# Communication: A reduced scaling J-engine based reformulation of SOS-MP2 using graphics processing units

S. A. Maurer,[1,2] J. Kussmann,[1,2] and C. Ochsenfeld[1,2,a)]
[1]*Chair of Theoretical Chemistry, Department of Chemistry, University of Munich (LMU), Butenandtstr. 7, D-81377 München, Germany*
[2]*Center for Integrated Protein Science (CIPSM) at the Department of Chemistry, University of Munich (LMU), Butenandtstr. 5–13, D-81377 München, Germany*

(Received 15 May 2014; accepted 18 July 2014; published online 6 August 2014)

We present a low-prefactor, cubically scaling scaled-opposite-spin second-order Møller-Plesset perturbation theory (SOS-MP2) method which is highly suitable for massively parallel architectures like graphics processing units (GPU). The scaling is reduced from $\mathcal{O}(N^5)$ to $\mathcal{O}(N^3)$ by a reformulation of the MP2-expression in the atomic orbital basis via Laplace transformation and the resolution-of-the-identity (RI) approximation of the integrals in combination with efficient sparse algebra for the 3-center integral transformation. In contrast to previous works that employ GPUs for post Hartree-Fock calculations, we do not simply employ GPU-based linear algebra libraries to accelerate the conventional algorithm. Instead, our reformulation allows to replace the rate-determining contraction step with a modified J-engine algorithm, that has been proven to be highly efficient on GPUs. Thus, our SOS-MP2 scheme enables us to treat large molecular systems in an accurate and efficient manner on a single GPU-server. © *2014 AIP Publishing LLC*. [http://dx.doi.org/10.1063/1.4891797]

## I. INTRODUCTION

Second-order Møller-Plesset perturbation theory (MP2)[1] is an important method in the field of *ab initio* quantum chemistry, however, the $\mathcal{O}(N^5)$ scaling with molecular size $N$ hampers its applicability to large systems. Therefore, many efficient low- or linear-scaling methods have been developed over the last decades, e.g., see Refs. 2–8. Despite its success, reducing the prefactors is a central issue which is the focus of our present work.

Our method is based on the Laplace scaled-opposite-spin MP2 approach within the resolution-of-the-identity (RI) approximation (SOS-RI-MP2) of Jung *et al.*[9] where the essential idea is a reversed order of summation compared to a conventional RI-MP2 calculation, i.e., the molecular orbital (MO) indices are contracted first before the summations over auxiliary functions are carried out. In this work, we use an atomic orbital-based reformulation which leads to an asymptotic cubic scaling, that is one order of magnitude smaller than in the original MO-based SOS-RI-MP2 approach, and which allows for efficient evaluation on graphics processing units (GPUs). In contrast to the asymptotically cubic scaling *local* SOS-RI-MP2 variant of Jung *et al.*,[10] our method does not rely on a local metric or orbital localization, but uses local atomic orbitals (AOs).

In the last years several methods focusing on using GPUs for post Hartree-Fock (HF) methods have been published[11–15] that employ GPUs only for linear algebra operations. In this work, however, our reformulation for SOS-RI-MP2 allows to replace the computationally demanding contraction step with a modified J-engine algorithm, which has already been proven to be highly efficient on GPUs within self-consistent field (SCF) calculations.[16,17] This integral-direct approach also enables a complete avoidance of disk-IO.

## II. THEORY

The conventional RI-MP2 expression for the opposite-spin MP2 term reads

$$E^{OS}_{RI-MP2} = -\sum_{ijab}\sum_{RSR'S'} \frac{(ia|R)[\mathbf{J}^{-1}]_{RS}(S|jb)(ia|R')[\mathbf{J}^{-1}]_{R'S'}(S'|jb)}{\epsilon_a+\epsilon_b-\epsilon_i-\epsilon_j}, \tag{1}$$

where the matrix $\mathbf{J}$ contains the two-center two-electron integrals in the auxiliary basis ($R$, $S$). Following the Laplace AO-MP2 approach[2–4] this expression can be transformed into the AO basis

$$\begin{aligned} E^{OS}_{RI-AO-MP2} = -\sum_{\alpha}\sum_{\substack{\mu\nu\lambda\sigma\\ \mu'\nu'\lambda'\sigma'}}\sum_{RSR'S'} \underline{P}_{\mu\mu'}\overline{P}_{\nu\nu'}\underline{P}_{\lambda\lambda'}\overline{P}_{\sigma\sigma'} \\ \times (\mu\nu|R)[\mathbf{J}^{-1}]_{RS}(S|\lambda\sigma)(\mu'\nu'|R')[\mathbf{J}^{-1}]_{R'S'} \\ \times (S'|\lambda'\sigma'). \end{aligned} \tag{2}$$

The sum over $\alpha$ corresponds to the quadrature sum of the Laplace transform of the energy denominator where usually 5-6 points have been found to provide good accuracy.[4,18,19] The pseudo-densities $\underline{\mathbf{P}}$ and $\overline{\mathbf{P}}$ are defined as

$$\begin{aligned} \underline{P}_{\mu\nu} &= (\omega_\alpha)^{\frac{1}{4}}\sum_{i}^{\text{occ}} c_{\mu i}e^{\epsilon_i t_\alpha}c_{\nu i}, \\ \overline{P}_{\mu\nu} &= (\omega_\alpha)^{\frac{1}{4}}\sum_{a}^{\text{virt}} c_{\mu a}e^{-\epsilon_a t_\alpha}c_{\nu a}. \end{aligned} \tag{3}$$

a)Electronic mail: Christian.Ochsenfeld@cup.uni-muenchen.de

 

For an efficient evaluation, we define intermediates

$$Z_{RS} = \sum_{\mu\nu\mu'\nu'} (R|\mu'\nu')\underline{P}_{\mu\mu'}\overline{P}_{\nu\nu'}(\mu\nu|S)$$
$$= \sum_{\mu\nu}(R|\underline{\mu}\overline{\nu})(\mu\nu|S), \quad (4)$$

which allow to calculate the OS-MP2 energy via

$$\tilde{\mathbf{Z}} = \mathbf{Z}\mathbf{J}^{-\mathbf{1}} \quad (5)$$

and

$$E^{OS}_{RI-AO-MP2} = -\sum_{\alpha}\sum_{RS}\tilde{Z}_{RS}\tilde{Z}_{SR}. \quad (6)$$

The crucial step is the calculation of the intermediate matrix $\mathbf{Z}$ which can be calculated with cubic scaling cost as shown below. It should be noted that the matrix $\mathbf{Z}$ differs from the matrices $\mathbf{X}$ and $\mathbf{Y}$ used in the methods of Jung *et al.* (see Eqs. (12) and (19) in Ref. 10), where the inverse, or inverse square root, respectively, of the matrix $\mathbf{J}$ is included.

The formation of the $\mathbf{Z}$ matrix requires transformed three-center integrals $(R|\underline{\mu}\overline{\nu})$ which we calculate in an asymptotically quadratic step from the untransformed integrals $(R|\mu\nu)$ using sparse matrix multiplications in the highly efficient block-compressed sparse row (BCSR) format[20] (see Ref. 21 for details of our BCSR implementation). For an efficient transformation, it is furthermore advantageous to perform a Cholesky decomposition of the occupied pseudo-density,

$$\underline{\mathbf{P}} = \underline{\mathbf{L}}\underline{\mathbf{L}}^{T}, \quad (7)$$

which provides a coefficient matrix $\underline{\mathbf{L}}$ for local occupied pseudo-MOs[19] that can be used as an intermediate basis. The transformation starts with a sparse square matrix of integrals for a given auxiliary index $\mathbf{M}^{(R)}$ with $M^{(R)}_{\mu\nu} = (R|\mu\nu)$ where the sparsity reflects the linear-scaling number of significant basis function products. The transformation to the transformed integral block

$$\overline{\underline{\mathbf{M}}}^{(R)} = \underline{\mathbf{P}}\mathbf{M}^{(R)}\overline{\mathbf{P}} \quad (8)$$

TABLE I. Outline of the algorithm for SOS-RI-AO-MP2. Steps (4)–(7) are repeated for every Laplace point.

| | | |
|---|---|---|
| (1) | Calculation of $(R\|\mu\nu)$ | $\mathcal{O}(\mathrm{N}^2)$ |
| (2) | Calculation of $J_{RS} = (R\|S)$ | $\mathcal{O}(\mathrm{N}^2)$ |
| (3) | Calculation of $\mathbf{J}^{-1}$ | $\mathcal{O}(\mathrm{N}^3)$ |
| (4) | Calculation of pseudo-densities | $\mathcal{O}(\mathrm{N}^3)$ |
| (5) | Transformation of $(R\|\mu\nu)$ to $(R\|\underline{\mu}\overline{\nu})$ | $\mathcal{O}(\mathrm{N}^2)$ |
| (6) | Contraction $\sum_{\mu\nu}(R\|\underline{\mu}\overline{\nu})(\mu\nu\|S)$ (on GPU) | $\mathcal{O}(\mathrm{N}^3)$ |
| (7) | Multiplication $\mathbf{Z}\,\mathbf{J}^{-1}$ | $\mathcal{O}(\mathrm{N}^3)$ |
| (8) | Contraction $\sum_{RS}\tilde{Z}_{RS}\tilde{Z}_{SR}$ | $\mathcal{O}(\mathrm{N}^2)$ |

with $\overline{\underline{M}}^{(R)} = (R|\underline{\mu}\overline{\nu})$ is then performed in three consecutive multiplications,

$$\overline{\underline{\mathbf{M}}}^{(R)} = \underline{\mathbf{L}}((\underline{\mathbf{L}}^{T}\mathbf{M}^{(R)})\overline{\mathbf{P}}). \quad (9)$$

It should be noted that the number of occupied Cholesky pseudo-MOs is equal or, in some cases, slightly smaller than the number of occupied MOs (see Ref. 19 for an in-depth discussion) and the dimension of the matrices is therefore largely reduced by the first transformation with the Cholesky coefficients. In any of the three multiplications, two matrix dimensions correspond to the number of atomic orbitals and one corresponds to the number of occupied pseudo-MOs. Further savings in the final transformation step are possible, if one exploits the restriction of the subsequent contraction Eq. (4) to significant basis function products: The final multiplication in Eq. (9) can be restricted to matrix blocks which contain $(\underline{\mu}, \overline{\nu})$ pairs that match significant basis function pairs. All matrices have a sparse structure and the matrix multiplications therefore scale asymptotically linearly with system size for a given auxiliary index which leads to an asymptotically quadratic scaling of the total transformation step.

An outline of the steps in our method is given in Table I and timings on linear alkanes and DNA systems are given in Tables II and III. Steps (1)–(4), (7), and (8) have negligible cost (<10% in all calculations, 2% for the largest DNA system) and the effective scaling for these steps is at most cubically with system size. The cubic scaling steps (3), (4), and (7) are performed using highly optimized linear algebra

TABLE II. Wall times and scaling behavior with respect to the number of basis functions for SOS-RI-AO-MP2 calculations on linear alkanes in a def2-SVP and def2-TZVP basis on a computing node with two Intel Xeon E5-2620 processors (12 CPU cores) and four Nvidia GeForce GTX Titan GPUs. Timings and scaling exponents $[\mathcal{O}(\mathrm{N}^x)]$ are given for the whole calculation as well as selected steps of the algorithm defined in Table I.

| def2 − SVP | # Basis | Total | | Transformation (5) | | Contraction (6) on GPU | | Steps (1)–(4), (7), (8) | |
|---|---|---|---|---|---|---|---|---|---|
| system | functions | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ |
| $C_{20}H_{42}$ | 490 | 98 | … | 51 | … | 39 | … | 8 | … |
| $C_{40}H_{82}$ | 970 | 575 | 2.59 | 374 | 2.92 | 171 | 2.16 | 30 | 1.94 |
| $C_{80}H_{162}$ | 1930 | 2248 | 1.98 | 1051 | 1.50 | 1048 | 2.64 | 149 | 2.33 |
| $C_{160}H_{322}$ | 3850 | 14 134 | 2.66 | 6206 | 2.57 | 7127 | 2.78 | 801 | 2.44 |
| $C_{320}H_{642}$ | 7690 | 95 765 | 2.77 | 33 881 | 2.45 | 56 358 | 2.99 | 5526 | 2.79 |
| def2 − TZVP | # Basis | Total | | Transformation (5) | | Contraction (6) on GPU | | Steps (1)–(4), (7), (8) | |
| system | functions | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ | time [s] | $\mathcal{O}(\mathrm{N}^x)$ |
| $C_{20}H_{42}$ | 872 | 406 | … | 261 | … | 120 | … | 25 | … |
| $C_{40}H_{82}$ | 1732 | 1831 | 2.19 | 932 | 1.85 | 784 | 2.74 | 115 | 2.22 |
| $C_{80}H_{162}$ | 3452 | 11 800 | 2.70 | 5790 | 2.65 | 5445 | 2.81 | 565 | 2.31 |

TABLE III. Wall times and scaling behavior with respect to the number of basis functions for SOS-RI-AO-MP2 calculations on DNA systems in a def2-SVP basis on a computing node with two Intel Xeon E5-2620 processors (12 CPU cores) and four Nvidia GeForce GTX Titan GPUs. Timings and scaling exponents [$\mathcal{O}(N^x)$] are given for the whole calculation as well as selected steps of the algorithm defined in Table I.

| | | Total | | Transformation (5) | | Contraction (6) on GPU | | Steps (1)–(4), (7), (8) | | Reference[a] | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| System | # Basis functions | time [s] | $\mathcal{O}(N^x)$ | time [s] | $\mathcal{O}(N^x)$ | time [s] | $\mathcal{O}(N^x)$ | time [s] | $\mathcal{O}(N^x)$ | time [s] | $\mathcal{O}(N^x)$ |
| $DNA_1$ | 625 | 238 | ... | 143 | ... | 82 | ... | 13 | ... | 438 | ... |
| $DNA_2$ | 1332 | 2096 | 2.88 | 1279 | 2.90 | 721 | 2.87 | 96 | 2.64 | 8290 | 3.88 |
| $DNA_4$ | 2746 | 19 601 | 3.09 | 13 289 | 3.24 | 5750 | 2.87 | 562 | 2.44 | 146 000 | 3.96 |
| $DNA_8$ | 5574 | 185 659 | 3.18 | 126 060 | 3.18 | 55 882 | 3.21 | 3717 | 2.67 | ... | ... |

[a]Reference calculation with conventional SOS-RI-MP2 (1 core@Intel Xeon E5-2620, only serial version available).

routines. The significant steps are the transformation of the three-center integrals, step (5), and the contraction with the untransformed integrals in step (6) which is performed on GPU – details of our algorithm are presented in Sec. III. The scaling of the transformation step (5) is asymptotically quadratic and this behavior is observed as the general trend for large linear alkanes but superimposed with considerable fluctuations due to the blocking in the BCSR format with the chosen block size of around 100 × 100. At the present stage of development, we do not adjust the block structure for the rectangular Cholesky matrices, which leads to some variable overhead in the multiplications which can be significant for smaller systems. For the DNA systems, the transformation matrices are less sparse and the quadratic scaling regime is not reached but an effective cubic scaling is observed.

It should be noted that the scaling of the computational cost given in Table I is based on increased sparsity in large molecular systems with a fixed choice of basis set. More generally, one can also express the scaling behavior in terms of the number of atom-centers ($N_c$) and number of AOs per center ($N_{ao/c}$). The number of auxilliary AOs per center can usually be given in terms of $N_{ao/c}$ multiplied by a constant $c_{aux}$, i.e., as $c_{aux} \times N_{ao/c}$. Thus, we obtain for the two rate-determining steps (5) and (6) a scaling behavior of $c_{occ} \times c_{aux} \times \mathcal{O}(N_c^2 N_{ao/c}^3)$ and $c_{aux}^2 \times \mathcal{O}(N_c^3 N_{ao/c}^4)$, respectively, where either $N_c$ or $N_{ao/c}$ is constant (scaling with basis-set size or system size, respectively). The first expression with the prefactor $c_{occ}$ comes from the use of the Cholesky factor $\underline{\mathbf{L}}$ in the sparse multiplications of step (5). These expressions show the scaling for a fixed system size with increasing basis set size, i.e., $N_{ao/c}^3$ for step (5) and $N_{ao/c}^4$ for step (6). Nevertheless, the asymptotic *scaling with respect to the system size* (i.e., $N_{ao/c}$ is constant) is a quadratic (5) or cubic (6) scaling behavior independent of the choice of the basis set.

In order to provide some measure of comparison for the efficiency of our algorithm, we performed SOS-RI-MP2 calculations with a development version of Q-Chem[22] for the first three DNA-fragments (1 core, Intel Xeon E5-2620@2.00GHz), showing a speed-up of approx. 7.5 for $DNA_4$. Here, it has to be stressed that a fair and balanced comparison of two different algorithms aiming for different computing architectures is very difficult. For larger systems, however, the speed-up of the presented algorithm will be far larger due to the less favorable $\mathcal{O}(N^4)$ of the conventional SOS-RI-MP2 algorithm.

In our present implementation, the untransformed three-center integrals are saved to disk once and read in the transformation step for every Laplace point. An implementation without any I/O can be easily realized by performing steps (3), (4), and (5) in batches of $R$. For each batch of this auxiliary index, all three-center integrals can be kept in memory. The transformation can be performed in memory and in step (5) the untransformed integrals are calculated on-the-fly and directly contracted so that all operations can be performed without any hard disk access. It also has to be stressed that the current implementation is strictly sequential with respect to the single steps given in Table I, i.e., the GPU-kernels block the further execution of the code in order to provide meaningful timings for the single steps of the algorithm. However, the overall computational time can be significantly reduced by executing the GPU-kernels in a non-blocking fashion, so that the contraction step (6) and the transformation step (5) can be executed simultaneously.

## III. DETAILS: GPU-BASED ALGORITHM

For an efficient evaluation of the contraction step (6), a modification of the J-engine[23,24] based GPU-algorithm for the evaluation of the Coulomb matrix in SCF calculations[17] is employed. The algorithm is based on the 1 thread/1 primitive integral (1T1PI) approach and the shell-pair ordering as proposed by Ufimtsev and Martínez.[16,25] To be consistent with this work with respect to the choice of bra and ket distributions and to make the analogy to the Coulomb matrix construction most clear, we rewrite Eq. (4) (for real functions) as

$$Z_{RS} = \sum_{\mu\nu} (S|\mu\nu) \overline{\underline{M}}_{\mu\nu}^{(R)}, \tag{10}$$

with $\overline{\underline{M}}_{\mu\nu}^{(R)} = (R|\underline{\mu}\overline{\nu})$ (cf. Eq. (8)). In contrast to the regular Coulomb matrix construction, we execute a Coulomb-type contraction for any auxiliary index R to calculate one row of $\mathbf{Z}$ using the matrix of transformed three-center integrals $\overline{\underline{\mathbf{M}}}^{(R)}$ in place of the density matrix.

In the regular algorithm to determine the Coulomb matrix for SCF calculations, the ket shell-pairs are contracted with a single density matrix only. Here, however, $N_{aux}$ density-like matrices have to be contracted, so that the use of the conventional algorithm would show the same large prefactor and some adjustments of the algorithm were required.

```
Loop over L_ket
   Loop over batches of R
      Determine max. elements for all shell-pairs of batch: {M̄^(R)_λσ}_max
      Sort ket shell-pairs λσ according to Q_λσ × {M̄^(R)_λσ}_max
      Contract 'densities' M̄^(R) with ket shell-pair data
      Copy ket-data onto GPU(s)
      Loop over L_bra
         Copy (pre-sorted) bra-data onto GPU(s)
         Execute GPU-kernel:  Z_R[p] = Σ_λσ (|p]|λσ) M̄^(R)_λσ
         Copy back primitive Integrals Z_R[p]
         Transform and contract:  Z_R[p] → Z_RS
      EndLoop
   EndLoop
EndLoop
```

FIG. 1. Scheme of the algorithm for the contraction step Eq. (10). If nothing else is indicated, the steps are executed on CPU.

Crucial to the performance of the algorithm is the use of shared memory to hold the intermediate values $J_{[pq]}$, however, this memory is strictly limited on GPUs. Thus, we decided to retain the two-dimensional thread-block setup but we extend the grid by a further dimension to represent the offset to the corresponding indices of the current $R$-batch. Therefore, we strongly reduce the communication between host and GPU-devices. It should be noted that in this case the bra-data does not represent shell-pairs but only the shells of the auxilliary basis.

The complete algorithm is also depicted in Fig. 1. Note that the elements of $(R|\underline{\mu}\overline{\nu})$ are already transformed to the cartesian basis and non-axial normalization coefficients have been incorporated. Furthermore, the elements are sorted according to the $l$-quantum number combination of $\underline{\mu}\overline{\nu}$, so that the batches can be read from disk in a coalesced fashion.

## IV. CONCLUSION

We present a reformulation of the SOS-RI-MP2 algorithm which not only reduces the computational effort to $\mathcal{O}(N^3)$, but also allows for an efficient evaluation of the rate-determining contraction step on GPUs by employing a modified J-engine algorithm. As a – to our knowledge – first reformulation of a post-HF method which is highly suitable for GPUs beyond the mere use of GPU-accelerated linear algebra libraries, our ansatz may also be applicable to other post-HF methods in order to enable an efficient use of massively parallel processors.

## ACKNOWLEDGMENTS

C.O. acknowledges financial support by the "Deutsche Forschungsgemeinschaft" (DFG) for the project Oc35/4-1. Further funding was provided by the DFG via the initiatives SFB 749 "Dynamik und Intermediate molekularer Transformationen" (project C7) and the DFG cluster of excellence EXC 114 "Center for Integrated Protein Science Munich" (CIPSM).

[1] C. Møller and M. S. Plesset, Phys. Rev. **46**, 618 (1934).
[2] J. Almlöf, Chem. Phys. Lett. **181**, 319 (1991).
[3] M. Häser and J. Almlöf, J. Chem. Phys. **96**, 489 (1992).
[4] M. Häser, Theor. Chim. Acta **87**, 147 (1993).
[5] P. Y. Ayala and G. E. Scuseria, J. Chem. Phys. **110**, 3660 (1999).
[6] S. Saebø and P. Pulay, J. Chem. Phys. **115**, 3975 (2001).
[7] H.-J. Werner, F. R. Manby, and P. J. Knowles, J. Chem. Phys. **118**, 8149 (2003).
[8] B. Doser, D. S. Lambrecht, J. Kussmann, and C. Ochsenfeld, J. Chem. Phys. **130**, 064107 (2009).
[9] Y. Jung, R. C. Lochan, A. D. Dutoi, and M. Head-Gordon, J. Chem. Phys. **121**, 9793 (2004).
[10] Y. Jung, Y. Shao, and M. Head-Gordon, J. Comput. Chem. **28**, 1953 (2007).
[11] R. Olivares-Amaya, M. A. Watson, R. G. Edgar, L. Vogt, Y. Shao, and A. Aspuru-Guzik, J. Chem. Theory Comput. **6**, 135 (2010).
[12] E. A. DePrince III and J. R. Hammond, J. Chem. Theory Comput. **7**, 1287 (2011).
[13] W. Ma, S. Krishnamoorthy, O. Villa, and K. Kowalski, J. Chem. Theory Comput. **7**, 1316 (2011).
[14] K. Bhaskaran-Nair, W. Ma, S. Krishnamoorthy, O. Villa, H. J. J. van Dam, E. Aprà, and K. Kowalski, J. Chem. Theory Comput. **9**, 1949 (2013).
[15] M. Del Ben, J. Hutter, and J. VandeVondele, J. Chem. Theory Comput. **9**, 2654 (2013).
[16] I. S. Ufimtsev and T. J. Martinez, J. Chem. Theory Comput. **5**, 1004 (2009).
[17] J. Kussmann and C. Ochsenfeld, J. Chem. Phys. **138**, 134114 (2013).
[18] S. A. Maurer, D. S. Lambrecht, J. Kussmann, and C. Ochsenfeld, J. Chem. Phys. **138**, 014101 (2013).
[19] S. Maurer, L. Clin, and C. Ochsenfeld, J. Chem. Phys. **140**, 224112 (2014).
[20] F. G. Gustavson, ACM Trans. Math. Softw. **4**, 250 (1978).
[21] J. Kussmann and C. Ochsenfeld, J. Chem. Phys. **127**, 054103 (2007).
[22] Development version of the Q-Chem program package, http://www.q-chem.com.
[23] C. A. White and M. Head-Gordon, J. Chem. Phys. **104**, 2620 (1996).
[24] G. R. Ahmadi and J. Almlöf, Chem. Phys. Lett. **246**, 364 (1995).
[25] I. S. Ufimtsev and T. J. Martínez, J. Chem. Theory Comput. **4**, 222 (2008).