JCTC Journal of Chemical Theory and Computation
Article
pubs.acs.org/JCTC

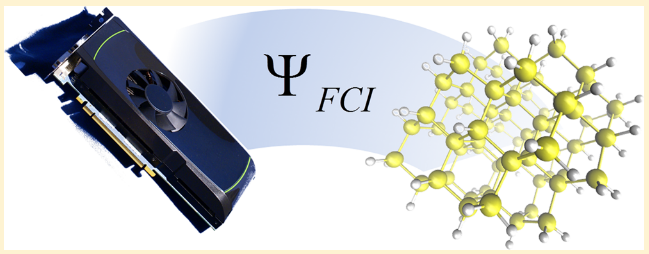
Nanoscale Multireference Quantum Chemistry: Full Configuration Interaction on Graphical Processing Units

B. Scott Fales and Benjamin G. Levine*

Department of Chemistry, Michigan State University, East Lansing, Michigan 48824, United States

Supporting Information

ABSTRACT: Methods based on a full configuration interaction (FCI) expansion in an active space of orbitals are widely used for modeling chemical phenomena such as bond breaking, multiply excited states, and conical intersections in small-to-medium-sized molecules, but these phenomena occur in systems of all sizes. To scale such calculations up to the nanoscale, we have developed an implementation of FCI in which electron repulsion integral transformation and several of the more expensive steps in σ vector formation are performed on graphical processing unit (GPU) hardware. When applied to a 1.7 × 1.4 × 1.4 nm silicon nanoparticle ($Si_{72}H_{64}$) described with the polarized, all-electron 6-31G** basis set, our implementation can solve for the ground state of the 16-active-electron/16-active-orbital CASCI Hamiltonian (more than 100,000,000 configurations) in 39 min on a single NVidia K40 GPU.



1. INTRODUCTION

Computational science has recently seen a resurgence in the vector processing paradigm due to the availability of low-cost video acceleration coprocessors tailored for the gaming market known as graphical processing units (GPUs). With a large number of floating point units operating in parallel and a fast hierarchical memory subsystem, GPUs provide computational power comparable to that of a small cluster of CPU-based machines at a cost of only a few hundreds or thousands of dollars. Electronic structure calculations are particularly amenable to GPU acceleration because many of the relevant algorithms are formulated in terms of matrix operations which can be efficiently vectorized. Applications of GPUs to electron repulsion integral evaluation and the self-consistent field procedure have proven fruitful,[1−14] and correlated, linear-scaling, and excited state methods have also been accelerated by GPUs.[15−26] Production calculations which take advantage of GPUs for the study of complex nanoscale or biological systems or for automated discovery of new chemistries are now regularly reported.[27−31]

To date most GPU-accelerated development efforts in ab initio quantum chemistry have focused on single-reference methods, though exceptions include recent implementations of the complete active space self-consistent field (CASSCF) method[32] and multireference coupled cluster theory.[33] Many chemical phenomena, such as bond breaking, multiply excited electronic states, and nonradiative decay via conical intersections, benefit from treatment at a multireference level of theory,[34,35] as evidenced by the popularity of CASSCF, which is the standard tool used for calculating zeroth-order wave functions of such strongly correlated systems.[36] Inspired by the success of CASSCF, a class of efficient and accurate alternatives has been introduced: the *two-step* complete active space configuration interaction (CASCI) methods, in which the optimization of the orbitals is decoupled from that of the configuration interaction (CI) coefficients. The various two-step methods are differentiated by the procedure used to determine the orbitals on which the CI expansion is based. Orbital determination schemes suitable for describing bond breaking include the unrestricted natural orbital (UNO) method[37] and methods based on the natural orbitals of correlated single-reference methods.[38] Two-step methods have been gaining popularity for the description of excited states as well, and various schemes exist: improved virtual orbitals (IVO),[39,40] floating occupation molecular orbitals (FOMO),[41,42] high-multiplicity natural orbitals (HMNO),[43,44] variationally optimal orbitals from restricted active space calculations (singly excited active space; SEAS),[45] and configuration interaction singles natural orbitals (CISNO).[46] In addition to computational efficiency, two-step methods often exhibit other desirable behaviors, such as a reduced propensity for spatial symmetry breaking,[45] size intensivity,[46] and simplification of the choice of active space.[46,47]

CASCI is simply full CI (FCI) in a user-defined active space of orbitals, and the last several decades have seen the development of many efficient FCI algorithms.[48] These *direct* algorithms achieve efficiency by circumventing the formation of the full Hamiltonian matrix. Though spin-adapted formulations significantly reduce the dimensionality of the FCI problem,[49,50] determinantal CI algorithms became popular in the 1980s due to the ease with which the coupling coefficients can be

Received: July 2, 2015
Published: September 14, 2015

evaluated.[51] Though several FCI algorithms with reduced operation counts have been developed,[52−54] the seminal algorithm introduced by Knowles and Handy (KH)[55] is an excellent starting point for a GPU implementation of FCI because of its amenability to vectorization and the regular memory access pattern it allows. Specifically, the performance limiting step of the KH algorithm is a large matrix multiplication, for which extremely efficient GPU algorithms exist.

Further acceleration of FCI or FCI-like calculations can be achieved in various ways. It is possible to improve performance by taking advantage of the sparsity of the data structures involved.[56−58] The density matrix renormalization group approach allows chemists to solve problems with larger active spaces than previously possible.[59−63] In systems with larger one-electron bases, performance may be determined not by the cost of the diagonalization itself but by the preceding two-electron integral transformation. The formal scaling of this step has been reduced by employing approximations to the integrals,[64−67] and reduced effective scaling has also been observed when efficient prescreening is employed in the absence of any additional approximation.[32,68]

Herein we report the implementation of a vectorized direct determinantal FCI algorithm on GPU hardware. Section 2 describes our adaptations of two potentially performance-limiting steps to GPU hardware: the two-electron integral transformation and the KH algorithm for formation of the σ vector. In section 3 we report the performance of our algorithm for systems as large as a 1.7 × 1.4 × 1.4 nm silicon nanoparticle. In section 4 we draw conclusions and discuss future prospects.

2. METHODS

A typical CASCI calculation requires a) determination of a suitable set of orbitals, b) transformation of integrals from the atomic orbital (AO) to the molecular orbital (MO) basis, and c) solving for the lowest eigenvalues and eigenstates of the CASCI Hamiltonian. Each of these steps, respectively, contain a potential bottleneck: a) solution of the Hartree−Fock (HF) wave function (or perhaps configuration interaction singles (CIS), in the case of CISNO), b) two-electron integral transformation, and c) formation of the σ vector. GPU algorithms for HF and CIS have been well documented.[1−3,18] Below we report the details of our GPU implementations of each of the latter two potential bottlenecks. Note that double precision floating point math was used for all calculations presented in this work, even though GPUs perform floating point math faster in single precision.

2.1. Integral Transformation. The transformation of the electron repulsion integrals (ERIs) is the performance-limiting step for a FCI calculation using a large single-electron basis and modest active space, so we focus on the GPU implementation of this step here. When fast integral evaluation is possible (as on the GPU), integral-direct calculation of the transformed integrals can provide good scaling without the need for storage of large matrices of intermediate sums.[68−70] In our implementation we take advantage of the existence of efficient GPU-accelerated code for computing generalized Coulomb matrices, $\mathbf{J}_{ij}$, to achieve these goals.[2] ($[J_{ij}]_{\mu\nu} = (\mu\nu|ij)$; two-electron integrals are expressed in chemists' notation, μ and ν index the AO basis, and i and j index the transformed (MO) basis.)

Our transformation proceeds as follows: The matrix $\mathbf{J}_{ij}$ is computed for a particular pair, ij, from a density matrix created by taking the outer product of molecular orbital i with orbital j. From $\mathbf{J}_{ij}$, a two-index transformation is performed to build the fully transformed integrals

$$(ij|kl) = \sum_{\mu\nu} C_{\mu k} C_{\nu l} [J_{ij}]_{\mu\nu} \tag{1}$$

where k and l (like i and j) index the MO basis, and $C_{\mu k}$ is the coefficient of basis function μ in MO k. Then, $\mathbf{J}_{ij}$ is discarded before computing $\mathbf{J}_{ij}$ for the next pair ij. Note that to perform a CASCI calculation i, j, k, and l need only span the space of active orbitals. Formally, the operation count of this calculation scales as $O(N^4m^2)$, where N and m are the number of AOs and active MOs, respectively, and thus $m \ll N$ in most cases. Only enough storage for $(ij|kl)$ (m^4) and a single instance of $\mathbf{J}_{ij}$ (N^2) is required. Further reduction of the formal scaling of the operation count to $O(N^4m)$ would be possible by taking an alternative approach: performing four successive one-index transforms rather than two two-index transforms as above. However, the observed scaling of the two-index algorithm, which will be analyzed in detail below, is considerably less than $O(N^4m^2)$ because we prescreen the two-electron integrals in the process of forming $\mathbf{J}_{ij}$, as described by Ufimtsev et al.[2]

2.2. KH Direct FCI. Direct CI methods require access only to the transformed integrals, the one-electron coupling coefficients, γ, and a matrix listing the nonzero elements of γ, $\mathbf{l}$. As noted above, we work in a determinantal basis to simplify evaluation of γ. As a result, computational performance depends strongly on how efficiently the integrals and coupling coefficients are accessed in memory. A lexical configuration ordering scheme allows for contiguous memory access, an important consideration for many high performance computing platforms but especially important for GPUs. By writing configurations as strings of α and β occupied orbitals and by taking advantage of their symmetry, γ and $\mathbf{l}$ can be accessed contiguously in all cases. The graphical method described by Duch[71] was used to map the addresses for the α and β strings and is functionally identical to the method described by KH.[55] Below we review the KH algorithm and then detail our GPU adaptation, with particular attention to memory access.

Our code uses the iterative Davidson solver[72] for diagonalization of the Hamiltonian matrix. The most computationally expensive step in this procedure is generally the formation of the σ vector

$$\sigma_I = \sum_J H_{IJ} c_J \tag{2}$$

where c_J is an element of the CI coefficient vector, H_{IJ} is a Hamiltonian matrix element, and I and J index determinants. For problems involving a large active space and small single electron basis, it is this step rather than the integral transformation that is performance-limiting. The Hamiltonian matrix elements can be represented as[73]

$$H_{IJ} = \sum_{ij}^{n} \left[\gamma_{ij}^{IJ}(i|\hat{h}|j) + \sum_k \gamma_{ik}^{IJ}(ij|jk) \right] + \frac{1}{2} \sum_{ijkl} \sum_K \gamma_{ij}^{IK} \gamma_{kl}^{KJ} (ij|kl) \tag{3}$$

where K (like I and J) indexes determinants, and $\hat{h}$ is the one-electron Hamiltonian. The one-electron coupling coefficients are defined

$$\gamma_{ij}^{IJ} = \langle \Psi_I | E_{ij} | \Psi_J \rangle \tag{4}$$

where E_{ij} is the excitation operator that promotes an electron from orbital j to orbital i, and Ψ_I is a determinant. In terms of α and β strings ($|\Psi_I\rangle = |\Psi_{\alpha\beta}\rangle$ and $|\Psi_J\rangle = |\Psi_{\alpha'\beta'}\rangle$), we could equivalently represent the coupling coefficients as

$$\gamma_{ij}^{\alpha\beta\alpha'\beta'} = \langle\Psi_{\alpha\beta}|E_{ij}|\Psi_{\alpha'\beta'}\rangle \tag{5}$$

Of course E_{ij} is a one-electron operator, and therefore $\gamma_{ij}^{\alpha\beta\alpha'\beta'}$ can only be nonzero when either $\alpha = \alpha'$ or $\beta = \beta'$. If $\beta = \beta'$, for example, $\gamma_{ij}^{\alpha\beta\alpha'\beta}$ is the same for all β, and thus the dimensionality of $\boldsymbol{\gamma}$ can be reduced, with all nonzero elements described by

$$\gamma_{ij}^{\alpha\alpha'} = \langle\Psi_{\alpha\beta}|E_{ij}|\Psi_{\alpha'\beta}\rangle \tag{6}$$

for all β and

$$\gamma_{ij}^{\beta\beta'} = \langle\Psi_{\alpha\beta}|E_{ij}|\Psi_{\alpha\beta'}\rangle \tag{7}$$

for all α. Further, for a given pair of strings, $\alpha \neq \alpha'$, $\gamma_{ij}^{\alpha\alpha'}$ is nonzero for at most a single pair of orbitals, ij. Thus, storage is saved by storing each nonzero element, $\gamma_{ij}^{\alpha\alpha'}$, in a pair of array elements: $l[\alpha,\alpha']$, which stores the value of ij for which $\gamma_{ij}^{\alpha\alpha'}$ is nonzero, and $\gamma[\alpha,\alpha']$ which stores the value of $\gamma_{ij}^{\alpha\alpha'}$ itself.

The two-electron term on the far right of eq 3 forms the bulk of the effort. In the KH algorithm, the contributions to σ from this term are calculated in a series of three steps[55]

$$D_{ij}^{K} = \sum_{J} \gamma_{ij}^{JK} c_J \tag{8a}$$

$$E_{kl}^{K} = \sum_{ij} \frac{1}{2}(ij|kl) D_{ij}^{K} \tag{8b}$$

$$\sigma_I = \sum_{K}\sum_{kl} \gamma_{kl}^{KI} E_{kl}^{K} \tag{8c}$$

Applying the definition of $\boldsymbol{\gamma}$ in eqs 6 and 7 to eqs 8a–8c, we arrive at the serial form of the FCI algorithm originally described by KH (Algorithm 1, see Chart 1).[55] This algorithm has five steps: calculation of the α and β contributions to the **D** matrix, formation of the **E** matrix (the central matrix–matrix multiply), and calculation of the α and β contributions to the σ vector. Eq 8b is simply a matrix–matrix multiplication, and it is this step that dominates the operation count of the KH algorithm. This operation is trivially ported to the GPU by a call to the CUBLAS library,[74] which contains a highly optimized matrix–matrix multiplication routine. We treat this as a dense matrix operation in the present implementation. The **D** matrix can be quite sparse in the first several iterations of the CI procedure, however,[73] and accounting for this sparsity to further reduce the computational effort is an area of active development. Naive implementation of the lower-scaling operations in eqs 8a and 8c results in performance that is limited by these steps. Thus, efficient implementation of these lower-scaling steps is essential for high performance, and it is the GPU implementation of these that will be presented below.

The nested loop structures that form **D** and σ are similar. Each comprises three nested loops over strings, with the string corresponding to the innermost loop differing in spin from those of the outer two loops. In what follows, we consider in detail the GPU implementation of the first set of nested loops in Algorithm 1 (see Chart 1): the calculation of the α contributions to **D**. The GPU algorithm for this step is presented in Algorithm 2 (see Chart 2). Though we do not explicitly present the GPU algorithms for the calculations of the

Chart 1. Algorithm 1: Serial Knowles and Handy Algorithm

```
for strings $\alpha$ do
    for strings $\alpha'$ differing from $\alpha$ by zero or one occupations do
        for strings $\beta$ do
            $ij \leftarrow l[\alpha,\alpha']$
            $D[ij,\alpha,\beta] += \gamma[\alpha,\alpha']C[\alpha',\beta]$
        end
    end
end
for strings $\beta$ do
    for strings $\beta'$ differing from $\beta$ by zero or one occupations do
        for strings $\alpha$ do
            $ij \leftarrow l[\beta,\beta']$
            $D[ij,\alpha,\beta] += \gamma[\beta,\beta']C[\alpha,\beta']$
        end
    end
end
matrix multiply: $E[kl,\alpha,\beta] = (ij|kl)D[ij,\alpha,\beta]$
for strings $\alpha$ do
    for strings $\alpha'$ differing from $\alpha$ by zero or one occupations do
        for strings $\beta$ do
            $ij \leftarrow l[\alpha,\alpha']$
            $\sigma[ij,\alpha',\beta] += \gamma[\alpha,\alpha']E[ij,\alpha,\beta]$
        end
    end
end
for strings $\beta$ do
    for strings $\beta'$ differing from $\beta$ by zero or one occupations do
        for strings $\alpha$ do
            $ij \leftarrow l[\beta,\beta']$
            $\sigma[ij,\alpha,\beta'] += \gamma[\beta,\beta']E[ij,\alpha,\beta]$
        end
    end
end
```

Chart 2. Algorithm 2: GPU Vectorized Calculation of the α Terms in D

```
GPU vectorize over strings $\alpha,\beta$
    for all orbitals $i$ occupied in $\alpha$ do
        $D[ii,\alpha,\beta] += C[\alpha,\beta]$
    end
end GPU vectorize
GPU vectorize over strings $\alpha,\alpha',\beta$, where $\alpha$ and $\alpha'$ differ by exactly one occupation
    $ij \leftarrow l[\alpha,\alpha']$
    $D[ij,\alpha,\beta] += \gamma[\alpha,\alpha']C[\alpha',\beta]$
end GPU vectorize
```

β contributions to **D** and both components of σ, their implementation is analogous to that of the α contributions to **D**. Key differences will be noted below.

The expressions for the diagonal ($\alpha = \alpha'$) and off-diagonal ($\alpha \neq \alpha'$) contributions to the matrix elements for **D** and σ differ, and as such we split the calculation of these two sets of terms into two different GPU kernels. In the computation of the diagonal terms (first block in Algorithm 2 (see Chart 2)), the calculation is vectorized across α and β. Because $\alpha = \alpha'$, no loop over α' is required, and this term is low enough scaling so as to never be performance limiting. We therefore focus our attention on the efficient calculation of the off-diagonal terms (second block in Algorithm 2 (see Chart 2)). Here $\alpha \neq \alpha'$ so we vectorize over a third string, α', in addition to α and β. However, while we must compute terms for all possible values of α and β, we take advantage of the sparse nature of γ by including only those α' for which there is some nonzero $\gamma[\alpha,\alpha']$. Each thread loads a coupling coefficient, $\gamma[\alpha,\alpha']$ (along with its corresponding $l[\alpha,\alpha']$), and CI coefficient, $C[\alpha',\beta]$, multiplies them, and adds the product to element $D[ij,\alpha,\beta]$. For optimal performance, it would be desirable to access memory in one of two ways: such that contiguous memory is simultaneously accessed by neighboring threads (*coalesced* memory access, in GPU parlance) or such that blocks of threads all access exactly the same element of memory. An optimal memory access pattern is achieved through careful ordering of array and thread indices. Figure 1 depicts the arrangement of data structures whose efficient access is critical to algorithm performance. Note that the **D** and **E** matrices are ordered using the same convention, as are the **C** and σ data structures. Access to $D[ij,\alpha,\beta]$ and $C[\alpha',\beta]$ are coalesced, and each thread block requires access to only a single value of **l** and γ.

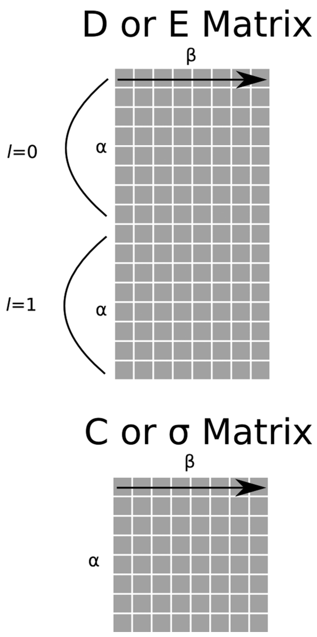


Figure 1. Memory layout for relevant data structures. The contiguous memory direction is defined here as being row-major, as indicated by the arrow, keeping with the C programming language convention. We depict the number of $l = ij$ pairs here as being 2, while in practice the number of ij pairs is m^2.

The off-diagonal β contributions to **D** are calculated in a similar fashion to the α contributions. The thread indices are reorganized such that $C[\alpha,\beta']$, $l[\beta,\beta']$, and $\gamma[\beta,\beta']$ are accessed in a coalesced fashion. Access to $D[ij,\alpha,\beta]$ itself is not coalesced, however. In an attempt to eliminate this noncoalesced memory access, we have implemented a second algorithm in which **D** is transposed between the calculation of the α and β terms. It was found that the prior approach provides better performance than the latter because the transpose of **D** requires an inefficient memory access pattern similar to that required by the noncoalesced algorithm. Therefore, it is the noncoalesced algorithm that was used for calculation of the β terms of **D**, though utilization of a more optimized transpose algorithm may be worth further investigation.[75]

The α and β contributions to the σ vector are computed using identical indexing patterns and vectorization schemes to their **D** counterparts, but there is one significant difference. Formation of σ requires several threads to sum into the same location of memory which, implemented naively, results in a race condition. The solution is to employ an atomic add operation to ensure inclusion of all contributions to the final σ vector. While hardware support for double precision floating point atomic addition is not currently available on NVidia GPU hardware, we were able to achieve the same functionality using a custom kernel that invokes the atomic compare-and-swap library function to ensure synchronized memory access across thread blocks.[76] A similar approach has previously been used in the implementation of radial distribution function histograms on GPU hardware.[77] Finally, we have implemented the one-electron contributions to the σ vector using a GPU vectorized approach very similar to that which was taken for the two-electron contributions.

2.3. Computational Details. The above-described direct FCI algorithm was implemented for NVidia GPU hardware in a development version of the TeraChem software package[3] using the Compute Unified Device Architechure (CUDA) API.[76] All benchmark calculations were performed on a system comprising an Intel Xeon E5-2680 2.80 GHz processor and a single NVidia K40 GPU. The accuracy of our code was established by comparison to identical calculations performed using the Molpro software package.[55,68,78−81] Except where noted, all CASCI calculations in this work were performed in the basis of canonical Hartree−Fock orbitals for simplicity, but note that the orbital determination steps for CISNO, IVO, and FOMO have all been implemented either by our group or by Hohenstein et al.[32,46] and will be available in a future release of TeraChem. Throughout this work active spaces are abbreviated (n,m) where n is the number of active electrons and m is the number of active orbitals. Except where noted, all calculations use the 6-31G** basis and thus describe all electrons explicitly and include polarization functions on all atoms. Reported times-to-solution include the integral transformation and direct diagonalization but exclude any orbital determination step. HF times-to-solution, which are comparable to orbital determination times-to-solution in many cases, have been reported separately for the reader's convenience.

3. RESULTS AND DISCUSSION

The test set of molecules used in our benchmark calculations is shown in Figure 2. It includes one of the classic systems in molecular photodynamics (pyrazine), two models of graphitic carbon nitride photocatalyst (dimelamine and dimelem), and

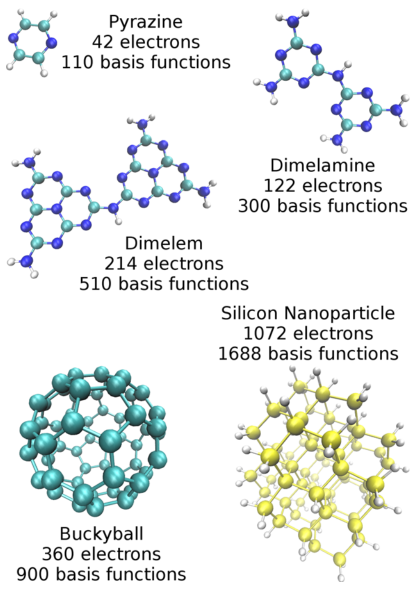


Figure 2. Test set molecules. Carbon, nitrogen, silicon, and hydrogen atoms are shown in teal, blue, yellow, and white, respectively. The numbers of basis functions correspond to the 6-31G** basis.

two nanoscale systems (C_{60} and $Si_{72}H_{64}$). Note that $Si_{72}H_{64}$ is a silicon nanoparticle measuring 1.7 × 1.4 × 1.4 nm.

3.1. Overall Performance. The total times-to-solution for CASCI/6-31G** calculations of the test set molecules (excluding any orbital determination step) are reported in Figure 3a and Table 1 with active spaces ranging from (6,6) to (16,16). The times range from subsecond (0.51 s for pyrazine with a (6,6) active space) to just over half an hour (2343.93 s = 39 min for $Si_{72}H_{64}$ with a (16,16) active space). Note that for the (6,6) active space the CASCI calculation is always faster than the HF calculation required to determine the orbitals. Even for the (12,12) active space, solving the HF equations requires more time than the subsequent CI calculation for the larger systems (dimelem, C_{60}, and $Si_{72}H_{64}$).

There is a very noticeable difference in scaling behavior for the different active spaces. This is more easily understood if we consider the percentages of the total times-to-solution required to perform the ERI transformations, which are reported in Figure 3b and in parentheses in Table 2. As expected, the ERI cost dominates when large molecules are treated with small active spaces, and the CI (which consumes the remainder of the time) dominates when small molecules are treated with large active spaces. The ERI transformation requires more than 90% of the total time-to-solution for all systems when they are computed with the small (6,6) active space and consumes more than half of the time for all but the smallest system (pyrazine) when the (12,12) space is used. The ERI transformation requires 87% of the time-to-solution for the largest reported calculation ($Si_{72}H_{64}$, (16,16)). For these systems where the performance is dominated by the ERI transformation the total time-to-solution appears to scale strongly with system size. Only when smaller molecules are treated with the (16,16) active space does the CI diagonalization dominate. In these cases there is no apparent dependence of performance on the size of the basis set, which is consistent with the fact that the CI diagonalization dominates the cost of the calculation. Instead, it is observed that the number of CI iterations (reported in parentheses in Table 1) determines the relative time-to-solution.

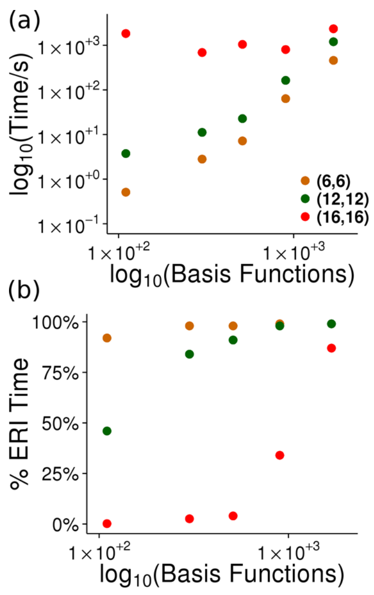


Figure 3. CASCI/6-31G** times-to-solution (a) and percent of total time spent performing the ERI transformation (b) for our test set molecules plotted as a function of the number of single-electron basis functions (N). Results for the (6,6), (12,12), and (16,16) active spaces are shown in orange, green, and red, respectively.

Table 1. CASCI/6-31G Times-to-Solution Using (6,6), (12,12), and (16,16) Active Spaces**[a]

	time-to-solution (s)			
system	(6,6)	(12,12)	(16,16)	HF
pyrazine	0.51 (5)	3.75 (7)	1834.74 (11)	1.14
dimelamine	2.81 (5)	11.17 (6)	689.23 (6)	8.48
dimelem	7.19 (4)	22.75 (6)	1047.50 (7)	31.36
C_{60}	63.69 (3)	163.42 (5)	807.82 (6)	258.01
$Si_{72}H_{64}$	458.42 (1)	1200.40 (2)	2343.93 (2)	1595.36

[a]The numbers of CI iterations required for convergence are shown in parentheses. Times-to-solution for the HF step, which is required for orbital determination but is not included in the CASCI times reported here, are shown in the final column for comparison.

3.2. ERI Transformation Performance. The times required to compute the transformed two-electron integrals for our test set molecules using three different size active spaces ((6,6), (12,12), and (16,16)) are presented in Figure 4 and Table 2. These times range from subsecond (0.47 s for pyrazine, m = 6) to roughly half an hour (2043.81 s = 34 min for $Si_{72}H_{64}$, m = 16). Though the direct ERI transformation procedure described above formally scales as $O(N^4m^2)$, it is worthwhile to investigate the scaling observed in practice, as the observed scaling may be significantly more favorable than the theoretical scaling when the integrals are prescreened.[32,68] Upon attempting to fit the data in Figure 4 to extract the effective scaling exponent we noted that a linear function fit the data very poorly. We hypothesized that this is because the planar graphitic carbon nitride systems exhibit different scaling behavior than denser, three-dimensional structures, like the

Table 2. Comparison of the Times Required To Perform ERI Transformations with Different Sized Active Spaces and Single-Electron Basis Sets[a]

		ERI transformation time (s)		
system	no. of basis functions	(6,6)	(12,12)	(16,16)
pyrazine	110	0.47 (92%)	1.73 (46%)	3.24 (0.2%)
dimelamine	300	2.78 (98.9%)	9.34 (84%)	17.77 (2.6%)
dimelem	510	7.06 (98.2%)	20.77 (91%)	42.22 (4.0%)
C_{60}	900	63.18 (99.2%)	161.12 (98.6%)	277.35 (34%)
$Si_{72}H_{64}$	1688	455.66 (99.4%)	1196.02 (99.6%)	2043.81 (87%)

[a]The percentage of the total CASCI time-to-solution attributable to the ERI transformation is reported in parentheses. All calculations used the 6-31G** basis and were performed on a single NVidia K40 GPU.

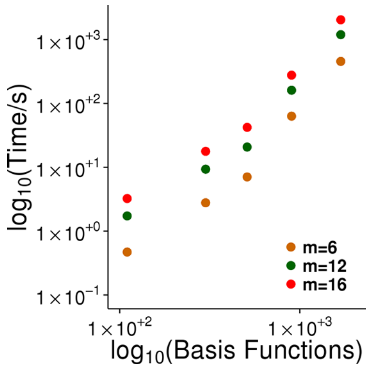


Figure 4. Times required to perform the ERI transformations as a function of the number of single-electron basis functions (N) for different numbers of active orbitals (m). All calculations used the 6-31G** basis and were performed on a single NVidia K40 GPU.

silicon nanoparticle. We thus investigate the scaling of the integral transformation by performing benchmark calculations on two series of molecules with basis sets of consistent sparsity: basis-function-dense, three-dimensional silicon clusters, and models of planar graphitic carbon nitride for which we expect the ERIs to be sparser. The silicon systems computed are $Si_{15}H_{25}$, $Si_{25}H_{30}$, $Si_{44}H_{44}$, $Si_{50}H_{50}$, and $Si_{72}H_{64}$, while the carbon nitride systems are $C_4N_2H_4$, $C_6N_{11}H_9$, $C_{12}N_{19}H_9$, $C_{18}N_{27}H_9$, and $C_{36}N_{52}H_{12}$. Both sets of clusters are treated with the 6-31G** basis. Additionally, the series of silicon clusters was computed using the much smaller LANL2DZ effective core potential basis set[82] to investigate the effect of atomic basis set size on scaling. All three series were computed with three different size active spaces (m = 6, m = 12, and m = 16).

The results of these scaling studies are presented in Figures 5 and 6. Geometries and tables of the timing data, including total times-to-solution and ERI transformation times, are presented in the Supporting Information. In all cases, we have performed linear regression analysis of the log(ERI transform time) as a function of log(N) to determine the effective scaling exponents. For all three series the effective scaling exponent is considerably lower than the theoretical value of 4.0. When the silicon cluster series is computed with the LANL2DZ basis set (Figure 5) scaling exponents in the range 2.2−2.4 are observed for the various active spaces. The same series computed with the larger 6-31G** basis set (Figure 5) yields nearly identical scaling exponents: 2.3−2.5. Thus, the size of the atomic basis set and number of active orbitals do not affect the scaling with respect to N. On the other hand, the two-dimensional carbon nitride series (Figure 6) scales more favorably, with an exponent ranging from 1.8 to 1.9 observed, indicating that the geometry of a series of molecules does affect the scaling.

3.3. Direct CI Performance. When small molecules are treated with large active spaces, the direct CI step, the cost of which scales factorially with the number of active orbitals, limits performance. We have performed benchmark calculations for active spaces ranging from 10^5−10^8 configurations. The times to complete a single σ formation are reported in Figure 7 and Table 3. Note that we report two different times in Table 3: the σ formation time and the total time per σ formation. The prior is the time spent computing the σ vector itself, while the latter is the total time needed to perform the direct CI calculation divided by the total number of σ formations performed. Thus, the total time per σ formation includes overhead which is excluded from the σ formation time. All calculations were

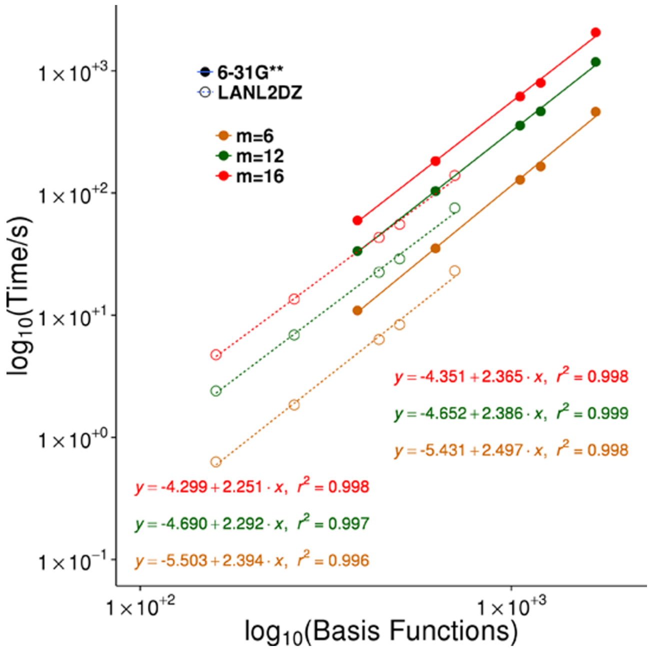


Figure 5. Times required to perform the ERI transformations for a series of silicon clusters as a function of the number of single-electron basis functions (N) for different numbers of active orbitals (m). Calculations using the LANL2DZ(6-31G**) basis are depicted using open (filled) circles and were performed on a single NVidia K40 GPU. Linear fits used to determine scaling exponents are shown as dashed (solid) lines.

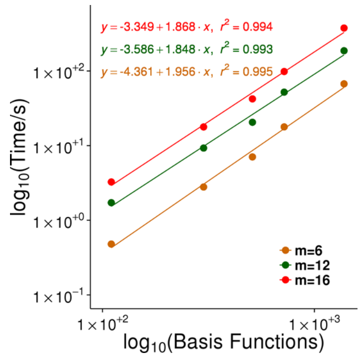


Figure 6. Times required to perform the ERI transformations for a series of models of graphitic carbon nitride as a function of the number of single-electron basis functions (N) for different numbers of active orbitals (m). All calculations used the 6-31G** basis and were performed on a single NVidia K40 GPU. Linear fits used to determine scaling exponents are shown as solid lines.

performed on ethylene (C_2H_4), but note that CI performance is independent of the total number of basis functions and electrons. It depends only on the number of active electrons and active orbitals, and therefore equivalent CI performance is observed for systems of any size.

The largest active space included in this study, (16,16), requires less than 1 min per σ formation. Note that the computational cost scales approximately linearly with the size of the configuration space; fitting $\log_{10}$(time) to $\log_{10}$(number of configurations) yields a scaling exponent of 1.1. The present implementation uses only a single GPU device, and the maximum configuration space is limited by the available device memory (12 GB on the NVIDIA K40) rather than the time

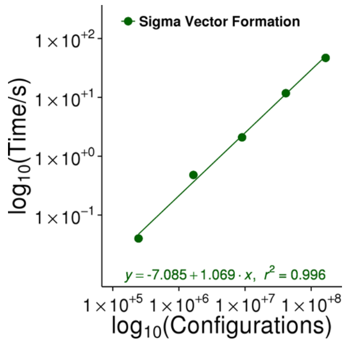


Figure 7. Time per σ formation as a function of the number of determinants for active spaces ranging from (16,12) to (16,16). All calculations were performed on a single NVidia K40 GPU.

Table 3. Time per σ Formation and Average Iteration Time for Active Spaces Ranging from (16,12) to (16,16)[a]

active space	no. of configurations	σ formation time (s)	total time per σ formation (s)
(16,12)	245025	0.04	0.08
(16,13)	1656369	0.48	0.74
(16,14)	9018009	2.09	3.55
(16,15)	41409225	11.77	18.25
(16,16)	165636900	46.89	70.82

[a]All calculations were performed on a single NVidia K40 GPU.

required for the calculation. We expect to be able to expand to (18,18) and possibly (20,20) active spaces by extending our implementation to take advantage of the direct inter-GPU communication and distributed GPU memory addressing features available on the most recent devices.

Note that the total time per σ vector formation for the (16,16) active space is 70.82 s, and only 46.89 s is attributable to the formation of σ itself. The remaining time (approximately 34% of the average iteration time) corresponds to overhead associated with other low-scaling steps in the Davidson algorithm, memory allocation and access, CPU-GPU communication, and other housekeeping tasks such as stack frame setup and context changes between the CPU and GPU. This again underlines the fact that for all but the simplest algorithms achieving optimal GPU performance requires optimization of several steps, and these steps often include operations that would not be performance-limiting in a serial implementation. The performance is often limited not by the step with the highest operation count but by the step with the highest operation count that is not (or cannot be) effectively implemented on the GPU.

In Table 4 we present timings for each individual step of a single σ vector formation for a (16,16) active space. The central matrix−matrix multiplication operation consumes approximately 45% of the overall computational effort for a single σ vector formation. The calculations of the off-diagonal terms in **D**, σ, and the one-electron part requires most of the remainder of the time, while the cost of the diagonal contributions is almost negligible. Within the off-diagonal elements, note that the β contributions, which require costly noncoalesced memory accesses, require more effort than the α contributions by a factor of 3.2. This underscores the necessity of careful management of the memory access pattern.

3.4. Accuracy. As noted above, the CASCI procedure is performed in double precision in this work. Additionally, no approximations are made to the Hamiltonian matrix to improve scaling. Here we demonstrate the accuracy of our implementation by comparison to a code of known quality. Single point energies for a survey of several systems and active spaces calculated on the CPU using Molpro and on the GPU using TeraChem are shown in Table 5. Calculated energies are in agreement to within a maximum error of 1.1×10^{-6} E_h. Note that for the largest system (dimelem) the GPU and CPU results agree to 9 digits, more than would be available if the calculations were performed entirely in single precision.

Table 4. Times To Perform Different Steps in σ Vector Formation for a (16,16) Active Space

step	time (s)
one-electron contributions to σ	6.43
α diagonal **D**	0.12
β diagonal **D**	0.19
α off-diagonal **D**	1.50
β off-diagonal **D**	5.30
E formation (matrix multiply)	21.06
α diagonal σ	0.08
β diagonal σ	0.13
α off-diagonal σ	1.74
β off-diagonal σ	5.21
total time (GPU time)	41.76
total time (CPU + GPU time)	46.89

4. CONCLUSIONS

We have presented an implementation of FCI/CASCI on GPU hardware. Even when a 1.7 × 1.4 × 1.4 nm silicon nanoparticle is calculated with a large (16,16) active space and a polarized, all-electron basis set, the cost of a CASCI calculation is only 39 min on a single GPU processor. This capability opens up many interesting scientific possibilities—e.g. the study of the photochemistry of large biological chromophores in complex environments or photocatalytic reactions on the surfaces of semiconducting nanoparticles, treating the entire system via a multireference level of theory capable of describing bond breaking, multiply excited states, and conical intersections. This algorithm performs well in the limits of both large active space and large single-electron basis, but memory constrains our present implementation to configuration spaces on the order of 10^8 determinants, which corresponds to a (16,16) active space.

It is reasonable to ask whether a (16,16) or smaller active space is capable of describing chemistry at the nanoscale. Of course the answer to this question depends on the problem of interest, but there is reason for optimism in many cases. For many problems where a multirefrence description would be beneficial, including those described in the previous paragraph, the region where a multireference description is necessary is no larger than a typical molecule. Thus, an active space comparable to those employed in the study of molecular chemistry would likely be sufficient. Even when an electronic excitation is delocalized over more than a nanometer, it is not necessarily true that more active orbitals are required for an adequate description of the electronic structure. For example, when the low-lying electronic excitations of semiconductor nanoparticles and conjugated polymers are computed via a single reference level of theory such as time-dependent density functional theory, the lowest excited state can often be described in terms of a single natural transition orbital pair, indicating that a very small active space (possibly even (2,2)) would be suffi-

Table 5. Absolute Ground State Energies of Several Systems Computed at the HF-CASCI/6-31G** Level of Theory with Various Active Spaces

system	active space	TeraChem energy (E_h)	Molpro energy (E_h)	difference (E_h)
ethylene	(16,10)	−78.063304843314	−78.063304824538	1.9×10^{-8}
	(16,11)	−78.066690699464	−78.066690639571	6.0×10^{-8}
	(16,12)	−78.074505748149	−78.074505683619	6.5×10^{-8}
	(16,13)	−78.083428418532	−78.083428366856	5.2×10^{-8}
	(16,14)	−78.091109138010	−78.091109107138	3.1×10^{-8}
	(16,15)	−78.100302547285	−78.100302524072	2.3×10^{-8}
pyrazine	(6,6)	−262.726271748762	−262.726271553812	1.9×10^{-7}
pyrazine	(12,12)	−262.731145249735	−262.731145075701	1.7×10^{-7}
dimelamine	(6,6)	−831.559377872566	−831.559377131972	7.4×10^{-7}
dimelamine	(12,12)	−831.571307352380	−831.571306630668	7.2×10^{-7}
dimelem	(6,6)	−1494.410755484148	−1494.410754288612	1.1×10^{-6}
dimelem	(12,12)	−1494.425909386718	−1494.425908295103	1.1×10^{-6}

cient.[83−85] Processes which require a larger active space than this could be approached by pairing methods which restrict the configuration space[52,86−90] with carefully defined orbital determination schemes to reduce the factorial scaling of the problem.

The present FCI implementation can trivially be merged with the recently reported CASSCF orbital optimization algorithm of Hohenstein et al.,[32] which makes use of the sparsity of the AO basis in order to decrease the scaling of the full CASSCF procedure. Because the orbital gradients are computed directly from integrals in the AO basis, CASSCF does not require transformed integrals beyond those required by CASCI to be computed and stored. This combined CASSCF algorithm will allow calculations of nanoscale systems with large (16,16) active spaces to be performed in minutes or hours on a single GPU. Further acceleration would be possible by employing lower-scaling approximations to the electron repulsion integrals such as density fitting.

Our GPU FCI code also demonstrates two key points regarding the development of scientific code on GPUs. First, it is not sufficient to simply accelerate the highest scaling step or steps in a given algorithm and implement the remaining code in serial. GPU acceleration can reduce the cost of a high-scaling portion of an algorithm by 2 orders of magnitude, leaving lower-scaling steps as performance limiting. In our case it was necessary to accelerate all five steps in Algorithm 1 (see Chart 1) in order to achieve high performance. Second, many brilliant computational algorithms were developed for the vector processors of the 1980s, and it is fruitful to revisit them when designing quantum chemical algorithms for the GPU.

■ ASSOCIATED CONTENT

Ⓢ Supporting Information

The Supporting Information is available free of charge on the ACS Publications website at DOI: 10.1021/acs.jctc.5b00634.

Tables of benchmark timings for series of silicon and graphitic carbon nitride clusters and geometries for all structures in Cartesian coordinates (PDF)

■ AUTHOR INFORMATION

Corresponding Author

*Phone: 517-355-9715 x169. E-mail: levine@chemistry.msu.edu.

Notes

The authors declare no competing financial interest.

■ ACKNOWLEDGMENTS

The authors thank Warren Beck, Ed Hohenstein, Todd Martínez, and Piotr Piecuch for helpful discussions. We are grateful to Paul Reed for technical assistance. Calculations in this work were performed, in part, using computer time obtained via NSF XSEDE allocation CHE-140101. We gratefully acknowledge startup funding from Michigan State University and a generous donation of hardware from NVidia.

■ REFERENCES

(1) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222−231.
(2) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 1004−1015.
(3) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2009**, *5*, 2619−2628.
(4) Asadchev, A.; Allada, V.; Felder, J.; Bode, B. M.; Gordon, M. S.; Windus, T. L. *J. Chem. Theory Comput.* **2010**, *6*, 696−704.
(5) Luehr, N.; Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2011**, *7*, 949−954.
(6) Wilkinson, K. A.; Sherwood, P.; Guest, M. F.; Naidoo, K. J. *J. Comput. Chem.* **2011**, *32*, 2313−2318.
(7) Maia, J. D. C.; Carvalho, G. A. U.; Mangueira, C. P.; Santana, S. R.; Cabral, L. A. F.; Rocha, G. B. *J. Chem. Theory Comput.* **2012**, *8*, 3072−3081.
(8) Wu, X.; Koslowski, A.; Thiel, W. *J. Chem. Theory Comput.* **2012**, *8*, 2272−2281.
(9) Andrade, X.; Aspuru-Guzik, A. *J. Chem. Theory Comput.* **2013**, *9*, 4360−4373.
(10) Asadchev, A.; Gordon, M. S. *J. Chem. Theory Comput.* **2013**, *9*, 3385−3392.
(11) Miao, Y. P.; Merz, K. M. *J. Chem. Theory Comput.* **2013**, *9*, 965−976.
(12) Nitsche, M. A.; Ferreria, M.; Mocskos, E. E.; Lebrero, M. C. G. *J. Chem. Theory Comput.* **2014**, *10*, 959−967.
(13) Yasuda, K.; Maruoka, H. *Int. J. Quantum Chem.* **2014**, *114*, 543−552.
(14) Miao, Y. P.; Merz, K. M. *J. Chem. Theory Comput.* **2015**, *11*, 1449−1462.
(15) Vogt, L.; Olivares-Amaya, R.; Kermes, S.; Shao, Y.; Amador-Bedolla, C.; Aspuru-Guzik, A. *J. Phys. Chem. A* **2008**, *112*, 2049−2057.
(16) Olivares-Amaya, R.; Watson, M. A.; Edgar, R. G.; Vogt, L.; Shao, Y.; Aspuru-Guzik, A. *J. Chem. Theory Comput.* **2010**, *6*, 135−144.
(17) DePrince, A. E.; Hammond, J. R. *J. Chem. Theory Comput.* **2011**, *7*, 1287−1295.
(18) Isborn, C. M.; Luehr, N.; Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2011**, *7*, 1814−1823.
(19) Ma, W. J.; Krishnamoorthy, S.; Villa, O.; Kowalski, K. *J. Chem. Theory Comput.* **2011**, *7*, 1316−1327.

(20) Asadchev, A.; Gordon, M. S. *J. Chem. Theory Comput.* **2012**, *8*, 4166−4176.
(21) Ma, W. J.; Krishnamoorthy, S.; Villa, O.; Kowalski, K.; Agrawal, G. *Cluster Comput.* **2013**, *16*, 131−155.
(22) DePrince, A. E.; Kennedy, M. R.; Sumpter, B. G.; Sherrill, C. D. *Mol. Phys.* **2014**, *112*, 844−852.
(23) Leang, S. S.; Rendell, A. P.; Gordon, M. S. *J. Chem. Theory Comput.* **2014**, *10*, 908−912.
(24) Maurer, S. A.; Kussmann, J.; Ochsenfeld, C. *J. Chem. Phys.* **2014**, *141*, 051106.
(25) Kussmann, J.; Ochsenfeld, C. *J. Chem. Theory Comput.* **2015**, *11*, 918−922.
(26) Yoshikawa, T.; Nakai, H. *J. Comput. Chem.* **2015**, *36*, 164−170.
(27) Meek, G. A.; Baczewski, A. D.; Little, D. J.; Levine, B. G. *J. Phys. Chem. C* **2014**, *118*, 4023−4032.
(28) Wang, L. P.; Titov, A.; McGibbon, R.; Liu, F.; Pande, V. S.; Martinez, T. J. *Nat. Chem.* **2014**, *6*, 1044−1048.
(29) Wang, L.; Fried, S. D.; Boxer, S. G.; Markland, T. E. *Proc. Natl. Acad. Sci. U. S. A.* **2014**, *111*, 18454−18459.
(30) Yasaei, P.; Kumar, B.; Hantehzadeh, R.; Kayyalha, M.; Baskin, A.; Repnin, N.; Wang, C. H.; Klie, R. F.; Chen, Y. P.; Kral, P.; Salehi-Khojin, A. *Nat. Commun.* **2014**, *5*, 4911.
(31) Shu, Y. N.; Levine, B. G. *J. Chem. Phys.* **2015**, *142*, 104104.
(32) Hohenstein, E. G.; Luehr, N.; Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Phys.* **2015**, *142*, 224103.
(33) Bhaskaran-Nair, K.; Ma, W. J.; Krishnamoorthy, S.; Villa, O.; van Dam, H. J. J.; Apra, E.; Kowalski, K. *J. Chem. Theory Comput.* **2013**, *9*, 1949−1957.
(34) Schmidt, M. W.; Gordon, M. S. *Annu. Rev. Phys. Chem.* **1998**, *49*, 233−266.
(35) Szalay, P. G.; Muller, T.; Gidofalvi, G.; Lischka, H.; Shepard, R. *Chem. Rev.* **2012**, *112*, 108−181.
(36) Roos, B. O.; Taylor, P. R. *Chem. Phys.* **1980**, *48*, 157−173.
(37) Bofill, J. M.; Pulay, P. *J. Chem. Phys.* **1989**, *90*, 3637−3646.
(38) Abrams, M. L.; Sherrill, C. D. *Chem. Phys. Lett.* **2004**, *395*, 227−232.
(39) Potts, D. M.; Taylor, C. M.; Chaudhuri, R. K.; Freed, K. F. *J. Chem. Phys.* **2001**, *114*, 2592−2600.
(40) Chattopadhyay, S.; Chaudhuri, R. K.; Freed, K. F. *J. Phys. Chem. A* **2011**, *115*, 3665−3678.
(41) Granucci, G.; Persico, M.; Toniolo, A. *J. Chem. Phys.* **2001**, *114*, 10608−10615.
(42) Slavicek, P.; Martinez, T. J. *J. Chem. Phys.* **2010**, *132*, 234102.
(43) Lu, Z.; Matsika, S. *J. Chem. Theory Comput.* **2012**, *8*, 509−517.
(44) Lu, Z.; Matsika, S. *J. Phys. Chem. A* **2013**, *117*, 7421−7430.
(45) Shu, Y.; Levine, B. G. *J. Chem. Phys.* **2013**, *139*, 074102.
(46) Shu, Y.; Hohenstein, E. G.; Levine, B. G. *J. Chem. Phys.* **2015**, *142*, 024102.
(47) Keller, S.; Boguslawski, K.; Janowski, T.; Reiher, M.; Pulay, P. *J. Chem. Phys.* **2015**, *142*, 244104.
(48) Sherrill, C. D.; Schaefer, H. F., III In *The Configuration Interaction Method: Advances in Highly Correlated Approaches*; Per-Olov Lowdin, M. C. Z., Sabin, J. R., Brandas, E., Eds.; Advances in Quantum Chemistry; Academic Press: 1999; Vol. *34*, pp 143−269.
(49) Paldus, J. In *Theoretical Chemistry: Advances and Perspectives*; Eyring, H., Henderson, D., Eds.; Academic: New York, NY, 1976.
(50) Shavitt, I. *Int. J. Quantum Chem.* **1978**, *14*, 5−32.
(51) Handy, N. C. *Chem. Phys. Lett.* **1980**, *74*, 280−283.
(52) Olsen, J.; Roos, B. O.; Jorgensen, P.; Jensen, H. J. A. *J. Chem. Phys.* **1988**, *89*, 2185−2192.
(53) Zarrabian, S.; Sarma, C. R.; Paldus, J. *Chem. Phys. Lett.* **1989**, *155*, 183−188.
(54) Bendazzoli, G. L.; Evangelisti, S. *J. Chem. Phys.* **1993**, *98*, 3141−3150.
(55) Knowles, P. J.; Handy, N. C. *Chem. Phys. Lett.* **1984**, *111*, 315−321.
(56) Knowles, P. J. *Chem. Phys. Lett.* **1989**, *155*, 513−517.
(57) Mitrushenkov, A. O. *Chem. Phys. Lett.* **1994**, *217*, 559−565.
(58) Rolik, Z.; Szabados, A.; Surjan, P. R. *J. Chem. Phys.* **2008**, *128*, 144101.
(59) White, S. R.; Martin, R. L. *J. Chem. Phys.* **1999**, *110*, 4127−4130.
(60) Mitrushenkov, A. O.; Fano, G.; Ortolani, F.; Linguerri, R.; Palmieri, P. *J. Chem. Phys.* **2001**, *115*, 6815−6821.
(61) Chan, G. K. L.; Head-Gordon, M. *J. Chem. Phys.* **2002**, *116*, 4462−4476.
(62) Legeza, O.; Roder, J.; Hess, B. A. *Phys. Rev. B: Condens. Matter Mater. Phys.* **2003**, *67*, 125114.
(63) Moritz, G.; Reiher, M. *J. Chem. Phys.* **2007**, *126*, 244109.
(64) Martinez, T. J.; Mehta, A.; Carter, E. A. *J. Chem. Phys.* **1992**, *97*, 1876−1880.
(65) Ten-no, S.; Iwata, S. *J. Chem. Phys.* **1996**, *105*, 3604−3611.
(66) Aquilante, F.; Pedersen, T. B.; Lindh, R.; Roos, B. O.; De Meras, A. S.; Koch, H. *J. Chem. Phys.* **2008**, *129*, 024113.
(67) Delcey, M. G.; Freitag, L.; Pedersen, T. B.; Aquilante, F.; Lindh, R.; Gonzalez, L. *J. Chem. Phys.* **2014**, *140*, 174103.
(68) Schutz, M.; Lindh, R.; Werner, H. J. *Mol. Phys.* **1999**, *96*, 719−733.
(69) Head-Gordon, M.; Pople, J. A.; Frisch, M. J. *Chem. Phys. Lett.* **1988**, *153*, 503−506.
(70) Frisch, M.; Ragazos, I. N.; Robb, M. A.; Schlegel, H. B. *Chem. Phys. Lett.* **1992**, *189*, 524−528.
(71) Duch, W. In *GRMS or Graphical Representation of Model Spaces*; Berthier, G., Dewar, M. J. S., Fischer, H., Fukui, K., Hall, G. G., Hinze, J., Jaffe, H. H., Jortner, J., Kutzelnigg, W., Ruedenberg, K., Tomasi, J., Eds.; Springer Verlag: New York, 1986; Vol. *1*.
(72) Davidson, E. R. *J. Comput. Phys.* **1975**, *17*, 87−94.
(73) Siegbahn, P. E. M. *Chem. Phys. Lett.* **1984**, *109*, 417−423.
(74) *CUBLAS Library User Guide*; NVidia: 2012.
(75) Lyakh, D. I. *Comput. Phys. Commun.* **2015**, *189*, 84−91.
(76) *CUDA C Programming Guide*; NVidia: 2015.
(77) Levine, B. G.; Stone, J. E.; Kohlmeyer, A. *J. Comput. Phys.* **2011**, *230*, 3556−3569.
(78) Werner, H.-J.; Knowles, P. J.; Knizia, G.; Manby, F. R.; Schütz, M.; Celani, P.; Korona, T.; Lindh, R.; Mitrushenkov, A.; Rauhut, G.; Shamasundar, K. R.; Adler, T. B.; Amos, R. D.; Bernhardsson, A.; Berning, A.; Cooper, D. L.; Deegan, M. J. O.; Dobbyn, A. J.; Eckert, F.; Goll, E.; Hampel, C.; Hesselmann, A.; Hetzer, G.; Hrenar, T.; Jansen, G.; Köppl, C.; Liu, Y.; Lloyd, A. W.; Mata, R. A.; May, A. J.; McNicholas, S. J.; Meyer, W.; Mura, M. E.; Nicklass, A.; O'Neill, D. P.; Palmieri, P.; Peng, D.; Pflüger, K.; Pitzer, R.; Reiher, M.; Shiozaki, T.; Stoll, H.; Stone, A. J.; Tarroni, R.; Thorsteinsson, T.; Wang, M. MOLPRO, version 2010.1, a package of ab initio programs, 2010. See: http://www.molpro.net (accessed Sept 1, 2015).
(79) Werner, H. J.; Knowles, P. J. *J. Chem. Phys.* **1985**, *82*, 5053−5063.
(80) Knowles, P. J.; Werner, H. J. *Chem. Phys. Lett.* **1985**, *115*, 259−267.
(81) Knowles, P. J.; Handy, N. C. *Comput. Phys. Commun.* **1989**, *54*, 75−83.
(82) Wadt, W. R.; Hay, P. J. *J. Chem. Phys.* **1985**, *82*, 284−298.
(83) Martin, R. L. *J. Chem. Phys.* **2003**, *118*, 4775−4777.
(84) Badaeva, E.; May, J. W.; Ma, J.; Gamelin, D. R.; Li, X. S. *J. Phys. Chem. C* **2011**, *115*, 20986−20991.
(85) Nayyar, I. H.; Batista, E. R.; Tretiak, S.; Saxena, A.; Smith, D. L.; Martin, R. L. *J. Phys. Chem. Lett.* **2011**, *2*, 566−571.
(86) Nakano, H.; Hirao, K. *Chem. Phys. Lett.* **2000**, *317*, 90−96.
(87) Ivanic, J. *J. Chem. Phys.* **2003**, *119*, 9364−9376.
(88) Ma, D. X.; Manni, G. L.; Gagliardi, L. *J. Chem. Phys.* **2011**, *135*, 044128.
(89) Li Manni, G.; Ma, D. X.; Aquiliante, F.; Olsen, J.; Gagliardi, L. *J. Chem. Theory Comput.* **2013**, *9*, 3375−3384.
(90) Parker, S. M.; Seideman, T.; Ratner, M. A.; Shiozaki, T. *J. Chem. Phys.* **2013**, *139*, 021108.