

Reduced-Cost Sparsity-Exploiting Algorithm for Solving Coupled-Cluster Equations

Jiri Brabec,^{*[a]} Chao Yang,^[a] Evgeny Epifanovsky,^[b,c] Anna I. Krylov,^[b] and Esmond Ng^[a]

We present an algorithm for reducing the computational work involved in coupled-cluster (CC) calculations by sparsifying the amplitude correction within a CC amplitude update procedure. We provide a theoretical justification for this approach, which is based on the convergence theory of inexact Newton iterations. We demonstrate by numerical examples that, in the sim-

plest case of the CCD equations, we can sparsify the amplitude correction by setting, on average, roughly 90% non-zero elements to zeros without a major effect on the convergence of the inexact Newton iterations.

DOI: 10.1002/jcc.24293

Introduction

The coupled-cluster (CC) theory^[1] is the most successful approach for treating dynamic electron correlation in quantum chemistry.^[2–4] The CC wave function is generated by the exponential expansion of the cluster operator; thus, compared with an equivalent linear (configuration interaction) ansatz, it includes configurations of higher excitation levels appearing as products of lower-order cluster operators. Among the advantages of the CC hierarchy of approximations are size-extensivity, orbital-rotation invariance within either occupied or virtual space, and a systematic convergence towards the exact solution in the full configuration interaction limit.^[2] The single-reference CC formalism can be easily extended to tackle various multiconfigurational wave functions using the EOM-CC approach.^[5–8] CC theory has also been combined with a multi-reference ansatz giving rise to multireference CC methods.^[3,9–14]

Unfortunately, the size of systems amenable to CC treatment (or to other many-body methods) is limited by the polynomial scaling of the CC methods. In the lowest-level CC models,^[4] the CC operator is truncated at the level of double excitations giving rise to the CCD (coupled-cluster doubles) and CCSD (coupled-cluster with single and double substitutions) methods. Thus, the number of unknown variables (amplitudes) is on the order of $O(n_o^2 n_v^2)$ and the computational complexity scales as $O(n_o^2 n_v^4)$, where n_o and n_v are the number of the occupied and virtual orbitals, respectively (the separation between the virtual and occupied subspaces is determined by the reference determinant). However, a large fraction of the amplitudes is known to be small; these amplitudes give negligible contributions to the total correlation energy. If efficiently exploited, the sparsity may lead to a linear-scaling implementation of a correlated method. In other words, the number of cluster amplitudes can be significantly reduced if sparsity and linear dependencies are taken into account. Many approaches attempt to exploit the sparsity of the amplitude tensor or to construct its compact representation through some type of low-rank tensor decomposition techniques. These approaches

often exploit the locality of correlation^[15–34] or use mathematical decomposition techniques such as the singular value (and high-order singular value) decomposition, density fitting, Cholesky decomposition, canonical product decomposition, frozen natural orbitals, etc to represent the tensors in a compact form.^[35–58] This article does not aim at formulating a local or low-scaling CC method; rather, it presents a reduced-cost algorithm that takes advantage of sparsity during solving the CC equations, whereas the final converged amplitudes are not truncated giving rise to the exact (for a given CC method) correlation energy.

We present a numerical method for solving the CC amplitude equations, which can be written in the general form as

$$R(T)=0, \quad (1)$$

where the T is an amplitude tensor and R is a (nonlinear) function of T . Our method uses fewer floating point operations than the standard inexact Newton method. The cost reduction results from exploiting sparsity in the successive corrections to the approximate CC amplitudes, rather than the amplitudes themselves. The sparsity allows us to reduce the cost of tensor contraction required to evaluate $R(T)$ in each inexact Newton step (i.e., CC amplitudes' update equation).

The proposed approach is close in spirit to the recursive formula for constructing the Fock matrix in self-consistent field

[a] J. Brabec, C. Yang, E. Ng
Computational Research Division, Lawrence Berkeley National Laboratory,
Berkeley, California 94720
E-mail: jiri.brabec@jh-inst.cas.cz

[b] E. Epifanovsky, A. I. Krylov
Department of Chemistry, University of Southern California, Los Angeles,
California 90089-0482

[c] E. Epifanovsky
Q-Chem Inc, Suite 105 Pleasanton, California 94588

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences.

© 2016 Wiley Periodicals, Inc.

(SCF) calculations (incremental Fock build) by Almlöf et al.,^[59] which, barring numerical instabilities, leads to the exact SCF solution. The technique has been successfully employed in direct SCF calculations, and its numerical properties have been studied.^[60] Using sparsified corrections to solve the CC equations has several algorithmic and numerical implications that we present in this article.

For the sake of simplicity, we focus on the CCD equation ($T \equiv T_2$), which is a 'second-order' nonlinear equation in T . T_2 is a fourth-order tensor of dimension $n_o \times n_o \times n_v \times n_v$. In the next section, we establish the notations used in the rest of the article and outline the basic strategy for keeping the amplitude correction sparse. In section "The Implementation," we discuss practical issues that are important for developing an efficient implementation of the algorithm. Computational examples presented in section "Numerical Examples" illustrate the potential cost savings that can be achieved by our approach.

Theory

Projected CC equations

The CC ansatz has the following form:

$$|\Psi\rangle = e^T |\Phi_0\rangle, \quad (2)$$

where $|\Phi_0\rangle$ is the reference wave function (typically, the Hartree-Fock Slater determinant) and T is the cluster operator defined as $T = T_1 + T_2 + T_3 + \dots$ (each T_n consists of a linear combination of all possible n -tuple excitation operators). The coefficients of the linear combination are called the amplitudes. Inserting the CC ansatz into the Schrödinger equation and multiplying from the left by e^{-T} yields

$$e^{-T} H e^T |\Phi_0\rangle = E |\Phi_0\rangle, \quad (3)$$

where $\bar{H} \equiv e^{-T} H e^T$ is the similarity transformed Hamiltonian, which is not Hermitian. To obtain enough equations to determine the energy and amplitudes, eq. (3) is projected into the Hartree-Fock determinant and into the manifold of excited determinants (of the same excitation level as the truncation level in T). Thus, the CC energy is

$$\langle \Phi_0 | \bar{H} | \Phi_0 \rangle = E, \quad (4)$$

and the projected amplitude equations are

$$\langle \Phi_\mu | \bar{H} | \Phi_0 \rangle = E \langle \Phi_\mu | \Phi_0 \rangle = 0, \quad (5)$$

where Φ_μ represents all singly, doubly, or higher excited determinants with respect to Φ_0 .

The inexact Newton algorithm

The simplest iterative scheme for solving eq. (5) is Newton's method, which produces a sequence of approximate amplitudes $T^{(k)}$ by the following updating formula:

$$T^{(k+1)} = T^{(k)} + \Delta^{(k)}, \quad (6)$$

where the Newton correction amplitude $\Delta^{(k)}$ is obtained from

$$\Delta^{(k)} = -J^{-1}(T^{(k)}) R(T^{(k)}), \quad (7)$$

with $J(T^{(k)})$ being the Jacobian of the nonlinear function R evaluated at $T^{(k)}$.

Evaluating and inverting exact $J(T^{(k)})$ is expensive. However, the Jacobian matrix is known to be diagonally dominant in the canonical representation. Therefore, in practice, one often replaces $J(T^{(k)})$ with approximate diagonal \hat{J} . The diagonal elements of \hat{J} are chosen to be orbital energy differences. As a result, the updating formula for $T^{(k)}$ becomes

$$T^{(k+1)} = T^{(k)} - \hat{J}^{-1} R(T^{(k)}). \quad (8)$$

An algorithm based on the above updating formula is called an inexact Newton scheme, which is summarized in Algorithm 1.

Algorithm 1: The inexact Newton algorithm for solving CC amplitude equations.

Input: An initial approximation of $T^{(0)}$, a convergence tolerance tol ;

Output: Approximate T that satisfies $\|R(T)\| \leq tol$.

1: $k \leftarrow 0$;

2: **while** $\|R(T^{(k)})\| > tol$ **do**

3: Evaluate residuals $R(T^{(k)})$;

4: Construct a correction $\hat{\Delta}^{(k)} = -\hat{J}^{-1} R(T^{(k)})$;

5: $T^{(k+1)} \leftarrow T^{(k)} + \hat{\Delta}^{(k)}$;

6: $k \leftarrow k + 1$;

7: **end while**

The new algorithm

An iteration based on eq. (8) can be regarded as an inexact Newton method because we may view the update $\hat{\Delta}^{(k)} = -\hat{J}^{-1} R(T^{(k)})$ as an approximate solution to the Newton correction equation:

$$J(T^{(k)}) \hat{\Delta}^{(k)} = -R(T^{(k)}).$$

It is well known^[61,62] that if $\hat{\Delta}^{(k)}$ satisfies the condition

$$\|J(T^{(k)}) \hat{\Delta}^{(k)} + R(T^{(k)})\| \leq \eta_k \|R(T^{(k)})\| \quad (9)$$

for a sequence of what is often known as the "forcing" parameters $\eta_k < 1$, then $T^{(k)}$ is guaranteed to converge to the solution of eq. (1), provided $T^{(0)}$ is sufficiently close to the solution.

Although the condition listed in eq. (9) is generally difficult to verify because computing the true Jacobian $J(T^{(k)})$ is prohibitively expensive, it is often the case that $J(T^{(k)})$ is diagonally dominant and its diagonal is identical to that of \hat{J} . It follows that we have $J(T^{(k)})\hat{J}^{-1} = I + E_k$ for some matrix E_k that is relatively small in magnitude. As a result, we have

$$\begin{aligned} \|J(T^{(k)})\hat{\Delta}^{(k)} + R(T^{(k)})\| &= \|(I + E_k)R(T^{(k)}) + R(T^{(k)})\| = \|E_k R(T^{(k)})\| \\ &\leq \|E_k\| \|R(T^{(k)})\|. \end{aligned}$$

If $\|E_k\| \leq \eta_k < 1$, then eq. (9) holds.

Note that in the first few iterations, $\|R(T^{(k)})\|$ is likely to be relatively large. Hence, it is relatively easy to obtain an inexact correction $\hat{\Delta}^{(k)}$ that satisfies eq. (9). There are likely many ways to modify $\hat{\Delta}^{(k)}$ without affecting the convergence of the inexact Newton iteration. In this paper, we propose to modify $\hat{\Delta}^{(k)}$ by setting the elements that are "small" in magnitude to zero. This can be viewed as a "sparsification" process. By sparsifying $\hat{\Delta}^{(k)}$ and keeping the number of nonzero elements in $\hat{\Delta}^{(k)}$ small, we can reduce the tensor contraction cost for evaluating $R(T^{(k)})$ if an efficient sparse tensor contraction software is available.

To sparsify $\hat{\Delta}^{(k)}$, we define a sequence of dynamically adjusted thresholds τ_k , and set the elements of $\hat{\Delta}^{(k)}$ that are less than τ_k in absolute value to zeros. These zeros do not need to be stored if a sparse representation of the $\hat{\Delta}^{(k)}$ tensor is implemented. By setting elements of $\hat{\Delta}^{(k)}$ to zeros, we effectively introduce a small perturbation W in $\hat{\Delta}^{(k)}$. One can rewrite $\hat{\Delta}^{(k)}$ as

$$\hat{\Delta}^{(k)} = -\hat{J}^{-1}R(T^{(k)}) + W,$$

where W contains elements of $\hat{\Delta}^{(k)}$ that are to be dropped. If W can be chosen to satisfy

$$\|W\| < \alpha \|R(T^{(k)})\| \quad (10)$$

for some constant $\alpha \ll 1$, we have

$$\begin{aligned} \|J(T^{(k)})\hat{\Delta}^{(k)} + R(T^{(k)})\| &= \|-E_k R(T^{(k)}) + J(T^{(k)})W\| \\ &\leq (\|E_k\| + \alpha \|J(T^{(k)})\|) \|R(T^{(k)})\|. \end{aligned} \quad (11)$$

If $\|E_k\| + \alpha \|J(T^{(k)})\| < 1$, then one can expect that the sparsified inexact Newton algorithm will still converge. Note that in order to make $\|E_k\| + \alpha \|J(T^{(k)})\|$ not too much larger than $\|E_k\|$, α should be chosen to be much smaller than $1/\|J(T^{(k)})\|$. However, since the choice of α and $\|R(T^{(k)})\|$ will determine the value of the sparsification threshold (τ_k) in the k th iteration, it should not be too small to prevent $\hat{\Delta}^{(k)}$ from being sparsified.

As $T^{(k)}$ converges to the solution of eq. (1), $\|R(T^{(k)})\|$ decreases. Consequently, for a fixed α , τ_k should also be gradually reduced, to ensure that eq. (10) is satisfied. Because $\|\hat{\Delta}^{(k)}\|$

generally becomes smaller also, it is still possible to identify elements in $\hat{\Delta}^{(k)}$ that can be set to zero without violating eq. (10).

To facilitate sparse tensor contraction, we need to rewrite the expression for $R(T^{(k+1)})$ by substituting $T^{(k+1)}$ with

$$T^{(k+1)} = T^{(k)} + \hat{\Delta}^{(k)} \quad (12)$$

and regrouping terms. The terms that do not contain $\hat{\Delta}^{(k)}$ do not need to be recomputed. They can be reused from the previous iteration. Only the terms that contain $\hat{\Delta}^{(k)}$ need to be computed. However, these are the terms that can be computed by using sparse contraction techniques. To illustrate this, let us assume that $R(T^{(k)})$ contains second-order terms in the form of TVT (we use the notation TV to denote the contraction of tensors T and V). We can rewrite $T^{(k+1)}V$ as

$$\begin{aligned} (T^{(k)} + \hat{\Delta}^{(k)})V(T^{(k)} + \hat{\Delta}^{(k)}) &= T^{(k)}VT^{(k)} + T^{(k)}V\hat{\Delta}^{(k)} + \hat{\Delta}^{(k)}VT^{(k)} \\ &\quad + \hat{\Delta}^{(k)}V\hat{\Delta}^{(k)}. \end{aligned} \quad (13)$$

The intermediate terms, $T^{(k)}VT^{(k)}$, $T^{(k)}V$, and $VT^{(k)}$, can be saved from the previous iteration and therefore do not need to be recomputed. Once the evaluation of eq. (13) is completed, we overwrite $T^{(k)}VT^{(k)}$ with this new product. Likewise, $T^{(k)}V$ and $VT^{(k)}$ are overwritten by $T^{(k)} + \hat{\Delta}^{(k)}$ and $VT^{(k)} + V\hat{\Delta}^{(k)}$ once $\hat{\Delta}^{(k)}$ and $V\hat{\Delta}^{(k)}$ are computed.

Algorithm 2: Sparse correction update procedure for the CCD amplitude equations.

Input: An initial approximation of $T^{(0)}$, a convergence tolerance tol ;

Output: Approximate T that satisfies $\|R(T)\| \leq tol$.

- 1: $k = 0$;
 - 2: **while** $\|R(T^{(k)})\| > tol$ **do**
 - 3: Calculate a new threshold τ from $\|R(T^{(k)})\|$;
 - 4: Construct a sparse correction $\hat{\Delta}^{(k)}$ by thresholding $|r_{ij}^{ab}| > \tau$, where $r_{ij}^{ab} \in \|R(T^{(k)})\|$ and multiplying by \hat{J}^{-1} ;
 - 5: Calculate all contractions with sparse $\hat{\Delta}^{(k)}$.
 - 6: Update intermediates. Updated l_{ij}^{ab} corresponds to $R^{(k+1)}$.
 - 7: $T^{(k+1)} \leftarrow T^{(k)} + \hat{\Delta}^{(k)}$;
 - 8: $k \leftarrow k + 1$;
 - 9: **end while**
-

The Implementation

Tensor contraction and intermediates

The programmable CC amplitude equations contain many terms that require high-order tensor contractions (see, for example, Refs. [58,63]). The CCD equations are of the simplest type in which only double excitation are considered. Using the standard quantum-chemistry notation, we can write the CCD amplitude equations as follows:

$$\begin{aligned}
r_{ij}^{ab} = & \langle ij||ab \rangle - P(ij) \sum_k t_{ik}^{ab} f_j^{fk} + P(ab) \sum_c t_{ij}^{ac} f_c^{bc} \\
& + \frac{1}{2} \sum_{kl} t_{kl}^{ab} \langle kl||ij \rangle + \frac{1}{2} \sum_{cd} t_{ij}^{cd} \langle ab||cd \rangle \\
& + P(ij)P(ab) \sum_{kc} t_{ik}^{ac} \langle kb||cj \rangle \\
& + \frac{1}{4} \sum_{klcd} t_{ij}^{cd} t_{kl}^{ab} \langle kl||cd \rangle \\
- P(ij) \frac{1}{2} \sum_{klcd} t_{ik}^{ab} t_{jl}^{cd} \langle kl||cd \rangle - & P(ab) \frac{1}{2} \sum_{klcd} t_{ik}^{ac} t_{jl}^{bd} \langle kl||cd \rangle \\
& + P(ij)P(ab) \frac{1}{2} \sum_{klcd} t_{ik}^{ac} t_{jl}^{db} \langle kl||cd \rangle,
\end{aligned} \quad (14)$$

where $P(pq)A^{pq} = A^{pq} - A^{qp}$ is an anti-symmetrization operator, f denotes the matrix of the one-electron Fock operator, and $\langle ij||ab \rangle$ represents an antisymmetrized two-electron integrals (The reference determines the separation of orbital space into the occupied and virtual subspaces. Here we use indexes i, j, k, \dots and a, b, c, \dots to denote the orbitals from the two subspaces. To denote orbitals that can be either occupied or virtual, letters p, q, r, s, \dots will be used. f_q^p and $\langle pq||rs \rangle$ denote the matrix elements of the Fock operator and antisymmetrized two-electron repulsion integrals, respectively).

Separating the previous iteration amplitude $T^{(k)}$ and new correction $\hat{\Delta}(T^{(k)})$, inserting eq. (12) into (14), and using $v_{pq}^{rs} = \langle rs||pq \rangle$, the linear terms on the right hand side of eq. (14) can be rewritten as

$$\begin{aligned}
P(ab) t_{ij}^{ac} f_c^{fb} - P(ij) t_{ik}^{ab} f_j^{fk} + \frac{1}{2} t_{kl}^{ab} v_{ij}^{kl} + \frac{1}{2} t_{ij}^{cd} v_{cd}^{ab} + P(ij)P(ab) t_{ik}^{ac} v_{cj}^{kb} \\
+ P(ab) \hat{\Delta}_{ij}^{ac} f_c^{fb} - P(ij) \hat{\Delta}_{ik}^{ab} f_j^{fk} + \frac{1}{2} \hat{\Delta}_{kl}^{ab} v_{ij}^{kl} + \frac{1}{2} \hat{\Delta}_{ij}^{cd} v_{cd}^{ab} + P(ij)P(ab) \hat{\Delta}_{ik}^{ac} v_{cj}^{kb},
\end{aligned} \quad (15)$$

while quadratic terms assume the following form

$$\begin{aligned}
\frac{1}{4} t_{ij}^{cd} t_{kl}^{ab} v_{cd}^{kl} - P(ij) \frac{1}{2} t_{ik}^{ab} t_{jl}^{cd} v_{cd}^{kl} - P(ab) \frac{1}{2} t_{ij}^{ac} t_{kl}^{bd} v_{cd}^{kl} \\
+ P(ij)P(ab) \frac{1}{2} t_{ik}^{ac} t_{jl}^{db} v_{cd}^{kl} \\
+ \frac{1}{4} \hat{\Delta}_{ij}^{cd} \hat{\Delta}_{kl}^{ab} v_{cd}^{kl} - P(ij) \frac{1}{2} \hat{\Delta}_{ik}^{ab} \hat{\Delta}_{jl}^{cd} v_{cd}^{kl} - P(ab) \frac{1}{2} \hat{\Delta}_{ij}^{ac} \hat{\Delta}_{kl}^{bd} v_{cd}^{kl} \\
+ P(ij)P(ab) \frac{1}{2} \hat{\Delta}_{ik}^{ac} \hat{\Delta}_{jl}^{db} v_{cd}^{kl} \\
+ \frac{1}{4} t_{ij}^{cd} \hat{\Delta}_{kl}^{ab} v_{cd}^{kl} - P(ij) \frac{1}{2} t_{ik}^{ab} \hat{\Delta}_{jl}^{cd} v_{cd}^{kl} - P(ab) \frac{1}{2} t_{ij}^{ac} \hat{\Delta}_{kl}^{bd} v_{cd}^{kl} \\
+ P(ij)P(ab) \frac{1}{2} t_{ik}^{ac} \hat{\Delta}_{jl}^{db} v_{cd}^{kl}.
\end{aligned} \quad (16)$$

By standard convention, the repeated indices (either the subscripts or the superscripts) in eqs. (15) and (16) are the indices

Table 1. The computational scaling and memory requirements for keeping and updating intermediates marked by bold font in eq. (16) and a contraction scheme, which does not require these intermediates.

Contraction	Intermediate	Memory	Scaling
$\hat{\Delta}_{ij}^{cd} (v_{cd}^{kl} t_{kl}^{ab})$	$I_{cd}^{ab} \leftarrow I_{cd}^{ab} + v_{cd}^{kl} \hat{\Delta}_{kl}^{ab}$	n_v^4	$n_o^2 n_v^4$
$(\hat{\Delta}_{ij}^{cd} v_{cd}^{kl}) t_{kl}^{ab}$	$\hat{\Delta}_{ij}^{cd} v_{cd}^{kl}$	n_o^4	$n_o^4 n_v^2$
$(t_{ik}^{ab} v_{cd}^{kl}) \hat{\Delta}_{jl}^{cd}$	$I_{icd}^{abl} \leftarrow I_{icd}^{abl} + \hat{\Delta}_{ij}^{ab} v_{cd}^{kl}$	$n_o^2 n_v^4$	$n_o^3 n_v^4$
$t_{ik}^{ab} (v_{cd}^{kl} \hat{\Delta}_{jl}^{cd})$	$v_{cd}^{kl} \hat{\Delta}_{jl}^{cd}$	n_o^2	$n_o^2 n_v^2$
$(t_{ij}^{ad} v_{cd}^{kl}) \hat{\Delta}_{kl}^{bd}$	$I_{ijd}^{akl} \leftarrow I_{ijd}^{akl} + \hat{\Delta}_{ij}^{ac} v_{cd}^{kl}$	$n_o^4 n_v^2$	$n_o^4 n_v^3$
$t_{ij}^{ad} (v_{cd}^{kl} \hat{\Delta}_{kl}^{bd})$	$v_{cd}^{kl} \hat{\Delta}_{kl}^{bd}$	n_v^2	$n_o^2 n_v^3$

to be contracted. The remaining indices are the indices of the resulting tensor product. For example, in $t_{ik}^{ac} v_{cj}^{kb}$, k , and c are indices to be contracted and the resulting tensor is indexed by i, a, b and j ; the cost of the contraction is $O(N^6)$.

At the k th iteration, the zero-order term $\langle ij||ab \rangle$, as well as all linear terms that involve the contractions of integrals v and the previous amplitudes t , are assumed to have already been evaluated in the $(k-1)$ th iteration, so they are stored as the I_{ij}^{ab} intermediate. Therefore, only the contractions with $\hat{\Delta}$ are performed and the result is added to the intermediate.

The update of the quadratic terms needs to be performed with care. Ideally, we would like to keep the contracted products of the integral and the previous amplitude as intermediates in the second and third rows of eq. (16) so that we only need to contract the sparse correction $\hat{\Delta}$ with the intermediates. However, for some of the quadratic terms in (16), this approach would result in a factorization that involves expensive contractions and a higher storage requirement for the intermediates.^[15,64] For example, if we were to keep $I_{cd}^{ab} \equiv t_{kl}^{ab} v_{cd}^{kl}$ as an intermediate in the first term of the second row of eq. (16), we would need to update I_{cd}^{ab} by performing

$$I_{cd}^{ab} \leftarrow I_{cd}^{ab} + t_{kl}^{ab} \hat{\Delta}_{cd}^{kl}, \quad (17)$$

after the contraction of $\hat{\Delta}_{ij}^{cd}$ with the previous I_{cd}^{ab} is completed. However, the asymptotic complexity of the contraction $t_{kl}^{ab} \hat{\Delta}_{cd}^{kl}$ in eq. (17) is $O(n_o^2 n_v^4)$, which is less favorable than the overall $O(n_o^4 n_v^2)$ complexity resulting from contracting $\hat{\Delta}_{ij}^{cd}$ with v_{cd}^{kl} first followed by another contraction of the contracted product with t_{kl}^{ab} when $n_v \gg n_o$, even though the approach based on forming and updating I_{cd}^{ab} involves only sparse contractions. In terms of memory requirement, if we were to keep the product of t_{ik}^{ab} and v_{cd}^{kl} as an intermediate in the second term of the third of row in eq. (16), we would need $O(n_o^2 n_v^4)$ storage, which is excessive compared with the amplitude tensor of the size $n_o^2 n_v^2$.

We highlight the terms in eq. (16) that require special attention by using bold font. For these terms, we do not keep the product of the integrals and the previous amplitudes as an intermediate. The less favorable complexity and storage requirement for keeping such intermediates is summarized and compared in Table 1 with the standard approach, which contracts the sparse correction with the integrals first followed by another dense contracts of the product with the amplitudes.

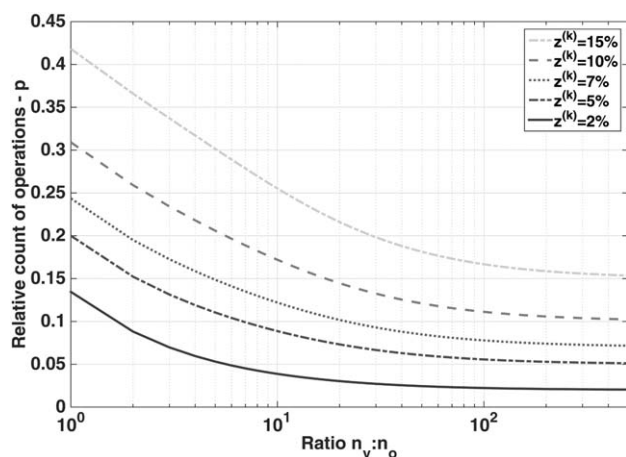


Figure 1. The ratio of $c^{(k)}$ over c as a function of $z^{(k)}$ and n_v when n_o is set to 50.

We now give an estimate of possible savings in terms of the number of operations when the correction amplitude is sparse. Only $O(N^6)$ terms are taken into account and the restricted Hartree-Fock spin-orbitals are assumed. We use $z^{(k)}$ to denote the sparsity of $\hat{\Delta}^{(k)}$, which is the percentage of nonzero elements remaining in $\hat{\Delta}^{(k)}$ after the sparsification. The number of operations required in all contractions used by our algorithm in step k is proportional to

$$c^{(k)} = 3z^{(k)} \cdot n_o^4 n_v^2 + 20z^{(k)} \cdot n_o^3 n_v^3 + z^{(k)} \cdot n_o^2 n_v^4 + n_o^4 n_v^2. \quad (18)$$

If we denote by c the number of contractions in the original dense algorithm, it is easy to verify that

$$c = 2n_o^4 n_v^2 + 8n_o^3 n_v^3 + n_o^2 n_v^4. \quad (19)$$

Figure 1 shows the ratio of $c^{(k)}$ over c for different $z^{(k)}$ values and different n_v/n_o ratios. As follows from eq. (18), a larger reduction of operations can be achieved when $n_v \gg n_o$ with the asymptotic limit $z^{(k)} O(n_o^2 n_v^4)$.

Sparsifying the correction

To determine which elements in $\hat{\Delta}^{(k)}$ can be set to zero to sparsify the correction, we need to first choose the parameter α in eq. (10). Ideally, α should be chosen to satisfy

$$\alpha < \frac{\gamma}{\|J(T^{(k)})\|},$$

for some small fudge factor $\gamma < 1$ (e.g., $\gamma = 0.1$). Since $J(T^{(k)})$ is not known, we take its diagonal part

$$\max_{i,j,a,b} = \epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j$$

as an approximate $\|J(T^{(k)})\|$. Once α is chosen, we can set τ_k to $\alpha \|R(T^{(k)})\|$ as suggested in eq. (10).

If we treat all tensors as vectors and choose $\|\cdot\|$ to be the ∞ -norm (which is given by the element with the largest abso-

lute value) in eq. (10), we can simply sparsify $\hat{\Delta}^{(k)}$ by setting elements whose absolute values are smaller than τ_k to zero.

If we choose $\|\cdot\|$ to be the vector 2-norm, then the sparsification process can be carried out by sorting the absolute values of all elements of $R(T^{(k)})$ and summing the sorted elements in an increasing order until

$$\sqrt{\sum_{i=1}^{M+1} r_i^2} > \tau_k. \quad (20)$$

for the smallest integer M , where $r_1 \leq r_2 \leq \dots$ are the sorted elements of $|R(T^{(k)})|$. The first M elements of the sorted correction are set to zero. To reduce the cost of sorting, we can first determine a threshold $\tau_k/\sqrt{n} \leq \hat{\tau} \leq \tau_k$, and check if (20) is satisfied for the elements whose absolute values are below the threshold $\hat{\tau}$. This threshold can be adjusted until eq. (20) is satisfied for as many elements in the correction tensor as possible.

Instead of using eq. (20), an alternative method for sparsifying the correction tensor $\hat{\Delta}(T^{(k)})$ is to simply set a fixed number of elements of $\hat{\Delta}(T^{(k)})$ with the smallest absolute values to zeros. However, this scheme may lead to an increase in the number of the inexact Newton iterations required to reach convergence, as shown in the next section.

The sparsification approach introduced here for the CCD model can be easily extended for other CC models. For an arbitrary T_n , where n is the excitation level, we can rewrite eq. (12) as n -specific equation

$$T_n^{(k+1)} = T_n^{(k)} + \hat{\Delta}_n^{(k)}, \quad (21)$$

where $\hat{\Delta}_n^{(k)}$ is sparsified correction using the Algorithm 2 with threshold τ_n calculated from $\|R_n(T^{(k)})\|$. All T_n in amplitude equations have to be replaced by eq. (21) and we can define intermediates the same way as described in the previous section. If n is the maximal excitation level, the memory requirements for storing the intermediates scales as N^n , where N is the size of the system.

Numerical Examples

We now illustrate the effectiveness of using sparse correction for solving the amplitude equations using a pilot CCD code. We perform numerical tests for seven benchmark systems. They range from small molecules (LiF and O₃) to larger systems—pentane, alanine, asparagine, dodecane, and porphine (see Supporting Information for geometries). Alanine and asparagine are chosen as medium-size nonsymmetric systems, dodecane and porphine represent large molecules. Our code is written in MATLAB (MATLAB R2014b, Mathworks, Natick, MA). The correctness of the implementation is validated against the CCD code in NWChem.^[65] Our code uses the Fock matrix, two-electron integrals and orbital energies that are dumped from NWChem. The CCD equations are solved by performing inexact Newton iterations; DIIS convergence acceleration algorithm was not used.

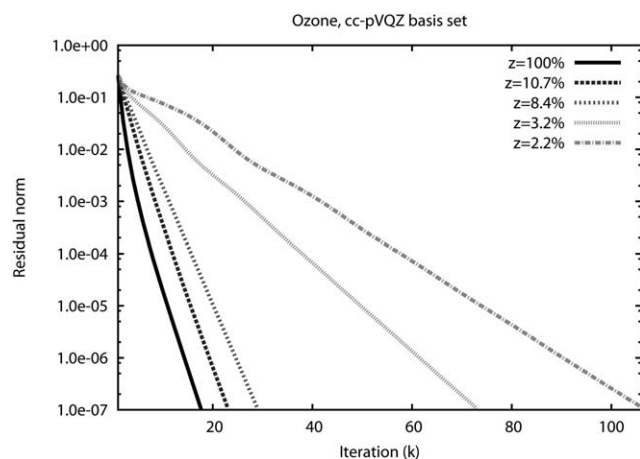


Figure 2. Convergence of CCD equations for O_3 using various levels of sparsification measured by the average sparsity of the amplitude correction z defined by eq. (23). The cc-pVQZ basis set was employed.

The calculations are based on the restricted Hartree-Fock reference and spatial symmetry is taken into account. That is, the code works only with nonzero amplitudes such that all amplitudes which are zero due to symmetry are ignored. Permutational symmetry is taken into account when counting nonzero elements, that is, all equivalent amplitudes due to permutation symmetry are counted only once. All electrons are active in our calculations.

The initial guess of the amplitude is set to zero. We set the parameter α in (10) to $0.1/\max_{i,j,a,b}(\epsilon_a + \epsilon_b - \epsilon_i - \epsilon_j)$, and use the infinity norm to determine which elements in $\hat{\Delta}(T^{(k)})$ are set to zero. We set the convergence tolerance to $tol = 10^{-7}$, that is, we terminate the iterations when the residual norm $\|R(T^{(k)})\|$ is $< 10^{-7}$.

Figures (2 and 4), and 5 show the change of residual norms as a function of the iteration number k for ozone (using the cc-pVQZ basis set), dodecane (with the 6-311G basis set), and dodecane with the cc-pVDZ basis set, respectively. The residual history associated with the original inexact Newton iteration in

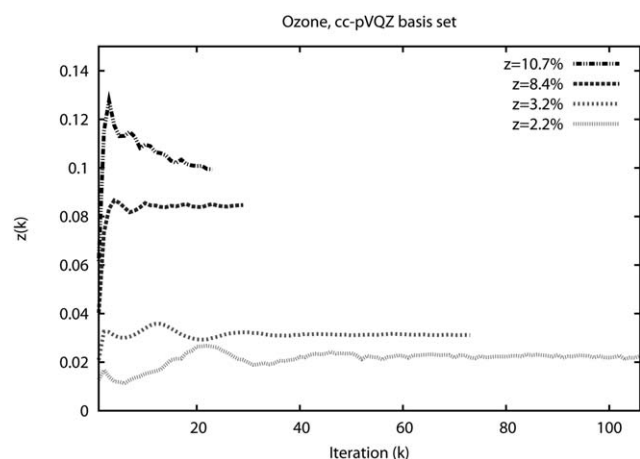


Figure 3. The variation of the sparsity in the amplitude correction with respect to the CCD iteration number k measured by $z(k)$ for the O_3 molecule. Each curve corresponds to a curve labeled by the same average sparsity z in Figure 2.

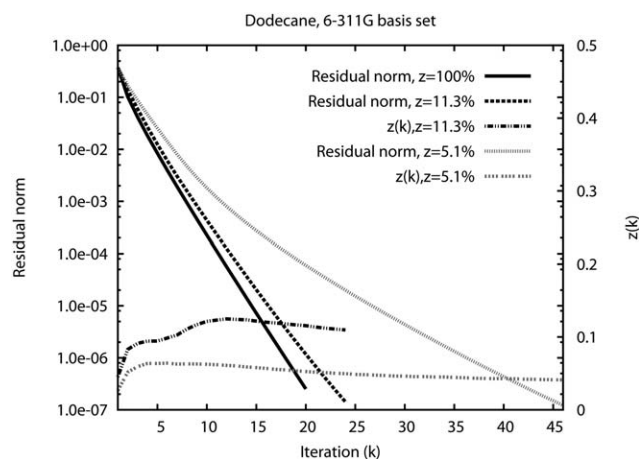


Figure 4. Convergence of CCD equations for dodecane at the equilibrium geometry using various levels of sparsification z . The 6-311G basis set was employed.

which no elements in $\hat{\Delta}(T^{(k)})$ were set to zero is shown as the blue curve in these figures for comparison.

To quantify the degree of cost reduction in each iteration, we track the percentage of nonzero elements retained in $\hat{\Delta}(T^{(k)})$ after the sparsification process is performed and denote the percentage by $z(k)$. An element of $\hat{\Delta}(T^{(k)})$, denoted by d_i , is considered to be numerically nonzero if

$$|d_i| > 10^{-16} |d_{\max}|, \quad (22)$$

where d_{\max} is the element of $\hat{\Delta}(T^{(k)})$ with the largest absolute value. Figure 3 shows that the variation of $z(k)$ with respect to k is relatively small after the first few iterations.

We define the average percentage of nonzero elements in the sparsified $\hat{\Delta}(T^{(k)})$ as

$$z = \frac{\sum_{k=1}^K z(k)}{K}, \quad (23)$$

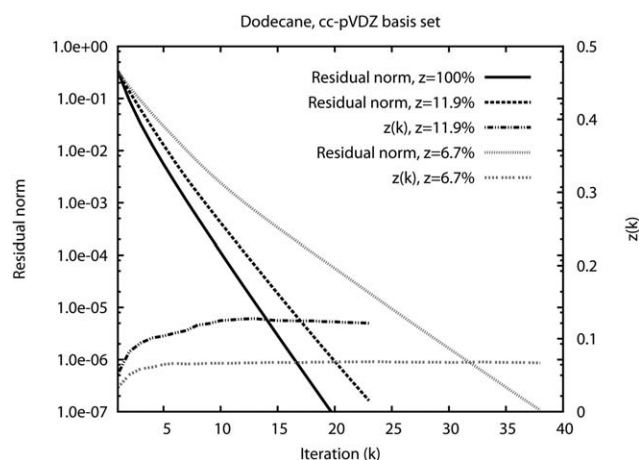


Figure 5. Convergence of CCD equations for dodecane at the equilibrium geometry using various levels of sparsification z . The cc-pVDZ basis set was employed.

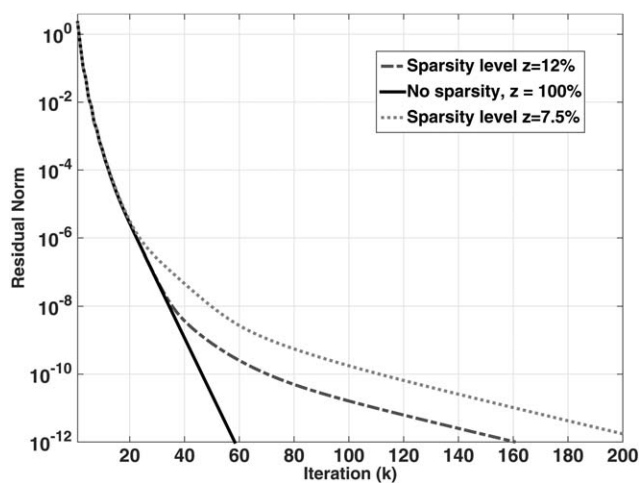


Figure 6. Convergence of CCD equations for O_3 at the equilibrium geometry using various levels of sparsification z and tight threshold 10^{-12} . The cc-pVTZ basis set was employed. By sparsifying the amplitude correction to approximate solutions to the CC amplitude equations, we can significantly reduce the number of operations in the tensor contraction performed in the inexact Newton algorithm. We can maintain nearly the same convergence rate even when roughly 90% of nonzero elements in the amplitude correction are set to zero (i.e., $z = 10\%$).

where K is the total number of iterations required to reach convergence.

Figure 2 also shows additional convergence curves obtained by progressively relaxing the tolerance τ_k , which results in setting more elements of $\hat{\Delta}(T^{(k)})$ to zero (i.e., keeping fewer nonzero elements).

As one can clearly see, if 10% of the nonzero elements in $\hat{\Delta}(T^{(k)})$ are retained on average, the convergence rate of the inexact Newton iteration is nearly identical to that without sparsification. Only two more iterations (with a total 21 iterations) are required to reach convergence. When the number of retained nonzero elements is lowered to 2.2%, more than 100 iterations are required to reach convergence. Similar pattern in convergence behavior is observed for other molecules and other basis sets, as illustrated in Figures 4 and 5. Even though more inexact Newton iterations may be required to reach convergence when a large percentage of elements of $\hat{\Delta}(T^{(k)})$ are set to zero, the total number of floating point operations performed in the amplitudes update procedure may decrease significantly.

We point out that the percentage of nonzero elements (z) that should be retained depends on the required accuracy. When a highly accurate solution is required, z can still be set to a relatively small value in the early iterations without affecting the convergence rate because in these earlier iterations, $R(T^{(k)})$ is relatively large even if evaluated exactly. However, when $R(T^{(k)})$ becomes small, z may need to be increased to ensure that sparsification does not introduce too large an error in the evaluation of $R(T^{(k)})$. Figure 6 shows that with the error tolerance set to 10^{-12} , the convergence of the sparsified algorithm is relatively insensitive to the sparsity level (z) in the first 20 iterations. However, as the approximation approaches the exact solution (indicated by the residual error going below 10^{-6} or 10^{-8}), keeping z at a small value (e.g., 7.5 or 12%) may delay convergence.

Table 2. Convergence of the CCD equations for various sparsity levels z in the amplitude correction and the overall cost of the calculation measured by the effective number of the full (nonsparse) CCD iteration, K_{eff} .

System	Basis set (n_o/n_v)	z	K	p	K_{eff}
LiF	cc-pVTZ (6/54)	1.000	45	1.00	45.0
		0.090	45	0.16	7.2
		0.024	157	0.05	7.9
LiF	cc-pVQZ (6/104)	1.000	54	1.00	54.0
		0.071	54	0.11	5.9
		0.024	133	0.04	5.3
O_3	cc-pVDZ (12/30)	1.000	28	1.00	28.0
		0.114	31	0.27	8.4
		0.073	42	0.19	8.0
		0.022	126	0.08	10.1
		0.018	157	0.07	11.0
O_3	cc-pVTZ (12/78)	1.000	25	1.00	25.0
		0.104	28	0.20	5.6
		0.069	39	0.14	5.5
		0.024	105	0.05	5.3
		0.017	145	0.04	5.8
		0.007	145	0.04	5.8
O_3	aug-cc-pVTZ (12/126)	1.000	26	1.00	26.0
		0.105	27	0.18	4.9
		0.078	35	0.13	4.6
		0.041	64	0.07	4.5
		0.018	141	0.03	4.2
O_3	cc-pVQZ (12/153)	1.000	24	1.00	24.0
		0.107	26	0.17	4.4
		0.084	30	0.14	4.2
		0.032	75	0.05	3.8
		0.022	107	0.04	4.3

The convergence threshold is 10^{-7} .

Assuming that the number of operations performed in our sparse algorithm is proportional to

$$p^{(k)} = \frac{c^{(k)}}{c}, \quad (24)$$

one can quantify the total amount of computational work by the product of the number of iterations taken (K) and the average number of p . This product can be viewed as an

Table 3. Convergence of the CCD equations for various sparsity levels z in the amplitude correction and the overall cost of the calculation measured by the effective number of the full (nonsparse) CCD iteration, K_{eff} .

System	Basis set (n_o/n_v)	z	K	p	K_{eff}
Pentane	cc-pVDZ (21/109)	1.000	20	1.00	20.0
		0.105	38	0.21	8.0
		0.075	52	0.16	8.3
Pentane	aug-cc-pVDZ (21/202)	0.027	136	0.07	9.5
		1.000	21	1.00	21.0
		0.120	35	0.21	7.4
		0.083	49	0.15	7.4
Alanine	cc-pVDZ (24/95)	0.053	75	0.09	6.8
		1.000	71	1.00	71.0
		0.085	64	0.19	12.2
Asparagine	cc-pVDZ (35/131)	0.044	118	0.11	13.0
		0.025	211	0.07	14.8
		1.000	34	1.00	34.0
		0.060	87	0.14	12.2
		0.029	171	0.08	13.7

The amplitude convergence threshold is 10^{-7} .

Table 4. Convergence of the CCD equations for various sparsity levels z in the amplitude correction and the overall cost of the calculation measured by the effective number of the full (nonsparse) CCD iteration, K_{eff} .

System	Basis set (n_o/n_v)	z	K	p	K_{eff}
Dodecane	cc-pVDZ (49/249)	1.000	20	1.00	20.0
		0.119	24	0.24	5.8
		0.067	39	0.14	5.5
Dodecane	6-311G (49/185)	1.000	22	1.00	22.0
		0.113	25	0.25	6.3
		0.051	47	0.12	5.6
Porphine	6-31G (81/163)	1.000	36	1.00	36.0
		0.094	34	0.25	8.5
		0.089	36	0.24	8.6
Porphine	cc-pVDZ (81/325)	1.000	46	1.00	46.0
		0.063	45	0.14	6.3

The convergence threshold is 10^{-7} .

effective iteration number K_{eff} associated with an inexact Newton iteration without sparsification.

Tables 2–4, compare the convergence behavior and the effective cost of the inexact Newton iteration with sparsified correction as a function of the parameter z , which measures the average number of nonzero elements kept in $\hat{\Delta}(T^{(k)})$.

We observe that as z decreases, the number of iterations K required to reach convergence increases. However, when z is not too small, the increase in K is less rapid than the decrease in z . As a result, the total cost of the inexact Newton iteration measured in terms of the number of effective nonsparse inexact Newton iterations, K_{eff} , is smallest for z 0.1. A further decrease of z leads to a rapid growth of K , which overweights the drop in p .

As evidenced by Tables 2 and 3 (for O_3 and pentane), diffuse basis functions, which often spoil the performance of local CC methods, do not affect the behavior of our algorithm.

The DIIS algorithm is often used in CC calculations to accelerate convergence. With DIIS, the number of inexact Newton iterations required to reach convergence may be reduced. For example, for the asparagine example, the inexact Newton iteration converges in 34 iterations without DIIS. The use of DIIS reduces the number of iterations to 18. Similarly, for porphine in the cc-pVDZ basis, 46 inexact Newton iterations are required to reach convergence without DIIS. The use of DIIS reduces the number of iterations to 21. However, in terms of the total number of floating point operations, our algorithm still has a lower cost. We can measure the cost as the number of effective iterations with no sparsification, K_{eff} . Using this metric, asparagine requires $K_{\text{eff}} = 12$ iterations to converge when the sparsity level is kept at $z = 6\%$, to be compared with 18 iterations when employing DIIS. Porphine requires only $K_{\text{eff}} = 6$ effective iterations when the sparsity level is kept at $z = 6.3\%$, which is significantly lower than the 21 iterations required by DIIS. It should be possible to combine the sparse update algorithm with DIIS, however, more work is required to understand the tradeoff between sparsity and the convergence rate in such a scheme.

Conclusion

We presented the algorithm for reducing the computational work involved in CC calculations by sparsifying the amplitude

correction within the CC amplitude update procedure. We provided a theoretical justification for this approach, which is based on the convergence theory of inexact Newton iterations. We demonstrated by numerical examples that, in the simplest case of the CCD equations, we can sparsify the amplitude correction by setting roughly 90% nonzero elements to zeros on average with only minor effects on convergence of the inexact Newton iterations. As we set more nonzero elements of the amplitude correction to zeros, more inexact Newton iterations are required to reach convergence. However, up to a certain sparsity level (often between 5 and 8%), the increase in the number of iterations is less rapid than the decrease in the number of nonzero elements in the amplitude correction, resulting in the further reduction in the overall cost of the inexact Newton iteration which we measured by an effective number of full inexact Newton iteration, K_{eff} . We showed that K_{eff} , hence the total number of floating point operations, in an inexact Newton procedure can be reduced by 70–90% when sparsification is used.

In practice, the overall efficiency of the algorithm depends on the availability of an efficient sparse tensor contraction library, which we currently do not have. When a sparse storage format is used to store the sparse amplitude correction, the indirect addressing required in a sparse tensor contraction is likely to introduce an overhead relative to dense tensor contractions. The overhead is expected to be machine dependent. More benchmarks are required to evaluate overall savings in the floating point operations in an inexact Newton with sparsified correction amplitudes. In addition, to make use of sparsified amplitude correction in the residual function evaluation, unlike original CCD implementation we need to store additional intermediate tensors of size $O(n_o^2 \cdot n_v^2)$.

The present sparsification scheme sets many nonzeros elements in $R(T^{(k)})$ to zero after they are computed, which is, of course, wasteful. It would be more efficient to predict which nonzero elements will be set to zero, and to only compute those elements that are to be retained in the next iteration. This scheme will further reduce the number of floating operations in the function evaluation. However, we found that the nonzero elements to be set to zero by our algorithm vary from one iteration to another. We are currently investigating whether it is possible to develop a strategy to predict the location of these sparsifiable elements.


Finally, we note that similar sparsification techniques have successfully been used in the context of SCF calculations for the purpose of increasing the sparsity of two-electron integrals and thus reducing work involved in the construction of the Fock matrix.^[59,60] Combined with the dynamic thresholding, the sparsification in SCF is implemented in most modern quantum chemistry packages. This work presented the algorithmic and numerical basis for exploiting these ideas within the CC hierarchy of methods.

Acknowledgments

Support for this work was provided through Scientific Discovery through Advanced Computing (SciDAC) program.

Keywords: coupled-cluster methods · sparsity · sparse correction · quasi-Newton · solvers

How to cite this article: J. Brabec, C. Yang, E. Epifanovsky, A. I. Krylov, E. Ng. *J. Comput. Chem.* **2016**, *37*, 1059–1067. DOI: 10.1002/jcc.24293

 Additional Supporting Information may be found in the online version of this article.

- [1] J. Čížek, *J. Chem. Phys.* **1966**, *45*, 4256.
- [2] T. Helgaker, P. Jørgensen, J. Olsen, *Molecular electronic structure theory*; Wiley: New York, **2000**.
- [3] R. J. Bartlett, *Mol. Phys.* **2010**, *108*, 2905.
- [4] R. J. Bartlett, M. Musial, *Rev. Mod. Phys.* **2007**, *79*, 291.
- [5] R. J. Bartlett, *Int. J. Mol. Sci.* **2002**, *3*, 579.
- [6] A. I. Krylov, *Annu. Rev. Phys. Chem.* **2008**, *59*, 433.
- [7] K. Snekskov, O. Christiansen, *WIRs Comput. Mol. Sci.* **2012**, *2*, 566.
- [8] R. J. Bartlett, *WIRs. Comput. Mol. Sci.* **2012**, *2*, 126.
- [9] B. Jeziorski, H. J. Monkhorst, *Phys. Rev. A* **1981**, *24*, 1668.
- [10] S. A. Kucharski, R. J. Bartlett, *J. Chem. Phys.* **1991**, *95*, 8227.
- [11] I. Hubač, In *New Methods in Quantum Theory*; A. Tsipis, V. S. Popov, D. R. Herschbach, J. S. Avery, Eds.; Kluwer: Dordrecht, **1996**; Vol. 8 of *NATO ASI Series 3: High Technology*; pp. 183–202.
- [12] J. Pittner, *J. Chem. Phys.* **2003**, *118*, 10876.
- [13] F. A. Evangelista, W. D. Allen, H. F. Schaefer, III, *J. Chem. Phys.* **2007**, *127*, 024102.
- [14] U. S. Mahapatra, B. Datta, D. Mukherjee, *J. Chem. Phys.* **1999**, *110*, 6171.
- [15] G. D. Purvis, R. J. Bartlett, *J. Chem. Phys.* **1982**, *76*, 1910.
- [16] P. Pulay, *Chem. Phys. Lett.* **1983**, *100*, 151.
- [17] S. Saebø, P. Pulay, *Chem. Phys. Lett.* **1985**, *113*, 13.
- [18] P. Pulay, S. Saebø, *Theor. Chim. Acta* **1986**, *69*, 357.
- [19] S. Saebø, P. Pulay, *J. Chem. Phys.* **1987**, *86*, 914.
- [20] K. Raghavachari, G. W. Trucks, J. A. Pople, M. Head-Gordon, *Chem. Phys. Lett.* **1989**, *157*, 479.
- [21] E. Schwegler, M. Challacombe, M. Head-Gordon, *J. Chem. Phys.* **1997**, *106*, 9708.
- [22] C. Hampel, H. J. Werner, *J. Chem. Phys.* **1996**, *104*, 6286.
- [23] M. Schütz, G. Hetzer, H. J. Werner, *J. Chem. Phys.* **1999**, *111*, 5691.
- [24] M. Schütz, H. J. Werner, *Chem. Phys. Lett.* **2000**, *318*, 370.
- [25] M. Schütz, *J. Chem. Phys.* **2000**, *113*, 9986.
- [26] M. Schütz, H. J. Werner, *J. Chem. Phys.* **2001**, *114*, 661.
- [27] M. Schütz, *Phys. Chem. Chem. Phys.* **2002**, *4*, 3941.
- [28] M. Schütz, *J. Chem. Phys.* **2002b**, *116*, 8772.
- [29] D. Kats, T. Korona, M. Schütz, *J. Chem. Phys.* **2006**, *125*, 104106.
- [30] D. Kats, T. Korona, M. Schütz, *J. Chem. Phys.* **2007**, *127*, 064107.
- [31] Z. Rolik, M. Kállay, *J. Chem. Phys.* **2011**, *135*, 104111.
- [32] C. Riplinger, F. Neese, *J. Chem. Phys.* **2013**, *138*, 034106.
- [33] J. E. Subotnik, M. Head-Gordon, *J. Chem. Phys.* **2005**, *123*, 064108.
- [34] M. Ziolkowski, B. Jansík, T. Kjærgaard, P. Jørgensen, *J. Chem. Phys.* **2010**, *133*, 014107.
- [35] J. D. Carroll, J. J. Chang, *Psychometrika* **1970**, *35*, 283.
- [36] N. H. F. Beebe, J. Linderberg, *Int. J. Quantum Chem.* **1977**, *12*, 683.
- [37] C. Eckart, G. Young, *Psychometrika* **1936**, *1*, 211.
- [38] T. Kinoshita, O. Hino, R. J. Bartlett, *J. Chem. Phys.* **2003**, *119*, 7756.
- [39] U. Benedikt, A. A. Auer, M. Espig, W. Hackbusch, *J. Chem. Phys.* **2011**, *134*, 054118.
- [40] O. Hino, T. Kinoshita, R. J. Bartlett, *J. Chem. Phys.* **2004**, *121*, 1206.
- [41] N. Beebe, J. Linderberg, *Int. J. Quant. Chem.* **1977**, *12*, 683.
- [42] H. Koch, A. de Merás, T. Pedersen, *J. Chem. Phys.* **2003**, *118*, 9481.
- [43] F. Aquilante, R. Lindh, T. B. Pedersen, *J. Chem. Phys.* **2007**, *127*, 114107.
- [44] F. Aquilante, L. Gagliardi, T. B. Pedersen, R. Lindh, *J. Chem. Phys.* **2009**, *130*, 154107.
- [45] F. Aquilante, T. B. Pedersen, R. Lindh, *Theor. Chem. Acc.* **2009**, *124*, 1.
- [46] F. Aquilante, L. Boman, J. Boström, H. Koch, R. Lindh, A. de Merás, T. Pedersen, In *Linear-Scaling Techniques in Computational Chemistry and Physics*; R. Zalesny, M. Papadopoulos, P. Mezey, J. Leszczynski, Eds.; Springer: United States, **2011**; pp. 301–343.
- [47] M. Feyereisen, G. Fitzgerald, A. Komornicki, *Chem. Phys. Lett.* **1993**, *208*, 359.
- [48] O. Vahtras, J. Almlöf, M. Feyereisen, *Chem. Phys. Lett.* **1993**, *213*, 514.
- [49] D. Bernholdt, R. Harrison, *Chem. Phys. Lett.* **1996**, *250*, 477.
- [50] Y. Jung, A. Sodt, P. M. W. Gill, M. Head-Gordon, *Proc. Natl. Acad. Sci. USA* **2005**, *102*, 6692.
- [51] F. Weigend, M. Kattannek, R. Ahlrichs, *J. Chem. Phys.* **2009**, *130*, 164106.
- [52] E. G. Hohenstein, R. M. Parrish, T. J. Martínez, *J. Chem. Phys.* **2012**, *137*, 044103.
- [53] E. G. Hohenstein, R. M. Parrish, C. D. Sherrill, T. J. Martínez, *J. Chem. Phys.* **2012b**, *137*, 221101.
- [54] R. M. Parrish, E. G. Hohenstein, T. J. Martínez, C. D. Sherrill, *J. Chem. Phys.* **2012**, *137*, 224106.
- [55] F. Bell, D. Lambrecht, M. Head-Gordon, *Mol. Phys.* **2010**, *108*, 2759.
- [56] G. J. O. Beran, M. Head-Gordon, *J. Chem. Phys.* **2004**, *121*, 78.
- [57] A. Landau, K. Khistyayev, S. Dolgikh, A. Krylov, *J. Chem. Phys.* **2010**, *132*, 014109.
- [58] E. Epifanovsky, D. Zuev, X. Feng, K. Khistyayev, Y. Shao, A. I. Krylov, *J. Chem. Phys.* **2013**, *139*, 134105.
- [59] J. Almlöf, J. K. Faegri, K. Korsell, *J. Comput. Chem.* **1982**, *3*, 385.
- [60] E. Schwegler, M. Challacombe, M. Head-Gordon, *J. Chem. Phys.* **1997**, *106*, 9708.
- [61] R. Dembo, S. Eisenstat, T. Steihaug, *SIAM J. Numer. Anal.* **1982**, *19*, 400.
- [62] S. Eisenstat, H. Walker, *SIAM J. Optimization* **1994**, *4*, 393.
- [63] E. Epifanovsky, M. Wormit, T. Kuś, A. Landau, D. Zuev, K. Khistyayev, P. Manohar, I. Kaliman, A. Dreuw, A. Krylov, *J. Comput. Chem.* **2013**, *34*, 2293.
- [64] C. Sherrill, A. Krylov, E. Byrd, M. Head-Gordon, *J. Chem. Phys.* **1998**, *109*, 4171.
- [65] M. Valiev, E. Bylaska, N. Govind, K. Kowalski, T. Straatsma, H. V. Dam, D. Wang, J. Nieplocha, E. Apra, T. Windus, et al. *Comput. Phys. Commun.* **2010**, *181*, 1477.

Received: 9 July 2015
Revised: 21 October 2015
Accepted: 17 December 2015
Published online on 24 January 2016