Placing Rigorous Bounds on Numerical Errors in Hartree−Fock Energy Computations

Pete P. Janes and Alistair P. Rendell*

Research School of Computer Science, The Australian National University, Canberra, Australia

ABSTRACT: The accuracy of electronic structure calculations are affected to some degree by numerical errors. Assessing whether these errors are at an acceptable level for chemical accuracy is difficult. This paper demonstrates how interval arithmetic can be used to address this issue in the context of a Hartree−Fock computation. Using the method proposed here, the effect of system size and basis set type on the overall numerical accuracy of the Hartree−Fock total energy is studied. Consideration is also given to the impact of various algorithmic design decisions. Examples include the use of integral screening, computing some integrals in single precision, and reducing the accuracy of the interpolation tables used to compute the reduced incomplete Gamma function required by some integral evaluation algorithms. All of these issues have relevance to the use of novel computing devices such as graphics processing units (GPU) and the Sony Toshiba IBM Cell Broadband, to exascale and green computing, and to the exploitation of the emerging generation of massively multicore processors.

1. INTRODUCTION

During the past 20 years, the vast majority of computational chemistry programs have been written to use IEEE 754 double precision floating point arithmetic.[1] This was motivated both by the perception that double precision was required in order to obtain sufficiently accurate results and the fact that efficient implementations of this standard were available on all commonly used high performance microprocessors. With a few notable exceptions,[2,3] relatively little attention was given to using alternative levels of numerical precision. In the past few years, this situation has changed dramatically. This has been largely prompted by the move by manufacturers of graphics and gaming hardware into the general purpose computing market, the ability of this new hardware to perform floating point computations at a rate that was significantly higher than that of contemporary general purpose microprocessors, and the fact that this was (initially) only possible when using single precision arithmetic with rounding modes that were not entirely IEEE 754 compliant. Two high profile early examples are the Sony Toshiba IBM (STI) Cell Broadband Engine[4] that forms the basis for the Playstation 3 and the NVIDIA GTX8800 CUDA based graphics card.[5]

More recently, initiatives in exascale computing, green computing, and many-core architectures have provided added impetus for reconsidering the role of numerical precision in computational chemistry codes. In these cases, the sheer number of operations performed, the energy required to perform the operations, and the available bandwidth for moving data on and off the processor chip are all causes for wanting to represent data with the minimal number of bytes possible. Also, developments in field programmable gate array (FPGA) technology mean that computational devices can now be constructed on the fly to use arbitrary levels of precision that may not be standard single or double precision.

Mindful of the above, we have developed a Hartree−Fock code that uses interval arithmetic to place rigorous bounds on the numerical errors associated with computing the total electronic energy. Our aim is to build a tool that can be used to explore the effects of numerical precision and various other underlying numerical approximations on the quality of the final result. Examples considered here include the effects of basis set and system size, as well as "hidden" issues such as the use of different levels of integral prescreening and the choice of different interpolation schemes for evaluating the base quantities from which all two-electron integrals are assembled.

In *interval analysis*,[6] a quantity is represented as an interval $X = [\underline{X}, \overline{X}]$, which contains the set of all real numbers between a *lower bound* $\underline{X}$ and an *upper bound* $\overline{X}$. Intervals can be used to bound all uncertainties and errors, including those caused by rounding to finite precision, truncation or approximating a series, limits in the accuracy of a measured quantity, etc. When computing with intervals, the uncertainties encapsulated within each interval are propagated on to the final result. That is, the final result is an interval which rigorously bounds the entire range of possible results when all of those uncertainties are taken into account. Inferences can then be made about the errors associated with the result based on the width of the final interval.

In comparison to a traditional code that represents a data quantity using a single floating point number, an interval code will use two floating point numbers, one representing the lower bound and one representing the upper bound. Where a traditional floating point code rounds the result of a floating point operation to a machine representable number, an interval code must broaden the interval result to the nearest machine representable interval that includes the exact interval result. Interval codes should also take care to include errors associated with truncating infinite series when reporting the result, e.g., in functions to compute e^x, $\log(x)$, $\sin(x)$, etc. Needless to say, the goal in an interval code is to minimize the width of the final interval.

Received: January 12, 2011
Published: May 13, 2011

The work reported here follows on from a number of related works. Specifically, Takashima et al.[7] investigated whether large scale Hartree−Fock energy (E_{HF}) calculations were chemically accurate given a potentially large accumulation of numerical errors. Their study focused on the calculation of E_{HF}, given an initial set of integrals and a density matrix. Error contaminated data were simulated by uniformly perturbing each double precision floating point quantity involved in the Fock matrix construction step at the last bit of the mantissa. The error was measured by comparing E_{HF} calculated using contaminated data with E_{HF} calculated using uncontaminated data, calculating both cases using double precision floating point arithmetic. This approach can be implemented using existing quantum chemistry software, although it makes the broad assumption that the errors in the initial data are uniformly perturbed. By contrast, when using intervals, errors are modeled without making such assumptions. In the same study, the authors applied a technique known as partial summation to improve the accuracy of E_{HF} and used linear regression to predict errors for very large problems involving 10 000 basis functions.

Ramdas et al.[3] proposed the use of interval arithmetic as an *off-line design space tool* to experiment with different arithmetic unit configurations. In particular, they were interested in whether a mix of different arithmetic precisions could be used for calculating the Hartree−Fock energy on an FPGA, while maintaining an acceptable degree of chemical accuracy. Their work adopted the same approach as Yasuda:[2] determining which integral or batches of integrals to calculate using single precision based on its Schwarz[8] upper bound, the assumption being that smaller integrals can be calculated in single precision without substantially affecting the accuracy of the final result. The investigation in Ramdas et al. was, however, confined to a single H_2O molecule using the 6-31G** basis set.

The rest of this paper is organized as follows. Section 2 gives a more detailed description of interval analysis and how it can be used as an error analysis tool. Section 3 outlines the interval implementation of the Hartree−Fock method, while section 4 details the experimental methodology used and presents and discusses the results. Section 5 concludes the paper and discusses future work.

2. INTERVAL ANALYSIS

Interval arithmetic was proposed by Ramon Moore in 1959.[9] It represents numerical quantities in terms of closed intervals $X = [\underline{X}, \overline{X}]$, where each interval represents the set of all real numbers between the lower bound $\underline{X} \in \mathbb{R}$ and upper bound $\overline{X} \in \mathbb{R}$. The set of interval numbers is denoted as $I(\mathbb{R})$, whereas the set of n dimensional vectors with interval elements is denoted as $I(\mathbb{R}^n)$.

Intervals were first proposed as a tool for bounding rounding errors;[9] however, in practice, an interval can be used to bound any uncertain quantity. This includes any combination of rounding errors, truncation errors, measurement errors, unknown/poorly understood parameters, etc.

2.1. Interval Arithmetic. The set of basic arithmetic operations, $\bullet = +, -, \times$, between two intervals is defined such that

$$X \bullet Y = \{x \bullet y | x \in X, y \in Y\} \tag{1}$$

By this definition, the arithmetic operations between two intervals $X \in I(\mathbb{R})$ and $Y \in I(\mathbb{R})$ result in another interval that contains the results of operations between all possible combinations of real values contained in X and Y. This property is generally referred to as the *containment property* for interval arithmetic operations.[10]

For practical purposes, interval arithmetic operations are almost always carried out using IEEE 754 floating point arithmetic, referred to as *Floating Point Interval Arithmetic*.[6] To ensure containment, both upward and downward rounding modes (instead of round to nearest) are used:

$$\begin{aligned}
X + Y &= [\oplus_{\nabla}(\underline{X}, \underline{Y}), \oplus_{\Delta}(\overline{X}, \overline{Y})] && \text{addition}\\
X - Y &= [\ominus_{\nabla}(\underline{X}, \overline{Y}), \oplus_{\Delta}(\overline{X}, \underline{Y})] && \text{subtraction}\\
X \times Y &= [a, b] && \text{multiplication}\\
a &= \min(\otimes_{\nabla}(\underline{X}, \underline{Y}), \otimes_{\nabla}(\underline{X}, \overline{Y}),\\
&\quad \otimes_{\nabla}(\overline{X}, \underline{Y}), \otimes_{\nabla}(\overline{X}, \overline{Y}))\\
b &= \max(\otimes_{\Delta}(\underline{X}, \underline{Y}), \otimes_{\Delta}(\underline{X}, \overline{Y}),\\
&\quad \otimes_{\Delta}(\overline{X}, \underline{Y}), \otimes_{\Delta}(\overline{X}, \overline{Y}))\\
1/Y &= [\oslash_{\nabla}(1, \overline{Y}), \oslash_{\nabla}(1, \underline{Y})] && \text{division}\\
X/Y &= X \times 1/Y
\end{aligned} \tag{2}$$

where the subscripts Δ and ∇ refer respectively to upward and downward rounding. With no specific hardware support, interval arithmetic requires more instructions and storage to implement than floating point arithmetic.

2.2. Interval Functions. An *interval function* is a function which takes a set of intervals as input and returns a set of intervals as output.[6] The containment principle is extended to interval functions. We say that an interval function $\mathbf{F}: I(\mathbb{R}^m) \rightarrow I(\mathbb{R}^n)$ is an *interval extension*[6] of a function $\mathbf{f}: \mathbb{R}^m \rightarrow \mathbb{R}^n$ if

$$\mathbf{F}(\mathbf{X}) \supseteq \{\mathbf{f}(\mathbf{x}) | \mathbf{x} \in \mathbf{X}\} \tag{3}$$

An interval extension of $\mathbf{f}$ therefore contains the range of $\mathbf{f}$ over the domain spanned by $\mathbf{X} \in I(\mathbb{R}^m)$. Furthermore, if $\mathbf{g}(\mathbf{x})$ is an approximation function of $\mathbf{f}(\mathbf{x})$ and ε is the maximum truncation error in $\mathbf{X}$, then if $\mathbf{G}(\mathbf{X})$ is the interval extension of $\mathbf{g}(\mathbf{x})$, we can write the interval extension of $\mathbf{f}(\mathbf{x})$ as $\mathbf{F}(\mathbf{X}) = [\underline{\mathbf{G}(\mathbf{X})} - \varepsilon, \overline{\mathbf{G}(\mathbf{X})} + \varepsilon]$.

The interval extension of standard mathematical functions such as e^X, $\log(X)$, $\sin(X)$, and $\cos(X)$ are relatively straightforward to implement, e.g., $e^X = [e^{\underline{X}}, e^{\overline{X}}]$. For *rational functions*[6] whose expression consists entirely of the four basic arithmetic operations, interspersed with standard mathematical functions for which an interval extension is known, there is a also relatively straightforward procedure to obtain an interval extension: first, write the mathematical expression for the function; then, replace each elementary arithmetic operation by its equivalent interval arithmetic operation and each standard function by its interval extension, and then replace each variable by an interval. An interval extension formulated using this procedure is known as a *natural interval extension*.[6]

2.3. Interval Arithmetic for Error Analysis. Interval extensions provide a way to calculate rigorous bounds on the range of function values due to uncertain inputs. In combination with interval arithmetic, it allows uncertainties caused by rounding and truncation errors to be propagated toward the final result.

The final interval will almost always overestimate the actual numerical errors due to the fact that interval arithmetic must account for the worst possible interaction between the interval quantities involved in the calculation. The problem of finding the exact range over a given domain of uncertainty is generally an NP-Hard problem. This implies that interval arithmetic can only provide a *worst-case error analysis* of the calculation in question.

Therefore, we can only rigorously state what set of factors can or cannot guarantee accuracy within a given threshold. However, other useful inferences can be made about the behavior of numerical errors, albeit with the above caveat in mind.

Another issue that also causes interval error overestimation is the *dependency problem*.[10] Interval arithmetic assumes all quantities are independent of one another. Ignoring the dependencies between quantities leads to illogical results. For example, suppose $X = [0,1]$; then using interval arithmetic, $X - X$ returns $[-1,1]$ instead of $[0,0]$. It is not difficult to deduce that this problem is likely to affect most interval calculations, except those involving expressions where each variable occurs only once, such as $X_1 + X_2 + X_3 + ...$ In this work, however, the dependency effects will be small, as we mainly deal with near-degenerate intervals containing relatively small rounding and truncation errors.

Similar approaches which allow errors to be propagated exist in uncertainty and sensitivity analysis;[11] almost all of these involve extensively sampling the domain of uncertain variables in order to obtain a statistical representation of variations in the result. However, this approach is difficult to apply for larger models with many uncertain quantities. In Takashima et al.,[7] the errors in the Hartree−Fock total energy due to numerical errors in the input (the one and two electron integrals) are estimated by comparing the results calculated from a set of perturbed inputs to those calculated from a set of unperturbed inputs. This is not as rigorous as a sampling based approach but reflects the difficulties involved in effectively sampling a large set of uncertain parameters which exists in a typical Hartree−Fock calculation.

3. INTERVAL HARTREE−FOCK

Hartree−Fock theory provides an *ab initio* model of the interactions between particles on the molecular scale. Although not a particularly accurate model, it provides the framework from which many other, more advanced, methods are derived. For this reason, Hartree−Fock theory is still widely used as a starting point for studies of electronic structure. The Hartree−Fock energy for a closed shell molecular system with n electrons can be expressed as

$$E_{\mathrm{HF}} = V_{nn} + 2\sum_{\mu}^{n/2} H_{\mu} + \sum_{\mu}^{n/2}\sum_{v}^{n/2}(2J_{\mu,v} - K_{\mu,v}) \qquad (4)$$

where V_{nn} is the nuclear repulsion energy and H_μ, $J_{\mu,v}$, and $K_{\mu,v}$ are integrals over the *Core-Hamiltonian*, *Coulomb*, and *Exchange* operators. The latter are defined in terms of the molecular orbitals (Φ) as

$$\begin{aligned} H_\mu &= (\Phi_\mu|H_{\mathrm{core}}|\Phi_\mu) \\ J_{\mu,v} &= (\Phi_\mu\Phi_\mu|\Phi_v\Phi_v) \\ K_{\mu,v} &= (\Phi_\mu\Phi_v|\Phi_\mu\Phi_v) \end{aligned} \qquad (5)$$

with the Coulomb and Exchange integrals collectively referred to as the two-electron repulsion integrals (ERI).

Normally, the set of molecular orbitals $\{\phi\}$ is expanded in terms of a linear combination of N atomic orbitals $\{\chi\}$:

$$\phi_\mu = \sum_{a=1}^{N} C_{\mu a}\chi_a \qquad (6)$$

where $\mathbf{C} = \{C_{\mu a}\}, \forall \mu,a \in \{1, 2, 3, .., N\}$ are the *molecular orbital coefficients*.

For most practical purposes, the atomic orbitals in the LCAO are represented by *Contracted Gaussian Functions* (CGF):

$$\begin{aligned} \chi_a(r) &= \sum_{k=1}^{K_a} D_{ak}\,\chi_{ak}(r), \text{ where} \\ \chi_{ak}(r) &= (X - X_a)^{x_a}(Y - Y_a)^{y_a}(Z - Z_a)^{z_a}\, e^{\alpha_{ak}|r - r_a|^2} \end{aligned} \qquad (7)$$

where K_a is the degree of contraction for function a, D_{ak} is a contraction coefficient, and $\chi_{ak}(r)$ is a *Primitive Gaussian Function* (PGF) with exponent $\{\alpha_{ai}\}$, coordinates $r_a = (X_a, Y_a, Z_a)$, and angular components $l_a = \{x_a, y_a, z_a\}$.

A Self-Consistent Field (SCF) calculation involves minimizing the value of E_{HF} with respect to the molecular orbital coefficients $\mathbf{C}$ to produce the ground state energy, E_0. This problem can be solved iteratively by reformulating the minimization problem as a nonlinear eigenvalue problem known as the *Roothaan equations*

$$\mathbf{FC} = \mathbf{SC}\varepsilon \qquad (8)$$

where $\mathbf{F} \in \mathbb{R}^{N\times N}$ is the *Fock matrix* that represents the one and two-electron interactions, $\mathbf{S} \in \mathbb{R}^{N\times N}$ is the *overlap matrix* that represents the overlap between atomic orbitals, and $\varepsilon \in \mathbb{R}^{N\times N}$ is the *molecular orbital energies*.

In this paper, we focus on evaluating the numerical errors in the ground state Hartree−Fock total energy E_{HF}—using as input the set of ground state molecular orbital coefficients calculated using a conventional floating point SCF code. The interval bounds of the total energy reflect errors propagated from (i) the one- and two-electron integrals, (ii) the construction of the Fock matrix, (iii) the calculation of the nuclear repulsion, and (iv) the calculation of the total energy. The three latter steps are implicit in eq 4.

Of the various operations performed during a Hartree−Fock/self-consistent field calculation, evaluating the ERIs is by far the most computationally demanding.[12] It is also the aspect of the calculation that requires the most attention when developing an interval Hartree−Fock code.

A primitive ERI involving four atomic orbitals is given by

$$[\chi_a\chi_b|\chi_c\chi_d] \equiv \int\!\!\int \frac{\chi_a(r_1)\,\chi_b(r_1)\,\chi_c(r_2)\,\chi_d(r_2)\,\mathrm{d}r_1\,\mathrm{d}r_2}{|r_1 - r_2|} \qquad (9)$$

while a contracted ERI is given by

$$(\chi_a\chi_b|\chi_c\chi_d) \equiv \sum_{i=1}^{K_i}\sum_{j=1}^{K_j}\sum_{k=1}^{K_k}\sum_{l=1}^{K_l} D_{ai}D_{bj}D_{ck}D_{dl}[\chi_{ai}\chi_{bj}|\chi_{ck}\chi_{dl}] \qquad (10)$$

with the number of ERIs required for a given problem scaling as $O(N^4)$ in the problem size.

There are a number of numerical approaches for evaluating primitive ERIs. Some of the most widely used are by *McMurchie−Davidson*[13] (MD), *Obara−Saika*[14] (OS), *Rys−Depuis−King*[15] (RDK), and *Head-Gordon−Pople*[16] (HGP). All of these schemes use recursion relations to assemble the final integral from some very simple integrals. In the case of the MD, OS, and HGP schemes, the simple integrals are *Reduced Incomplete Gamma Functions*[17] which are denoted as $F_m(T)$:

$$F_m(T) = \int_0^1 t^{2m}\, e^{-Tt^2}\, \mathrm{d}t \qquad (11)$$

In the above, T is a real non-negative number with a value that is dependent on the distance between the basis function centers and

Table 1. Decimal Digits of Precision of E_{HF} for TIP4P Water Clusters of n Water Molecules Computed Using 3-21G, 6-31G**, cc-pVDZ, 6-31G++**, and 6-311G(2df,2pd) Basis Sets[a]

$(H_2O)_n$	N	3-21G	N	6-31G**	N	cc-pVDZ	N	6-31++G**	N	6-311G(2df,2pd)
2	26	12.63	50	12.26	50	12.14	62	11.98	130	11.48
3	39	12.30	75	11.92	75	11.76	93	11.63	195	11.12
4	52	12.08	100	11.69	100	11.57	124	11.41	260	10.88
5	65	11.87	125	11.48	125	11.31	155	11.19	325	10.56
6	78	11.70	150	11.29	150	11.16	186	11.01		
7	91	11.56	175	11.15	175	11.04	217	10.85		
8	104	11.41	200	11.00	200	10.85	248	10.69		
9	117	11.28	225	10.88	225	10.73	279	10.58		
10	130	11.17	250	10.77	250	10.64	310	10.46		
11	143	11.08								
12	156	11.00								
13	169	10.92								
14	182	10.84								
15	195	10.77								

[a] N is the number of basis functions.

the values of their exponents, while m is an integer with a value that depends on the total angular momentum of all of the Gaussian functions in the ERI.

There are no general analytical approaches for evaluating $F_m(T)$. Many years ago, Shavitt presented a scheme for evaluating $F_m(T)$ that used either a series or an asymptotic approximation depending on the value of T.[18] This approach was subsequently refined to involve precomputing a select number of values for $F_m(T)$ using the Shavitt scheme and then using Taylor or Chebyshev polynomial interpolation to compute specific values.[19] Truncation errors arise in both the Shavitt scheme and in the interpolation, and it is important that any interval scheme to evaluate ERIs correctly bounds both of these sources of errors. We discuss how to achieve this in a previous paper.[20]

3.1. Implementation Details. The interval extension of the Hartree–Fock energy (eq 4) was formulated as a natural interval extension using the procedure described in section 2.2. For this study, ERIs are evaluated using a natural interval extension of the Head-Gordon–Pople (HGP) algorithm,[16] with values of $F_m(T)$ evaluated by default using sixth-degree Chebyshev polynomial interpolation. Truncation errors in $F_m(T)$ were manually bounded and propagated.

4. COMPUTATIONAL RESULTS

The following experiments investigate the numerical errors in E_{HF} as the size and complexity of the molecular problem increases, and when different algorithms are used. We focus particularly on issues relevant to the use of Graphics Processing Units (GPUs). It is important to note, however, that all the results given here are obtained using the interval HF code run on a conventional CPU; that is, we are using the interval code to pose "what-if" questions without needing to have access to hardware that supports that "what-if" scenario. Before considering the experiments in detail, it is pertinent first to define and discuss the error terminology used.

Supposing that E^I_{HF} is the interval bound of E_{HF}, the relative numerical error is expressed as

$$\text{err} = 2.0 \times \frac{\overline{E^I_{HF}} - \underline{E^I_{HF}}}{\overline{E^I_{HF}} + \underline{E^I_{HF}}}$$

This is equivalent to dividing the width of E^I_{HF} by its midpoint. For clarity, we will often state the result in terms of *precision* instead of error, where $-\log_{10}(\text{err})$ is used to indicate the number of *decimal digits of precision* in the result. For instance, err = 1×10^{-13} implies 13 decimal digits of precision. If the computation is implemented in double precision, there are a maximum of 16 decimal digits, so a relative error of 1×10^{-13} implies that three decimal digits have been lost due to numerical errors. The results should be construed as an estimate of errors in the *worst case*, since we are using interval arithmetic.

The total electronic energy is generally regarded to be *chemically accurate* if its error is no greater than 0.01 kcal/mol or around 1.59×10^{-5} Hartree.[7] As a consequence, larger systems with larger total energies will require more digits of precision in order to be chemically accurate. However, calculations on large molecules also require more operations to be performed so there is greater potential for accumulating errors. As interval arithmetic bounds the numerical errors, it can be used to rigorously guarantee that chemical accuracy is achieved.

Experimental Platform. All experiments were conducted on a Sun Microsystem V1280 UltraSPARCIIICu System with 12 × 900 MHz cores, using the Solaris 10 operating system.[21] All programs were implemented in either C++ or SPARC assembler and compiled using SunStudio 11 compilers at patch level 20060426.[22] The build options used are

```
-fast -xopenmp -xia -xtarget=ultra3 -
xarch=v9b -xlic_lib=sunperf
```

where the `-xia` flag enables the interval arithmetic library.[23] Source code for all the interval SCF programs used here is available at the authors' Web site.[24]

4.1. Effect of System Size and Basis Set. To study the effects of system and basis set size, we have used H_2O clusters of varying size and basis sets that range from the near minimal 3-21G set to the extensive 6-311G(2df,2pd) basis set. The geometries used are those of the TIP4P water clusters at the Cambridge Cluster Database.[25]

Results are shown in Table 1 for systems with total basis set sizes of up to around 325 basis functions. As expected, increasing the number of atoms and/or the basis set size decreases the number of decimal digits of precision. Use of the 3-21G basis set

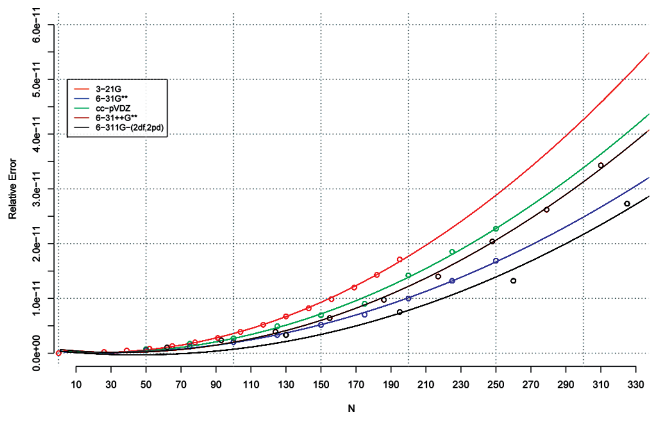

Figure 1. Relative error of E_{HF} for water clusters fitted against the number of basis functions N using second degree polynomial regression.

and going from 2 to 15 water molecules results in a loss of two significant figures in the total energy. In terms of chemical accuracy, another significant figure is effectively lost as the total energy also increases by more than 1 order of magnitude on going from 2 to 15 water molecules. With 15 water molecules and a total energy on the order of 10^3, 10.77 digits of decimal precision are still, however, enough to give total energies accurate to 10^{-5} Hartrees.

For the smaller systems, increasing the basis set size from 3-21G to 6-311G (2df,2pd) also results in a loss of more than one decimal digit of precision in E_{HF}. Comparing the 6-31G** and cc-pVDZ basis sets, both of which have the same number of contracted functions but were achieved using different contraction schemes, reveals that the cc-pVDZ correlation consistent basis set gives slightly fewer digits of precision. Adding diffuse functions to the 6-31G** basis gives a slight further loss in precision. Interestingly, the number of digits of precision appears to be relatively constant for a given total number of functions regardless of whether this results from a large number of atoms or from a large basis set.

To explore the latter point further, we plot in Figure 1 the relative errors from Table 1 as a function of N. A second-degree polynomial regression was used to fit the data for each basis set type. The R^2 (coefficient of determination) values of the regressions were found to be 0.9989, 0.9983, 0.9958, 0.9977, and 0.9841 for the 3-21G, 6-31G**, cc-pVDZ, 6-31++G, and 6-311G (2df,2pd) basis sets, respectively. This strongly suggests that a second-degree polynomial model is sufficient to explain the variation of numerical errors with N. Linear regression was also considered, but it was found not to yield as good a fit. The success of the second polynomial is likely to be related to the numerical complexity of the Hartree–Fock method. For example, in an N-term summation of $O(N)$ complexity, the growth in numerical errors is on the order $O(N)$ but also depends on the type of terms involved.[26] In the Hartree–Fock method, the numerical complexity grows as $O(N^4)$. For real systems, however, larger problem sizes mean larger spatial extent, and since the Coulombic interaction decreases with distance, it is not surprising to find that the numerical errors increase in a nonlinear fashion, but at a rate that is less than quartic. Finally, aggregating together the results for all basis set types and then applying regression also leads to a poorer fit.

While the least precise result found in Table 1 is still well within chemical accuracy, the loss of precision will become an

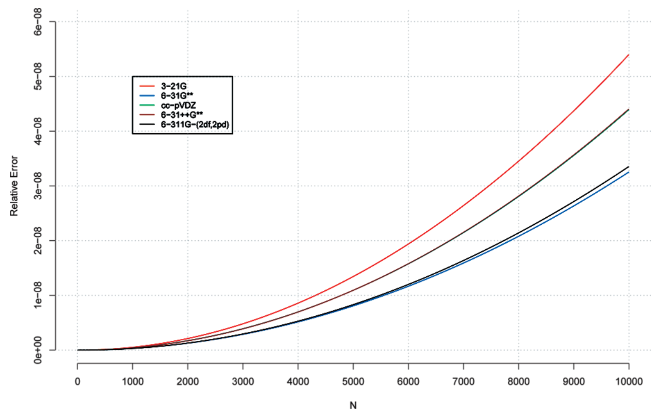

Figure 2. Predicted relative error of E_{HF} for water clusters of size up to $N = 10\,000$ using second degree polynomial regression models derived from existing data.

Table 2. Predicted Relative Numerical Error at $N = 10\,000$ for TIP4P Water Clusters Computed Using Different Types of Basis Sets

3-21G	6-31G**	cc-pVDZ	6-31++G**	6-311G(2df,2pd)
5.39×10^{-8}	3.25×10^{-8}	4.39×10^{-8}	4.40×10^{-8}	3.35×10^{-8}

issue in larger clusters and/or with larger basis sets. The polynomial regression models shown in Figure 1 can be used to predict errors for larger values of N. This is shown in Figure 2 for the four basis sets used. The plot indicates that relative numerical errors are on the order of 1×10^{-8} when $N = 10\,000$, with the specific values given in Table 2.

The largest system considered in the experiment above was $(H_2O)_5$ 6-311G(2df,2pd) with 325 basis functions. The system sizes that can be studied are limited by the performance of the interval code. Interval computation is inherently slower than floating point computation due to the additional storage requirement and the additional number of floating operations required to achieve a single interval arithmetic operation (eq 2). The code implemented for this paper is also experimental in nature and therefore not optimized to the same degree as better established floating point codes. Moreover, in order to guarantee rigorous bounds on numerical errors, time saving heuristics such as two-electron integral screening cannot be used as with floating point codes. In fact, all $O(N^4)$ two-electron integrals were calculated using interval arithmetic in our implementation. In actual experiments, the interval Hartree–Fock code was observed to be around 10 to 25 times slower than its floating point equivalent.

4.2. Effect of Two-Electron Integral Screening.

It is customary in HF calculations to prescreen the ERIs and only compute those with a magnitude greater than some threshold. The most widely used screening technique is based on the Schwartz test.[8] This sets the following upper bound for value of an ERI:

$$|(\chi_a\chi_b|\chi_c\chi_d)| \leq \kappa_{ab}\kappa_{cd}, \text{where } \kappa_{ab} = \sqrt{(\chi_a\chi_b|\chi_a\chi_b)} \quad (12)$$

Use of Schwartz screening requires precomputation of two center ERIs of the form $(\chi_a\chi_b|\chi_a\chi_b)$. Before computing a specific integral, the value of $\kappa_{ab}\kappa_{cd}$ is computed and compared against the *screening threshold* τ. If $\kappa_{ab}\kappa_{cd} > \tau$, the actual integral is

Table 3. Decimal Digits of Precision of E_{HF} for Different ERI Screening Thresholds[a]

			τ			
system	N	no. ERI ($\times 10^6$)	ε	1×10^{-12}	1×10^{-8}	1×10^{-4}
$(H_2O)_4$	100	11.29	11.69	11.59 [13.74%]	8.43 [34.84%]	4.05 [68.93%]
alanine	125	29.11	11.41	11.34 [5.80%]	8.07 [17.65%]	3.44 [50.31%]
serine	140	44.77	11.30	11.24 [7.94%]	7.98 [21.73%]	3.40 [54.73%]
cytocine	145	59.77	11.24	11.18 [9.08%]	7.91 [23.27%]	3.37 [53.30%]
$(H_2O)_8$	200	154.91	11.00	10.90 [23.92%]	7.90 [53.02%]	3.60 [85.60%]

[a] The entries inside the square brackets indicate the percentage of unique ERI that is screened with respect to each cut-off.

evaluated; otherwise, it is neglected. For efficiency, Schwartz screening is usually performed for batches of integrals, not for individual integrals.

In Schwartz screening, choosing a value for τ represents a trade-off between speed and accuracy: set too high and the accuracy will be affected, set too low and the time to solution increases but with insignificant improvement in numerical accuracy. Interval arithmetic can be used to quantify the numerical error that results from using different values of τ, and to determine rigorously the maximum value of τ possible while still maintaining chemical accuracy. To do this, we assign each screened integral an interval value of $[-\kappa_{ab}\kappa_{cd},\kappa_{ab}\kappa_{cd}]$.

In Table 3, we show the number of digits of precision in E_{HF} for ERI screening thresholds ranging from the machine epsilon (ε) to 1×10^{-4} for computations performed using a 6-31G** basis set. Five different systems are considered, two water clusters and three amino acid systems. The values in square brackets denote the percentage of unique nonzero ERIs that are screened away.

The results show that when using screening at machine epsilon, E_{HF} is accurate to between 11 and 12 significant digits. When using a screening value of 1×10^{-12}, there is minimal effect on the number of significant digits in E_{HF}, but the number of computed integrals decreases by between 6 and 24%. Increasing the screening value to 1×10^{-8} results in a significantly larger loss of precision and values for E_{HF} that are now barely accurate to 10^{-5} Hartrees. Further increasing the value of τ clearly gives unsatisfactory results.

4.3. Effect of Using Mixed Precision. While general purpose CPUs have traditionally provided strong double precision performance, this has not always been the case for GPUs and other specialized processors. For example, the high end NVIDIA GTX280 GPU is capable of 933 GFLOP/s in single precision, but only 78 GFLOP/s in double precision. Similarly the CellBE PowerXCell 8i is capable of 230.4 GLOP/s in single precision, but only 108.8 GFLOP/s in double precision.

To early users, it was clear that in order to harness the full potential of these systems, single precision arithmetic must be prioritized over double precision arithmetic. Yasuda[2] sought to address this issue by partitioning integrals based on their Schwarz upper bounds: computing all integrals below some bound (λ_{GPU}) in single precision on the GPU, while other integrals greater than λ_{GPU} were calculated in double precision on the host CPU. The larger the value of λ_{GPU}, the greater the proportion of the ERIs calculated in single precision and the faster the overall computation, but the lower the overall accuracy of the result. Yasuda showed that evaluating integrals entirely in single precision was not accurate enough for production runs.

Ufimtsev and Martinez,[27] while accepting Yasuda's approach, questioned whether mixing different floating point precisions was worthwhile given the arrival of double precision capable GPUs designed specifically for scientific computing. It can be argued, however, that GPUs first emerged as a cost-effective solution not for scientific computing but for an entirely different market segment where double precision is not important. This market segment is considerably larger than for scientific computing and will continue to only require single precision floating point and integer arithmetic for the foreseeable future. Thus, the demands of scientific computing are always going to come second in driving GPU innovation. However, and as discussed in the Introduction, there are a number of other factors that motivate the use of short data types where possible.

Interval arithmetic can be used to model the numerical errors in E_{HF} due to different thresholds for λ_{GPU}. This differs from previously published work that compares single precision results to double precision results in that the intervals rigorously bound interactions with other sources of errors, not only those due to the value of λ_{GPU}. To do this, all integrals with a Schwarz upper bound below λ_{GPU} are calculated using single precision interval arithmetic, while the rest are calculated using the default double precision interval arithmetic.

Results obtained with different values of λ_{GPU} are given in Table 4 for some of the water cluster systems using the 6-31G** basis set. Values for λ_{GPU} of 10^{-12}, 10^{-8}, 10^{-4}, and 1 are used. The columns labeled D.P and S.P correspond to results calculated using solely double or single precision arithmetic, respectively. The values in the square brackets denote the percentage of the unique nonzero ERIs that were evaluated using single precision arithmetic.

As expected, the number of decimal digits of precision in E_{HF} generally decreases as the problem size increases. An exception is for $(H_2O)_3$, where the results obtained using $\lambda_{GPU} = 1 \times 10^{-4}$ are less precise than the equivalent numbers for the larger $(H_2O)_4$ and $(H_2O)_5$ systems. This is due to the composition of the ERIs: $(H_2O)_3$ has more ERIs with magnitudes between 1×10^{-6} and 1×10^{-4} (38%) compared to $(H_2O)_4$ (26%) and $(H_2O)_5$ (20%). Thus, a disproportionately large number of integrals near the threshold are calculated in single precision for the $(H_2O)_3$ case. This shows that while partitioning between single and double precision calculation based solely on the Schwarz upper bound of the ERI is generally reliable, the specific distribution of integral values can also be a factor for some systems. Since the magnitude of a two electron integral is determined by the distance between each electron, we can expect denser molecular systems to have higher concentrations of large integrals.

The results also show that numerical errors increase as the cutoff is relaxed and more integrals are calculated in single precision. For the problems considered, using a λ_{GPU} value as large as 10^{-2} still provides sufficient precision to give E_{HF} values

Table 4. Decimal Digits of Precision of E_{HF} for TIP4P Water Clusters Using the 6-31G** Basis Set, with Different Cutoff Points (λ_{GPU}) Where ERIs Are Evaluated Using Single Precision Instead of Double Precision Floating Point.[a]

			λ_{GPU}					
$(H_2O)_n$	N	D.P	1×10^{-12}	1×10^{-8}	1×10^{-4}	1×10^{-2}	$1 \times 10^{+0}$	S.P
2	50	12.26	12.26 [3.61%]	12.26 [14.95%]	10.81 [42.89%]	8.52 [67.20%]	6.15 [99.90%]	6.02
3	75	11.92	11.92 [5.98%]	11.92 [21.86%]	10.36 [56.79%]	8.17 [81.86%]	6.07 [99.95%]	5.95
4	100	11.69	11.68 [13.74%]	11.68 [34.84%]	10.56 [68.93%]	8.29 [87.73%]	6.03 [99.97%]	5.91
5	125	11.48	11.46 [21.87%]	11.46 [47.38%]	10.39 [77.97%]	8.11 [91.31%]	6.01 [99.98%]	5.88
6	150	11.29	11.20 [17.14%]	11.20 [42.81%]	10.12 [78.74%]	7.93 [93.51%]	5.97 [99.99%]	5.84
7	175	11.15	11.05 [21.18%]	11.05 [49.75%]	10.14 [82.80%]	7.94 [94.86%]	5.94 [99.99%]	5.82
8	200	11.00	10.98 [23.92%]	10.90 [53.02%]	9.88 [85.60%]	7.70 [96.15%]	5.90 [99.99%]	5.78

[a] D.P stands for a fully double precision ERI calculation, and S.P stands for a fully single precision ERI calculation, with cutoff points of various magnitudes in between (the numbers in the square brackets indicate the percentage of single precision ERIs).

Table 5. Predicted Relative Numerical Error at $N = 10\,000$ for TIP4P Water Clusters for Different λ_{GPU}'s

	λ_{GPU}					
D.P	1×10^{-12}	1×10^{-8}	1×10^{-4}	1×10^{-2}	$1 \times 10^{+0}$	S.P
3.25×10^{-8}	3.02×10^{-8}	4.16×10^{-8}	2.68×10^{-7}	3.07×10^{-5}	5.40×10^{-5}	7.13×10^{-5}

that are chemically accurate, while using a value for λ_{GPU} of 1 is clearly inadequate. Alternatively, setting a λ_{GPU} threshold of 10^{-8} gives results that are almost as precise as exclusively using double precision.

In section 4.1, we used a second-degree polynomial regression model to predict numerical errors when $N = 10\,000$. If we apply the same approach to the data in Table 4, we find the projections given in Table 5. The columns show the predicted (relative) numerical errors for each respective λ_{GPU} value. If only single precision is used, the predicted error is on the order of 1×10^{-5}, which is not chemically accurate.

4.4. Effect of $F_m(T)$ Interpolation Table Size.

Memory management is one of the most important factors affecting the performance of CPUs, GPUs, and special purpose processors. This can mean managing a complex cache hierarchy, small amounts of user controlled local memory, or a combination of both. For example, the NVIDIA GTX280 has 16 KB of fast local shared memory available to each thread block,[28] the synergistic processing elements (SPE) on the STI CellBE each have 256 KB of local memory for SPE instructions and data,[4] and the SPARC VIIIfx CPU has a user controlled on-chip memory that can be cache, local memory, or a combination of both.[29]

How to use small fast on-chip memory is therefore an increasingly important question for program developers. In ERI evaluation, the $F_m(T)$ interpolation table is a good candidate for storage in fast memory. Use of interpolation gives rise to a significant reduction in floating point operations, but this is only beneficial if the data that comprise the interpolation table can be accessed quickly. So an issue is how small can the $F_m(T)$ interpolation table be made while still obtaining sufficiently accurate results. For the purpose of this study, we will assume an ERI evaluation scheme similar to that employed by Ufimtsev and Martinez, where each thread or thread block on a GPU calculates a single primitive integral or contracted integral, with the $F_m(T)$ interpolation table replicated for each thread block in the fast local memory.

The size of the interpolation table depends on the interpolation scheme used, the degree of the polynomial interpolation, the largest total angular momentum of any ERI, and the truncation error tolerance. Table 6 shows the amount of memory required to represent a third and sixth degree Chebyshev polynomial interpolation table for integrals of type $(ss|ss)$ through $(gg|gg)$ when using different $F_m(T)$ truncation error tolerances. Third degree polynomials require larger interpolation tables than sixth degree polynomials but require less operations to evaluate each subsequent $F_m(T)$ value. Thus, the choice of whether to use a third or sixth order polynomial involves a trade-off between floating point operation count and memory usage. For GPUs and the CellBT systems, memory size is probably the major bottleneck; thus, sixth degree interpolation appears to be the best choice. The results in Table 6 show that keeping truncation error around machine precision requires an interpolation table much larger than that which can be stored in the shared memory on a current GPU system. However, by relaxing the error tolerances, it is possible to fit the interpolation table in fast shared memory. Supposing that 8 KB is the maximum table size allowed, then using sixth-degree Chebyshev polynomials, $(ss|ss)$ type integrals

Table 6. Chebyshev Interpolation Table Sizes (KB) for Different Integral Types and Truncation Error Tolerances

		$e_{F_m(T)}$			
integral	degree	ε	1×10^{-12}	1×10^{-8}	1×10^{-4}
(ss|ss)	3	1134	138	13	1
	6	25	7	2	<1
(pp|pp)	3	1581	193	19	1
	6	35	10	2	<1
(dd|dd)	3	1925	235	23	2
	6	43	12	3	1
(ff|ff)	3	2269	277	27	2
	6	50	15	4	1
(gg|gg)	3	2544	310	31	3
	6	56	17	4	1

Table 7. Decimal Digits of Precision of E_{HF} for Different $F_m(T)$ Truncation Error Tolerances (e_{FMT})[a]

			$e_{F_m(T)}$			
system	basis	N	ε	1×10^{-12}	1×10^{-8}	1×10^{-4}
$(H_2O)_4$	6-31G**	100	11.69	11.68	9.05	4.97
alanine	6-31G**	125	11.41	11.37	8.40	4.22
serine	6-31G**	140	11.30	11.27	8.39	4.19
cytocin	6-31G**	145	11.24	11.21	8.31	4.14
$(H_2O)_8$	6-31G**	200	11.00	11.00	8.85	4.74
	table size (KB)		43	12	3	1

[a] The last row indicates the interpolation table sizes associated with each tolerance.

can be calculated with errors on the order of 10^{-12} and $(pp|pp)$, $(dd|dd)$, $(ff|ff)$, and $(gg|gg)$ integrals with errors on the order of 10^{-8}.

Using intervals, we can place rigorous bounds on the effect of varying the interpolation table sizes on the final value of E_{HF}. This is shown in Table 7 when using sixth-degree Chebyshev polynomial interpolation and a range of different $F_m(T)$ thresholds. The results show that the precision of E_{HF} is decreased when the error tolerance is relaxed and that chemical accuracy is maintained in all cases except when the $F_m(T)$ threshold is 10^{-4}. Interestingly, and in a similar manner to what was observed in section 4.3, when using error tolerances of 10^{-8} and 10^{-4}, the precision for $(H_2O)_8$ is greater than for any of the amino acid cases despite the fact the total number of basis functions (N) is greater. This is again because there is a higher proportion of large magnitude ERIs for the amino acids than for $(H_2O)_8$.

5. CONCLUSION AND FUTURE WORK

In this paper, interval arithmetic is used as an error analysis tool to explore the effect of numerical errors in calculating the Hartree–Fock total energy under a variety of scenarios. The results highlight the fact that under certain conditions chemical accuracy can still be achieved even when the Hartree–Fock energy is calculated using relatively unconventional parameters, for example, (i) by varying the integral screening threshold, (ii) by varying the numerical precision between single and double precision, and (iii) by varying the granularity of integral interpolation tables. This result gives reasonable grounds to suggest that some key modifications to existing evaluation schemes can be made while maintaining chemical accuracy. This is especially relevant when looking to circumvent the limitations of specialized processing hardware in regard to double precision performance and fast memory capacity.

The results also highlight the gradual accumulation of numerical errors with increasing basis set and system sizes, which will eventually reach a stage where chemical accuracy can no longer be guaranteed. Another significant finding is that the estimated growth in numerical errors can be accurately modeled by regressing the number of basis functions against the width of the interval bound E_{HF}. The best fit was achieved using second degree polynomials. Nonlinear scaling is expected as the number of ERIs scales nonlinearly with basis set size, but the number of ERIs scales as $O(N^4)$ not $O(N^2)$. This finding supports the generally accepted view that it is possible to use integral screening to significantly reduce the computational scaling of large HF computations. On a cautionary note, the results also show that there are other factors that affect accuracy, such as the density of the molecular system.

The main drawback of using interval arithmetic for numerical accuracy studies is that it provides a worst-case error analysis. The fact that interval results get rounded out to the nearest larger machine representable interval and the dependency problem means that these results are almost always overly pessimistic. That said, when the interval result is within the accuracy required, we do have guaranteed assurance of the numerical accuracy.

The current results also rely on a conventional Self-Consistent-Field (SCF) calculation to find the ground state molecular orbitals from a fixed atomic geometry. From the results of this calculation, the interval total energy is then calculated. This work does not consider how accurate a solution these coefficients are to the SCF problem, or indeed whether the solution represents the true ground state. Interval methods can be extended to address both issues, as will be discussed in another upcoming publication.

Finally, the results presented are confined to a limited set of small- and medium-sized systems. Another next step is to consider larger systems involving a larger variety of atoms, molecules, and basis set types, in particular, to test further the effectiveness of the polynomial regression model and to explore further other factors that affect numerical accuracy such as near linearly dependent basis sets.

AUTHOR INFORMATION

Corresponding Author
*E-mail: Alistair.Rendell@anu.edu.au.

ACKNOWLEDGMENT

This work was supported in part by Australian Research Council grant DP0558228.

REFERENCES

(1) IEEE Computer Society Standards Committee. Working group of the Microprocessor Standards Subcommittee, American National Standards Institute. *IEEE standard for binary floating-point arithmetic, ANSI/IEEE Std 754–1985*; IEEE Computer Society Press: Silver Spring, MD, 1985; p 18.
(2) Yasuda, K. *J. Comput. Chem.* **2008**, *29*, 334–342.
(3) Ramdas, T.; Egan, G.; Abramson, D.; Baldridge, K. *Theor. Chem. Acc.* **2008**, *120*, 133–153.
(4) Kahle, J. A.; Day, M. N.; Hofstee, H. P.; Johns, C. R.; Maeurer, T. R.; Shippy, D. *IBM J. Res. Dev.* **2005**, *49*, 589–604.
(5) Zein, A.; Mccreath, E.; Rendell, A.; Smola, A. Performance Evaluation of the NVIDIA GeForce 8800 GTX GPU for Machine Learning. http://dx.doi.org/10.1007/978-3-540-69384-0_52 (accessed April 2011).
(6) Caprani, O.; Madsen, K.; Nielsen, H. B. *Introduction to Interval Analysis*; Danmarks Tekniske Universitet: Copenhagen, Denmark, 2002.
(7) Takashima, H.; Kitamura, K.; Tanabe, K.; Nagashima, U. *J. Comput. Chem.* **1999**, *20*, 443–454.
(8) Almlöf, J. In *Modern Electronic Structure Theory Part II*; Yarkony, D. R., Ed.; World Scientific: Singapore, 1995; Vol. 1, Chapter: Direct Methods in Electronic Structure Theory, pp 110–151.
(9) Moore, R. E. *Automatic Error Analysis in Digital Computation*; Technical Report Space Div. Report LMSD84821, Lockheed Missiles and Space Co.: Sunnyvale, CA, 1959
(10) Hansen, E. R. *Global optimization using interval analysis*; Marcel Dekker, Inc.: New York, 1992; pp 1–21.

(11) Saltelli, A.; Ratto, M.; Tarantola, S.; Campolongo, F. *RESS* **2006**, *91*, 1109−1125, The Fourth International Conference on Sensitivity Analysis of Model Output (SAMO 2004)—SAMO 2004.
(12) Lindh, R. Integrals of Electron Repulsion. *Encyclopaedia of Computational Chemistry*; John Wiley and Sons: New York, 1998; p 1337.
(13) McMurchie, L. E.; Davidson, E. R. *J. Chem. Phys.* **1988**, *89*, 5777–5786.
(14) Obara, S.; Saika, A. *J. Chem. Phys.* **1986**, *84*, 3963.
(15) Rys, J.; Dupuis, M.; King, H. *J. Comput. Chem.* **1983**, *4*, 154–157.
(16) Head-Gordon, M.; Pople, J. A. *J. Comput. Phys.* **1978**, *26*, 218–231.
(17) Gill, P. M. W. *Adv. Quantum Chem.* **1994**, *25*, 141–205.
(18) Shavitt, I. *Meth. Comput. Phys.* **1963**, *2*, 1–44.
(19) Gill, P. M.; Johnson, B. G.; Pople, J. A. *Int. J. Quantum Chem.* **1991**, *40*, 745–752.
(20) Janes, P. P.; Rendell, A. P. *CSE* **2008**, *0*, 75–82.
(21) Sun Microsystems Inc. *The Sun Fire V1280 Server Architecture*. http://www.sun.com/servers/midrange/pdfs/V1280_wp_final.pdf (accessed Jan 9, 2011).
(22) Sun Microsystems Inc. *SunStudio 11: C++ User's Guide*. http://download.oracle.com/docs/cd/E19422-01/819-3690-10/819-3690-10.pdf (accessed Feb 6, 2011).
(23) Sun Microsystems Inc. *Sun Studio 11: C++ Interval Arithmetic Programming Reference*. http://download.oracle.com/docs/cd/E19422-01/819-3696-10/819-3696-10.pdf (accessed Feb 6, 2011).
(24) http://cs.anu.edu.au/people/Pete.Janes/applications/ (accessed Jan 11, 2011).
(25) Wales, D.; Doye, J.; Dullweber, A.; Hodges, M.; Naumkin, F.; Calvo, F.; Hernández-Rojas, J.; Middleton, T. *The Cambridge Cluster Database*. http://www-wales.ch.cam.ac.uk/CCD.html (accessed Jan 2, 2011).
(26) Higham, N. J. *Accuracy and Stability of Numerical Algorithms*, 2nd ed.; Society for Industrial and Applied Mathematics: Philadelphia, PA, 2002; pp 79−92.
(27) Ufimtsev, I. S.; Martinez, T. J. *J. Chem. Theory Comput.* **2008**, *4*, 222–231.
(28) *NVIDIA CUDA Programming Guide*, Version 3.0; NVIDIA: Santa Clara, CA, 2010.
(29) Maruyama, T.; Yoshida, T.; Kan, R.; Yamazaki, I.; Yamamura, S.; Takahashi, N.; Hondou, M.; Okano, H. *IEEE Micro* **2010**, *30*, 30–40.