Efficient evaluation of three-center Coulomb integrals

Gyula Samu and Mihály Kállay

Citation: The Journal of Chemical Physics **146**, 204101 (2017); doi: 10.1063/1.4983393
View online: http://dx.doi.org/10.1063/1.4983393
View Table of Contents: http://aip.scitation.org/toc/jcp/146/20
Published by the American Institute of Physics

Efficient evaluation of three-center Coulomb integrals

Gyula Samu[a] and Mihály Kállay[b]
MTA-BME Lendület Quantum Chemistry Research Group, Department of Physical Chemistry and Materials Science, Budapest University of Technology and Economics, P.O. Box 91, H-1521 Budapest, Hungary

(Received 20 February 2017; accepted 28 April 2017; published online 22 May 2017)

In this study we pursue the most efficient paths for the evaluation of three-center electron repulsion integrals (ERIs) over solid harmonic Gaussian functions of various angular momenta. 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 is discussed. Several algorithmic aspects, such as the order of the various operations and primitive loops as well as prescreening strategies, are analyzed. Second, the number of floating point operations (FLOPs) is estimated for the various algorithms derived, and based on these results the most promising ones are selected. We report the efficient implementation of the latter algorithms invoking automated programming techniques and also evaluate their practical performance. We conclude 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 herein are preferred for particular shell triplets. Our 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 more efficient integral prescreening and memory layout. *Published by AIP Publishing.* [http://dx.doi.org/10.1063/1.4983393]

I. INTRODUCTION

Electron repulsion integrals (ERIs), which describe the Coulomb interaction of two charge distributions, are one of the basic quantities in quantum chemistry. In conventional formulations, these are four-center integrals defined as

$$(\phi_A\phi_B|\phi_C\phi_D) = \int\int \frac{\phi_A(\mathbf{r_1})\phi_B(\mathbf{r_1})\phi_C(\mathbf{r_2})\phi_D(\mathbf{r_2})}{|\mathbf{r_1}-\mathbf{r_2}|}\mathrm{d}\mathbf{r_1}\,\mathrm{d}\mathbf{r_2} \tag{1}$$

for basis functions ϕ_A, ϕ_B, ϕ_C, and ϕ_D with $\mathbf{r_1}$ and $\mathbf{r_2}$ being the coordinates of the electrons. The evaluation of such integrals is often the limiting step for Hartree–Fock (HF) and density functional theory (DFT) calculations, while their transformation from the atomic orbital (AO) to the molecular orbital (MO) basis can be a bottleneck for correlated methods. The computational requirements for both of these tasks can be efficiently reduced by invoking the density fitting (DF) approximation, which is equivalent to the resolution of identity technique if the so-called Coulomb metric is used.[1–5] In this approach, the generalized electron densities given by the product of two basis functions are expanded in an auxiliary (fitting) basis in a manner that minimizes the error of the electric field generated by the charge distributions[3,4] as

$$(\phi_A\phi_B|\phi_C\phi_D) \approx \sum_{Q,R}(\phi_A\phi_B|\rho_Q)V_{QR}^{-1}(\rho_R|\phi_C\phi_D), \tag{2}$$

where ρ_Q and ρ_R denote functions from the fitting basis, V_{QR}^{-1} is the element of the inverse of the matrix containing the two-center ERIs $(\rho_Q|\rho_R)$, and $(\phi_A\phi_B|\rho_Q)$ is a three-center Coulomb integral. The main advantage of applying this approximation is that the $\mathcal{O}(N^4)$ scaling of evaluating and processing the ERIs breaks down to $\mathcal{O}(N^2M)$ with N (M) being the size of the AO (auxiliary) basis, and the calculation of these integrals over a reduced number of Gaussian basis functions is also considerably simpler than that for the four-center ones. When dealing with large systems, even the necessary three-center ERIs become too numerous to store on a disk or it is more advantageous to recalculate them since the sparsity of the integrals can be efficiently utilized with prescreening techniques. These observations had led to the development of integral-direct algorithms, where the ERIs are recalculated whenever they are needed, e.g., in each cycle of a direct self-consistent field (SCF) procedure[6–8] or for the overlapping domains in a local correlation calculation.[9] The efficiency of such algorithms obviously depends on the speed of the integral evaluation.

For the evaluation of four-center ERIs, several efficient schemes have been constructed. The oldest of the still popular methods is the one developed by King, Dupuis, and Rys,[10–16] commonly referred to as the Rys quadrature scheme, which is a Gaussian quadrature based technique for the evaluation of integrals containing functions with arbitrary angular momenta. Other methods are mainly based on recurrence relations using scaled Boys functions[17] as their starting values. The scheme of McMurchie and Davidson[18] (MD) utilizes the fact that Cartesian Gaussian overlap distributions can be

[a]Electronic mail: gysamu@mail.bme.hu
[b]Electronic mail: kallay@mail.bme.hu

written in terms of Hermite Gaussian functions and also that the two-center Hermite integrals necessary for this expansion can be reduced to one-center ones. Later, Obara and Saika[19,20] (OS) presented their method based on recurrence relations connecting auxiliary integrals of various angular momenta. Their scheme arguably remains the most widely used one, due to the subsequent introduction of the horizontal recurrence relation[21–23] (HRR) by Head-Gordon and Pople and the electron transfer relation[24] (ETR) by Hamilton and Schaefer. The latter recurrence was also presented by Lindh, Riu, and Liu utilizing the close relationship between the OS and the Rys quadrature schemes, and these authors also developed the reduced multiplication scheme by combining the Rys quadrature approach with the ETR and the HRR.[25,26] Gill and co-workers,[27–34] amongst other contributions, achieved a synthesis of the OS and the MD methods by moving the transformation of Hermite integrals into integrals over Cartesian overlap distributions to the contracted level by properly scaling the intermediate one-center integrals, resulting in a scheme that is very efficient for integrals over highly contracted functions.

Concerning the evaluation of three-center integrals, fewer studies can be found in the literature. Köster, exploiting the uncontracted nature of the auxiliary basis sets, combined the OS, MD, and Gill–Head-Gordon–Pople (GHP) algorithms for three-center ERIs over Cartesian Gaussians.[35] Later he also proposed the use of Hermite Gaussian auxiliary functions,[36,37] which saves the transformation from Hermite to Cartesian functions in an MD scheme. Reine, Tellgren, and Helgaker[38,39] showed that Hermite Gaussians transform into solid harmonic ones exactly the same way as Cartesian Gaussians do, and utilizing this finding these authors also put forward a scheme for the evaluation of three-center integrals over solid harmonic Gaussians which avoids the Hermite to Cartesian transformation. A remarkable improvement on the OS scheme for solid harmonic three-center ERIs was achieved by Ahlrichs,[40] who realized that the recurrence relation for the build-up of angular momentum on the fitting function greatly simplifies for three-center integrals. Efficient three-center ERI implementations can be found in the Libint library of Valeev[41,42] and the adaptive integral core code of Knizia,[43] who both applied the results of Ahlrichs.

It is also important to mention here that there exist several approaches that employ the DF approximation but at least partly avoid the explicit construction of the three-center ERI lists. The so-called J-engine and the related schemes exploit the structure of the Coulomb term in a direct SCF calculation, and instead of performing the relatively expensive recursions and transformations for the ERIs the reverse operations are carried out for the quantities by which the ERIs are multiplied.[36,44,45] These algorithms are particularly useful for Kohn–Sham SCF calculations where the significantly more costly exact exchange term is not computed, but efficient DF HF and hybrid DFT algorithms can also be designed if the J-engine approaches are combined with low-cost schemes for the evaluation of the Fock exchange.[9,46–51] A further possibility for the reduction of the costs of DF SCF calculations is to approximate far-field ERIs invoking asymptotic or multipole expansions and to evaluate only the near-field integrals analytically.[52–54] Nonetheless, there are numerous applications where the explicit evaluation of the three-center ERIs cannot be avoided. For the evaluation of the Fock exchange in a DF SCF calculation or for any correlated calculation employing the DF approximation, at least one AO index of the three-center integrals must be transformed to the MO basis, and, to the best of our knowledge, there exist no algorithms that use similar tricks as the J-engine scheme. In the above cases, at least the near-field three-center integrals must be computed, which requires a considerable computation time, especially with basis sets including functions of high angular momentum. Thus, the cost-effective evaluation of three-center Coulomb integrals is of utmost importance for DF methods.

The aim of this paper is to find the most efficient route for the evaluation of three-center ERIs over solid harmonic Gaussian functions of various angular momenta. We compare the OS, MD, GHP, and Rys quadrature schemes and their combinations and discuss several algorithmic aspects for the evaluation of three-center ERIs. In Sec. II the adaptation of the aforementioned methods for the evaluation of three-center ERIs is presented. The given equations form the basis for the estimation of the floating point operations (FLOPs) required by the various approaches, detailed in Sec. III. The implementation of the schemes with the lowest theoretical FLOP counts along with various prescreening strategies and orders of the operations is discussed in Sec. IV, and the comparison of practical performances is done in Sec. V. Finally, in Sec. VI the efficiency of our implementation is demonstrated by calculating the ERIs for medium to large systems.

II. THEORY

A. Three-center Coulomb integrals

In this work we are concerned with the evaluation of three-center ERIs over contracted solid harmonic Gaussian basis functions, which are gained by linear transformations of integrals over unnormalized primitive Cartesian Gaussian basis functions. These functions are defined as

$$G_{IJK}(\mathbf{r}, a, \mathbf{A}) = x_{\mathrm{A}}^{I} y_{\mathrm{A}}^{J} z_{\mathrm{A}}^{K} \exp(-a r_{\mathrm{A}}^{2}) , \tag{3}$$

where $\mathbf{r}$ denotes the position vector of the electron, $\mathbf{A}$ is the position of the nucleus on which the function is centered, a is a constant Gaussian exponent, and r_{A} is the magnitude of the vector $\mathbf{r}_{\mathbf{A}} = \mathbf{r} - \mathbf{A}$ with x_{A} being the x component of $\mathbf{r}_{\mathbf{A}}$. $L = I + J + K$ will be called the angular momentum of G_{IJK}, and the vector $\boldsymbol{L} = (I, J, K)$ will be referred to as the angular momentum vector of G_{IJK}. 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, $G_I = x_{\mathrm{A}}^{I} \exp(-a x_{\mathrm{A}}^{2})$. They also obey the following recurrence relation for differentiation with respect to a nuclear coordinate (given here for the x direction only):

$$\frac{\partial G_I}{\partial A_x} = 2a G_{I+1} - I G_{I-1}, \tag{4}$$

where A_x is the x component of $\mathbf{A}$.

For solid harmonic Gaussian functions, one needs to combine functions with the same exponent, angular momentum, and center, but different angular momentum vectors as

$$G_{Lm}(\mathbf{r}, a, \mathbf{A}) = \sum_{I+J+K=L} C_{IJK}^{Lm} G_{IJK}(\mathbf{r}, a, \mathbf{A}). \tag{5}$$

A shell of solid harmonic Gaussians consists of functions with $0 \le |m| \le L$, having $2L+1$ components. The C_{IJK}^{Lm} coefficients in Eq. (5) only depend on the angular momentum vector and the value of L and m.[17]

We obtain contracted Gaussians by linearly combining functions with different exponents a but the same angular momentum vector and center,

$$\chi_{ALm}(\mathbf{r}, \mathbf{A}) = \sum_a G_{Lm}(\mathbf{r}, a, \mathbf{A}) d_{a\chi_A}, \tag{6}$$

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. Of course, the transformation given by Eq. (5) can also be applied to integrals in the contracted basis and the one defined by Eq. (6) to the integrals in the primitive Cartesian basis as well.

Three-center ERIs over primitive Gaussian functions are defined as

$$(\boldsymbol{L}_a \boldsymbol{L}_b | \boldsymbol{L}_c) = \int\int \frac{G_{I_aJ_aK_a}(\mathbf{r_1}, a, \mathbf{A}) G_{I_bJ_bK_b}(\mathbf{r_1}, b, \mathbf{B}) G_{I_cJ_cK_c}(\mathbf{r_2}, c, \mathbf{C})}{|\mathbf{r_1} - \mathbf{r_2}|} \mathrm{d}\mathbf{r_1}\, \mathrm{d}\mathbf{r_2}, \tag{7}$$

where $\boldsymbol{L}_a = (I_a, J_a, K_a)$ stands for the angular momentum vector and $L_a = I_a + J_a + K_a$ is the angular momentum of the function with exponent a. From these, integrals over solid harmonic contracted Gaussians are computed by applying Eqs. (5) and (6) in an arbitrary order on the three centers. We will refer to primitive integrals sharing angular momenta L_a, L_b, and L_c, centers **A**, **B**, and **C**, and exponents a, b, and c as a primitive class, e.g., the class (**11**|**1**) consists of 27 primitive integrals. Similarly, the members of contracted classes are integrals over contracted Gaussians of the same angular momenta and centers. A shell triplet will refer to all the integrals over solid harmonic Gaussians belonging to centers **A**, **B**, and **C** and angular momenta L_a, L_b, and L_c.

An important special case is the primitive integral where $\boldsymbol{L}_a = \boldsymbol{L}_b = \boldsymbol{L}_c = \mathbf{0}$, the value of which can be expressed directly[40] as

$$(\mathbf{00}|\mathbf{0})^{(0)} = (\mathbf{00}|\mathbf{0}) = \theta_{pc} \kappa_{ab} F_0(\alpha R_{\mathrm{PC}}^2), \tag{8}$$

with

$$\kappa_{ab} = \exp(-\mu R_{\mathrm{AB}}^2), \qquad \theta_{pc} = \frac{2\pi^{5/2}}{pc\sqrt{p+c}},$$
$$\mu = \frac{ab}{a+b}, \qquad \mathbf{P} = \frac{a\mathbf{A} + b\mathbf{B}}{p},$$
$$\mathbf{R}_{\mathrm{AB}} = \mathbf{A} - \mathbf{B}, \qquad \mathbf{R}_{\mathrm{PC}} = \mathbf{P} - \mathbf{C},$$
$$p = a + b, \qquad \alpha = \frac{pc}{p+c},$$

and F_n being the Boys function of order n, defined as

$$F_n(x) = \int_0^1 t^{2n} \exp(-xt^2) \mathrm{d}t. \tag{9}$$

The integral in Eq. (8) and also other auxiliary integrals where the order of the Boys function is greater than 0 are the starting points for the OS,[19] MD,[18] and GHP[27] schemes for the evaluation of the integrals in Eq. (7) for arbitrary angular momenta.

B. Obara–Saika recursion

The OS scheme utilizes recurrence relations of auxiliary intermediate integrals to construct the true ERIs with the desired angular momenta. An efficient application of this method to three-center ERIs was presented by Ahlrichs.[40] This approach will be referred to as OS1. The first step here is to evaluate the required auxiliary integrals

$$(\mathbf{00}|\mathbf{0})^{(n)} = \theta_{pc} \kappa_{ab} F_n(\alpha R_{\mathrm{PC}}^2) \tag{10}$$

for $L_c \le n \le L_a + L_b + L_c$. Then the vertical recurrence relation[19] (VRR) is used to increment the angular momentum of the first function on the bra side (given here for the x direction) as

$$([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} = X_{\mathrm{PA}} (\boldsymbol{l}_a \mathbf{0}|\mathbf{0})^{(n)} - \frac{\alpha}{p} X_{\mathrm{PC}} (\boldsymbol{l}_a \mathbf{0}|\mathbf{0})^{(n+1)} + \frac{i_a}{2p} \times \left(([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n)} - \frac{\alpha}{p} ([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\mathbf{0})^{(n+1)} \right), \tag{11}$$

where X_{PA} is the x component of vector $\mathbf{R}_{\mathrm{PA}}$, and generally $\mathbf{1}_\sigma = (\delta_{\sigma,x}, \delta_{\sigma,y}, \delta_{\sigma,z})$ for $\sigma = x, y, z$. Here and later, l_a, i_a, and $\boldsymbol{l}_a$ refer to the angular momentum, its x component, and the angular momentum vector of the first Gaussian in the intermediate integrals, respectively, and a similar notation will be used for the angular momenta of the second and third functions and their components. With Eq. (11), the classes $(\boldsymbol{l}_a \mathbf{0}|\mathbf{0})^{(L_c)}$ are calculated for $\max(1, L_a - L_c) \le l_a \le L_a + L_b$. Next, in the case where solid harmonic basis functions are supposed to be on the ket side, $\boldsymbol{l}_c$ can be built up by a two-term VRR,[40]

$$(\boldsymbol{l}_a \mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} = \frac{\alpha}{c} X_{\mathrm{PC}} (\boldsymbol{l}_a \mathbf{0}|\boldsymbol{l}_c)^{(n+1)} + \frac{i_a}{2(p+c)} ([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n+1)}. \tag{12}$$

Eq. (12) is used to produce $(\boldsymbol{l}_a \mathbf{0}|\boldsymbol{L}_c)^{(0)}$ classes for $L_a \le l_a \le L_a + L_b$. From here on, superscript (n) will be dropped when it is equal to 0. The last step is to increment $\boldsymbol{l}_b$, which is efficiently done by the HRR of Head-Gordon and Pople,[21]

$$(\boldsymbol{l}_a [\boldsymbol{l}_b + \mathbf{1}_x] | \boldsymbol{l}_c) = ([\boldsymbol{l}_a + \mathbf{1}_x] \boldsymbol{l}_b | \boldsymbol{l}_c) + X_{\mathrm{AB}} (\boldsymbol{l}_a \boldsymbol{l}_b | \boldsymbol{l}_c). \tag{13}$$

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. (10) as in OS1 and then applies the VRR to the ket side first as

$$(\mathbf{00}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} = \frac{\alpha}{c} X_{\mathrm{PC}} (\mathbf{00}|\boldsymbol{l}_c)^{(n+1)} \tag{14}$$

to construct the classes $(\mathbf{00}|\boldsymbol{l}_c)^{(n)}$ for $\max(1, L_c - L_a - L_b) \le l_c \le L_c$ and $L_c - l_c \le n \le L_a + L_b$. This is followed by building up the angular momentum of the first function on the bra side as

$$\begin{aligned}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n)} &= X_{\mathrm{PA}} (\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n)} - \frac{\alpha}{p} X_{\mathrm{PC}} (\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c)^{(n+1)} \\ &+ \frac{i_a}{2p} \Big(([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c)^{(n)} \\ &- \frac{\alpha}{p} ([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|l_c)^{(n+1)} \Big) \\ &+ \frac{i_c}{2(p+c)} (\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n+1)}, \end{aligned} \tag{15}$$

to compute $(\boldsymbol{l}_a\mathbf{0}|\boldsymbol{L}_c)$ for $L_a \le l_a \le L_a + L_b$, and finally the algorithm is finished with Eq. (13).

Apart from the VRR, another way to build up $\boldsymbol{l}_a$ or $\boldsymbol{l}_c$ is to use the ETR[24] arising from the translational invariance of integrals, and also Eqs. (4) and (13). For three-center ERIs, the ETR has the form

$$\begin{aligned}([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) &= -\frac{b}{p} X_{\mathrm{AB}} (\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c) + \frac{i_a}{2p} ([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \\ &- \frac{c}{p} (\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x]) + \frac{i_c}{2p} (\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c - \mathbf{1}_x]) \end{aligned} \tag{16}$$

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

$$\begin{aligned}(\boldsymbol{l}_a\mathbf{0}|[\boldsymbol{l}_c + \mathbf{1}_x]) &= -\frac{b}{c} X_{\mathrm{AB}} (\boldsymbol{l}_a\mathbf{0}|\boldsymbol{l}_c) + \frac{i_a}{2c} ([\boldsymbol{l}_a - \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \\ &- \frac{p}{c} ([\boldsymbol{l}_a + \mathbf{1}_x]\mathbf{0}|\boldsymbol{l}_c) \end{aligned} \tag{17}$$

for the $\boldsymbol{l}_a \to \boldsymbol{l}_c$ transfer. We note that, in principle, Eq. (17) 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[40] when transforming to the solid harmonic basis. This cancellation also takes place for the third and fourth terms in both Eqs. (11) and (15), and the second term in Eq. (16), but only in the case when $\boldsymbol{L}_b = \mathbf{0}$. It should be noted that the numerical instability in the ETR associated with the addition $pX_{\mathrm{PA}}/(c+d) + X_{\mathrm{QC}}$[55] (where, in the four-center case, d is the exponent of the fourth Gaussian and $\mathbf{Q} = (c\mathbf{C} + d\mathbf{D})/(c+d)$, $\mathbf{D}$ being the center of the fourth function) does not appear here. This is because in the absence of the fourth center, Eq. (13) 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. (13) with Eq. (16), we cannot use Eq. (14) for the construction of the $(\mathbf{00}|\boldsymbol{l}_c)$ type classes, instead we have to employ the full vertical recurrence,[40]

$$\begin{aligned}(\mathbf{00}|[\boldsymbol{l}_c + \mathbf{1}_x])^{(n)} &= \frac{\alpha}{c} X_{\mathrm{PC}} (\mathbf{00}|\boldsymbol{l}_c)^{(n+1)} + \frac{i_c}{2c} \Big((\mathbf{00}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n)} \\ &- \frac{\alpha}{c} (\mathbf{00}|[\boldsymbol{l}_c - \mathbf{1}_x])^{(n+1)} \Big), \end{aligned} \tag{18}$$

for the ket side. The terms corresponding to the ones in the big parentheses in Eq. (18) vanish in Eqs. (12) and (14) during the solid harmonic transformation[40] of the ket side; however, with Eq. (16) 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 we first employ Eq. (18) to build up $\boldsymbol{l}_c$ and then Eq. (16) for $\boldsymbol{l}_a$ will be referred to as OS3. In this route, we first use Eq. (10) 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. (18) for the classes $(\mathbf{00}|\boldsymbol{l}_c)$ with $\max(L_c - L_a - L_b, 1) \le l_c \le L_a + L_b + L_c$, thereafter we apply Eq. (16) 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. (11), and $\boldsymbol{l}_c$ is incremented by the ETR, Eq. (17). 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 $\max(L_a - L_c, 1) \le l_a \le L_a + L_b + L_c$.

C. McMurchie–Davidson scheme

The strategy of the MD method is to expand ERIs over Gaussian overlap distributions arising from multiplying $G_{I_aJ_aK_a}(\mathbf{r}_1, a, \mathbf{A})$ and $G_{I_bJ_bK_b}(\mathbf{r}_1, b, \mathbf{B})$ into integrals over Hermite Gaussian functions centered on $\mathbf{P}$, defined as

$$H_{\bar{I}_p\bar{J}_p\bar{K}_p}(\mathbf{r}_1, p, \mathbf{P}) = \frac{\partial^{\bar{L}_p} \exp(-pr_{\mathrm{P}}^2)}{\partial P_x^{\bar{I}_p} \partial P_y^{\bar{J}_p} \partial P_z^{\bar{K}_p}}, \tag{19}$$

where the bars over the total angular momentum and its components are used to distinguish from the corresponding Cartesian Gaussians. In this scheme, one has to evaluate two-center Coulomb integrals over Hermite Gaussians centered on $\mathbf{P}$ and $\mathbf{C}$, which, exploiting translational invariance (that is, $\partial/\partial P_x = -\partial/\partial C_x$), can be written as[18]

$$\begin{aligned}(\bar{\boldsymbol{l}}_p|\bar{\boldsymbol{l}}_c) &= \theta_{pc} (2c)^{-\bar{l}_c} \frac{\partial^{\bar{l}_p + \bar{l}_c} F_0(\alpha R_{\mathrm{PC}}^2)}{\partial P_x^{\bar{i}_p} \partial P_y^{\bar{j}_p} \partial P_z^{\bar{k}_p} \partial C_x^{\bar{i}_c} \partial C_y^{\bar{j}_c} \partial C_z^{\bar{k}_c}} \\ &= \theta_{pc} (-2c)^{-\bar{l}_c} \frac{\partial^{\bar{l}_p + \bar{l}_c} F_0(\alpha R_{\mathrm{PC}}^2)}{\partial P_x^{\bar{i}_p + \bar{i}_c} \partial P_y^{\bar{j}_p + \bar{j}_c} \partial P_z^{\bar{k}_p + \bar{k}_c}} = (\bar{\boldsymbol{l}}_u) \end{aligned} \tag{20}$$

with $\bar{\boldsymbol{l}}_u = \bar{\boldsymbol{l}}_p + \bar{\boldsymbol{l}}_c$. The scaling with $(2c)^{-\bar{l}_c}$ is applied since for the Hermite Gaussian in the ket we follow the definition of Reine and co-workers,[38] 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. (20) can be computed by the two-term recursion[18]

$$(\bar{\boldsymbol{l}}_u + \mathbf{1}_x)^{(n)} = X_{\mathrm{PC}} (\bar{\boldsymbol{l}}_u)^{(n+1)} + \bar{i}_u (\bar{\boldsymbol{l}}_u - \mathbf{1}_x)^{(n+1)} \tag{21}$$

with

$$(\bar{\mathbf{0}})^{(n)} = (-2\alpha)^n \kappa_{ab} \theta_{pc} (-2c)^{-\bar{L}_c} F_n(\alpha R_{\mathrm{PC}}^2). \tag{22}$$

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[18]

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\bar{\boldsymbol{L}}_c) = \sum_{\bar{i}_p=0}^{I_a+I_b} E_{\bar{i}_p}^{I_a,I_b} \sum_{\bar{j}_p=0}^{J_a+J_b} E_{\bar{j}_p}^{J_a,J_b} \sum_{\bar{k}_p=0}^{K_a+K_b} E_{\bar{k}_p}^{K_a,K_b} (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{L}}_c). \tag{23}$$

The E expansion coefficients appearing in Eq. (23) can be constructed by a set of recurrence relations,[17]

$$E_{\bar{0}}^{i_a+1,0} = X_{\mathrm{PA}} E_{\bar{0}}^{i_a,0} + E_{\bar{1}}^{i_a,0}, \tag{24}$$

$$E_{\bar{0}}^{i_a,i_b+1} = X_{\mathrm{PB}} E_{\bar{0}}^{i_a,i_b} + E_{\bar{1}}^{i_a,i_b}, \tag{25}$$

$$E_{\bar{i}_p+1}^{i_a,i_b} = \frac{1}{2p(\bar{i}_p+1)} (i_a E_{\bar{i}_p}^{i_a-1,i_b} + i_b E_{\bar{i}_p}^{i_a,i_b-1}),\ \bar{i}_p \geq 0, \tag{26}$$

with $E_{\bar{0}}^{0,0} = 1$.

The expansion defined by Eq. (23) can be applied to produce various types of three-center ERIs. In the MD1 algorithm, for example, we get the $(\boldsymbol{L}_a\boldsymbol{L}_b|\bar{\boldsymbol{L}}_c)$ 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 $\lceil \bar{L}_c/2 \rfloor + [\bar{L}_c \bmod 2] \leq n \leq L_a + L_b + \bar{L}_c$ with $\lceil x \rfloor$ denoting the integer part of x. The one-center integrals $(\bar{\boldsymbol{l}}_u)$ for $\bar{L}_c \leq \bar{l}_u \leq L_a + L_b + \bar{L}_c$ are built up by Eq. (21), from which the target integrals are readily assembled by Eq. (23). The work done in this assembly step can be reduced by performing it at an earlier stage to construct intermediate classes and using OS-type recursions for the evaluation of the target integrals. In the MD2 scheme, the $(\boldsymbol{l}_a\mathbf{0}|\bar{\boldsymbol{L}}_c)$ classes for $L_a \leq l_a \leq L_a + L_b$ are evaluated with Eq. (23). 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. (13). A third option (MD3) is to obtain the $(\boldsymbol{l}_a\mathbf{0}|\mathbf{0})^{(\bar{L}_c)}$ type intermediates for $\max(1, L_a - \bar{L}_c) \leq l_a \leq L_a + L_b$ with Eq. (23), then to build up $\bar{\boldsymbol{l}}_c$ with Eq. (12), and to finish with Eq. (13). Here the $(-1)^{\bar{L}_c}$ scaling factor is absent from Eq. (22), and the required $(\bar{\mathbf{0}})^{(n)}$ values are in the range of $\bar{L}_c \leq n \leq L_a + L_b + \bar{L}_c$ and used for calculating the $(\bar{\boldsymbol{l}}_u)^{(\bar{L}_c)}$ 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 for transforming the Hermite integrals into ones over Cartesian overlaps is the use of the

$$\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p} = x_{\mathrm{A}}^{i_a} y_{\mathrm{A}}^{j_a} z_{\mathrm{A}}^{k_a} x_{\mathrm{B}}^{i_b} y_{\mathrm{B}}^{j_b} z_{\mathrm{B}}^{k_b} \frac{\partial^{\bar{l}_p} \exp(-p r_{\mathrm{P}}^2)}{\partial P_x^{\bar{i}_p} \partial P_y^{\bar{j}_p} \partial P_z^{\bar{k}_p}} \tag{27}$$

hybrid functions[17] on the bra side. As it is clear from Eq. (27), these functions reduce to Hermite Gaussians if $\boldsymbol{l}_a = \boldsymbol{l}_b = \mathbf{0}$ and to Cartesian overlap distributions centered on $\mathbf{P}$ without the κ_{ab} factor if $\bar{\boldsymbol{l}}_p = \mathbf{0}$. Introducing the notation for the auxiliary integrals over hybrid bras and Hermite kets as $(\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c)$ and applying the recurrence relations[17] for the functions in Eq. (27) we can write

$$(\Omega_{\boldsymbol{l}_a+\mathbf{1}_x,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c) = \bar{i}_p (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p-\mathbf{1}_x}|\bar{\boldsymbol{l}}_c) + X_{\mathrm{PA}} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c) + \frac{1}{2p} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p+\mathbf{1}_x}|\bar{\boldsymbol{l}}_c) \tag{28}$$

and

$$(\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b+\mathbf{1}_x}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c) = \bar{i}_p (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p-\mathbf{1}_x}|\bar{\boldsymbol{l}}_c) + X_{\mathrm{PB}} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c) + \frac{1}{2p} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p+\mathbf{1}_x}|\bar{\boldsymbol{l}}_c). \tag{29}$$

Relying on these relations, one can start from the two-center Hermite integrals $(\Omega_{\mathbf{0},\mathbf{0}}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{L}}_c)$, which are, by Eq. (20), practically scaled one-center $(\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{L}}_c)$ integrals, and, through hybrid intermediates, convert these into the target $(\Omega_{\boldsymbol{L}_a,\boldsymbol{L}_b}^{\bar{\mathbf{0}}}|\bar{\boldsymbol{L}}_c) = (\boldsymbol{L}_a\boldsymbol{L}_b|\bar{\boldsymbol{L}}_c)$ 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. (23) we apply Eqs. (28) and (29) for the transformation of the bra side.

D. Gill–Head-Gordon–Pople algorithm

Here we consider the original algorithm of Gill, Head-Gordon, and Pople[27] with the modifications needed for three-center ERIs. In this method, the procedure is very similar to the MD5 scheme. The difference lies in the introduction of the β,ζ-scaled auxiliary integrals defined as

$$(\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c)_{\beta,\zeta} = \frac{(2b)^\beta}{(2p)^\zeta} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c), \tag{30}$$

where β and ζ are positive integers. With these quantities, substituting $X_{\mathrm{PA}} = -(2b)/(2p) X_{\mathrm{AB}}$, Eq. (28) can be rewritten as[27]

$$\begin{aligned}(\Omega_{\boldsymbol{l}_a+\mathbf{1}_x,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c)_{\beta,\zeta} = {} & \bar{i}_p (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p-\mathbf{1}_x}|\bar{\boldsymbol{l}}_c)_{\beta,\zeta} - X_{\mathrm{AB}} (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{l}}_c)_{\beta+1,\zeta+1} \\ & + (\Omega_{\boldsymbol{l}_a,\boldsymbol{l}_b}^{\bar{\boldsymbol{l}}_p+\mathbf{1}_x}|\bar{\boldsymbol{l}}_c)_{\beta,\zeta+1},\end{aligned} \tag{31}$$

which is a relation that does not depend explicitly on the Gaussian exponents and therefore can be applied to the β,ζ-scaled auxiliary integrals transformed to the contracted basis.

The strategy of the GHP scheme for three-center ERIs is thus the following. First, the necessary Hermite integrals $(\Omega_{\mathbf{0},\mathbf{0}}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{L}}_c) = (\bar{\boldsymbol{l}}_p + \bar{\boldsymbol{L}}_c)$ are computed for $0 \leq \bar{l}_p \leq L_a + L_b$. Then, all the scaled classes of these integrals required to compute the $(\Omega_{\boldsymbol{l}_a,\mathbf{0}}^{\bar{\mathbf{0}}}|\bar{\boldsymbol{L}}_c)_{0,0}$ classes with Eq. (31) for $L_a \leq l_a \leq L_a + L_b$ are produced. For each of these classes, we need to start from the $(\Omega_{\mathbf{0},\mathbf{0}}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{L}}_c)_{\beta,\zeta}$ scaled Hermite intermediates for $0 \leq \bar{l}_p \leq l_a$. To determine the β,ζ-scaled classes needed for each $(\Omega_{\mathbf{0},\mathbf{0}}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{L}}_c)$ that will be used for the calculation of a given $(\Omega_{\boldsymbol{l}_a,\mathbf{0}}^{\bar{\mathbf{0}}}|\bar{\boldsymbol{L}}_c)_{0,0}$, we have to trace back the recursion defined by Eq. (31). As each recursion step increments $\boldsymbol{l}_a$ by $\mathbf{1}_\sigma$, there are l_a steps. By analyzing the positions where $(\Omega_{\mathbf{0},\mathbf{0}}^{\bar{\boldsymbol{l}}_p}|\bar{\boldsymbol{L}}_c)_{\beta,\zeta}$ and the intermediates connected to it can appear in Eq. (31) during the recursion, we see 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. (31). 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 $1 \le n \le \lceil (l_a - \bar{l}_p)/2 \rfloor + 1$ have been calculated for all the primitive classes, the scaled one-center integrals are transformed to the contracted basis by Eq. (6). When using segmented basis sets, the multiplication work in this contraction step can be reduced to simply multiplying Eq. (22) with the appropriate $d_{a\chi_A}$ coefficients. Following the contraction Eq. (31) is applied, and lastly Eq. (13) is used to build up $\boldsymbol{l}_b$.

E. Rys quadrature method

The algorithms discussed before are all based on calculating scaled Boys functions of various orders and using them as starting values for a recursive procedure. Inspecting these methods and utilizing Eq. (9) it is evident that the target integral can be expressed as

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c) = \sum_{n=0}^{L_a+L_b+L_c} Z_n F_n(\alpha R_{\mathrm{PC}}^2) = \int_0^1 \sum_{n=0}^{L_a+L_b+L_c} Z_n t^{2n} \exp(-\alpha R_{\mathrm{PC}}^2 t^2)\mathrm{d}t, \quad (32)$$

where the values of the coefficients Z_n can be obtained by, for example, backtracking the OS recursions until the integral is only expanded in Boys functions. Eq. (32) is an integral over a polynomial $f(t^2) = \sum_{n=0}^{L_a+L_b+L_c} Z_n t^{2n}$ multiplied by a weight function $W(T,t^2) = \exp(-Tt^2)$ with $T = \alpha R_{\mathrm{PC}}^2$. According to the theory of Gauss–Rys quadrature,[10,17] these integrals can be evaluated exactly as

$$\int_0^1 f(t^2)W(T,t^2)\mathrm{d}t = \sum_{n=1}^{N_{rts}} f(t_n^2)w_n \quad (33)$$

with N_{rts} being an integer satisfying $N_{rts} > \lceil (L_a + L_b + L_c)/2 \rfloor$ and t_n^2 is the square of the nth positive root of the $(2N_{rts})$th order 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_n is the T-dependent weight factor of the quadrature associated with t_n^2. For the calculation of the roots of the Rys polynomials and the weight factors, we followed the approach of King and Dupuis[10] for the $N_{rts} \le 5$ cases and the work of Flocke and Lotrich[56] for $N_{rts} > 5$.

Substituting the identity

$$\frac{1}{|\mathbf{r_1} - \mathbf{r_2}|} = \frac{2}{\pi^{1/2}} \int_0^\infty \exp(-|\mathbf{r}_1 - \mathbf{r_2}|^2 u^2)\mathrm{d}u \quad (34)$$

into Eq. (7) and changing the order of integration, we get

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c) = \frac{2}{\pi^{1/2}} \int_0^\infty \Big[\int\int G_{I_aJ_aK_a}(\mathbf{r_1}, a, \mathbf{A}) G_{I_bJ_bK_b}(\mathbf{r_1}, b, \mathbf{B}) \times \exp(-|\mathbf{r_1} - \mathbf{r_2}|^2 u^2) G_{I_cJ_cK_c}(\mathbf{r_2}, c, \mathbf{C})\, \mathrm{d}\mathbf{r_1}\, \mathrm{d}\mathbf{r_2} \Big] \mathrm{d}u. \quad (35)$$

It is possible to factorize the bracketed integrand in Eq. (35) into three two-dimensional (2D) integrals associated with the three Cartesian directions[12] to get

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c) = \frac{2}{\pi^{1/2}} \int_0^\infty \underline{\Theta}_x^{I_a,I_b,I_c}(u^2)\underline{\Theta}_y^{J_a,J_b,J_c}(u^2)\underline{\Theta}_z^{K_a,K_b,K_c}(u^2)\mathrm{d}u, \quad (36)$$

where

$$\underline{\Theta}_x^{I_a,I_b,I_c}(u^2) = \int\int x_{\mathrm{A}}^{I_a} x_{\mathrm{B}}^{I_b} x_{\mathrm{C}}^{I_c} \exp\left[-\mu X_{\mathrm{AB}} - p x_{\mathrm{P}}^2 - c x_{\mathrm{C}}^2 - u^2 |x_1 - x_2|^2\right] \mathrm{d}x_1\, \mathrm{d}x_2. \quad (37)$$

By making a change of variable from u to t as

$$u^2 = \frac{\alpha\, t^2}{1 - t^2}, \quad (38)$$

$$\mathrm{d}u = \mathrm{d}t\, \alpha^{1/2} \left(\frac{1}{1 - t^2}\right)^{3/2}, \quad (39)$$

defining the modified 2D integrals as

$$\Theta_x^{I_a,I_b,I_c}(t^2) = \underline{\Theta}_x^{I_a,I_b,I_c}(u^2) \exp(\alpha X_{\mathrm{PC}}^2 t^2)(1 - t^2)^{-1/2}, \quad (40)$$

and also noting that as u varies from 0 to infinity, t varies from 0 to 1, we can rewrite Eq. (35) as

$$(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c) = 2\left(\frac{\alpha}{\pi}\right)^{1/2} \int_0^1 \Theta_x^{I_a,I_b,I_c}(t^2)\Theta_y^{J_a,J_b,J_c}(t^2)\Theta_z^{K_a,K_b,K_c}(t^2) \times W(T,t^2)\mathrm{d}t. \quad (41)$$

From Eq. (41) it is clear that $f(t^2)$ can be written as

$$f(t^2) = 2\left(\frac{\alpha}{\pi}\right)^{1/2} \Theta_x^{I_a,I_b,I_c}(t^2)\Theta_y^{J_a,J_b,J_c}(t^2)\Theta_z^{K_a,K_b,K_c}(t^2), \quad (42)$$

and, since the 2D integrals are polynomials in t^2,[12] Eq. (33) takes the form

$$\int_0^1 f(t^2)W(T,t^2)\mathrm{d}t = 2\left(\frac{\alpha}{\pi}\right)^{1/2} \sum_{n=1}^{N_{rts}} \Theta_x^{I_a,I_b,I_c}(t_n^2)\Theta_y^{J_a,J_b,J_c}(t_n^2) \times \Theta_z^{K_a,K_b,K_c}(t_n^2)w_n. \quad (43)$$

The value of $\Theta_x^{I_a,I_b,I_c}(t_n^2)$ can be calculated recursively[17] (and similarly for the y and z directions) as

$$\Theta_x^{i_a+1,0,0}(t_n^2) = \left(X_{\mathrm{PA}} - \frac{\alpha}{p} X_{\mathrm{PC}} t_n^2\right)\Theta_x^{i_a,0,0}(t_n^2) + \frac{i_a}{2p}\left(1 - \frac{\alpha}{p} t_n^2\right)\Theta_x^{i_a-1,0,0}(t_n^2) \quad (44)$$

for i_a and as

$$\Theta_x^{i_a,0,i_c+1}(t_n^2) = \frac{\alpha}{c} X_{\mathrm{PC}} t_n^2 \Theta_x^{i_a,0,i_c}(t_n^2) + \frac{i_a t_n^2}{2(p+c)} \Theta_x^{i_a-1,0,i_c}(t_n^2) \quad (45)$$

for i_c. Finally, i_b is built up by

$$\Theta_x^{i_a,i_b+1,i_c}(t_n^2) = \Theta_x^{i_a+1,i_b,i_c}(t_n^2) + X_{\mathrm{AB}} \Theta_x^{i_a,i_b,i_c}(t_n^2). \quad (46)$$

We note that, in the general case, Eq. (45) contains a third term which can be neglected if the ket side is to be transformed to the solid harmonic Gaussian basis. The derivation of Eq. (45) is given in Appendix A. Instead of performing the assembly step as it is defined by Eq. (43) and starting the recursion of Eq. (44) with $\Theta_x^{0,0,0}(t_n^2) = \pi \exp(-\mu X_{\mathrm{AB}}^2)(1/\sqrt{pc})$ (and analogously for the other directions),[11] it is more beneficial to start with $\Theta_z^{0,0,0}(t_n^2) = \theta_{ab}\kappa_{ab}w_n$ and $\Theta_x^{0,0,0}(t_n^2) = \Theta_y^{0,0,0}(t_n^2) = 1$, making the equation for the assembly

$$\int_0^1 f(t^2)W(T,t^2)\mathrm{d}t = \sum_{n=1}^{N_{rts}} \Theta_x^{I_a,I_b,I_c}(t_n^2)\Theta_y^{J_a,J_b,J_c}(t_n^2)\Theta_z^{K_a,K_b,K_c}(t_n^2). \tag{47}$$

For the four-center ERIs, it has also been shown[25] that the direct evaluation of the target integrals from the 2D integrals is not the only possibility, but it can be advantageous to construct intermediate integrals from the 2D ones and use OS-type recursions to get the target integral.

Here we will investigate three possibilities for the three-center ERIs. In the RYS1 algorithm, we evaluate the $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ integrals directly by Eq. (47). For this purpose, we have to compute $\Theta_x^{i_a,i_b,i_c}(t_n^2)$ for $1 \le i_a \le L_a$, $1 \le i_b \le L_b$, and $1 \le i_c \le L_c$ 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 $L_a \le l_a \le L_a + L_b$, then the OS-type HRR, Eq. (13), is applied. The indices of the necessary 2D integrals here are in the range of $1 \le i_a \le L_a + L_b$, $i_b = 0$, and $1 \le i_c \le L_c$. We also explored here a completely different strategy which has not yet been considered in the literature even for four-center integrals. We utilize that it is also possible to construct the $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0})^{(L_c)}$ auxiliary integrals as

$$\begin{aligned}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0})^{(L_c)} &= \sum_{n=0}^{l_a} Z_n F_{n+L_c}(\alpha R_{\mathrm{PC}}^2) \\ &= \int_0^1 \sum_{n=0}^{l_a} Z_n t^{2(n+L_c)} \exp(-\alpha R_{\mathrm{PC}}^2 t^2)\mathrm{d}t. \end{aligned} \tag{48}$$

In this case, the value of the polynomial f of t_n^2 can be written as

$$f(t_n^2) = t_n^{2L_c} 2(\alpha/\pi)^{1/2}\Theta_x^{i_a,0,0}(t_n^2)\Theta_y^{j_a,0,0}(t_n^2)\Theta_z^{k_a,0,0}(t_n^2). \tag{49}$$

The extra multiplication with $t_n^{2L_c}$ can also be built into $\Theta_z^{0,0,0}(t_n^2)$. In this algorithm (RYS3), the needed 2D integrals are $\Theta_x^{i_a,0,0}(t_n^2)$ for $1 \le i_a \le L_a + L_b$ for all the roots and directions, the $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0})^{(L_c)}$ classes are constructed for $\max(0, L_a - L_c) \le l_a \le L_a + L_b$ by Eq. (47), and the target integrals are built up via Eqs. (12) and (13).

F. Algorithmic considerations

Since its introduction the HRR equation, Eq. (13), 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 that a smaller number of integrals are 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.[56] On the other hand, if we choose to use Eqs. (13) and (5) at the contracted level, we have to first contract the components of the classes $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{L}_c)$ for $L_a \le l_a \le L_a + L_b$, 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 central processing unit (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 we see that it is not a trivial decision where Eqs. (13) and (5) 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 a significant number of uncontracted functions in generally contracted basis sets, e.g., in the cc-pVXZ bases.[57,58] 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,[59] 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. (6) are divided by the above mentioned norm. Further notes on the efficient treatment of integral contraction will be discussed in Sec. IV.

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

Algorithm 1. abc primitive loop order.

```
Loop over a
  Loop over b
    Algorithm pPRE2: estimate (00|0) for the smallest c
      Loop over c
        Algorithm pPRE1: estimate (00|0)
        Algorithm OUT: Build up (l_a 0|L_c) for L_a ≤ l_a ≤ L_a + L_b in the Cartesian
          Gaussian basis
        Algorithm IN: Build up (L_a L_b|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 χ_A (executed only in the case of algorithm OUT)
  Loop over χ_B
    Algorithm cPRE2: Look up the integral of highest absolute value in
      the contracted (l_a 0|L_c) classes needed for the contracted (L_a L_b|L_c) class with
      the smallest c
    Algorithm cPRE3: Estimate the integral of highest absolute value in
      the contracted (l_a 0|L_c) classes needed for the contracted (L_a L_b|L_c) class with
      the smallest c
  Loop over χ_C
    Algorithm cPRE1: Look up the integral of highest absolute value in
      the contracted (l_a 0|L_c) classes needed for the contracted (L_a L_b|L_c) class
    Algorithm OUT: perform HRR to get (L_a L_b|L_c), perform solid harmonic
      transformation
    End loop
  End loop
End loop
```

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 we have to precalculate the a- and b-dependent quantities 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.

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 our code, as usual, the entire shell triplets are prescreened invoking the Schwartz inequality,[7] and we also employ the distance-dependent estimator of Valeev and co-workers.[60] 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, **A**) which contributes to only one contracted function χ_A. Then, ε is not divided by the number of primitives on **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, we will use the value of the (**00**|**0**) ERI evaluated with the exponents of the functions of higher angular momentum. Instead of directly calculating (**00**|**0**) according to Eq. (8), we can use the upper bound for the zeroth-order Boys function,[17] from which we get

$$(\mathbf{00}|\mathbf{0}) = \theta_{pc}\kappa_{ab}F_0(\alpha R_{\mathrm{PC}}^2) \leq \theta_{pc}\kappa_{ab}\min\left(1, \sqrt{\frac{\pi}{4\alpha R_{\mathrm{PC}}^2}}\right). \quad (50)$$

The minimum criterion appears since the approximation used in Eq. (50) 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 $\alpha R_{\mathrm{PC}}^2 = 0$). In actual calculations, it is more beneficial to use the square of the rightmost side of Eq. (50) for screening, so the expensive square root calculation only has to be done for classes with small αR_{PC}^2 that survive the prescreening. In this method (algorithm pPRE1), the estimate for $|(\mathbf{00}|\mathbf{0})|^2$ is compared to the square of the threshold, and if the former value is greater, the class is evaluated. This is not an exact screening since Eq. (50) is not a rigorous upper bound for the target ERIs. Instead, this approach is related to the one proposed by Almlöf and co-workers,[6] who used the common factor (in our case $\kappa_{ab}\theta_{pc}$) by which all the integrals in a class are multiplied to gain an estimate for the magnitude of the primitive integrals in a given class. In our scheme, this value is multiplied by a number smaller than 1, resulting in a less precise but more efficient screening method. In practice, we found that it can be more efficient to screen a batch of primitive exponent triplets than each individual one. Here we make use of the fact that the value of the right-hand side of Eq. (50) increases

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 (00|0) for the smallest b
     Loop over b
       Algorithm pPRE1: estimate (00|0)
       Algorithm OUT: Build up (l_a 0|L_c) for L_a ≤ l_a ≤ L_a + L_b in the Cartesian
         Gaussian basis
       Algorithm IN: Build up (L_a L_b|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 χ_A (executed only in the case of algorithm OUT)
  Loop over χ_B
    Algorithm cPRE1: Look up the integral of highest absolute value in
      the contracted (l_a 0|L_c) classes needed for the contracted (L_a L_b|L_c) class
    Algorithm cPRE4: Estimate the integral of highest absolute value in
      the contracted (l_a 0|L_c) classes needed for the contracted (L_a L_b|L_c) class
    Algorithm OUT: perform HRR to get (L_a L_b|L_c), perform solid harmonic
      transformation
   End loop
  End loop
End loop
```

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 we found this choice to be inefficient, as it will be discussed in Sec. V. The accuracy of the pPRE2 screening method and the effect of its inexact nature on HF energies are discussed in the supplementary material. The derivation of an exact, but less efficient prescreening method based on the Schwartz inequality,[7] is presented in Appendix B.

The primitive prescreening described above does not reduce the work of the HRR 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. (13) and (5) 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, we will work with the 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 accuracy is demonstrated in the supplementary material. An alternative method is to estimate the integral with the highest absolute value out of the screened batch. For this purpose, we save the estimates of the $(\mathbf{00}|\mathbf{0})$ integrals made by Eq. (50). 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) we cannot assume 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).

Finally, from the recursive formulas for the calculation of six-dimensional integrals given in Secs. II B–II D 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.[22,28] In our algorithms, the related tree-search problems were treated utilizing the ideas of Ryu and co-workers.[22]

III. FLOATING POINT OPERATION COUNTS

The FLOP requirements of the discussed schemes were estimated by a simple 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, we 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 our 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 Sec. II supposing that reusable compound quantities, such as $(\alpha/p)X_{\mathrm{PC}}$ in Eq. (11), 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 were 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[57] 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[59] (cc-pVXZ-RI) for the ket side.

The overall FLOP counts for all the shell triplets for the various algorithms are presented in Table I. Figures that show the theoretical performance of the other algorithms relative to the OS1 scheme can be found in the supplementary material. 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 than the OS1 for any shell triplets except for $(\boldsymbol{ss}|\boldsymbol{p})$, where the MD1, MD2, MD4, and MD5 schemes are slightly cheaper since the additional calculation of $\frac{\alpha}{c}$ from Eq. (12) is not necessary. Looking at the best performing MD3 and MD6 schemes, we see that the use of Eqs. (28) and (29) is preferred to the assembly of Eq. (23), 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. (31) 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. (31) 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. (11) 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. (12) 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 $\max(0, L_a - L_c) \leq l_a \leq L_a + L_b$, the growth of $\boldsymbol{L}_c$ also increases the work in the quadrature 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 FLOP 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 Secs. IV and V, 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.

TABLE I. FLOP counts for the various algorithms with the cc-pVXZ basis sets.

	X			
Algorithm	D	T	Q	5
OS1	445 777	2 231 707	14 074 904	71 407 908
OS2	545 297	2 967 883	19 981 747	106 671 377
OS3	632 210	3 465 805	22 599 746	116 871 757
OS4	754 037	3 587 118	21 812 481	106 908 381
MD1	599 215	3 801 560	30 617 263	198 278 829
MD2	555 359	3 165 292	22 249 286	125 117 638
MD3	474 978	2 473 785	15 766 358	80 931 165
MD4	616 235	3 824 184	29 178 035	173 497 467
MD5	570 267	3 272 596	22 532 400	121 220 098
MD6	470 050	2 420 151	15 243 362	77 170 230
GHP	499 430	3 188 703	25 032 932	152 491 888
RYS1	622 518	3 181 603	20 929 684	112 060 719
RYS2	585 778	2 902 749	18 203 750	92 155 512
RYS3	467 187	2 308 256	14 413 073	72 659 045

TABLE II. FLOP counts for the four different OS1 algorithms with the cc-pV*X*Z basis sets.

	X			
Algorithm	D	T	Q	5
IN-abc	566 748	2 664 883	15 919 037	79 233 985
IN-cab	565 054	2 662 374	15 916 043	79 232 960
OUT-abc	445 777	2 231 707	14 074 904	71 407 908
OUT-cab	443 201	2 227 609	14 069 600	71 404 912

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 II. 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 ***d*** and higher shells are uncontracted in the investigated bases. These results, 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.

IV. 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 Sec. II F have been implemented in the Mrcc program suite[61] by means of automated code generation. The abc and cab variants of the GHP algorithm for the (***ps***|***s***)–(***ps***|***g***) triplets have been implemented in the conventional way. An individual Fortran 95 subroutine was created for every shell triplet up to (***hh***|***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 of Eqs. (11)–(13) and the 3D recurrence of Eq. (21), 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. (44). 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 has been implemented utilizing the reduced multiplication scheme of Lindh and co-workers.[25] All the intermediate and target integrals are stored in one-index arrays. 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. (13) 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. 1, 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 (***ps***| bra, where the target integrals are calculated directly from the one-center ones, Eq. (31) 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. We 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 our implementation, and the values of the coefficients in Eq. (5) are explicitly generated into the code. We have also considered 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,[56] 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. It was our experience 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. (6) 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 **C** and N_S is the number of integrals in the class (for algorithm IN) or in the

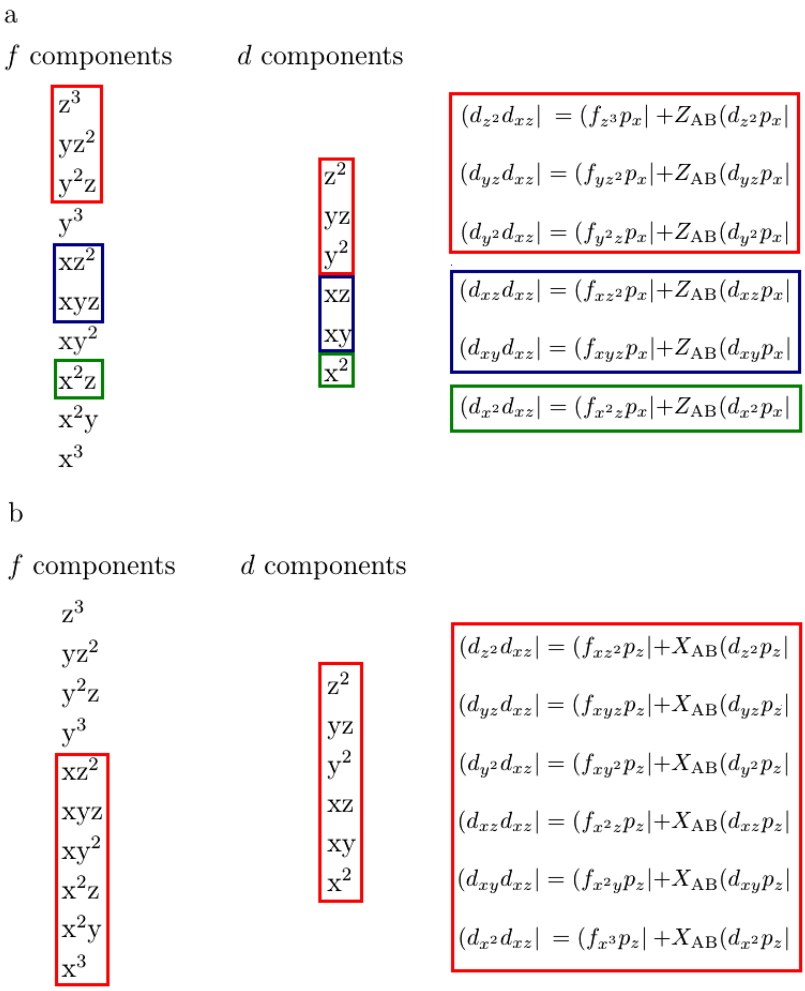

FIG. 1. Two possible ways of calculating integrals with a (***dd***| bra side and $\boldsymbol{l}_b = (1, 0, 1)$ are by (a) incrementing z and (b) incrementing x in $\boldsymbol{l}_b$. The indices for the Cartesian components increase as we proceed from top to bottom in the columns for the f and d shells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

necessary ($\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{L}_c$) classes (for algorithm OUT). For example, for a (***dd***|***d***) class $N_S = 5 \times 5 \times 5 = 125$ for algorithm IN and $N_S = 6 \times 1 \times 6 + 10 \times 1 \times 6 + 15 \times 1 \times 6 = 186$ for algorithm OUT because here we need the (***ds***|***d***), (***fs***|***d***), and (***gs***|***d***) classes for the HRR. It is also noteworthy that, at the contraction of the functions centered on **B**, instead of performing the summation of Eq. (6) 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 **A**), it is more cache-friendly to do the summation in a buffer array of size $N_{\chi_C} N_S$, than to copy the data into the array that will be used for the contraction of primitives centered on **A**.

The implementation of ERIs also utilizes a coarse-grained OpenMP parallelization for the innermost atomic loop. A figure showing the performance of the parallelization can be found in the supplementary material. We also note that, to demonstrate the efficiency of the generated implementation, we have also coded a subroutine that uses the OS1 scheme for arbitrary angular momenta. Here, the recursions of Eqs. (11) and (12) 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[56] as described above.

V. PERFORMANCE TESTS

In this section, we present the wall time performances of the implemented algorithms measured using a single core of a 2-core 3.00 GHz Intel Xeon E3110 CPU. The generated subroutines were compiled with the Intel Fortran compiler using the highest level of optimization. Measurements were carried out for penicillin[62] (PEN) and two DNA systems with one (DNA_1) and two (DNA_2) adenine-thymine base pairs.[63] The threshold ε for contracted integrals was set to 10^{-10} E_h in all of the calculations. Only the results 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. Cache simulations were performed for hydrogen peroxide ($R_{OO} = 2.7514$ bohrs, $R_{HO} = 1.8274$ bohrs, $\sphericalangle_{HOO} = 102.32°$, dihedral angle = 115.89°) with the Valgrind program package[64] 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 4 MB 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.

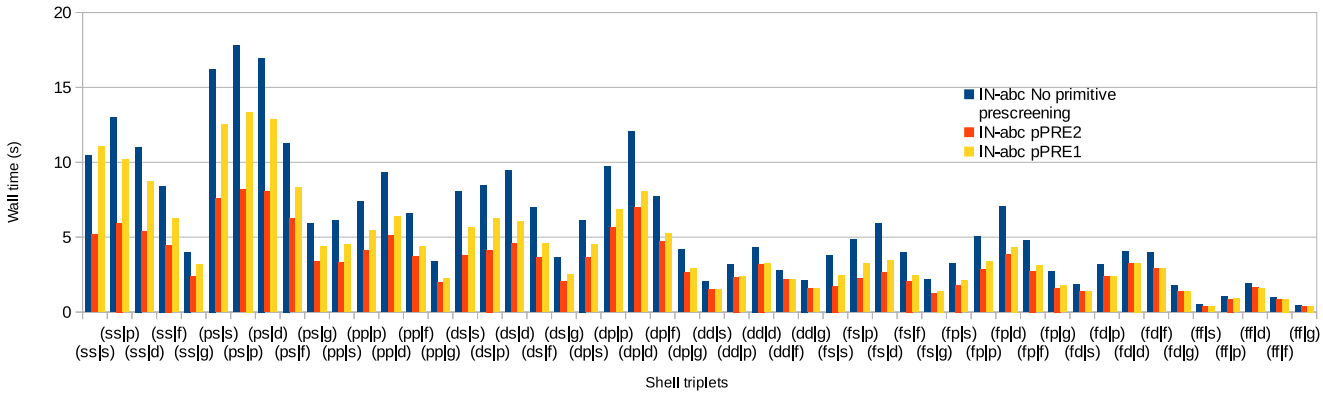

FIG. 2. Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA_2 molecule with the cc-pVTZ basis set by applying the OS1-IN-abc algorithm with various prescreening strategies.

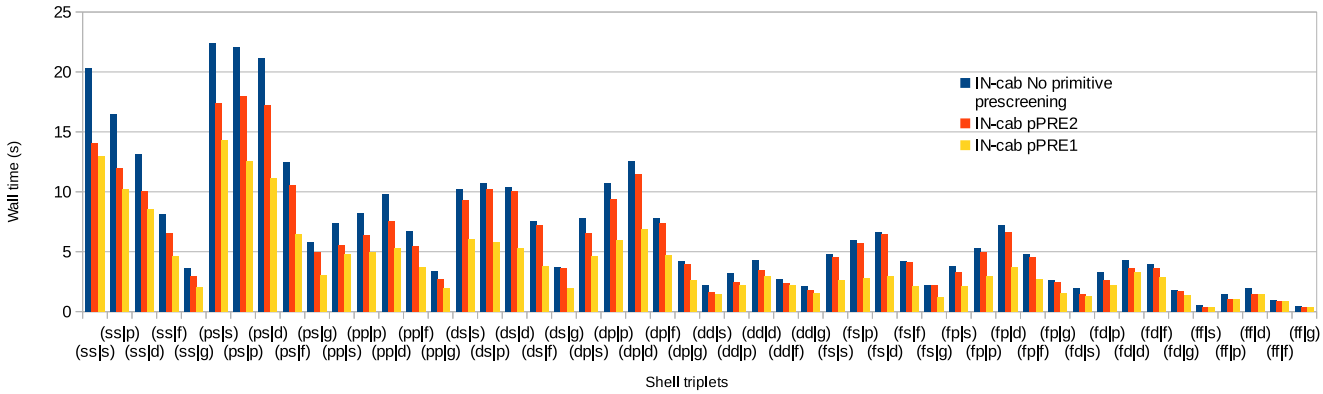

FIG. 3. Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA_2 molecule with the cc-pVTZ basis set by applying the OS1-IN-cab algorithm with various prescreening strategies.

Fig. 2 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. 3 we see that the pPRE1 prescreening is more economical in the case of a cab scheme since the Schwartz 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. 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 are more contracted

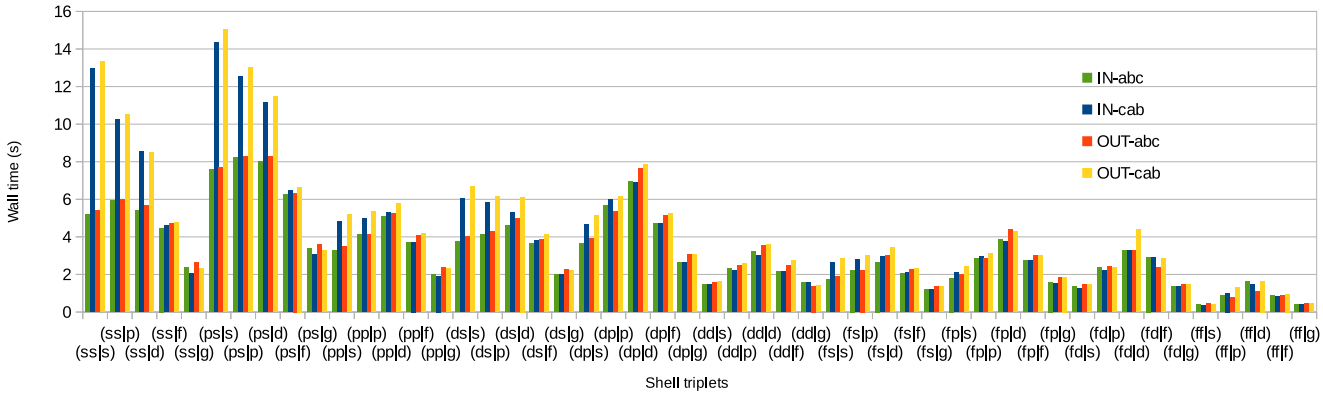

FIG. 4. Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA_2 molecule with the cc-pVTZ basis set by applying the four OS1 algorithms with the most efficient prescreening strategies.

TABLE III. Cache performance simulation results for H_2O_2 with the cc-pVTZ basis set.

	Algorithm			
Event	IN-abc	IN-cab	OUT-abc	OUT-cab
L1 instruction fetch miss	687 995	720 295	665 954	820 972
LL instruction fetch miss	576 727	582 575	641 900	666 989
L1 data read miss	219 741	192 668	252 112	202 708
LL data read miss	199 655	191 036	200 703	199 053
L1 data write miss	484 132	385 047	552 062	407 074
LL data write miss	482 194	383 336	544 077	404 681

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. We conclude 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 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. 4. For triplets containing small angular momenta, the cab schemes are inefficient, even without primitive prescreening (see also Figs. 2 and 3). 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** for (***ss***|***s***) is $N_{\chi C}$ 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 $N_{\chi C}$ 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. We must note, however, that only the ***s*** and ***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 presented in Table III, while the results with the other basis sets can be found in the supplementary material. 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 is 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 (***fd***|***f***) and (***ff***|***d***) cases with the TZ basis, and also explains why the performance of this approach improves with higher basis sets. 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. 5 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 Fig. 4, 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

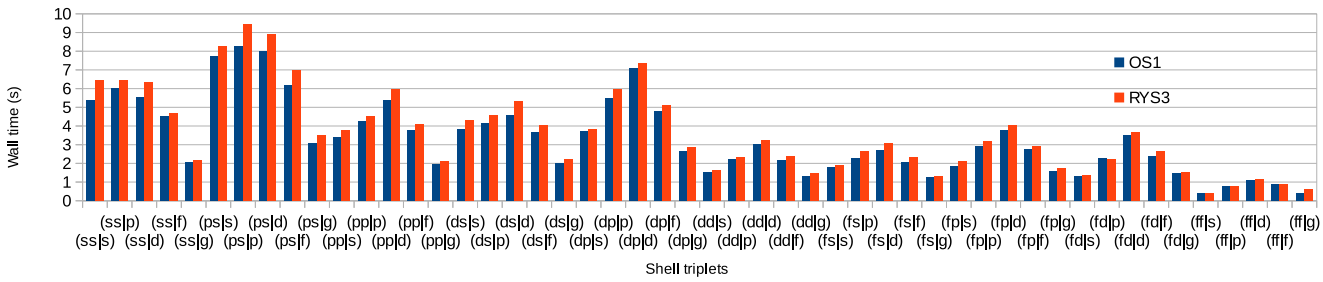

FIG. 5. Wall times measured in seconds obtained by calculating all three-center ERIs of the DNA_2 molecule with the cc-pVTZ basis set by applying the most efficient OS1 and RYS3 algorithms.

TABLE IV. Recommended algorithms for the various shell triplets.

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

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[65] and Golub's matrix method,[66] while otherwise the less expensive schemes proposed by King and Dupuis[10] 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 (*fd*|*p*) and (*gd*|*p*) shell triplets. The GHP scheme is competitive for the implemented cases (see Sec. IV) with the 5Z basis, where the degree of contraction is the highest. For smaller basis sets, for the (*ps*|*p*) triplet, GHP performs slightly

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

	X							
	D		T		Q		5	
Test system	Time	N + M	Time	N + M	Time	N + M	Time	N + M
Penicillin	0.008	430 + 2 136	0.022	946 + 2 478	0.088	1 864 + 3 504	0.372	3 178 + 5 033
DNA_1	0.016	625 + 3 071	0.049	1428 + 3 575	0.201	2 735 + 5 087	0.883	4 670 + 7 351
Indinavir	0.033	865 + 4 231	0.118	2008 + 4 965	0.492	3 885 + 7 167	2.251	6 680 + 10 471
Angiotensin II	0.104	1405 + 6 883	0.380	3244 + 8 055	1.609	6 255 + 11 571	7.245	10 730 + 16 843
DNA_4	0.474	2746 + 19 820	1.777	6192 + 15 794	8.307	11 774 + 22 202	33.174	20 012 + 31 744
Halloysite	1.306	3700 + 19 820	4.607	7970 + 22 435	19.854	14 855 + 30 280	68.447	24 985 + 41 510

better than OS1 since here the number of integrals to be contracted, that is, the number of integrals included in the scaled classes $(\Omega^{\bar{0}}_{\mathbf{0},\mathbf{0}}|\bar{\mathbf{1}})_{1,1}$ and $(\Omega^{\bar{1}}_{\mathbf{0},\mathbf{0}}|\bar{\mathbf{1}})_{0,1}$ needed for Eq. (31), 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 bases, respectively.

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 we investigated, it was our experience that the best algorithm for a given shell triplet with a given basis is mostly independent of the calculated system. Based on our measurements with the cc-pV*X*Z bases for first row elements, in Table IV we present our recommendations for the algorithms for the shell triplets up to $(\boldsymbol{hh}|\boldsymbol{i})$. The list compiled in Table IV 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.

VI. BENCHMARK CALCULATIONS

To demonstrate the efficiency of our implementation based on the above recommendation, in Table V we present the wall times measured for the evaluation of three-center ERIs for test systems of various size, namely, penicillin,[62] DNA fragments containing 1 and 4 adenine-thymine base pairs[67] (DNA_1 and DNA_4, respectively), indinavir,[68] angiotensin II,[69] and a halloysite clay structure.[70] The measurements were carried out using 8 cores of a 3.00 GHz Intel Xeon E5-1660 CPU. The results are close to quadratic scaling with the total number of basis functions due to the various integral screenings, and the prefactor is kept small by the efficient implementation. We have also experienced a constant speedup of about 3 compared to our general purpose routine using the OS1 scheme, which shows that we can gain an efficient implementation optimized for each shell triplet separately. We 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, we compared our three-center code to our previous OS-based four-center integral program[71] and have found that the former program is roughly 3.5 times faster than the latter one. We also note that the efficiency of our integral code has been recently demonstrated also in the case of the integral-direct local correlation approach of Ref. 9, where roughly one-third of the entire computation time is spent on the calculation of three-center ERIs.

VII. CONCLUSIONS

We have compared 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. 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, we concluded that the Obara–Saika scheme, utilizing the vertical recurrence relation of Ahlrichs,[40] 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. The reason for this is that performing these two tasks on primitive integrals allows for the use of more effective prescreening and memory layout. Based on our investigations, we have presented a recommendation 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.

SUPPLEMENTARY MATERIAL

See supplementary material for the analysis of the prescreening schemes presented in Sec. II F, for the relative theoretical performances of the investigated algorithms referred to in Sec. III, for the wall time measurement and cache simulation results discussed in Sec. V, for the performance of the ERI calculation on multiple CPU cores, and for the geometries of the molecules used in the performance tests and benchmark calculations.

ACKNOWLEDGMENTS

The authors are indebted to Professor Reinhart Ahlrichs and Dr. Gerald Knizia for useful discussions. The computing time granted on the Hungarian HPC Infrastructure at NIIF Institute, Hungary, is gratefully acknowledged.

APPENDIX A: IMPROVED RECURRENCE RELATION FOR THE 2D INTEGRALS OF THE RYS SCHEME

In the general case, Eq. (45) contains a third term and has the form[17]

$$\Theta_x^{i_a,0,i_c+1}(t_n^2) = \frac{\alpha}{c} X_{\mathrm{PC}} t_n^2 \Theta_x^{i_a,0,i_c}(t_n^2) + \frac{i_a t_n^2}{2(p+c)} \Theta_x^{i_a-1,0,i_c}(t_n^2) + \frac{i_c}{2c}\left(1 - \frac{\alpha}{c} t_n^2\right) \Theta_x^{i_a,0,i_c-1}(t_n^2). \tag{A1}$$

With the help of Eq. (12) we can show that, if the ket side will be transformed into the solid harmonic Gaussian basis, the third term on the left-hand side of Eq. (A1) can be omitted. To see this, we first notice from backtracking the recursion defined by Eq. (12) 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

$$l_c^{\#} = l_c - n \tag{A2}$$

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

$$(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)} = \sum_{n=1}^{N_{rts}} t_n^{2m} \Theta_x^{i_a,0,i_c}(t_n^2) \Theta_y^{j_a,0,j_c}(t_n^2) \Theta_z^{k_a,0,k_c}(t_n^2)\,. \tag{A3}$$

Substituting Eq. (A1) into Eq. (A3) we get

$$\begin{aligned}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)} = \sum_{n=1}^{N_{rts}} t_n^{2m} \Big[&\frac{\alpha}{c} X_{\mathrm{PC}} t_n^2 \Theta_x^{i_a,0,i_c-1}(t_n^2) + \frac{i_a t_n^2}{2(p+c)} \Theta_x^{i_a-1,0,i_c-1}(t_n^2) + \frac{i_c}{2c}\left(1 - \frac{\alpha}{c} t_n^2\right) \Theta_x^{i_a,0,i_c-2}(t_n^2)\Big] \\ \times \Big[&\frac{\alpha}{c} Y_{\mathrm{PC}} t_n^2 \Theta_y^{j_a,0,j_c-1}(t_n^2) + \frac{j_a t_n^2}{2(p+c)} \Theta_y^{j_a-1,0,j_c-1}(t_n^2) + \frac{j_c}{2c}\left(1 - \frac{\alpha}{c} t_n^2\right) \Theta_y^{j_a,0,j_c-2}(t_n^2)\Big] \\ \times \Big[&\frac{\alpha}{c} Z_{\mathrm{PC}} t_n^2 \Theta_z^{k_a,0,k_c-1}(t_n^2) + \frac{k_a t_n^2}{2(p+c)} \Theta_z^{k_a-1,0,k_c-1}(t_n^2) + \frac{k_c}{2c}\left(1 - \frac{\alpha}{c} t_n^2\right) \Theta_z^{k_a,0,k_c-2}(t_n^2)\Big].\end{aligned} \tag{A4}$$

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. (A3), which is used in the expansion of $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_c)^{(m)}$ by Eq. (12) if we go three steps back in the recursion. The terms containing the third 2D integral from one or more brackets in Eq. (A4) are used to build the $(\boldsymbol{l}_a^{\#}\boldsymbol{0}|\boldsymbol{l}_c^{\#})^{(m+n)}$ classes with Eq. (A3) where $0 \le n \le 3$ (because the third 2D integral can be multiplied by a quantity that does or does not contain t_n^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. (A2), the contributions containing the third terms in the brackets in Eq. (A4) will be canceled during the solid harmonic transformation and can be taken to be zero, which means that Eq. (A1) reduces to Eq. (45). The same reasoning applies to the second term in Eq. (44) in the case of $\boldsymbol{L}_b = \boldsymbol{0}$, when the third and fourth terms in Eq. (11) vanish.

APPENDIX B: A RIGOROUS UPPER BOUND FOR PRIMITIVE THREE-CENTER ERIs

It is possible to construct an exact prescreening scheme for the primitive integrals based on the Schwartz inequality,

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)|^2 \le |(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)|\;|(\boldsymbol{L}_c|\boldsymbol{L}_c)|, \tag{B1}$$

by giving upper bounds to the integrals on the right-hand side of Eq. (B1). In fact, the exact value of $(\boldsymbol{L}_c|\boldsymbol{L}_c)$ can be simply calculated by using Eq. (12) and noting that in this special case $R_{\mathrm{PC}} = 0$, which gives

$$(\boldsymbol{L}_c|\boldsymbol{L}_c) = \frac{L_c!}{(4c)^{L_c}} (\boldsymbol{0}|\boldsymbol{0})^{(L_c)} = \frac{L_c!}{(4c)^{L_c}} \theta_{cc} \frac{1}{2L_c+1}, \tag{B2}$$

where it was also exploited that $F_n(0) = 1/(2n+1)$.[17] To gain an upper bound for $|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)|$, we have to track back the recursions necessary to build up this integral. Let us first define the maximum absolute value component of $\mathbf{R}_{\mathrm{AB}}$ as

$$mR_{\mathrm{AB}} = \max(|X_{\mathrm{AB}}|, |Y_{\mathrm{AB}}|, |Z_{\mathrm{AB}}|). \tag{B3}$$

Then, by Eq. (13), an upper bound for $|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)|$ is

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)| \leq \Big[\sum_{l_b=0}^{L_b}\binom{L_b}{l_b} mR_{\mathrm{AB}}^{l_b}\Big] M_{L_aL_bl_b} = U_{HRR}M_{L_aL_bl_b}, \tag{B4}$$

where $M_{L_aL_bl_b}$ is a value that is greater than the absolute value of any of the integrals $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{l}_b\boldsymbol{0})$ for $L_a \leq l_b \leq L_a + L_b$. Proceeding in the same manner for the bra side, we get

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)| \leq U_{HRR}^2 M_{l_al_b}, \tag{B5}$$

where, similarly, $M_{l_al_b}$ is an upper bound for $|(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_b\boldsymbol{0})|$ with $L_a \leq l_a \leq L_a + L_b$ and $L_a \leq l_b \leq L_a + L_b$. To get an upper bound for these types of integrals, we inspect the VRR for four-center ERIs[19]

$$\begin{aligned}(\boldsymbol{l}_a\boldsymbol{0}|[\boldsymbol{l}_b + \boldsymbol{1}_x]\boldsymbol{0})^{(n)} = X_{\mathrm{PA}}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_b\boldsymbol{0})^{(n)} + \frac{i_b}{2p}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_b - \boldsymbol{1}_x\boldsymbol{0})^{(n)} \\ - \frac{i_b}{4p}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_b - \boldsymbol{1}_x\boldsymbol{0})^{(n+1)} \\ + \frac{i_a}{4p}(\boldsymbol{l}_a - \boldsymbol{1}_x\boldsymbol{0}|\boldsymbol{l}_b\boldsymbol{0})^{(n+1)},\end{aligned} \tag{B6}$$

which can be used to expand $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{l}_b\boldsymbol{0})$ type ERIs in $(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})$ type ones. The highest number of terms in this expansion, N_{VRR1}, will belong to $([\boldsymbol{L}_a + \boldsymbol{L}_b]\boldsymbol{0}|[\boldsymbol{L}_b + \boldsymbol{L}_b]\boldsymbol{0})$. We can then write

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)| \leq U_{HRR}^2 N_{VRR1} U_{VRR} M_{l_a}, \tag{B7}$$

where

$$U_{V\,RR} = \max\Big[mR_{\mathrm{PA}}^{L_a+L_b}, \Big(\frac{L_a + L_b}{2p}\Big)^{\lfloor\frac{L_a+L_b}{2}\rceil}, 1\Big] \tag{B8}$$

is the biggest recursion coefficient that can occur, and M_{l_a} is an upper bound for $|(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})|$ with $0 \leq l_a \leq L_a + L_b$. N_{VRR1} can be given as

$$N_{VRR1} = \sum_{m=0}^{\lfloor\frac{L_a+L_b}{2}\rceil}\binom{L_a + L_b - m}{m} 2^{L_a+L_b-m}. \tag{B9}$$

It only remains to give an appropriate value of M_{l_a}, for which we use the VRR

$$\begin{aligned}(\boldsymbol{l}_a + \boldsymbol{1}_x\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})^{(n)} = X_{\mathrm{PA}}(\boldsymbol{l}_a\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})^{(n)} \\ + \frac{i_a}{2p}(\boldsymbol{l}_a - \boldsymbol{1}_x\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})^{(n)} - \frac{i_a}{4p}(\boldsymbol{l}_a - \boldsymbol{1}_x\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})^{(n+1)}\end{aligned} \tag{B10}$$

to expand $([\boldsymbol{L}_a + \boldsymbol{L}_b]\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})$ in N_{VRR2} $(\boldsymbol{0}\boldsymbol{0}|\boldsymbol{0}\boldsymbol{0})^{(n)}$ type integrals, the greatest of which will be $\kappa_{ab}^2\theta_{pp}F_0(0) = \kappa_{ab}^2\theta_{pp}$. We then get

$$|(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)| \leq U_{HRR}^2 N_{VRR1} N_{VRR2} U_{VRR}^2 \kappa_{ab}^2\theta_{pp} \tag{B11}$$

with

$$N_{VRR2} = \sum_{m=0}^{\lfloor\frac{L_a+L_b}{2}\rceil}\binom{L_a + L_b - m}{m} 2^m. \tag{B12}$$

Note that U_{HRR} only depends on the inter-nuclear distances in the bra and L_b, N_{VRR1}, and N_{VRR2} only depend on $L_a + L_b$, and $mR_{\mathrm{PA}} = b/p\, mR_{\mathrm{AB}}$. If desired, a bound for integrals over spherical harmonic Gaussians can be given by multiplying the screening value by $(2L_a + 1)\,(2L_b + 1)\,(2L_c + 1)$ and the maximal coefficients in Eq. (5) for the three shells. In our experience if we neglect this, the integrals that are falsely discarded have the same magnitude as the tolerance. Applying the scheme described above, roughly an extra 5% and 10% of the integrals are calculated with respect to the approaches presented in Sec. II F for the TZ and QZ bases, respectively, and the wall times increase by about 10%.

We note that an upper bound can also be derived directly for the $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_c)$ integrals in a way similar to the one outlined here for $(\boldsymbol{L}_a\boldsymbol{L}_b|\boldsymbol{L}_a\boldsymbol{L}_b)$, but the resulting scheme is less efficient due to the increased number of FLOPs and logical operations necessary inside the primitive loops.

[1] S. F. Boys and I. Shavitt, University of Wisconsin Naval Research Laboratory Report No. WIS-AF-13, 1959.
[2] E. J. Baerends, D. E. Ellis, and P. Ros, Chem. Phys. **2**, 41 (1973).
[3] J. L. Whitten, J. Chem. Phys. **58**, 4496 (1973).
[4] B. I. Dunlap, J. W. D. Connolly, and J. R. Sabin, J. Chem. Phys. **71**, 3396 (1979).
[5] B. I. Dunlap, Phys. Chem. Chem. Phys. **2**, 2113 (2000).
[6] J. Almlöf, K. Fægri, Jr., and K. Korsell, J. Comput. Chem. **3**, 385 (1982).
[7] M. Häser and R. Ahlrichs, J. Comput. Chem. **10**, 104 (1989).
[8] F. Weigend, Phys. Chem. Chem. Phys. **4**, 4285 (2002).
[9] P. R. Nagy, G. Samu, and M. Kállay, J. Chem. Theory Comput. **12**, 4897 (2016).
[10] H. F. King and M. Dupuis, J. Comput. Phys. **21**, 144 (1976).
[11] M. Dupuis, J. Rys, and H. F. King, J. Chem. Phys. **65**, 111 (1976).
[12] J. Rys, M. Dupuis, and H. F. King, J. Comput. Chem. **4**, 154 (1983).
[13] A. Komornicki and H. F. King, J. Chem. Phys. **134**, 244115 (2011).
[14] H. F. King, J. Phys. Chem. A **120**, 9348 (2016).
[15] M. Dupuis and A. Marquez, J. Chem. Phys. **114**, 2067 (2001).
[16] M. Dupuis, Comput. Phys. Commun. **134**, 150 (2001).
[17] T. Helgaker, P. Jørgensen, and J. Olsen, *Molecular Electronic Structure Theory* (Wiley, Chichester, 2000).
[18] L. E. McMurchie and E. R. Davidson, J. Comput. Phys. **26**, 218 (1978).
[19] S. Obara and A. Saika, J. Chem. Phys. **84**, 3963 (1986).
[20] H. Honda, T. Yamaki, and S. Obara, J. Chem. Phys. **117**, 1457 (2002).
[21] M. Head-Gordon and J. A. Pople, J. Chem. Phys. **89**, 5777 (1988).
[22] U. Ryu, Y. S. Lee, and R. Lindh, Chem. Phys. Lett. **185**, 562 (1991).
[23] B. G. Johnson, P. M. W. Gill, and J. A. Pople, Chem. Phys. Lett. **206**, 229 (1992).
[24] T. P. Hamilton and H. F. Schaefer III, Chem. Phys. **150**, 163 (1991).
[25] R. Lindh, U. Ryu, and B. Liu, J. Chem. Phys. **95**, 5889 (1991).
[26] R. Lindh, Theor. Chim. Acta **85**, 423 (1993).
[27] P. M. W. Gill, M. Head-Gordon, and J. A. Pople, Int. J. Quantum Chem. **36**, 269 (1989).
[28] B. G. Johnson, P. M. W. Gill, and J. A. Pople, Int. J. Quantum Chem. **40**, 809 (1991).
[29] P. M. W. Gill, B. G. Johnson, and J. A. Pople, Chem. Phys. Lett. **217**, 65 (1994).
[30] B. G. Johnson, P. M. W. Gill, J. A. Pople, and D. J. Fox, Chem. Phys. Lett. **206**, 239 (1993).
[31] P. M. W. Gill, M. Head-Gordon, and J. A. Pople, J. Phys. Chem. **94**, 5564 (1990).
[32] P. M. W. Gill and J. A. Pople, Int. J. Quantum Chem. **40**, 753 (1991).
[33] P. M. W. Gill and B. G. Johnson, Int. J. Quantum Chem. **40**, 745 (1991).
[34] P. M. W. Gill, Adv. Quantum Chem. **25**, 141 (1994).
[35] A. M. Köster, J. Chem. Phys. **104**, 4114 (1996).
[36] A. M. Köster, J. Chem. Phys. **118**, 9943 (2003).
[37] P. Calaminici, V. D. Domínguez-Soria, G. Geudtner, E. Hernández-Marín, and A. M. Köster, Theor. Chem. Acc. **115**, 221 (2006).
[38] S. Reine, E. Tellgren, and T. Helgaker, Phys. Chem. Chem. Phys. **9**, 4771 (2007).
[39] S. Reine, T. Helgaker, and R. Lindh, Wiley Interdiscip. Rev.: Comput. Mol. Sci. **2**, 290 (2012).
[40] R. Ahlrichs, Phys. Chem. Chem. Phys. **6**, 5119 (2004).
[41] E. F. Valeev, Libint: A library for the evaluation of molecular integrals of many-body operators over Gaussian functions, http://libint.valeyev.net/.
[42] E. F. Valeev and C. L. Janssen, J. Chem. Phys. **121**, 1214 (2004).
[43] H.-J. Werner, G. Knizia, and F. R. Manby, Mol. Phys. **109**, 407 (2011).
[44] Y. Shao and M. Head-Gordon, Chem. Phys. Lett. **323**, 425 (2000).
[45] A. Sodt, J. E. Subotnik, and M. Head-Gordon, J. Chem. Phys. **125**, 194109 (2006).
[46] R. Polly, H.-J. Werner, F. R. Manby, and P. J. Knowles, Mol. Phys. **102**, 2311 (2004).

[47]S. Reine, E. Tellgren, A. Krapp, T. Kjærgaard, T. Helgaker, B. Jansik, S. Høst, and P. Salek, J. Chem. Phys. **129**, 104101 (2008).
[48]S. F. Manzer, E. Epifanovsky, and M. Head-Gordon, J. Chem. Theory Comput. **11**, 518 (2014).
[49]D. Mejía-Rodríguez and A. M. Köster, J. Chem. Phys. **141**, 124114 (2014).
[50]D. Mejía-Rodríguez, X. Huang, J. M. del Campo, and A. M. Köster, Adv. Quantum Chem. **71**, 41 (2015).
[51]C. Köppl and H.-J. Werner, J. Chem. Theory Comput. **12**, 3122 (2016).
[52]M. Sierka, A. Hogekamp, and R. Ahlrichs, J. Chem. Phys. **118**, 9136 (2003).
[53]A. Alvarez-Ibarra and A. M. Köster, J. Chem. Phys. **139**, 024102 (2013).
[54]A. Alvarez-Ibarra and A. M. Köster, Mol. Phys. **113**, 3128 (2015).
[55]K. Ishida, J. Chem. Phys. **98**, 2176 (1993).
[56]N. Flocke and V. Lotrich, J. Comput. Chem. **29**, 2722 (2008).
[57]T. H. Dunning, Jr., J. Chem. Phys. **90**, 1007 (1989).
[58]D. E. Woon and T. H. Dunning, Jr., J. Chem. Phys. **98**, 1358 (1993).
[59]F. Weigend, A. Köhn, and C. Hättig, J. Chem. Phys. **116**, 3175 (2002).
[60]D. S. Hollman, H. F. Schaefer III, and E. F. Valeev, J. Chem. Phys. **142**, 154106 (2015).
[61]MRCC, a quantum chemical program suite written by M. Kállay, Z. Rolik, J. Csontos, I. Ladjánszki, L. Szegedy, B. Ladóczki, G. Samu, K. Petrov, M. Farkas, P. Nagy, D. Mester, and B. Hégely, see also Ref. 71 as well as http://www.mrcc.hu/.
[62]F. Neese, A. Hansen, and D. G. Liakos, J. Chem. Phys. **131**, 064103 (2009).
[63]T. Helgaker, J. Gauss, P. Jørgensen, and J. Olsen, J. Chem. Phys. **106**, 6430 (1997).
[64]J. Weidendorfer, M. Kowarschik, and C. Trinitis, in *Proceedings of the 4th International Conference on Computational Science (ICCS 2004)*, Krakow, Poland, 2004.
[65]J. C. Wheeler, Rocky Mt. J. Math. **4**, 287 (1974).
[66]H. Golub and J. H. Welsch, Math. Comput. **23**, 221 (1969).
[67]B. Doser, D. S. Lambrecht, J. Kussmann, and C. Ochsenfeld, J. Chem. Phys. **130**, 064107 (2009).
[68]M. Schütz, G. Hetzer, and H.-J. Werner, J. Chem. Phys. **111**, 5691 (1999).
[69]H. Eshuis, J. Yarkony, and F. Furche, J. Chem. Phys. **132**, 234114 (2010).
[70]J. Hári, P. Polyák, D. Mester, M. Mitušík, M. Omastová, M. Kállay, and B. Pukánszky, Appl. Clay Sci. **132**, 167 (2016).
[71]Z. Rolik, L. Szegedy, I. Ladjánszki, B. Ladóczki, and M. Kállay, J. Chem. Phys. **139**, 094105 (2013).