Many-body quantum chemistry on graphics processing units

A. Eugene DePrince III, Jeff R. Hammond, and Stephen K. Gray, Argonne National Laboratory
{adeprince,jhammond,gray}@anl.gov

Abstract

Heterogeneous nodes composed of a multicore CPU and at least one graphics processing unit (GPU) are increasingly common in high-performance scientific computing, and significant programming effort is currently being undertaken to port existing scientific algorithms to these unique architectures. We present implementations for two many-body quantum chemistry methods on heterogeneous nodes: the coupled-cluster with single and double excitations (CCSD) and time-dependent configuration interaction with single and double excitations (TD-CISD) methods. Both methods can be implemented on a computer as a series of dense matrix-matrix multiplications, operations that GPUs are particularly adept at performing. The GPU-accelerated CCSD algorithm is as much as 4.3 times faster than the corresponding CPU algorithm and 9.7 times faster than the algorithm in the Molpro package. The TD-CISD algorithm is accelerated by as much as a factor of 3.9 by GPUs. Enhanced performance is achieved by overlapping CPU and GPU computations.

I. Introduction

The current era in supercomputing is one of ever increasing node-level parallelism that can come in one of two forms: multicore CPUs or many-core GPUs. Both are associated with unique challenges that are exagerated when programming in a heterogeneous environment. When programming multicore CPUs, a balance must be struck between the convenience of the process-based parallelism of MPI or Global Arrays, treating each core as an independent processsor, and the benefits of shared memory when threading. For GPUs, existing scientific code must often be entirely rewritten in Cuda or OpenCL, with an additional layer of difficulty in repeatedly transferring data between host and device. To add to the complexity, supercomputers that exploit graphics processors tend to boast multicore CPUs as well. Fully utlizing such heterogenous nodes requires careful attention to load balancing between processors of vastly different capability. Regardless of these complexities, a number of quantum chemistry methods have been ported recently to GPUs, including Gaussian integral generation [1], [2], [3]; Hartree-Fock [4], [5] and density-functional [6], [7], [8], [9] theories; low-order perturbation theory [10], [11], [12]; quantum Monte Carlo [13], [14], [15]; the coupled-cluster doubles (CCD) method [16]; and the perturbative triples correction beyond the coupled-cluster singles doubles (CCSD) method, the CCSD(T) method [17].

We previously showed that the CCD equations can be solved entirely on an NVIDIA C2050 Tesla (Fermi) GPU at a cost that is 4–5 times less than the corresponding CPU algorithm (when fully utilizing 2 Intel Xeon X5550 processors) [16]. This preliminary algorithm sought to minimize communication by storing all integrals and amplitudes on the device and was thus severely limited in its applicability to large molecules in large basis sets. We have since developed an algorithm that stresses scalability by minimizing the storage requirements on the GPU. An unfortunate consequence of the minimal storage algorithm is the requirement that significant amounts of data be repeatedly copied to and from the device. However, when fully utilizing both CPU and GPU by overlapping computations on each processor, the resulting overhead can effectively masked.

In this paper, we summarize recent node-level optimizations of a GPU-accelerated spin-free CCSD algorithm as well as an implementation of explicitly time-dependent configuration interaction with single and double excitations (TD-CISD). The hybrid CCSD algorithm is as much as 4.3 times more efficient than the corresponding pure CPU algorithm and as much as 9.7 times faster per iteration than the algorithm found in the Molpro package [18]. The TD-CISD implementation represents the first implementation of explicitly time-dependent CISD to date in the literature on a heterogeneous system. For large systems, the hybrid TD-CISD algorithm is as much as 3.2 times more efficient than the pure CPU implementation. We present an application illustrating the metallic behavior of very long hydrogen chains.

II. Theory

A. Coupled-cluster singles doubles

Coupled-cluster methods are well described in the literature, so we limit our discussion to the working equations of our implementation; similar equations can be found elsewhere [19], [20]. The ground-state energy for the CCSD wavefunction is determined by the projective solution of the Schrödinger equation, resulting in the following system of equations for the doubles amplitudes, t_{ij}^{ab},

$$D_{ij}^{ab}t_{ij}^{ab} = v_{ij}^{ab} + P(ia,jb)\left[t_{ij}^{ae}I_e^b - t_{im}^{ab}I_j^m + \frac{1}{2}v_{ef}^{ab}c_{ij}^{ef} + \frac{1}{2}c_{mn}^{ab}I_{ij}^{mn} - t_{mj}^{ae}I_{ie}^{mb} - I_{ie}^{ma}t_{mj}^{eb} + (2t_{mi}^{ea} - t_{im}^{ea})I_{ej}^{mb} + t_i^e I_{ej}^{\prime ab} - t_m^a I_{ij}^{\prime mb}\right], \quad (1)$$

and similarly for the singles amplitudes, t_i^a,

$$D_i^a t_i^a = I_e^a t_i^e - I_i'^m t_m^a + I_e^m (2t_{mi}^{ea} - t_{im}^{ea}) + (2v_{ei}^{ma} - v_{ei}^{am}) t_m^e - v_{ei}^{mn} (2t_{mn}^{ea} - t_{mn}^{ae}) + v_{ef}^{ma} (2t_{mi}^{ef} - t_{im}^{ef}). \tag{2}$$

Throughout, occupied indices are given by i, j, k, l, m, and n and virtual indices by a, b, c, d, e, and f. The intermediate tensors found in Eqs. (1) and (2) are slightly modified versions of those presented in Ref. [20] and are given in Table I. The energy

TABLE I

INTERMEDIATE TENSORS FOR THE SPIN-FREE CCSD EQUATIONS. THE SYMBOL v_{ij}^{ab} DENOTES A TWO-ELECTRON INTEGRAL, $v_{ij}^{ab} = \langle ab|ij\rangle$, AND c_{ij}^{ab} REPRESENTS A SUM OF SINGLES AND DOUBLES AMPLITUDES: $c_{ij}^{ab} = t_{ij}^{ab} + t_i^a t_j^b$. THE PERMUTATION OPERATOR $P(ia, jb)$ IMPLIES A SUM OF TWO TERMS: $P(ia, jb) v_{ij}^{ab} = v_{ij}^{ab} + v_{ji}^{ba}$.

I_b^a	$(2v_{be}^{am} - v_{be}^{ma}) t_m^e - (2v_{eb}^{mn} - v_{be}^{mn}) c_{mn}^{ea}$
I_a^i	$(2v_{ae}^{im} - v_{ea}^{im}) t_m^e$
$I_j'^i$	$(2v_{je}^{im} - v_{ej}^{im}) t_m^e + (2v_{ef}^{mi} - v_{ef}^{im}) t_{mj}^{ef}$
I_j^i	$I_j'^i + I_e^i t_j^e$
I_{kl}^{ij}	$v_{kl}^{ij} + v_{ef}^{ij} c_{kl}^{ef} + P(ik/jl) t_k^e v_{el}^{ij}$
$I_{ci}'^{ab}$	$v_{ci}^{ab} - v_{ci}^{am} t_m^b - v_{ci}^{mb} t_m^a$
$I_{jk}'^{ia}$	$v_{jk}^{ia} + v_{ef}^{ia} c_{jk}^{ef}$
I_{jb}^{ia}	$v_{jb}^{ia} - \frac{1}{2} v_{eb}^{im} (t_{jm}^{ea} + 2t_j^e t_m^a) + v_{eb}^{ia} t_j^e - v_{jb}^{im} t_m^a$
I_{bj}^{ia}	$v_{bj}^{ia} - \frac{1}{2} v_{be}^{im} (t_{mj}^{ae} + 2t_m^a t_j^e) + v_{be}^{ia} t_j^e - v_{bj}^{im} t_m^a + \frac{1}{2} (2v_{be}^{im} - v_{eb}^{im}) t_{mj}^{ea}$

denominators denoted by D_{ij}^{ab} and D_i^a are defined by the diagonal elements of the Fock matrix, $D_{ij}^{ab} = f_i^i + f_j^j - f_a^a - f_b^b$, and we have assumed that the molecular orbitals are canonical Hartree-Fock orbitals. Throughout this paper, the Einstein summation convention is assumed whereby repeated upper and lower indices are summed, but we note that the terms containing the energy denominators involve no sum. The CC equations are usually solved by simple substitution, beginning with a trial wave function and iteratively updating the amplitudes according to Eqs. (1) and (2). The equations are considered converged when the norm of the change in the amplitudes between iterations falls below 10^{-7}. The DIIS extrapolation technique is used to accelerate the convergence of the algorithm [21].

B. Time-dependent singles doubles configuration interaction

Consider the discrete form of the time-dependent Schrödinger equation (TDSE)

$$i \frac{d}{dt} \mathbf{c}(t) = \mathbf{H}(t) \cdot \mathbf{c}(t), \tag{3}$$

where $\mathbf{c}$ is a column vector of complex components, and $\mathbf{H}(t)$ is the time-dependent Hamiltonian matrix. The time-dependent Hamiltonian in the dipole approximation is given by

$$\mathbf{H}(t) = \mathbf{H}_0 - \hat{\mu} \cdot \mathbf{E}(t), \tag{4}$$

where $\mathbf{H}_0$ is the time-independent electronic Hamiltonian, $\hat{\mu}$ is the dipole operator, and $\mathbf{E}(t)$ is the applied laster field. $\mathbf{E}(t)$ is z-polarized (here, along the hydrogen chain) and is taken to be a 1-cycle $\sin^2$-shaped pulse,

$$E_z(t) = E_0 \, \sin^2(\omega t/2) \, \sin(\omega t), \quad 0 < t < 2\pi/\omega, \tag{5}$$

where the frequency is chosen to fulfill the condition that the pulse be 1 fs in duration. The field strength, E_0 is taken to be 0.05 a.u. This short pulse is not tuned to any specific excitation energy; its parameters are similar to those given in Ref. [22] and are intended to excite electrons into a broad superposition of states.

The TDSE can be evolved in one of two ways. Often, the wave function is expanded in the basis of eigenfunctions of the time-independent Hamiltonian and propagated in by split-operator methods [23]. This reprentation places most of the computational effort in the diagonalization of the Hamiltonian, requiring that many, if not all, of the electronic states be determined and stored. Alternatively, the usual configuration interaction expansion of the wave function can be propagated directly. In this basis, Eq. (3) is numerically integrated by high-order Runge-Kutta or much more stable and accurate symplectic integrators inspired by classical molecular dynamics but tailored specifically to quantum mechanical problems [24], [25], [26]. This representation places all of the computational effort into the repeated construction of the so-called sigma vector (the action of the Hamiltonian on the CI vector), the explicit form of which is given in Table II for the spin-free CISD method. In our algorithm, Eq. (3) is evolved according to the 5^{th}-order symplectic integrator of Ref. [26], which requires a total of six complex sigma vector evaluations per time step.

TABLE II
THE SPIN-FREE FORM OF THE CISD SIGMA VECTOR. THE CI EXPANSION COEFFICIENTS ARE GIVEN BY c_{ij}^{ab} AND c_i^a FOR DOUBLY- AND SINGLY-EXCITED DETERMINANTS, RESPECTIVELY.

$(\mathbf{H}\cdot\mathbf{c})_{ij}^{ab} = v_{ij}^{ab}$
$+P(ia,jb)[\frac{1}{2}v_{ef}^{ab}c_{ij}^{ef} + \frac{1}{2}c_{mn}^{ab}v_{ij}^{mn} - c_{mj}^{ae}v_{ie}^{mb} - v_{ie}^{ma}c_{mj}^{eb}$
$+(2c_{mi}^{ea} - c_{im}^{ea})v_{ej}^{mb} + c_i^e v_{ej}^{ab} - c_m^a v_{ij}^{mb} + c_{ij}^{ae}f_e^b - c_{im}^{ab}f_j^m]$
$(\mathbf{H}\cdot\mathbf{c})_i^a = f_i^a + f_e^a c_i^e - f_i^m c_m^a + f_a^i(2c_{mi}^{ea} - c_{im}^{ea})$
$-v_{ei}^{mn}(2c_{mn}^{ea} - c_{mn}^{ae}) + v_{ef}^{ma}(2c_{mi}^{ef} - c_{im}^{ef}) + (2v_{ei}^{ma} - v_{ei}^{am})c_m^e$

III. ALGORITHM DETAILS

As is shown in Eqs. (1) and (2) and Tables I and II, both the CCSD and TD-CISD methods can be expressed as a series of dense matrix-matrix multiplications. These spin-free CCSD equations consist of 34 tensor contractrions ranging in cost from the 3^{rd} to the 6^{th} power of system size. The CISD equations, which are considerably simpler, require 15 tensor contractions, with similar scaling. We note that, due to the interaction with the laser field, several terms arise in TD-CISD that were assumed to be zero in CCSD because of the use of canonical Hartree-Fock molecular orbitals; these terms scale at worst as the 5^{th} power of system size. On a multicore CPU, these equations would best be implemented using tuned BLAS libraries (e.g., threaded MKL or GotoBLAS); on a modern GPU, we have found that a similar strategy works well. The CCD equations can be solved entirely on an NVIDIA C2050 GPU using the Cublas library at a cost that is 4-5 times less than the implementation on two quad-core Intel Xeon X5550 processors [16]. The implementation in Ref. [16] sought to minimize host-device communication by storing all integrals and amplitudes in device memory. For systems where the larger data structures do not fit entirely in GPU global memory, the associated tensor contraction were tiled and the required data copied to the GPU as needed without significantly reducing performance. This result suggests that a low-storage scheme wherein integrals and amplitudes are repeatedly copied to the device could be viable; such a strategy becomes necessary when treating very large systems in very large basis sets.

Unfortunately, repeated transfers of integrals and amplitudes will reduce performance, but this overhead can be effectively masked by overlapping CPU and GPU computations. A simplistic partitioning work requires that all terms scaling as the 5^{th} power or less with system size be evaluated on the CPU, while the 6^{th}-power terms be evaluated on the GPU. In practice, however, significantly better performance can be achieved by recognizing situations when "easy" work is better suited for the GPU. All required input arrays may already be on the device, the contraction may involve a very large intermediate (too large to fit entirely in GPU memory), or the enhanced memory bandwidth of the GPU can be exploited for tensor permutations. Effectively utilizing a heterogeneous node amounts to load balancing between processors of markedly different capabilities.

In CCSD (and CISD), many of contractions involve the CC amplitudes and a block of two-electron integrals. To perform this contraction on the device, we obviously need buffers to accommodate both of these structures as well as the result of the contraction. For the doubles equations, we require buffers to accommodate three general types of data: the doubles amplitudes (or a similarly sized intermediate), the residual of the CC equations (up to a tensor permutation), and a general integral buffer. Additionally, we allocate one extra buffer of the size of the doubles amplitudes (o^2v^2) in order to limit the need for host-device communication. For the singles equations, we require buffers to accommodate the singles amplitudes and the residual of the equation to be allocated on the GPU. Ideally, the general integral buffer should hold up to v^4 elements (corresponding to the v_{cd}^{ab} block of integrals), but modest GPU memory often limits the size of this buffer. As such, our algorithm supports the tiling of all data structures larger than o^2v^2, making the minimum size of the integral buffer o^2v^2. In all, our algorithm requires that a minimum of $4o^2v^2 + 2ov$ elements be stored on the device. To put this in perspective, this algorithm should be able to treat a system of 70 electrons with 250 virtual orbitals in full double precision.

The spin-free CCSD (and CISD) equations were implemented in the Psi3 electronic structure package [27]. All hybrid computations were performed with two Intel Xeon X5550 CPUs and one NVIDIA Tesla C2050 (Fermi) GPU. Computations using the pure-CPU implementation and the CC implementation in the Molpro [18] package were performed using the same CPU hardware. The pure-CPU fully utilizes all 8 CPU cores through threaded GotoBLAS DGEMM calls, and Molpro makes use of process-based parallelism through the Global Arrays toolkit, with 7 of 8 cores dedicated to computations (one was reserved as a communication helper thread). All computations were performed in C_1 symmetry.

IV. RESULTS

A. CCSD

The relative preformance of the hybrid, pure-CPU, and Molpro CCSD implementations are presented in Table III for a wide range of system sizes. We consider first those systems for which the hybrid algorithm significantly outperforms Molpro on a per-iteration basis. For the smallest fullerene (C_{20}) and $C_{18}H_{20}$, the hybrid implementation is 4.1 and 4.3 times more efficient than the corresponding CPU implementation and 9.7 and 7.1 times more efficient than Molpro. For $C_{18}H_{20}$, a single

TABLE III
COMPARISON OF CPU AND GPU IMPLEMENTATIONS OF CCSD. TIMINGS PER CC ITERATION ARE GIVEN IN SECONDS. THE SYMBOLS o AND v REPRESENT THE NUMBER OF DOUBLY-OCCUPIED AND VIRTUAL ORBITALS IN EACH SYSTEM, RESPECTIVELY. GPU SPEEDUP SIGNIFIES THE RELATIVE COST OF CPU ALGORITHMS AS COMPARED TO THE C2050 ALGORITHM.

				Iteration Time (s)			GPU Speedup	
Molecule	Basis	o	v	Hybrid	CPU	Molpro	CPU	Molpro
CH_3OH	aug-cc-pVTZ	7	175	2.5	4.5	2.8	1.80	1.12
C_6H_6	aug-cc-pVDZ	15	171	5.1	14.7	17.4	2.88	3.41
$CH_3OSOOCH_3$	aug-cc-pVDZ	23	167	9.0	33.2	31.2	3.69	3.47
$C_{10}H_{12}$	6-31G	26	78	1.4	4.1	7.2	3.02	5.27
$C_{10}H_{12}$	cc-pVDZ	26	164	10.7	39.5	56.8	3.69	5.31
C_{20}	6-31G	40	120	10.5	43.2	102.0	4.12	9.72
$C_{18}H_{20}$	6-31G	46	138	22.5	95.9	161.4	4.27	7.07

hybrid iteration requires 22.5 seconds; Molpro requires more than 2.5 minutes for the same iteration. These timings aside, in several instances the hybrid algorithm only barely outperforms Molpro, the most dramatic of which is CH_3OH in an augmented triple-zeta basis. The hybrid implementation is only 1.1 times faster than Molpro here. Molpro is optimized for those cases when $o \gg v$, while our hybrid implementation requires only that o and v be much greater than 1. The poor performance of the hybrid algorithm is related only to the small size of the occupied space, rather than the relative sizes of the occupied and virtual spaces. For $C_{10}H_{12}$, changing from a 6-31G to a cc-pVDZ basis more than doubles the size of the virtual space but does not significantly change the relative performance of the hybrid algorithm and Molpro. For this system, the performance of the hybrid algorithm is bound by the size of the occupied space. These results suggest, not surprisingly, that massive amounts of work are necessary before the use of GPUs becomes advantageous in scientific computing.

B. Time-dependent CISD

Explicitly time-dependent simulations can capture a variety of interesting phenomena that cannot be described by time-independent or linear response methods. We illustrate metallic behavior in long one-dimensional hydrogen chains by applying a femtosecond laser pulse and observing the collective long-range motion of electronic density. We observe very large oscillations in the instantaneous dipole moment as a function of time, well after the end of the laser interaction. Figure 1 illustrates these oscillations for H_n with $n = 10 - 100$ in a minimal STO-3G basis set. Note that the dipole oscillations dampen within just a

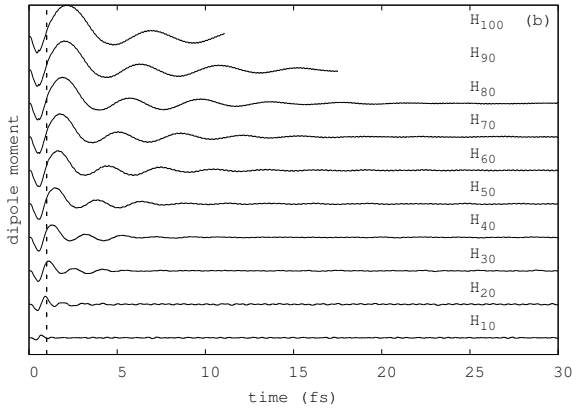

Fig. 1. Dipole moments (a.u.) for hydrogen chains of various lengths as a function of time (fs) as computed by the TD-CISD method. The vertical dashed line at 1 fs denotes the end of the laser pulse.

few cycles. This loss of coherence is due to pure dephasing, a result of the pulse exciting the system into an extremely broad superposition of closely spaced states.

The relative performance of the GPU-accelerated and pure CPU TD-CISD algorithms are presented in Table IV. For the largest system, H_{120} (60 doubly occupied and 60 virtual orbitals), a single time step of TD-CISD is performed about 3.9 times faster by the hybrid implementation than by the pure CPU implementation. For smaller systems, the speedup is much smaller, only a factor of 1.8 for H_{30}. A TD-CISD simulation may require many tens of thousands of time steps; a factor of 3.9 speedup is therefore desirable for these long simulations. The best accelerations for TD-CISD are slightly less than what was observed for CCSD, a result that is consistent with the current implementation strategy. CCSD is a more complicated theory with many more simple terms that can be performed on the CPU, thus boosting performance.

TABLE IV
COMPARISON OF CPU AND HYBRID CPU/GPU IMPLEMENTATIONS OF TD-CISD. TIMINGS PER TIME STEP ARE GIVEN IN SECONDS. THE SYMBOLS o AND v REPRESENT THE NUMBER OF DOUBLY OCCUPIED AND VIRTUAL ORBITALS IN EACH SYSTEM, RESPECTIVELY. SPEEDUP SIGNIFIES THE RELATIVE COST OF CPU ALGORITHM AS COMPARED TO THE HYBRID ALGORITHM.

			Iteration Time (s)		
Molecule	o	v	Hybrid	CPU	Speedup
H_{30}	15	15	0.06	0.11	1.83
H_{40}	20	20	0.20	0.38	1.90
H_{50}	25	25	0.57	1.24	2.18
H_{60}	30	30	1.23	2.88	2.34
H_{70}	35	35	2.68	6.77	2.53
H_{80}	40	40	4.61	12.45	2.70
H_{90}	45	45	8.73	24.32	2.78
H_{100}	50	50	12.86	42.12	3.28
H_{110}	55	55	21.58	72.92	3.38
H_{120}	60	60	28.57	111.34	3.90

V. CONCLUSIONS AND FUTURE WORK

We have presented implementations of the spin-free CCSD and TD-CISD methods that fully utilize the capabilities of a modern heterogenous compute node consisting of a multicore Intel CPU and an NVIDIA Fermi GPU. Both implementations make use of tuned BLAS libraries on both CPU (threaded GotoBLAS) and GPU (Cublas) and thus achieve a significant portion of the theoretical peak performance of each processor. The repeated memory transfers that the limited global memory of the GPU necessesitate result in an unfortunate overhead that, thankfully, can be masked quite effectively by overlapping GPU and CPU computations. By partitioning work appropriately between the processors, we achieve an acceleration of up to a factor of 4.3 using the hybrid hardware relative to the pure CPU algorithm. The hybrid algorithm outperforms the Molpro CCSD implementation on the same CPU hardware by up to a factor of 9.7. We observe similar accelerations for the TD-CISD method, where, for a system with 120 electrons in 120 orbitals, the hybrid implementation can propagate the TDSE 3.9 times more efficiently than can the pure CPU implementation. A factor of 3.9 speedup can be very important for a method such as TD-CISD where the TDSE must be evolved for tens or even hundreds of thousands of iterations.

Future work will include the extension of these implementation strategies to the "gold standard" in quantum chemsitry, the CCSD(T) method. The perturbative triples correction beyond CCSD is a series of very large tensor contractions that is perfectly suited for acceleration by a GPU. Further, both the CCSD and TD-CISD algorithms are amenable to very simple task-based parallelism whereby entire tensor contractions are distributed among nodes (or multiple GPUs within a node). As currently implemented, the CCSD and TD-CISD methods are well suited for a coarse three-GPU parallelization. Three-way parallelism may seem odd at first glance, but such an algorithm is ideal for a system such as the Keeneland system at Oak Ridge National Laboratory, a single node of which consists of two Intel Westmere CPUs and three NVIDIA Fermi GPUs. By tiling or blocking individual contractions, the efficient use of many heterogeneous nodes is certainly possible, but proper load balancing is made very difficult by the disproportionate performance of GPU and CPU.

REFERENCES

[1] I. S. Ufimtsev and T. J. Martínez, "Quantum chemistry on graphical processing units. 1. Strategies for two-electron integral evaluation," *Journal of Chemical Theory and Computation*, vol. 4, no. 2, pp. 222–231, 2008. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct700268q

[2] A. Asadchev, V. Allada, J. Felder, B. M. Bode, M. S. Gordon, and T. L. Windus, "Uncontracted Rys quadrature implementation of up to G functions on graphical processing units," *Journal of Chemical Theory and Computation*, vol. 6, no. 3, pp. 696–704, 2010. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct9005079

[3] A. V. Titov, V. V. Kindratenko, I. S. Ufimtsev, and T. J. Martínez, "Generation of kernels for calculating electron repulsion integrals of high angular momentum functions on GPUs – preliminary results," in *Proceedings of SAAHPC 2010*, 2010, pp. 1–3. [Online]. Available: http://saahpc.ncsa.illinois.edu/papers/paper_53.pdf

[4] I. S. Ufimtsev and T. J. Martínez, "Quantum chemistry on graphical processing units. 2. Direct self-consistent-field (SCF) implementation," *Journal of Chemical Theory and Computation*, vol. 5, no. 4, pp. 1004–1015, 2009. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct800526s

[5] ——, "Quantum chemistry on graphical processing units. 3. Analytical energy gradients, geometry optimization, and first principles molecular dynamics," *Journal of Chemical Theory and Computation*, vol. 5, no. 10, pp. 2619–2628, 2009. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct9003004

[6] K. Yasuda, "Accelerating density functional calculations with graphics processing unit," *Journal of Chemical Theory and Computation*, vol. 4, no. 8, pp. 1230–1236, 2008. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct8001046

[7] C. J. Woods, P. Brown, and F. R. Manby, "Multicore parallelization of Kohn-Sham theory," *Journal of Chemical Theory and Computation*, vol. 5, no. 7, pp. 1776–1784, 2009. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct900138j

[8] L. Genovese, M. Ospici, T. Deutsch, J.-F. Mehaut, A. Neelov, and S. Goedecker, "Density functional theory calculation on many-cores hybrid central processing unit-graphic processing unit architectures," *The Journal of Chemical Physics*, vol. 131, no. 3, p. 034103, 2009. [Online]. Available: http://link.aip.org/link/?JCP/131/034103/1

[9] P. Brown, C. J. Woods, S. McIntosh-Smith, and F. R. Manby, "A massively multicore parallelization of the Kohn-Sham energy gradients," *Journal of Computational Chemistry*, vol. 31, no. 10, pp. 2008–2013, 2010. [Online]. Available: http://pubs.acs.org/doi/abs/10.1002/jcc.21485

[10] L. Vogt, R. Olivares-Amaya, S. Kermes, Y. Shao, C. Amador-Bedolla, and A. Aspuru-Guzik, "Accelerating resolution-of-the-identity second-order Møller-Plesset quantum chemistry calculations with graphical processing units," *Journal of Physical Chemistry A*, vol. 112, no. 10, pp. 2049–2057, 2008. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/jp0776762

[11] R. Olivares-Amaya, M. A. Watson, R. G. Edgar, L. Vogt, Y. Shao, and A. Aspuru-Guzik, "Accelerating correlated quantum chemistry calculations using graphical processing units and a mixed precision matrix multiplication library," *Journal of Chemical Theory and Computation*, vol. 6, no. 1, pp. 135–144, 2010.
[12] A. Koniges, R. Preissl, J. Kim, D. Eder, A. Fisher, N. Masters, V. Mlaker, S. Ethier, W. Wang, and M. Head-Gordon, "Application acceleration on current and future Cray platforms," in *CUG 2010, the Cray User Group meeting*, Edinburgh, Scotland, May 2010. [Online]. Available: http://www.nersc.gov/news/reports/technical/Cug2010Alice.pdf
[13] A. G. Anderson, W. A. Goddard III, and P. Schröder, "Quantum Monte Carlo on graphical processing units," *Computer Physics Communications*, vol. 177, no. 3, pp. 298–306, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.cpc.2007.03.004
[14] K. Esler, J. Kim, L. Shulenburger, and D. Ceperley, "Fully accelerating quantum Monte Carlo simulations of real materials on GPU clusters," *Computing in Science and Engineering*, vol. 99, no. PrePrints, 2010.
[15] A. Gothandaraman, G. D. Peterson, G. Warren, R. J. Hinde, and R. J. Harrison, "FPGA acceleration of a quantum Monte Carlo application," *Parallel Computing*, vol. 34, no. 4-5, pp. 278–291, 2008, Reconfigurable Systems Summer Institute 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6V12-4S03R96-2/2/a82f9a0290231c0adfbde92a9d0ec2f4
[16] A. E. DePrince III and J. R. Hammond, "Coupled cluster theory on graphics processing units I. The coupled cluster doubles method," *Journal of Chemical Theory and Computation*, vol. 7, no. 5, pp. 1287–1295, 2011. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct100584w
[17] W. Ma, S. Krishnamoorthy, O. Villa, and K. Kowalski, "GPU-based implementations of the noniterative regularized-CCSD(T) corrections: Applications to strongly correlated systems," *Journal of Chemical Theory and Computation*, vol. 7, no. 5, pp. 1316–1327, 2011. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ct1007247
[18] H.-J. Werner, P. J. Knowles, F. R. Manby, M. Schütz, P. Celani, G. Knizia, T. Korona, R. Lindh, A. Mitrushenkov, G. Rauhut, T. B. Adler, R. D. Amos, A. Bernhardsson, A. Berning, D. L. Cooper, M. J. O. Deegan, A. J. Dobbyn, F. Eckert, E. Goll, C. Hampel, A. Hesselmann, G. Hetzer, T. Hrenar, G. Jansen, C. Köppl, Y. Liu, A. W. Lloyd, R. A. Mata, A. J. May, S. J. McNicholas, W. Meyer, M. E. Mura, A. Nicklass, P. Palmieri, K. Pflüger, R. Pitzer, M. Reiher, T. Shiozaki, H. Stoll, A. J. Stone, R. Tarroni, T. Thorsteinsson, M. Wang, and A. Wolf, "MOLPRO, version 2010.1, a package of ab initio programs," Cardiff, UK, 2010, see http://www.molpro.net.
[19] G. E. Scuseria, C. L. Janssen, and H. F. Schaefer III, "An efficient reformulation of the closed-shell coupled cluster single and double excitation (ccsd) equations," *Journal of Chemical Physics*, vol. 89, no. 12, 1988.
[20] P. Piecuch, S. A. Kucharski, K. Kowalski, and M. Musiał, "Efficient computer implementation of the renormalized coupled-cluster methods: The R-CCSD[T], R-CCSD(T), CR-CCSD[T], and CR-CCSD(T) approaches," *Computer Physics Communications*, vol. 149, no. 2, pp. 71–96, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/B6TJ5-46G4SD7-4/2/86165c69dd473dea6dd41dcc5b05648c
[21] P. Pulay, "Convergence acceleration of iterative sequences. the case of scf iteration," *Chemical Physics Letters*, vol. 73, no. 2, pp. 393 – 398, 1980. [Online]. Available: http://www.sciencedirect.com/science/article/B6TFN-4490431-79/2/5234d016c4cad3b7ec90ac847d09d282
[22] T. Klamroth and M. Nest, *Physical Chemistry and Chemical Physics*, vol. 11, p. 349, 2009.
[23] B. Schlegel, S. M. Smith, and X. Li, *Journal of Chemical Physics*, vol. 126, p. 244110, 2008.
[24] S. K. Gray and J. M. Verosky, *Journal of Chemical Physics*, vol. 100, p. 5011, 1994.
[25] S. K. Gray and D. E. Manolopoulos, *Journal of Chemical Physics*, vol. 104, p. 7099, 1996.
[26] J. M. Sanz-Serna and A. Portillo, *Journal of Chemical Physics*, vol. 104, p. 2349, 1996.
[27] T. D. Crawford, C. D. Sherrill, E. F. Valeev, J. T. Fermann, R. A. King, M. L. Leininger, S. T. Brown, C. L. Janssen, E. T. Seidl, J. P. Kenny, and W. D. Allen, "PSI3: An open-source *ab initio* electronic structure package," *Journal of Computational Chemistry*, vol. 28, no. 9, pp. 1610–1616, 2007.

ACKNOWLEDGMENTS

AED is supported by the Computational Postdoctoral Fellowship program at Argonne National Laboratory. JRH is supported by the Director's Postdoctoral Fellowship program at Argonne National Laboratory.

This research used the "Dirac" GPU testbed system of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

We gratefully acknowledge use of "Fusion," a computing cluster operated by the Mathematics and Computer Science Division at Argonne National Laboratory as part of its Laboratory Computing Resource Center, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357.