular momenta where the contraction of primitives is the most demanding step of the computation.

As it was pointed out, the relative performances of the discussed approaches depend on the number of functions and the degree of contraction, therefore on the applied basis set itself. For the three test molecules I investigated it was my experience that the best algorithm for a given shell triplet with a given basis is mostly independent of the calculated system. Based on the measurements with the cc-pVXZ bases for first row elements, in Table 2.9 recommendations are presented for the algorithms for the shell triplets up to $(\boldsymbol{hh}|\boldsymbol{i})$. The list compiled in Table 2.9 was composed by selecting the schemes that are the most beneficial ones for the TZ and the QZ basis sets, because such bases are used most frequently in DF calculations. The best algorithm for the triplets is the same with both basis sets for most of the cases. As we can see, even though the considered basis sets have the similarity that only the $\boldsymbol{s}$ and $\boldsymbol{p}$ shells are contracted, the increase of the number of functions and the level of contraction makes the *cab* and OUT schemes more beneficial with the bigger bases.

Table 2.9: Recommended algorithms for the various shell triplets

Shell triplet	Algorithm	Shell triplet	Algorithm	Shell triplet	Algorithm
$(\boldsymbol{ss}|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{fp}|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{gg}|\boldsymbol{s})$	OS1-OUT-abc
$(\boldsymbol{ss}|\boldsymbol{p})$	OS1-IN-abc	$(\boldsymbol{fp}|\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{gg}|\boldsymbol{p})$	OS1-IN-cab
$(\boldsymbol{ss}|\boldsymbol{d})$	OS1-IN-abc	$(\boldsymbol{fp}|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{gg}|\boldsymbol{d})$	OS1-IN-cab
$(\boldsymbol{ss}|\boldsymbol{f})$	OS1-IN-abc	$(\boldsymbol{fp}|\boldsymbol{f})$	OS1-IN-cab	$(\boldsymbol{gg}|\boldsymbol{f})$	OS1-IN-cab
$(\boldsymbol{ss}|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{fp}|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{gg}|\boldsymbol{g})$	OS1-IN-cab

$(\boldsymbol{ss}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{fp}\|\boldsymbol{h})$	OS1-OUT-cab	$(\boldsymbol{gg}\|\boldsymbol{h})$	OS1-OUT-cab
$(\boldsymbol{ss}\|\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{fp}\|\boldsymbol{i})$	OS1-OUT-cab	$(\boldsymbol{gg}\|\boldsymbol{i})$	OS1-OUT-abc
$(\boldsymbol{ps}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{fd}\|\boldsymbol{s})$	OS1-IN-cab	$(\boldsymbol{hs}\|\boldsymbol{s})$	RYS3-IN-abc
$(\boldsymbol{ps}\|\boldsymbol{p})$	GHP-abc	$(\boldsymbol{fd}\|\boldsymbol{p})$	RYS3-IN-cab	$(\boldsymbol{hs}\|\boldsymbol{p})$	OS1-IN-abc
$(\boldsymbol{ps}\|\boldsymbol{d})$	OS1-IN-abc	$(\boldsymbol{fd}\|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{hs}\|\boldsymbol{d})$	OS1-IN-abc
$(\boldsymbol{ps}\|\boldsymbol{f})$	OS1-IN-abc	$(\boldsymbol{fd}\|\boldsymbol{f})$	OS1-OUT-abc	$(\boldsymbol{hs}\|\boldsymbol{f})$	OS1-IN-abc
$(\boldsymbol{ps}\|\boldsymbol{g})$	GHP-cab	$(\boldsymbol{fd}\|\boldsymbol{g})$	OS1-OUT-abc	$(\boldsymbol{hs}\|\boldsymbol{g})$	OS1-IN-abc
$(\boldsymbol{ps}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{fd}\|\boldsymbol{h})$	OS1-IN-abc	$(\boldsymbol{hs}\|\boldsymbol{h})$	OS1-IN-cab
$(\boldsymbol{ps}\|\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{fd}\|\boldsymbol{i})$	OS1-OUT-abc	$(\boldsymbol{hs}\|\boldsymbol{i})$	OS1-IN-cab
$(\boldsymbol{pp}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{ff}\|\boldsymbol{s})$	OS1-IN-cab	$(\boldsymbol{hp}\|\boldsymbol{s})$	RYS3-IN-cab
$(\boldsymbol{pp}\|\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{ff}\|\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{hp}\|\boldsymbol{p})$	OS1-IN-cab
$(\boldsymbol{pp}\|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{ff}\|\boldsymbol{d})$	OS1-OUT-abc	$(\boldsymbol{hp}\|\boldsymbol{d})$	OS1-IN-abc
$(\boldsymbol{pp}\|\boldsymbol{f})$	OS1-IN-cab	$(\boldsymbol{ff}\|\boldsymbol{f})$	OS1-IN-cab	$(\boldsymbol{hp}\|\boldsymbol{f})$	OS1-OUT-abc
$(\boldsymbol{pp}\|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{ff}\|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{hp}\|\boldsymbol{g})$	OS1-OUT-abc
$(\boldsymbol{pp}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{ff}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{hp}\|\boldsymbol{h})$	OS1-IN-cab
$(\boldsymbol{pp}\|\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{ff}\|\boldsymbol{i})$	OS1-OUT-cab	$(\boldsymbol{hp}\|\boldsymbol{i})$	OS1-IN-cab
$(\boldsymbol{ds}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{gs}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{hd}\|\boldsymbol{s})$	RYS3-IN-cab
$(\boldsymbol{ds}\|\boldsymbol{p})$	OS1-IN-abc	$(\boldsymbol{gs}\|\boldsymbol{p})$	OS1-IN-abc	$(\boldsymbol{hd}\|\boldsymbol{p})$	OS1-OUT-abc
$(\boldsymbol{ds}\|\boldsymbol{d})$	OS1-IN-abc	$(\boldsymbol{gs}\|\boldsymbol{d})$	OS1-IN-abc	$(\boldsymbol{hd}\|\boldsymbol{d})$	OS1-OUT-cab
$(\boldsymbol{ds}\|\boldsymbol{f})$	OS1-IN-abc	$(\boldsymbol{gs}\|\boldsymbol{f})$	OS1-IN-abc	$(\boldsymbol{hd}\|\boldsymbol{f})$	OS1-OUT-cab
$(\boldsymbol{ds}\|\boldsymbol{g})$	OS1-IN-abc	$(\boldsymbol{gs}\|\boldsymbol{g})$	OS1-IN-abc	$(\boldsymbol{hd}\|\boldsymbol{g})$	OS1-OUT-abc
$(\boldsymbol{ds}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{gs}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{hd}\|\boldsymbol{h})$	OS1-OUT-cab
$(\boldsymbol{ds}\|\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{gs}\|\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{hd}\|\boldsymbol{i})$	OS1-OUT-cab
$(\boldsymbol{dp}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{gp}\|\boldsymbol{s})$	OS1-IN-cab	$(\boldsymbol{hf}\|\boldsymbol{s})$	OS1-OUT-abc
$(\boldsymbol{dp}\|\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{gp}\|\boldsymbol{p})$	OS1-IN-cab	$(\boldsymbol{hf}\|\boldsymbol{p})$	OS1-OUT-abc
$(\boldsymbol{dp}\|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{gp}\|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{hf}\|\boldsymbol{d})$	OS1-IN-cab
$(\boldsymbol{dp}\|\boldsymbol{f})$	OS1-IN-cab	$(\boldsymbol{gp}\|\boldsymbol{f})$	OS1-OUT-abc	$(\boldsymbol{hf}\|\boldsymbol{f})$	OS1-IN-cab
$(\boldsymbol{dp}\|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{gp}\|\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{hf}\|\boldsymbol{g})$	OS1-IN-cab
$(\boldsymbol{dp}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{gp}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{hf}\|\boldsymbol{h})$	OS1-OUT-cab
$(\boldsymbol{dp}\|\boldsymbol{i})$	OS1-OUT-cab	$(\boldsymbol{gp}\|\boldsymbol{i})$	OS1-OUT-abc	$(\boldsymbol{hf}\|\boldsymbol{i})$	OS1-OUT-abc
$(\boldsymbol{dd}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{gd}\|\boldsymbol{s})$	OS1-IN-cab	$(\boldsymbol{hg}\|\boldsymbol{s})$	OS1-OUT-abc
$(\boldsymbol{dd}\|\boldsymbol{p})$	OS1-IN-cab	$(\boldsymbol{gd}\|\boldsymbol{p})$	OS1-IN-cab	$(\boldsymbol{hg}\|\boldsymbol{p})$	OS1-IN-cab
$(\boldsymbol{dd}\|\boldsymbol{d})$	OS1-IN-cab	$(\boldsymbol{gd}\|\boldsymbol{d})$	OS1-OUT-abc	$(\boldsymbol{hg}\|\boldsymbol{d})$	OS1-IN-cab
$(\boldsymbol{dd}\|\boldsymbol{f})$	OS1-IN-cab	$(\boldsymbol{gd}\|\boldsymbol{f})$	OS1-OUT-abc	$(\boldsymbol{hg}\|\boldsymbol{f})$	OS1-OUT-cab
$(\boldsymbol{dd}\|\boldsymbol{g})$	OS1-OUT-abc	$(\boldsymbol{gd}\|\boldsymbol{g})$	OS1-OUT-abc	$(\boldsymbol{hg}\|\boldsymbol{g})$	OS1-OUT-cab
$(\boldsymbol{dd}\|\boldsymbol{h})$	OS1-OUT-cab	$(\boldsymbol{gd}\|\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{hg}\|\boldsymbol{h})$	OS1-OUT-cab
$(\boldsymbol{dd}\|\boldsymbol{i})$	OS1-OUT-cab	$(\boldsymbol{gd}\|\boldsymbol{i})$	OS1-OUT-cab	$(\boldsymbol{hg}\|\boldsymbol{i})$	OS1-OUT-abc
$(\boldsymbol{fs}\|\boldsymbol{s})$	OS1-IN-abc	$(\boldsymbol{gf}\|\boldsymbol{s})$	RYS3-IN-cab	$(\boldsymbol{hh}\|\boldsymbol{s})$	OS1-IN-cab

$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{p})$	OS1-OUT-abc	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{p})$	OS1-IN-cab
$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{d})$	OS1-IN-abc	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{d})$	OS1-OUT-abc	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{d})$	OS1-IN-cab
$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{f})$	OS1-IN-abc	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{f})$	OS1-OUT-abc	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{f})$	OS1-OUT-cab
$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{g})$	OS1-IN-cab	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{g})$	OS1-OUT-cab
$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{h})$	OS1-IN-cab	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{h})$	OS1-IN-abc	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{h})$	OS1-OUT-abc
$(\boldsymbol{f}\boldsymbol{s}\vert\boldsymbol{i})$	OS1-IN-cab	$(\boldsymbol{g}\boldsymbol{f}\vert\boldsymbol{i})$	OS1-OUT-abc	$(\boldsymbol{h}\boldsymbol{h}\vert\boldsymbol{i})$	OS1-OUT-cab

2.5 Benchmark calculations

To demonstrate the efficiency of the implementation based on the above recommendation, Table 2.10 presents the wall times measured for the evaluation of three-center ERIs for test systems of various size, namely penicillin [103], DNA fragments containing 1 and 4 adenine-thymine base pairs [106] (DNA_1 and DNA_4, respectively), indinavir [107], angiotensin II [108] and a halloysite clay structure [109]. The measurements were carried out using 8 cores of a 3.00 GHz Intel Xeon E5-1660 CPU. A constant speedup of about 3 was experienced compared to the general purpose routine using the OS1 scheme, which shows that we can gain an efficient implementation optimized for each shell triplet separately. Note that three-center ERIs can also be easily computed with the algorithms developed for four-center ones constraining two of the four centers to be coincident. Since many quantum chemistry software packages evaluate three-center Coulomb integrals in this way, it is instructive to compare the speed of an explicitly three-center code to that of a four-center one for three-center ERIs. Therefore, the new three-center code and the previous OS-based four-center integral program in the Mrcc program suite [110] was compared and it was found that the former program is roughly twice faster than the latter one. Both the general purpose and the code generated subroutines incorporate the efficient primitive contraction procedure discussed in Subsect. 2.1.5 and Sect. 2.3. This also contributes to a speedup of a factor of two compared to using our previous implementation utilizing only the nested loop structure. Finally, Fig. 2.8 shows the efficiency of our coarse-grained parallelization scheme compared to the ideal, linear parallelization. So far we have not considered other, more elaborate techniques to parallelize the ERI code, such as load balancing [111], but taking into account that the optimized implementation of Ref. 111 achieves 90 % parallelization efficiency for 6 cores, and from Fig. 2.8 this is roughly 75 % in our case, we can consider our implementation relatively efficient.

TABLE 2.10: Wall times of three-center ERI calculations in minutes measured for various test systems with the cc-pVXZ basis sets. N+M denotes the total number of ordinary basis functions and fitting functions.

Test system	X							
	D		T		Q		5	
	Time	N+M	Time	N+M	Time	N+M	Time	N+M
Penicillin	0.008	430+2136	0.022	946+2478	0.088	1864+3504	0.372	3178+5033
DNA$_1$	0.016	625+3071	0.049	1428+3575	0.201	2735+5087	0.883	4670+7351
Indinavir	0.033	865+4231	0.118	2008+4965	0.492	3885+7167	2.251	6680+10471
Angiotensin II	0.104	1405+6883	0.380	3244+8055	1.609	6255+11571	7.245	10730+16843
DNA$_4$	0.474	2746+19820	1.777	6192+15794	8.307	11774+22202	33.174	20012+31744
Halloysite	1.306	3700+19820	4.607	7970+22435	19.854	14855+30280	68.447	24985+41510

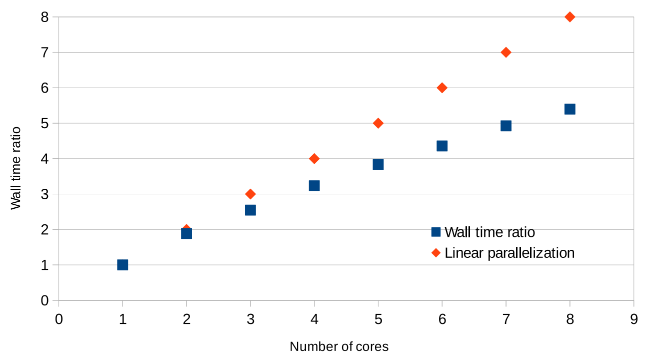

FIGURE 2.8: The wall time on one CPU core divided by the wall time on n cores compared to the theoretical linear scaling

2.6 Conclusions

The Obara–Saika, McMurchie–Davidson, Gill–Head-Gordon–Pople, and Rys quadrature schemes as well as their combinations for the evaluation of three-center Coulomb integrals were compared. Various algorithmic considerations, such as the order of loops for primitive functions, the application of the horizontal recurrence relation and the solid harmonic transformation at the primitive or contracted level, and several prescreening strategies have also been investigated. Based on estimations for the number of necessary floating point operations for a simple model system, it was concluded that the Obara–Saika scheme, utilizing the vertical recurrence relation of Ahlrichs [93], is the most efficient choice, with the Gill–Head-Gordon–Pople algorithm and the combination of the Rys quadrature and the Obara–Saika schemes being competitive for a few special cases. The most promising algorithms were implemented via automated code generation for all shell triplets up to $(\boldsymbol{hh}|\boldsymbol{i})$ along with the discussed algorithmic approaches. Wall time measurements for medium sized molecules also showed the Obara–Saika scheme to be superior, and the most effective prescreening technique was determined for each algorithmic approach. Even though the floating point operation counts suggested that the horizontal recurrence relation and the solid harmonic transformation are significantly more efficient when applied to contracted integrals, this does not seem to be the case for the majority of shell triplets encountered in practical calculations applying primitive prescreening. The reason for this is that performing these two tasks on primitive integrals allows for the use of a more effective memory layout. Based on the above mentioned investigations, a recommendation has been presented for the algorithms to be used for the various shell triplets, favoring the ones that perform the best with triple- and quadruple-zeta basis sets.

Chapter 3

Optimization of three-center integral derivatives

This Chapter is largely based on Ref. 112, and it presents the optimization of the calculation of the differentiated three-center ERIs. The procedure is carried out in a similar manner as for the undifferentiated integrals: first the considered algorithms are listed, then their FLOP counts are estimated, finally the most promising schemes are implemented via automated code generation, and their practical performance is assessed. For the most part the problem of evaluating the derivatives is centered around the computation of the ERIs required for Eq. (1.1.4) or (1.1.10). However, the MD, GHP, and Rys schemes have not been extended to Hermite Gaussian integrals, even though the simplicity of Eq. (1.1.10) and the fact that Hermite Gaussians can also be transformed into the solid harmonic Gaussian basis makes this approach worthy of discussion in the context of ERI derivative calculations. Even if the differentiation rule is simpler for Hermite Gaussians, the comparison of Eqs. (1.2.26) and (1.4.29) suggests that the recursions for Cartesian Gaussians are less expensive, and it will be demonstrated that this is often the case. The advantageous properties of both types of basis functions for derivative calculations gives the motivation to consider mixed Gaussian ERIs for this purpose, which contain both Hermite and Cartesian Gaussians.

In some cases the translational invariance of the integrals will be utilized, which allows us to write

$$\Big(\frac{\partial \boldsymbol{L}_a}{\partial A_x}\boldsymbol{L}_b|\boldsymbol{L}_c\Big) + \Big(\boldsymbol{L}_a\frac{\partial \boldsymbol{L}_b}{\partial B_x}|\boldsymbol{L}_c\Big) = -\Big(\boldsymbol{L}_a\boldsymbol{L}_b|\frac{\partial \boldsymbol{L}_c}{\partial C_x}\Big)\ . \tag{3.0.1}$$

Utilizing Eq. (3.0.1) we only need to explicitly calculate 6 of the 9 possible derivatives of a three-center ERI. When Cartesian Gaussians are used, this means the classes $([\boldsymbol{L}_a{+}\boldsymbol{1}_x]\boldsymbol{L}_b|\boldsymbol{L}_c)$, $([\boldsymbol{L}_a{-}\boldsymbol{1}_x]\boldsymbol{L}_b|\boldsymbol{L}_c)$, $(\boldsymbol{L}_a[\boldsymbol{L}_b{+}\boldsymbol{1}_x]|\boldsymbol{L}_c)$, and $(\boldsymbol{L}_a[\boldsymbol{L}_b{-}\boldsymbol{1}_x]|\boldsymbol{L}_c)$, for example, have to be evaluated to construct the necessary derivatives. Kahn has shown [50] that, from the rotational invariance of the integrals, it is possible to recover all of the derivatives of an $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ integral by just computing, for example, $\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)/\partial A_x$, $\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)/\partial A_y$, $\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)/\partial B_z$, and three other auxiliary integrals which are linear combinations of the ERIs in the $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ class. The remaining derivatives can be calculated by solving three simultaneous linear equations, optionally at the contracted level. The possible advantage is that not all components of the classes with increased angular momenta have to be evaluated. The approach, however, has drawbacks. It is required to compute the extra class $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$. When the HRR is applied, as it is often the case, these ERIs appear as intermediates for $(\boldsymbol{L}_a[\boldsymbol{L}_b+\mathbf{1}_x]|\boldsymbol{L}_c)$. However, this fact can not be exploited if the buildup of the final angular momenta takes place at the contracted level since according to Eq. (1.1.4) the class with incremented $\boldsymbol{L}_b$ will be scaled by $2b$, meaning that the class required for the auxiliary integrals has to be computed by a separate recursion. Furthermore, none of the functions of the $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ ERIs can be transformed into the solid harmonic Gaussian basis until the auxiliary integrals have been formed, while performing these operations on earlier intermediates is known [60] to enhance the speed of the calculation. This also hinders the exploitation of $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ as an intermediate of another class. The formation of the auxiliary integrals is also relatively expensive, each requiring five additions and six multiplications. The solution of the system of linear equations is also considerably more expensive than the application of Eq. (3.0.1). Finally, the advantage of not evaluating every component of the more expensive classes becomes less significant with the increase of the angular momenta. For example, when we wish to compute every $\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)/\partial B_z$, we need the same number of components from the $(\boldsymbol{L}_a[\boldsymbol{L}_b+\mathbf{1}_x]|\boldsymbol{L}_c)$ class as the number of ERIs in $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$, with the ratio of the number of the components in the two classes getting smaller with increasing $\boldsymbol{L}_b$. Because of these considerations I only deal with the invariance property defined by Eq. (1.1.24) in this work.

The notation for the various investigated methods will display the basic algorithm, the applied basis function, the center the coordinates of which we differentiate with respect to, and an additional index if different approaches for a given problem were inspected. For example, $\mathrm{GHP}_{\mathrm{Cart,A2}}$ will denote the GHP-based scheme where Cartesian Gaussians are used and differentiate with respect to **A**, while the approach to which "2" refers will be explained later. I will omit the indices when I talk about all the possible values for those indices, e.g., $\mathrm{OS}_{\mathrm{Cart}}$ means all the discussed methods within the OS scheme that apply Cartesian Gaussians. I will use an upper left index "3" to denote schemes where all the derivatives of the shell triplets are produced by a common recursion. For example, ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ refers to the OS-based scheme using Cartesian Gaussians where the derivatives with respect to **A** and **B** are simultaneously evaluated by recursion, and the **C** derivatives are calculated via Eq. (3.0.1).

3.1 Three-center ERI derivative evaluation algorithms

3.1.1 Obara–Saika method

It was concluded in Chapter 2 that the scheme denoted as OS1 in Subsect. 2.1.1 is the most efficient OS-type algorithm to compute three-center Cartesian ERIs. Here it will be more useful to use the starting ERIs

$$(\mathbf{00}|\mathbf{0})^{(n)} = (-2\alpha)^n \theta_{pc}\kappa_{ab} F_n(\alpha R_{\mathrm{PC}}^2) \ , \tag{3.1.1}$$

which modifies the equations presented in Subsect. 2.1.1 so that the ERIs with superscript $(n+1)$ get divided by -2α. The recursion to build up $\boldsymbol{l}_a$ becomes

$$\begin{aligned}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} &= X_{\mathrm{PA}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)} + \frac{1}{2p}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n+1)} \\ &+ \frac{i_a}{2p}\Big(([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} + \frac{1}{2p}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n+1)}\Big) \ ,\end{aligned} \tag{3.1.2}$$

and the equation for incrementing $\boldsymbol{l}_c$ is now

$$(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} = -\frac{1}{2c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} - \frac{i_a}{4pc}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} \ . \tag{3.1.3}$$

In the OS1 approach one applies Eqs. (3.1.2), (3.1.3), and (2.1.3) to build up $\boldsymbol{l}_a$, $\boldsymbol{l}_c$, and $\boldsymbol{l}_b$, respectively. The application of this method is straightforward for the derivatives with respect to the coordinates of $\mathbf{A}$ and $\mathbf{B}$, where we use Eqs. (2.1.1) to (2.1.3) to evaluate the required classes for Eq. (1.1.4). This slightly modifies the index ranges, e.g., for $([\partial\boldsymbol{L}_a/\partial A_x]\boldsymbol{L}_b|\boldsymbol{L}_c)$ the lower and upper limits that depend on L_a decrease and increase by one, respectively. Eq. (3.1.3) can, however, not be applied if we wish to evaluate $(\boldsymbol{L}_a\boldsymbol{L}_b|[\partial\boldsymbol{L}_c/\partial C_x])$. Instead, the original VRR equation [33] reduced to the three-center case will be used:

$$\begin{aligned}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} &= -\frac{1}{2c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} - \frac{i_a}{4pc}([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} \\ &+ \frac{i_c}{2c}\Big((\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n)} + \frac{1}{2c}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n+1)}\Big) \ .\end{aligned} \tag{3.1.4}$$

For undifferentiated ERIs the last two terms of Eq. (3.1.4) cancel when we transform $\boldsymbol{L}_c$ to the solid harmonic Gaussian basis [93]. Now we are changing to the differentiated solid harmonic Gaussian basis, and from Eqs. (1.1.4) and (1.1.22) it can be seen that functions with angular momenta higher and lower than L_c get transformed by the coefficients belonging to L_c, hence the cancellation does not take place. The $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)}$ ERIs required for the calculation of an $(\boldsymbol{L}_a\mathbf{0}|\boldsymbol{L}_c)$ class by Eq. (3.1.4) are those with the $\mathrm{mod}[L_c, 2] \leq n \leq L_c$ and $L_a - \mathrm{mod}[L_c, 2] - 2\lceil(n - \mathrm{mod}[L_c, 2])/2\rfloor \leq l_a \leq L_a$ index ranges.

From Eqs. (1.4.28) and (3.1.1) the bra side VRR to build up $\tilde{\boldsymbol{l}}_a$ in Hermite ERIs is

$$\begin{aligned}([\tilde{\boldsymbol{l}}_a + \tilde{\mathbf{1}}_x]\mathbf{0}|\mathbf{0})^{(n)} &= X_{\mathrm{PA}}(\tilde{\boldsymbol{l}}_a\mathbf{0}|\mathbf{0})^{(n)} + \frac{1}{2p}X_{\mathrm{PC}}(\tilde{\boldsymbol{l}}_a\mathbf{0}|\mathbf{0})^{(n+1)} \\ &+ \frac{\tilde{i}_a}{2p}\Big(-\frac{b}{a}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\mathbf{0}|\mathbf{0})^{(n)} + \frac{1}{2p}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\mathbf{0}|\mathbf{0})^{(n+1)}\Big) \ . \qquad (3.1.5)\end{aligned}$$

The VRR for the increment of $\tilde{\boldsymbol{l}}_c$ is the same as Eq. (3.1.3). Based on Eq. (1.4.29) we can write the HRR for Hermite three-center ERIs as

$$\begin{aligned}(\tilde{\boldsymbol{l}}_a[\tilde{\boldsymbol{l}}_b + \tilde{\mathbf{1}}_x]|\tilde{\boldsymbol{L}}_c) &= ([\tilde{\boldsymbol{l}}_a + \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{L}}_c) + X_{\mathrm{AB}}(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{L}}_c) \\ &+ \frac{\tilde{i}_a}{2a}([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\tilde{\boldsymbol{l}}_b|\tilde{\boldsymbol{L}}_c) - \frac{\tilde{i}_b}{2b}(\tilde{\boldsymbol{l}}_a[\tilde{\boldsymbol{l}}_b - \tilde{\mathbf{1}}_x]|\tilde{\boldsymbol{L}}_c) \ . \qquad (3.1.6)\end{aligned}$$

The range for the necessary $(\tilde{\boldsymbol{l}}_a\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ intermediates for Eq. (3.1.6) is $\tilde{L}_a - \tilde{L}_b \le \tilde{l}_a \le \tilde{L}_a + \tilde{L}_b$. The ERIs required for **A** and **B** derivatives are computed with the above described scheme with modified index limits, e.g., for $([\partial\tilde{\boldsymbol{L}}_a/\partial A_x]\tilde{\boldsymbol{L}}_b|\tilde{\boldsymbol{L}}_c)$ the lower and upper limits that depend on $\tilde{L}_a$ increase by one according to Eq. (1.1.10). Note that the multiplication with the double of the exponent appearing in Eq. (1.1.10) can be built into Eq. (2.1.1) if we calculate a derivative with respect to one center at a time. When we differentiate with respect to **C**, the necessary ket VRR equation is analogous with Eq. (3.1.4), however, the third term is zero. This is because in the corresponding equation for four-center ERIs [46] this third term is multiplied by $-d/c$, where d is the exponent for the fourth function, and d is zero for three-center integrals. Hence the relation becomes

$$\begin{aligned}(\tilde{\boldsymbol{l}}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c + \tilde{\mathbf{1}}_x])^{(n)} &= -\frac{1}{2c}X_{\mathrm{PC}}(\tilde{\boldsymbol{l}}_a\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)} - \frac{1}{4pc}\tilde{i}_a([\tilde{\boldsymbol{l}}_a - \tilde{\mathbf{1}}_x]\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)} \\ &+ \frac{1}{4c^2}\tilde{i}_c(\tilde{\boldsymbol{l}}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c - \tilde{\mathbf{1}}_x])^{(n+1)} \ , \qquad (3.1.7)\end{aligned}$$

for which we need $(\tilde{\boldsymbol{l}}_a\mathbf{0}|\mathbf{0})^{(n)}$ type ERIs to build an $(\tilde{\boldsymbol{L}}_a\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ class for $\tilde{L}_c - \lceil \tilde{L}_c/2 \rfloor \le n \le \tilde{L}_c$ and $\tilde{L}_a + \tilde{L}_c - 2n \le \tilde{l}_a \le \tilde{L}_a$.

We see that both kinds of Gaussian basis functions have advantageous properties. With Hermite Gaussians, the explicit evaluation of the primitive derivatives can be avoided and the recursion for $\tilde{\boldsymbol{L}}_c$ during the computation of $(\tilde{\boldsymbol{L}}_a\tilde{\boldsymbol{L}}_b|[\partial\tilde{\boldsymbol{L}}_c/\partial C_x])$ is simpler, while for the Cartesian Gaussians the HRR is cheaper, independent of the exponents, and requires a smaller range of intermediates. For higher L_b values this latter property compensates for the widening of the index ranges due to Eq. (1.1.4), while if the differentiated function is of $\boldsymbol{s}$ type the direct application of Eq. (1.1.4) can be avoided similarly to the Hermite case. In the following it is exploited that, for first derivatives, only one of the functions has to be Hermite Gaussian to utilize Eq. (1.1.10), and it will be investigated how calculating such mixed Gaussian ERIs affects the complexity of the task.

For the **A** derivatives, we choose the first and second bra functions to be a Hermite and a Cartesian Gaussian, respectively, and use Eq. (3.1.5) for the bra VRR. Since the choice for the function in the ket side is irrelevant now, we apply Eq. (3.1.3) to gain the $(\tilde{\boldsymbol{l}}_a\mathbf{0}|\tilde{\boldsymbol{l}}_c)$ type intermediates. For the $\tilde{\boldsymbol{l}}_a \to \boldsymbol{l}_b$ translation we need a new HRR. We substitute Eq. (1.1.3) into the bra side overlap distribution $\tilde{H}_{\tilde{i}_a}G_{i_b+1}$, then apply Eqs. (1.1.14) and (1.1.9) to get

$$\begin{aligned}\tilde{H}_{\tilde{i}_a}G_{i_b+1} &= x_{\mathrm{B}}\tilde{H}_{\tilde{i}_a}G_{i_b} = x_{\mathrm{A}}\tilde{H}_{\tilde{i}_a}G_{i_b} + X_{\mathrm{AB}}\tilde{H}_{\tilde{i}_a}G_{i_b} \\ &= \tilde{H}_{\tilde{i}_a+\tilde{1}}G_{i_b} + \frac{\tilde{i}_a}{2a}\tilde{H}_{\tilde{i}_a-\tilde{1}}G_{i_b} + X_{\mathrm{AB}}\tilde{H}_{\tilde{i}_a}G_{i_b} \end{aligned} \tag{3.1.8}$$

from which the new HRR,

$$(\tilde{\boldsymbol{l}}_a[\boldsymbol{l}_b+\mathbf{1}_x]|\tilde{\boldsymbol{L}}_c) = ([\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x]\boldsymbol{l}_b|\tilde{\boldsymbol{L}}_c) + X_{\mathrm{AB}}(\tilde{\boldsymbol{l}}_a\boldsymbol{l}_b|\tilde{\boldsymbol{L}}_c) + \frac{\tilde{i}_a}{2a}([\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x]\boldsymbol{l}_b|\tilde{\boldsymbol{L}}_c)\ , \tag{3.1.9}$$

can be written. Eq. (3.1.9) is one term less expensive than Eq. (3.1.6). In the case of the derivatives for $\mathbf{B}$ we reverse the type of the three functions and use Eqs. (3.1.2) and (3.1.3) for the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)$ ERIs. The recurrence to move angular momentum from $\boldsymbol{l}_a$ to $\tilde{\boldsymbol{l}}_b$ is derived similarly to Eq. (3.1.9). By inserting Eq. (1.1.9) into $G_{i_a}\tilde{H}_{\tilde{i}_b+\tilde{1}}$ then applying Eq. (1.1.14) and (1.1.3) we obtain

$$\begin{aligned}G_{i_a}\tilde{H}_{\tilde{i}_b+\tilde{1}} &= x_{\mathrm{B}}G_{i_a}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_b}{2b}G_{i_a}\tilde{H}_{\tilde{i}_b-\tilde{1}} \\ &= G_{i_a+1}\tilde{H}_{\tilde{i}_b} + X_{\mathrm{AB}}G_{i_a}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_b}{2b}G_{i_a}\tilde{H}_{\tilde{i}_b-\tilde{1}}\ . \end{aligned} \tag{3.1.10}$$

Hence, the suitable recurrence is

$$(\boldsymbol{l}_a[\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x]|\boldsymbol{L}_c) = ([\boldsymbol{l}_a+\mathbf{1}_x]\tilde{\boldsymbol{l}}_b|\boldsymbol{L}_c) + X_{\mathrm{AB}}(\boldsymbol{l}_a\tilde{\boldsymbol{l}}_b|\boldsymbol{L}_c) - \frac{\tilde{i}_b}{2b}(\boldsymbol{l}_a[\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x]|\boldsymbol{L}_c)\ , \tag{3.1.11}$$

for which the range of the necessary starting intermediates is the same as for Eq. (2.1.3). The obvious choice for the $\mathbf{C}$ derivatives is to use the $(\boldsymbol{l}_a\boldsymbol{l}_b|\tilde{\boldsymbol{l}}_c)$ ERIs since this would allow us to employ Eq. (2.1.3) for the HRR. To arrive at the required equation to construct $(\boldsymbol{l}_a\mathbf{0}|\tilde{\boldsymbol{l}}_c)$ type intermediates let us express the incremented integral according to the expansion in the McMurchie–Davidson scheme as

$$(\boldsymbol{l}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c+\tilde{\mathbf{1}}_x])^{(n)} \;=\; \left(\frac{1}{2c}\right)^{\tilde{l}_c+\tilde{1}} \sum_{\bar{i}_p,\bar{j}_p\bar{k}_p} E^{i_a,0}_{\bar{i}_p}E^{j_a,0}_{\bar{j}_p}E^{k_a,0}_{\bar{k}_p}(-1)^{\tilde{l}_c+\tilde{1}}(\bar{\boldsymbol{l}}_c+\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x)^{(n)}\ , \tag{3.1.12}$$

where we expanded the Cartesian overlap distribution in the bra in terms of unscaled Hermite Gaussians defined by Eq. (1.1.11), and the scaling with the exponent present in Eq. (1.1.8) has been accounted for the ket side function. The sums over i_p, j_p, and k_p

run from 0 to i_a, j_a, and k_a, respectively. Inserting Eq. (1.2.5) into Eq. (3.1.12) we get

$$\begin{aligned}
(\boldsymbol{l}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c+\tilde{\mathbf{1}}_x])^{(n)} &= X_{\mathrm{PC}}\Big(\frac{1}{2c}\Big)^{\tilde{l}_c+\tilde{1}}\sum_{\bar{i}_p,\bar{j}_p,\bar{k}_p} E_{\bar{i}_p}^{i_a,0}E_{\bar{j}_p}^{j_a,0}E_{\bar{k}_p}^{k_a,0}(-1)^{\tilde{l}_c+\tilde{1}}(\bar{\boldsymbol{l}}_c+\bar{\boldsymbol{l}}_p)^{(n+1)} \\
&+ \Big(\frac{1}{2c}\Big)^{\tilde{l}_c+\tilde{1}}\sum_{\bar{i}_p,\bar{j}_p,\bar{k}_p} E_{\bar{i}_p}^{i_a,0}E_{\bar{j}_p}^{j_a,0}E_{\bar{k}_p}^{k_a,0}(-1)^{\tilde{l}_c+\tilde{1}}(\bar{i}_p+\bar{i}_c)(\bar{\boldsymbol{l}}_c+\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x)^{(n+1)} \\
&= -\frac{1}{2c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)}+\frac{\tilde{i}_c}{4c^2}(\boldsymbol{l}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c-\tilde{\mathbf{1}}_x])^{(n+1)} \\
&+ \Big(\frac{1}{2c}\Big)^{\tilde{l}_c+\tilde{1}}\sum_{\bar{i}_p,\bar{j}_p,\bar{k}_p} \bar{i}_p E_{\bar{i}_p}^{i_a,0}E_{\bar{j}_p}^{j_a,0}E_{\bar{k}_p}^{k_a,0}(-1)^{\tilde{l}_c+\tilde{1}}(\bar{\boldsymbol{l}}_c+\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x)^{(n+1)}\ , \quad (3.1.13)
\end{aligned}$$

where the first two terms in the last equation arise from substituting back Eq. (3.1.12). By factoring out $1/(2p)$ in the last term we can use Eq. (1.2.18) and, noting the conditions given in Eq. (1.2.10), we can substitute Eq. (3.1.12) for the last term. It becomes $-i_a/(4pc)([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)}$, and we arrive at

$$\begin{aligned}
(\boldsymbol{l}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c+\tilde{\mathbf{1}}_x])^{(n)} &= -\frac{1}{2c}X_{\mathrm{PC}}(\boldsymbol{l}_a\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)}-\frac{i_a}{4pc}([\boldsymbol{l}_a-\mathbf{1}_x]\mathbf{0}|\tilde{\boldsymbol{l}}_c)^{(n+1)} \\
&+ \frac{\tilde{i}_c}{4c^2}(\boldsymbol{l}_a\mathbf{0}|[\tilde{\boldsymbol{l}}_c-\tilde{\mathbf{1}}_x])^{(n+1)}\ . \quad (3.1.14)
\end{aligned}$$

Eq. (3.1.14) is analogous to Eq. (3.1.7), but since the function on the first center is now a Cartesian Gaussian, we can use Eq. (2.1.3) to build up $\boldsymbol{L}_b$. In addition to the OS methods for Cartesian ($\mathrm{OS}_{\mathrm{Cart}}$) or Hermite ($\mathrm{OS}_{\mathrm{Herm}}$) Gaussian ERI derivatives, the new schemes based on these mixed Gaussian ERIs ($\mathrm{OS}_{\mathrm{Mixed}}$) will also be analyzed, and it will be concluded that the use of such mixed integrals is often superior to the pure Cartesian and always to the pure Hermite algorithms. For the case when the derivatives of all three centers are evaluated at the same time, the purely Cartesian Gaussian based methods (${}^3\mathrm{OS}_{\mathrm{Cart}}$) will be compared with ${}^3\mathrm{OS}_{\mathrm{Herm,AB}}$, ${}^3\mathrm{OS}_{\mathrm{Mixed,AC}}$, and ${}^3\mathrm{OS}_{\mathrm{Mixed,BC}}$. As it will be apparent, the ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ route is very competitive because of the simple form of Eq. (2.1.3). The OS-based algorithms are summarized in Table 3.1.

3.1.2 McMurchie–Davidson method

According to the results of Chapter 2 it is more efficient to transform unscaled Hermite Gaussians into the desired overlap distribution in the bra side of a three-center ERI via recurrences rather than utilizing Eq. (2.1.11). Hence, only this approach will be followed to evaluate the necessary ERI classes required for the derivative calculation. When using Cartesian Gaussian functions in the bra, the needed classes can be built up by Eqs. (1.2.24) and (1.2.25), as in the MD4 scheme discussed in Subsect. 2.1.2.

We can also work with mixed or Hermite Gaussian integrals in the MD scheme [112]. In the case of Hermite ERIs I use an auxiliary function different from Eq. (1.2.23),

TABLE 3.1: Serial number of the equations to be applied to calculate the necessary classes in the various OS-based algorithms. Eq. (2.1.1) is omitted since its use is common to all routes. Separate recursion denotes the schemes where only derivatives with respect to one center are evaluated, while in the common recursion schemes derivatives for all the centers are calculated at the same time for a shell triplet.

Separate recursion	
$\mathrm{OS}_{\mathrm{Cart,A}}$ and $\mathrm{OS}_{\mathrm{Cart,B}}$	(3.1.2), (3.1.3), (2.1.3)
$\mathrm{OS}_{\mathrm{Cart,C}}$	(3.1.2), (3.1.4), (2.1.3)
$\mathrm{OS}_{\mathrm{Mixed,A}}$	(3.1.5), (3.1.3), (3.1.9)
$\mathrm{OS}_{\mathrm{Mixed,B}}$	(3.1.2), (3.1.3), (3.1.11)
$\mathrm{OS}_{\mathrm{Mixed,C}}$	(3.1.2), (3.1.14), (2.1.3)
$\mathrm{OS}_{\mathrm{Herm,A}}$ and $\mathrm{OS}_{\mathrm{Herm,B}}$	(3.1.5), (3.1.3), (3.1.6)
$\mathrm{OS}_{\mathrm{Herm,C}}$	(3.1.5), (3.1.7), (3.1.6)
Common recursion	
${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$	(3.1.2), (3.1.3), (2.1.3)
${}^3\mathrm{OS}_{\mathrm{Cart,AC}}$ and ${}^3\mathrm{OS}_{\mathrm{Cart,BC}}$	(3.1.2), (3.1.4), (2.1.3)
${}^3\mathrm{OS}_{\mathrm{Cart,AC}}$ and ${}^3\mathrm{OS}_{\mathrm{Cart,BC}}$	(3.1.2), (3.1.4), (2.1.3)
${}^3\mathrm{OS}_{\mathrm{Mixed,AC}}$	(3.1.5), (3.1.7), (3.1.9)
${}^3\mathrm{OS}_{\mathrm{Mixed,BC}}$	(3.1.2), (3.1.14), (3.1.11)

which is written as

$$\Omega^{\bar{i}_p}_{\tilde{i}_a,\tilde{i}_b} = \Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}\Lambda^x_{ab} \ . \tag{3.1.15}$$

$\Omega^{\bar{i}_p}_{\tilde{i}_a,\tilde{i}_b}$ is the x-dependent part of the function $\Omega^{\bar{l}_p}_{\tilde{l}_a,\tilde{l}_b} = \Omega^{\bar{i}_p}_{\tilde{i}_a,\tilde{i}_b}\Omega^{\bar{j}_p}_{\tilde{j}_a,\tilde{j}_b}\Omega^{\bar{k}_p}_{\tilde{k}_a,\tilde{k}_b}$. Our goal is to connect the $\Omega^{\bar{i}_p}_{0,0}$ type intermediates to the $\Omega^{0}_{\tilde{i}_a,\tilde{i}_b}$ Hermite Gaussian overlaps by recursion. After incrementing $\tilde{i}_a$ we can write

$$\begin{aligned}\Omega^{\bar{i}_p}_{\tilde{i}_a+\tilde{1},\tilde{i}_b} &= \Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a+\tilde{1}}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}\Lambda^x_{ab} \\ &= \frac{1}{2a}\Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}2ax_{\mathrm{A}}\Lambda^x_{ab} \\ &= \Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}(x_{\mathrm{P}}+X_{\mathrm{PA}})\Lambda^x_{ab} \ ,\end{aligned} \tag{3.1.16}$$

where in the second line Eq. (1.1.5) was differentiated by A_x once, and in the third line x_{A} was rewritten according to Eq. (1.2.2). Next, it is exploited that from Eq. (1.1.5)

$$x_{\mathrm{P}}\Lambda^x_{ab} = \frac{1}{2p}\frac{\partial}{\partial P_x}\Lambda^x_{ab} \ , \tag{3.1.17}$$

and also

$$\Big[\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}, X_{\mathrm{PA}}\Big] = 0 \ , \tag{3.1.18}$$

which follows from $X_{\mathrm{PA}} = P_x - A_x$ and Eq. (1.2.15). Inserting Eqs. (3.1.17) and (3.1.18) into Eq. (3.1.16) we write

$$\Omega^{\tilde{i}_p}_{\tilde{i}_a+\tilde{1},\tilde{i}_b} = \Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big[\frac{1}{2p}\Big(\frac{\partial}{\partial P_x}\Big)^{\tilde{i}_p+\tilde{1}} + X_{\mathrm{PA}}\Big(\frac{\partial}{\partial P_x}\Big)^{\tilde{i}_p}\Big]\Lambda^x_{ab} \; . \tag{3.1.19}$$

For the expression stemming from the first term in the bracket we can substitute Eq. (3.1.15). We also apply

$$\Big[\Big(\frac{\partial}{\partial B_x}\Big)^{\tilde{i}_b}, X_{\mathrm{PA}}\Big] = \tilde{i}_b\frac{b}{p}\Big(\frac{\partial}{\partial B_x}\Big)^{\tilde{i}_b-\tilde{1}} \; , \tag{3.1.20}$$

which can be proven by induction, and write

$$\begin{aligned}\Omega^{\tilde{i}_p}_{\tilde{i}_a+\tilde{1},\tilde{i}_b} &= \frac{1}{2p}\Omega^{\tilde{i}_p+\tilde{1}}_{\tilde{i}_a,\tilde{i}_b} \\ &+ \Big(\frac{1}{2b}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big[\tilde{i}_b\frac{b}{p}\Big(\frac{\partial}{\partial B_x}\Big)^{\tilde{i}_b-\tilde{1}} + X_{\mathrm{PA}}\Big(\frac{\partial}{\partial B_x}\Big)^{\tilde{i}_b}\Big]\Big(\frac{\partial}{\partial P_x}\Big)^{\tilde{i}_p}\Lambda^x_{ab} \; . \end{aligned}\tag{3.1.21}$$

Next, in the same manner as it was done for Eq. (3.1.19), Eq. (3.1.15) is substituted for the part arising from the first term in the bracket, and the commutator

$$\Big[\Big(\frac{\partial}{\partial A_x}\Big)^{\tilde{i}_a}, X_{\mathrm{PA}}\Big] = -\tilde{i}_a\frac{b}{p}\Big(\frac{\partial}{\partial A_x}\Big)^{\tilde{i}_a-\tilde{1}} \tag{3.1.22}$$

is inserted for the other part to get

$$\begin{aligned}\Omega^{\tilde{i}_p}_{\tilde{i}_a+\tilde{1},\tilde{i}_b} &= \frac{1}{2p}\Omega^{\tilde{i}_p+\tilde{1}}_{\tilde{i}_a,\tilde{i}_b} + \frac{\tilde{i}_b}{2p}\Omega^{\tilde{i}_p}_{\tilde{i}_a,\tilde{i}_b-\tilde{1}} \\ &+ \Big(\frac{1}{2a}\Big)^{\tilde{i}_a}\Big[-\tilde{i}_a\frac{b}{p}\Big(\frac{\partial}{\partial A_x}\Big)^{\tilde{i}_a-\tilde{1}} + X_{\mathrm{PA}}\Big(\frac{\partial}{\partial A_x}\Big)^{\tilde{i}_a}\Big]\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\tilde{i}_p}\Lambda^x_{ab} \; . \end{aligned}\tag{3.1.23}$$

We can now utilize Eq. (3.1.15) to arrive at the recurrence

$$(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a+\mathbf{1}_x,\tilde{\mathbf{i}}_b}| = \frac{1}{2p}(\Omega^{\tilde{\mathbf{i}}_p+\mathbf{1}_x}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}| + X_{\mathrm{PA}}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}| - \frac{\tilde{i}_a}{2p}\frac{b}{a}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a-\mathbf{1}_x,\tilde{\mathbf{i}}_b}| + \frac{\tilde{i}_b}{2p}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b-\mathbf{1}_x}| \; , \tag{3.1.24}$$

where the more general $\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}$ functions are used, and the irrelevant ket side of the ERIs is omitted. By a similar procedure one can derive the recursion to increment $\tilde{i}_b$ as

$$(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b+\mathbf{1}_x}| = \frac{1}{2p}(\Omega^{\tilde{\mathbf{i}}_p+\mathbf{1}_x}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}| + X_{\mathrm{PB}}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}| + \frac{\tilde{i}_a}{2p}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a-\mathbf{1}_x,\tilde{\mathbf{i}}_b}| - \frac{\tilde{i}_b}{2p}\frac{a}{b}(\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b-\mathbf{1}_x}| \; , \tag{3.1.25}$$

where the commutators

$$\Big[\Big(\frac{\partial}{\partial A_x}\Big)^{i_a}, X_{\mathrm{PB}}\Big] = i_a\frac{a}{p}\Big(\frac{\partial}{\partial A_x}\Big)^{i_a-1} \tag{3.1.26}$$

$$\Big[\Big(\frac{\partial}{\partial B_x}\Big)^{i_b}, X_{\mathrm{PB}}\Big] = -i_b\frac{a}{p}\Big(\frac{\partial}{\partial B_x}\Big)^{i_b-1} \tag{3.1.27}$$

$$\Big[\Big(\frac{\partial}{\partial P_x}\Big)^{\tilde{i}_p}, X_{\mathrm{PB}}\Big] = 0 \tag{3.1.28}$$

were applied. To calculate ERIs over $\Omega^{\mathbf{0}}_{\tilde{\mathbf{i}}_a,\tilde{\mathbf{i}}_b}$ with Eq. (3.1.25) we need the $\Omega^{\tilde{\mathbf{i}}_p}_{\tilde{\mathbf{i}}_a,\mathbf{0}}$ integrals for $\tilde{L}_a - \tilde{L}_b \le \tilde{l}_a \le \tilde{L}_a$ and $0 \le \bar{l}_p \le \tilde{L}_b - (\tilde{L}_a - \tilde{l}_a)$, while the $\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\mathbf{0}}$ ERIs are gained by applying Eq. (3.1.24) starting from $\Omega^{\bar{\boldsymbol{l}}_o}_{\mathbf{0},\mathbf{0}}$ ERIs for $\bar{l}_p \le \bar{l}_o \le \bar{l}_p + \tilde{l}_a$.

To reach mixed Gaussian ERIs with an $(\tilde{\boldsymbol{l}}_a \boldsymbol{l}_b|$ bra side we utilize recurrences between the auxiliary functions

$$\Omega^{\bar{i}_p}_{\tilde{i}_a,i_b} = x_{\mathrm{B}}^{i_b}\Big(\frac{\partial}{2a\partial A_x}\Big)^{\tilde{i}_a}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}\Lambda^x_{ab} \ . \tag{3.1.29}$$

The relation to increment $\tilde{i}_a$ is obtained similarly to Eq. (3.1.24): Λ^x_{ab} is differentiated with respect to A_x, then Eqs. (1.2.2), (3.1.17), (3.1.18), and (3.1.22) are applied to get

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}| = \frac{1}{2p}(\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}| + X_{\mathrm{PA}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}| - \frac{\tilde{i}_a}{2p}\frac{b}{a}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}| \ . \tag{3.1.30}$$

If we increment i_b in Eq. (3.1.29), we can write

$$\begin{aligned}
\Omega^{\bar{i}_p}_{\tilde{i}_a,i_b+1} &= x_{\mathrm{B}}^{i_b+1}\Big(\frac{\partial}{2a\partial A_x}\Big)^{i_a}\Big(\frac{\partial}{\partial P_x}\Big)^{i_p}\Lambda^x_{ab} \\
&= x_{\mathrm{B}}^{i_b}\Big(\frac{\partial}{2a\partial A_x}\Big)^{i_a} x_{\mathrm{B}}\Big(\frac{\partial}{\partial P_x}\Big)^{i_p}\Lambda^x_{ab} \\
&= x_{\mathrm{B}}^{i_b}\Big(\frac{\partial}{2a\partial A_x}\Big)^{i_a}(x_{\mathrm{P}} + X_{\mathrm{PB}})\Big(\frac{\partial}{\partial P_x}\Big)^{i_p}\Lambda^x_{ab} \\
&= x_{\mathrm{B}}^{i_b}\Big(\frac{\partial}{2a\partial A_x}\Big)^{i_a}\Big[\frac{1}{2p}\Big(\frac{\partial}{\partial P_x}\Big)^{i_p+1} + i_p\Big(\frac{\partial}{\partial P_x}\Big)^{i_p-1} + X_{\mathrm{PB}}\Big(\frac{\partial}{\partial P_x}\Big)^{i_p}\Big]\Lambda^x_{ab} \ ,
\end{aligned} \tag{3.1.31}$$

where in the last line the commutator

$$\Big[x_{\mathrm{P}}, \Big(\frac{\partial}{\partial P_x}\Big)^{i_p}\Big] = i_p\Big(\frac{\partial}{\partial P_x}\Big)^{i_p-1} \tag{3.1.32}$$

and also Eq. (3.1.17) were utilized. Substituting Eq. (3.1.26) into Eq. (3.1.31) we arrive at

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b+\mathbf{1}_x}| = \frac{1}{2p}(\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}| + X_{\mathrm{PB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}| + \bar{i}_p(\Omega^{\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}| + \frac{\tilde{i}_a}{2p}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}| \ . \tag{3.1.33}$$

For integrals with $(\boldsymbol{l}_a \tilde{\boldsymbol{l}}_b|$ bras the auxiliary functions

$$\Omega^{\bar{i}_p}_{i_a,\tilde{i}_b} = x_{\mathrm{A}}^{i_a}\Big(\frac{\partial}{2b\partial B_x}\Big)^{\tilde{i}_b}\Big(\frac{\partial}{\partial P_x}\Big)^{\bar{i}_p}\Lambda^x_{ab} \tag{3.1.34}$$

are used. Following through derivations analogous to the ones presented above we get

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a+\mathbf{1}_x,\tilde{\boldsymbol{l}}_b}| = \frac{1}{2p}(\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}| + X_{\mathrm{PA}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}| + \bar{i}_p(\Omega^{\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}| + \frac{\tilde{i}_b}{2p}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}| \tag{3.1.35}$$

and

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x}| = \frac{1}{2p}(\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}| + X_{\mathrm{PB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}| - \frac{\tilde{i}_b}{2p}\frac{a}{b}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}| \ . \tag{3.1.36}$$

Using Eqs. (3.1.30) and (3.1.33) the index range requirements are the same as for Eqs. (3.1.24) and (3.1.25), respectively. Reaching $\Omega^{\mathbf{0}}_{\boldsymbol{L}_a,\tilde{\boldsymbol{L}}_b}$ ERIs with Eq. (3.1.36) demands integrals over $\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{L}_a,\mathbf{0}}$ for $0 \le \bar{l}_p \le \tilde{L}_b$, while these are evaluated with Eq. (3.1.35) from $\Omega^{\bar{\boldsymbol{l}}_o}_{\mathbf{0},\mathbf{0}}$ for $\bar{l}_p - L_a \le \bar{l}_o \le \bar{l}_p + L_a$. Note that when all the target classes have the same $\tilde{\boldsymbol{L}}_c$, the necessary scaling with $(-2c)^{-\tilde{L}_c}$ present in Eq. (2.1.9) can be done at the level of the Boys functions. These newly presented equations will be mostly useful in setting up the GHP scheme for Hermite and mixed Gaussian integral derivatives.

For undifferentiated Cartesian ERIs it was concluded in Chapter 2 that the application of both Eq. (2.1.3) for $\boldsymbol{l}_b$ and Eq. (1.7.8) for $\boldsymbol{l}_c$ enhances the performance compared to using only Eqs. (1.2.5), (1.2.24), and (1.2.25). The following approaches will be considered within the MD scheme: when calculating $\mathbf{A}$ or $\mathbf{B}$ derivatives we will produce the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{\tilde{L}_c}$ or $(\tilde{\boldsymbol{l}}_a\mathbf{0}|\mathbf{0})^{\tilde{L}_c}$ intermediates via the MD recursions, then proceed as the corresponding OS scheme. If $\mathbf{C}$ derivatives are needed, this is not necessarily the best approach since it might be more efficient to increment the ket side function with Eq. (1.2.5) rather than with the VRR, which is more expensive now. Here we will produce the $(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$ target class from $(\bar{\boldsymbol{l}}_p|\tilde{\boldsymbol{L}}_c)$ by either directly with Eqs. (1.2.24) and (1.2.25) ($\mathrm{MD}_{\mathrm{Mixed,C1}}$) or using Eq. (1.2.24) for $(\boldsymbol{l}_a\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ ERIs then Eq. (2.1.3) ($\mathrm{MD}_{\mathrm{Mixed,C2}}$), or we apply Eq. (1.2.24) to the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)}$ classes followed by Eqs. (3.1.14) and (2.1.3) ($\mathrm{MD}_{\mathrm{Mixed,C3}}$). Only the mixed Gaussian scheme is considered since transforming $\tilde{\boldsymbol{L}}_c$ into $\boldsymbol{L}_c$ does not offer any advantage. When we wish to evaluate the derivatives with respect to all three centers at the same time, only the scheme applying the OS VRR to the ket side will be investigated since in the majority of the above mentioned cases (that is, differentiating with respect to a single center) this turned out to be the most efficient choice. In these cases we can also utilize the approach of Helgaker and Taylor [55] (further denoted as ${}^{3}\mathrm{MD}_{\mathrm{PR}}$) which was discussed in the last paragraph of Sect. 1.2. Here we produce derivatives with respect to $\mathbf{P}$ and $\mathbf{R}_{\mathrm{AB}}$ using the equations

$$\frac{\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)^{(n)}}{\partial P_x} = \kappa_{ab}(-2c)^{-\tilde{L}_c}\sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b}\sum_{\bar{j}_p=0}^{J_a+J_b} E_{\bar{j}_p}^{J_a,J_b}\sum_{\bar{k}_p=0}^{K_a+K_b} E_{\bar{k}_p}^{K_a,K_b}(\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x+\bar{\boldsymbol{L}}_c)^{(n)} \tag{3.1.37}$$

and, with $\partial E_{\bar{i}_p}^{I_a,I_b} \equiv \partial(\kappa_{ab}E_{\bar{i}_p}^{I_a,I_b})/\partial X_{\mathrm{AB}}$

$$\frac{\partial(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)^{(n)}}{\partial X_{\mathrm{AB}}} = (-2c)^{-\tilde{L}_c}\sum_{\bar{i}_p=0}^{I_a+I_b} \partial E_{\bar{i}_p}^{I_a,I_b}\sum_{\bar{j}_p=0}^{J_a+J_b} E_{\bar{j}_p}^{J_a,J_b}\sum_{\bar{k}_p=0}^{K_a+K_b} E_{\bar{k}_p}^{K_a,K_b}(\bar{\boldsymbol{l}}_p+\bar{\boldsymbol{L}}_c)^{(n)} . \tag{3.1.38}$$

The advantage of this scheme is that the summation limits remain the same as for undifferentiated ERIs. Note that using Eqs. (3.1.37) and (3.1.38) one should omit κ_{ab} from Eq. (2.1.1). The evaluation of the differentiated E coefficients is explained in Ref. 55. The $\mathbf{A}$ and $\mathbf{B}$ derivatives are then recovered by Eq. (1.2.29) and

$$\frac{\partial}{\partial B_x} = \frac{\partial}{\partial P_x} - \frac{\partial}{\partial A_x} \tag{3.1.39}$$

following from Eq. (1.2.15). The derivatives with respect to $\mathbf{C}$ are trivially obtained by the translational invariance, $\partial/\partial C_x = -\partial/\partial P_x$. The considered routes are summarized

TABLE 3.2: Serial numbers of the equations to be applied to calculate the necessary classes in the various MD-based algorithms. Eqs. (2.1.1) and (1.2.5) are omitted since all the schemes start with the use of these relations. See also the caption of Table 3.1.

Separate recursion	
$\mathrm{MD}_{\mathrm{Cart,A}}$ and $\mathrm{MD}_{\mathrm{Cart,B}}$	(1.2.24), (1.7.8), (2.1.3)
$\mathrm{MD}_{\mathrm{Mixed,A}}$	(3.1.30), (1.7.8), (3.1.9)
$\mathrm{MD}_{\mathrm{Mixed,B}}$	(1.2.24), (1.7.8), (3.1.11)
$\mathrm{MD}_{\mathrm{Mixed,C1}}$	(1.2.24), (1.2.25)
$\mathrm{MD}_{\mathrm{Mixed,C2}}$	(1.2.24), (2.1.3)
$\mathrm{MD}_{\mathrm{Mixed,C3}}$	(1.2.24), (3.1.14), (2.1.3)
Common recursion	
$^{3}\mathrm{MD}_{\mathrm{Cart,AB}}$	(1.2.24), (1.7.8), (2.1.3)
$^{3}\mathrm{MD}_{\mathrm{Cart,AC}}$ and $^{3}\mathrm{MD}_{\mathrm{Cart,BC}}$	(1.2.24), (3.1.4), (2.1.3)
$^{3}\mathrm{MD}_{\mathrm{Herm,AB}}$	(3.1.30), (1.7.8), (3.1.6)
$^{3}\mathrm{MD}_{\mathrm{Mixed,AC}}$	(3.1.30), (3.1.7), (3.1.9)
$^{3}\mathrm{MD}_{\mathrm{Mixed,BC}}$	(1.2.24), (3.1.14), (3.1.11)
$^{3}\mathrm{MD}_{\mathrm{PR}}$	(3.1.37), (3.1.38)

in Table 3.2. For the ^{3}MD algorithms where **C** is differentiated Cart or Mixed refers to the bra-side functions, while $\tilde{\boldsymbol{L}}_c$ is always a Hermite Gaussian.

3.1.3 Gill–Head-Gordon–Pople method

When three-center ERI derivatives are computed with the GHP method using Cartesian Gaussian basis functions on the bra side and a Hermite Gaussian in the ket the first step is to calculate $(\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ integrals using Eq. (1.2.5). Note that the starting values defined in Eq. (1.2.6) have to be multiplied by $(-2c)^{\tilde{L}_c}$ because of Eq. (1.1.8). Then, one performs the necessary scalings according to Eq. (1.3.1) to produce all the λ, β, ζ-scaled classes of these ERIs that are going to be required by Eqs. (1.3.2) and (1.3.3), performs the primitive contraction on these scaled classes, then builds up $\boldsymbol{l}_a$ and $\boldsymbol{l}_b$. At the contracted level we can work in two directions: in $\mathrm{GHP}_{\mathrm{Cart,1}}$ we build up $\boldsymbol{l}_a$ and $\boldsymbol{l}_b$ by Eq. (1.3.2) and Eq. (1.3.3), respectively, while the $\mathrm{GHP}_{\mathrm{Cart,2}}$ scheme, identical to the algorithm presented in Subsect. 2.1.3, uses Eq. (1.3.2) to construct the appropriate ${}_{\lambda,\beta,\zeta}(\Omega^{\mathbf{0}}_{\boldsymbol{l}_a,\mathbf{0}}|\tilde{\boldsymbol{L}}_c)$ intermediates for Eq. (2.1.3). The target classes are ${}_{1,0,0}([\boldsymbol{L}_a + \mathbf{1}_x]\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$ and ${}_{0,0,0}([\boldsymbol{L}_a - \mathbf{1}_x]\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c)$ for the **A** derivatives, and ${}_{0,1,0}(\boldsymbol{L}_a[\boldsymbol{L}_b+\mathbf{1}_x]|\tilde{\boldsymbol{L}}_c)$ and ${}_{0,0,0}(\boldsymbol{L}_a[\boldsymbol{L}_b-\mathbf{1}_x]|\tilde{\boldsymbol{L}}_c)$ for **B**. For the same reason as in MD, only mixed ERI based schemes will be considered for the **C** derivatives, where Eqs. (1.3.2) and (1.3.3) are employed to build up ${}_{0,0,0}(\boldsymbol{L}_a\boldsymbol{L}_b|\tilde{\boldsymbol{L}}_c + \tilde{\mathbf{1}}_x)$. Also, in the ^{3}GHP algorithms where **C** is differentiated, the function on **C** will always be a Hermite Gaussian, and the $^{3}\mathrm{GHP}_{\mathrm{Cart}}$ and $^{3}\mathrm{GHP}_{\mathrm{Mixed}}$ notations will refer to the functions in the bra side.

If we wish to work with Hermite integrals, we have to rewrite Eqs. (3.1.24) and (3.1.25) as we did so with Eqs. (1.2.24) and (1.2.25) for Eqs. (1.3.2) and (1.3.3), respectively. That is, when a term is multiplied by $2a$, $2b$, or $(2p)^{-1}$, index λ, β, or ζ, respectively, is incremented by one. The resulting equations are (omitting the ket sides and displaying the λ, β, ζ indices on the right for a more transparent presentation)

$$\begin{aligned}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta+1} - X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta+1,\zeta+1} &- \tilde{i}_a(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda-1,\beta+1,\zeta+1} \\ &+ \tilde{i}_b(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta+1} \quad (3.1.40)\end{aligned}$$

and

$$\begin{aligned}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta+1} + X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda+1,\beta,\zeta+1} &+ \tilde{i}_a(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta+1} \\ &- \tilde{i}_b(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda+1,\beta-1,\zeta+1} \,. \quad (3.1.41)\end{aligned}$$

Transforming Eq. (3.1.6) the same way we get

$$\begin{aligned}(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} + X_{\mathrm{AB}}(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} &+ \tilde{i}_a(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda-1,\beta,\zeta} \\ &- \tilde{i}_b(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda,\beta-1,\zeta} \,. \quad (3.1.42)\end{aligned}$$

We will use these equations for the ${}^3\mathrm{GHP}_{\mathrm{Herm,AB}}$ schemes. In ${}^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$, $\tilde{\boldsymbol{l}}_a$ and $\tilde{\boldsymbol{l}}_b$ are built up by Eqs. (3.1.40) and (3.1.41), respectively, while in ${}^3\mathrm{GHP}_{\mathrm{Herm,AB2}}$ Eq. (3.1.42) is applied to increment $\tilde{\boldsymbol{l}}_b$.

To employ mixed integrals for **A** derivatives we transform Eqs. (3.1.30) and (3.1.33), respectively, as

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta+1} - X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda,\beta+1,\zeta+1} - \tilde{i}_a(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}|_{\lambda-1,\beta+1,\zeta+1} \quad (3.1.43)$$

and

$$\begin{aligned}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b+\mathbf{1}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta+1} + X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda+1,\beta,\zeta+1} &+ \tilde{i}_a(\Omega^{\bar{\boldsymbol{l}}_p}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta+1} \\ &+ \bar{i}_p(\Omega^{\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} \,, \quad (3.1.44)\end{aligned}$$

and also Eq. (3.1.9) as

$$(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b+\mathbf{1}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a+\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} + X_{\mathrm{AB}}(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a,\boldsymbol{l}_b}|_{\lambda,\beta,\zeta} + \tilde{i}_a(\Omega^{\mathbf{0}}_{\tilde{\boldsymbol{l}}_a-\tilde{\mathbf{1}}_x,\boldsymbol{l}_b}|_{\lambda-1,\beta,\zeta} \,. \quad (3.1.45)$$

In the $\mathrm{GHP}_{\mathrm{Mixed,A}}$ algorithms Eq. (3.1.43) is exploited to build up $\tilde{\boldsymbol{l}}_a$, while to $\boldsymbol{l}_b$ we apply Eqs. (3.1.44) and (3.1.45) in $\mathrm{GHP}_{\mathrm{Mixed,A1}}$ and $\mathrm{GHP}_{\mathrm{Mixed,A2}}$, respectively. For **B** derivatives Eqs. (3.1.35) and (3.1.36) are rewritten, respectively, as

$$\begin{aligned}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a+\mathbf{1}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta+1} - X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta+1,\zeta+1} &+ \tilde{i}_b(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta+1} \\ &+ \bar{i}_p(\Omega^{\bar{\boldsymbol{l}}_p-\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} \quad (3.1.46)\end{aligned}$$

TABLE 3.3: Serial numbers of the equations to be applied to calculate the necessary classes in the various GHP-based algorithms. Eqs. (2.1.1) and (1.2.5) are omitted since all the schemes start with the use of these relations. See also the caption of Table 3.1.

Separate recursion	
$\mathrm{GHP}_{\mathrm{Cart,A1}}$, $\mathrm{GHP}_{\mathrm{Cart,B1}}$, and $\mathrm{GHP}_{\mathrm{Mixed,C1}}$	(1.3.2), (1.3.3)
$\mathrm{GHP}_{\mathrm{Cart,A2}}$, $\mathrm{GHP}_{\mathrm{Cart,B2}}$, and $\mathrm{GHP}_{\mathrm{Mixed,C2}}$	(1.3.2), (2.1.3)
$\mathrm{GHP}_{\mathrm{Mixed,A1}}$	(3.1.43), (3.1.44)
$\mathrm{GHP}_{\mathrm{Mixed,A2}}$	(3.1.43), (3.1.45)
$\mathrm{GHP}_{\mathrm{Mixed,B1}}$	(3.1.46), (3.1.47)
$\mathrm{GHP}_{\mathrm{Mixed,B2}}$	(3.1.46), (3.1.48)
Common recursion	
${}^3\mathrm{GHP}_{\mathrm{Cart,AB1}}$, ${}^3\mathrm{GHP}_{\mathrm{Cart,AC1}}$, and ${}^3\mathrm{GHP}_{\mathrm{Cart,BC1}}$	(1.3.2), (1.3.3)
${}^3\mathrm{GHP}_{\mathrm{Cart,AC2}}$,${}^3\mathrm{GHP}_{\mathrm{Cart,AC2}}$, and ${}^3\mathrm{GHP}_{\mathrm{Cart,BC2}}$	(1.3.2), (2.1.3)
${}^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$	(3.1.40), (3.1.41)
${}^3\mathrm{GHP}_{\mathrm{Mixed,AC1}}$	(3.1.43), (3.1.44)
${}^3\mathrm{GHP}_{\mathrm{Mixed,BC1}}$	(3.1.46), (3.1.47)
${}^3\mathrm{GHP}_{\mathrm{Herm,AB2}}$	(3.1.40), (3.1.42)
${}^3\mathrm{GHP}_{\mathrm{Mixed,AC2}}$	(3.1.43), (3.1.45)
${}^3\mathrm{GHP}_{\mathrm{Mixed,BC2}}$	(3.1.46), (3.1.48)

and

$$(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\bar{\boldsymbol{l}}_p+\bar{\mathbf{1}}_x}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta+1} + X_{\mathrm{AB}}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda+1,\beta,\zeta+1} - \tilde{i}_b(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda+1,\beta-1,\zeta+1}\ , \quad (3.1.47)$$

while Eq. (3.1.11) is recast as

$$(\Omega^{\mathbf{0}}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b+\tilde{\mathbf{1}}_x}|_{\lambda,\beta,\zeta} = (\Omega^{\mathbf{0}}_{\boldsymbol{l}_a+\mathbf{1}_x,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} + X_{\mathrm{AB}}(\Omega^{\mathbf{0}}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b}|_{\lambda,\beta,\zeta} - i_b(\Omega^{\mathbf{0}}_{\boldsymbol{l}_a,\tilde{\boldsymbol{l}}_b-\tilde{\mathbf{1}}_x}|_{\lambda,\beta-1,\zeta}\ . \quad (3.1.48)$$

Eq. (3.1.46) is utilized for $\boldsymbol{l}_a$ in the $\mathrm{GHP}_{\mathrm{Mixed,B}}$ schemes, and $\tilde{\boldsymbol{l}}_b$ is built up by Eqs. (3.1.47) and (3.1.48) in $\mathrm{GHP}_{\mathrm{Mixed,B1}}$ and $\mathrm{GHP}_{\mathrm{Mixed,B2}}$, respectively. The presentation of the necessary index ranges for the intermediates required by Eqs. (3.1.40) to (3.1.48) is omitted since the presence of the scaling indices makes these ranges quite complicated. The reader can find these index ranges in the Supplementary Material of Ref. 112 [113]. The considered GHP-based schemes are given in Table 3.3.

3.1.4 Rys polynomial method

To calculate the integral derivatives via the Rys method we consider two options. The first one is to evaluate the undifferentiated ERIs necessary for the direct construction of the derivatives by either Eq. (1.1.4) or (1.1.10). In Chapter 2 it was found that for Cartesian ERIs it is always more advantageous to calculate only the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)}$ classes with the quadrature and proceed with OS recursions thereafter. To compute derivatives, the 2D integrals $\Theta_x^{i_a,0,0}(t_r^2)$ (and their y and z counterparts) have to be evaluated by Eq. (2.1.21).

To use Hermite or mixed Gaussian ERIs in the Rys scheme a method to compute the 2D integrals over such functions has to be given. This can be achieved in the same way as the Cartesian recursions were derived in Sect. 1.5. That is, if we find the coefficients to expand the desired bra side overlap distribution in unscaled Hermite Gaussians, we can define the 2D integrals analogously to Eq. (2.1.20). Then, we can use the recurrence relations of the coefficients to arrive at recursions for these integrals. Following the similar derivation in Ref. 46 consider the x-dependent part of an $(\tilde{\boldsymbol{l}}_a\tilde{\boldsymbol{l}}_b|$ bra:

$$\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b} = \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} H_{\bar{i}_p} \ . \tag{3.1.49}$$

Incrementing $\tilde{i}_a$ by one and inserting Eq. (1.1.9) we get

$$x_{\mathrm{A}}\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b} - \frac{\tilde{i}_a}{2a}\tilde{H}_{\tilde{i}_a-\tilde{1}}\tilde{H}_{\tilde{i}_b} = \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{1}+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a+\tilde{1},\tilde{i}_b} H_{\bar{i}_p} \ . \tag{3.1.50}$$

Rewriting x_{A} according to Eq. (1.2.2), utilizing Eq. (3.1.49), and using Eq. (1.1.12) on $x_{\mathrm{P}}H_{\bar{i}_p}$ we obtain

$$\begin{aligned}
\sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{1}+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a+\tilde{1},\tilde{i}_b} H_{\bar{i}_p} &= \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b}\left(\frac{1}{2p}H_{\bar{i}_p+\bar{1}} + \bar{i}_p H_{\bar{i}_p-\bar{1}}\right) + X_{\mathrm{PA}}\sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} H_{\bar{i}_p} \\
&- \frac{\tilde{i}_a}{2a}\sum_{\bar{i}_p=0}^{\tilde{i}_a-\tilde{1}+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} H_{\bar{i}_p} \\
&= \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{1}+\tilde{i}_b} H_{\bar{i}_p}\left(\frac{1}{2p}E_{\bar{i}_p-\bar{1}}^{\tilde{i}_a,\tilde{i}_b} + (\bar{i}_p+\bar{1})E_{\bar{i}_p+\bar{1}}^{\tilde{i}_a,\tilde{i}_b}\right) + X_{\mathrm{PA}}\sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} H_{\bar{i}_p} \\
&- \frac{\tilde{i}_a}{2a}\sum_{\bar{i}_p=0}^{\tilde{i}_a-\tilde{1}+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} H_{\bar{i}_p} \ .
\end{aligned} \tag{3.1.51}$$

Collecting the terms belonging to the unscaled Hermite Gaussian of the same order we write

$$E_{\bar{i}_p}^{\tilde{i}_a+\tilde{1},\tilde{i}_b} = X_{\mathrm{PA}}E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} + \frac{1}{2p}E_{\bar{i}_p-\bar{1}}^{\tilde{i}_a,\tilde{i}_b} + (\bar{i}_p+\bar{1})E_{\bar{i}_p+\bar{1}}^{\tilde{i}_a,\tilde{i}_b} - \frac{\tilde{i}_a}{2a}E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} \ . \tag{3.1.52}$$

After incrementing $\tilde{i}_b$ in Eq. (3.1.49) we arrive at

$$E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b+\tilde{1}} = X_{\mathrm{PB}}E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} + \frac{1}{2p}E_{\bar{i}_p-\bar{1}}^{\tilde{i}_a,\tilde{i}_b} + (\bar{i}_p+\bar{1})E_{\bar{i}_p+\bar{1}}^{\tilde{i}_a,\tilde{i}_b} - \frac{\tilde{i}_b}{2b}E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b-\tilde{1}} \ . \tag{3.1.53}$$

Eqs. (3.1.52) and (3.1.53) use the starting value defined in Eq. (1.2.13). From Eqs. (1.2.13), (1.2.14), and (1.2.15) it follows that the $E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b}$ coefficients do not depend on P_x, hence we can apply Eq. (1.2.15) to Eq. (3.1.49) to write

$$2a\tilde{H}_{\tilde{i}_a+\tilde{1}}\tilde{H}_{\tilde{i}_b} + 2b\tilde{H}_{\tilde{i}_a}\tilde{H}_{\tilde{i}_b+\tilde{1}} = \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} H_{\bar{i}_p+\bar{1}} \ . \tag{3.1.54}$$

On the left-hand side we utilize Eqs. (1.1.9), (3.1.49), and

$$ax_{\mathrm{A}} + bx_{\mathrm{B}} = px_{\mathrm{P}} \ , \tag{3.1.55}$$

while on the right-hand side we substitute Eq. (1.1.12) to get

$$\begin{aligned} 2p & \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} x_{\mathrm{P}} H_{\bar{i}_p} - \tilde{i}_a \sum_{\bar{i}_p=0}^{\tilde{i}_a-\tilde{1}+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} H_{\bar{i}_p} - \tilde{i}_b \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b-\tilde{1}} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b-\tilde{1}} H_{\bar{i}_p} \\ = & \ 2p \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} x_{\mathrm{P}} H_{\bar{i}_p} - 2p \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} H_{\bar{i}_p-\bar{1}} \ . \end{aligned} \tag{3.1.56}$$

After canceling terms and collecting the E values belonging to $H_{\bar{i}_p}$ we obtain

$$\tilde{i}_a E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} + \tilde{i}_b E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b-\tilde{1}} = 2p(\bar{i}_p+\bar{1}) E_{\bar{i}_p+\bar{1}}^{\tilde{i}_a,\tilde{i}_b} \ , \tag{3.1.57}$$

which has the same structure as Eq. (1.2.18). To arrive at the analogue of Eq. (1.2.19) for Hermite Gaussians we substitute Eq. (3.1.57) into the third term in Eq. (3.1.52) and exploit that, from $p = a + b$,

$$\frac{1}{2p} - \frac{1}{2a} = -\frac{1}{2p}\frac{b}{a} \tag{3.1.58}$$

to get

$$E_{\bar{i}_p}^{\tilde{i}_a+\tilde{1},\tilde{i}_b} = X_{\mathrm{PA}} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} + \frac{1}{2p} E_{\bar{i}_p-\bar{1}}^{\tilde{i}_a,\tilde{i}_b} + \frac{\tilde{i}_b}{2p} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b-\tilde{1}} - \frac{\tilde{i}_a}{2p}\frac{b}{a} E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} \ , \tag{3.1.59}$$

and similarly

$$E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b+\tilde{1}} = X_{\mathrm{PB}} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} + \frac{1}{2p} E_{\bar{i}_p-\bar{1}}^{\tilde{i}_a,\tilde{i}_b} + \frac{\tilde{i}_a}{2p} E_{\bar{i}_p}^{\tilde{i}_a-\tilde{1},\tilde{i}_b} - \frac{\tilde{i}_b}{2p}\frac{a}{b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b-\tilde{1}} \ . \tag{3.1.60}$$

Defining the three-center 2D integrals over Hermite Gaussians as

$$\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c}(t^2) = \Big(-\frac{1}{2c}\Big)^{\tilde{i}_c} \sum_{\bar{i}_p=0}^{\tilde{i}_a+\tilde{i}_b} E_{\bar{i}_p}^{\tilde{i}_a,\tilde{i}_b} \mathcal{H}_{\bar{i}_p+\tilde{i}_c}(t^2) \ , \tag{3.1.61}$$

then, as it was done in Sect. 1.5, Eqs. (1.5.8), (3.1.59), (3.1.60), and (3.1.57) are applied to obtain

$$\begin{aligned} \Theta_x^{\tilde{i}_a+\tilde{1},\tilde{i}_b,\tilde{i}_c}(t^2) &= \Big(X_{\mathrm{PA}} - \frac{\alpha}{p}X_{\mathrm{PC}}t^2\Big)\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c}(t^2) + \frac{\tilde{i}_a}{2p}\Big(-\frac{b}{a} - \frac{\alpha}{p}t^2\Big)\Theta_x^{\tilde{i}_a-\tilde{1},\tilde{i}_b,\tilde{i}_c}(t^2) \\ &+ \frac{\tilde{i}_b}{2p}\Big(1 - \frac{\alpha}{p}t^2\Big)\Theta_x^{\tilde{i}_a,\tilde{i}_b-\tilde{1},\tilde{i}_c}(t^2) + \frac{\alpha\tilde{i}_c t^2}{2pc}\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c-\tilde{1}}(t^2) \ , \end{aligned} \tag{3.1.62}$$

$$\begin{aligned}\Theta_x^{\tilde{i}_a,\tilde{i}_b+\tilde{1},\tilde{i}_c}(t^2) &= \Big(X_{\mathrm{PB}}-\frac{\alpha}{p}X_{\mathrm{PC}}t^2\Big)\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c}(t^2)+\frac{\tilde{i}_a}{2p}\Big(1-\frac{\alpha}{p}t^2\Big)\Theta_x^{\tilde{i}_a-\tilde{1},\tilde{i}_b,\tilde{i}_c}(t^2)\\ &+ \frac{\tilde{i}_b}{2p}\Big(-\frac{a}{b}-\frac{\alpha}{p}t^2\Big)\Theta_x^{\tilde{i}_a,\tilde{i}_b-\tilde{1},\tilde{i}_c}(t^2)+\frac{\alpha\tilde{i}_c t^2}{2pc}\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c-\tilde{1}}(t^2)\ , \qquad (3.1.63)\end{aligned}$$

and

$$\begin{aligned}\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c+\tilde{1}}(t^2) &= \frac{\alpha}{c}X_{\mathrm{PC}}t^2\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c}(t^2)-\frac{\tilde{i}_c\alpha t^2}{2c^2}\Theta_x^{\tilde{i}_a,\tilde{i}_b,\tilde{i}_c-\tilde{1}}(t_n^2)\\ &+ \frac{\alpha t^2}{2pc}[\tilde{i}_a\Theta_x^{\tilde{i}_a-\tilde{1},\tilde{i}_b,\tilde{i}_c}(t^2)+\tilde{i}_b\Theta_x^{\tilde{i}_a,\tilde{i}_b-\tilde{1},\tilde{i}_c}(t^2)]\ . \qquad (3.1.64)\end{aligned}$$

Turning to the mixed Gaussian overlap distributions it should be noted that during the derivation of Eqs. (1.2.11) and (3.1.52) [and also Eqs. (1.2.12) and (3.1.53)] only the function centered on **A** (or **B**) was manipulated. This means that for $E_{\tilde{i}_p}^{\tilde{i}_a,i_b}$ type coefficients Eqs. (3.1.52) and (1.2.12) are valid, just like Eqs. (1.2.11) and (3.1.53) for the computation of the $E_{\tilde{i}_p}^{i_a,\tilde{i}_b}$ quantities. If we consider the expansion

$$\tilde{H}_{\tilde{i}_a}G_{i_b}=\sum_{\tilde{i}_p=0}^{\tilde{i}_a+i_b}E_{\tilde{i}_p}^{\tilde{i}_a,i_b}H_{\tilde{i}_p} \qquad (3.1.65)$$

and apply Eq. (1.2.15), we can write

$$2a\tilde{H}_{\tilde{i}_a+\tilde{1}}G_{i_b}+2b\tilde{H}_{\tilde{i}_a}G_{i_b+1}-i_b\tilde{H}_{\tilde{i}_a}G_{i_b-1}=\sum_{\tilde{i}_p=0}^{\tilde{i}_a+i_b}E_{\tilde{i}_p}^{\tilde{i}_a,i_b}H_{\tilde{i}_p+\tilde{1}}\ . \qquad (3.1.66)$$

Utilizing Eqs. (1.1.3), (1.1.9), (3.1.66), and (3.1.55) it can be seen that an equation analogous to Eqs. (1.2.18) and (3.1.57) also holds for the $E_{\tilde{i}_p}^{\tilde{i}_a,i_b}$ coefficients, and also for the $E_{\tilde{i}_p}^{i_a,\tilde{i}_b}$ ones, which can be proven in a completely similar manner. Since only Eqs. (3.1.52) and (3.1.57) are used to derive Eq. (3.1.59), and Eq. (3.1.59) and (3.1.57) to do so for Eq. (3.1.62), it follows that the recurrence to increment $\tilde{i}_a$ in, for example, $\Theta_x^{\tilde{i}_a,i_b,\tilde{i}_c}(t^2)$ has the same structure as Eq. (3.1.62), as well as the relation to build up $\tilde{i}_b$ in $\Theta_x^{i_a,\tilde{i}_b,\tilde{i}_c}(t^2)$ is similar to Eq. (3.1.63). By the same argument it can be seen that i_a in $\Theta_x^{i_a,\tilde{i}_b,\tilde{i}_c}(t^2)$ is incremented by an expression corresponding to Eq. (2.1.17), and similarly for i_b in $\Theta_x^{\tilde{i}_a,i_b,\tilde{i}_c}(t^2)$. Because the proof for Eq. (3.1.64) requires only Eqs. (3.1.61), (1.5.4), and (3.1.57), the equation is suitable for the buildup of $\tilde{i}_c$ in all types of 2D integrals. The same is true for Eq. (2.1.18) regarding the increment of i_c.

For brevity, the working equations for the Cartesian, Hermite, and mixed Gaussian 2D ERIs are presented in a single expression because of their very similar form. The 2D integrals $\Theta_x^{i_a,0,0}(t_r^2)$ for $0\le i_a\le L_a+1$ are computed by

$$\Theta_x^{i_a+1,0,0}(t^2)=\Big(X_{\mathrm{PA}}-\frac{\alpha}{p}X_{\mathrm{PC}}t^2\Big)\Theta_x^{i_a,0,0}(t^2)+\frac{i_a}{2p}\Big(\omega_{ba}-\frac{\alpha}{p}t^2\Big)\Theta_x^{i_a-1,0,0}(t^2)\ , \qquad (3.1.67)$$

where $\omega_{ba} = -b/a$ when the function centered on **A** is a Hermite Gaussian, and 1 otherwise. From these, we can compute $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(n)}$ on the quadrature. When we can use Eq. (3.1.3), the necessary multiplication by $w_r t_r^{2L_c}$ is easily accounted for by starting the recursion for the z direction from $\Theta_z^{0,0,0}(t^2) = \theta_{pc}\kappa_{ab}w_r t_r^{2L_c}$, hence we can use Eq. (2.1.29). When **C** is differentiated, we have, however, more than one n values to consider due to the fact that Eqs. (3.1.4) or (3.1.7) have to be used. Here the quadrature formula have to explicitly contain the scaling with t_r^{2n} resulting in the more expensive equation

$$(\boldsymbol{l}_a\boldsymbol{l}_b|\boldsymbol{l}_c)^{(n)} = \sum_{r=1}^{N_{rts}} \Theta_x^{i_a,i_b,i_c}(t_r^2)\Theta_y^{j_a,j_b,j_c}(t_r^2)\Theta_z^{k_a,k_b,k_c}(t_r^2)t_r^{2n} \tag{3.1.68}$$

and a similar one for Hermite and mixed Gaussian auxiliary ERIs. Thus in these cases we will also inspect the schemes where Eq. (2.1.29) is used to construct the target ERIs, and also when the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ integrals are calculated by it. From Eq. (3.1.64) the 2D recurrence for incrementing i_c is

$$\Theta_x^{i_a,0,i_c+1}(t^2) = \frac{\alpha}{c}X_{\mathrm{PC}}t^2\Theta_x^{i_a,0,i_c}(t^2) + \frac{\alpha i_a t^2}{2pc}\Theta_x^{i_a-1,0,i_c}(t^2) + \frac{i_c}{2c}\Big(\omega_c - \frac{\alpha t^2}{c}\Big)\Theta_x^{i_a,0,i_c-1}(t^2)\ , \tag{3.1.69}$$

where $\omega_c = 1$ if the function on **C** is a Cartesian and 0 if a Hermite Gaussian. Note that in the cases where **C** is not differentiated, because of the same reasons as discussed in Subsect. 2.1.4, the third term in Eq. (3.1.69) could be omitted if the ket side was transformed into the solid harmonic basis, resulting in an equation being analogous to Eq. (2.1.28). Finally a recurrence to increment i_b can be obtained by subtracting the equation to build up i_a from the recursion to increment i_b for each type of 2D integral to get

$$\begin{aligned}\Theta_x^{i_a,i_b+1,i_c}(t^2) &= \Theta_x^{i_a+1,i_b,i_c}(t^2) + X_{\mathrm{AB}}\Theta_x^{i_a,i_b,i_c}(t^2) + \omega_a\frac{i_a}{2a}\Theta_x^{i_a-1,i_b,i_c}(t^2)\\ &- \omega_b\frac{i_b}{2b}\Theta_x^{i_a,i_b-1,i_c}(t^2)\ ,\end{aligned} \tag{3.1.70}$$

where $\omega_a = 1$ if the **A** function is a Hermite and 0 if it is a Cartesian Gaussian, and analogously for ω_b. As for the MD and GHP methods, there are no advantages the Cartesian Gaussian ERIs offer for calculating **C** derivatives compared to the mixed integrals, hence in these cases only the mixed algorithms will be characterized. Similarly to the MD schemes, it turns out that the route applying Eq. (3.1.68) and then the HRR and VRR equations appropriate for the applied Gaussians is the most efficient choice for the **C** derivatives, hence only this method will be investigated for the ^{3}RYS algorithms.

The other option is the approach of Flocke and Lotrich [64], where the 2D integrals are differentiated, and Eq. (2.1.29) directly produces the differentiated ERIs:

$$\frac{\partial}{\partial A_x}(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c) = \sum_{r=1}^{N_{rts}} \frac{\partial\Theta_x^{I_a,I_b,I_c}(t_r^2)}{\partial A_x}\Theta_y^{J_a,J_b,J_c}(t_r^2)\Theta_z^{K_a,K_b,K_c}(t_r^2)\ . \tag{3.1.71}$$

TABLE 3.4: Serial numbers of the equations to be applied to calculate the necessary classes in the various Rys-based algorithms. See also the caption of Table 3.1.

Separate recursion	
$\mathrm{RYS}_{\mathrm{Cart,A}}$ and $\mathrm{RYS}_{\mathrm{Cart,B}}$	(3.1.67), (2.1.29), (1.7.8), (2.1.3)
$\mathrm{RYS}_{\mathrm{Mixed,A}}$	(3.1.67), (2.1.29), (1.7.8), (3.1.9)
$\mathrm{RYS}_{\mathrm{Mixed,B}}$	(3.1.67), (2.1.29), (1.7.8), (3.1.11)
$\mathrm{RYS}_{\mathrm{Mixed,C1}}$	(3.1.67), (3.1.69), (3.1.70), (3.1.68)
$\mathrm{RYS}_{\mathrm{Mixed,C2}}$	(3.1.67), (3.1.69), (3.1.68), (2.1.3)
$\mathrm{RYS}_{\mathrm{Mixed,C3}}$	(3.1.67),(3.1.68), (3.1.14), (2.1.3)
Common recursion	
$^3\mathrm{RYS}_{\mathrm{Cart,AB}}$	(3.1.67), (2.1.29), (1.7.8), (2.1.3)
$^3\mathrm{RYS}_{\mathrm{Herm,AB}}$	(3.1.67), (2.1.29), (1.7.8), (3.1.6)
$^3\mathrm{RYS}_{\mathrm{Cart,AC}}$ and $^3\mathrm{RYS}_{\mathrm{Cart,BC}}$	(3.1.67), (3.1.68), (3.1.4), (2.1.3)
$^3\mathrm{RYS}_{\mathrm{Mixed,AC}}$	(3.1.67), (3.1.68), (3.1.14), (3.1.9)
$^3\mathrm{RYS}_{\mathrm{Mixed,BC}}$	(3.1.67), (3.1.68), (3.1.14), (3.1.11)

This scheme will be further referred to as $\mathrm{RYS}_{\mathrm{2D}}$ (e.g., $\mathrm{RYS}_{\mathrm{Cart,A2D}}$ when we use Cartesian Gaussians and differentiate with respect to $\mathbf{A}$). Since the nuclear coordinate dependent parts of the 2D integrals are the x, y, or z components of the Cartesian or Hermite Gaussian basis functions, they follow the differentiation rule of Eq. (1.1.4) or (1.1.10). The discussed schemes are summarized in Table 3.4, with the exception of the $\mathrm{RYS}_{\mathrm{2D}}$ schemes: these algorithms always calculate the 2D integrals with Eqs. (3.1.67), (3.1.69), and (3.1.70), compute their derivatives, and then apply Eq. (3.1.71).

3.1.5 Algorithmic considerations

The contraction of the primitive functions for a shell triplet is performed inside a nested loop structure with paying specific attention to the treatment of uncontracted AOs and the use of cache-friendly buffer arrays as described in Subsect. 2.1.5.

The main algorithmic difference between the approaches based on Cartesian and Hermite Gaussians for the OS, MD, and RYS methods is the applicability of the HRR at the contracted level. The fact that Eq. (2.1.3) does not depend on any Gaussian exponents is an obvious advantage of the Cartesian Gaussians. However, for the $(\tilde{\boldsymbol{L}}_a\tilde{\boldsymbol{L}}_b|$ and $(\tilde{\boldsymbol{L}}_a\boldsymbol{L}_b|$ bra sides with $L_b = 1$ and for the $(\boldsymbol{L}_a\tilde{\boldsymbol{L}}_b|$ ones with $L_b = 2$ only one of the classes in the HRR recursion tree will be multiplied by a factor with an exponent, that is, $([\tilde{\boldsymbol{L}}_a - \mathbf{1}_x]\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ using Eqs. (3.1.6) and (3.1.9), and $(\boldsymbol{L}_a\mathbf{0}|\tilde{\boldsymbol{L}}_c)$ using Eq. (3.1.11). If these classes are multiplied by their respective exponent dependent factors at the primitive level, the HRR can be performed on the contracted basis in these cases. This is noted because AOs with angular momenta greater than 2 are rarely contracted. In Chapter 2 it was found that while working with contracted integrals generally lowers the FLOP count, performing the operations at the primitive level is usually more cache friendly. The reason for this is that most cache misses happen during the reading or writing of large arrays that store the partially and fully contracted integrals. The calculation steps which are performed inside the primitive loops use the same small and fixed length arrays for every exponent triplet, while, at the contracted level, the data have to be read from and written into arrays that store all the classes for a shell triplet. The size of these arrays, and thus the number of cache misses can be reduced by applying the solid harmonic transformation at the primitive level. Such algorithms are especially advantageous when prescreening is applied for the evaluation of the primitive classes, which can greatly reduce the actual work performed inside the primitive loops. We also note that the HRR work can be reduced in the cases when we evaluate $\mathbf{B}$ derivatives for $L_a = L_b$ shell triplets by switching the two functions in the bra, then proceeding by differentiating the first function.

It is not trivial to decide at what stage of the algorithms the solid harmonic transformations and the construction of the derivatives should be carried out. The solid harmonic transformations reduce the number of components for further operations, but their own costs also depend on what number of intermediates they are performed on. Constructing the derivatives of a Hermite Gaussian function with angular momentum $\tilde{\boldsymbol{L}}$ with Eq. (1.1.23) always increases the number of the intermediates compared to the $\tilde{\boldsymbol{L}} + \tilde{\mathbf{1}}_x$ shell, which is required for this operation. This is not necessary true for Cartesian Gaussians, because applying Eqs. (1.1.4) and then (1.1.22) decreases the number of components compared to the sum of the components in the $\boldsymbol{L}+\mathbf{1}_x$ and $\boldsymbol{L}-\mathbf{1}_x$ shells if $L > 3$. The possible stages where these transformations can be performed also vary from scheme to scheme. For example, in a $\mathrm{GHP}_{\mathrm{Cart,A1}}$ algorithm, if we first build up $\boldsymbol{L}_a$ with Eq. (1.3.2) and then $\boldsymbol{L}_b$ with Eq. (1.3.3), we can perform both the differentiation and the solid harmonic transformation of $\boldsymbol{L}_a$ after using Eq. (1.3.2). With a $\mathrm{GHP}_{\mathrm{Mixed,A1}}$ scheme, where Eqs. (3.1.43) and (3.1.44) are used, we do not have this option since Eq. (3.1.44) depends on $\tilde{\boldsymbol{l}}_a$. Note also that, when Cartesian Gaussian derivatives are calculated with an algorithm employing Eq. (2.1.3), moving the HRR outside of the primitive loops makes it impossible to use common intermediates for the buildup of the two classes that appear in Eq. (1.1.4) since the intermediates for the term with the increased angular momentum have to be multiplied by an exponent-dependent factor before the contraction. These considerations make it very complicated to say which order of operations will be the most preferable for a given algorithm. The most efficient positions for the various computational steps were thus determined by calculating floating point operation counts.

3.1.6 Prescreening of three-center integral derivatives

For the prescreening of significant ERI derivative batches we consider two approaches. A straightforward strategy is to apply the Cauchy–Schwarz inequality, which, for example, for the first derivative of the first function, is written as [84]

$$\Big|\Big(\frac{\partial \boldsymbol{L}_a}{\partial A_x}\boldsymbol{L}_b|\boldsymbol{L}_c\Big)\Big| \leq \sqrt{\Big|\Big(\frac{\partial \boldsymbol{L}_a}{\partial A_x}\boldsymbol{L}_b|\frac{\partial \boldsymbol{L}_a}{\partial A_x}\boldsymbol{L}_b\Big)\Big|}\sqrt{|(\boldsymbol{L}_c|\boldsymbol{L}_c)|}\ . \tag{3.1.72}$$

The calculation of the second derivatives appearing in Eq. (3.1.72) is a challenging task, and the efficiency of evaluating the quantities necessary for prescreening becomes important, especially when the three-center derivative integral list is only computed once for a given geometry. We can reduce the cost of this type of prescreening if we use Eq. (3.1.72) to determine upper bounds for primitive ERI derivatives instead of derivatives of ERIs over AOs, and multiply these with the maximal contraction coefficients and contraction degrees belonging to the primitives in question. A batch of AOs are then screened out using the largest of the associated primitive bounds. This method has the advantage that the calculation of the required second derivative integrals scales with the second power of the number of primitive functions, which would be fourth power if we aimed to give upper bounds for AO ERI derivatives. Furthermore, $\mathbf{R}_{\mathrm{PQ}}$ becomes 0 in these cases, which makes the Boys functions trivial to calculate [35]. An MD based calculation is even more simplified since the one-center ERIs given by Eq. (1.2.5) reduce to

$$(\bar{\boldsymbol{l}}_u + \bar{\mathbf{1}}_x)^{(n)} = \bar{i}_u(\bar{\boldsymbol{l}}_u - \bar{\mathbf{1}}_x)^{(n+1)}\ . \tag{3.1.73}$$

Eq. (3.1.73) implies that only those one-center ERIs have a nonzero value where $\bar{i}_u$, $\bar{j}_u$, and $\bar{k}_u$ have an even parity. Using

$$(\mathbf{0})^{(n)} = (-p)^n\theta_{pp}\kappa_{ab}^2 F_n(0)\ , \tag{3.1.74}$$

where it was exploited that $2\alpha = p$ since $p = q$, the nonzero one-center ERIs are computed as

$$(\bar{\boldsymbol{l}}_p) = (\bar{i}_p - 1)!!(\bar{j}_p - 1)!!(\bar{k}_p - 1)!!\kappa_{ab}^2\theta_{pp}(-p)^{l_p/2}F_{l_p/2}(0)\ . \tag{3.1.75}$$

Since the $(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_q)$ ERIs will also be nonzero only when $\bar{i}_p + \bar{i}_q$, etc. have even parity, the cost of an assembly of the type of Eq. (2.1.11) is reduced. It is also advantageous to use Hermite Gaussians to evaluate the second derivatives. Note that for the application of Eq. (1.1.23) we only need to calculate the cases where $\tilde{\boldsymbol{l}}_a$ and $\tilde{\boldsymbol{l}}_b$ denote the same solid harmonic or differentiated solid harmonic Gaussians in the bra and the ket. In addition to screening entire AO batches, the calculated primitive bounds are readily used to screen the evaluation of primitive integral derivatives. In the following this prescreening scheme will be referred to as **C-S** when it is only applied to screen shell triplets, and when primitive ERI derivatives are also screened the notation will be **C-S prim**. In a scheme where one computes all the derivatives of a shell triplet at the same time Eq. (3.1.72) can be used to see if the derivatives with respect to all three centers will be significant. If not, a less expensive algorithm computing only the derivatives with respect to a single center can be applied, along with Eq. (3.0.1). This approach will not be pursued in the present work.

Another option one could use for the prescreening is to assume from Eq. (1.6.4) that

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)| \leq |(\mathbf{ss}|\mathbf{s})| \, . \tag{3.1.76}$$

In Chapter 2 it was concluded that this approximation results in an efficient bound with satisfactory accuracy. From Eqs. (1.1.4) and (2.1.30) we can write for the derivatives of the first function that

$$\left|\frac{\partial}{\partial A_x}(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)\right| \leq |2\max(a,b,c)+\max(L_a,L_b,L_c)]\theta_{pc}\kappa_{ab}\min\Big(1,\sqrt{\frac{\pi}{4\alpha R_{\mathrm{PC}}^2}}\Big) \tag{3.1.77}$$

and use this for the screening of AO batches and primitive ERI derivatives as well. The screening method using this approximate bound will be denoted as **sss** and **sss prim**, respectively. The efficiency of both methods and the error introduced by this latter screening strategy will be analyzed.

3.2 Floating point operation counts

To decide which of the various discussed methods is the most beneficial we performed a preliminary FLOP counting, similar to the one discussed in Sect. 2.2. The considered computational steps include the evaluation of the two- and three-center auxiliary quantities, the buildup of the primitive Gaussian ERIs, the construction of derivatives for Cartesian integrals by Eq. (1.1.4), and the transformation into the solid harmonic and contracted bases. The model system consists of three separate carbon atoms, hence the derivatives with respect to three centers are considered. The program counts the FLOP requirements for every possible route which can be derived by varying the order of the operations that can be performed at different parts of the algorithms. The spherical harmonic transformation of $\boldsymbol{L}_a$, $\boldsymbol{L}_b$, or $\boldsymbol{L}_c$ can be carried out at any place where no more operations depend on $\boldsymbol{l}_a$, $\boldsymbol{l}_b$, or $\boldsymbol{l}_c$, respectively. For the derivatives Eq. (1.1.4) or Eq. (1.1.23) can be applied after there is no more dependence on the shell to be differentiated for any recursion, even at the contracted level if the multiplication with the double of the exponent is performed before the contraction. Eq. (2.1.3) can be carried out anywhere after the $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ intermediates have been produced. As explained in Sect. 3.1.5 the same is possible for Eqs. (3.1.6) and (3.1.9) with $\tilde{L}_b = 1$ and for Eq. (3.1.11) with $L_b \leq 2$. For the MD$_1$ and GHP$_1$ type schemes the order in which the bra-side angular momenta are built up is also arbitrary. For the GHP schemes we only considered routes where at least one of the bra-side recursions is performed at the contracted level since, when both angular momenta are constructed in the primitive basis, the scheme is equal to an MD algorithm. The VRR equations, the one-center recursion of the MD and GHP schemes, and the 2D recursions and the quadrature assembly of the Rys method are executed at the primitive level.

Tables 3.5 to 3.8 display the FLOP counts for the various schemes. The numbers for each shell triplet with the best order of the computational steps are presented in the Supplementary Material of Ref. 112 [113], and the conclusions regarding the efficiency of the methods for different angular momenta will rely on these results.

Table 3.5 contains the FLOP requirements of the OS-based algorithms. Here, using the mixed Gaussian integrals for each shell triplet reduces the FLOP count by 7-4 % for the $\mathbf{A}$, 2-1 % for the $\mathbf{B}$, and 22-32 % for the $\mathbf{C}$ derivatives for the DZ-5Z basis sets compared to applying only the $\mathrm{OS}_{\mathrm{Cart}}$ schemes. These savings are 1-5 %, 5-19 %, and 6-24 %, respectively, compared to the case when we only use the $\mathrm{OS}_{\mathrm{Herm}}$ algorithms. As L_b increases, the simplicity of Eq. (2.1.3) compensates for the extra work required for the shell triplet with the decreased angular momentum stemming from the last term of Eq. (1.1.4). Generally, when $L_b \geq 3$, $\mathrm{OS}_{\mathrm{Cart}}$ becomes the best choice for $\mathbf{A}$ and $\mathbf{B}$ derivatives. In the latter case, the L_b that we have to build up is increased by one, but Eq. (3.1.11) is also more efficient than Eq. (3.1.9) since it requires fewer intermediates. For $\mathbf{C}$ derivatives $\mathrm{OS}_{\mathrm{Mixed,C}}$ is always the best choice. The FLOP counts using mixed integrals for differentiating with respect to one center are nearly optimal, that is, they are on average about 1 % higher than the counts resulting from the best suited Gaussian combination ($\mathrm{OS}_{\mathrm{Cart}}$, $\mathrm{OS}_{\mathrm{Herm}}$, or $\mathrm{OS}_{\mathrm{Mixed}}$) for every shell triplet. The most efficient way of evaluating all three nuclear derivatives is ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$. Here we only have one L_c to build up, so there are more common intermediates for Eq. (2.1.3) that are shared among the different required shell triplets than in the ${}^3\mathrm{OS}_{\mathrm{Cart,AC}}$ and ${}^3\mathrm{OS}_{\mathrm{Cart,BC}}$ cases. The ket side VRR is also cheaper, while the HRR is more efficient than for ${}^3\mathrm{OS}_{\mathrm{Herm,AB}}$. Applying the ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ scheme requires 12-20 % fewer FLOPs for calculating all the derivatives for each shell triplet in the DZ-5Z basis sets than for evaluating the derivatives with respect to the two cheapest centers separately and recover the third one via Eq. (3.0.1). This is the decrease in the cost that is attributed to the exploitation of the fact that the recursion to compute the derivatives for the two centers share recursion intermediates.

The results for the MD algorithms are presented in Table 3.6. The conclusions for the schemes are very similar to those for OS. For the $\mathbf{C}$ derivatives of the shell triplets with small bra angular momenta and high $\tilde{L}_c$ the $\mathrm{MD}_{\mathrm{Mixed,C2}}$ scheme demands the fewest FLOPs, however, using $\mathrm{MD}_{\mathrm{Mixed,C3}}$ for every shell triplet only results in 3-1 % increase in the cost. The ${}^3\mathrm{MD}_{\mathrm{PR}}$ algorithm is only competitive in the $(\boldsymbol{ss}|\boldsymbol{s})$ case.

TABLE 3.5: FLOP counts for the various OS-based algorithms with the cc-pVXZ (X=D,T,Q,5) basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D ($\boldsymbol{dd}\vert\boldsymbol{f}$)	T ($\boldsymbol{ff}\vert\boldsymbol{g}$)	Q ($\boldsymbol{gg}\vert\boldsymbol{h}$)	5 ($\boldsymbol{hh}\vert\boldsymbol{i}$)
$\mathrm{OS}_{\mathrm{Cart,A}}$	834039	3586356	19552557	90066913
$\mathrm{OS}_{\mathrm{Cart,B}}$	878444	3774088	20114653	90629426
$\mathrm{OS}_{\mathrm{Cart,C}}$	963485	4685749	28813487	145680780
$\mathrm{OS}_{\mathrm{Herm,A}}$	781668	3397430	19121889	91472513
$\mathrm{OS}_{\mathrm{Herm,B}}$	904647	4135692	23578592	111565131
$\mathrm{OS}_{\mathrm{Herm,C}}$	793992	3960789	24936806	130619870
$\mathrm{OS}_{\mathrm{Mixed,A}}$	778308	3333764	18431265	86727029
$\mathrm{OS}_{\mathrm{Mixed,B}}$	858221	3691859	19875872	90166857
$\mathrm{OS}_{\mathrm{Mixed,C}}$	749055	3445287	20089055	98798104
${}^{3}\mathrm{OS}_{\mathrm{Cart,AB}}$	1331840	5712048	30371707	136326843
${}^{3}\mathrm{OS}_{\mathrm{Cart,AC}}$	1782179	8575091	51458794	253017583
${}^{3}\mathrm{OS}_{\mathrm{Cart,BC}}$	1914024	9009709	52763347	255133654
${}^{3}\mathrm{OS}_{\mathrm{Herm,AB}}$	1419613	6213209	33738005	154821887
${}^{3}\mathrm{OS}_{\mathrm{Mixed,AC}}$	1508292	7288759	43730208	217161994
${}^{3}\mathrm{OS}_{\mathrm{Mixed,BC}}$	1628374	7579886	43448054	207249588

TABLE 3.6: FLOP counts for the various MD-based algorithms with the cc-pVXZ (X=D,T,Q,5) basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D ($\boldsymbol{dd}\vert\boldsymbol{f}$)	T ($\boldsymbol{ff}\vert\boldsymbol{g}$)	Q ($\boldsymbol{gg}\vert\boldsymbol{h}$)	5 ($\boldsymbol{hh}\vert\boldsymbol{i}$)
$\mathrm{MD}_{\mathrm{Cart,A}}$	916213	4043754	21895647	100400942
$\mathrm{MD}_{\mathrm{Cart,B}}$	958288	4218790	22394767	100711632
$\mathrm{MD}_{\mathrm{Mixed,A}}$	865972	3827843	21051221	98842722
$\mathrm{MD}_{\mathrm{Mixed,B}}$	941824	4146520	22178126	100298776
$\mathrm{MD}_{\mathrm{Mixed,C1}}$	909103	5157925	35351909	197374785
$\mathrm{MD}_{\mathrm{Mixed,C2}}$	842553	4521333	28852865	150054265
$\mathrm{MD}_{\mathrm{Mixed,C3}}$	787388	3780076	22521838	111879063
${}^{3}\mathrm{MD}_{\mathrm{Cart,AB}}$	1413790	6166368	32693989	146570332
${}^{3}\mathrm{MD}_{\mathrm{Cart,AC}}$	2032851	9988407	59788722	293139967
${}^{3}\mathrm{MD}_{\mathrm{Cart,BC}}$	2160810	10406048	61017238	294977623
${}^{3}\mathrm{MD}_{\mathrm{Herm,AB}}$	1511629	6721972	36405233	167086412
${}^{3}\mathrm{MD}_{\mathrm{Mixed,AC}}$	1721805	8452087	50373647	249640343
${}^{3}\mathrm{MD}_{\mathrm{Mixed,BC}}$	1825007	8630926	49302884	234896789
${}^{3}\mathrm{MD}_{\mathrm{PR}}$	3111352	19451574	142582639	848103741

The FLOP counts of the investigated GHP paths are displayed in Table 3.7. With the GHP schemes, the use of the best Gaussian type results in 6-3 % and 3-1 % (0-7 % and 0-2 %) lower FLOP requirements than using only Cartesian Gaussian (mixed Gaussian) integrals for the **A** and **B** derivatives, respectively. Applying only GHP_1 type schemes costs 2-21 %, 2-24 %, and 1-23 % more than using the best route for the **A**, **B**, and **C** derivatives, respectively. For GHP_2 these values are 2-0 %, 4-1 %, and 2-0 %. As L_b or $\tilde{L}_b$ increases, the application of the HRR becomes more favored since it requires fewer kinds of λ, β, ζ scaled intermediates than the GHP recursions. Cartesian Gaussians are preferred for higher bra sides, especially for **A** derivatives. This is because for these classes it is more advantageous to perform the recursion for the first angular momentum at the primitive level, and, while for **B** derivatives the necessary intermediates for the HRR are the same in this case for the mixed and the Cartesian ERIs, for the **A** derivatives there are more HRR intermediates for the mixed integrals. When differentiating **A**, the preferred algorithm is $\mathrm{GHP}_{\mathrm{Cart,A2}}$ for $L_a \geq 3$ and $L_b \geq 1$ and also for $(\boldsymbol{dd}|$ bras, and $\mathrm{GHP}_{\mathrm{Mixed,A1}}$ for the rest of the shell triplets. For **B** $\mathrm{GHP}_{\mathrm{Cart,B2}}$ is the best choice for $(\boldsymbol{dd}|$,$(\boldsymbol{ff}|$,$(\boldsymbol{gf}|$,$(\boldsymbol{gg}|$,$(\boldsymbol{hg}|$, and $(\boldsymbol{hh}|$ bras, $\mathrm{GHP}_{\mathrm{Mixed,B1}}$ for bra sides up to $(\boldsymbol{dp}|$ and also $(\boldsymbol{fs}|$, and $\mathrm{GHP}_{\mathrm{Mixed,B2}}$ for the other shell triplets. In the **C** case, where we only investigated the mixed Gaussian algorithms, the $\mathrm{GHP}_{\mathrm{Mixed,C2}}$ route is the best when $L_a \geq 3$ and $L_b \neq 0$ and also for $(\boldsymbol{dd}|$ bra sides, otherwise $\mathrm{GHP}_{\mathrm{Mixed,C1}}$ is the method of choice. If we wish to calculate the three derivatives in a common algorithm, ${}^3\mathrm{GHP}_{\mathrm{Mixed,AC1}}$ is preferred for $L_b = 0$, ${}^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$ for $\tilde{L}_a = \tilde{L}_b = 1$, and ${}^3\mathrm{GHP}_{\mathrm{Herm,AB2}}$ for most of the remaining cases. For a number of shell triplets with $(\boldsymbol{gg}|$,$(\boldsymbol{hg}|$, and $(\boldsymbol{hh}|$ bras ${}^3\mathrm{GHP}_{\mathrm{Cart,AC2}}$ is the most efficient because of the simplicity of Eq. (2.1.3). Calculating the derivatives with the ^{3}GHP schemes reduces the cost by 14-8 % compared to evaluating the two cheapest derivatives and applying Eq. (3.0.1). Unlike in the OS and MD cases this method becomes less efficient as the cardinal number of the basis set increases since performing operations at the contracted level limits the use of common intermediates in the recursions because of the different exponent scalings for the derivatives coming from Eqs. (1.1.4) and (1.1.10).

The results for the RYS schemes are presented in Table 3.8. For the $\mathrm{RYS}_{\mathrm{C}}$ algorithms we observe that the schemes applying VRR for $\tilde{L}_c$ are always the most efficient, thus the results are very similar to the OS case. Compared to Cartesian ERIs, the mixed Gaussian schemes reduce the cost by 7 % and 2 % for the **A** and **B** derivatives, respectively. As with OS, the ${}^3\mathrm{RYS}_{\mathrm{Cart,AB}}$ route is the best way of evaluating all three derivatives at the same time.

Concerning the order of the various operations we observe that it is preferred to carry out the HRR step at the contracted stage for $L_b = 1$ or $\tilde{L}_b = 1$. For the considered basis sets the higher shell triplets are uncontracted, and in these instances applying Eq. (2.1.3) on contracted ERIs results in higher FLOP counts for the $L_b > 2$ cases than doing so at the primitive level, except for the **C** derivatives. This is because the HRR for the two terms in Eq. (1.1.4) cannot share any intermediates if the scaling with $2a$ or $2b$ has already been performed for the primitive ERIs. For example, if an ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ scheme is

TABLE 3.7: FLOP counts for the various GHP-based algorithms with the cc-pVXZ (X=D,T,Q,5) basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D $(\boldsymbol{dd}\|\boldsymbol{f})$	T $(\boldsymbol{ff}\|\boldsymbol{g})$	Q $(\boldsymbol{gg}\|\boldsymbol{h})$	5 $(\boldsymbol{hh}\|\boldsymbol{i})$
$\mathrm{GHP}_{\mathrm{Cart,A1}}$	893738	5206588	34696058	181827684
$\mathrm{GHP}_{\mathrm{Cart,A2}}$	886662	4718681	28260468	136426246
$\mathrm{GHP}_{\mathrm{Cart,B1}}$	932694	5488016	37012933	195655585
$\mathrm{GHP}_{\mathrm{Cart,B2}}$	944946	4993045	29474616	140521684
$\mathrm{GHP}_{\mathrm{Mixed,A1}}$	841451	4782519	31904972	168362436
$\mathrm{GHP}_{\mathrm{Mixed,A2}}$	847333	4577449	28616778	143174999
$\mathrm{GHP}_{\mathrm{Mixed,B1}}$	909815	5259732	35075833	183173164
$\mathrm{GHP}_{\mathrm{Mixed,B2}}$	939050	5002224	29675460	142225345
$\mathrm{GHP}_{\mathrm{Mixed,C1}}$	662485	4118817	30732628	178382077
$\mathrm{GHP}_{\mathrm{Mixed,C2}}$	671760	3855113	25984570	138067909
$^3\mathrm{GHP}_{\mathrm{Cart,AB1}}$	1380773	8382919	55370303	288455800
$^3\mathrm{GHP}_{\mathrm{Cart,AB2}}$	1605952	8784298	54930312	280549055
$^3\mathrm{GHP}_{\mathrm{Cart,AC1}}$	1343819	8593244	60728861	332513082
$^3\mathrm{GHP}_{\mathrm{Cart,AC2}}$	1353354	8067045	52558001	267957804
$^3\mathrm{GHP}_{\mathrm{Cart,BC1}}$	1383307	8878565	62975698	342948786
$^3\mathrm{GHP}_{\mathrm{Cart,BC2}}$	1467546	8511933	54385447	273898475
$^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$	1297987	8213273	57886000	318368393
$^3\mathrm{GHP}_{\mathrm{Herm,AB2}}$	1573955	8192973	49124752	242974909
$^3\mathrm{GHP}_{\mathrm{Mixed,AC1}}$	1298470	8393091	61268253	344809058
$^3\mathrm{GHP}_{\mathrm{Mixed,AC2}}$	1314025	7925813	52946831	275481605
$^3\mathrm{GHP}_{\mathrm{Mixed,BC1}}$	1365938	8882016	64397121	357490370
$^3\mathrm{GHP}_{\mathrm{Mixed,BC2}}$	1466426	8502294	54433543	275552831

applied, the $([\boldsymbol{l}_a+\mathbf{1}_x]\boldsymbol{l}_b|\boldsymbol{l}_c)$ and $(\boldsymbol{l}_a[\boldsymbol{l}_b-\mathbf{1}_x]|\boldsymbol{l}_c)$ classes appear as recursion intermediates for $(\boldsymbol{l}_a[\boldsymbol{l}_b+\mathbf{1}_x]|\boldsymbol{l}_c)$. We cannot exploit this at the contracted stage since these classes will have different scalings. Our results also suggest that it is the most advantageous to perform the solid harmonic transformation of the bra side functions, Eq. (1.1.4) for Cartesian ERIs, and the solid harmonic transformation of the differentiated shell as the last operations in every investigated algorithm. In the case of the OS, MD, and RYS algorithms both the costs of the solid harmonic transformation of the ket function and the HRR are reduced if we transform $\boldsymbol{L}_c$ or $\tilde{\boldsymbol{L}}_c$ before the HRR in most of the cases. The solid harmonic transformation is slightly more expensive this way for $(\tilde{\boldsymbol{L}}_a\tilde{\boldsymbol{L}}_b|$ and $(\tilde{\boldsymbol{L}}_a\boldsymbol{L}_b|$ bra sides with $\tilde{L}_b = 1$ or $L_b = 1$, and for $(\boldsymbol{L}_a\tilde{\boldsymbol{L}}_b|$ bra sides with $\tilde{L}_b = 1$, but the reduction in the FLOP count of the HRR compensates for this. In the most efficient $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ algorithm the $\boldsymbol{L}_c$ solid harmonic transformation is always preferred to be carried out at the primitive stage since the intermediates required by Eq. (2.1.3) are less numerous before the $2a$ and $2b$ scaling of Eq. (1.1.4) is applied. For the solid harmonic transformation of $\tilde{\boldsymbol{L}}_c$ in the GHP

TABLE 3.8: FLOP counts for the various Rys-based algorithms with the cc-pVXZ (X=D,T,Q,5) basis sets. The integrals containing the functions with the highest possible angular momenta for a given basis are also shown.

Algorithm	X			
	D $(\boldsymbol{dd}\vert\boldsymbol{f})$	T $(\boldsymbol{ff}\vert\boldsymbol{g})$	Q $(\boldsymbol{gg}\vert\boldsymbol{h})$	5 $(\boldsymbol{hh}\vert\boldsymbol{i})$
$\text{RYS}_{\text{Cart,A}}$	909536	3821275	20459245	93163062
$\text{RYS}_{\text{Cart,A2D}}$	1584986	7524841	46468848	242012893
$\text{RYS}_{\text{Cart,B}}$	953406	4002579	20982691	93546498
$\text{RYS}_{\text{Cart,B2D}}$	1618478	7631728	46915887	243064918
$\text{RYS}_{\text{Mixed,A}}$	847432	3543792	19249913	89536499
$\text{RYS}_{\text{Mixed,A2D}}$	1562392	7428420	46092369	240372510
$\text{RYS}_{\text{Mixed,B}}$	932616	3918769	20740050	93074206
$\text{RYS}_{\text{Mixed,B2D}}$	1622394	7643554	46907993	242640950
$\text{RYS}_{\text{Mixed,C1}}$	1107643	4917716	27808428	135735251
$\text{RYS}_{\text{Mixed,C2}}$	990588	4423282	24829490	118200624
$\text{RYS}_{\text{Mixed,C3}}$	839702	3841094	22218908	108540338
$\text{RYS}_{\text{Mixed,C2D}}$	2091075	9351259	54608020	274107117
$^3\text{RYS}_{\text{Cart,AB}}$	1409731	5954407	31297787	139483332
$^3\text{RYS}_{\text{Cart,AC}}$	2060190	9785375	57785166	281094275
$^3\text{RYS}_{\text{Cart,BC}}$	2193980	10211841	58993050	282670260
$^3\text{RYS}_{\text{Herm,AB}}$	1497616	6457450	34677023	158057580
$^3\text{RYS}_{\text{Mixed,AC}}$	1721569	8170373	48084554	235930334
$^3\text{RYS}_{\text{Mixed,BC}}$	1857226	8509113	47938028	226251462

schemes we found two options to be efficient for the majority of shell triplets: before the scaling of the $(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{0},\boldsymbol{0}}|\tilde{\boldsymbol{L}}_c) = (\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ classes or after the contraction of the ${}_{\lambda,\beta,\zeta}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{0},\boldsymbol{0}}|\tilde{\boldsymbol{L}}_c)$ ones. Transforming $\tilde{\boldsymbol{L}}_c$ before the scaling means that we do not have to perform this operation for all the necessary λ, β, ζ scaled classes. However, in this case, first we have to transform the $(\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ integrals into ${}_{0,0,0}(\Omega^{\bar{\boldsymbol{l}}_p}_{\boldsymbol{0},\boldsymbol{0}}|\tilde{\boldsymbol{L}}_c)$ ones according to Eq. (2.1.9), so the scaling itself and the contraction of the primitive functions will become more expensive because of the increased number of intermediates. In most of the cases the second route is more efficient when the second function in the bra side is $\boldsymbol{s}$ and also for $(\boldsymbol{pp}|$ bras. Otherwise the first option is more efficient. Picking the better route for each shell triplet is 15-5 % and 3-33 % cheaper than going with only the first or second option, respectively, for **A** derivatives. These savings are 15-4 % and 2-36 % for **B** derivatives.

Tables 3.5 to 3.8 imply that for most scenarios the OS based algorithms provide the lowest FLOP count. Let us take a look at if any benefit can be predicted from using the best suited algorithm for each shell triplet compared to applying exclusively OS based schemes. Our results show that the MD schemes do not offer significant advantage in any of the cases. It turns out that for a lot of $L_c = 0$ or $\tilde{L}_c = 0$ shell triplets the Rys scheme provides the smallest FLOP count. However, the gain is rather limited since the saving it introduces is less than 1 %, thus, in the following we do not consider this algorithm. The

GHP method, in turn, considerably lowers the FLOP count for certain shell triplets. If we use the most advantageous OS or GHP algorithm for each triplet, the FLOP requirement is reduced by 6-1 %, 7-1 %, and 17-5 % for the **A**, **B**, and **C** derivatives, respectively. For the **A** and **B** derivatives the $\mathrm{GHP}_{\mathrm{Mixed,A1}}$ and $\mathrm{GHP}_{\mathrm{Mixed,B1}}$ schemes are preferred for most of the shell triplets with $(\boldsymbol{ss}|$, $(\boldsymbol{ps}|$, and $(\boldsymbol{pp}|$ bras, while for higher bra angular momenta the GHP recursions become inefficient with the investigated basis sets. In the case of **C** derivatives the $\mathrm{GHP}_{\mathrm{Mixed,C1}}$ algorithm is the best method for a number of $L_b = 0$ and $L_b = 1$ cases. As $\tilde{L}_c$ increases, the advantage of Eq. (1.2.5) over Eq. (3.1.14) becomes more pronounced, although for higher L_b values $\mathrm{OS}_{\mathrm{Mixed,C}}$ is still superior. Compared to using only $^3\mathrm{OS}_{\mathrm{Cart,AB}}$, the combined application of $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ and $^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$ reduces the cost of calculating the derivatives by 16-2 %. Up to $(\boldsymbol{pp}|$ bras the GHP method is the better choice, and for derivatives with $(\boldsymbol{ds}|$ bra side the $\tilde{L}_c$ from which $^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$ is more efficient depends on the basis set. As the cardinal number increases the ratio of the primitive and contracted functions becomes smaller, so recursions at the contracted stage are less preferred. However, with increasing $\tilde{L}_c$ building up this angular momentum with Eq. (1.2.5) becomes more advantageous, and the primitive contraction is also cheaper when performed on the $_{\lambda,\beta,\zeta}(\bar{\boldsymbol{l}}_p + \tilde{\boldsymbol{L}}_c)$ intermediates. Applying the most efficient ^{3}OS and ^{3}GHP schemes costs 22-20 % fewer FLOPs for the DZ-5Z bases than calculating the derivatives with respect to the cheapest two centers for every shell triplet and then using Eq. (3.0.1). This is an improved result compared to applying only the OS or GHP schemes since now we utilize both the advantage of GHP for small-angular-momentum bra sides and the superiority of OS for higher angular momenta. If only $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ and $^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$ are considered this reduction in cost changes to 20% for each basis set. For this reason in the following I only deal with the implementation of the $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ schemes along with the ^{3}GHP algorithms that are predicted to be the best performing ones.

It should be noted, however, that we are comparing very dissimilar algorithms with a different amount of memory operations. For example, the work associated with the transformation of one-center Hermite ERIs to two-center ones via Eq. (2.1.9) is not accounted for since it does not require any FLOPs if the exponent scalings have been taken care of beforehand. For this reason it is necessary to compare the actual wall time performances of the two approaches to decide which one is more beneficial. We also note that the most advantageous GHP algorithms turned out to be the ones using mixed or Hermite Gaussian ERIs. With the exception of the **C** derivatives, these always use the newly derived recursions, Eqs. (3.1.40) to (3.1.48).

Algorithm 4 Overview of the separate recursion algorithms. D denotes the center of differentiation.

```
Loop for $D$
  Loop for $A$
    Loop for $l_a$
      Loop for $B$
        Loop for $l_b$
          Loop for $C$
            If $D = A$, $B$, or $C$
              Loop for $l_c$
                sss or C-S screening
                  Call generated codes
```

Algorithm 5 Overview of the common recursion algorithms.

```
Loop for $A$
  Loop for $l_a$
    Loop for $B$
      Loop for $l_b$
        Loop for $C$
          Loop for $l_c$
            sss or C-S screening
              Call generated codes
```

3.3 Implementation

We implemented the $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ algorithms as well as the best performing ^{3}GHP schemes for each shell triplet, which include the approaches $^3\mathrm{GHP}_{\mathrm{Mixed,AC1}}$, $^3\mathrm{GHP}_{\mathrm{Herm,AB1}}$, $^3\mathrm{GHP}_{\mathrm{Herm,AB2}}$, and $^3\mathrm{GHP}_{\mathrm{Cart,AC2}}$, by means of automated code generation. Similarly to the implementation of the ERI code detailed in Sect. 2.3 there is an optimized, generated subroutine for every shell triplet. The code generation in this case was performed using the results of the FLOP counter program directly: for a given method and shell triplet the counter program determines the optimal order of the operations and the code generator prints out the corresponding subroutines. For the $^3\mathrm{OS}_{\mathrm{Cart,AB}}$ schemes, in addition to the `dfint_triplets_der.f`, `intsub_der.f`, and `brasub_der.f` source files, which contain subroutines analogous to those used in the evaluation of undifferentiated ERIs, the files `hrr_der.f` and `derspher.f` were also produced. The former file contains subroutines that perform the Cartesian HRR for the classes required to evaluate the integral derivatives, and also the solid harmonic transformation of $\boldsymbol{L}_c$ if the HRR is performed at the primitive level. Subroutines that transform $\boldsymbol{L}_a$ and $\boldsymbol{L}_b$ to the solid harmonic basis and evaluate the **A** and **B** derivatives of the ERIs reside in the `derspher.f` source file. In the case of the GHP algorithms there are different subroutine types for Eq. (1.2.5), the transformation of one-center ERIs to two-center ones, the scaling of the bra side according to Eq. (1.3.1), the solid harmonic transformation of $\tilde{\boldsymbol{L}}_c$ for bra side derivatives, the MD recursions, the GHP recursions, the HRR, and the remaining solid harmonic transformations and production of the differentiated ERIs by either Eq. (1.1.22) or (1.1.23), in addition to the drivers of the evaluation of primitive ERIs and differentiated AO ERIs, analogously to the OS case. The implementation of the GHP method involves more types of subroutines compared to OS because the algorithm involves more steps, and the optimal order of operations varies from class to class in more aspects.

The solid harmonic transformations, the contraction of the primitives, and the recursions are as vectorized as possible, and the unnecessary components of the intermediate shells are omitted from the calculation [99]. Schematic representations of the algorithms applying separate and common recursions are presented in Algorithm 4 and 5, respectively. Both of the algorithms use a general driver subroutine that handles three-center ERI and ERI derivative calculations for various methods. This driver contains the loops from A (first atom) to l_c (third angular momentum), initializes the necessary arrays and variables, and computes the scaled contraction coefficients for efficient primitive contraction [95]. The order of the loops for the primitive functions in the generated codes is a, b, c. When prescreening is carried out at the primitive stage, Eq. (3.1.72) is applied before the loop for c using the maximal $\sqrt{|(\boldsymbol{L}_c|\boldsymbol{L}_c)|}$ and $\sqrt{|(\partial\boldsymbol{L}_c/\partial C_x|\partial\boldsymbol{L}_c/\partial C_x)|}$ for the given **C** and $\boldsymbol{L}_c$, or the smallest and largest c when Eq. (3.1.77) is used. The prescreening ERIs and ERI derivatives for Eq. (3.1.72) are calculated using a general routine which applies the MD scheme using Eq. (1.2.3) and Hermite Gaussian basis functions [46]. This allows us to exploit most of the simplifications discussed in Sect. 3.1.5.

3.4 Performance tests

In the following tests, a medium-sized molecule of 94 atoms, acetoacetyl-CoA was used as the model system. Wall time and cache simulation measurements were performed using a single core of an Intel Core i7-7700K 4.2 GHz CPU. The cache memory of this hardware includes 32-32 kB of data and instruction level 1, 256 kB of level 2, and 8 MB of level 3 cache.

Table 3.9 compares the different prescreening approaches using a threshold of 10^{-10} $\mathrm{E_h}$. We see that the **C-S** screening is more efficient when we apply screening only for the shell triplets, while with primitive level prescreening the **sss prim** method gives a tighter bound. **sss prim**, however, increases the error introduced by roughly an order of magnitude compared to **sss**, which reaches 10^{-7} E_h with the QZ basis. This could be prohibitive when one wishes to converge a geometry optimization to high accuracy. Table 3.10 compares the efficiency of the **C-S** and **C-S prim** screening approaches for the cc-pVXZ and the aug-cc-pVXZ (X=D,T,Q) basis sets. The conclusions are very similar to the case of the undifferentiated ERIs (see Table 2.7). The screening is more efficient for the non-diffuse basis sets, and the application of primitive level screening is of increased importance regarding the aug-cc-pVXZ bases.

Table 3.9: Comparison of the **C-S** and **sss** prescreening approaches for acetoacetyl-CoA with the cc-pVXZ (X=D,T,Q) basis sets. Efficiency refers to the percentage of derivatives screened out, while accuracy is the maximum error larger than 10^{-10} a.u. in the final AO ERI derivatives.

	DZ		TZ		QZ	
	Efficiency	Accuracy	Efficiency	Accuracy	Efficiency	Accuracy
sss	35	$4.7 \cdot 10^{-10}$	40	$9.3 \cdot 10^{-9}$	43	$6.6 \cdot 10^{-8}$
C-S	45	–	46	–	47	–
sss prim	78	$7.6 \cdot 10^{-9}$	78	$3.7 \cdot 10^{-8}$	80	$3.5 \cdot 10^{-7}$
C-S prim	70	–	74	–	76	–

Table 3.10: Efficiency of the **C-S** and **C-S prim** screening of the three-center differentiated ERI evaluation for the acetoacetyl-CoA molecule with the cc-pVXZ and the aug-cc-pVXZ (X=D,T,Q) basis sets. The values give the wall time expressed as percentage of the screening-free calculation.

Basis	DZ	aug-DZ	TZ	aug-TZ	QZ	aug-QZ
C-S	63.80	95.93	56.39	92.96	54.45	77.55
C-S prim	38.73	64.31	34.30	61.62	37.51	50.66

Fig. 3.1 shows the estimated FLOP counts and wall times for those shell triplets in the cc-pVTZ basis where the counts of the GHP method are superior to those of the OS algorithms (the related FLOP counts can be found in the Supplementary Material of Ref. 112 [113], namely in Table SIV for the ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ route, Tables SXXX to SXXXIX for the ${}^3\mathrm{GHP}_{\mathrm{Cart}}$ schemes, Tables SXL and SXLI for the ${}^3\mathrm{GHP}_{\mathrm{Herm}}$, and SXLII to SLI for the ${}^3\mathrm{GHP}_{\mathrm{Mixed}}$ methods). The predictions correlate well with the timings without prescreening up to the $(\boldsymbol{ps}|\boldsymbol{g})$ triplets. For the other cases the discrepancy can be attributed to the work that is not accounted for in the FLOP counts, e.g., the rearrangement of one-center ERIs to two-center ones. The GHP based methods also require more Boys function evaluations. Also, ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ always uses the Cartesian HRR, Eq. (2.1.3), which is more vectorizable than Eq. (3.1.40) or (3.1.43). When prescreening is applied on the primitive level, the ${}^3\mathrm{OS}_{\mathrm{Cart,AB}}$ algorithm always performs better than the GHP ones. This is understandable since the main merit of the GHP scheme is to move much of the work to the contracted stage.

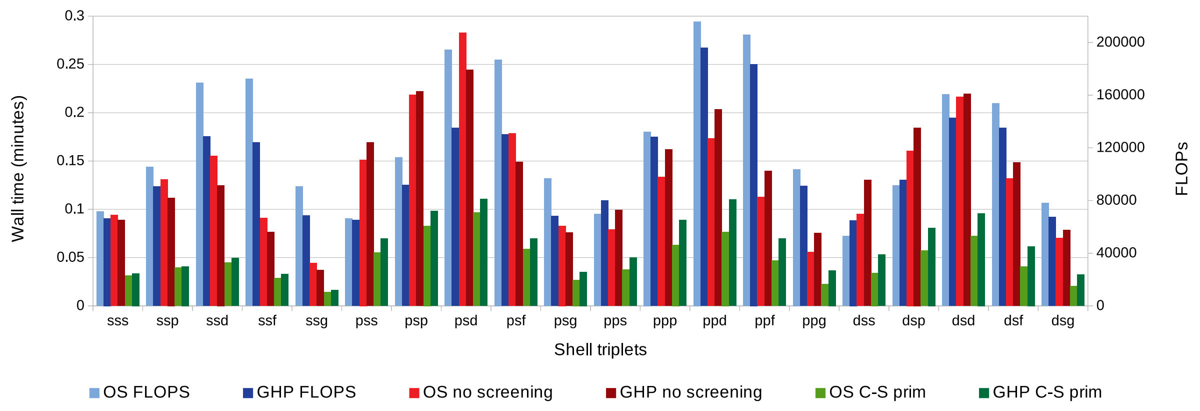

FIGURE 3.1: FLOP counts and wall times for the derivatives of the $(\boldsymbol{ss}|\boldsymbol{s})$-$(\boldsymbol{ds}|\boldsymbol{g})$ shell triplets calculated with the implemented ^{3}OS and ^{3}GHP algorithms for acetoacetyl-CoA with the cc-pVTZ basis set.

To compare the separate and common recursion algorithms the wall times required for the $^3\text{OS}_{\text{Cart,AB}}$ and the OS_{Herm} algorithms were measured, of which the latter I previously implemented by automated code generation, to calculate the derivatives of every shell triplet for our test system. The wall times for the full execution of Algorithms 1 and 2 are displayed in Fig. 3.2, while the contribution of the generated codes to the timings can be seen in Fig. 3.3. Comparing the results in Fig. 3.3 with the times measured without prescreening in Fig. 3.2 we see that, while for $^3\text{OS}_{\text{Cart,AB}}$ the runtime is almost equivalent to the time spent in the generated codes, for OS_{Herm} there is a significant discrepancy increasing with the basis set. This is because of the greater overhead of Algorithm 1 compared to Algorithm 2. Without prescreening, $^3\text{OS}_{\text{Cart,AB}}$ offers speedups of factors of 3.2, 4.7, and 6.6 compared to OS_{Herm} with the DZ, TZ, and QZ bases, respectively. These increase to 4.7, 8.7, and 15.5 if we apply **C-S prim** type prescreening. Screening increases the efficiency of Algorithm 2 more than it does so for Algorithm 1 because it reduces the work performed in the generated codes, which contribute to the total runtime with a higher percentage in the common recursion approach. It is interesting to compare the ratios of the wall times for the two schemes in Fig. 3.3 with the quotients of the summed FLOP counts of $\text{OS}_{\text{Herm,A}}$, $\text{OS}_{\text{Herm,B}}$, $\text{OS}_{\text{Herm,C}}$ and the FLOP counts of $^3\text{OS}_{\text{Cart,AB}}$ from Table 3.5. The ratios of the wall times with the DZ, TZ, and QZ basis sets are 2.1, 2.2, and 2.4, respectively. These values for the FLOP counts are 1.9, 2.0, and 2.2. The good agreement shows that FLOP counting for similar algorithms is a valid tool for predicting their relative runtime performances.

Finally, Table 3.11 contains the results of CPU cache simulations for H_2O_2 performed by the VALGRIND [105] program suite. In most aspects Algorithm 2 produces a

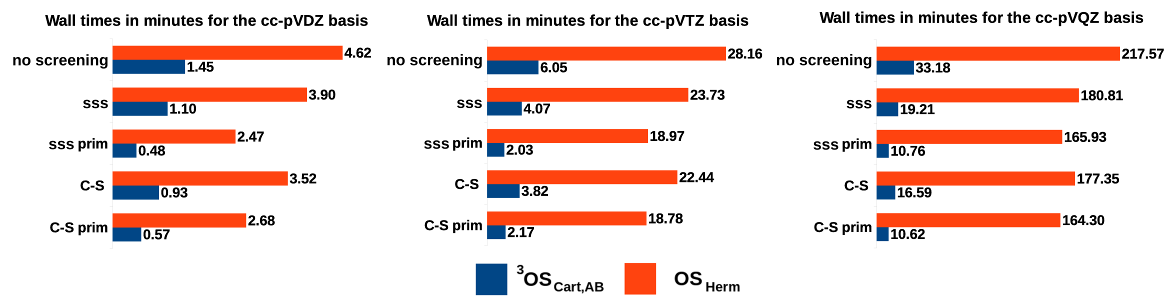

FIGURE 3.2: Wall times in minutes for the execution of Algorithm 1 (red) and Algorithm 2 (blue) for acetoacetyl-CoA with the cc-pVXZ (X=D,T,Q) basis sets. The number of the applied basis functions (AO + fitting basis) was the following: 972+4868 for the DZ, 2196+5634 for the TZ, and 4186+7932 functions for the QZ basis set.

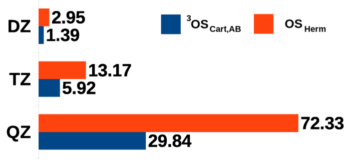

FIGURE 3.3: Contribution (in minutes) of the generated codes to the wall time of Algorithm 1 (red) and Algorithm 2 (blue) without prescreening for acetoacetyl-CoA with the cc-pVXZ (X=D,T,Q) basis sets.

TABLE 3.11: Cache simulation results for the evaluation of the ERI derivatives of H_2O_2 with the cc-pVXZ (X=D,T,Q) basis sets. The results indicate the percentage of cache misses relative to all events. Notations: I and D mean instructions and data, respectively, 1 and L mean that the referenced memory address did not have a copy in the first level of the CPU cache memory or in all of it, respectively. r and w refer to data read and data write operations.

	DZ		TZ		QZ	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
I1	0.21	0.32	0.76	0.72	1.75	1.23
IL	0.01	0.01	0.04	0.01	0.08	0.03
Dr1	0.29	1.32	0.48	2.13	0.87	2.71
DrL	0.01	0.02	0.01	0.01	0.01	0.01
Dw1	1.81	3.72	3.97	6.51	6.75	8.37
DwL	0.05	0.05	0.28	0.05	0.65	0.13

smaller percentage of cache misses because of the more local nature of the common recursion route. The exceptions are the level 1 data read and write misses, where Algorithm 1 performs better. This observation is in accordance with the conclusion in Sect. 2.4 that most data cache misses occur during the contraction of the primitives because of the large size of the arrays involved. Since in Algorithm 2 we calculate derivatives with respect to two centers, there are more integral derivatives to contract, and therefore there are more cache misses.

3.5 Conclusions

The evaluation of the first derivatives of three-center Coulomb integrals over contracted solid harmonic Gaussians in two steps was optimized. First, estimations of the FLOP requirements of several schemes were determined, including ones that apply mixed Gaussian ERIs. The FLOP counts suggest that these integrals containing both Cartesian and Hermite Gaussians are better suited for the task when we compute the derivatives for one center at a time (Algorithm 1) except for bra-side derivatives with $L_b \geq 3$. The OS method is predicted to be superior for most of the cases, however, for shell triplets up to $(\boldsymbol{pp}|$ bra sides $\text{GHP}_{\text{Mixed}}$ produces the lowest FLOP counts. For Algorithm 2 the most efficient scheme is $^3\text{OS}_{\text{Cart,AB}}$, with the exception of $^3\text{GHP}_{\text{Mixed,AC1}}$ for $(\boldsymbol{ss}|$ and $(\boldsymbol{ps}|$, and $^3\text{GHP}_{\text{Herm,AB1}}$ for $(\boldsymbol{pp}|$ and a few other shell triplets with $L_b = 0$ and high L_c. In the second step the wall time performances of certain selected methods were compared. The results for $^3\text{OS}_{\text{Cart,AB}}$ and those ^3GHP schemes which were predicted to be the best based on the FLOP requirement suggest that FLOP counts alone are insufficient to decide between two methods that are algorithmically very different. On the other hand, the wall times for the generated codes for $^3\text{OS}_{\text{Cart,AB}}$ and OS_{Herm} reproduce the ratios of the FLOP counts reasonably well, which suggests that this type of prediction is useful when variants of the same basic algorithm are compared. When we do not only measure the timings for the generated codes but also the total time for the execution of Algorithms 1 and 2, we get significantly larger speedups for the common recursion scheme than expected from the FLOP counts, which is the result of the increased overhead in Algorithm 1.

An efficient approximation of the Cauchy–Schwarz upper bound for ERI derivatives was discussed, and also an inexact screening quantity derived from the Boys function. The test calculations for these approaches reveal that it is important to screen both the primitive integrals and the shell triplets. The **sss prim** type screening is slightly more efficient, but it can introduce errors which are three orders of magnitude greater than the tolerance. This problem does not arise with the **C-S prim** approach. When primitive prescreening is applied, the $^3\text{OS}_{\text{Cart,AB}}$ scheme always performs better than the ^3GHP algorithms. Based on the results I recommend the usage of the Schwarz screening instead of the approximate bound, and the application of $^3\text{OS}_{\text{Cart,AB}}$ when it is not necessary to compute the derivatives with respect to one center at a time. For the separate recursion schemes the OS_{Mixed} approach is predicted to be the most efficient. Our results indicate, however, that in this case the overhead of the loops over the shell triplets of the molecule dominates the wall time rather than the evaluation of the derivatives.

Chapter 4

Applications of the optimized integral code

The efficient local density fitting HF local correlation methods recently developed in our group [1, 2] utilize the rapid integral evaluation procedure described in Chapter 2. In the following the role of three-center ERI evaluation in these calculations will be briefly discussed, and results will be presented stemming from large-scale applications of our DF-HF local natural orbital (LNO) coupled-cluster (CC) with single, double, and perturbative triple excitations [LNO-CCSD(T)] scheme. These computations were performed for one of the key transition states ($\mathrm{TS}^{\mathrm{RS}}_{\mathrm{CC}} \cdots \mathrm{pnp}$) of a Michael-addition reaction, the crambin protein, and the core domain of the HIV-1 integrase enzyme (integrase for short), illustrated in the a), b), and c) parts of Fig. 4.1, respectively. The integrase test system consisted of 2380 atoms and 44035 AO basis functions, and as such, according to our knowledge, represents the largest system for which a correlation calculation was performed using a single processor. For this molecule the details of the (overall rate-determining) DF-HF calculation are also presented. The computations were performed using 127 GB of main memory and 8 cores of a 3.00 GHz Intel Xeon E5-1660 CPU.

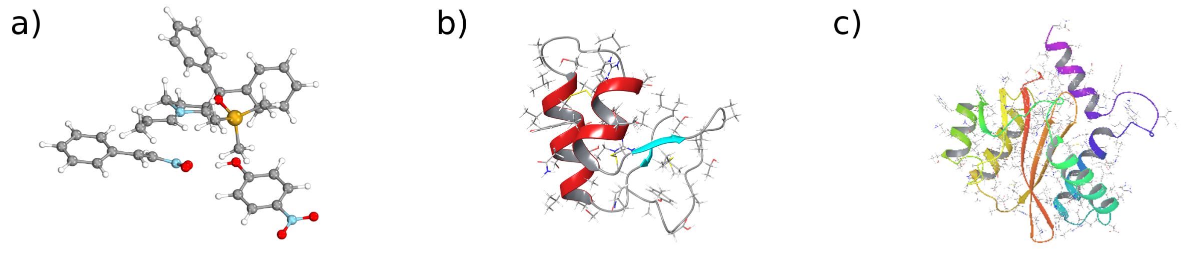

FIGURE 4.1: Test systems for the demonstration of the efficiency of the LNO-CCSD(T) scheme. a) $\mathrm{TS}^{\mathrm{RS}}_{\mathrm{CC}} \cdots \mathrm{pnp}$ b) crambin c) integrase

4.1 Density fitting Hartree–Fock

Most correlation calculations require reference orbitals optimized by HF. In a direct SCF procedure the computation of the ERIs is naturally redundant, and even more so when the DF approximation is invoked, especially with limited amount of main memory. In a DF-HF calculation three-center ERIs are required for two purposes: the construction of the Coulomb and exchange matrices. The Coulomb term **J** is computed in the steps given by the equations

$$g_Q = \sum_{p,q} (\chi_p \chi_q | \rho_Q) \mathcal{P}_{pq} \ , \tag{4.1.1}$$

$$h_R = \sum_Q g_Q V^{-1}_{QR} \ , \tag{4.1.2}$$

and

$$J_{rs} = \sum_R h_R (\chi_r \chi_s | \rho_R) \ , \tag{4.1.3}$$

where $\mathcal{P}_{pq}$ is an element of the single-particle density matrix. For large systems the three-center ERI list usually does not fit into the RAM, hence in practice only a batch of the $(\chi\chi|\rho)$ integrals are computed at a time, and they get discarded after their contribution has been added to the g_Q intermediates. The J_{rs} matrix elements are computed in a similar fashion (the $\mathbf{V}^{-1}$ matrix and the $\mathbf{g}$ and $\mathbf{h}$ vectors can be stored in the RAM), which means that the ERIs are recalculated twice for the buildup of **J** alone.

For the computation of the exchange term **K** first the half-transformed three-center ERIs

$$I_{pi,Q} = (\chi_p \psi_i | \rho_Q) = \sum_q (\chi_p \chi_q | \rho_Q) C_{qi} \tag{4.1.4}$$

are required, where ψ denotes a (localized) MO, and C_{qi} is an MO coefficient. The half-transformed integrals are then used to compute the matrix

$$\mathbf{B} = \mathbf{I}(\mathbf{L}^{\mathrm{T}})^{-1} \ , \tag{4.1.5}$$

where **I** is the matrix containing the $I_{pi,Q}$ integrals, and **L** is the lower triangular Cholesky matrix of **V**, $\mathbf{V} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$. Note that in the **B** and **I** matrices the row index is a superindex composed of AO and LMO indices, while the column index denotes the AF. An element of **K** is finally computed as

$$K_{rs} = \sum_{i,Q} B_{ri,Q} B_{si,Q} \ . \tag{4.1.6}$$

It is often not possible to store the entire tensor holding the **I** intermediates in the main memory. Therefore these integrals are computed for a batch of MOs at a time, and they are discarded after their contribution has been added to **K**. For each such batch, a significant portion of the entire AO ERI list has to be computed. As for the Coulomb term, the integrals are evaluated for one shell triplet at a time, and their contribution is added to the half-transformed ERIs. The prescreening of the AO ERIs computed for this purpose can be made tighter if the screening value is multiplied by the largest C_{qi} coefficient for the given MO batch. A similar trick can be utilized for Eqs. (4.1.1) and (4.1.3), where the screening value for a shell triplet is multiplied by the largest associated element of the density matrix or $\mathbf{h}$, respectively.

To accelerate the computation of the exchange contribution local fitting approximations can be applied [16]. In each iteration step the MOs are localized, and for each localized MO ψ_i a subset of the fitting basis is selected, using which the exchange contribution of ψ_i can be accurately fitted. The selection of these local fitting domains (LFDs) for each MO is detailed in Ref. 1. Its essence is to compute Löwdin atomic charges for ψ_i and include the AFs residing on atoms with a charge greater then 0.05 into the LFD. In addition, an atom A is also included in the domain if the Schwarz estimate of any of the $(\chi_A \chi_B|\rho_{AB})$ ERIs reaches a given threshold (1 $\mathrm{E_h}$ by default), where B is any atom assigned to ψ_i based on the Löwdin charges, and ρ_{AB} is any AF residing on A or B. Utilizing these LFDs the set of necessary AFs is different for each MO in Eqs. (4.1.4)-(4.1.6). After convergence, an extra SCF cycle is performed with the above mentioned threshold set to 10^{-3} $\mathrm{E_h}$ to obtain satisfying accuracy.

The reference orbitals for integrase were produced by a DF-HF calculation using the def2-TZVP basis set [114] and the def2-QZVPP-RI-JK fitting basis [91]. The details of the computation are compiled in Table 4.1. As it can be seen from the table, the local fitting approach drastically reduces the number of AFs necessary for a given i index. Still, utilizing 127 GB of RAM, the computation of $\mathbf{K}$ had to be performed in 20 batches for each SCF cycle, that is, the $(\chi\psi|\rho)$ ERIs could be evaluated for 1/20th of the MOs at a time. The significant $(\chi\chi|\rho)$ ERIs had to be recomputed for every batch. In the last SCF cycle with the extended LFD the calculation had to be divided into 111 batches. Because of the efficient implementation of the AO integrals the limiting step is the evaluation of the half transformed ERIs for the exchange term. This statement can be rationalized by using the data presented in Table 4.1, according to which the AO integral computation and the production of the half transformed ERIs by Eq. (4.1.4) is significantly more demanding than the rest of the calculations performed by the DF-HF algorithm. The division of the computational effort between these two steps (that is, the evaluation of the AO and the half-transformed ERIs) can be approximately characterized by estimating their respective FLOP requirements. The FLOP count of the AO ERI calculation can be assessed by using the program described in Sect. 2.2, which can be easily modified to consider integrals between C, H, O, N, and S atoms using the def2-TZVP basis set and the def2-QZVPP-RI-JK fitting basis. Taking the permutational symmetry of the ERIs into account the evaluation of all of the AO ERIs of the molecule costs approximately

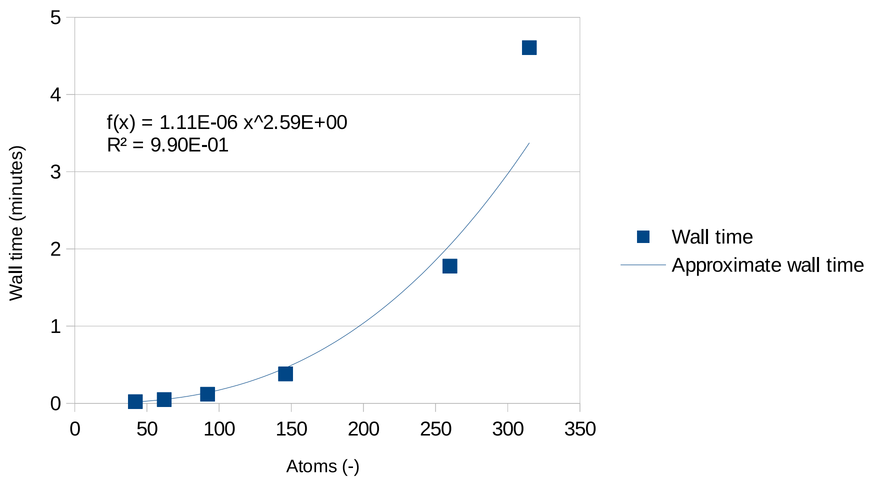

FIGURE 4.2: Approximate wall time of the three-center AO ERI evaluation as a function of the number of atoms in the system. The points of the plot are calculated from the TZ results of Table 2.10.

$6.73 \cdot 10^{15}$ FLOPs. For a system of such size the prescreening of the integrals is a crucial factor in the computational cost. If we take the wall time measurements for the TZ basis presented in Table 2.10 and plot them as a function of the number of atoms in the test molecules, we see that the scaling of the wall time with the number of atoms is $\mathcal{O}(N^{2.59})$ rather than $\mathcal{O}(N^3)$ (see Figure 4.2). If we assume that the Coulomb interaction does not decay significantly in the considered molecules, then the scaling with respect to the atoms in the bra side is $\mathcal{O}(N^{1.59})$. Taking this into account means considering all 2380 atoms for the first center and only $2380^{0.59} = 98$ for the second, that is, we assume that each atom in the bra side interacts significantly with only 98 atoms near to it. Let it also be assumed that the ratio of different atoms in these local domains is the same as for the entire molecule; for example, for the first center 759 out of the 2380 atoms can be carbons, while for the second $759/2380 \cdot 2380^{0.59} \simeq 31.3$ different carbon atoms are considered. With these approximations the estimation for the FLOP count changes to $2.88 \cdot 10^{14}$. Since the ERI evaluations associated with this cost are performed for each batch of MOs, the approximate FLOP cost of the AO integrals is $2.88 \cdot 10^{14} \cdot (20 \cdot 10 + 111) = 8.96 \cdot 10^{16}$. The cost of Eq. (4.1.4) is calculated from the number of AOs, MOs, and the average number of AFs in an LFD from Table 4.1. Assuming an average number of MOs of $4510/20 \simeq 225$ in a batch, and $4510/111 \simeq 40$ for a batch in the last SCF cycle, the cost of the half transformed ERIs is estimated to be $2.92 \cdot 10^{17}$ FLOPs. From these results the AO ERI computing demands roughly 23% of the FLOPs associated with the rate determining step of the DF-HF algorithm. The evaluation of the AO integrals would be the most expensive computational step without the improvements detailed in Chapter 2, which account for a speedup of more than an order of magnitude compared to using a general four-center code and assuming a zero-valued Gaussian exponent on the fourth center.

TABLE 4.1: Details of the DF-HF computation for integrase using the def2-TZVP basis set and the def2-QZVPP-RI-JK fitting basis. The iteration was started from a density matrix converged using the cc-pVDZ basis set for both the AO and the fitting basis. The wall times are given in minutes.

Number of atoms	2380
Number of AOs	44035
Number of AFs	111728
Number of MOs	4510
Average (maximum) number of atoms in LFDs	22.4 (44)
Average (maximum) number of AFs in LFDs	1053 (2068)
Average (maximum) number of atoms in LFDs in the last SCF cycle	133.4 (278)
Average (maximum) number of AFs in LFDs in the last SCF cycle	6270 (13066)
Wall time of the complete calculation	98035
Wall time of the three-center AO integral computing and Eq. (4.1.4)	86019
Wall time of the fitting [Eq. (4.1.5)] and the assembly [Eq. (4.1.6)] of the exchange contribution	9332

4.2 Integral transformations for the second-order Møller–Plesset algorithm

After the HF orbitals have been produced our LNO-CCSD(T) scheme requires a local second-order Møller–Plesset (LMP2) computation so that local natural orbitals can be constructed, and also to obtain energy contributions that can be used to correct some of the approximations of the local CC calculation on the MP2 level. This step is performed using our fragmentation-based, fully integral-direct LMP2 algorithm [1], which assigns an extended domain (ED) to each occupied localized molecular orbital (LMO) and calculates a fragment MP2 energy for each such domain (only valence MOs were considered for our examples). These domains are constructed very carefully to ensure the most compact domains possible; the details of the determination of the domains can be found in Ref. 1, in the following only a brief summary is given. The occupied basis of the extended domain consists of the central LMO, for which the domain is constructed, and other LMOs for which a multipole-based opposite-spin MP2 pair energy estimation with the central LMO exceeds a given threshold (10^{-5} $\mathrm{E_h}$ by default). A list of atoms is compiled requiring that the LMOs can be expanded to a given accuracy in the AOs centered on these atoms, then the LMOs are truncated to the AOs residing on their respective atom lists. The union of these AO sets is the AO basis of the ED. The truncated LMOs are orthogonalized, leaving the central LMO intact. The virtual subspace of the ED is spanned by projected atomic orbitals (PAOs) [115], which are obtained by projecting AOs into the complete virtual subspace. Not all of the AOs in the ED are used for this purpose, only those centered on a subset of the atoms of the ED, denoted as the PAO center domain (PCD). The resulting delocalized PAOs are truncated to the AO basis of the ED and then orthogonalized among each other. Finally, the Fock matrix constructed from the LMOs (PAOs) is diagonalized, and the resulting eigenvectors give the (quasi-)canonical occupied (virtual) MO basis of the ED. The AF basis of the ED does not consist of the AFs that reside on all of the atoms of the ED, but rather only the ones belonging to atoms of the PCD. Note that even with a careful domain construction strategy resulting in compact domains the AO and AF bases of the EDs still have significant overlap. As Table 4.2 shows, each of the EDs of the 3301 LMOs of the integrase enzyme contains on average 138 atoms, while the average number of atoms in the PCDs is 47. Since the total number of atoms is 2380, there will be numerous three-center AO ERIs that will be computed for more than one ED.

To evaluate the MP2 energy of an ED with central LMO ψ_i $(\tilde{\phi}\psi_i|\tilde{\phi}\tilde{\psi})$ type ERIs are necessary, where $\tilde{\psi}$ $(\tilde{\phi})$ denotes a quasi-canonical occupied (virtual) MO. This requires the evaluation of $(\tilde{\phi}\tilde{\psi}|\rho)$ type three-center ERIs for all $\tilde{\psi}$ in the ED and all $\tilde{\phi}$ and ρ functions in the PCD. Additionally, $(\tilde{\phi}\psi_i|\rho)$ integrals are also needed for the central LMO. The three-center ERIs are calculated in the following steps. First, one shell triplet of the $(\chi\chi|\rho)$ AO integrals is computed. Then, the contribution of these ERIs is added to the half-transformed integrals using Eq. (4.1.4) with the sum running over the functions centered on the second atom of the shell triplet. Using this relatively small batch of AO integrals implies that there is probably still a copy of them in the lower levels of the CPU cache, reducing the overhead coming from data traffic. When only an LMP2 calculation is performed, the $(\chi\psi|\rho)$ ERIs are accumulated for all the AOs and LMOs of the ED, and for a group of auxiliary functions belonging to the same atom with the same angular momentum. If the LMP2 calculation is a step of the LNO-CCSD(T) algorithm, however, all of the half-transformed ERIs are kept in the memory for a given ED. The other AO index in the bra is transformed into the pseudocanonical virtual basis to obtain $(\tilde{\phi}\psi|\rho)$ type ERIs:

$$(\tilde{\phi}_a\psi_i|\rho_P) = \sum_p (\chi_p\psi_i|\rho_P)\mathcal{T}_{pa} \ , \tag{4.2.1}$$

where $\mathcal{T}_{pa}$ is an element of the $\chi \rightarrow \tilde{\phi}$ transformation matrix. The total number of these integrals is about 2 orders of magnitude smaller than the number of ERIs over AOs, so they can be kept in the main memory for the remaining operations, in which the LMO functions are transformed into the pseudocanonical LMO basis, and the $(\tilde{\phi}\psi_i|\tilde{\phi}\tilde{\psi})$ four-center ERIs are produced via Eq. (1.7.5). Before computing a shell triplet of AO integrals the following Schwarz bound based estimates are evaluated:

$$E_{aq,P} = \max\left[\sqrt{(\chi_p\chi_q|\chi_p\chi_q)}\right] \max\left[\sqrt{(\rho_P|\rho_P)}\right] \max(\mathcal{T}_{pa}) \ , \tag{4.2.2}$$

and

$$E_{ai,P} = \max[E_{aq,P} \max(C_{qi}), E_{ap,P} \max(C_{pi})] \ . \tag{4.2.3}$$

The p, q, and P indices refer to the functions on the first, second, and third center of the shell triplet, respectively, while for i and a all ψ and $\tilde{\phi}$ functions have to be considered. Thus, $E_{ai,P}$ is an upper bound for the contribution of any ERI in the shell triplet to the $(\tilde{\phi}\psi|\rho)$ type integrals. The contribution of the AO ERIs to the half-transformed integrals for one LMO can be screened by the tighter bound $E_{aq,P} \max(C_{qi})$, where i is the index of the LMO in question. The evaluation of the $(\tilde{\phi}\psi|\rho)$ ERIs is a rapid step, hence the integral transformation part of the LMP2 calculation is dominated by the computation of AO ERIs and the half transformed ERIs appearing in the right-hand side of Eq. (4.2.1), produced by Eq. (4.1.4). Using the same estimation procedure that for the DF-HF algorithm the FLOP count for the AO ERI computation in a single ED is $4,49 \cdot 10^{11}$ operations, where the entries of Table 4.2 regarding the average atom numbers of the EDs and the PCDs were used. The cost of the $(\chi_p\psi_i|\rho_P)$ integrals for an ED is $1.92 \cdot 10^{12}$ FLOPs calculated from the average number of AOs and occupied LMOs in an ED and the average number of AFs in a PCD. Considering these estimations, and noting from Ref. 1 that the AO ERI evaluations and transformations account for about 50 % of the total wall time of an LMP2 calculation of this size, the contribution of the AO integral computation to the wall time can be taken to be roughly 10 %. Here, similar to the direct SCF algorithm, this step could easily be the rate-determining one if the AO ERI evaluation was an order of magnitude slower. Note that the above estimations consider neither the prescreening of the AO ERIs nor that of the half-transformed integrals.

4.3 Integral transformations for the coupled-cluster algorithm

After the MP2 computation is concluded for a given ED, the next step in the LNO-CCSD(T) scheme is to construct the (truncated) LNO basis of the local interacting subspace (LIS) of the central LMO, in which the CCSD(T) calculation will be carried out.

The occupied and virtual LNOs (further denoted as $\bar{\psi}$ and $\bar{\phi}$, respectively) of the LIS are obtained by building, transforming, and diagonalizing the appropriate blocks of the MP2 density matrix of the ED; the process is detailed in Ref. 2. Next, three-center ERIs in the LNO basis are required for the CCSD(T) calculation in the LIS, namely $(\bar{\phi}\bar{\psi}|\rho)$, $(\bar{\psi}\bar{\psi}|\rho)$, and $(\bar{\phi}\bar{\phi}|\rho)$ type integrals. The first of these lists is gained by transforming the $(\tilde{\phi}\tilde{\psi}|\rho)$ integrals available from the MP2 computation into the LNO basis. The $(\bar{\psi}\bar{\psi}|\rho)$ ERIs are also easy to evaluate by computing the $(\tilde{\psi}\psi|\rho)$ intermediates at the same time when Eq. (4.2.1) is applied, transforming ψ to $\tilde{\psi}$ at the same step when this is done for the $(\tilde{\phi}\psi|\rho)$ ERIs, then transforming the integrals into the LNO basis after the LNOs have been determined (note that in this case the factor responsible for the highest $\mathcal{T}_{pa}$ in Eq. (4.2.2) also have to include the largest matrix element for the $\chi \to \tilde{\psi}$ transformation). There are no intermediates at hand, however, for the computation of the so-called two-external list, containing the $(\bar{\phi}\bar{\phi}|\rho)$ integrals. The direct transformation of AOs into virtual LNOs is about an order of magnitude more expensive than the transformation of AOs into LMOs since the latter functions are more local and there are usually 3 times fewer LMOs in the ED than LNOs in the LIS. In our implementation the cost of this transformation is reduced by computing approximate versions of the two-external integrals which are numerically very close to the original ones. For this purpose a new set of functions, denoted as ϕ', is defined which, when projected into the subspace of virtual LNOs, give functions very similar to the ones resulting from projecting the PAOs into the virtual LNO subspace. Then, instead of evaluating the $(\bar{\phi}\bar{\phi}|\rho)$ ERIs, $(\bar{\phi}'\bar{\phi}'|\rho)$ ones are calculated where $\bar{\phi}'$ is a $\bar{\phi}$ virtual LNO projected into the subspace spanned by the ϕ' functions. The definition of the ϕ' set and the rationalization of using the ERIs over $\bar{\phi}'$ functions is given in Ref. 2; for now the only relevant fact is that these integrals can be efficiently expressed by ERIs over the AOs of the PCD (rather than those of the ED) and the LMOs of the ED, that is, $(\chi\chi|\rho)$, $(\chi\tilde{\psi}|\rho)$, and $(\tilde{\psi}\tilde{\psi}|\rho)$ type integrals. The latter two of these lists is easily obtained from the $(\chi\psi|\rho)$ ERIs kept from the MP2 step and the $(\tilde{\psi}\tilde{\psi}|\rho)$ ones that were used to evaluate the $(\bar{\psi}\bar{\psi}|\rho)$ integrals. The $(\chi\chi|\rho)$ AO ERIs, however, have to be again recomputed. The estimated FLOP cost of the evaluation of all AO ERIs in an average PCD (without screening) is $5.18 \cdot 10^{10}$ operations. The majority of the computational expenses of calculating the $(\bar{\phi}'\bar{\phi}'|\rho)$ integrals is attributed to the steps

$$T_{pb,Q} = \sum_q A_{qb}(\chi_p\chi_q|\rho_Q) \tag{4.3.1}$$

and

$$(\bar{\phi}'_a\bar{\phi}'_b|\rho_Q) \leftarrow \sum_p A_{pa}T_{pb,Q}\ , \tag{4.3.2}$$

where the p and q indices denote AOs that reside in the PCD, and A_{pa} is an expansion coefficient of $\bar{\phi}'_a$ in the AO basis. Assuming that the average number of AOs in the

Table 4.2: Average (maximum) domain sizes, orbital space dimensions, and detailed wall-clock times in minutes for LNO-CCSD(T) computations on large molecules using the default threshold set [2] for the construction of the domains.

Molecule	$\mathrm{TS}_{\mathrm{CC}}^{\mathrm{RS}}\cdots$pnp	$\mathrm{TS}_{\mathrm{CC}}^{\mathrm{RS}}\cdots$pnp	crambin	crambin	integrase
No. of atoms	90	90	644	644	2380
Basis	aug-cc-pVTZ	aug-cc-pVQZ	def2-TZVP	def2-QZVP	def2-TZVP
Total AOs	3155	5742	12075	28227	44035
Total AFs	7001	11362	29829	64015	108414
Total LMOs	123	123	909	909	3301
Atoms in ED	78 (90)	76 (90)	128 (270)	127 (270)	138 (314)
Atoms in PCD	48 (77)	45 (76)	46 (89)	46 (89)	47 (95)
AOs in ED	2769 (3155)	4904 (5742)	2615 (5694)	5747 (12310)	2712 (6359)
AFs in PCD	3669 (6163)	5946 (9626)	2398 (4917)	4945 (9910)	2368 (4902)
LMOs in ED	49 (88)	49 (88)	55 (117)	57 (117)	55 (102)
PAOs in ED	1678 (2645)	2891 (4659)	943 (1936)	2071 (4107)	938 (1950)
Occupied LNOs	29 (62)	29 (62)	28 (56)	29 (56)	28 (57)
Virtual LNOs	134 (273)	154 (300)	108 (191)	138 (231)	108 (212)
HF[a]+localization	384	1244	3906	16122	98220
LMP2+LIS const.	552	2309	802	5482	3158
Two-external trf.	799	3145	716	5594	2651
LNO-CCSD	1065	1401	2318	3934	7647
(T)	366	539	428	988	1436
Total LNO-CCSD(T)	2782	7395	4264	15988	14892

[a]Canonical DF-HF for $\mathrm{TS}_{\mathrm{CC}}^{\mathrm{RS}}\cdots$pnp, DF-HF with local fitting domains employed for the exchange part for crambin and integrase.

PCD is $2712/138 \cdot 47 \simeq 924$ the total operation count of Eqs. (4.3.1) and (4.3.2) is $4.36 \cdot 10^{11} + 5.10 \cdot 10^{10} = 4.87 \cdot 10^{11}$. Considering only the AO ERI evaluation and Eqs. (4.3.1) and (4.3.2) for a PCD, the former step accounts for about 10 % of the estimated FLOP cost. In practice the computation of the AO integrals can be screened more efficiently, and the calculation of these quantities only accounts for a few percent of the wall time referred to as "Two-external transformation" in Table 4.2, which means that it would probably not be the rate determining step even if the computation of the $(\chi\chi|\rho)$ ERIs was an order of magnitude slower. Nevertheless, the contribution of the AO integrals to the wall time would be much more noticeable for this step if that was the case.

4.4 Conclusions

The evaluation of three-center AO ERIs plays a crucial role in our density fitting HF and LMP2 implementations and is moderately important for the two-external integral transformation step of the CC algorithm. In the extreme example of the integrase enzyme it is clear that the (cubic-scaling) DF-HF algorithm demands the lion's share of the computational cost. Since the AO integral routines are called frequently (for each shell triplet), it is problematic to directly measure the wall time contribution of this step without significantly increasing the overall wall time due to the overhead of the measurement. A conservative estimate of 10% would still mean, however, that, if the ERI evaluation procedure would be an order of magnitude slower, the DF-HF algorithm would take twice as much time to converge. The same conclusion holds for the LMP2 calculation. In Table 4.2 the "LMP2+LIS construction" step includes the determination of the LNOs and the production of the $(\bar{\phi}\bar{\psi}|\rho)$ and $(\bar{\psi}\bar{\psi}|\rho)$ lists for each ED in addition to the MP2 computation, but this latter step requires almost all of the 3158 minutes for integrase. With the old AO integral algorithm the cost of this step would be comparable to the rate-determining LNO-CCSD computation. This is also true for the other triple-ζ quality calculations in Table 4.2. For the quadruple-ζ examples the LMP2 step accounts for a larger portion of the wall time of the complete LNO-CCSD(T) scheme. In these cases, because of the larger number of AOs and higher angular momentum integrals, the AO ERI evaluation is a more important step from the point of view of efficiency. With previous integral evaluation procedures the computation of AO ERIs would be without doubt the rate-determining operation for these calculations.

Summary

The dissertation presented an optimization procedure for three-center Coulomb-integrals and their geometrical first-order derivatives. In chapter 2 the most efficient paths for the evaluation of three-center ERIs over solid harmonic Gaussian functions of various angular momenta were determined. First, the adaptation of the well-established techniques developed for four-center ERIs, such as the Obara–Saika, McMurchie–Davidson, Gill–Head-Gordon–Pople, and Rys quadrature schemes, and the combinations thereof for three-center ERIs was discussed. Several algorithmic aspects, such as the order of the various operations and primitive loops as well as prescreening strategies were analyzed. Second, the number of FLOPs was estimated for the various algorithms derived, and based on these results the most promising ones were selected. The efficient implementation of the latter algorithms invoking automated programming techniques as well as the evaluation of their practical performance took place. It was concluded that the simplified Obara–Saika scheme of Ahlrichs is the most cost-effective one in the majority of cases, but the modified Gill–Head-Gordon–Pople and Rys algorithms proposed in Ref. 95 are preferred for particular shell triplets. The presented numerical experiments also show that even though the solid harmonic transformation and the horizontal recurrence require significantly fewer FLOPs if performed at the contracted level, this approach does not improve the efficiency in practical cases. Instead, it is more advantageous to carry out these operations at the primitive level, which allows for a more efficient memory layout.

In Chapter 3 the calculation of the geometrical derivatives of three-center ERIs over contracted solid harmonic Gaussians was optimized. Various methods were compared based on the Obara–Saika, McMurchie–Davidson, Gill–Head-Gordon–Pople, and Rys polynomial algorithms using Cartesian, Hermite, and mixed Gaussian integrals for each scheme. The latter ERIs contain both Hermite and Cartesian Gaussians, and they combine the advantageous properties of both types of basis functions. Furthermore, prescreening of the ERI derivatives was discussed, and an efficient approximation of the Cauchy–Schwarz bound for first derivatives was presented. Based on the estimated operation counts the most promising schemes were implemented by automated code generation, and their relative performances were evaluated. The benefits of computing all of the

derivatives of a shell triplet simultaneously compared to calculating them just for one degree of freedom at a time were analyzed, and it was found that the former scheme offers a speedup close to an order of magnitude with a triple-zeta quality basis when appropriate prescreening is applied. In these cases the Obara–Saika method with Cartesian Gaussians proved to be the best approach, but when derivatives for one degree of freedom are required at a time the mixed Gaussian Obara–Saika and Gill–Head-Gordon–Pople algorithms are predicted to be the best performing ones.

Finally, in Chapter 4, applications of the optimized integral code were presented on the example of DF-HF, LMP2, and LNO-CCSD(T) calculations for various test systems. The role of the three-center ERI evaluation in the various computational steps was discussed, and it was concluded that the computations would be significantly less feasible with previous integral evaluation procedures, especially when a quadruple-ζ basis set is applied.

Appendix: Determining the index ranges for the recurrence relations

Here we will demonstrate on the example of Eqs. (2.1.3), (3.1.4), and (3.1.2) how the index ranges for the starting values of the recursive equations in Chapters 2 and 3 were obtained. The basic idea is to consider a hypothetical recursion with only one term, determine which would be the starting ERI class if the recursion only contained this term, and then investigate how including the other terms one by one widens the required index range. Here the method will be applied to the $(\boldsymbol{dd}|[\partial \boldsymbol{d}/\partial C_x])$ class of derivative integrals, for which we need the ERI classes $(\boldsymbol{dd}|\boldsymbol{p})$ and $(\boldsymbol{dd}|\boldsymbol{f})$. These are gained by the use of the Cartesian HRR, Eq. (2.1.3). An intermediate ERI can take two kinds of roles (positions) in this recursion, as it is shown in Fig. A.1. The first position denoted with a circled 1 on the figures will be referred to as p1 in the text, and similarly for the other positions. If the HRR only contained p1, the starting class for the buildup of $(\boldsymbol{dd}|\boldsymbol{f})$ would be the $(\boldsymbol{gs}|\boldsymbol{f})$ one. This means that this class occupies only p1 in the process, let us call this route 11. Also considering p2, the possible routes are now 11, 12, and 22, which means that we also need $(\boldsymbol{fs}|\boldsymbol{f})$ and $(\boldsymbol{ds}|\boldsymbol{f})$. Since p1 increases l_b and decreases l_a, and p2 only increases l_b, the starting range for a general target class is $L_a \leq l_a \leq L_a + L_b$ for $(\boldsymbol{l}_a\boldsymbol{s}|\boldsymbol{L}_c)$ ERIs. If L_c denotes the angular momentum on the function to be differentiated, the $(\boldsymbol{l}_a\boldsymbol{s}|\boldsymbol{l}_c)$, $l_c = L_c - 1, L_c + 1$ classes are required.

$$\begin{tikzcd}[column sep=small]
 & (\boldsymbol{dd}|\boldsymbol{f}) \arrow[dl] \arrow[dr] & \\
(\boldsymbol{fp}|\boldsymbol{f}) & & (\boldsymbol{dp}|\boldsymbol{f}) \\
\textcircled{1} & & \textcircled{2} \\
\vdots & & \vdots
\end{tikzcd}$$

FIGURE A.1: The two possible positions that an intermediate ERI used by Eq. (2.1.3) can take shown on the example of the $(\boldsymbol{dd}|\boldsymbol{f})$ class

$$\begin{tikzcd}[column sep=small]
 & & (\boldsymbol{ds}|\boldsymbol{f})^{(0)} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] & \\
(\boldsymbol{ds}|\boldsymbol{d})^{(1)} & (\boldsymbol{ps}|\boldsymbol{d})^{(1)} & (\boldsymbol{ds}|\boldsymbol{p})^{(0)} & (\boldsymbol{ds}|\boldsymbol{p})^{(1)} \\
\textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

FIGURE A.2: The four possible positions that an intermediate ERI used by Eq. (3.1.4) can take shown on the example of the $(\boldsymbol{ds}|\boldsymbol{f})$ class

The situation is more complicated with Eq. (3.1.4), although it is enough to assess the required starting values for the $\boldsymbol{l}_c = \boldsymbol{f}$ classes for our example, since the construction of these requires the $\boldsymbol{l}_c = \boldsymbol{p}$ ERIs. As it can be seen in Fig. A.2 for the $(\boldsymbol{ds}|\boldsymbol{f})$ class, an intermediate can take four kinds of positions in each recursion step. For example, in route 111, $(\boldsymbol{ds}|\boldsymbol{s})^{(3)}$ takes p1 to get $(\boldsymbol{ds}|\boldsymbol{p})^{(2)}$, then this takes p1 again to construct $(\boldsymbol{ds}|\boldsymbol{d})^{(1)}$, and p1 is taken one more time to get the target class. If our recursion contained only this one term, the required starting integrals would be the $(\boldsymbol{ds}|\boldsymbol{s})^{(3)}$ ones for the $(\boldsymbol{ds}|\boldsymbol{f})$ class, and $(\boldsymbol{L}_a\boldsymbol{s}|\boldsymbol{s})^{(L_c)}$ for the general $(\boldsymbol{L}_a\boldsymbol{s}|\boldsymbol{L}_c)$ case. If we include p2 into our hypothetical recursion, we require a range of classes to start, since we need $(\boldsymbol{ps}|\boldsymbol{s})^{(3)}$ for route 112 and $(\boldsymbol{ss}|\boldsymbol{s})^{(3)}$ for route 122. Generally, p2 can be occupied $\min(l_a, l_c)$ times, which makes the index range $L_a - L_c \le l_a \le L_a$ with $n = L_c$. Note that this case is equivalent to Eq. (1.7.8). For our example $(\boldsymbol{ds}|\boldsymbol{f})$ class p4 can be occupied once for the routes 14 and 24, which need the $(\boldsymbol{ds}|\boldsymbol{s})^{(2)}$ and $(\boldsymbol{ps}|\boldsymbol{s})^{(2)}$ starting ERIs, respectively. p4 reduces n by one and increases l_c by two. With respect to the index range of the would-be recursion with routes containing p1 and p2 only, the index range is modified to $n = L_c - m$ and $L_a - L_c + 2m \le l_a \le L_a$, where m is the number of occurrences of p4, which is $\lceil L_c/2 \rfloor$ at maximum, making the ranges $L_c - \lceil L_c/2 \rfloor \le n \le L_c$ and $L_a + L_c - 2n \le l_a \le L_a$ for the general case. This is in fact the starting index range for Eq. (3.1.7), from which the p3 term is absent. For Eq. (3.1.4), we have to investigate the effect of this last term on the index ranges. Remaining at the $(\boldsymbol{ds}|\boldsymbol{f})$ class, p3 can appear in routes 13 and 23 starting from $(\boldsymbol{ds}|\boldsymbol{s})^{(1)}$ and $(\boldsymbol{ps}|\boldsymbol{s})^{(1)}$, respectively. If for the buildup of a general $(\boldsymbol{L}_a\boldsymbol{s}|\boldsymbol{L}_c)$ ERI class p3 is taken once, then the remaining integral, that is, $(\boldsymbol{L}_a\boldsymbol{s}|\boldsymbol{L}_c - \boldsymbol{2}_\sigma)$, can be constructed from $L_c - \lceil L_c/2 \rfloor - 1 \le n \le L_c - 2$, $L_a + L_c - 2n - 2 \le l_a \le L_a$ ERIs from the discussion above. This means that if we only allowed routes where p3 appears no more than once, then compared to the case when p3 is not allowed we would have

- unchanged number of needed ERIs with $n = L_c, L_c - 1$

$$\begin{tikzcd}
& (\boldsymbol{gs}|\boldsymbol{s})^{(0)} \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
(\boldsymbol{fs}|\boldsymbol{s})^{(0)} & (\boldsymbol{fs}|\boldsymbol{s})^{(1)} & (\boldsymbol{ds}|\boldsymbol{s})^{(0)} & (\boldsymbol{ds}|\boldsymbol{s})^{(1)} \\
\textcircled{1} & \textcircled{2} & \textcircled{3} & \textcircled{4} \\
\vdots & \vdots & \vdots & \vdots
\end{tikzcd}$$

FIGURE A.3: The four possible positions that an intermediate ERI used by Eq. (2.1.2) can take shown on the example of the $(\boldsymbol{gs}|\boldsymbol{s})$ class

- new required ERIs with $n = L_c - \lceil L_c/2 \rfloor - 1$ and $L_a - \mathrm{mod}(L_c, 2) \leq l_a \leq L_a$, that is, classes with one time smaller n and the same l_a range with respect to the lowest n classes when we do not allow p3
- for the n values appearing in both cases (routes not containing p3 and containing it once), the lower limit of the l_a range reduces by 2. Note that the l_a range becomes the same for $n = L_c - 1, L_c - 2$ since this range for the $n = L_c - 1$ integrals is unchanged.

So in this case, with $n = L_c$ the l_a range is $L_a - L_c \leq l_a \leq L_a$, for $n = L_c - 1, L_c - 2$ it is $L_a - L_c + 2 \leq l_a \leq L_a$, for $n = L_c - 3$ it is $L_a - L_c + 4 \leq l_a \leq L_a$, and with every decrement of n from this point the lower limit increases by two. Allowing p3 to be occupied up to two times, the l_a range for $n = L_c - 4$ increases by two, becoming the same as for the $n = L_c - 3$ classes. From this thought process we can see that if we allow p3 $\lceil L_c/2 \rfloor$ times, the overall range for n becomes $\mathrm{mod}(L_c, 2) \leq n \leq L_c$, with the l_a range being $L_a - L_c \leq l_a \leq L_a$ for the $n = L_c$ classes, $L_a - L_c + 2 \leq l_a \leq L_a$ for $n = L_c - 1$, and from this point every second decrement of n increases the lower limit by two. This behavior is expressed algebraically as $L_a - \mathrm{mod}[L_c, 2] - 2\lceil (n - \mathrm{mod}[L_c, 2])/2 \rfloor \leq l_a \leq L_a$. To meet the starting requirements of Eq. (2.1.3), the upper limit has to be extended to $L_a + L_b$, while if L_c is the angular momentum of the differentiated function the upper range of n becomes $L_c + 1$, hence the ranges in Sect. 3.1.1.

Eq. (2.1.2) is also a four-term recursion (see Fig. A.3), for which $(\boldsymbol{ss}|\boldsymbol{s})^{(n)}$ type ERIs are needed to start. With only p1, the starting ERI would be the $(\boldsymbol{ss}|\boldsymbol{s})$ one for the example $(\boldsymbol{gs}|\boldsymbol{s})$ class, and also for the general case. p2 reduces n by one, so including this term makes the n range $0 \leq n \leq 4$. Since p3 and p4 do not decrease n more than they increase l_a, they do not widen this index range. This means that for an arbitrary $(\boldsymbol{L}_a\boldsymbol{s}|\boldsymbol{s})^{(m)}$ class the required range is $m \leq n \leq m + L_a$. For Eq. (3.1.4) $(\boldsymbol{l}_a\boldsymbol{s}|\boldsymbol{s})^{(n)}$ ERIs are needed for $\mathrm{mod}(L_c, 2) \leq n \leq L_c$, so the required initial integrals have the n range of $\mathrm{mod}(L_c, 2) \leq n \leq L_a + L_b + L_c$.

Bibliography

[1] P. R. Nagy, G. Samu and M. Kállay, J. Chem. Theory Comput. **12**, 4897 (2016).

[2] P. R. Nagy, G. Samu and M. Kállay, J. Chem. Theory Comput. **14**, 4193 (2018).

[3] D. Mester, P. R. Nagy and M. Kállay, J. Chem. Phys. **146**, 194102 (2017).

[4] D. Mester, P. R. Nagy and M. Kállay, J. Chem. Phys. **148**, 094111 (2018).

[5] T. Janowski and P. Pulay, J. Am. Chem. Soc. **134**, 17520 (2012).

[6] K. M. Langner, T. Janowski, R. W. Góra, P. Dziekoǹski, W. A. Sokalski and P. Pulay, J. Chem. Theory Comput. **7**, 2600 (2011).

[7] M. Pitoňák, T. Janowski, P. Neogrády, P. Pulay and P. Hobza, J. Chem. Theory Comput. **5**, 1761 (2009).

[8] T. Janowski, P. Pulay, A. S. Karunarathna, A. Sygula and S. Saebø, Chem. Phys. Lett. **512**, 155 (2011).

[9] T. Janowski and P. Pulay, Theor. Chim. Acta. **130**, 419 (2011).

[10] R. Sedlak, T. Janowski, M. Pitoňák, J. Řezáč, P. Pulay and P. Hobza, J. Chem. Theory Comput. **9**, 3364 (2013).

[11] J. Almlöf, K. Fægri Jr. and K. Korsell, J. Comput. Chem. **3**, 385 (1982).

[12] M. Häser and R. Ahlrichs, J. Comput. Chem. **10**, 104 (1989).

[13] F. Weigend, Phys. Chem. Chem. Phys. **4**, 4285 (2002).

[14] S. F. Boys and I. Shavitt. University of Wisconsin Naval Research Laboratory Report WIS-AF-13, 1959.

[15] J. L. Whitten, J. Chem. Phys. **58**, 4496 (1973).

[16] R. Polly, H.-J. Werner, F. R. Manby and P. J. Knowles, Mol. Phys. **102**, 2311 (2004).

[17] C. Köppl and H.-J. Werner, J. Chem. Theory Comput. **12**, 3122 (2016).

[18] D. Mejía-Rodríguez and A. M. Köster, J. Chem. Phys. **141**, 124114 (2014).

[19] A. E. DePrince and C. D. Sherrill, J. Chem. Theory Comput. **9**, 2687 (2013).

[20] F. Neese and G. Olbrich, Chem. Phys. Lett. **362**, 170 (2002).

[21] U. Bozkaya and C. D. Sherrill, J. Chem. Phys. **144**, 174103 (2016).

[22] H.-J. Werner and M. Schütz, J. Chem. Phys. **135**, 144116 (2011).

[23] M. Schütz and F. R. Manby, Phys. Chem. Chem. Phys. **5**, 3349 (2003).

[24] H.-J. Werner and F. R. Manby, J. Chem. Phys. **124**, 054114 (2006).

[25] F. R. Manby, J. Chem. Phys. **119**, 4607 (2003).

[26] H.-J. Werner, F. R. Manby and P. J. Knowles, J. Chem. Phys. **118**, 8149 (2003).

[27] T. B. Adler, H.-J. Werner and F. R. Manby, J. Chem. Phys. **130**, 054106 (2009).

[28] M. Schütz, H.-J. Werner, R. Lindh and F. R. Manby, J. Chem. Phys. **121**, 737 (2004).

[29] S. Höfener and W. Klopper, Mol. Phys. **108**, 1783 (2010).

[30] S. Loibl and M. Schütz, J. Chem. Phys. **137**, 084107 (2012).

[31] L. E. McMurchie and E. R. Davidson, J. Comp. Phys. **26**, 218 (1978).

[32] P. M. W. Gill, M. Head-Gordon and J. A. Pople, Int. J. Quantum Chem. **36**, 269 (1989).

[33] S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).

[34] H. F. King and M. Dupuis, J. Comput. Phys. **21**, 144 (1976).

[35] T. Helgaker, P. Jørgensen and J. Olsen. *Molecular Electronic Structure Theory.* Wiley, Chichester, 2000.

[36] S. F. Boys, Proc. R. Soc. London Ser. A. **200**, 542 (1950).

[37] J. A. Pople and W. J. Hehre, J. Comp. Phys. **27**, 161 (1978).

[38] M. Hayami, J. Seino and H. Nakai, J. Chem. Phys. **142**, 204110 (2015).

[39] M. Hayami, J. Seino and H. Nakai, Int. J. Quantum Chem. **118**, e25640 (2018).

[40] C. A. White and M. Head-Gordon, J. Chem. Phys. **104**, 2620 (1996).

[41] Y. Shao and M. Head-Gordon, Chem. Phys. Lett. **323**, 425 (2000).

[42] A. M. Köster, J. Chem. Phys. **118**, 9943 (2003).

[43] M. Sierka, A. Hogekamp and R. Ahlrichs, J. Chem. Phys. **118**, 9136 (2003).

[44] A. Alvarez-Ibarra and A. M. Köster, J. Chem. Phys. **139**, 024102 (2013).

[45] A. Alvarez-Ibarra and A. M. Köster, Mol. Phys. **113**, 3128 (2015).

[46] S. Reine, E. Tellgren and T. Helgaker, Phys. Chem. Chem. Phys. **9**, 4771 (2007).

[47] A. Banerjee, J. O. Jensen and J. Simons, J. Chem. Phys. **82**, 4566 (1985).

[48] A. Komornicki, K. Ishida, K. Morokuma, R. Ditchfield and M. Conrad, Chem. Phys. Lett. **4**, 595 (1977).

[49] M. Dupuis and H. King, J. Chem. Phys. **68**, 3998 (1978).

[50] L. R. Kahn, J. Chem. Phys. **75**, 3962 (1981).

[51] M. A. Vincent, P. Saxe and H. F. Schaefer III, Chem. Phys. Lett. **94**, 351 (1983).

[52] M. A. Vincent and H. F. Schaefer III, Theor. Chim. Acta. **64**, 21 (1983).

[53] M. Page, P. Saxe, G. F. Adams and B. H. Lengsfield, Chem. Phys. Lett. **104**, 587 (1984).

[54] M. Head-Gordon and J. A. Pople, J. Chem. Phys. **89**, 5777 (1988).

[55] T. Helgaker and P. R. Taylor, Theor. Chim. Acta. **83**, 177 (1992).

[56] P. M. W. Gill, M. Head-Gordon and J. A. Pople, J. Phys. Chem. **94**, 5564 (1990).

[57] H. Honda, T. Yamaki and S. Obara, J. Chem. Phys. **117**, 1457 (2002).

[58] H. B. Schlegel, J. Chem. Phys. **77**, 3676 (1982).

[59] T. P. Hamilton and H. F. Schaefer III, Chem. Phys. **150**, 163 (1991).

[60] R. Lindh, U. Ryu and B. Liu, J. Chem. Phys. **95**, 5889 (1991).

[61] K. Ishida, J. Chem. Phys. **98**, 2176 (1993).

[62] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes in Fortran 77*. Fortran Numerical Recipes. Cambridge University Press, 1986.

[63] H. F. King, J. Phys. Chem. A. **120**, 9348 (2016).

[64] N. Flocke and V. Lotrich, J. Comput. Chem. **29**, 2722 (2008).

[65] N. Flocke, J. Chem. Phys. **131**, 064107 (2009).

[66] J. C. Wheeler, Rocky Mountain J. Math. **4**, 287 (1974).

[67] G. H. Golub and J. H. Welsch, Math. Comput. **23**, 221 (1969).

[68] J. Rys, M. Dupuis and H. F. King, J. Comput. Chem. **4**, 154 (1983).

[69] H. B. Schlegel, J. S. Binkley and J. A. Pople, J. Chem. Phys. **80**, 1976 (1984).

[70] R. E. Christoffersen, Adv. Quantum Chem. **6**, 333 (1972).

[71] E. Clementi, Proc. Natl. Acad. Sci. USA. **69**, 2942 (1972).

[72] V. Dyczmons, Theor. Chim. Acta. **28**, 307 (1973).

[73] S. Reine, T. Helgaker and R. Lindh, WIREs Comput. Mol. Sci. **2**, 290 (2012).

[74] R. Ahlrichs, Theor. Chim. Acta. **33**, 157 (1974).

[75] D. Cremer and J. Gauss, J. Comput. Chem. **7**, 279 (1986).

[76] C. C. J. Roothaan, Rev. Mod. Phys. **23**, 69 (1951).

[77] P. M. W. Gill, B. G. Johnson and J. A. Pople, Chem. Phys. Lett. **217**, 65 (1994).

[78] S. R. Gadre and R. K. Pathak, Proc. Indian Acad. Sci. (Chem. Sci.). **100**, 483 (1988).

[79] D. S. Lambrecht and C. Ochsenfeld, J. Chem. Phys. **123**, 184101 (2005).

[80] D. S. Lambrecht, B. Doser and C. Ochsenfeld, J. Chem. Phys. **123**, 184102 (2005).

[81] B. Doser, D. S. Lambrecht and C. Ochsenfeld, Phys. Chem. Chem. Phys. **10**, 3335 (2008).

[82] S. A. Maurer, D. S. Lambrecht, D. Flaig and C. Ochsenfeld, J. Chem. Phys. **136**, 144107 (2012).

[83] S. A. Maurer, D. S. Lambrecht, J. Kussmann and C. Ochsenfeld, J. Chem. Phys. **138**, 014101 (2013).

[84] H. Horn, H. Weiss, M. Häser, M. Ehrig and R. Ahlrichs, J. Comput. Chem. **12**, 1058 (1991).

[85] B. I. Dunlap, J. W. D. Connolly and J. R. Sabin, J. Chem. Phys. **71**, 3396 (1979).

[86] E. J. Baerends, D. E. Ellis and P. Ros, Chem. Phys. **2**, 41 (1973).

[87] Y. Jung, A. Sodt, P. M. W. Gill and M. Head-Gordon, Proc. Natl. Acad. Sci. U.S.A. **102**, 6692 (2005).

[88] F. Weigend, A. Köhn and C. Hättig, J. Chem. Phys. **116**, 3175 (2002).

[89] M. E. Fornace, J. Lee, K. Miyamoto, F. R. Manby and T. F. Miller III, J. Chem. Theory Comput. **11**, 568 (2015).

[90] F. Weigend, M. Häser, H. Patzelt and R. Ahlrichs, Chem. Phys. Lett. **294**, 143 (1998).

[91] F. Weigend, J. Comput. Chem. **29**, 167 (2008).

[92] A. Hellweg and D. Rappoport, Phys. Chem. Chem. Phys. **17**, 1010 (2015).

[93] R. Ahlrichs, Phys. Chem. Chem. Phys. **6**, 5119 (2004).

[94] The representative access times and capacities were imported from the lecture notes of the Massachusetts Institute of Technology course Computation Structures (http://computationstructures.org/lectures/caches/caches.html#20).

[95] G. Samu and M. Kállay, J. Chem. Phys. **146**, 204101 (2017).

[96] T. H. Dunning Jr., J. Chem. Phys. **90**, 1007 (1989).

[97] D. E. Woon and T. H. Dunning Jr., J. Chem. Phys. **98**, 1358 (1993).

[98] B. G. Johnson, P. M. W. Gill and J. A. Pople, Int. J. Quantum Chem. **40**, 809 (1991).

[99] U. Ryu, Y. S. Lee, and R. Lindh, Chem. Phys. Lett. **185**, 562 (1991).

[100] D. S. Hollman, H. F. Schaefer III and E. F. Valeev, J. Chem. Phys. **142**, 154106 (2015).

[101] https://aip.scitation.org/doi/suppl/10.1063/1.4983393.

[102] Mrcc, a quantum chemical program suite written by M. Kállay, Z. Rolik, J. Csontos, P. Nagy, G. Samu, D. Mester, J. Csóka, B. Szabó, I. Ladjánszki, L. Szegedy, B. Ladóczki, K. Petrov, M. Farkas, P. D. Mezei, and B. Hégely. See http://www.mrcc.hu/.

[103] F. Neese, A. Hansen and D. G. Liakos, J. Chem. Phys. **131**, 064103 (2009).

[104] T. Helgaker, J. Gauss, P. Jørgensen and J. Olsen, J. Chem. Phys. **106**, 6430 (1997).

[105] J. Weidendorfer, M. Kowarschik and C. Trinitis. in *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, Krakow, Poland, 2004.

[106] B. Doser, D. S. Lambrecht, J. Kussmann and C. Ochsenfeld, J. Chem. Phys. **130**, 064107 (2009).

[107] M. Schütz, G. Hetzer and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999).

[108] H. Eshuis, J. Yarkony and F. Furche, J. Chem. Phys. **132**, 234114 (2010).

[109] J. Hári, P. Polyák, D. Mester, M. Mitušík, M. Omastová, M. Kállay and B. Pukánszky, Appl. Clay Sci. **132**, 167 (2016).

[110] Z. Rolik, L. Szegedy, I. Ladjánszki, B. Ladóczki and M. Kállay, J. Chem. Phys. **139**, 094105 (2013).

[111] P. Calaminici, V. D. Domínguez-Soria, G. Geudtner, E. Hernández-Marín and A. M. Köster, Theor. Chem. Acc. **115**, 221 (2006).

[112] G. Samu and M. Kállay, J. Chem. Phys. **149**, 124101 (2018).

[113] https://aip.scitation.org/doi/suppl/10.1063/1.5049529.

[114] F. Weigend and R. Ahlrichs, Phys. Chem. Chem. Phys. **7**, 3297 (2005).

[115] P. Pulay, Chem. Phys. Lett. **100**, 151 (1983).

Declaration

I, Gyula Samu hereby declare that this PhD dissertation is the result of my own work. All content that was taken from other sources –either literally or reworded– is clearly indicated, and its source is presented in the reference list.

Budapest, February 19, 2019 Gyula Samu