Efficient and Scalable Wave Function Compression Using Corner Hierarchical Matrices

Kenneth O. Berard,[1,2,a)] Hongji Gao,[1,3,b)] Alexander Teplukhin,[1,2,c)] Xiangmin Jiao,[1,3,d)] and Benjamin G. Levine[1,2,e)]

[1)] *Institute for Advanced Computational Science, Stony Brook University, Stony Brook, NY, USA 11794*

[2)] *Department of Chemistry, Stony Brook University, Stony Brook, NY, USA 11794*

[3)] *Department of Applied Mathematics and Statistics, Stony Brook University, Stony Brook, NY, USA 11794*

The exponential scaling of complete active space (CAS) and full configuration interaction (CI) calculations limits the ability of quantum chemists to simulate the electronic structures of strongly correlated systems. Herein, we present corner hierarchically approximated CI (CHACI), an approach to wave function compression based on corner hierarchical matrices (CH-matrices)—a new variant of hierarchical matrices based on a block-wise low-rank decomposition. By application to dodecacene, a strongly correlated molecule, we demonstrate that CH matrix compression provides superior compression compared to a truncated global singular value decomposition. The compression ratio is shown to improve with increasing active space size. By comparison of several alternative schemes, we demonstrate that superior compression is achieved by a) using a blocking approach that emphasizes the upper-left corner of the CI vector, b) sorting the CI vector prior to compression, and c) optimizing the rank of each block to maximize information density.

[a)] Contributed equally; Current address: Department of Chemistry, Brown University, Providence, RI, USA

[b)] Contributed equally

[c)] Current Address: Intel Corporation RA2, Hillsboro, OR USA 97124

[d)] Correspondence email address: xiangmin.jiao@stonybrook.edu

[e)] Correspondence email address: ben.levine@stonybrook.edu

I. INTRODUCTION

Complete active space (CAS) configuration interaction (CI) and its self-consistent field counterpart (CASSCF) remain the quantum chemist's workhorse method for describing multi-reference systems.[1] The most frequent targets for multi-reference calculations include strongly correlated systems, dissociated molecules, transition states, conical intersections involving the electronic ground state, and excited electronic states with multiple excited characters. CAS methods are viable when the multi-reference effects in question can be described in an active orbital space of fewer than ∼20 electrons. Unfortunately, the factorial scaling of the CAS wave function with active space size renders it impractical for anything larger.

As such, strategies for using compressed representations of the CASCI wave function have been widely explored in recent years, exploiting the large number of zero or near-zero elements of the CI coefficient vector ("configurational deadwood"[2]) to reduce compute time and storage. For example, restricted, generalized, and other active space methods use different truncations of the CAS expansion to reduce storage and cost.[3–9] Similarly, selected CI approaches use various physical criteria to develop a sparse representation of the CI vector on-the-fly.[10–30] Seniority CI achieves an efficient sparse representation of strongly correlated systems via a truncation of the CI space based on seniority number.[31] Full configuration interaction quantum Monte Carlo (FCI-QMC) methods can describe very large active spaces without explicit representation of the entire CI vector, allowing a fixed number of walkers to explore the most important regions of the CI vector stochastically.[32–34] An algorithm for choosing orbitals to maximize CI vector sparsity in exchange-coupled systems has also recently been reported.[35]

Other methods take advantage of data sparsity rather than sparsity. Whereas a sparse matrix has many zero elements, a data-sparse matrix may contain few zero elements but is compressible because it is low-rank or has other redundancies that may be exploited. For example, a highly efficient representation of the strongly correlated wave function of one-dimensional and quasi-one-dimensional systems can be achieved via the density matrix renormalization group (DMRG) approach.[36–38] The DMRG method can be thought of as leveraging the data sparsity of the CI vector. Rank-reduced configuration interaction methods also leverage data sparsity through a low-rank approximation of the CI vector,[39–43]

and in a similar spirit, ideas from compressive sensing have been adapted to wave function compression.[44] Going one step further, it has been argued that the wave function itself is not an essential descriptor of electronic structure and can be replaced by the electron density[45] or reduced density matrices.[46–48] The search for an efficient and broadly applicable compressed wave function representation remains an active and important research direction, and recently there have been efforts to systematize the benchmarking of methods for compression of strongly correlated wave functions.[49,50]

In this paper, we present a proof-of-concept demonstration of a novel wave function compression strategy based on corner hierarchical matrices (CH-matrices), a new variant of hierarchical matrices. CH-matrices are motivated by H-matrices and H^2-matrices,[51–53] which use a block-wise low-rank approximation to a non-sparse matrix, leveraging data sparsity to minimize both storage and the computational cost of matrix operations. For $N \times N$ matrices, the H-matrix representation allows storage and matrix multiplication with quasi-linear scaling ($\mathcal{O}(N \log N)$ and $\mathcal{O}(N \log^2 N)$, respectively). Open-source software is available to manipulate various flavors of hierarchical matrices, and given their block-wise nature, hierarchical matrices are well-suited for massively parallel implementation.

The effectiveness of H- and H^2-matrices relies on some degree of diagonal dominance of the matrices. This is because the matrix is subdivided into blocks, with smaller blocks on the diagonal and increasing block size as you move away from the diagonal, as illustrated in Figure 1. The blocks closest to the diagonal, which are shaded gray, are represented as dense matrices with no loss of accuracy. The off-diagonal blocks (shown in white) are approximated as low-rank. Typically, the rank is held constant despite the fact that the blocks are of different sizes, resulting in greater compression of the data farthest from the diagonal. These hierarchical-matrix techniques are beginning to see use in quantum chemistry. Chow and coworkers have leveraged hierarchical matrices to compress the electron repulsion integrals (ERI).[54–56] In this context, the hierarchical-matrix approach can be thought of as an algebraic generalization of the fast multipole method.[57–59] Using the proxy point method, the hierarchically compressed ERI tensor may be computed for an arbitrary contracted Gaussian basis, in a similar spirit. Unfortunately, the CASCI wave function is not diagonally dominant, so these hierarchical-matrix approaches are not very effective in our experience. Instead, we observe that the CASCI wave function is dominated by the upper-left corner of the CI vector. Based on this observation, we have developed a new variant of

hierarchical matrices as illustrated in Figure 2, which we call Corner Hierarchical matrices (CH-matrices), to compress matrices with such patterns.

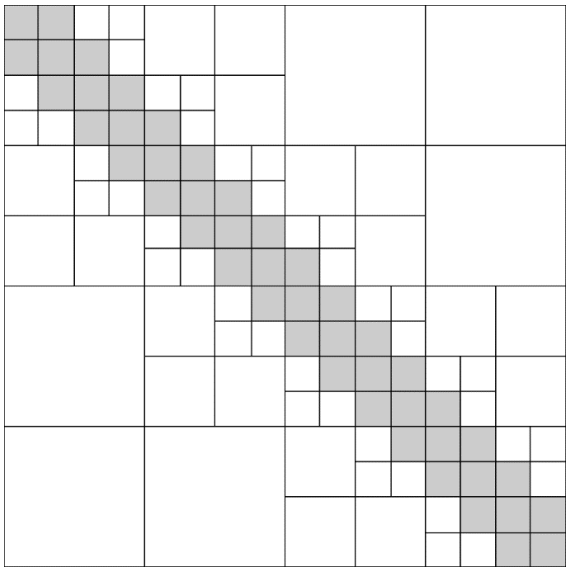

Figure 1. Block representation of an H-matrix. White blocks are stored as low-rank approximations, while gray blocks are stored as dense matrices.

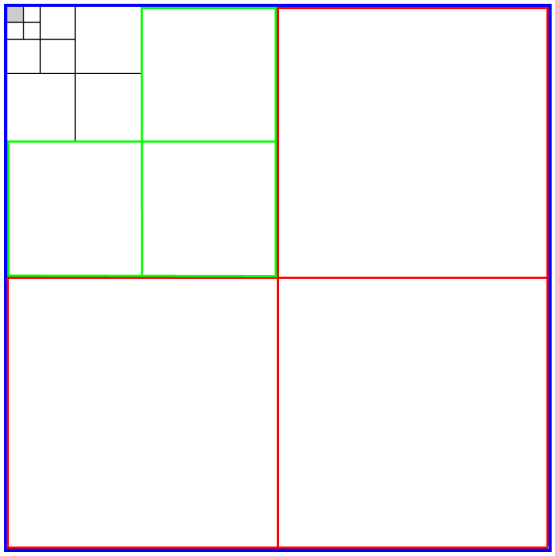

Figure 2. Schematic of the corner-hierarchical matrix structure. The upper-left corner is compressed hierarchically and the other blocks are compressed adaptively.

Herein, we introduce our CH-matrix approach to compress the CASCI vectors of strongly correlated systems. In section II, we define the corner hierarchically approximated CI (CHACI) representation of the wave function, along with several other candidate compression strategies. In section III, we compare the performance of CHACI to several other approaches, including a truncated singular value decomposition approach that represents an optimal global low-rank approximation, akin to rank-reduced full CI (RRFCI). Finally, in section IV, we present conclusions and discuss the prospects for the direct optimization of CHACI wave functions. Some details and analysis of the CHACI algorithm are presented in the appendix.

II. THEORY AND METHODS

Ultimately we seek to directly optimize the electronic wave function in a compressed format. However, in the present paper, we do not solve for the wave function in a compressed

format. The purpose of this paper is to demonstrate that hierarchical matrices provide a viable strategy for wave function compression. To this end, we will solve for the full CASCI wave function via traditional approaches, and then compress the resulting wave function into one of several forms. We can then decompress, and apply several metrics to quantify the accuracy of the compressed wave function.

Throughout this work, we will define the set of Slater determinants, $\{|\alpha\beta\rangle\}$, in the CASCI wave function in terms of their composite α and β strings. The α (β) strings index the possible ways that N_α (N_β) spin-up (spin-down) electrons may occupy N_{MO} active spatial molecular orbitals. Therefore, the CASCI wave function may be represented by

$$|C\rangle = \sum_{\alpha\beta} C_{\alpha\beta} |\alpha\beta\rangle . \tag{1}$$

The set of expansion coefficients, $\{C_{\alpha\beta}\}$, are typically known as the "CI vector." As such, it is typically thought of as an $M_\alpha M_\beta$ vector, but in this work, we will treat it as an $M_\alpha \times M_\beta$ matrix, $\mathbf{C}$. Here M_α and M_β represent the number of α and β strings, respectively,

$$M_\alpha = \frac{N_{\text{MO}}!}{N_\alpha!(N_{\text{MO}} - N_\alpha)!} \tag{2}$$

and

$$M_\beta = \frac{N_{\text{MO}}!}{N_\beta!(N_{\text{MO}} - N_\beta)!}. \tag{3}$$

Where not otherwise noted, the strings will be ordered according to the scheme of Duch.[60]

Below, we define several strategies for wave function compression. CHACI compression, described in subsection II.1, is the most efficient strategy we discovered in this work. Several other compression schemes, which we present in order to analyze the necessity of the various features of the CHACI algorithm, are also defined in subsection II.1. In subsections II.2 we describe the model problems that we choose to test our algorithm and present computational details. In subsection II.3, the performance metrics that we use to quantify the accuracy of the compressed wave function are defined.

II.1. CHACI Wave Function Compression

The CHACI wave function compression scheme is based on what we call a corner hierarchical (CH) compression. In contrast to the H-matrix scheme illustrated in Figure 1, which was conceived to compress diagonally-dominant matrices, the CH scheme (Figure 2)

is designed to compress matrices that are dominated by the upper-left corner of the matrix. Given the CI vector stored as an $M \times M$ matrix, the CH compression algorithm proceeds as follows. The total matrix (blue) is subdivided into four approximately equal-sized subblocks. The upper-right, lower-left, and lower-right subblocks (red in Figure 2) are compressed adaptively using either a low-rank or dense approximation. The upper left subblock is further subdivided, yielding four smaller subblocks. In a similar fashion, three (green) subblocks are again compressed. The number of levels is approximately logarithmic in M and determined *a priori*. Given the number of levels, we then determine whether to store each block as sparse or dense, as we describe in Algorithm 1 and analyze its near optimality in Appendix A.

Throughout this work, we will employ truncated singular value decomposition (TSVD) to compress individual blocks. The SVD of an $m \times n$ block, $\mathbf{B}$, is

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger, \tag{4}$$

where $\mathbf{U}$ and $\mathbf{V}$ are, respectively, $m \times m$ and $n \times n$ unitary matrices comprising the left and right singular vectors, and $\mathbf{\Sigma}$ is a diagonal matrix, with diagonal elements equal to the singular values. In the context of this work, m and n are typically equal, but they may differ in general for rectangular matrices. The SVD itself is an exact representation of the matrix $\mathbf{B}$, and does not provide any compression. However, truncation by elimination of the smallest singular values and corresponding singular vectors provides an efficient low-rank approximation to $\mathbf{B}$,

$$\mathbf{B} \approx \tilde{\mathbf{B}} = a\tilde{\mathbf{B}}' = a\mathbf{U}_T\mathbf{\Sigma}_T\mathbf{V}_T^\dagger, \tag{5}$$

where a is a scalar normalization factor, defined below. Upon compression $\mathbf{U}_T$ and $\mathbf{V}_T^\dagger$ are now $m \times k$ and $k \times n$, where k is the compression rank. Efficient compression can be achieved when an accurate representation of $\mathbf{B}$ is achieved for $k \ll \min(n, m)$. TSVD compression decreases the Frobenius norm of the block, and therefore the overall wave function. To compensate for this, we rescale each block to maintain its original norm after compression, according to

$$a = \frac{\|\mathbf{B}\|_F}{\|\mathbf{U}_T\mathbf{\Sigma}_T\mathbf{V}_T^\dagger\|_F}. \tag{6}$$

This block-wise renormalization prevents the shifting of population from sparser to denser blocks, which would be an artifact of global renormalization. Note that truncated SVD, without subsequent renormalization, is an optimal low-rank approximation to the block, in

that the Frobenius norm of the difference between $\tilde{\mathbf{B}}'$ and $\mathbf{B}$,

$$\|\tilde{\mathbf{B}}' - \mathbf{B}\|_F = \sum_{i,j} (\tilde{B}'_{ij} - B_{ij})^2, \tag{7}$$

is minimized, for a given rank. However, this condition does not guarantee variational optimality (minimal energy).

In CHACI, CH compression of $\mathbf{C}$ is supplemented by two additional strategies aimed at allowing more efficient compression. First, prior to compression, the rows (columns) of $\mathbf{C}$ are sorted such that the rows (columns) are arranged in descending order according to their L^2 norms. In this way, the large elements are concentrated in the upper left corner of $\mathbf{C}$, to the extent possible.

Secondly, in the CHACI scheme, we do not employ the same value of the compression rank for all blocks. Instead, the SVD of each block is truncated to a different rank, such that the overall Frobenius norm of the compressed wave function, $\|\tilde{\mathbf{C}}\|_F$, is optimized. A detailed derivation and algorithm are presented in the appendix, but here we summarize the primary features. The information density, ρ_i, associated with each singular vector pair is computed according to

$$\rho_i = \frac{\sigma_i^2}{n_{\text{row}} + n_{\text{col}} + 1}, \tag{8}$$

where σ_i is the singular value, and n_{row} and n_{col} are the lengths of the left and right singular vectors, respectively. Typically, n_{row} and n_{col} are equal in our algorithms, but they may differ in general for rectangular blocks. The information density can be thought of as the total contribution to $\|\tilde{\mathbf{C}}\|_F$ per unit of storage.

Only the singular vector pair whose information density is above a user-defined threshold will be stored. At a given storage value, this algorithm approximately maximizes the Frobenius norm of the wave function, $\|\tilde{\mathbf{C}}\|_F$. Equivalently, this can be thought of as discarding the least amount of information. However, this algorithm does not necessarily optimize the energy. However, when higher accuracy is required and the threshold for the Frobenius norm error is set very low, several of the upper-left corner blocks may become so dense that the storage cost of TSVD exceeds that of the dense format. To address this, we add another test after the TSVD: If the memory cost of TSVD is no less than the dense format, we directly store the block in dense format. For completeness, we present the pseudocode and its detailed analysis in the Appendix.

Table I. Summary of the features of the wave function compression schemes compared in this work.

Scheme	Blocking	Sorting?	Optimal Rank?
CHACI	Corner Hierarchical	Y	Y
SR-CHACI	Corner Hierarchical	Y	N
U-CHACI	Corner Hierarchical	N	Y
H-Matrix	Diagonally Dominant	N	N
Truncated SVD	None	N	N

Taken together, we refer to CH compression of the reordered $\mathbf{C}$ matrix with optimal rank as CHACI compression, without any additional modifier. To analyze the necessity of different features of the CHACI algorithm, we will present results for several other compression schemes, summarized in Table I. The static rank (SR-) CHACI scheme is identical to CHACI, except that the rank of the compressed blocks is held constant rather than optimized per block. We introduce SR-CHACI in order to quantify the utility of dynamically increasing the rank. The unsorted (U-) CHACI scheme is identical to CHACI, except that the rows/columns remain ordered according to the original Duch scheme, rather than sorted by norm. This scheme is introduced to quantify the impact of sorting.

In addition, we will test H-matrix compression[61] of the wave function. The H-matrix representation is an existing hierarchical representation that is not based on CH blocking. Instead, the blocking structure of the matrix is optimized to represent diagonally dominant matrices, as pictured in Figure 1. In our implementation, the sparse blocks are compressed by the same TSVD approach that we used for CHACI. The rows/columns are not sorted, and the rank is held constant for all blocks.

Finally, as an important point of comparison, we include a non-hierarchical compression scheme: TSVD of the entire $\mathbf{C}$ matrix. This form of the wave function is akin to that in RRFCI.[39–43] To maintain normalization, we rescale as in Eq. (6). Given that implementation of a CI solver based on a hierarchically compressed wave function will be challenging, CHACI must provide a significant improvement in performance over TSVD to warrant further consideration.

II.2. Model Problems and Computational Details

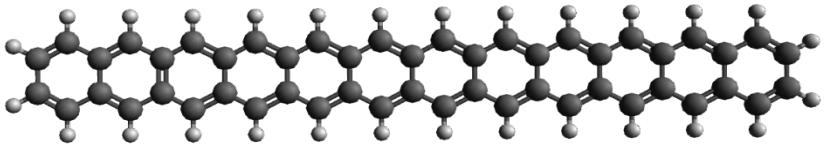

Figure 3. Molecular structure of dodecacene (12-acene).

Table II. Total number of double precision floating point values stored in a dense (uncompressed) representation of the singlet wave function CI vector as a function of active space. Data is given in kdoubles (thousands of doubles).

Active Space	Dense CI Vector Storage (kdoubles)
10-10	63
12-12	854
14-14	11,779
16-16	165,637

As a test case, we have chosen to compute the electronic structure of dodecacene (12-acene; Figure 3). Naively, one might think that longer acenes like 12-acene would have a simple closed-shell, aromatic electronic structure, but they are actually strongly correlated. These systems have significant poly-radical character, so much so that dodecacene is unstable under ambient conditions and has only recently been synthesized under ultrahigh vacuum.[62] The singlet ground state geometry of 12-acene was optimized at the CAM-B3LYP[63] level of theory. Floating occupation molecular orbital[64,65] (FOMO) CASCI calculations were then performed with active spaces of 10-10, 12-12, 14-14, and 16-16, where active spaces are abbreviated ⟨number of active electrons⟩-⟨number of active orbitals⟩. Table II presents the total storage required for the CI vector of the exact singlet wave function for each active space, measured in thousands of double-precision floating point values (kdoubles). The storage requirement for triplet wave functions is less by a factor of 0.7-0.8. The STO-3G minimal basis was used for all calculations. In all cases, both the singlet and triplet ground

state wave functions are computed at the singlet ground state geometry. All calculations were performed in the TeraChem electronic structure software package.[66–69]

II.3. Performance Metrics

We use several metrics to evaluate the performance of our various compression schemes. All are derived by comparing the properties of the compressed wave function, $\left|\tilde{C}\right\rangle$, with those of the uncompressed wave function, $|C\rangle$. Hereafter we will refer to the latter as the "exact" wave function, understanding that it is only exact within the FOMO-CASCI approximation, with a given active space and basis. In all cases, the properties of the compressed wave function are generated by a) outputting the CI vector of the uncompressed wave function from TeraChem, b) applying one of the lossy compression schemes described above, implemented in an external code, c) decompressing the wave function back to its full dimensionality, and d) feeding the resulting CI vector back into TeraChem to compute the relevant property.

To evaluate how well our compressed wave function reproduces relative energies, we compute the vertical singlet-triplet gaps of the exact and compressed wave functions, respectively

$$\Delta E_{S-T} = \langle C_T|\hat{H}|C_T\rangle - \langle C_S|\hat{H}|C_S\rangle \tag{9}$$

and

$$\Delta \tilde{E}_{S-T} = \langle \tilde{C}_T|\hat{H}|\tilde{C}_T\rangle - \langle \tilde{C}_S|\hat{H}|\tilde{C}_S\rangle. \tag{10}$$

Here the subscripted T and S denote the triplet and singlet wave functions, respectively. We report the absolute difference between these values as the error in the gap,

$$\Delta\Delta E_{S-T} = |\Delta\tilde{E}_{S-T} - \Delta E_{S-T}|. \tag{11}$$

To quantify the accuracy of the absolute energies, we also report signed errors in the singlet and triplet energies, respectively

$$\Delta E_S = \langle \tilde{C}_S|\hat{H}|\tilde{C}_S\rangle - \langle C_S|\hat{H}|C_S\rangle \tag{12}$$

and

$$\Delta E_T = \langle \tilde{C}_T|\hat{H}|\tilde{C}_T\rangle - \langle C_T|\hat{H}|C_T\rangle. \tag{13}$$

Per the variational principle, these errors are non-negative. Note that throughout this work, we compare the energies of the compressed wave functions to the energies of the exact wave functions using the same active space. Thus, our computed errors only include those associated with the compression algorithm, not with the size of the active space.

Lastly, as a separate measure of the accuracy of the wave function, we compute the absolute error in the total spin angular momentum squared (spin contamination) of the compressed singlet and triplet wave functions, respectively

$$\Delta\langle S^2\rangle = \langle\tilde{C}_S|\hat{S}^2|\tilde{C}_S\rangle \tag{14}$$

and

$$\Delta\langle S^2\rangle = |\langle\tilde{C}_T|\hat{S}^2|\tilde{C}_T\rangle - 2|. \tag{15}$$

The total spin expectation value is computed using the direct algorithm described in Ref. 70.

In many of the graphs below, the data is plotted as a function of the total number of double-precision variables stored in the compressed representation. The total storage is varied indirectly through the user-chosen parameters of the compression, as described in subsection II.1 and in the Appendix. This provides an apples-to-apples comparison of the accuracy of the wave function at a given level of compression across different compression schemes.

III. RESULTS AND DISCUSSION

Below, we analyze the performance of CHACI. In subsection III.1, we compare the accuracy of CHACI compression to that of TSVD compression, as a function of the total storage. In subsection III.2, we analyze the prospects for extending CHACI to larger active spaces. In subsections III.3 and III.4, we analyze the necessity of optimal rank and sorting, respectively. Finally, in subsection III.5, we compare H-matrix compression to TSVD compression.

III.1. Accuracy of CHACI Compression Versus TSVD

We start by comparing the performance of CHACI compression to TSVD compression for the 14-14 active space. Figure 4 presents the error in the singlet-triplet gap as a function of

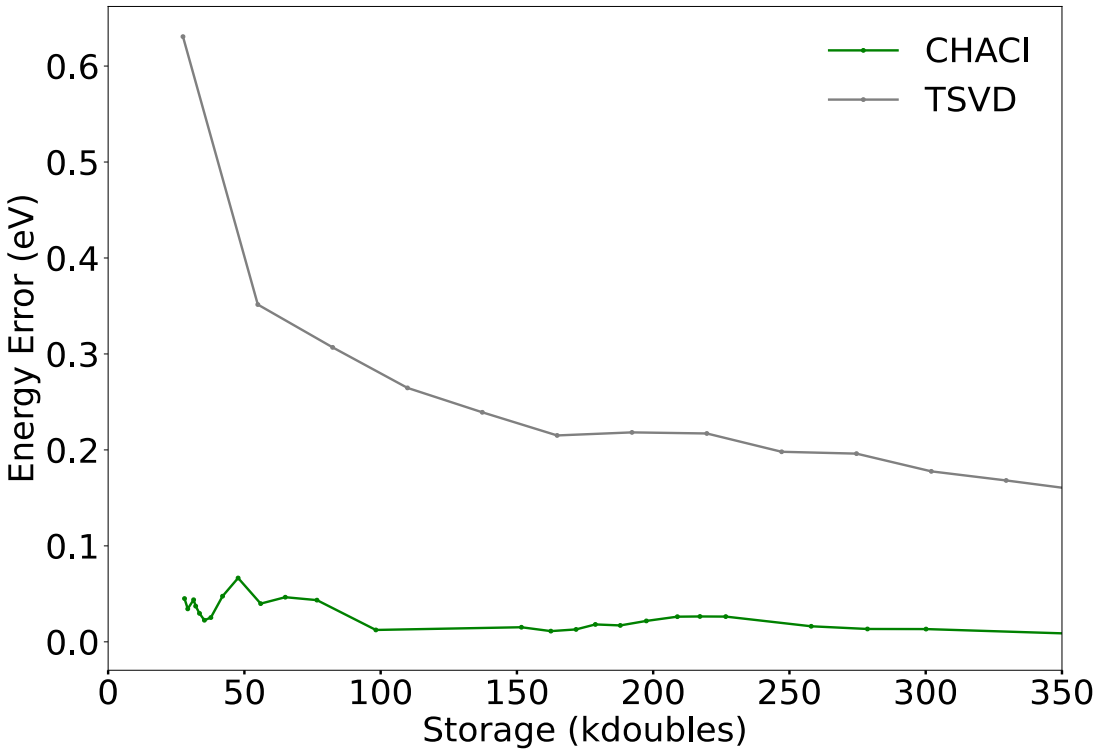

Figure 4. The error in the singlet-triplet gap of 12-acene as a function of total storage, computed with a 14-14 active space. The gray and green lines represent the error corresponding to the compression of the wave function using TSVD and CHACI, respectively.

total storage. Clearly, CHACI outperforms TSVD in this case. The CHACI error is always less than 0.07 eV, even when only 28 kdoubles are stored (compared to 11,778 kdoubles for dense storage). Truncated SVD also achieves substantial compression, but errors of 0.2 eV remain even with 220 kdoubles stored.

Figures 5a and b present errors in the singlet and triplet absolute energies, respectively. Again, the performance of CHACI is far superior to TSVD, especially in the small-storage regime, where the TSVD errors are a substantial fraction of an eV. Figure 5c and d present spin error. Here CHACI again outperforms TSVD. Even with very modest (28 kdoubles) storage, the error in $\langle S^2 \rangle$ is 0.02, and the error drops rapidly toward zero with additional storage. In contrast, TSVD spin errors for the singlet case do not drop below 0.02 until nearly 400 kdoubles are stored.

Figures 6 and 7 demonstrate that the difference in performance between CHACI and

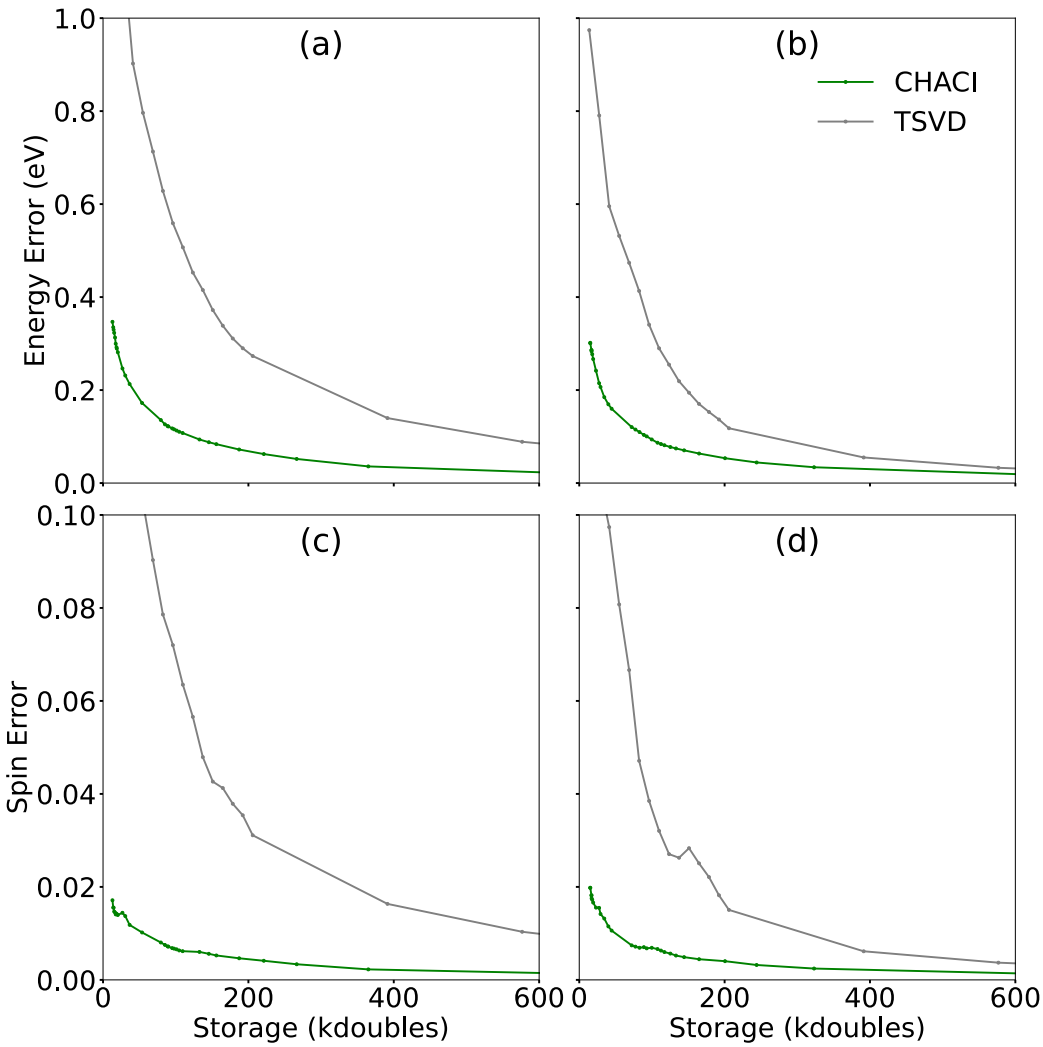

Figure 5. Absolute energy (a and b) and spin (c and d) errors as a function of the storage for 12-acene with a 14-14 active space. Panel (a) and (c) correspond to the singlet wave function, while (b) and (d) correspond to the triplet wave function. The gray and green lines correspond to TSVD and CHACI compression, respectively.

TSVD increases when the active space size is increased from 14-14 to 16-16. In Figure 6, it can be seen that errors in the singlet-triplet gap are at or below 0.1 eV for all cases, when CHACI is employed, even when only 59 kdoubles are stored (compared to 165,637 kdoubles for dense storage). Errors decrease rapidly with additional storage. In contrast, TSVD errors are above 0.2 eV for all cases. Similarly large differences in performance are observed for errors in absolute energy and spin in Figure 7. As in the 14-14 case, CHACI does a much better job of maintaining the spin symmetry of the wave function than TSVD.

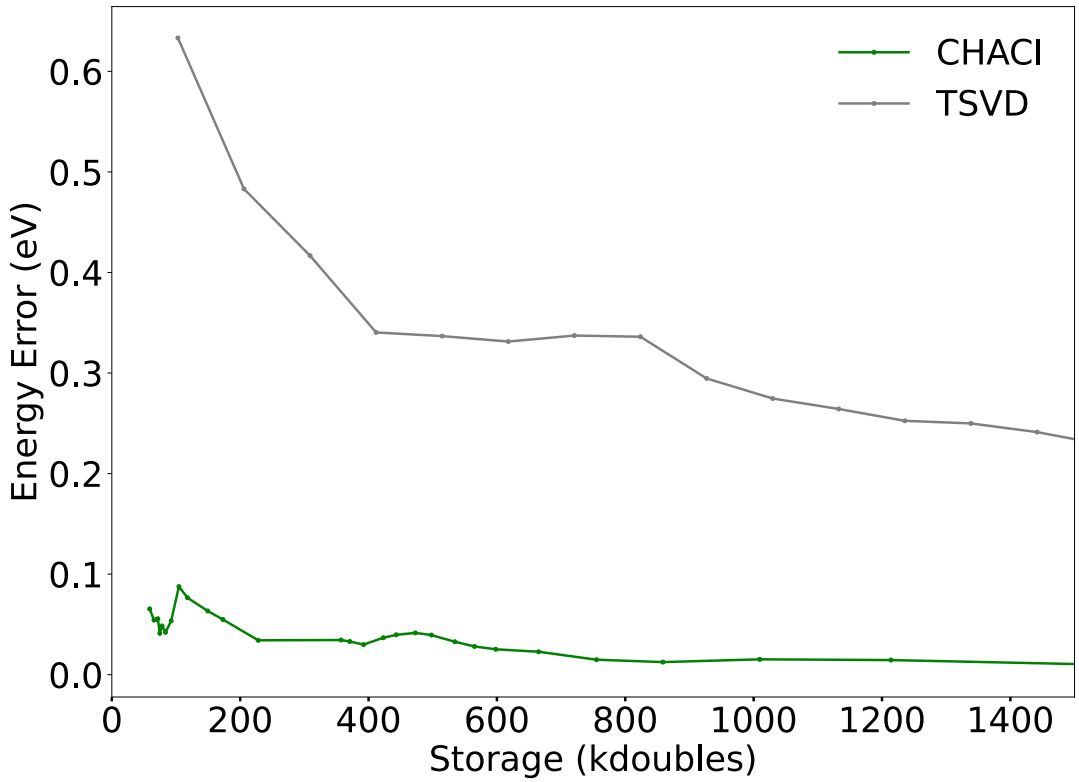

Figure 6. The error in the singlet-triplet gap of 12-acene as a function of total storage, computed with a 16-16 active space. The gray and green lines represent the error incurred by compression of the wave function using TSVD and CHACI, respectively.

III.2. Extrapolation of Performance to Larger Active Spaces

Ultimately, our goal is not to compute dense wave functions for subsequent compression. Our goal is to solve for large CI wave functions using a hierarchically compressed basis. Thus, in this section, we consider the behavior of CHACI compression as a function of active space size. In Figure 8, we consider several active spaces, comparing the convergence of several measures of the accuracy as a function of the percentage of dense storage used (the compression ratio). We find that as the size of the active space increases, the accuracies of both absolute energy and spin converge faster with increasing compression ratio.

To quantify this convergence behavior, we plot the compression ratio at which absolute energies of 0.2 eV accuracy are achieved as a function of the number of active orbitals/electrons

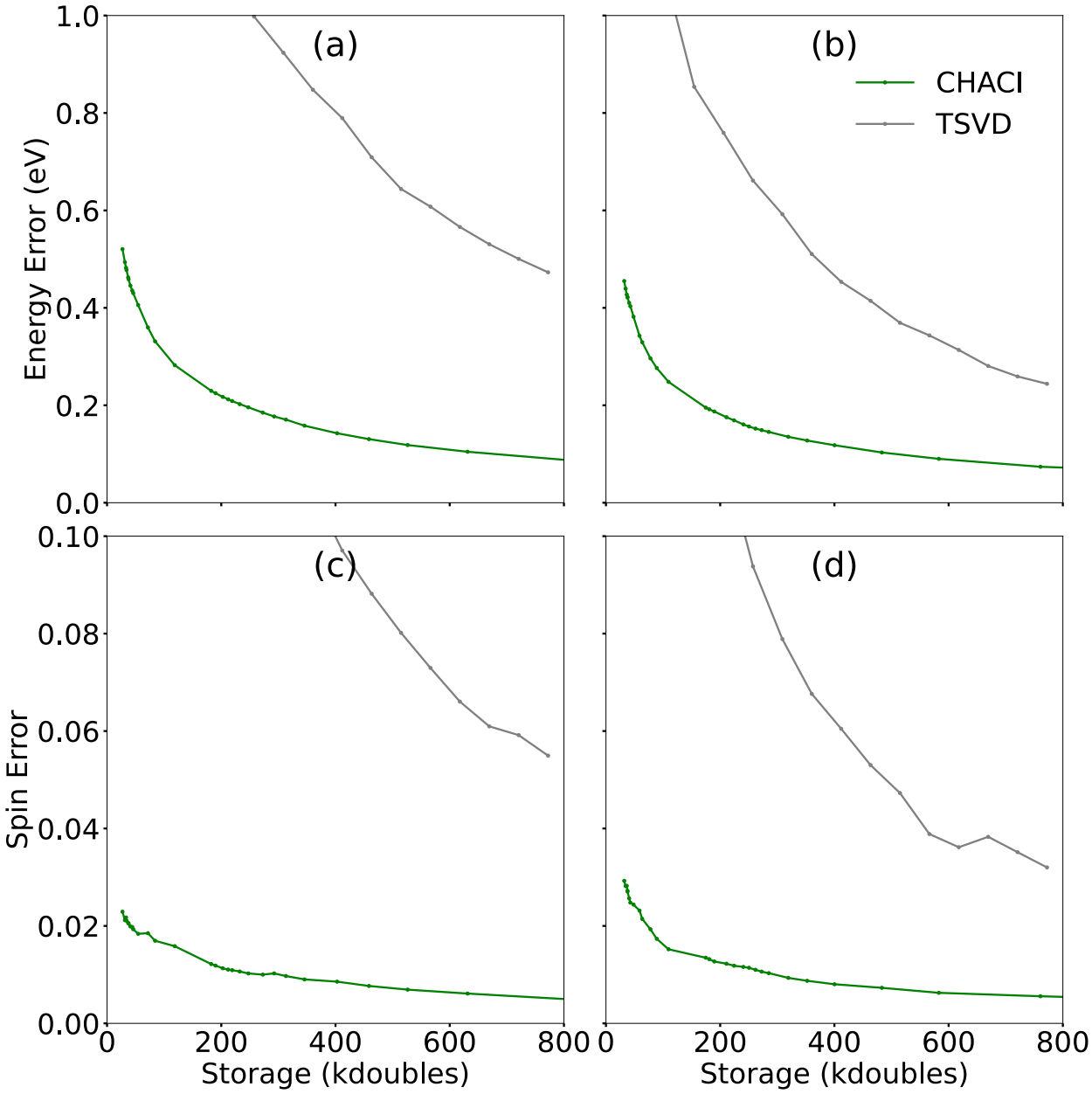

Figure 7. Absolute energy (a and b) and spin (c and d) errors as a function of the storage for 12-acene with a 16-16 active space. Panel (a) and (c) correspond to the singlet wave function, while (b) and (d) correspond to the triplet wave function. The gray and green lines correspond to TSVD and CHACI compression, respectively.

in Figure 9. Both the singlet and triplet compression ratios converge quickly with increasing active space. Of the two, the triplet energy converges more slowly, thus we fit it to an exponential in order to extrapolate to larger active spaces. We find that the required

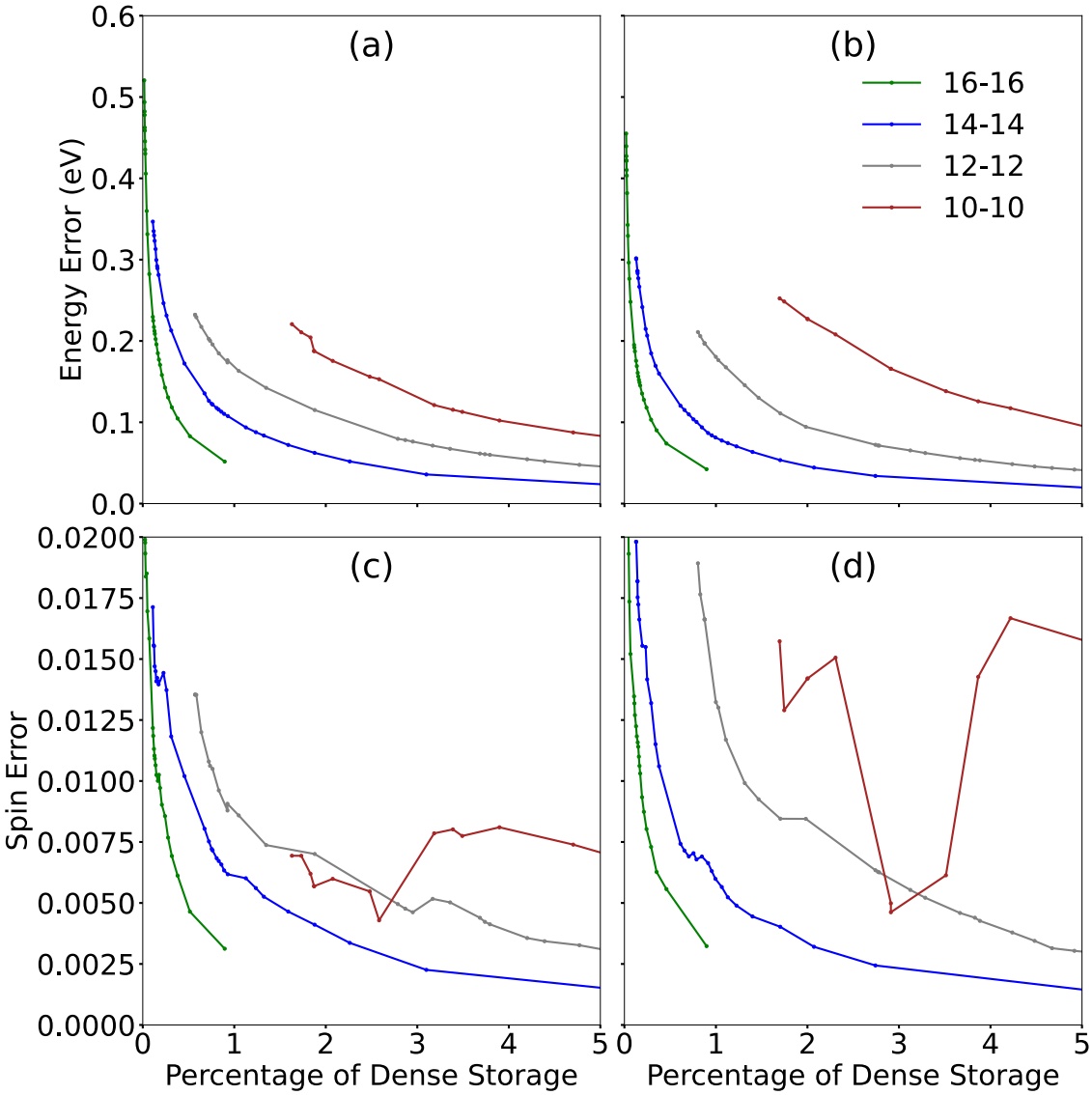

Figure 8. The absolute energy (a and b) and spin errors (c and d) as a function of the percentage of dense storage for 12-acene with 10-10, 12-12, 14-14, and 16-16 active spaces. Panels (a) and (c) correspond to the singlet wave function, while panels (b) and (d) correspond to the triplet spin wave function.

compression ratio decays proportional to

$$f(N_{\mathrm{MO}}) \propto e^{-0.561 N_{\mathrm{MO}}}. \tag{16}$$

Extrapolating to larger active spaces, we estimate that a 24-24 active space could converge to 0.2 eV accuracy at a storage cost of 77,370 kdoubles, which is less than the cost of the dense storage of a 16-16 active space (165,637 kdoubles). Though the convergence behavior

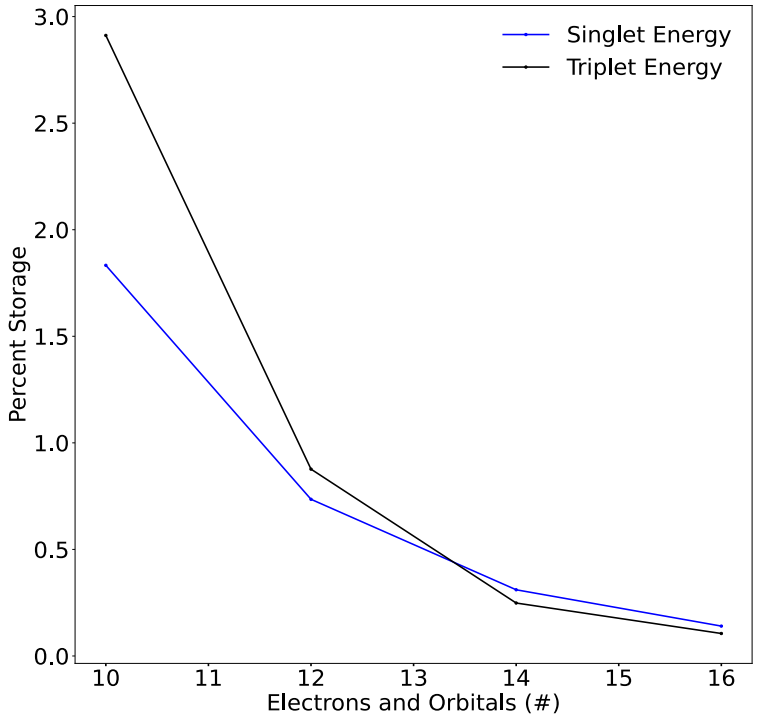

Figure 9. The storage ratio required to achieve < 0.2 eV accuracy in absolute energies as a function of the active space size. Blue and black lines correspond to the singlet and triplet wave functions, respectively.

is likely to be system-dependent, this result certainly encourages further study.

III.3. Effect of Using Optimal Rank on Compression

In order to assess the necessity of the optimal rank procedure, we compare SR-CHACI (which uses a static rank) to CHACI and TSVD. Figure 10 presents the error in the singlet-triplet gap as a function of the total storage for the 14-14 active space. Excepting one fortunate point at 160 kdoubles, the accuracy of SR-CHACI is significantly worse than that of CHACI, but still better than TSVD. However, considering the error in the absolute energies of the singlet and triplet states separately (Figure 11a and b, respectively), it is clear that this is due to error cancelation. Errors in the absolute energy of the singlet state derived from the SR-CHACI wave function are similar to those of TSVD, and much greater

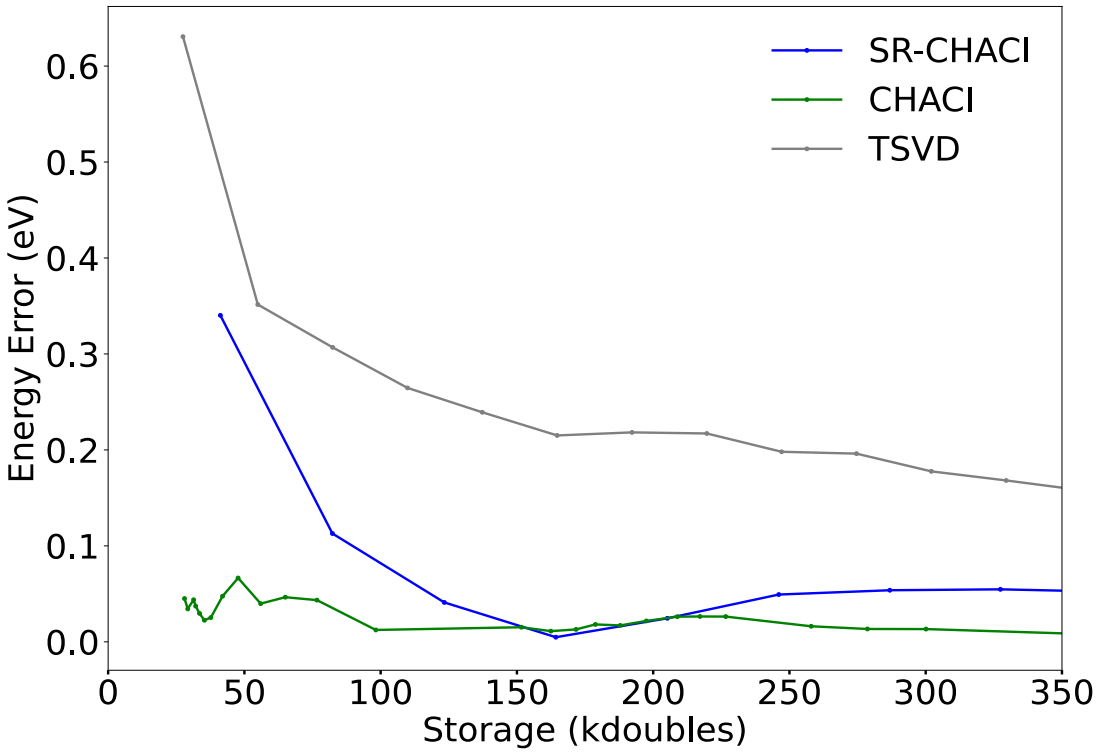

Figure 10. The SR-CHACI error (blue) in the singlet-triplet gap of 12-acene as a function of total storage, computed with a 14-14 active space. The TSVD and CHACI errors (gray and green, respectively) are shown for comparison.

than those of CHACI. Further, errors in the SR-CHACI triplet state are slightly larger than TSVD.

Analysis of spin errors tells a similar story. CHACI is much superior to SR-CHACI at reproducing the spin of the original wave function, and SR-CHACI has similar (and sometimes larger) spin errors compared to TSVD. Taken together, we conclude that optimal rank is an essential component of the CHACI algorithm.

III.4. Effect of Sorting on Compression

Here we assess the necessity of another feature of the CHACI compression algorithm: the sorting of rows/columns of the **C** matrix prior to compression. To this end, we compare U-CHACI, in which the rows/columns remain unsorted, to CHACI and TSVD. Figure 12

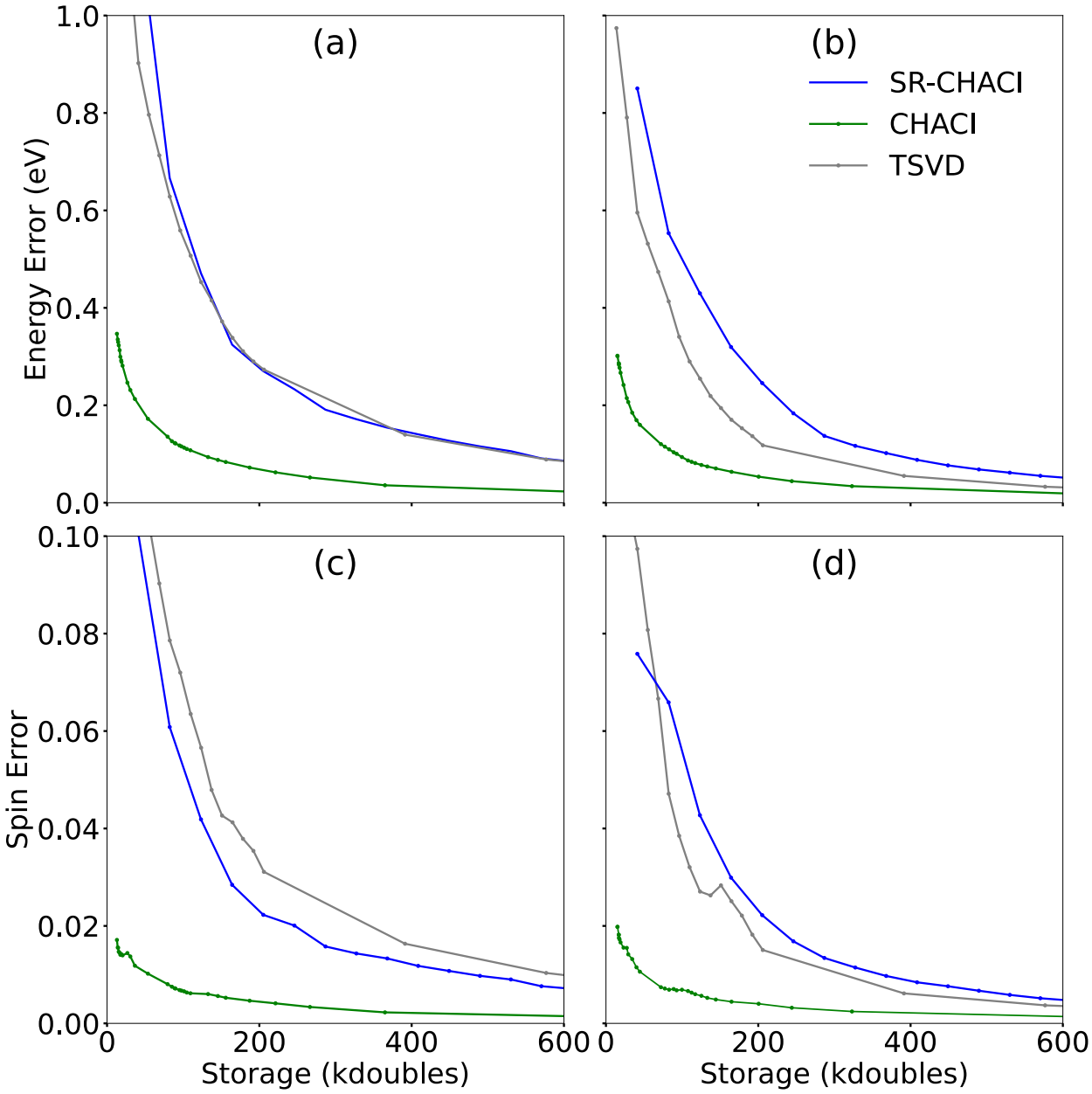

Figure 11. The SR-CHACI errors (blue) in the absolute energy (a and b) and spin (c and d) as a function of the storage for 12-acene (14-14 active space). Panels (a) and (c) correspond to the singlet wave function, while (b) and (d) correspond to the triplet wave function. The TSVD (gray) and CHACI (green) errors are shown for comparison.

presents a heat-map of the uncompressed $\mathbf{C}$ matrix of the singlet (panels a and c) and triplet (b and d) wave functions before (a and b) and after (c and d) sorting. Note that sorting concentrates larger elements into the upper-left corner of the matrix.

Figure 13 compares the U-CHACI singlet-triplet errors to those of CHACI and TSVD.

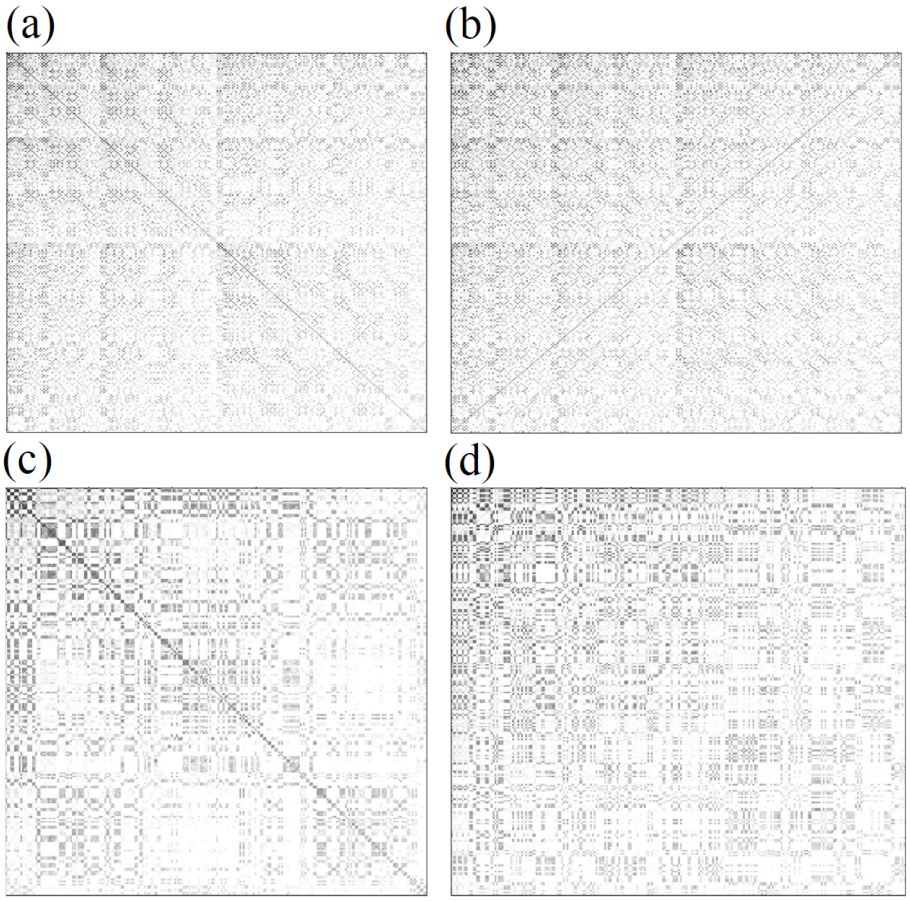

Figure 12. A heat map representation of the CI vector of 12-acene, computed with a 10-10 active space. To form the heat map, we take the logarithm (base 10) of the absolute value of the singlet and triplet CI vector coefficients. The color scale (white to black) ranges from 10^{-6} to 1. Panels (a) and (b) correspond to the unsorted CI vector (with strings indexed according to Duch[60]), while panels (c) and (d) are reordered according to the CHACI algorithm. Panels (a) and (c) show the singlet wave function, while panels (b) and (d) show the triplet wave function.

Though U-CHACI appears to be more accurate for predicting the relative energy than TSVD, it remains inferior to CHACI. Considering the errors in the absolute singlet and triplet energies (Figure 11a and b), we see that U-CHACI errors are on the order of the same size as those of TSVD, and considerably larger than those of CHACI. That being said, U-CHACI is solidly between CHACI and TSVD in its ability to accurately describe the total spin angular momentum (Figure 11c and d).

Taking this data together, we conclude that sorting is an essential component of the CHACI algorithm. However, our ultimate goal is not to compute the full wave function and

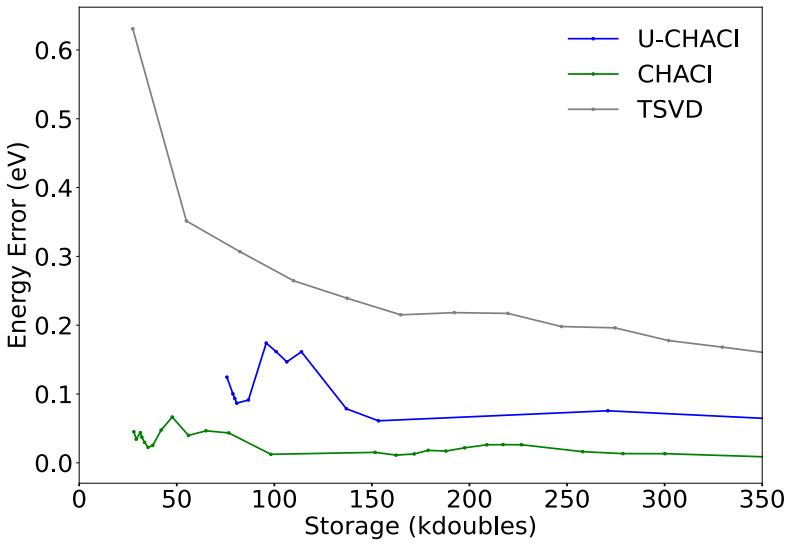

Figure 13. The U-CHACI error (blue) in the singlet-triplet gap of 12-acene as a function of total storage, computed with a 14-14 active space. The TSVD and CHACI errors (gray and green, respectively) are shown for comparison.

subsequently compress it, and the type of *a posteriori* sorting that we use in CHACI would not be possible if we were to directly solve for the wave function in compressed form. But given that the Duch ordering of spin strings does not allow for efficient compression, the determination of an *a priori* scheme by which strings may be ordered for efficient compression remains an important open question.

III.5. Effect of Upper Quadrant vs H-matrix Compression

Lastly, we consider compression of $\mathbf{C}$ into the H-matrix format,[61] which is designed to leverage the diagonally dominant nature of the matrix, in contrast to the CH blocking scheme used in CHACI. Figure 15 shows the error in the singlet-triplet gap, singlet absolute energy, and triplet absolute energy of the 14-14 wave function compressed into H-matrix format. H-matrix compression is inferior to TSVD, thus we conclude that the CH blocking scheme is an essential component of the CHACI scheme.

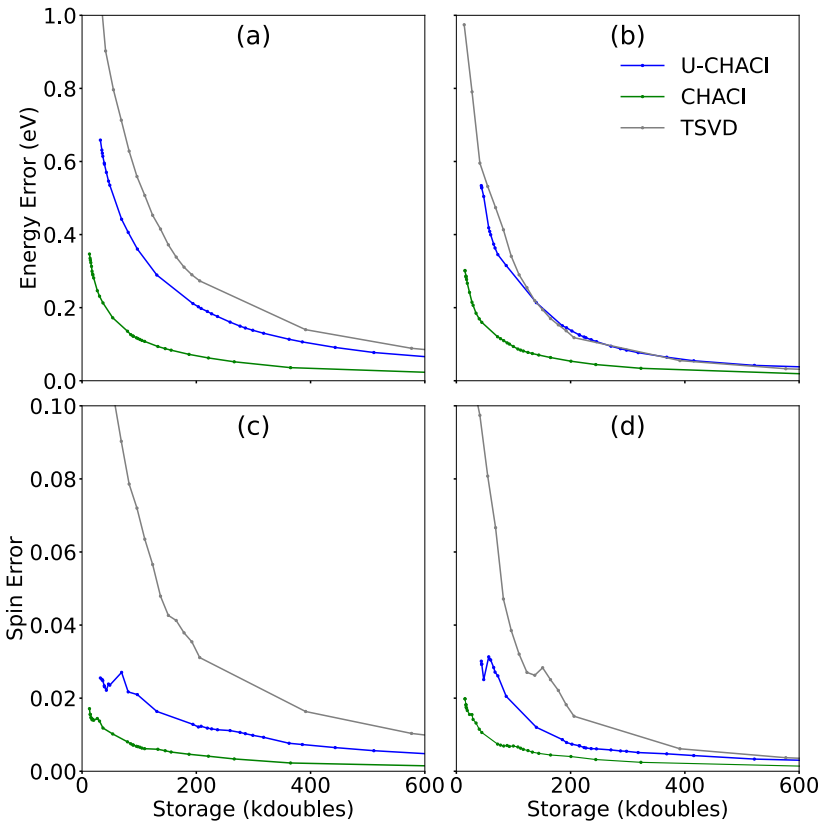

Figure 14. The U-CHACI errors (blue) in the absolute energy (a and b) and spin (c and d) errors as a function of the storage for 12-acene (14-14 active space). Panels (a) and (c) correspond to the singlet wave function, while (b) and (d) correspond to the triplet wave function. The TSVD (gray) and CHACI (green) errors are shown for comparison.

IV. CONCLUSIONS

Here we have presented an exploratory study in which we employ hierarchical matrices to perform lossy compression of CASCI wave functions of a strongly correlated system. We have demonstrated superior compression compared to a global low-rank approximation of the wave function, and we have demonstrated that CH blocking, optimal rank, and row/column sorting are essential features of the algorithm.

These promising results suggest several avenues for future study. The development of an algorithm to directly solve for the wave function in the CHACI representation would allow the accurate solution of strongly correlated wave functions with active spaces well beyond those that are accessible via brute-force CASCI implementations. That hierarchical matrices are well suited for implementation on large-scale distributed computing systems is

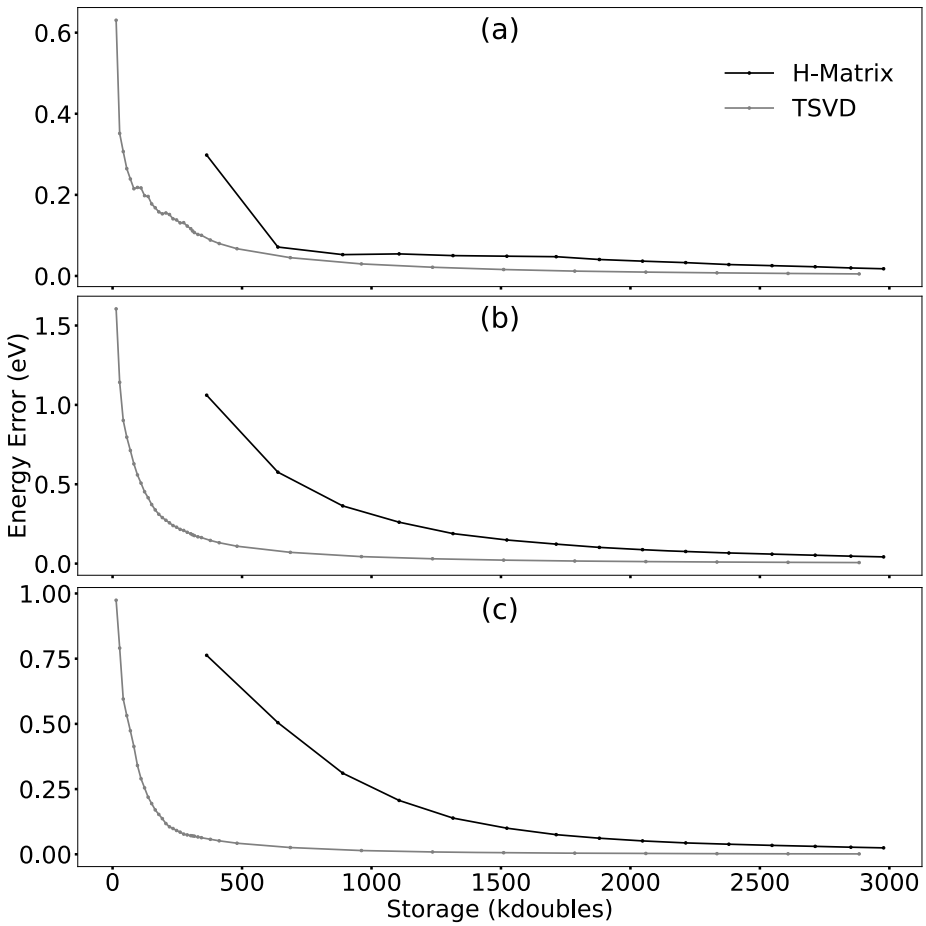

Figure 15. Panels (a), (b), and (c) show the errors in the singlet-triplet gap, singlet, and triplet energies, respectively, of 12-acene (14-14 active space) as a function of the required storage for H-matrix and TSVD compression (black and gray).

a noteworthy advantage of this approach. Yet a direct solution will only be possible after an effective *a priori* row/column sorting algorithm is developed, and the development of such an algorithm will require further analysis of the uncompressed wave function.

ACKNOWLEDGEMENTS

This work was supported by a seed grant from the Institute for Advanced Computational Science. KOB, AT, and BGL are grateful for support from the National Science Foundation under grant CHE-1954519 and for start-up funding from Stony Brook University.

Appendix A: Near-Optimal CHACI Compression

As described above, the CHACI algorithm introduces a strategy for performing recursive partitioning. Its key idea is to approximate the CI vector via truncated SVD (TSVD) of a collection of small block matrices constructed hierarchically. The algorithm leverages the fact that many CI vectors, even of relatively strongly correlated systems, have data that predominantly reside in the matrix's upper-left corner. Here we investigate this technique and propose an algorithm that can achieve near-optimal storage efficiency within this framework. For quantifying approximation accuracy, in this section, we employ the squared Frobenius norm. Specifically, if $\mathbf{A}$ represents the original CI vector and $\mathbf{A}'$ symbolizes its CHACI approximation, then the error is quantified by $\|\mathbf{A} - \mathbf{A}'\|_F^2$. Above, the numerical results consider the energy and spin errors as additional metrics for error evaluation.

A.1. Information Density

To derive the algorithm, we first introduce the concept of *information density*, which quantifies the amount of information contained in a singular value relative to the storage cost. Consider a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ with its singular value decomposition (SVD) given by

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\dagger = [\mathbf{u}_1, \ldots, \mathbf{u}_m]\mathbf{\Sigma}[\mathbf{v}_1, \ldots, \mathbf{v}_n]^T,$$

where $\mathbf{u}_i \in \mathbb{R}^m$, $\mathbf{v}_i \in \mathbb{R}^n$, and $\mathbf{\Sigma}$ is an $m \times n$ diagonal matrix composed of σ_i along its diagonal. Without loss of generality, let us assume $m \geq n$. Here, $\|\mathbf{A}\|_F^2 = \sum_{i=1}^n \sigma_i^2$ represents the total information contained in $\mathbf{A}$. Storing the first k singular-value triplets would capture $\sum_{i=1}^k \sigma_i^2$ information and lose the remaining $\sum_{i=k+1}^n \sigma_i^2$. This is the best approximation of $\mathbf{A}$ in Frobenius norm, as given by the TSVD. To quantify the efficiency of storing each singular-value triplet $(\mathbf{u}_i, \sigma_i, \mathbf{v}_i)$, we define the *information density* ρ_i as follows:

Definition 1. *The* information density ρ_i *of the singular-value triplet* $(\mathbf{u}_i, \sigma_i, \mathbf{v}_i)$ *is the ratio of the information content* σ_i^2 *to the storage cost* $m + n + 1$*, i.e.,*

$$\rho_i = \frac{\sigma_i^2}{m + n + 1}.$$

This measure is particularly useful when comparing matrices of different sizes. For instance, a 4×4 matrix with a singular value of 5 has a higher information density of 25/9

compared to a 8×8 matrix with a singular value of 6, which has an information density of $36/17$, less than the 4×4 matrix. This indicates why smaller matrices with higher information density are often preferred for storage when aiming to minimize memory usage while maximizing retained information.

Algorithm 1 Near-Optimal CHACI

Input: The sorted CI vector, number of partitioning levels p, and information density threshold ρ

Output: The CHACI format and storage information for each block

Split the CI vector into a corner hierarchical structure with $3p + 1$ blocks
Store the upper-left block in dense format
for each of the remaining $3p$ blocks **do**
    Let n_{row} and n_{col} be the numbers of rows and columns of the block
    $k \leftarrow 0$, $k_{\max} \leftarrow 1$, and perform rank-1 TSVD of the block
    while $k < \min\{n_{\text{row}}, n_{\text{col}}\}/2$ **do**
        if $k + 1 > k_{\max}$ **then**
            $k_{\max} \leftarrow \min\{2k_{\max}, n_{\text{row}}/2, n_{\text{col}}/2\}$
            Update the TSVD of this block to rank $k_{\max}$
        end if
        if $\sigma_{k+1}^2 \leq \rho(n_{\text{row}} + n_{\text{col}} + 1)$ **then**
            break
        end if
        $k \leftarrow k + 1$
    end while
    if $k = 0$ **then**
        Do not store the block
    else if $k = \min\{n_{\text{row}}, n_{\text{col}}\}/2$ **then**
        Store the block in dense format
    else
        Store the block in TSVD format with rank k
    end if
end for

A.2. Near-Optimal CHACI Algorithm

Based on the concept of information density, we propose our CHACI algorithm, which strategically chooses between dense and sparse formats for each block. The "near-optimal" descriptor emphasizes that among all CHACI configurations, this algorithm seeks to minimize memory usage within a small margin of the optimal solution that achieves the same accuracy. Algorithm 1 outlines this near-optimal CHACI algorithm. This algorithm takes the number of levels p as an input, which should be approximately logarithmic in M. Based on our experiments, we choose p to be the smallest integer that satisfies $k_0 2^p \geq M$, where $k_0 = 6$, which can be computed recursively. The algorithm includes a mechanism to switch to dense storage for small blocks. It also takes a threshold for the information density as input, and varying this threshold would result in different storage requirement, as we did in our numerical tests in Section III. It is possible to change the input to be a memory budget instead, and in this setting CHACI possesses some optimality properties, as we analyze next.

A.3. Analysis of Near Optimality

In Section III, we showed numerically that CHACI uses less memory than TSVD. To help understand why this is the case, we present a theoretical analysis of the CHACI algorithm. For simplicity and clarity, we make some simplifying assumptions in our analysis. First, we assume that each block is square, which is the case if the input matrix is a square matrix and its number of rows (and of columns) is a power of two. Second, we change the for and while loops in the algorithm to be in descending order of the information density among all the blocks. This modification requires maintaining a priority queue of the singular triplets of all the blocks, but it does not significantly affect the accuracy and performance compared to using the for-while loops in Algorithm 1.

Consider a CHACI-partitioned matrix with p levels, resulting in $3p + 1$ blocks, each represented by an SVD. Denote these blocks as $\mathbf{B}_1$ to $\mathbf{B}_{3p+1}$, with each block $\mathbf{B}_i$ having a size m_i. Each singular value $\sigma_{i,j}$ from block $\mathbf{B}_i$ contributes to the overall matrix's Frobenius norm squared, and hence

$$\|\mathbf{A}\|_F^2 = \sum_{i=1}^{3p+1} \sum_{j=1}^{m_i} \sigma_{i,j}^2.$$

Storing all singular-value triplets provides complete information, but practical constraints

often require storage optimization. Therefore, we compute the information density for each block and prioritize those with the highest values to minimize storage while maintaining accuracy. Let s_i represent the storage of the singular-value triplets and ρ_i the information density. The following lemma motivates us to minimize storage by sorting the singular-value triplets based on ρ_i:

Lemma 1. *Given a collection of positive-number pairs* $\{(s_i, \rho_i) \mid 1 \leq i \leq n\}$*, where* ρ_i *is sorted in descending order, for an arbitrary selection of* k *pairs* $(s_{j_1}, \rho_{j_1}), \ldots, (s_{j_k}, \rho_{j_k})$*, let* r *be a number between* 1 *and* n *such that*

$$\sum_{i=1}^{r-1} s_i < \sum_{i=1}^{k} s_{j_i} \leq \sum_{i=1}^{r} s_i,$$

it then follows that

$$\sum_{i=1}^{r} s_i \rho_i \geq \sum_{i=1}^{k} s_{j_i} \rho_{j_i}.$$

Proof. Let $S = \sum_{i=1}^{r}(s_i)$. Define a piecewise function $f(x)$ over $0 < x \leq S$ as

$$f(x) = \begin{cases} \rho_1 & \text{if } 0 < x \leq s_1, \\ \rho_2 & \text{if } s_1 < x \leq s_1 + s_2, \\ \vdots & \\ \rho_r & \text{if } \sum_{i=1}^{r-1} s_i < x \leq \sum_{i=1}^{r} s_i, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$\sum_{i=1}^{r} s_i \rho_i = \int_0^S f(x)\, dx.$$

Sorting the ρ_{j_i} in descending order, or equivalently sorting j_i in ascending order, we redefine the pairs as (s_{J_i}, ρ_{J_i}), where J_i is the ith smallest element in the set $\{j_i\}$. Define $\hat{f}(x)$ similarly as $f(x)$ but with ρ_{J_i} in place of ρ_i, i.e.,

$$\hat{f}(x) = \begin{cases} \rho_{J_q} & \text{if } \sum_{i=1}^{q-1} s_{J_i} < x \leq \sum_{i=1}^{q} s_{J_i} \text{ for } 1 \leq q \leq m, \\ 0 & \text{otherwise.} \end{cases}$$

Since ρ_i is sorted in descending order, we have $f(x) \geq \hat{f}(x)$ for all $x \geq 0$. Therefore,

$$\sum_{i=1}^{r} s_i \rho_i - \sum_{i=1}^{m} s_{j_i} \rho_{j_i} = \int_0^S (f(x) - \hat{f}(x))\, dx \geq 0.$$

□

Lemma 1 implies that by sorting the singular-value triplets based on their information densities, we can maximize the information retained for a given storage cost. This result suggests a greedy algorithm that selects the singular values with the highest information densities first, thereby minimizing storage while maintaining the desired accuracy. However, such a greedy strategy may not always yield the optimal solution for a given accuracy requirement. The following theorem provides a bound on the storage difference between the greedy algorithm and the optimal solution.

Theorem 1. *Given a collection of positive-number pairs* $\{(s_i, \rho_i) \mid 1 \le i \le n\}$ *as in Lemma 1 and* $t \in (0, \sum_i^n s_i]$*, for an arbitrary selection of* $k \le n$ *pairs* $(s_{j_1}, \rho_{j_1}), \ldots, (s_{j_k}, \rho_{j_k})$ *such that* $\sum_{i=1}^k s_{j_i}\rho_{j_i} \ge t$*, let* r *be the number of triplets such that*

$$\sum_{i=1}^{r-1} s_i\rho_i < t \le \sum_{i=1}^{r} s_i\rho_i,$$

it then follows that

$$\sum_{i=1}^{r} s_i - \sum_{i=1}^{k} s_{j_i} < N + 2,$$

where N *is the number of rows in the matrix. In other words, the storage difference between the greedy algorithm and the optimal CHACI solution is bounded by a number no more than* $N + 2$*.*

Proof. Assume that the selected pairs (s_{j_i}, ρ_{j_i}) satisfy that the storage

$$\sum_i s_{j_i} \le \sum_{i=1}^{r-1} s_i.$$

Then by Lemma 1,

$$\sum_i s_{j_i}\rho_{j_i} \le \sum_{i=1}^{r-1} s_i\rho_i < t,$$

which conflicts with the fact that this selection satisfies the requirement

$$\sum_i s_{j_i} \cdot \rho_{j_i} \ge t.$$

Thus we have

$$\sum_i s_{j_i} > \sum_{i=1}^{r-1} s_i,$$

so

$$\sum_{i=1}^{r} s_i - \sum_i s_{j_i} < \sum_{i=1}^{r} s_i - \sum_{i=1}^{r-1} s_i = s_r \le \max_{i=1}^{r}\{s_i\}.$$

Since the largest block is of size $\lceil N/2 \rceil \times \lceil N/2 \rceil$, we have $\max s_i \le N + 2$. □

In practice, the last block s_r is even much smaller than $N + 2$. Hence, the storage difference between the greedy algorithm and the optimal solution is bounded by a number typically far smaller than N. Therefore, CHACI is nearly optimal in its storage requirement. In the following, we further show that CHACI typically requires less memory and is more efficient than applying the TSVD to the global matrix.

A.4. Analysis of Complexity

The CHACI algorithms assume that the input CI vector is fully represented, requiring N^2 storage. The extra intermediate storage of the algorithm is about the same as the input size. Hence, we focus on the output's storage requirement, a common approach for compression techniques.

For each block i of size $m_i \times m_i$, if stored in TSVD format with rank k_i, the storage requirement includes $2k_im_i$ for the vectors (each vector has m_i entries) and k_i for the singular values, totaling $2k_im_i + k_i$. If a block is stored in a dense format, its requirement is m_i^2. Thus, the total storage for the output is

$$\text{Total output size} = \sum_{\text{TSVD block } i} (2k_im_i + k_i) + \sum_{\text{dense block } i} m_i^2.$$

It is worth noting that the dense blocks are typically for the smaller blocks, and the total storage is dominated by the TSVD blocks. Note that

$$\sum_{i=1}^{3p+1} m_i \leq \sum_{j=1}^{p} \frac{3N}{2^j} + \left\lceil \frac{N}{2^p} \right\rceil \leq 3N + 1,$$

and

$$\sum_{i=1}^{3p+1} (2m_i + 1) \leq 6N + 6p + 3 = 6N + o(N).$$

For simplicity, assume all the blocks are stored in TSVD format. The output can be bounded by

$$\text{Total output size} \lesssim 6N \max\{k_i\}$$

for sufficiently large N. This bound is tight if the ranks $\{k_i\}$ are uniform, but it is pessimistic if the ranks are highly non-uniform, as they are in practice for CI vectors.

Now we compare the storage requirement to that of global TSVD. A global TSVD for the CI vector traditionally stores K singular values and vectors, resulting in an output storage

of $2KN + K$. Hence, if the input matrix were a uniform low-rank matrix, CHACI would require more storage than the global TSVD.[71] However, CHACI is designed to exploit the structure of CI vectors, which are not uniformly low-rank. In our numerical experimentation, the output size of near-optimal CHACI is typically about 2% of that of a global TSVD of equal accuracy.

The time complexity of the near-optimal CHACI algorithm involves computing TSVDs for various blocks. For each block i of size $m_i \times m_i$, a rank-k_i TSVD can be approximately computed in $\mathcal{O}(k_i m_i^2)$ using iterative methods. An important detail worth emphasizing is Line 8 of Algorithm 1. In this step, we would double the TSVD rank if the desired accuracy is not met. Such a doubling process results in a geometric progression of the TSVD computation cost, calculated as follows:

$$\sum_{j=1}^{\lceil \log_2 k_i \rceil} 2^j m_i^2 \leq 4 k_i m_i^2.$$

Consequently, the total time complexity for all blocks in the CHACI algorithm is

$$\text{Total time complexity} = \sum_{i=1}^{3p+1} \mathcal{O}(k_i m_i^2),$$

where $\max\{m_i\} = N/2$. If the rank increases were linear when recomputing the TSVD, the worst-case time complexity could potentially reach $\mathcal{O}(k_i^2 m_i^2)$ for each block, potentially increasing the overall time complexity.

Now we compare the time complexity to that of global TSVD. Performing a global TSVD of an $N \times N$ matrix with a truncation rank K incurs a time complexity of $\mathcal{O}(KN^2)$. Since

$$\sum_{i=1}^{3p+1} m_i^2 \leq \sum_{j=1}^{\lceil \log N \rceil} 3 \frac{N^2}{2^{2j}} \leq N^2.$$

As long as $k_i < K$, the near-optimal CHACI algorithm has a lower computational cost than a global TSVD. In practice, $k_i \ll K$ in the near-optimal CHACI algorithm, as we will demonstrate in our numerical experimentation. Hence, we can expect the near-optimal CHACI algorithm to be significantly faster than a global TSVD for moderately large N.

REFERENCES

[1]B. O. Roos, P. R. Taylor, and P. E. M. Siegbahn, Chemical Physics **48**, 157 (1980).

[2]L. Bytautas and K. Ruedenberg, Chemical Physics **356**, 64 (2009).

[3]P. A. Malmqvist, A. Rendell, and B. O. Roos, Journal of Physical Chemistry **94**, 5477 (1990).

[4]T. Fleig, J. Olsen, and C. M. Marian, Journal of Chemical Physics **114**, 4775 (2001).

[5]T. Fleig, J. Olsen, and L. Visscher, Journal of Chemical Physics **119**, 2963 (2003).

[6]D. X. Ma, G. Li Manni, and L. Gagliardi, Journal of Chemical Physics **135** (2011).

[7]D. Casanova, Wiley Interdisciplinary Reviews-Computational Molecular Science **12** (2022).

[8]S. M. Parker, T. Seideman, M. A. Ratner, and T. Shiozaki, Journal of Chemical Physics **139** (2013).

[9]J. Ivanic, Journal of Chemical Physics **119**, 9364 (2003).

[10]B. Huron, J. P. Malrieu, and P. Rancurel, Journal of Chemical Physics **58**, 5745 (1973).

[11]R. J. Harrison, Journal of Chemical Physics **94**, 5021 (1991).

[12]J. P. Daudey, J. L. Heully, and J. P. Malrieu, Journal of Chemical Physics **99**, 1240 (1993).

[13]J. C. Greer, Journal of Chemical Physics **103**, 1821 (1995).

[14]J. C. Greer, Journal of Chemical Physics **103**, 7996 (1995).

[15]J. P. Coe and M. J. Paterson, Journal of Chemical Physics **141** (2014).

[16]J. P. Coe, P. Murphy, and M. J. Paterson, Chemical Physics Letters **604**, 46 (2014).

[17]F. A. Evangelista, Journal of Chemical Physics **140** (2014).

[18]J. P. Coe, Journal of Chemical Theory and Computation **14**, 5739 (2018).

[19]A. A. Holmes, N. M. Tubman, and C. J. Umrigar, Journal of Chemical Theory and Computation **12**, 3674 (2016).

[20]J. B. Schriber and F. A. Evangelista, Journal of Chemical Physics **144** (2016).

[21]Y. Ohtsuka and J. Y. Hasegawa, Journal of Chemical Physics **147** (2017).

[22]N. M. Tubman, C. D. Freeman, D. S. Levine, D. Hait, M. Head-Gordon, and K. B. Whaley, Journal of Chemical Theory and Computation **16**, 2139 (2020).

[23]J. J. Goings, H. Hu, C. Yang, and X. S. Li, Journal of Chemical Theory and Computation **17**, 5482 (2021).

[24]F. Neese, Journal of Chemical Physics **119**, 9428 (2003).

[25]H. Nakatsuji and M. Ehara, Journal of Chemical Physics **122** (2005).

[26]M. L. Abrams and C. D. Sherrill, Chemical Physics Letters **412**, 121 (2005).

[27]W. J. Liu and M. R. Hoffmann, Journal of Chemical Theory and Computation **12**, 1169 (2016).

[28]A. A. Holmes, C. J. Umrigar, and S. Sharma, Journal of Chemical Physics **147** (2017).

[29]J. H. Li, M. Otten, A. A. Holmes, S. Sharma, and C. J. Umrigar, Journal of Chemical Physics **149** (2018).

[30]V. Abraham and N. J. Mayhall, Journal of Chemical Theory and Computation **16**, 6098 (2020).

[31]L. Bytautas, T. M. Henderson, C. A. Jiménez-Hoyos, J. K. Ellis, and G. E. Scuseria, Journal of Chemical Physics **135** (2011).

[32]G. H. Booth, A. J. W. Thom, and A. Alavi, The Journal of Chemical Physics **131**, 054106 (2009).

[33]J. J. Shepherd, G. E. Scuseria, and J. S. Spencer, Physical Review B **90**, 155130 (2014).

[34]Z.-X. Li and H. Yao, Annual Review of Condensed Matter Physics **10**, 337 (2019).

[35]G. Li Manni, W. Dobrautz, and A. Alavi, Journal of Chemical Theory and Computation **16**, 2202 (2020).

[36]S. R. White, Physical Review Letters **69**, 2863 (1992).

[37]G. K. L. Chan and S. Sharma, "The density matrix renormalization group in quantum chemistry," in *Annual Review of Physical Chemistry*, Vol. 62, edited by S. R. Leone, P. S. Cremer, J. T. Groves, and M. A. Johnson (2011) pp. 465–481.

[38]F. Verstraete, T. Nishino, U. Schollwöck, M. C. Bañuls, G. K. Chan, and M. E. Stoudenmire, Nature Reviews Physics **5**, 273 (2023).

[39]J. Olsen, P.-A. Malmqvist, B. O. Roos, R. Lindh, and P.-O. Widmark, Chemical Physics Letters **133**, 91 (1987).

[40]R. Lindh, J. Olsen, and B. O. Roos, Chemical Physics Letters **148**, 276 (1988).

[41]H. Koch and E. Dalgaard, Chemical Physics Letters **198**, 51 (1992).

[42]P. R. Taylor, Journal of Chemical Physics **139** (2013).

[43]B. S. Fales, S. Seritan, N. F. Settje, B. G. Levine, H. Koch, and T. J. Martínez, Journal of Chemical Theory and Computation **14**, 4139 (2018).

[44]P. J. Knowles, Molecular Physics **113**, 1655 (2015).

[45]W. Kohn and L. J. Sham, Physical Review **140**, 1133 (1965).

[46]A. J. Coleman, Reviews of Modern Physics **35**, 668 (1963).

[47]S. Sharma, J. K. Dewhurst, N. N. Lathiotakis, and E. K. U. Gross, Physical Review B **78** (2008).

[48]D. A. Mazziotti, Chemical Reviews **112**, 244 (2012).

[49]N. H. Stair and F. A. Evangelista, Journal of Chemical Physics **153** (2020).

[50]J. J. Eriksen, Journal of Physical Chemistry Letters **12**, 418 (2021).

[51]S. Börmm, L. Grasedyck, and W. Hackbusch, Engineering Analysis with Boundary Elements **27**, 405 (2003).

[52]L. Lin, J. Lu, and L. Ying, Journal of Computational Physics **230**, 4071 (2011).

[53]W. Hackbusch, *Hierarchical Matrices: Algorithms and Analysis* (2015).

[54]X. Xing and E. Chow, SIAM Journal on Scientific Computing **42**, A162 (2020).

[55]X. Xing, H. Huang, and E. Chow, The Journal of Chemical Physics **153**, 084119 (2020).

[56]H. Huang, X. Xing, and E. Chow, ACM Transactions on Mathematical Software **47** (2021).

[57]J. Barnes and P. Hut, Nature **324**, 446 (1986).

[58]L. Greengard and V. Rokhlin, Journal of Computational Physics **73**, 325 (1987).

[59]M. C. Strain, G. E. Scuseria, and M. J. Frisch, Science **271**, 51 (1996).

[60]W. Duch, Chemical Physics Letters **124**, 442 (1986).

[61]W. Hackbusch, Computing **62**, 89 (1999).

[62]F. Eisenhut, T. Kühne, F. García, S. Fernández, E. Guitian, D. Pérez, G. Trinquier, G. Cuniberti, C. Joachim, D. Peña, and F. Moresco, ACS Nano **14**, 1011 (2020).

[63]T. Yanai, D. P. Tew, and N. C. Handy, Chemical Physics Letters **393**, 51 (2004).

[64]G. Granucci and A. Toniolo, Chemical Physics Letters **325**, 79 (2000).

[65]P. Slavíček and T. J. Martínez, Journal of Chemical Physics **132** (2010).

[66]I. S. Ufimtsev and T. J. Martinez, Journal of Chemical Theory and Computation **5**, 2619 (2009).

[67]S. Seritan, C. Bannwarth, B. S. Fales, E. G. Hohenstein, C. M. Isborn, S. I. L. Kokkila-Schumacher, X. Li, F. Liu, N. Luehr, J. W. Snyder, C. C. Song, A. V. Titov, I. S. Ufimtsev, L. P. Wang, and T. J. Martínez, Wiley Interdisciplinary Reviews-Computational Molecular Science **11** (2021).

[68]B. S. Fales and B. G. Levine, Journal of Chemical Theory and Computation **11**, 4708 (2015).

[69]E. G. Hohenstein, M. E. F. Bouduban, C. C. Song, N. Luehr, I. S. Ufimtsev, and T. J. Martínez, Journal of Chemical Physics **143** (2015).

[70]B. S. Fales, E. G. Hohenstein, and B. G. Levine, Journal of Chemical Theory and Computation **13**, 4162 (2017).

[71]H-matrices would also require more storage than the global TSVD if the input matrix is a uniform low-rank matrix.