JCTC Journal of Chemical Theory and Computation

Letter

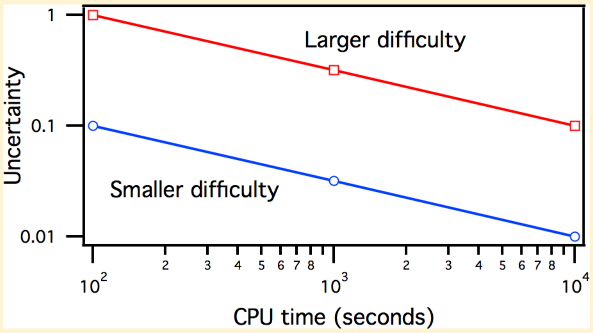
pubs.acs.org/JCTC

Quantifying Computational Effort Required for Stochastic Averages

Andrew J. Schultz and David A. Kofke*

Department of Chemical and Biological Engineering, University at Buffalo, The State University of New York, Buffalo, New York 14260-4200, United States

ABSTRACT: We propose a measure that quantifies the effort needed to compute a given stochastic average, considered in the context of molecular modeling. This “difficulty index” is defined in terms of CPU time, observed uncertainty, and a characteristic scale for the quantity being computed. This metric provides a focus for optimization and comparison of methods and, if applied broadly, could aid understanding of the impact of models, algorithms, implementations, and platforms on performance of molecular modeling techniques.



Data obtained by molecular simulation and related methods are reported with confidence limits that express the level of uncertainty associated with each measured value. The magnitude of the uncertainty depends on several factors, including the molecular system being modeled, the quantity being computed, and the algorithm used for its evaluation, as well as the amount of effort expended on the calculation. Some quantities are inherently more difficult than others to compute and require more computational effort to obtain a result with a useful level of precision. For example, the heat capacity is more difficult to compute than the energy; entropy and enthalpy differences are often more difficult than free-energy differences; the ninth virial coefficient is more difficult than the sixth.

This notion of the difficulty of a calculation is an important one, but it is poorly formalized—there is no accepted convention for reporting the effort required to compute a property. This situation is unfortunate, because such a measure would have considerable utility. First of all, a difficulty measure would facilitate comparison of different techniques for computing the same property, so that their relative effectiveness may be assessed and reported. This development would further allow for tracking of improvements in methods over time; but more broadly, establishment of such a measure would raise awareness of the extent that some quantities are inherently more difficult to compute than others, and in particular it would allow for a rank ordering of them. Such information can be useful to anyone considering whether to embark on computing a given quantity; alternatively, it can help to focus the efforts of method developers toward the formulation of new algorithms where they are most needed. Moreover, in quantifying the difficulty of various calculations, we can begin to isolate the effects of features that enter into them, such as details of the molecular model, the system size, the choice of programming language or simulation package, the algorithm and how it is tuned, the sampling scheme, and so on. One might envision the establishment of a comprehensive database of difficulty measures for various specific calculations, through which we could come to understand objectively and quantitatively the trade-offs in computational effort that accompany each modeling and algorithm choice.

■ DEFINITION

The difficulty of a property calculation is characterized not by the confidence limits achieved in the results, nor by the CPU time applied to its calculation, but by a combination of the two. Accordingly, we propose the following group D, which we call the *difficulty*, as a measure of computational effort required to compute a property y

$$D = t^{1/2}\sigma \qquad (1)$$

where t is the amount of CPU time required to obtain the result y with uncertainty σ. For specificity, we define the uncertainty here as the standard error of the mean, or more generally, the half-width of the 68% confidence interval. For a given computational algorithm and hardware platform, and for sufficiently long t, D is expected to be invariant with t. A more difficult calculation is one in which a larger t is required to achieve a given uncertainty, or alternatively, a given amount of computation time yields a result of larger σ. Hence, D is larger for more difficult calculations, and in this regard it directly relates to the inherent effort required to compute the quantity y.

Optimization of Multipart Calculations. The definition of D selected here has some interesting properties in the context of optimization of effort. In many situations, y is determined as a function of independent stochastic averages, x_n, each with uncertainty σ_n and difficulty D_n. The uncertainty in y depends on the uncertainties in the x_n according to the standard propagation-of-error formula, which has the form

Received: August 29, 2014
Published: November 7, 2014

$$\sigma^2 = \sum_n a_n^2 \sigma_n^2 \tag{2}$$

where $a_n \geq 0$ is a constant, typically given as $a_n \equiv |\partial y/\partial x_n|$. We consider now how to allocate a fixed amount of CPU time t optimally across the independent averages, defining t_n as the CPU allotted to calculation of x_n, such that

$$\sum_n t_n = t \tag{3}$$

We optimize on the square of the difficulty of y

$$\begin{aligned} D^2 &= t \sum_n a_n^2 \sigma_n^2 \\ &= t \sum_n a_n^2 {D_n}^2 / t_n \end{aligned} \tag{4}$$

via an unconstrained optimization of the Lagrange function Λ with multiplier λ:

$$\begin{aligned} \Lambda &\equiv D^2 - \lambda(t - \sum_n t_n) \\ &= t \sum_n a_n^2 {D_n}^2 / t_n - \lambda(t - \sum_n t_n) \end{aligned} \tag{5}$$

Setting to zero the derivative with respect to t_n, we have

$$t_n^{\mathrm{opt}} = a_n D_n (t/\lambda)^{1/2} \tag{6}$$

from which we may write, using (3)

$$\frac{t_n^{\mathrm{opt}}}{t} = \frac{a_n D_n}{\sum_j a_j D_j} \tag{7}$$

Thus, the optimal allocation of effort is to each of the x_n in proportion to their difficulty. If the effort is allocated this way, then substitution of eq 7 into 4 shows that

$$D = \sum_n a_n D_n \tag{8}$$

The difficulty of the overall calculation is a weighted sum of the component difficulties. We present an interesting application of this result in the section below.

This development shows that the difficulty forms the appropriate framework for optimizing allocation of processor time among the elements of a larger calculation. In this context it is more robust than variance-reduction approaches, because it accounts for the possibility that different elements may require different computational effort; an example of such an application may be found in ref 1.

■ RELATIVE DIFFICULTY

A significant shortcoming of D as defined in eq 1 — which we might call the absolute difficulty — is that its dimensions include the dimensions of y, meaning that its scale depends on the scale of y, as well as the units used to express σ. This makes it meaningless to compare values of D for different properties. As a remedy, we introduce the *relative difficulty*, defined as

$$\bar{D} \equiv D/Y \tag{9}$$

where $Y > 0$ is a characteristic value of y, against which σ can be gauged as being large or small. In most cases, it is appropriate to use $Y = |y|$, but there are times when another choice is suitable. For example, the magnitude of a virial coefficient B_n is judged in terms of its contribution to the equation of state, which is given by dividing by the molar volume v to an appropriate power; in this case it may be more useful to define $\bar{D}$ in terms of $Y = v^{n-1}$. As another example, some free-energy difference calculations may be of comparable difficulty but yield values of quite different magnitude, and there may be interest in comparing them without scaling by their size. Here, the temperature provides a suitable scaling parameter or energy scale to characterize the uncertainty, so one may select $Y = k_B T$. Such cases are likely to be the exception, however, so we propose a convention that $\bar{D}$ is defined in terms of $Y = |y|$, unless otherwise specified when reporting $\bar{D}$ values.

The relative difficulty has the dimension (time)$^{1/2}$, so any comparison of $\bar{D}$ values must be based on a common choice of time unit. We suggest that CPU-seconds be adopted as the standard. We discuss below an alternative proposition, that of forming a fully dimensionless difficulty by scaling the relative difficulty by a characteristic time.

■ DIFFICULTY INDEX

Values of the relative difficulty that are encountered in practice will range over many orders of magnitude, and it is convenient to work instead with an index that varies over a linear scale. Accordingly, we propose this measure

$$\mathcal{D} \equiv \log_{10}[\bar{D}/(\mathrm{sec})^{1/2}] \tag{10}$$

which we call the *difficulty index* or *diffidex*. Typical values of this index presently cover a range of about −2 to +2, although one may easily come up with examples falling outside these values. As a rule of thumb for reading the diffidex, we note that $2\mathcal{D}$ gives roughly the power of 10 of the CPU time in hours needed to achieve an uncertainty of about 1% (specifically, 1.7%), (e.g., $\mathcal{D} = 1$ indicates that about $10^{2\mathcal{D}} = 100$ CPU hours are required to get in the range of 1% precision).

A better way to develop a feeling for the diffidex is through examination of values for familiar calculations. To this end, we have compiled a set of values of the diffidex for various property calculations and present these in Table 1. This table is just a start, a snapshot showing the outcome for specific implementations applied to a few select properties. As the community uses and discusses this index more in relation to their own calculations, a better intuition of the meaning of the diffidex will be gained.

■ CONSIDERATIONS

Any practical attempt to catalog and interpret difficulty measures for different calculations will have to accommodate a wide range of complications and confounding factors. In this respect, it is an idealization to view the difficulty as a pure metric of the inherent effort required to compute a property; but like any idealization, it can guide thinking and provide a framework on which to gather data and build more nuanced understanding. Let us anticipate some of the complications here.

The difficulty index is affected by both the software and hardware that is used to implement the calculation and on the priorities, skill, and care of the programmer who coded the algorithm. The choice of programming language matters, as well as the design of the code, particularly whether it is special purpose or a general and extensible (perhaps object-oriented) simulation framework. Moreover, algorithms often are defined in terms of parameters that are used to tune their performance,

Table 1. Values of the Difficulty Index (Diffidex, $\mathcal{D}$) for Various Properties and Models, Ordered in Increasing Difficulty

property	model	$\mathcal{D}$	notes
internal energy	Lennard-Jones	−2.3	*j,m*
5th virial coefficient	hard sphere	−2.2	*a,b,m*
crystal free energy (HTTI)	Lennard-Jones	−1.1	*d,e,m*
Lo-T liquid sat'n density	TraPPE *n*-octane	−0.8	*g,h*
Lo-T liquid sat'n density	TIP4P water	−0.5	*g,h*
chem. pot'l, $\exp[-\mu/kT]$	Lennard-Jones	−0.4	*j,l,m*
Hi-T Liquid sat'n density	TraPPE *n*-octane	−0.2	*g,i*
crystal free energy (TI)	Lennard-Jones	−0.2	*d,f,m*
Hi-T liquid sat'n density	TIP4P water	0.0	*g,i*
heat capacity	Lennard-Jones	+0.7	*j,k,m*
Lo-T vapor pressure	TIP4P water	+0.8	*g,h*
Lo-T vapor pressure	TraPPE *n*-octane	+1.0	*g,h*
pressure	Lennard-Jones	+1.0	*j,m*
11th virial coefficient	hard sphere	+2.2	*a,c,m*

[a] Direct sampling of configurations. Full calculation described in ref 1. [b] Lookup table for evaluation of sum of graphs. [c] Wheatley's method[5] for evaluation of sum of graphs. [d] $\rho\sigma^3 = 1.2$, $kT/\varepsilon = 1.57$ (melting temperature). [e] Harmonically targeted thermodynamic integration[6] from harmonic reference at $T = 0$. [f] Standard thermodynamic integration from harmonic reference at $T = 0$. [g] Gibbs-ensemble Monte Carlo. Full calculation described in ref 2. [h] $T = 0.6T_c$. [i] $T = 0.9T_c$. [j] Canonical-ensemble (NVT) Monte Carlo simulation of 500 atoms at conditions of saturated liquid at $kT/\varepsilon = 1.0$. Potential is truncated at 2.5σ and cell neighbor lists are employed. [k] Energy-fluctuation average. [l] Chemical potential μ computed by Widom insertion. [m] Computed using Java-based etomica[7] molecular modeling framework.

so considerable detail is needed to specify them completely (see, e.g., ref 2). In some cases trade-offs are made between accuracy and precision. For example: decreasing the time step in a molecular dynamics simulation, increasing the convergence parameter of an Ewald sum, or increasing the number of steps in converging a polarizable force field all will increase the accuracy of the calculation, while also adding to the CPU time and the difficulty index for the calculation. Related to this, the effort required to equilibrate a system before beginning to accumulate ensemble averages may be significant. Such effort does not enter into the precision of the calculation and thus is not reflected in the difficulty, but appropriate effort spent on equilibration is important to obtaining an accurate result.

A molecular simulation often provides results for multiple properties at once and usually is not centered on the calculation of a single quantity. On the one hand, it is the basic need to sample configurations, rather than the property calculation *per se*, that occupies most of the CPU time required for a simulation. The difficulty of different properties from the same simulation will vary then due to their differing degrees of fluctuation over the course of the simulation. The effort added to compute one property at the same time as another in most cases will not affect the difficulty measure for either. On the other hand, some properties can be measured either directly (such as a phase coexistence point via the Gibbs-ensemble method[3]) or as a byproduct of a larger, more expensive calculation which can be used to evaluate not only the same property but also much more (such as a Wang−Landau[4] calculation of the full temperature-density free energy landscape). In such a case we simply have to treat these as separate properties and not try to compare the difficulty of the two methods as a means for evaluating the more specific property.

Finally, as with any quantity based on a stochastic average, the difficulty index itself is subject to uncertainty and can be meaningfully interpreted and compared only with some understanding of its confidence limits. The largest source of uncertainty in the diffidex will be the property uncertainty σ. One cannot expect reporting of the "uncertainty on the uncertainty", so it is not realistic for the difficulty to be reported with well-founded confidence limits. However, if the confidence limit on the property has any meaning, one should expect that the most significant digit of σ is correct to perhaps ± one digit. The effect on the diffidex depends on the digit, but this corresponds to an uncertainty of about ±0.15 in $\mathcal{D}$. Related to this, one should keep in mind that the difficulty is an invariant only asymptotically for large CPU times. Poor or highly correlated sampling can cause the difficulty measure to vary as the calculation proceeds. Hence it would be good practice to note its value periodically to ensure the asymptotic limit is reached. A good example of the behavior one should expect to observe is given in ref 2. Failure to reach this limit could be indicative of other problems with the simulation data.

■ PLATFORM DEPENDENCE

The effect of computing platform deserves special consideration. Steady improvements in compilers and processor capabilities by themselves lead over time to a continual reduction in the difficulty measures proposed here, and there may be interest in comparing improvements in algorithms across many years while controlling for these platform advances. A loose comparison can be achieved using Moore's law, which states that advances in processor capabilities lead to a doubling of computing speeds every 18 months. In terms of the diffidex, this means that for two calculations separated by n years, one should subtract $0.1n$ from the earlier value of $\mathcal{D}$ in order to compare it to a later value while controlling for hardware advances.

Nowadays, advances in platform capabilities result as much or more from parallellization than from faster processors. This presents a complication to the characterization of effort. On the one hand, the most useful measure of inherent difficulty should control for the number of processors and be defined in terms of the total CPU time expended across all processors (counting all of those on multicore CPUs as individual processors). On the other hand, we have graphics processing units (GPUs), which have emerged as an important platform for scientific computing. GPUs aggregate many weaker processors on a single chip to yield major performance improvements, in comparison to single-processor architectures. Adding up computation time across all of these processors is not really an appropriate way to characterize effort.

The crux of the problem is that difficulty and scalability are distinct features of a property calculation, but it is not easy to separate them to yield a pure difficulty measure. Different algorithms will scale differently as the amount of parallelism increases, depending on levels of communication, synchronization, and memory usage. So we cannot expect to define a difficulty that will necessarily produce the same value when used on these different platforms, by simply accounting for the number of processors involved.

One way to accommodate the dependence of difficulty on platform is simply to set aside the idea that the difficulty is a characteristic solely of the property being calculated and the algorithm used to do it and instead view the variation of the difficulty with platform as the object of any comparisons. The difficulty index then provides a versatile metric for gauging scalability, avoiding the need to ensure that the same number of simulation steps are performed when comparing run time on different platforms. Its use in this manner also facilitates comparisons of scalability of different properties or different simulation algorithms.

Alternatively, comparisons can be restricted to within a given category of computing platform, say, comparing the difficulty index for different calculations each performed on a GPU. In this view, we do not attempt to make sense of how the difficulty on a GPU relates to that on a single-core processor (for example); but even within a given category there will be differences resulting from platform details, particularly for calculations that are performed many years apart. An opportunity to compensate for this is by formulating a fully dimensionless difficulty measure, dividing the relative difficulty by an additional scaling factor that removes the time dimension from its definition. Such scaling might be done using quantities related to the clock speed of the processor (e.g., in GHz) or measures of floating point operations per second (FLOPS); but a more appropriate factor would be one based on the time required to complete an agreed-upon standard calculation, such as an established Netlib benchmark or, better yet, a simple molecular simulation.

That said, at this point we are not prepared to propose a specific benchmark. Gaining acceptance and use of a choice made here would hinder broader use of the difficulty measure. We think that ultimately the formulation of a good benchmark requires broad community input, which may be developed if and when the difficulty index gains currency. Nevertheless, we can provide a few comments to guide the formulation of a simulation-based benchmark.

- The basic benchmark should be a very simple molecular simulation, which need not do anything useful. The time-intensive part of most simulations is the calculation of the energy and forces, so the benchmark should repeatedly do this and not much more. The most elementary choice would be, say, a system of 1000 Lennard-Jones atoms starting from a simple-cubic arrangement of lattice constant 1.5 σ, all atoms with zero velocity and placed in a vacuum to avoid complications associated with periodic boundary conditions. They can then be propagated without neighbor listing in an NVE molecular dynamics simulation using a velocity Verlet algorithm (we would suggest for even greater simplicity not propagating the atoms at all, but some compilers might be smart enough to recognize that the energy and forces need not be recomputed each step, thus distorting the benchmark).
- The number of simulation steps should be large enough to provide a good timing sample but small enough to not be inconvenient to run−perhaps a minute or so. To avoid obselence as processor speeds increase over time, the benchmark can produce a result τ defined as CPU seconds per M simulation steps, where a convenient value of M is chosen, once and for all, at the time the benchmark is defined, say such that τ is of order 1.0 with today's computers. Then the user can run the simulation for whatever number of steps is convenient to get an accurate timing and simply scale the result to M steps to convert to the standard definition of τ. The dimensionless difficulty would then be defined as $\bar{D}/\tau^{1/2}$, and an index can be defined via its logarithm.
- The Standard Performance Evaluation Corporation (SPEC) maintains a set of proprietary benchmarks that are recognized as a standard for gauging the performance of computers. Many benchmarks have been established, focusing on different aspects of computer performance and mimicking different real-world applications, including a couple related to molecular simulation. The success of the SPEC framework suggests that formulating different benchmarks for different applications is not unreasonable. The molecular modeling community might decide that defining benchmarks for different broad categories of applications (e.g., proteins in solution, solid-state materials) is worthwhile. To minimize confusion, we would suggest however moving cautiously in adopting new benchmarks once one (or more) is established and used. The aim of the benchmark is to make the difficulty index more useful over time, and any changes in the standards will tend to defeat that purpose.
- The benchmark should be distributed as a fully functioning code that requires no special library calls (e.g., random number generators) and should not be written or modified by the user. Equivalent versions should be developed in commonly used programming languages, including Fortran, C/C++, and Java. The benchmark should be compiled using the same compiler and options used for the calculation being described by the dimensionless difficulty index.
- As for the prospect of invoking a dimensionless difficulty to counter the effects of differences across platform categories (e.g., GPUs vs CPUs), we will not completely rule out the possibility that a benchmark can be designed to be useful in this way − perhaps by averaging over algorithms using different parallelization strategies − but we do not see any certainty that it can be done. The assumption underlying the use of a benchmark to define a platform-independent dimensionless difficulty is that the benchmark characterizes how all simulation codes perform across platforms. This assumption is inexact even when applied across different single-core processors, and it becomes practically invalid when applied across platforms differing in number of processors, processors that can be exploited by a simulation algorithm in many different ways.

TWO APPLICATIONS

Thermodynamic Integration. We consider the difficulty of a free-energy difference ΔA computed via thermodynamic integration. For $\Delta A = \int f(\lambda) d\lambda$, a general Newton-Cotes or Gaussian quadrature formula is of the form

$$\Delta A \approx \sum_i w_i f(\lambda_i) \tag{11}$$

where w_i are weights that depend on the number and spacing of the points, as well as the quadrature method. With the $f(\lambda_i)$ evaluated by independent simulations, the uncertainty in ΔA is

$$\sigma_{\Delta A}^2 = \sum_i w_i^2 \sigma_{f(\lambda_i)}^2 \tag{12}$$

This has the form of eq 2, so the difficulty for ΔA is given as in eq 8, assuming optimal distribution of effort

$$D_{\Delta A} = \sum_i w_i D_{f(\lambda_i)} \tag{13}$$

$$\approx \int D(\lambda) d\lambda \tag{14}$$

where $D(\lambda)$ is the difficulty of the calculation of $f(\lambda)$. The key outcome is that the difficulty of ΔA is independent of the quadrature formula and in particular does not depend on the number of quadrature points used. Thus, from the standpoint of the precision of the integral, it does not matter whether integration is performed with a lot of sampling applied at a few quadrature points or a small amount of sampling applied at each of many points. The accuracy (as distinct from precision) of the result of course is affected by this choice, and other issues such as equilibration might enter to make the choice of some number of points more effective than others.

Crystal Free Energy. The free energy of crystals can be evaluated by thermodynamic integration from a low-temperature state that is well described by lattice dynamics techniques. Elsewhere[6] we describe methods for performing this calculation, introducing a new algorithm that is made more efficient by exploiting the approximately harmonic nature of the crystal (but without introducing any approximation in the calculated free energy). The effectiveness of the new method is illustrated in Figure 1, using the diffidex to compare it to the standard approach. The figure shows that the new method lowers the diffidex by almost one unit, corresponding to a two-order-of-magnitude increase in computation speed. We also note that the difficulty is nearly independent of system size. Although it costs more to get a fresh configuration in a large system, each measured sample has less variance; these two effects perfectly balance in a way that is clearly revealed by the diffidex. The increase in difficulty we might normally expect with system size is absent here because short-range interactions and the use of cell lists ensure that the cost of moving an atom is $O(1)$ and because we have not included equilibration time in our difficulty calculation.

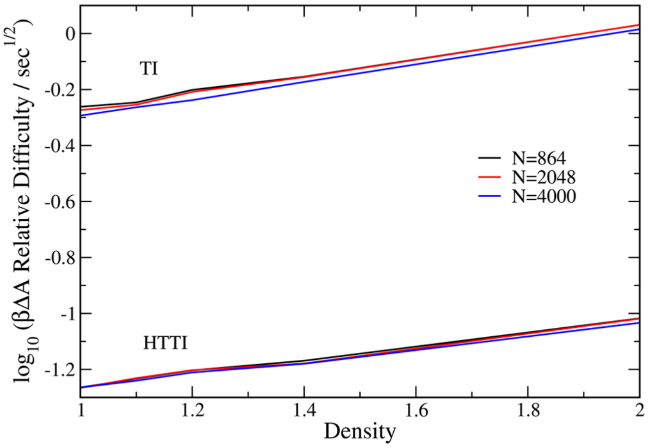


Figure 1. Diffidex for calculation of the anharmonic free energy of a Lennard-Jones crystal at its melting temperature, as a function of density. The upper group of lines is for standard thermodynamic integration, while the lower group is for a new harmonically targeted thermodynamic integration technique.[6] Values for different systems sizes are shown, as indicated by the legend, which lists the number of Lennard-Jones atoms in each simulated system.

CONCLUDING REMARKS

Our present understanding of the performance of molecular simulation methods is piecemeal and often anecdotal; it is certainly not comprehensive or in general well-sourced. The amount of CPU time required to compute a property is almost never included in papers reporting molecular simulation data. Yet such timing data are of extreme interest, as they implicitly or explicitly govern all decisions about how molecular simulation is applied. We can speculate on why this situation exists. One reason is that timing data are not particularly useful without being explicitly connected to the uncertainty of the results obtained. The difficulty index highlighted here is meant to address that issue. Timing data might also be devalued because of the broad range of complicating factors (some of which are reviewed above) that will confound its interpretation, even if presented in the form of a difficulty index. This could lead researchers to conclude that reporting of timing data is pointless. We would argue against this position, and suggest several ways that timing data in the form of a reported difficulty index would be of value.

First, it is not uncommon to encounter statements in publications or presentations simply remarking that a calculation was difficult to conduct or that a calculation was not performed because the error bars on the results were too large. A "difficult calculation" means different things to different people, and changes with time. Depending on how close to feasible a calculation may be, other researchers might consider devoting effort or resources toward it. A quantitative statement of the difficulty would make this assessment possible.

Second, even when dealing with routine calculations, it can be interesting if not useful to other researchers to know how much effort was required to generate particular results, regardless that such effort is specific to the author's software implementation, algorithmic details, and hardware platform. These examples can help us all to build intuition about the meaning of the difficulty index. Such information may also have the practical consequence of alerting others to potential improvements in performance that are realizable in their codes, which may be brought to light by comparing diffidex values for similar calculations.

Third, we suggest in particular that developers of algorithms and simulation techniques reference the difficulty when benchmarking new methods and comparing to established methods for the same property. In this way, we can settle on a universal, transferable metric that facilitates comparison of techniques for computing a given property, while controlling for key variables (e.g., hardware capabilities) using standards or benchmarks developed for this purpose.

Finally, given sufficient data, recorded in sufficient detail, the factors that confound interpretation and comparison of the difficulty index for different models and properties can be controlled, and in fact their effects on the computational effort can be studied and understood through reference to the difficulty index. This would be particularly effective if developed using a dimensionless difficulty measure, based on a well-conceived standard for scaling the time that accounts for platform differences (as discussed above). The main challenge then is in gathering and cataloguing the data. One can envision the establishment of an online, community-sourced database of diffidex values for a broad range of properties, models, algorithms, implementations, and platforms. Given enough input, this resource could be data-mined to generate insights regarding the performance impact of the elements that enter into molecular simulations. In this data-centric, interconnected age, such a development is certainly feasible technically. Still, we must concede that such a facility may not be realizable in practice, at least anytime soon — one still finds that even confidence limits, let alone timing information, are not reported conscientiously with simulation data. Small steps in this direction might be taken, however, if enough researchers find it worthwhile to report difficulty measures routinely, if only for the reasons summarized above.

AUTHOR INFORMATION

Corresponding Author

*E-mail: kofke@buffalo.edu.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work is supported by the U.S. National Science Foundation (NSF) (Grant No. CHE-1027963). We are grateful to Ilja Siepmann, Peter Cummings, Ed Maginn, and Vipin Chaudhary for comments that improved this manuscript.

REFERENCES

(1) Schultz, A. J.; Kofke, D. A. Fifth to Eleventh Virial Coefficients of Hard Spheres. *Phys. Rev. E* **2014**, *90*, 023301.

(2) Morales, A. D. C.; Economou, I. G.; Peters, C. J.; Siepmann, J. I. Influence of simulation protocols on the efficiency of Gibbs ensemble Monte Carlo simulations. *Mol. Simul.* **2013**, *39*, 1135−1142.

(3) Panagiotopoulos, A. Z. Direct determination of phase coexistence properties of fluids by Monte Carlo simulation in a new ensemble. *Mol. Phys.* **1987**, *61*, 813−826.

(4) Wang, F.; Landau, D. Efficient, Multiple-Range Random Walk Algorithm to Calculate the Density of States. *Phys. Rev. Lett.* **2001**, *86*, 2050−2053.

(5) Wheatley, R. J. Calculation of High-Order Virial Coefficients with Applications to Hard and Soft Spheres. *Phys. Rev. Lett.* **2013**, *110*, 200601.

(6) Moustafa, S. G.; Schultz, A. J.; Kofke, D. A. Harmonically Targeted Temperature Integration for Evaluation of Crystal Free Energies. *J. Chem. Phys.* **2014**, in preparation.

(7) Schultz, A. J.; Kofke, D. A. A Molecular Simulation Programming Interface. *J. Comput. Chem.* Submitted for publication.