Short note

Sparse approximate matrix-matrix multiplication for density matrix purification with error control

Anton G. Artemov, Emanuel H. Rubensson*

Division of Scientific Computing, Department of Information Technology, Uppsala University, Box 337, SE-751 05 Uppsala, Sweden

A R T I C L E   I N F O

Article history:
Available online 14 April 2021

Keywords:
Electronic structure calculations
Density matrix methods
Error control
Sparse matrices

A B S T R A C T

We propose an accelerated density matrix purification scheme with error control. The method makes use of the scale-and-fold acceleration technique and screening of submatrix products in the block-sparse matrix-matrix multiplies to reduce the computational cost. An error bound and a parameter sweep are combined to select a threshold value for the screening, such that the error can be controlled. We evaluate the performance of the method in comparison to purification without acceleration and without submatrix product screening.

© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Sparse matrix–matrix multiplication is a key operation in linear scaling electronic structure calculations based on, for example, Hartree–Fock or Kohn–Sham density functional theory. This operation has therefore received a lot of attention from method and software developers in this field [6]. This includes the development of sparse data structures [8,16,24,31], approximation techniques taking advantage of the special properties of the matrices involved [2,18,21], and different approaches to parallelization [3,4,11,22,33]. Sparse matrix–matrix multiplication is used in the construction of the density matrix defined by

$$D = \theta(\mu I - F),$$

where $I$ is the identity matrix, $\theta(x)$ is the Heaviside function, $\mu$ is the chemical potential and $F$ is the Fock/Kohn–Sham matrix. A number of different methods for the computation of the density matrix, including minimization and recursive polynomial expansion methods, use a sequence of matrix-matrix multiplications. Recursive polynomial expansion methods are also referred to as density matrix purification. The great performance of these methods for large systems can be attributed to the decay properties of the density matrix and any matrices that occur during the course of its computation. In exact arithmetics, the matrices involved contain many entries of small magnitude. In efficient implementations, this is utilized by the removal of small matrix entries from the matrix representation, meaning that they are treated as zeros [5,11,30,32]. A key issue and a common topic for discussion is how this is done and what implications it has for performance and accuracy.

In the recursive polynomial expansion methods it is possible to strictly control the accuracy of the final result if the norm of the matrix consisting of removed matrix entries can be controlled [15,26]. In the original approach, the truncation

* Corresponding author.

E-mail address: emanuel.rubensson@it.uu.se (E.H. Rubensson).

https://doi.org/10.1016/j.jcp.2021.110354

0021-9991/© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).
is performed separately from the matrix-matrix multiplication. First, the product is computed and then small matrix entries are removed from the matrix. This is repeated in every iteration. Multiplication of two sparse matrices often results in a large increase of the number of nonzero matrix entries [13,28]. In the recursive expansion, many of the just introduced nonzeros have small magnitude and will anyway be removed in the subsequent truncation. This means that computational resources are used to compute and temporarily store those matrix entries for no purpose. Several remedies for this issue have been proposed. For block-sparse matrix representations it has been proposed to skip submatrix products of blocks within small norm [3,7,23]. In the cutoff radius approach all matrix entries that correspond to distances between nuclei or basis function centers larger than some cutoff radius are excluded from the representation [5,18]. Since, in this case, the nonzero pattern is known in advance, the product may be computed directly in truncated form.

In the present work we are particularly interested in sparse matrix representations that make use of sparse quaternary trees (quadtree) to store matrices where any identically zero submatrix quadrant is left out of the representation [34]. In the quadtree representation a matrix $X$ is either 1) stored in a data structure used for small enough matrices, or it is 2) flagged as identically zero, or it is 3) composed of four quadrants,

$$X = \begin{pmatrix} X_{0,0} & X_{0,1} \\ X_{1,0} & X_{1,1} \end{pmatrix},$$

each a matrix recursively represented as a quadtree. The data structure used for small matrices may be a regular column- or row-wise dense matrix representation or some sparse matrix format. In the matrix product, zero branches in the quadtree are skipped. In the SpAMM approach [2,9] one also skips submatrix products whose norm is known to be small. Such skipping is performed at each level of the quadtree, see Algorithm 1.

The approaches described above alleviate the issue of fill-in and dispensable computations but do not offer error control. In this paper, we show how fill-in can be reduced while strictly controlling the error. We propose a new purification algorithm based on the second order spectral projection method (SP2) [17] using the scale-and-fold acceleration technique [14,19,20] and an error control that accounts for errors due to submatrix product screening. We make use of the SpAMM algorithm but add a preceding step to carefully select the SpAMM tolerance to achieve error control in the whole purification scheme.

## 2. Error control in SpAMM

We present in this section a method to select the SpAMM tolerance so that the Frobenius norm of the SpAMM product error can be controlled. Our method combines an error bound with a parameter sweep. We will first show how a bound of the SpAMM product error can be computed for given input matrices $A$ and $B$ and a given SpAMM tolerance $t$. Let us consider how the multiplication of 2-by-2 block matrices is performed with SpAMM. Assume that the blocks are enumerated as in (2). Then the product matrix $C = AB$ is given by

$$C = \begin{pmatrix} A_{0,0}B_{0,0} + A_{0,1}B_{1,0} & A_{0,0}B_{0,1} + A_{0,1}B_{1,1} \\ A_{1,0}B_{0,0} + A_{1,1}B_{1,0} & A_{1,0}B_{0,1} + A_{1,1}B_{1,1} \end{pmatrix}. \tag{3}$$

Assume that the SpAMM tolerance $t_1$ is such that the whole procedure is not performed, because $\|A\|_F \|B\|_F < t_1$. Then, clearly, the error matrix $E = C - AB$ and $\|E\|_F \leq \|A\|_F \|B\|_F < t_1$. So the error norm is bounded by the product of the multiplicands’ norms.

Suppose that we multiply the same matrices approximately with some other tolerance $t_2$ and that three of the submultiplications $A_{0,0}B_{0,0}, A_{0,1}B_{1,0}, A_{1,1}B_{1,1}$ are skipped because the product of norms is too small. The result of this operation is the matrix
\[
\tilde{C} = \begin{pmatrix}
0 & A_{0,0}B_{0,1} + A_{0,1}B_{1,1} \\
A_{1,0}B_{0,0} + A_{1,1}B_{1,0} & A_{1,0}B_{0,1}
\end{pmatrix}.
\]

Then, the error matrix \( E = C - \tilde{C} \) is
\[
E = \begin{pmatrix}
A_{0,0}B_{0,0} + A_{0,1}B_{1,0} & 0 \\
0 & A_{1,1}B_{1,1}
\end{pmatrix}
\]
and its Frobenius norm can be bounded from above as
\[
\|E\|_F = \left( \|A_{0,0}B_{0,0} + A_{0,1}B_{1,0}\|_F^2 + \|A_{1,1}B_{1,1}\|_F^2 \right)^{\frac{1}{2}} 
\leq \left( \|A_{0,0}\|_F \|B_{0,0}\|_F + \|A_{0,1}\|_F \|B_{1,0}\|_F \right)^2
+ \|A_{1,1}\|_F^2 \|B_{1,1}\|_F^2 \right)^{\frac{1}{2}}.
\]

The idea of our algorithm to find the optimal SpAMM tolerance is based on the observation outlined above: each skipped multiplication brings an error in the product matrix, and this error can be bounded at any level of the matrix hierarchy. The summation of the errors from the underlying multiplications can be done as in (6). This gives the error bound for given \( A, B \), and SpAMM tolerance \( t \).

In Algorithm 2 we combine the error bound with a parameter sweep. This algorithm computes a bound of the SpAMM product error \( \|\text{SpAMM}(A, B, t) - AB\|_F \) for each of \( N \) candidates \((t_1, \ldots, t_N)\) for the SpAMM tolerance. Once we know an error bound for each \( t_i \), it is straightforward to pick the right SpAMM tolerance so that the corresponding error is the largest below the tolerance for the product error. The SpAMM tolerance is returned by a routine get_spamm_thres, which is a wrapper around the CSE routine in Algorithm 2.

**Algorithm 2 Compute SpAMM Errors (CSE).**

1: procedure CSE(A, B, \( t_1, \ldots, t_N \))
2: \( Errors = (0, \ldots, 0) \) \( \triangleright \) length \( N \)
3: if \( \|A\|_F \|B\|_F = 0 \) then return Errors
4: end if
5: if lowest level then
6: \( \text{for } k = 1, \ldots, N \text{ do} \)
7: if \( \|A\|_F \|B\|_F < t_k \) then \( Errors[k] = \|A\|_F \|B\|_F \)
8: end if
9: end for
10: return Errors
11: end if
12: \( \text{for } i = 0, 1 \text{ do} \)
13: \( \text{for } j = 0, 1 \text{ do} \)
14: \( E_1 = \text{CSE}(A_{i,0}, B_{0,j}, (t_1, \ldots, t_N)) \)
15: \( E_2 = \text{CSE}(A_{i,1}, B_{1,j}, (t_1, \ldots, t_N)) \)
16: \( Errors = Errors + (E_1 + E_2)^2 \) \( \triangleright \) Hadamard power, vector sum
17: end for
18: end for
19: \( Errors = Errors^{\frac{1}{2}} \)
20: return Errors
21: end procedure

Both the CSE and SpAMM algorithms can be modified to operate on matrices stored in a shallower hierarchical representation or some flat block-sparse structure. For example, with a flat block-sparse structure, the parameter sweep in CSE is to be done for every block in the product matrix, and the number of intermediate error vectors (denoted as \( E_1 \) and \( E_2 \) in Algorithm 2) is proportional to the matrix size. The SpAMM algorithm on the flat structure simplifies to block-sparse matrix-matrix multiplication with an extra if-statement, which corresponds to on-the-fly matrix filtering as in refs. [3,7,23].

### 3. Accelerated density matrix purification with error control

We propose here an accelerated density matrix purification scheme with error control and prescreening of small submatrix products in the sparse matrix-matrix multiplications. The new scheme is presented in Algorithm 3 and is based on Algorithm 5 of ref. [14]. The main difference compared to ref. [14] is the use of the SpAMM approach to matrix-matrix multiplication, and its error control described in the previous section, and consequential adjustments of the error control in the purification scheme.

In Algorithm 3, \( \lambda_{\text{min}} \) and \( \lambda_{\text{max}} \) are the extremal eigenvalues of \( F \) or outer bounds thereof. In this work we use bounds provided by Gershgorin’s theorem. The parameter \( \varepsilon \) controls the accuracy in the occupied subspace such that \( \|D - P_{X_i}\|_F \leq \varepsilon \), where \( X_i \) is the occupied subspace of \( X_i \) and \( P_{X_i} \) is the matrix for orthogonal projection onto \( X_i \). The occupied subspace
of \( X_i \) is here defined as the subspace spanned by the eigenvectors of \( X_i \) that correspond to the \( n_{\text{occ}} \) largest (closest to 1) eigenvalues, and \( n_{\text{occ}} \) is the number of occupied orbitals. See ref. [26] for the theory behind the error control. The parameter \( n_{\max} \) is an upper bound of the number of iterations. Furthermore, \( \varepsilon_i \) is the homo-lumo gap of \( X_i \) or a lower bound, for \( i = 0, 1, \ldots, n_{\max} \). The parameters \( p_i \) and \( \alpha_i \) determine what polynomial to use in the expansion, for \( i = 1, \ldots, n_{\max} \). Here, \( p_i \) decides whether eigenvalues in iteration \( i \) are to be pushed towards 0 or 1 and \( \alpha_i \) is a scaling parameter controlling the acceleration. With \( \alpha_i = 1 \), for all \( i \), the algorithm reverts to the regular SP2 scheme without acceleration.

**Algorithm 3 SP2-ACC with error control and SpAMM.**

\[
\begin{align*}
1: & \quad \text{procedure } \text{Purification}(\mathcal{F}, \lambda_{\min}, \lambda_{\max}, \varepsilon, n_{\min}, n_{\max}, \delta, \bar{\delta}, p_i, \alpha_i, i = 1, \ldots, n_{\max}) \\
2: & \quad X_0 = \frac{\mathcal{F}}{\lambda_{\max}} \\
3: & \quad \tau_0 = \left( \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min}} \right) \left( 1 + \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min}} \right) \\
4: & \quad \tilde{X}_0 = \text{truncation}(X_0, \tau_0) : ||\tilde{X}_0 - X_0||_F \leq \tau_0 \\
5: & \quad \tau_1 = \left( \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min}} \right) \left( 1 + \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min}} \right) \\
6: & \quad \tau_{\alpha i} = \text{get_spamm_threshold}(\tilde{X}_0, \tilde{X}_0, \bar{\delta}(1 - \varepsilon_1)) \\
7: & \quad X_0 = \text{SpAMM}(\tilde{X}_0, \tilde{X}_0, \tau_{\alpha i}) \\
8: & \quad \varepsilon_1 = ||\tilde{X}_0 - X_0||_F \\
9: & \quad C = 6.8872 \\
10: & \quad \text{for } i = 1, \ldots, n_{\max} \text{ do} \\
11: & \quad \text{if } p_i = 1 \text{ then} \\
12: & \quad \quad X_i = (1 - \alpha_i)^2 I + 2(1 - \alpha_i)\alpha_i \tilde{X}_{i-1} + \alpha_i^2 X_{i-1} \\
13: & \quad \text{else} \\
14: & \quad \quad X_i = 2\alpha_i \tilde{X}_{i-1} - \alpha_i^2 X_{i-1} \\
15: & \quad \text{end if} \\
16: & \quad \tilde{X}_i = \text{truncation}(X_i, \tau_\delta) : ||\tilde{X}_i - X_i||_F \leq \tau_\delta \\
17: & \quad \tau_{i+1} = \left( \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min} + 1} \right) \left( 1 + \frac{\varepsilon}{\lambda_{\max} - \lambda_{\min} + 1} \right) \\
18: & \quad \tau_{\alpha i} = \text{get_spamm_threshold}(\tilde{X}_i, \tilde{X}_i, \bar{\delta}(1 - \varepsilon_i)) \\
19: & \quad X_i = \text{SpAMM}(\tilde{X}_i, \tilde{X}_i, \tau_{\alpha i}) \\
20: & \quad \varepsilon_i = ||\tilde{X}_i - X_i||_F \\
21: & \quad \text{if } i \geq n_{\min} \text{ and } (p_i \neq p_{i-1}) \text{ and } \varepsilon_i > Ce_{i-2} \text{ then} \\
22: & \quad \quad \text{break} \\
23: & \quad \text{end if} \\
24: & \quad \text{end for} \\
25: & \quad \text{end procedure}
\end{align*}
\]

We use the parameter-free stopping criterion of ref. [14] which accurately detects when numerical errors start to dominate and the matrix cannot get closer to idempotency. The parameter-free stopping criterion ensures that the error in eigenvalues gets as small as possible at the present level of truncation errors, controlled by the tolerance for the error in the occupied subspace. In density matrix purification methods, it is not possible to recover from a loss of accuracy in the occupied subspace. The eigenvalues, on the other hand, can always be pushed closer to idempotency by increasing the numerical accuracy at the end of the purification, as done in refs. [12,26]. The stopping criterion is formulated for SP2 without acceleration. However, when the scheme has reached the phase of quadratic convergence, the acceleration has virtually no effect. The acceleration [19] can be seen as a preconditioner to the quadratic purification phase, see refs. [12,26]. Therefore, when preparing the input of this algorithm, \( \alpha_i \) is set to 1, for \( i = n_{\min}, \ldots, n_{\max} \), where \( n_{\min} \) is a point after which the eigenvalues are close to idempotency and the acceleration has little effect. In this work, we turned off the acceleration as soon as each eigenvalue is within a distance of 0.01 from 0 or 1. After this point, i.e. after \( n_{\min} \) iterations, adjustments of the stopping criterion to account for the acceleration are not needed.

We use Algorithm 4 of ref. [14] to compute \( n_{\min}, n_{\max}, p_i, \alpha_i, i = 1, \ldots, n_{\max} \) with the minor adjustment of the algorithm to also return lower estimates \( \varepsilon_i, i = 0, 1, \ldots, n_{\max} \) of the homo-lumo gap in each iteration. Initial homo and lumo eigenvalue estimates were computed as a by-product of the purification process, as described in ref. [20]. Finally, \( \delta \) is a parameter between 0 and 1 that controls to what extent regular truncation and/or SpAMM with screening of submatrix products should be used: With \( \delta = 0 \), truncation or screening of products is only done within the SpAMM multiply whereas with \( \delta = 1 \), truncation is only done separately, and finally \( \delta = 0.5 \) gives a hybrid scheme. Note that with \( \delta = 1 \), the SpAMM threshold is zero and the SpAMM call is reduced to regular sparse matrix-matrix multiplication.

The expressions for the threshold values \( \tau_i, i = 0, 1, \ldots \) are derived in ref. [26], and give a bound on the error in the Frobenius norm in each iteration. If \( p_i = 0 \), then the condition

\[
||\tilde{X}_i - (2\alpha_i \tilde{X}_{i-1} - \alpha_i^2 \tilde{X}_{i-1}^2)||_F \leq \tau_i
\]

should hold, and correspondingly for \( p_i = 1 \). The tolerated error is split between SpAMM and the truncation as defined by \( \delta \). Note that to get strict error control with acceleration, i.e. with \( \alpha_i \neq 1 \), a scaling \( 1/\alpha_i^2 \) of the tolerance for SpAMM is added. Note also that the SpAMM error appears in the polynomial evaluation first in the iteration following the SpAMM evaluation,
which is why $\tau_{i+1}$, rather than $\tau_i$, is used to get the SpAMM threshold. In the routine `get_spamm_thresh` a tolerance to be used in SpAMM is chosen so that $\|X^2 - X^{c0}\|_F$ comes below the given tolerance, as described in the previous section.

In the regular truncation scheme, employed in the routine truncate, as many submatrices as possible are removed while satisfying the condition that the Frobenius norm of the error matrix is below the given threshold value. The submatrices with smallest Frobenius norms are removed first.

Utilization of matrix symmetry in the matrix square operation in Algorithm 3 is straightforward in the quadtree representation and can reduce the cost by a factor close to two [22,24]. Algorithm 2 may be adapted correspondingly. For symmetric matrices in upper triangular storage one needs to loop only over the upper triangle and double count off-diagonal contributions to the error.

### 4. Experimental evaluation

We used Algorithm 3 to compute the density matrix close to self-consistent field convergence in Hartree-Fock calculations with the EPRG software [30] using the standard Gaussian basis set STO-3G, for a water cluster with 5772 atoms from ref. [29], with xyz coordinates available at ergoscif.org, giving a matrix dimension of 13468. The water cluster coordinates were taken from a molecular dynamics calculation at standard temperature and pressure. A standard recursive binary divide-space procedure, based on the location of basis function centers, was used to order the basis functions for data locality in the matrices. See, for example, refs. [25,31] for different approaches to blocking and ordering of basis functions. The Roothaan-Hall eigenvalue equations were transformed to standard form using the inverse Cholesky factor of the overlap matrix.

The numerical experiments were performed on an HP-EliteDesk-800-G1-SFF carrying an Intel Core i7-4770 processor with 4 cores running at a base frequency of 3.4 GHz and 32 GB of memory. In our computations we used the hierarchical block-sparse library first presented in ref. [1]. This library implements the quadtree matrix representation described in the introduction and its source code is freely available at `chucks-and-tasks.org` as one of the leaf matrix libraries of the Chunks and Tasks matrix library 2.0 [27]. The quadtree recursion was terminated at a block size of 32, where a regular dense columnwise representation of submatrices was used. Submatrix products at the lowest level were carried out using OpenBLAS 0.3.13 which was built from sources with disabled multi-threading. The code was compiled with gcc version 9.3.0.

At each level of the quadtree representation in the hierarchical block sparse library, together with each submatrix is stored also its Frobenius norm. The Frobenius norms are precomputed for the whole matrix hierarchy, before any call to the CSE or SpAMM algorithms. The present implementation does not take advantage of matrix symmetry.

To evaluate the error control proposed in this work we have run Algorithm 3 with separate truncation only (Regular), with SpAMM truncation only (SpAMM), and in hybrid mode (Hybrid), controlled by the $\delta$ parameter of Algorithm 3, as described earlier. We also compare the performance of the algorithm with (SP2-ACC) and without (SP2) acceleration, controlled by the $\{\tau_i\}$ parameters. In all calculations, we set the tolerance for the subspace error to $\varepsilon = 10^{-2}$. In `get_spamm_thresh` we generate a set of possible SpAMM tolerances for Algorithm 2 by $\tau_{i+1} = \text{tol}_i \times \tau_i$ for $i = 2, \ldots, 15$, where tol is the input tolerance to `get_spamm_thresh`, which gives logarithmically equally spaced values. For the accelerated scheme with truncation in hybrid mode (SP2-ACC-Hybrid), the subspace error tolerance $\varepsilon = 10^{-2}$ resulted in $\tau_i$-values ranging between about $2 \cdot 10^{-5}$ and $5 \cdot 10^{-4}$ and SpAMM tolerances $\{\tau_i\}$ ranging between about $4 \cdot 10^{-10}$ and $2 \cdot 10^{-8}$.

We present overall performance figures in Table 1. Acceleration of SP2 gives a significant performance boost, both in terms of number of operations and wall time. Compared to the Regular approach, the SpAMM and Hybrid approaches to truncation both give smaller flop counts and smaller error in the density matrix, while the SpAMM approach gives the longest wall times. The larger error (but still below the tolerance) in the Regular approach indicates a tighter error estimate, but the scheme anyway gives the largest flop count. We note that compared to the other non-accelerated methods, SP2-Regular converged in fewer iterations. This is due to the larger error in this method with the chosen threshold values. The parameter-free stopping criterion was satisfied after 24 iterations since the idempotency error could not be further reduced.

Figs. 1a and 1b show how the number of floating point operations and the total wall time is split over the iterations for SP2-ACC using the three different approaches to truncation. For each curve in Figs. 1a and 1b, the sum over all the iterations gives the corresponding number in Table 1. As a reference, a single dense matrix-matrix multiplication with the present

### Table 1

<table>
<thead>
<tr>
<th>Method</th>
<th>$n$</th>
<th>TFlap</th>
<th>Wall time [sec]</th>
<th>$\epsilon_n = |X_n - X^{c0}|_F$</th>
<th>$|D - X_n|_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SP2-Regular</td>
<td>24</td>
<td>1.79</td>
<td>696.9</td>
<td>$1.4 \cdot 10^{-4}$</td>
<td>$3.1 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>SP2-SpAMM</td>
<td>26</td>
<td>2.11</td>
<td>909.1</td>
<td>$4.4 \cdot 10^{-6}$</td>
<td>$5.7 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>SP2-Hybrid</td>
<td>26</td>
<td>1.97</td>
<td>623.8</td>
<td>$6.5 \cdot 10^{-5}$</td>
<td>$1.7 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>SP2-ACC-Regular</td>
<td>15</td>
<td>1.78</td>
<td>443.8</td>
<td>$1.8 \cdot 10^{-4}$</td>
<td>$4.2 \cdot 10^{-4}$</td>
</tr>
<tr>
<td>SP2-ACC-SpAMM</td>
<td>15</td>
<td>1.30</td>
<td>549.9</td>
<td>$6.3 \cdot 10^{-6}$</td>
<td>$7.8 \cdot 10^{-5}$</td>
</tr>
<tr>
<td>SP2-ACC-Hybrid</td>
<td>15</td>
<td>1.19</td>
<td>376.4</td>
<td>$9.0 \cdot 10^{-5}$</td>
<td>$2.3 \cdot 10^{-4}$</td>
</tr>
</tbody>
</table>
hybridization, our approach outperforms the other two.

5. Concluding remarks

In summary, we have presented a new density matrix purification scheme, based on the accelerated second order recursive polynomial expansion, with control of errors due to screening of submatrix products in the block sparse matrix-matrix multiplications. We evaluate the algorithm using three different modes of truncation: the standard approach to truncate matrices separately from the multiplication (referred to as Regular), the approach to do screening of submatrix products in the block-sparse matrix-matrix multiplication (SpAMM), and the hybrid approach (Hybrid). We show that the cost, both in terms of memory usage and number of floating point operations, is significantly reduced when comparing the SpAMM and Hybrid approaches to the Regular approach. In terms of wall time, the Hybrid approach outperforms both the other methods. Thus, our results indicate that the Hybrid approach is the best option. In all cases, the acceleration gives a substantial performance boost.
Although the SP2-ACC-Hybrid approach outperforms the other methods, there is room for improvements. The method still involves the computation of a significant number of matrix entries that are thrown away in the subsequent truncation. This can be explained by the inability of the CSE algorithm to detect all possible products that could be removed. In Table 1, the actual error in the density matrix is more than an order of magnitude smaller than the given tolerance. This indicates that the performance could be improved while still achieving the desired accuracy. There are several sources of this overestimation. The estimate of the error in the occupied subspace is based on a worst case erroneous rotation between the eigenvectors close to the homo-lumo gap, see the expression with the homo-lumo gap in Algorithm 3; there may also be error cancellation between the iterations. Most relevant to the present work is a significant overestimation of the SpAMM error in Algorithm 2. The schemes that only make use of SpAMM truncation give smaller errors than the other methods. However, a basic requirement of the present work is rigorous control of the error. Note that the actual error is available only because we have computed an accurate reference solution at high cost. Besides improving the error bound in the Frobenius norm, future work could also address error control for SpAMM in other norms.

In the experimental evaluation we made use of the hierarchical block-sparse library of ref. [1]. A two-stage procedure is used for the sparse matrix-matrix multiplication where lists of nonzero submatrix multiplies are first prepared and then executed in batches. Although not implemented in the present work, the batched multiply could be executed in parallel or offloaded to GPU as in refs. [3,22].

We note that asymptotic error analyses with respect to both the SpAMM tolerance and system size have been carried out in earlier work [1,10]. Here, we have proposed a scheme to select the SpAMM tolerance so that the error is below a predefined tolerance as needed in our accelerated density matrix purification scheme with prescreening of submatrix products and error control.

CRediT authorship contribution statement

Anton G. Artemov: Conceptualization, Investigation, Methodology, Software, Writing – original draft. Emanuel H. Rubensson: Conceptualization, Investigation, Methodology, Software, Supervision, Writing – review & editing.
Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This work was supported by the Swedish National strategic e-science research program (eSENSE). Besides the presented experimental evaluation, computational experiments were also performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the PDC Center for High Performance Computing, KTH Royal Institute of Technology.

References