Fifth to eleventh virial coefficients of hard spheres

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, USA

(Received 10 June 2014; published 4 August 2014)

Virial coefficients B_n of three-dimensional hard spheres are reported for $n = 5$ to 11, with precision exceeding that presently available in the literature. Calculations are performed using the recursive method due to Wheatley, and a binning approach is proposed to allow more flexibility in where computational effort is directed in the calculations. We highlight the *difficulty* as a general measure that quantifies performance of an algorithm that computes a stochastic average and show how it can be used as the basis for optimizing such calculations.

DOI: 10.1103/PhysRevE.90.023301 PACS number(s): 05.10.−a, 05.70.Ce, 51.30.+i

I. INTRODUCTION

The virial coefficients B_n appear in the expansion of the pressure P as a power series in the number density ρ, written here in terms of the compressibility factor Z [1–3],

$$Z \equiv \beta P/\rho = 1 + B_2\rho + B_3\rho^2 + \ldots, \tag{1}$$

where β is the reciprocal temperature $1/k_BT$. All of the virial coefficients can, in principle, be determined once a model of the intermolecular interactions is specified. Thus the virial equation of state (VEOS) is a rare example of a thermodynamic model that can be derived rigorously from molecular considerations. It can accommodate a broad range of complicating factors, such as mixtures [1,4], molecular flexibility [5–7], non-pair-wise interactions [1,8–12], inhomogeneous densities [13–18], nuclear quantum effects [19–23], and so on, without introducing any approximations. Furthermore, the general series-based formulation is not restricted to an equation for the pressure—it can be modified to treat almost any equilibrium property of interest [24–26]. In practice, the difficulty of computing B_n from a molecular model grows very quickly with n, such that is it feasible to complete such calculations only for n up to about 10, depending on the complexity of the model. Consequently, it has been difficult to study the VEOS to ascertain its convergence properties and how it behaves in relation to thermodynamic phase transitions.

In this context, hard-particle models, and hard spheres in particular, have attracted much attention because they exhibit nontrivial behaviors while possessing simplifying features that make calculation of their virial coefficients a bit more tractable. In general, the coefficients B_n are functions of temperature, but for hard potentials of a given shape and size they are simple constants. The coefficients B_2, B_3, and B_4 for hard spheres can be determined analytically, while higher-order coefficients require numerical computation. Beginning with B_5 in the 1950s [27], continuous progress has been made in computing three-dimensional hard-sphere virial coefficients at higher orders and with greater precision. Notable results reported in the past decade include B_5 and B_6 by Kolafa *et al.* [28], B_6 to B_9 by Labík *et al.* [29], B_9 and B_{10} by Clisby and McCoy [30], B_{11} and B_{12} by Wheatley [31], and B_5 to B_{12} by Zhang and Pettitt [32]. Many results have been reported for hard spheres in other dimensions as well; see in particular Ref. [32]. For hard potentials, the VEOS is sometimes expressed as a series in packing fraction $\eta = \rho v_0$, instead of density,

$$Z = 1 + \hat{B}_2\eta + \hat{B}_3\eta^2 + \ldots. \tag{2}$$

Here v_0 is the volume of the hard particle, and for spheres of diameter d, $v_0 = (\pi/6)d^3$. We use $\hat{B}_n$ to represent the coefficients for the equivalent packing-fraction series as follows: $\hat{B}_n = B_n/v_0^{n-1}$.

In the present work, we report new, more precise, values for the virial coefficients of three-dimensional hard spheres from B_5 to B_{11}. We make use of an important algorithm recently proposed by Wheatley [31], and we introduce some other methods that aid the calculation and which may find general use. Our focus is on detailing these calculations and analyzing their performance. We leave the study of the coefficients and the VEOS to a future work. We begin in the next section by proposing some simple measures that quantify the effort needed to calculate a stochastic average to some precision. These measures are useful in comparing the performance of algorithms, optimizing multipart calculations, and identifying where to direct effort to have the greatest effect on precision. The concepts we present there are general, and we would suggest that they be adopted for broader use in molecular modeling and other calculations having a stochastic element. In Sec. III, we detail the techniques we use for the virial-coefficient calculations, in Sec. IV we present our results and analyze the performance of the calculations, and we conclude in Sec. V.

II. CHARACTERIZATION OF EFFORT

Stochastic methods yield results that are reproducible to within a precision that depends on the amount of sampling performed. When assessing and optimizing such calculations, it is useful to work with a measure that is independent of the amount of sampling. Accordingly, we define a quantity we call the *difficulty* as follows:

$$D_y = t_y^{1/2}\sigma_y, \tag{3}$$

where t_y is the amount of CPU time required to obtain a result for a quantity y with uncertainty σ_y. For a given computational algorithm and hardware platform, and for sufficiently long t_y, D_y is expected to be invariant with t_y. The difficulty is useful

*kofke@buffalo.edu

as a focus for optimization of algorithms and for comparison of different algorithms for computing the same quantity y. A desirable algorithm is one that produces a result with the least difficulty, D_y. Rather than CPU time t_y, the difficulty could instead be defined in terms of the number of samples, or the number of floating-point operations, or a similar quantity, and this definition would have the advantage of being less dependent on the computational hardware. However, the time-based definition is more useful if optimization includes balancing the effort applied to different components of a larger calculation.

When comparing the difficulties for calculation of different quantities, it is useful to work with the relative difficulty, defined as $\bar{D}_y \equiv D_y/|y|$, and which has dimensions of (time)$^{1/2}$. This provides a convenient measure for specifying the effort required for any stochastic calculation. Thus, for example, the amount of CPU time required to obtain a result to 10% uncertainty would be equal to $100\bar{D}_y^2$.

Let us consider now the case in which the uncertainty in y has contributions from independent stochastic averages:

$$\sigma_y^2 = \sum\nolimits_n a_n^2 \sigma_n^2, \tag{4}$$

where σ_n is the uncertainty in the average labeled n, which has difficulty D_n, and $a_n \geqslant 0$ is a constant (made positive by absolute value, if necessary). An example is easily found in Eq. (1), from which the uncertainty in Z is given in terms of the uncertainties in the coefficients B_n, with $a_n = \rho^{n-1}$. Of interest is the question of how to allocate a fixed amount of CPU time t_y across the averages that contribute to y, defining t_n as the computation time devoted to average n. We optimize on the square of the difficulty of y, which is

$$\begin{aligned} D_y^2 &= t_y \sum_n a_n^2 \sigma_n^2 \\ &= \sum_n a_n^2 {D_n}^2 t_y/t_n. \end{aligned} \tag{5}$$

Minimization of D_y^2 with respect to the t_n, subject to the constraint $\sum_n t_n = t_y$, and assuming D_n is constant, shows that the optimal allocation has the time allotted in proportion to the difficulty,

$$\frac{t_n^{\text{opt}}}{t_y} = \frac{a_n D_n}{\sum_j a_j D_j}. \tag{6}$$

If the effort is allocated this way, then substitution of Eq. (6) into (5) shows that the difficulty for the calculation of y will be given as a weighted sum of the component difficulties,

$$D_y = \sum_n a_n D_n. \tag{7}$$

The situation can arise in which previous calculations have made available values of average n to some uncertainty σ_n, and a decision is needed regarding where to direct new effort toward the averages to have the largest effect in reducing the uncertainty in y. For this purpose, we define the *computational impact*, I_n, given in terms of the change in σ_y^2 with computational effort t_n applied to average n. A simple analysis yields

$$I_n \equiv \left(-\frac{\partial \sigma_y^2}{\partial t_n}\right)^{1/2} = a_n \frac{\sigma_n^2}{D_n}. \tag{8}$$

We write the impact in terms of σ_n rather than t_n: available data for average n are likely to have been generated by different researchers using different methods and computing platforms, so the time expended on them (if it is reported at all) is not relevant to the present analysis, which projects the impact based on D_n for the method and platform to be used for the new calculation. We further define the relative impact, dividing by the uncertainty in y, yielding again a quantity that does not depend on the units of y: $\bar{I}_n \equiv I_n/\sigma_y$ [having dimensions (time)$^{1/2}$].

New effort should be directed toward the average having the largest I_n. The impact goes inversely with D_n, because a larger difficulty means that computational effort has less impact on the coefficient precision and thus less impact on σ_y. As new effort is directed toward the highest-impact average, its precision improves (σ_n decreases), and its computational impact diminishes. Eventually, it goes below the I_n of another average, and computational effort can then be directed to that quantity. If taken to completion, this process would culminate in all I_n being equal. This, by the way, is the situation one arrives at if starting from scratch and applying to each coefficient the effort prescribed by Eq. (6).

We now turn to issues specific to the calculation of the virial coefficients.

III. METHODS

A. General considerations

For pairwise-additive interactions, the coefficient B_n is related to the intermolecular potential via the configurational integral [2],

$$B_n = \frac{1-n}{n!} \int f_B(\mathbf{r}^n) d\mathbf{r}_{12} d\mathbf{r}_{13} \ldots d\mathbf{r}_{1n}. \tag{9}$$

Here f_B is the sum of biconnected graphs,

$$f_B(\mathbf{r}^n) = \sum_G \left[\prod_{ij \in G} f_{ij} \right], \tag{10}$$

where f_{ij} is the Mayer function for the particles labeled i and j in the configuration $\mathbf{r}^n$ (we sometimes refer to this as an "f bond" between i and j). In general, the Mayer function is given in terms of the pair potential U by $f_{ij} = f(r_{ij}) = e^{-\beta U(r_{ij})} - 1$, where $r_{ij} = |\mathbf{r}_j - \mathbf{r}_i|$; for hard-particle potentials this simply has the value -1 or 0 for the cases in which the particles i and j at separation r_{ij} are or are not overlapping, respectively. The graphs G are formed from n vertices representing the particle coordinates, with f bonds joining some of the vertex pairs. The product in Eq. (10) is taken over all pairs having a bond in a given graph G, and the sum is over all doubly connected graphs of n vertices.

Except for low-order coefficients for simple pair potentials, evaluation of the integral in Eq. (9) must be accomplished numerically. Quadrature methods may be employed for small n, perhaps supplemented with Fourier transform techniques

to compute convolution integrals [33–36]. For higher-order coefficients, Monte Carlo (MC) methods should be used: configurations are generated at random, sampled in proportion to a distribution $\pi(\mathbf{r}^n)$; for each configuration, $f_B(\mathbf{r}^n)$ is evaluated, and the integral is estimated as

$$B_n = \frac{1-n}{n!} \times \Pi \times \left\langle \frac{f_B}{\pi} \right\rangle_\pi . \tag{11}$$

Here the angle brackets indicate an average of the quantity therein, taken over samples generated according to π, as described; also, $\Pi \equiv \int \pi(\mathbf{r}^n) d\mathbf{r}^{n-1}$ and is assumed to be known. There are two technical problems arising in the MC approach. First, one has to generate configurations according to π, and, second, one has to evaluate f_B and π for each such configuration. We will consider each of these in turn.

One way to generate configurations distributed as π is via a Markov process. This method is appealing because it enables π to be arbitrarily complex, as long as the reference integral Π can be determined by some means. We can then tailor π toward some form of importance sampling [37], aiming to compute an average having the highest precision for a given amount of sampling of configurations. The Mayer sampling MC method [38] is formulated this way [although it usually employs an expression somewhat more complicated [39] than Eq. (11)]. Often f_B for a system of hard spheres is used to define π, and the reference integral Π is given in terms of a hard-sphere virial coefficient or cluster integral. Obviously this choice is not available if the objective is to compute the hard-sphere virial coefficients, and alternatives have been examined that allow Mayer sampling to be applied here [32].

As an alternative to the Markov process, configurations may be generated directly from π, as long as π is sufficiently simple. For simple potentials such as hard spheres, and perhaps spherically symmetric models in general, it is possible to define a π that is simple enough for direct sampling, yet sufficiently similar to f_B to provide a useful level of importance sampling. Direct generation has the significant advantage of yielding uncorrelated configurations, so each one offers completely new information for the average. This contrasts with the Markov process, in which each new configuration is given as a perturbation on the one preceding it—thus many configurations must be generated for the correlation to die off and new information is obtained.

Another requirement of π, as used in Eq. (11), is that the set of configuration where it is non-negligible—its important configurations—must encompass those important to f_B [40–42]. Otherwise there will exist an unsampled, or rarely sampled, set of configuration for which the ratio f_B/π is very large. The result is a biased average.

The second problem, that of evaluating f_B for a given configuration, is naturally handled in an additive manner, meaning that one evaluates each of the terms of the sum in Eq. (10) and adds them together. This approach rapidly becomes unworkable with increasing order of the coefficient, as the number of graphs in the sum in Eq. (10) grows faster than exponentially with n (see Table I). The behavior places a practical limit of about $n = 8$ on the order in which a virial coefficient may be computed for most model potentials. Wheatley [31] recently proposed a remarkable algorithm for evaluation of f_B that is formulated in a completely different way. His method is based on subtracting non-bi-connected graphs from the sum of all graphs obtainable from n points and some or no f bonds. This sum-of-all-graphs starting point is easily computed as the single graph of n points with an $(f+1)$ bond between each point. In Wheatley's method, the disconnected and singly connected graphs are subtracted from this starting point using a recursive algorithm. The resulting method has memory and computation requirements that scale (only) exponentially with n. Wheatley demonstrated his algorithm with a calculation of B_{10} to B_{12} for hard spheres and B_5 to B_{10} for r^{-12} soft spheres.

Wheatley's algorithm treats the vertices as labeled and distinguishable, so distinct graphs that would be topologically equivalent if the vertices were unlabeled are all included in the sum. For example, the 10 graphs appearing in the sum for B_4 are shown in Fig. 1. Two sets of isomorphs—graphs that differ only in the labeling of the points—are indicated. For a given configuration of particles represented by the vertices, isomorphic graphs differ in value. However, these

TABLE I. Number of biconnected graphs on n vertices. The total number of graphs (unconnected, singly connected, and biconnected) on n vertices is $2^{n(n-1)/2}$, and the third column records the fraction of these that are biconnected.

n	Unlabeled[a]	Labeled	Fraction of all labeled graphs
2	1	1	0.5
3	1	1	0.125
4	3	10	0.156
5	10	238	0.232
6	56	11 368	0.347
7	468	1 014 888	0.484
8	7123	166 537 616	0.620
9	194 066	50 680 432 112	0.737
10	9 743 542	29 107 809 374 336	0.827
11	900 969 091	32 093 527 159 296 128	0.891
12	1.54×10^{11}	6.88×10^{19}	0.933
13	4.84×10^{13}	2.90×10^{23}	0.960
14	2.84×10^{16}	2.42×10^{27}	0.976
15	3.10×10^{19}	4.00×10^{31}	0.986

[a]OEIS Foundation Inc. (2011), *The On-Line Encyclopedia of Integer Sequences* [http://oeis.org/A002218].

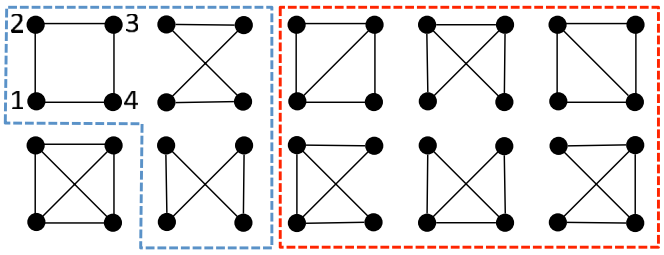

FIG. 1. (Color online) Biconnected graphs on four vertices. When labeled as indicated in the top leftmost graph, all these graphs are distinct. Graphs that are isomorphs are enclosed in the dashed-line boxes—these graphs can be made equivalent by appropriately rearranging the labels.

graphs become equal upon averaging over configurations. In some formulations of the cluster sum for the coefficient B_n, such graphs would be represented by a single isomorph, a symmetry number would be associated with the definition of its value, and the prefactor of the integral in Eq. (9) would instead be $(1-n)/n$. The additive approach to evaluate f_B for a configuration usually (but not always [10]) considers just the single isomorph, because to include the others would require explicitly evaluating them and adding them to the sum for f_B. Although each distinct labeling delivers to the average additional information for each configuration, these contributions are correlated, and often it is not worth the cost to include them. In Wheatley's method, all of the isomorphs are included "for free" in that no extra calculations are required to represent them all. Rather, the process of subtracting from the sum-of-all-graphs starting point is such that it causes no doubly connected diagrams to be eliminated, and, consequently, all isomorphs are included. This also simplifies matters by eliminating any need to determine the weight (symmetry number) associated with each graph.

Wheatley's algorithm applied to the evaluation of a coefficient B_n collects information that, with little additional effort, allows computation of averages for all coefficients B_k, $k < n$. Unfortunately, it is not advantageous to collect these averages because the configurations that are sampled for B_n, which are generated using some form of importance sampling (including the screening approach described in Sec. III C), are not the best ones to use to evaluate the lower-order coefficients. We also note that Wheatley's method can be extended to compute only the diagrams needed to correct the virial coefficients given by an incremental application of Percus-Yevick theory [36]. However, our attempts to calculate the B_n using this strategy did not show any advantage over the direct calculation of the coefficients using the techniques described here.

B. Generation of configurations

Direct generation of a configuration begins with random selection of a nominal bonding arrangement or template; the template can be represented by a graph G. We define a spherically symmetric pair weight function $w(r)$, which, with the graph G, defines a weight for a labeled configuration $\mathbf{r}^n$ as follows:

$$W(\mathbf{r}^n;G)=\prod_{ij\in G} w_{ij}, \tag{12}$$

where $w_{ij}\equiv w(|\mathbf{r}_j-\mathbf{r}_i|)$. Spheres are placed in such a manner that the configuration $\mathbf{r}^n$ is produced with probability density $\pi(\mathbf{r}^n;G)\equiv W(\mathbf{r}^n;G)/W(G)$, where $W(G)\equiv \int W(\mathbf{r}^n;G)d\mathbf{r}^{n-1}$. Once $\mathbf{r}^n$ is generated, evaluation of its total probability density $\pi(\mathbf{r}^n)$, needed for the average in Eq. (11), must consider all ways the configuration could have been obtained using this sampling algorithm. This requires summing $\pi(\mathbf{r}^n;G)$ over all the graphs G eligible for selection at the outset of the process, weighted by their selection probability. This procedure is detailed below.

It is important that $w(r)$ be nonzero anywhere that $f(r)$ is nonzero, and an appropriate $w(r)$ will also behave such that $f(r)/w(r)\to$ constant (perhaps zero) as $r\to\infty$. In the simplest case—suitable for finite-range potentials such as hard-sphere or square-well—$w(r)$ is just an overlap function, such that

$$w(r)=\begin{cases}1 & r\leqslant d\\ 0 & r>d\end{cases}. \tag{13}$$

With this choice, spheres that have a bond in G will be overlapping in configuration $\mathbf{r}^n$; spheres that do not have a bond in G may or may not be overlapping in $\mathbf{r}^n$. We used this pair weight function in the present work, but we will nevertheless describe our methods in this section for general $w(r)$.

A linear chain structure is among the simplest choices for the nominal bonding arrangement G. However, for $n\geqslant 6$, this template is not sufficient to produce all configurations that contribute to f_B (meaning that π will be zero for some configurations where f_B is not), so a MC method that uses it as the only template to generate configurations will yield a biased result. Other researchers [28–30,43–46] have treated this problem by generating configurations also according to one or more tree structures, selected to span biconnected graphs that are not spanned by chains. Wheatley [31] instead used just the chain template, but he did not restrict insertions to the sphere overlap region [as in Eq. (13)]—instead, he included a tail in $w(r)$ that introduces some probability that adjacent spheres in the chain are not overlapping. This approach has the advantage of being easily extended to higher-order coefficients, whereas the approach of selecting specific tree structures requires examination of the biconnected graphs to ensure that all of them are spanned by one of the nominal bonding structures.

In the present work, rather than attempting to identify the required spanning trees, we sample uniformly from *all* tree structures. In addition, we introduce an algorithm based on generation of ring structures, as well as the simple chain structure (even though it is an element of the set of trees). The choice of the template class is made at random at the outset of the configuration trial, such that on average the fraction of chain, tree, and ring templates selected are ϕ_C, ϕ_T, and ϕ_R, respectively (with $\phi_C+\phi_T+\phi_R=1$). Once the configuration is generated, its probability for purposes of computing the average in Eq. (11) is evaluated according to

$$\pi(\mathbf{r}^n)=\phi_C\pi_C(\mathbf{r}^n)+\phi_T\pi_T(\mathbf{r}^n)+\phi_R\pi_R(\mathbf{r}^n), \tag{14}$$

where $\pi_X(\mathbf{r}^n), X\in\{C,T,R\}$, is the probability density for generating configuration $\mathbf{r}^n$ given selection of template class X. For the ϕ to operate as intended, each π_X should be normalized; consequently, $\pi(\mathbf{r}^n)$ is normalized, and $\Pi=1$ in Eq. (11). Accordingly,

$$\pi_X(\mathbf{r}^n)=\frac{1}{\Omega_X}\sum_{G\in X}\pi(\mathbf{r}^n;G), \tag{15}$$

where Ω_X is the number of terms in the sum, which includes all distinct labelings of each graph in the template class. This form assumes that selection of each graph within template class X is equally likely.

In the following, we detail the generation of configurations and evaluation of $\pi_X(\mathbf{r}^n)$ for each of the template classes. Going forward, we define V as the set of all n sphere positions (or the set of vertices in a graph representing the sphere positions, depending on the context). It is useful at times to

make reference to the "graph for a configuration." By this we mean that, for definitions of $w(r)$ that have it equal to zero when r is greater than some cutoff value [such as Eq. (13)], we can associate a graph with a configuration $\mathbf{r}^n$ such that a bond is present between vertices i and j where $w(r_{ij}) \neq 0$, and no bond is placed between vertices for which $w(r_{ij}) = 0$.

1. Chains

Generation of a configuration in a chain arrangement follows a simple process of sequential insertion: sphere 1 defines the origin, and the kth sphere is inserted at a position randomly directed with respect to the location of sphere $k-1$, with separation selected according to $w(r)r^2$.

There is but a single graph in this template class, for which $W(G) = W_2^{n-1}$, where $W_2 \equiv \int_0^\infty w(r) 4\pi r^2 dr$; for $w(r)$ given by Eq. (13), $W_2 = \frac{4\pi}{3} d^3$. The sum in Eq. (15) is over the $\Omega_C = n!/2$ distinct labelings of the chain,

$$\pi_C(\mathbf{r}^n) = \frac{2}{n! W_2^{n-1}} \sum_{\tau \in \mathcal{P}(V)} \prod_{k=2}^{n} w_{\tau(k)\tau(k-1)} \equiv \frac{2}{n! W_2^{n-1}} W_C(\mathbf{r}^n), \tag{16}$$

where τ is a permutation, and the sum is over all permutations $\mathcal{P}$ of the set V; Eq. (16) defines $W_C(\mathbf{r}^n)$. An algorithm for computing this sum can be implemented as follows. Let v be a sphere in V, and let S be a subset of V containing v and sphere 1; $v = 1$ is allowed only for the initialization case in which 1 is the only sphere in S. Define

$$W_{C,v}(S) = \sum_{\substack{\tau \in \mathcal{P}(S): \\ \tau(1)=1, \\ \tau(|S|)=v}} \prod_{k=2}^{|S|} w_{\tau(k)\tau(k-1)}, \tag{17}$$

i.e., $W_{C,v}(S)$ is a sum over all chains in S beginning from sphere 1 and ending at sphere v. Then, starting with $S = \{1\}$ [for which $W_{C,1}(S) \equiv 1$], for each sphere u not in S, we compute

$$W_{C,u}(S \cup u) = \sum_{v \in S} w_{uv} W_{C,v}(S).$$

This formula is applied recursively until we have determined $W_{C,v}(S)$ for all subsets $S \subseteq V$, and all $v \in S$, $v \neq 1$. Finally, we obtain the sum over all distinct labelings of the chain, via

$$W_C(\mathbf{r}^n) = \sum_{S \subseteq V} \sum_{\substack{v \in S \\ u \in S^*}} W_{C,v}(S) W_{C,u}(S^* \cup 1), \tag{18}$$

where S^* is the complement of S with respect to the set V. The sum is taken over all subsets S which contain both sphere 1 and (to prevent double counting of subsets) another specific sphere (say, sphere 2). Wheatley [31] employed a chain-growth algorithm to generate configurations, using a recursive algorithm very similar to this one to compute the weights.

This algorithm for computing $W_C(\mathbf{r}^n)$ is a dynamic programming approach that has been applied to similar problems [47,48]. It can be used, for example, to compute the number of Hamiltonian paths [49,50] for a graph, which for $w(r)$ given by Eq. (13), is equal to $W_C(\mathbf{r}^n)$.

2. Trees

We first select a tree structure at random, chosen uniformly from among all possible n^{n-2} labeled trees that can be formed from n vertices. This is done by generating a Prüfer code [51], which is a sequence of $(n-2)$ integers each selected with uniform probability from 1 to n. This code uniquely specifies a labeled tree, and a simple iterative algorithm [49,50] can be applied to generate the tree from the code. Then as the spheres are inserted in sequence, the tree specifies which sphere to reference when inserting each new sphere according to $w(r)$.

All tree graphs have the same normalization, $W(G) = W_2^{n-1}$. Analogously to Eq. (16), the probability density $\pi_T(\mathbf{r}^n)$ to generate $\mathbf{r}^n$ using the tree algorithm is given in terms of the sum $W_T(\mathbf{r}^n)$ over all $\Omega_T = n^{n-2}$ distinct labeled trees,

$$\pi_T(\mathbf{r}^n) = \frac{1}{n^{n-2} W_2^{n-1}} W_T(\mathbf{r}^n). \tag{19}$$

The sum W_T can be computed quickly using an extension of Kirchoff's matrix-tree theorem [50,52]: One forms a matrix M such that element $m_{ij} = -w_{ij}$ for $i \neq j$, and $m_{ii} = \sum_{j \neq i} w_{ij}$. Then $W_T(\mathbf{r}^n)$ is equal to any cofactor of M. For the $w(r)$ given by Eq. (13), $W_T(\mathbf{r}^n)$ is the number of spanning trees of the graph of the configuration $\mathbf{r}^n$. In the calculations we report here, we computed W_T using a recursive algorithm somewhat like that used to compute W_C; we do not detail it here because we expect that the matrix-based approach is more efficient.

3. Rings

We label the spheres $1, 2, \ldots, n, n+1$, with $n+1$ another label for sphere 1. With sphere 1 defining the origin, we repeatedly insert each sphere k in a position that depends on the locations of the two previously inserted spheres closest to it in the ordered sequence, having labels i and j, respectively. In particular, the position $\mathbf{r}_k$ is selected with weight (unnormalized probability) $W_k(\mathbf{r}_k; i, j)$ such that

$$W_k(\mathbf{r}_k; i, j) \propto w_{1+|k-j|}(|\mathbf{r}_k - \mathbf{r}_j|) w_{1+|k-i|}(|\mathbf{r}_k - \mathbf{r}_i|), \tag{20}$$

where $w_m(r)$ is the weight for two spheres at specific positions separated by r when at the ends of a chain of m spheres having adjacent spheres distributed according to $w(r)$ [thus, $w_2(r) \equiv w(r)$]. It is obtained as the integral of $W(\mathbf{r}^m; G)$ over all positions of the spheres in the interior of the chain graph G. This can be given analytically via Fourier transforms as follows:

$$\begin{aligned} w_m(r) &= \mathcal{F}^{-1}[(\mathcal{F}[w(r)])^{m-1}] \\ &= \mathcal{F}^{-1}\left(\left\{ \frac{3[\sin(kd) - kd\cos(kd)]}{(kd)^3} \right\}^{m-1} \right), \end{aligned} \tag{21}$$

where the latter equality is for $w(r)$ given by Eq. (13). Each inverse transform yields a high-order polynomial in r. These formulas are too complicated to reproduce here, but they can be derived using a suitable symbolic mathematics package. We note that care must be taken in coding these functions to guard against loss of precision when evaluating the polynomial sum.

To select a point with weight $W_k(\mathbf{r}_k; i, j)$, we approximate $w_m(r)$ as Gaussian in three dimensions, yielding via Eq. (20) an approximate $W_k(\mathbf{r}_k; i, j)$ as a product of the approximate $w_m(r)$. This approximate $W_k(\mathbf{r}_k; i, j)$ is also Gaussian and thus can be sampled using standard methods. We then apply a rejection algorithm to produce a sample with the desired weight. The chain-growth process ends with one or more insertions of spheres that each must overlap two previously

TABLE II. Normalization of ring graphs of n spheres for $w(r)$ as given by Eq. (13) (for example, $W(R_3) = 8.22467d^6$).

n	$W(R_n)/(\pi d^3)^{n-1}$
3	$\frac{5}{6} = 0.833\ 333\ldots$
4	$\frac{2176}{2835} = 0.767\ 549\ldots$
5	$\frac{40949}{54432} = 0.752\ 296\ldots$
6	$\frac{23648512}{30405375} = 0.777\ 774\ldots$
7	$\frac{25040879363}{30023136000} = 0.834\ 053\ldots$
8	$\frac{35836384927744}{38979295480125} = 0.919\ 370\ldots$
9	$\frac{6060781370812815989}{5854170457175040000} = 1.035\ 29\ldots$
10	$\frac{430588656377655296}{363092137397364375} = 1.185\ 89\ldots$
11	$\frac{279634343277877995897619523}{203006266387416011520000000} = 1.377\ 47\ldots$
12	$\frac{2181664367287144834983919616}{1347828286825972065254765625} = 1.618\ 65\ldots$

inserted spheres adjacent to it in the sequence. This requires insertion into a lens shaped region formed by the overlap of two spheres of diameter $2d$ located at the positions of the adjacent-index spheres. This can be accomplished via repeated insertion into the rectangular region that circumscribes the lens, rejecting until a point in the lens is generated. Given the high frequency that this lens insertion is performed, we developed a slightly more efficient approach (not detailed here) that inserts in the lens region more directly.

The normalization for the ring graph is $W(R_n) \equiv \int_0^\infty w_n(r)w(r)4\pi r^2 dr$; this is tabulated as a function of n for the $w(r)$ of Eq. (13) in Table II. Defining $W_R(\mathbf{r}^n)$ as the sum over all $\Omega_R = (n-1)!/2$ distinct labelings of the ring, the probability density to generate $\mathbf{r}^n$ using the ring algorithm is as follows:

$$\pi_R(\mathbf{r}^n) = \frac{2}{(n-1)!W(R_n)} W_R(\mathbf{r}^n). \tag{22}$$

To compute $W_R(\mathbf{r}^n)$, we evaluate $W_{C,v}(S)$ as described above, but instead of Eq. (18), we have

$$W_R(\mathbf{r}^n) = \frac{1}{2}\sum_{v\neq 1} W_{C,v}(V)w_{1v}, \tag{23}$$

where the $1/2$ prefactor compensates for double counting. For the $w(r)$ given by Eq. (13), $W_R(\mathbf{r}^n)$ is the number of Hamiltonian rings of the graph of the configuration $\mathbf{r}^n$.

C. Calculation of averages

1. Screening

Once a configuration is generated, Wheatley's algorithm [31] provides a means to compute $f_B(\mathbf{r}^n)$. For the vast majority (more than 90%) of the configurations generated using the methods outlined above, f_B evaluates to zero. Consequently, the efficiency of the calculation can be greatly enhanced by application of methods to identify many of these $f_B = 0$ configurations without explicit calculation of f_B. We check first that each point has at least two bonds (which is assured only when using the ring template), and then we search for clique separators. A clique is a set of points that are all mutually connected; if removal of all points in a clique causes the graph to separate into two or more pieces, then it is a clique separator. Graphs having such features will have $f_B = 0$ [32]. Wheatley [31] employed this as a screening method as well, stopping the search at cliques of no more than five vertices. We found it worthwhile to not truncate the search, so we examine the graph of each configuration for clique separators of any size (it appears that Zhang and Pettitt [32] also screened considering cliques of all sizes). If one is found, we add zero to the simulation average; otherwise we evaluate f_B using Wheatley's algorithm and add $f_B(\mathbf{r}^n)/\pi(\mathbf{r}^n)$ to the sum for the simulation average.

To search for clique separators, we consider each subset of points and determine whether the subset is a clique. As in Wheatley's algorithm, this can be done recursively by recognizing that a subset is a clique if and only if all of its subsets are cliques; in fact, only three subsets need to be considered so long as they together include all pairs in the set. For each clique found, we check to see whether it is possible to walk along the bonds from one remaining (nonclique) point to every other remaining point without visiting a point in the clique. This operation theoretically requires $n-m$ steps (for a clique of size m), but this can be accelerated using bitmasks to track the walk such that all paths are considered simultaneously. While it would be slower than Tarjan's algorithm [32,53] for large n, the algorithm we use is simple, faster than other parts of the overall algorithm, and yields the full list of cliques, which we use for binning as described below.

2. Binning

The sample average appearing in Eq. (11) can be expressed as a weighted sum of averages as follows:

$$\langle f_B/\pi\rangle_\pi = \frac{1}{N}\sum_{\boldsymbol{\alpha}} n_{\boldsymbol{\alpha}}\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}. \tag{24}$$

Here $\boldsymbol{\alpha}(\mathbf{r}^n)$ is a parameter vector (one or more parameters) that characterizes the graph of a configuration: Each graph maps (independently of the graph labeling) onto a single, well-defined $\boldsymbol{\alpha}$ vector and, in general, each $\boldsymbol{\alpha}$ represents more than one graph (e.g., $\boldsymbol{\alpha}$ could be a single parameter equal to the number of bonds in the graph). Then $n_{\boldsymbol{\alpha}}$ is a random variable for the number of configurations having parameter $\boldsymbol{\alpha}$ from N total configurations generated according to π. The expected value of $n_{\boldsymbol{\alpha}}$ is $Np_{\boldsymbol{\alpha}}$, where we introduce $p_{\boldsymbol{\alpha}}$ as the probability that a configuration sampled from π will have parameter $\boldsymbol{\alpha}$. Also in Eq. (24), $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$ is the π-weighted average of f_B/π over the population of graphs having parameter $\boldsymbol{\alpha}$.

The key idea of the binning strategy is recognition that not all samples generated in bin $\boldsymbol{\alpha}$ need be used toward estimation of $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$. Given the expense of computing f_B/π, it may not be worthwhile to do so if the average for bin $\boldsymbol{\alpha}$ is already well characterized. Thus, upon generating a configuration according to π, we determine its $\boldsymbol{\alpha}$, increment the counter for $n_{\boldsymbol{\alpha}}$, and then decide with probability $P_{\boldsymbol{\alpha}}$ whether to compute f_B/π. If we do compute it, we accumulate the result in an average for $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$; if not, we do nothing to the average and continue on to the next sample. In either case, the contribution to $n_{\boldsymbol{\alpha}}$ is recorded. Hence, each sample is used toward estimation of $p_{\boldsymbol{\alpha}}$, even if it is not used toward $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$.

We set values for the $P_{\boldsymbol{\alpha}}$ to minimize the uncertainty σ_π in $\langle f_B/\pi\rangle_\pi$ for a fixed total computation time. This uncertainty will have contributions from the uncertainty $\sigma_{\boldsymbol{\alpha}}$ in each $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$ average and from fluctuations in the sampling of each of the $n_{\boldsymbol{\alpha}}$; the latter are correlated and also cannot be optimized with respect to each other, so we collect their contribution to σ_π into a single term, labeled σ_s, thus,

$$\sigma_\pi^2 = \sigma_s^2 + \sum_{\boldsymbol{\alpha}} p_{\boldsymbol{\alpha}}^2 \sigma_{\boldsymbol{\alpha}}^2. \tag{25}$$

Equation (25) is in the form of Eq. (4), so, according to Eq. (6), the optimal ratio of computing time $t_{\boldsymbol{\alpha}}$ spent on averaging $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$ to time t_s spent generating samples from π to estimate $p_{\boldsymbol{\alpha}}$ is given in terms of their respective difficulties by

$$\frac{t_{\boldsymbol{\alpha}}}{t_s} = \frac{p_{\boldsymbol{\alpha}} D_{\boldsymbol{\alpha}}}{D_s}. \tag{26}$$

The $P_{\boldsymbol{\alpha}}$ enter Eq. (26) through the $t_{\boldsymbol{\alpha}}$. We define τ_c as the CPU time required to compute f_B/π for a given configuration (and perform any other computations related to contributing to $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$); we assume τ_c is independent of $\boldsymbol{\alpha}$. Then $t_{\boldsymbol{\alpha}} = P_{\boldsymbol{\alpha}} n_{\boldsymbol{\alpha}} \tau_c$. Also, the difficulty of the average for bin $\boldsymbol{\alpha}$ is $D_{\boldsymbol{\alpha}} = \tau_c^{1/2} \Sigma_{\boldsymbol{\alpha}}$, where $\Sigma_{\boldsymbol{\alpha}}^2$ is the π-weighted variance of the f_B/π values for configurations in bin $\boldsymbol{\alpha}$.

The uncertainty due to sampling is defined as follows:

$$\sigma_s^2 = \frac{1}{N^2} \sum_{\boldsymbol{\alpha}} \sum_{\boldsymbol{\beta}} \mathrm{Cov}(n_{\boldsymbol{\alpha}}, n_{\boldsymbol{\beta}}) \langle f_B/\pi\rangle_{\boldsymbol{\alpha}} \langle f_B/\pi\rangle_{\boldsymbol{\beta}}. \tag{27}$$

The $n_{\boldsymbol{\alpha}}$ are sampled from a multinomial distribution with parameters N and $p_{\boldsymbol{\alpha}}$, and from this one can show that Eq. (27) can be expressed as follows:

$$\sigma_s^2 = \frac{1}{N} \Sigma_s^2, \tag{28}$$

where Σ_s^2 is the variance of the $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$ averages across the bins. This can be estimated by

$$\Sigma_s^2 = \frac{1}{N} \sum_{\boldsymbol{\alpha}} n_{\boldsymbol{\alpha}} \langle f_B/\pi\rangle_{\boldsymbol{\alpha}}^2 - \left(\frac{1}{N} \sum_{\boldsymbol{\alpha}} n_{\boldsymbol{\alpha}} \langle f_B/\pi\rangle_{\boldsymbol{\alpha}} \right)^2. \tag{29}$$

We define τ_s as the CPU time required to generate and screen one new configuration and determine its $\boldsymbol{\alpha}$. Then, with Eqs. (3) and (28), the difficulty associated with sampling is $D_s = \tau_s^{1/2} \Sigma_s$.

Putting together the pieces developed here, each side of Eq. (26) is transformed,

$$\frac{P_{\boldsymbol{\alpha}} n_{\boldsymbol{\alpha}} \tau_c}{N \tau_s} = \frac{p_{\boldsymbol{\alpha}} \tau_c^{1/2} \Sigma_{\boldsymbol{\alpha}}}{\tau_s^{1/2} \Sigma_s}, \tag{30}$$

which, substituting $n_{\boldsymbol{\alpha}}/N$ back for $p_{\boldsymbol{\alpha}}$, yields the prescription for $P_{\boldsymbol{\alpha}}$,

$$P_{\boldsymbol{\alpha}} = \varphi^{-\frac{1}{2}} \frac{\Sigma_{\boldsymbol{\alpha}}}{\Sigma_s}, \tag{31}$$

where $\varphi \equiv \tau_c/\tau_s$ is the ratio of the cost of computing f_B/π to the cost of sampling a new configuration. Thus, the probability to compute f_B/π is in proportion to the standard deviation in bin $\boldsymbol{\alpha}$, relative to the standard deviation of the average across the bins. It is possible that the probability computed this way is greater than 1, indicating a need to compute f_B/π more while sampling fewer configurations. Of course, because sampling is a prerequisite for each f_B/π calculation, this optimum cannot be met; instead we just cap $P_{\boldsymbol{\alpha}}$ at 1.

There are two practical issues that complicate application of this formula for $P_{\boldsymbol{\alpha}}$. The first is that at the beginning of the simulation, the values of $\Sigma_{\boldsymbol{\alpha}}$ cannot be estimated accurately because they have insufficient samples (or even no samples at all). A similar problem can occur for a bin with many samples that happen to have identical values. To handle both of these cases, we use the following formula for the variance in bin $\boldsymbol{\alpha}$ appearing in Eq. (31):

$$\Sigma_{\boldsymbol{\alpha}}^2 = \langle \Sigma_{\boldsymbol{\alpha}}^2 \rangle + \frac{\langle (f_B/\pi)^2 \rangle}{n_{\boldsymbol{\alpha}c}}, \tag{32}$$

where $\langle \Sigma_{\boldsymbol{\alpha}}^2 \rangle$ is the running average of the variance of bin $\boldsymbol{\alpha}$, based on $n_{\boldsymbol{\alpha}c}$ samples. This modification forces the variance for each bin to be a bit larger, as though an additional sample was included that is typical for all bins but atypical for the given bin. As the simulation progresses, $n_{\boldsymbol{\alpha}c}$ grows and this additional term becomes less important, but its presence assures that every bin has a finite probability, even if its $\langle \Sigma_{\boldsymbol{\alpha}}^2 \rangle$ is 0. To ensure that at least two values are computed for each bin, we force $P_{\boldsymbol{\alpha}} = 1$ whenever there are fewer than two samples for that bin.

This binning approach admits considerable flexibility for optimizing the calculation through a judicious definition of $\boldsymbol{\alpha}$. Unlike look-up tables, which are similar in spirit and have been used by others for hard-sphere virial calculations [29,30,43,44,46,54], the approach proposed here does not require that the bin parameter specify a graph uniquely (a canonical label). A good choice will allow quick determination of $\boldsymbol{\alpha}$ for a configuration and also will cause $\Sigma_{\boldsymbol{\alpha}}$ to be small, meaning there is not much variation in f_B/π within each bin. The conventional look-up table represents a limiting case in which $\langle f_B/\pi\rangle_{\boldsymbol{\alpha}}$ has zero variance, except here, as also suggested by Zhang and Pettitt [32], the table is populated dynamically instead of being precomputed. Generation of a canonical label for a graph is not simple, and McKay [55,56] has developed and implemented an efficient algorithm to do this; Zhang and Pettitt found the speed of McKay's algorithm still to be limiting their calculations, and they devised a less-thorough alternative. The larger problem is that for n greater than about 9 or 10, there are too many graphs to permit storing f_B/π for every $\boldsymbol{\alpha}$ (see Table I). With Wheatley's algorithm, it becomes more effective to compute f_B/π on the fly, and one is then back to computing the average in the direct fashion indicated by Eq. (11). As a stopgap remedy, the table can be populated dynamically, as suggested by Zhang and Pettitt [32], which avoids computing and storing values for graphs that are never encountered. At the other extreme, $\boldsymbol{\alpha}$ is the same for all graphs and does nothing to distinguish them. Then the sum in Eq. (24) consists of a single term and thus reduces again to Eq. (11).

We did not attempt a thorough investigation of how to define $\boldsymbol{\alpha}$ to maximize the efficiency of the calculation. For these calculations, we used the following parameters for $\boldsymbol{\alpha}$:

(i) the number of f bonds,
(ii) the number of cliques based on f bonds,
(iii) the number of cliques based on e bonds,

(iv) the number of f bond clique pairs that share some (but not all) points, and
(v) the number of e-bond clique pairs that share some (but not all) points.

D. Computational details

We performed calculation of virial coefficients B_n of the hard-sphere model for n from 5 to 11. All calculations were performed using codes developed in Java. The number of samples and total amount of CPU applied to calculate each coefficient is reported in Table III. The calculations required CPU times on the order of years and could be accomplished only through the application of many (thousands) of processors performing independent calculations that are subsequently collected to yield results with the precision given below. The number of samples represents the total number of configurations generated, regardless of whether the configuration led to a calculation of f_B/π, and the CPU time is totaled over all independent processors applied to the calculation. Calculations were performed on a heterogeneous cluster composed of mostly Intel Xeon compute nodes with 8 to 32 cores and speeds from 2.13 to 2.66 GHz.

Each coefficient was computed by averaging together the results from numerous runs. Computation probabilities ($P_{\boldsymbol{\alpha}}$) were determined during the first run. These were used as input for further runs, which were performed in parallel and did not attempt to update $P_{\boldsymbol{\alpha}}$. When this set of runs had finished, their results were combined and new $P_{\boldsymbol{\alpha}}$ were computed. This process continued until the desired number of simulations had finished.

We can increase the efficiency of the calculation by increasing the number of parameters used to determine $\boldsymbol{\alpha}$, so every bin has a distinct value with small $\Sigma_{\boldsymbol{\alpha}}$. However, when performing calculations for high-order coefficients the memory needed to store data for each bin can exceed the memory available on a compute node, especially when it is multi-CPU and multicore, where we might run simultaneous calculations on all cores. To remedy this problem, we modified our Java program to run independent calculations in different threads which shared a common storage for the bin data. This multithreaded implementation is efficient because each thread can perform most of its work without synchronization—the only activities requiring any synchronization are incrementing the number of unscreened configurations, adding a measured f_B/π to a bin's sums and recomputing $P_{\boldsymbol{\alpha}}$ for each bin.

TABLE III. Sampling measures for calculation of the virial coefficients B_n.

n	Number of samples $\times 10^{-12}$	CPU (years)
5	48	4.3
6	32	3.8
7	38	5.3
8	41	6.6
9	47	8.8
10	87	21.0
11	115	30.7

To generate random numbers during the calculations, we use the MT19937 implementation of the Mersenne Twister [57] pseudorandom number generator (PRNG). MT19937 is seeded at the beginning of each simulation with four 4-byte integers from the Linux kernel's /dev/urandom PRNG. When running with multiple threads, each thread has its own PRNG initialized with different seeds.

IV. RESULTS

A. Analysis of performance

Before computing coefficient values, it was necessary to conduct benchmarks to determine optimal template class fractions (ϕ_R,ϕ_C,ϕ_T) as well as the ratio of computational costs (φ). To determine the cost of sampling a configuration for each template, we conducted a 10^{10}-step calculation (10^9 steps for $n > 6$) with $P_{\boldsymbol{\alpha}} = 0$ for all $\boldsymbol{\alpha}$; we generated samples, screened each by looking for clique separators, and determined $\boldsymbol{\alpha}$ but did not compute f_B/π. To determine τ_c, the cost of computing f_B and π, we conducted a second calculation with 10^9 steps (10^8 for $n > 9$) and $P_{\boldsymbol{\alpha}} = 1$ such that f_B/π was computed for every unscreened configuration. The cost per configuration was the difference in time between the two calculations divided by the number of unscreened configurations. We found that τ_c is almost independent of template, so we have taken the average τ_c. To determine the ϕ_X, we conducted 10^9-step calculations for each order and for various values of each ϕ_X, taking the most efficient choice for use in further calculations. We considered only $\phi_T = 0, 0.1, 0.2$ and taking candidate values for ϕ_R and ϕ_C based on calculations with fewer steps. All of these benchmarks were run single-threaded on 2.83-GHz Intel Core 2 Quad Q9550 processors. The resulting values of τ_s, τ_c, ϕ, and φ are reported in Table IV. The benchmarks project total CPU times that are 50% ($n = 5$) to 95% ($n = 11$) of the times given in Table III, due partly to differences in hardware but primarily because of poor efficiency for the multithreaded implementation at low order.

We observe from Table IV that the time to compute f_B/π grows exponentially with n, as expected, while the time to generate and screen a configuration grows only a bit faster than linearly with n; the algorithm we use to detect clique separators in principle grows exponentially with n, but apparently this

TABLE IV. Benchmarks for B_n. The τ_s are CPU times to generate and screen a configuration and evaluate its $\boldsymbol{\alpha}$ for the tree (T), chain (C), and ring (R) template classes, and τ_c is the CPU time to compute f_B and π for one configuration. The ϕ are the optimized probabilities to select the tree, chain, or ring template class when generating a new configuration. φ is the ratio of τ_c to the ϕ-averaged τ_s. All τ values are reported in units of μs.

n	τ_{sT}	τ_{sC}	τ_{sR}	τ_c	ϕ_T	ϕ_C	ϕ_R	φ
5	1.40	1.01	2.12	9.0	0	0.2	0.8	4.69
6	1.91	1.36	3.13	17.7	0.1	0.2	0.7	5.96
7	2.48	1.86	4.26	32.2	0.1	0.3	0.6	8.88
8	3.09	2.29	6.27	76.1	0.1	0.5	0.4	18.08
9	3.78	2.82	7.79	174.9	0.1	0.6	0.3	41.73
10	4.82	3.25	9.84	560.1	0.1	0.6	0.3	97.41
11	6.05	4.28	12.73	1560.9	0.1	0.7	0.2	272.57

TABLE V. Binning parameters and properties for B_n, with n indicated in the first column. The values in the remaining columns are: the set of binning parameters used at each order, made in reference to the list given in Sec. III C; the number of bins; the fraction of bins observed to have zero variance in f_B/π; the fraction of configurations identified as having $f_B = 0$ by screening; the fraction of all configurations for which computation ought to be performed (N_c/N); the optimized fractional contribution of sampling to the total variance (σ_s^2/σ_π^2); and the fraction of the total CPU time spent on sampling and screening (t_s/t).

n	$\boldsymbol{\alpha}$ Parameters	No. of bins	Bins with $\sigma_{\boldsymbol{\alpha}}^2 = 0$	Configurations that pass screening	N_c/N	σ_s^2/σ_π^2	t_s/t
5	1	5	100%	20.1%	0	1.00	1.00
6	1–5	23	100%	8.9%	0	1.00	1.00
7	1–5	150	89%	4.9%	0.103%	0.98	0.99
8	1–5	1776	73%	2.8%	0.146%	0.97	0.97
9	1–5	31 794	53%	2.0%	0.173%	0.92	0.93
10	1–5	631 832	39%	1.9%	0.152%	0.86	0.87
11	1–5	8 926 893	30%	1.4%	0.102%	0.77	0.78

behavior is not yet dominant for the values of n encountered here. Rings are most costly to sample, taking about 2 to 3 times longer than a chain or tree of the same size. The optimized template distribution de-emphasizes the use of rings as n increases, with chains becoming the most probable template choice.

Parameters and observations related to the screening and binning processes are reported in Table V. The binning is perfect for $n \leqslant 6$, with the bins essentially serving as a look-up table. With increasing n, the fraction of bins with zero variance decreases to 30% for $n = 11$. At the same time, the fraction of configurations that pass the screening process diminishes to almost 1%, and, as a consequence, the number of configurations where f_B/π is actually computed never exceeds 0.2% of the total configurations generated for all n. The amount of time spent sampling relative to the total time should in principle equal the ratio σ_s^2/σ_π^2 for the optimum allocation of effort. It is seen in Table V that they differ slightly, and this is due to capping of some of the $P_{\boldsymbol{\alpha}}$ at 1.0.

The relative difficulty $\bar{D}_n$ for the calculation of virial coefficient B_n, based on our calculations, is displayed in Fig. 2. The figure in fact shows the difficulty for several variations of the binning method, projected from the data in Tables IV and V. Due to an error in our original development, the algorithm we used to optimize $P_{\boldsymbol{\alpha}}$ differs from that described in Sec. III C, so in addition to presenting the relative difficulty for the calculations as we performed them, we show also $\bar{D}_n$ projected for the optimal selection of $P_{\boldsymbol{\alpha}}$ (this is optimal for the fixed definition of $\boldsymbol{\alpha}$). In addition, we project $\bar{D}_n$ for the case in which no binning is performed (f_B/π is computed for all configurations that pass the screening), and for the (idealized) case in which we have perfect binning, such that the each bin has zero variance and f_B/π never needs to be computed more than once for any bin.

For all variations of the binning approach, we observe a smooth exponential growth in $\bar{D}_n$ with virial-coefficient order. What is rather surprising is what seems at first glance to be an insensitivity of $\bar{D}_n$ to how much f_B/π is computed—the no-binning and perfect-binning cases do not differ radically on the scale of the figure. The effect of the optimization of $P_{\boldsymbol{\alpha}}$ is attenuated considerably by the fact that only a low fraction of trial configurations make it through the screening process. With only a few percentages of configurations available to be considered for calculation of f_B/π, there is not much on which to optimize. Still, the benefit of binning is apparent in the inset, which shows the ratio of each $\bar{D}_n$ to the perfect-binning ideal. For B_{11}, $\bar{D}_n$(no-binning) is almost double $\bar{D}_n$(optimal-binning), which corresponds to a nearly fourfold increase in CPU time. Further, Fig. 2 suggests that the binning method is becoming increasingly effective as n increases.

A fit to the relative-difficulty data for the optimum implementation yields

$$\log_{10}[\bar{D}_n/(\text{sec})^{1/2}] = -4.78 + 0.408n + 0.0205n^2, \quad (33)$$

which can be used to project the amount of CPU required to obtain higher-order virials to a desired precision. Thus, we expect that using the methods and hardware employed for the present work, evaluation of B_{12}, B_{13}, and B_{14}, each to a precision of 10%, would require 4.1, 280, and 24 000 CPU-years, respectively. If binning were not used, we project

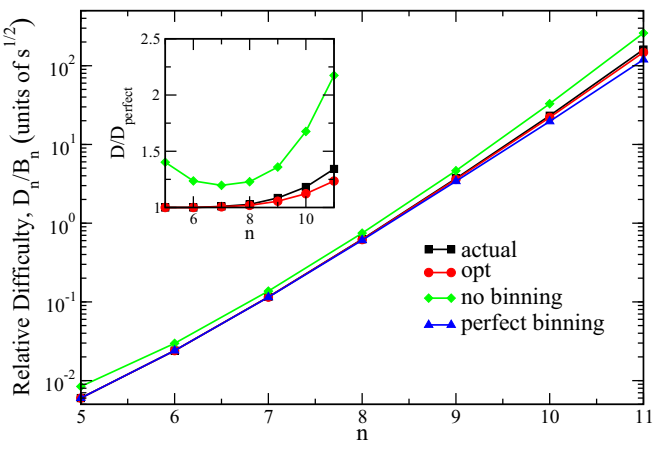

FIG. 2. (Color online) Relative difficulty of the virial-coefficient calculation as a function of the order of the coefficient. Points are based on the calculations reported in this work, and each line is a quadratic fit to the corresponding data. The points labeled "actual" are from the (nonoptimal) binning algorithm employed here; "opt" is the estimated data if using the optimal algorithm described in Sec. III C; "no binning" indicates projected results if the binning method were not used at all; "perfect binning" projects the results if $P_{\boldsymbol{\alpha}} = 0$ for all $\boldsymbol{\alpha}$. The inset shows the ratio of $\bar{D}_n$ for each case relative to the perfect-binning case.

that evaluation of B_{12}, B_{13}, and B_{14} would instead require 21, 2700, and 470 000 CPU-years.

Using Wheatley's method, the time required for calculation of f_B for each configuration is expected to increase as 3^n [31]. If this were the only factor affecting the difficulty, then we would expect $\log_{10} \bar{D}_n = \text{const} + 0.239n$. We observe an increase faster than this, because at larger n more sampling is required to achieve a result to a given relative precision. It appears that the contribution of this effect to the relative difficulty is about the same as the growth of effort needed to compute f_B. We note that τ_C is increasing at about 2.9^n near $n = 10$.

B. Virial coefficients

The hard-sphere virial coefficients computed here are reported in Table VI. The new values are 2 to 6 times more precise than previously available values from the literature, which are included in the table for comparison. All values are mutually consistent, within their reported uncertainties. We form an average of our values with the literature data, with each value weighted inversely with the square of its uncertainty, to arrive at the "best estimate" reported in the table for each coefficient.

The packing-fraction coefficients $\hat{B}_n$ have been observed to increase with n roughly as n^2 [58]. This behavior has been known at least since Carnahan and Starling [59], who developed their equation of state based on the observation that the virial coefficients are well approximated by $\hat{B}_n = n^2 + n - 2$. Table VI demonstrates this for the actual hard-sphere virial coefficients, where we tabulate $\hat{B}_n/n^2$. It should be noted that this observation of course does not provide any guarantee that as-yet-unknown higher-order coefficients will adhere to this trend.

When developing an equation of state based on the virial coefficients, one must address the question of how many coefficients to compute and what level of precision is sufficient

TABLE VI. Hard-sphere virial coefficients. The table records the values computed in this work, values from the literature with sources as indicated; to ease comparison of the precision for each n, zeros are appended to the literature values (and their uncertainties) such that they show the same number of digits as given in the corresponding new value. The best estimate represents a weighted average of the new and literature values. The fifth column gives the packing-fraction coefficients $\hat{B}_n \equiv B_n/(\frac{\pi}{6}d^3)^{n-1}$ divided by the n^2 approximation to them. The last column gives the coefficients of the approximant defined in Eq. (36). Numbers in parentheses indicate the 68% uncertainty in the last digit(s) of the reported values (σ_{B_n}). One should note that the $\hat{C}_n$ values are correlated, so their uncertainties cannot be propagated when combining them to compute Z; instead, it is necessary to work with the $\hat{B}_n$ uncertainties, as given in Eq. (38).

	$B_n/d^{3(n-1)}$				
n	This work	Literature	Best estimate	$\hat{B}_n/n^2$	$\hat{C}_n$
2	—	$\frac{2\pi}{3}$	2.094395102 . . .	1	0
3	—	$\frac{5\pi^2}{18}$	2.741556778 . . .	1.1111 . . .	1
4	—	2.636218008 . . .[f]	2.636218008 . . .	1.147798024 . . .	−1.635231617 . . .
5	2.1213811(13)	2.1213850(30)[a] 2.1213870(80)[b]	2.1213819(11)	1.1289751(6)	0.765304(15)
6	1.5669044(41)	1.5668970(90)[a] 1.5669300(400)[c] 1.5669130(150)[b]	1.566904(4)	1.105979(3)	−0.52890(11)
7	1.099157(11)	1.099190(20)[a] 1.099150(30)[c]	1.099162(10)	1.088614(10)	0.7337(6)
8	0.739342(37)	0.739430(90)[a] 0.739490(110)[c]	0.73937(3)	1.07075(5)	−1.010(4)
9	0.48487(12)	0.48470(40)[a] 0.48480(50)[d] 0.48470(50)[c]	0.48485(11)	1.0596(2)	1.46(2)
10	0.31248(30)	0.31380(110)[a] 0.31400(300)[e] 0.31290(120)[d]	0.3126(3)	1.0568(9)	−1.01(12)
11	0.19582(97)	0.19800(600)[a] 0.19800(700)[e]	0.1959(10)	1.046(5)	−0.6(7)
12	—	0.13(3)[a] 0.09(2)[e]	0.106(19)	0.91(16)	–17(24)

[a] Zhang and Pettitt [32].
[b] Kolafa *et al.* [28].
[c] Labik *et al.* [29].
[d] Clisby and McCoy [30].
[e] Wheatley [31].
[f] Exact: $\frac{\pi^2(219\sqrt{2}-712\pi+4131\text{atan}\sqrt{2})}{7560}$.

for each. Assuming the coefficients are of interest for their ability to yield an accurate value for the compressibility factor Z, these questions can be addressed through analysis of the difficulty and computational impact introduced in Sec. II.

The relation between the difficulty of computing Z, D_Z and the difficulty of the virial coefficients D_n provides a basis to consider how many coefficients should be computed. We consider a coefficient B_n worthwhile to compute only if its contribution to Z, Δ_n exceeds the uncertainty σ_Z when computing all coefficients up to B_n using total CPU t_Z. This comparison is density dependent. From the relations developed in Sec. II, we can show that the coefficient B_n should be computed if t_Z exceeds the threshold value,

$$t_Z > \left[\frac{1}{\Delta_n}\sum_{k=2}^{n} Z_k B_k \bar{D}_k\right]^2, \tag{34}$$

where $Z_k = |\partial Z/\partial B_k|$. With Z given by Eq. (2), and approximating the B_n for hard spheres using $\hat{B}_n \approx n^2$, we can write

$$t_Z > \left[\sum_{k=2}^{n} \eta^{k-n}\left(\frac{k}{n}\right)^2 \bar{D}_k\right]^2. \tag{35}$$

The right-hand side can be evaluated with the aid of Eq. (33). We find that it is insensitive to η for $\eta > 0.15$ or so, and, moreover, that for such densities the sum is well approximated by its last term, where $k = n$. Making this approximation, we find that that a coefficient B_n should be included in the calculation if $t_Z > \bar{D}_n^2$, i.e., the total CPU allotted to calculation of all coefficients is greater than the square of the relative difficulty for coefficient B_n. This suggests, for example, that if planning to use more than about 15 CPU-days in total, then calculation of coefficients up to and including B_{12} should be performed. This result assumes that the coefficients will be used directly in the VEOS as written in Eqs. (1) or (2) to evaluate Z, but as we show below, it is not always advantageous to use them this way.

We now turn to the issue of allocation of computational effort from the standpoint of improving the current results. Where should effort be directed next if one wants to improve the accuracy and precision of the hard-sphere equation of state as computed from the virial coefficients? First, we consider whether to direct effort to higher coefficients or whether to improve the precision of the available values. Figure 3 presents the uncertainty of Z—using the "best estimate" coefficient uncertainties recorded in Table VI—in comparison to the error incurred from truncation of the VEOS as given in Eq. (1). The figure shows that for densities greater than $\rho d^3 \approx 0.3$, neglect of the contribution from B_{13} introduces more error than the uncertainty in the virial coefficients. This suggests that improvement in the equation of state for these densities would benefit more from effort applied to calculation of B_{13}, and perhaps B_{14}, rather than toward increasing the precision of lower-order coefficients. This situation in fact held even before we generated the results presented in this work, except that the threshold for relevance of B_{13} was at about $\rho d^3 \approx 0.4$. From this perspective, the new data for B_n have not done anything to improve the characterization of Z in this range of density.

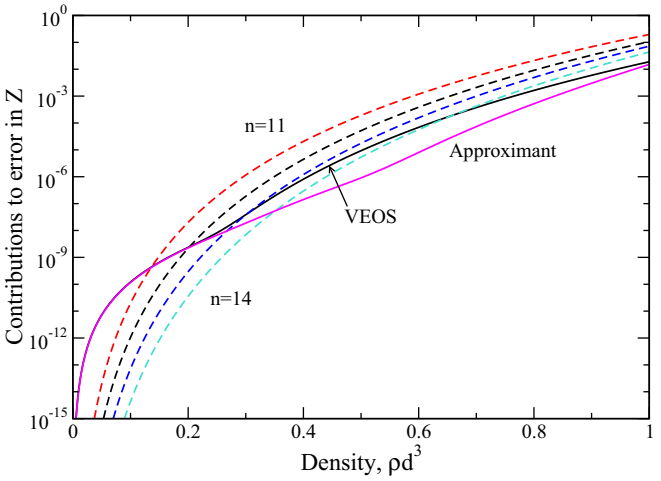

FIG. 3. (Color online) Errors in the compressibility factor Z when given by the virial equation of state, Eq. (1). Dashed lines are, from top to bottom, the estimated contribution to Z from B_{11} (red), B_{12} (green), B_{13} (blue), and B_{14} (orange). The latter two coefficients are estimated from $\hat{B}_n \approx n^2$. The solid black line (labeled "VEOS") is the uncertainty in Z as given by the VEOS propagated from the "best estimate" coefficients presented in Table VI, up to and including B_{12}. The solid pink line (labeled "Approximant") is the uncertainty in Z as given by the approximant including terms up to $n = 11$. The solid lines in this figure are the same as those given in Fig. 4.

This picture changes considerably if we make use of the regular behavior of B_n with n. We can form an approximant by summing the infinite series with the assumption that $\hat{B}_n = n^2$ and applying a series-based correction to force the approximant to match the known virial coefficients. Thus,

$$Z = \frac{1+\eta}{(1-\eta)^3}(1 + \hat{C}_2\eta + \hat{C}_3\eta^2 + \hat{C}_4\eta^3 + \ldots). \tag{36}$$

The coefficients $\hat{C}_k$ that cause the series expansion of Eq. (36) to match Eq. (2) are recorded in Table VI. The uncertainty in each of these coefficients is propagated from the $\hat{B}_n$ according to

$$\sigma_{\hat{C}_k} = \left[\sum_{n=2}^{k}\left(\frac{\partial \hat{C}_k}{\partial \hat{B}_n}\right)^2 \sigma_{\hat{B}_n}^2\right]^{\frac{1}{2}}, \tag{37}$$

(for hard spheres, $\sigma_{B_n} \equiv 0$ for $n = 2,3,4$). The uncertainty in Z is obtained via

$$\sigma_Z = \frac{1+\eta}{(1-\eta)^3}\left[\sum_{n=2}^{m}\left(\sum_{k=n}^{m}\left(\frac{\partial \hat{C}_k}{\partial \hat{B}_n}\right)\eta^{k-1}\right)^2 \sigma_{\hat{B}_n}^2\right]^{\frac{1}{2}}, \tag{38}$$

with m such that $\hat{C}_m$ is the last coefficient appearing in Eq. (36). Other choices of an approximant that matches known virial coefficients are possible and have been proposed [30,44,58,60–69]. The one suggested here is given just to make a point about gauging the precision required of the B_n, and the observations we make regarding it should to some degree apply to other approximants as well.

It is evident from the table that the $\hat{C}_n$ are determined with much less precision than the corresponding $\hat{B}_n$. This is because

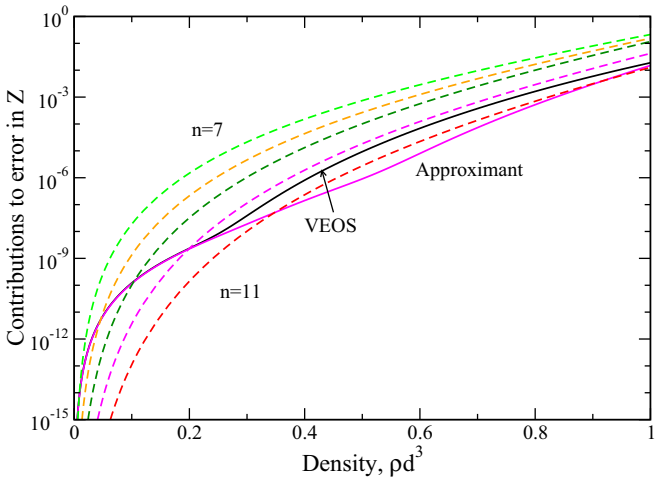

FIG. 4. (Color online) Sources of error in the approximant defined by Eq. (36). Dashed lines are, from top to bottom, the contribution (in absolute value) to Z from $\hat{C}_7$ to $\hat{C}_{11}$, respectively. Uncertainties in the lines representing the $\hat{C}_7$ to $\hat{C}_{10}$ contributions are not shown but would be barely discernible on the plot if they were. The uncertainty in the $\hat{C}_{11}$ line is significant, such that the line could be shifted up as high as the solid black line or as low as being off the bottom of the scale of the figure (inasmuch as $\hat{C}_{11} = 0$ is within the uncertainty). Solid lines are as described in Fig. 3.

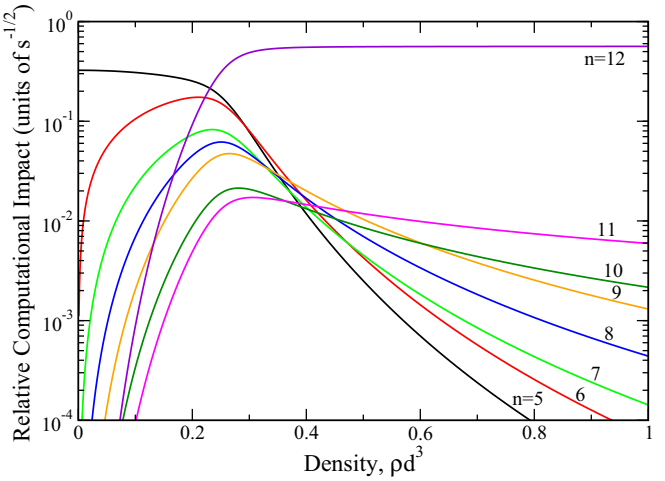

FIG. 5. (Color online) Relative impact on the precision of Z of additional computation directed toward each B_n, as a function of density, when Z is given by VEOS (Eq. (1)). Curves are presented for $n = 5$ (black) varying smoothly to $n = 11$ (pink), while the purple curve is for $n = 12$.

the $\hat{C}_n$ relate to the deviation of the $\hat{B}_n$ from their dominant behavior with n, and this causes them to be computed as the small difference of large numbers. Indeed, with the level of precision we now have for the $\hat{B}_n$, the approximant coefficients $\hat{C}_{11}$ and $\hat{C}_{12}$ cannot be distinguished from zero. Whether it is worthwhile to determine them more precisely depends on how much they contribute to Z, relative to the uncertainly in Z using the approximant with terms up to $n = 11$. This comparison is made in Fig. 4. First, we note that for intermediate densities, the uncertainty in the approximant is 10 times less than the uncertainty of the VEOS. If the VEOS were truncated (like the approximant) at $n = 11$, its uncertainty would be less than the approximant's, but given that the error of VEOS is already dominated by its truncation at these densities, such an adjustment would not be helpful overall.

Next we notice that above $\rho \approx 0.2$ (where the solid line showing the uncertainty in the approximant crosses the dashed line showing the contribution from $\hat{C}_{10}$), the approximant's uncertainty is considerably less than the contributions from $\hat{C}_{10}$ (and of course all lower-order coefficients), so B_{10} (which is the highest virial coefficient contributing to $\hat{C}_{10}$) makes a contribution that is significant in comparison to σ_Z. The uncertainty in $\hat{C}_{11}$ makes it difficult to gauge its contribution to Z, but it appears to be comparable to the present uncertainty in Z itself. If we exclude the new results, a plot similar to Fig. 4 would show the $\hat{C}_{10}$ contribution about equal to the uncertainty, and the $\hat{C}_{11}$ contribution well below it, so from this perspective the new results have been effective in reducing the overall error in Z. It seems clear that additional effort toward reducing the error in $\hat{C}_{11}$ would be helpful in reducing the error in Z further and would be more important than improving the precision of B_{12} or determining higher-order coefficients.

Thus, the advantage of the approximant is that is allows Z to be determined to a given accuracy using at least two fewer virial coefficients (based on comparison of contributions shown in Figs. 3 and 4). It does not offer any particular advantage with respect to precision, except to the extent that it allows less-precise, higher-order coefficients to be omitted from the calculation of Z without loss of accuracy. If computed for the same number of coefficients, the difficulty D_Z for the approximant is larger than that for VEOS, but if computed for the same accuracy, the difficulty for the approximant is much less.

We now turn to the issue of deciding where to direct new effort toward improving the precision of the known coefficients in order to have the most impact on the uncertainty in Z. The

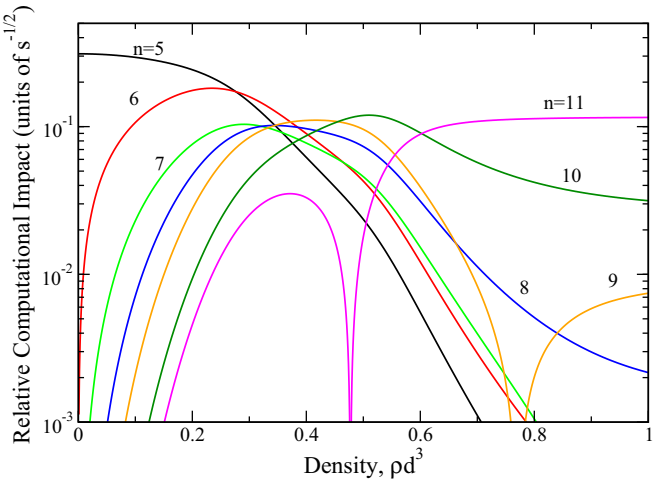

FIG. 6. (Color online) Relative impact on the precision of Z of additional computation directed toward each B_n, as a function of density, when Z is given by the approximant (Eq. (36)). Curves are presented for $n = 5$ (black) varying smoothly to $n = 11$ (pink). Cusps appear in some of the curves where $|\partial Z/\partial B_n| = 0$.

analysis is made by reference to the computational impact I_n, defined in Eq. (8). The computational impact differs depending on whether Z is given by the standard VEOS or an approximant. Using the results for the difficulty given in Fig. 2, we compute the relative impact $\bar{I}_n$ (impact relative to the uncertainty in Z) for each coefficient based on the "best estimate" B_n and σ_{B_n} as given in Table VI and present the results in Figs. 5 and 6. Figure 5 shows that for VEOS, reduction of error in Z for densities $\rho d^3 < 0.25$ is best obtained by more effort at computing B_5, while for densities greater than this, there is a clear indication to improve B_{12}. Figure 6 indicates that, for the approximant, effort toward B_5 is again the best strategy to reduce the error in Z for densities $\rho d^3 < 0.25$ and reducing the uncertainty of B_{11} is the best choice to improve Z at densities $\rho d^3 > 0.6$; for densities in between, attention to B_6, B_9, or B_{10} is best at different points. We estimate that for $\rho d^3 = 0.94$, another 70 CPU-years directed at B_{11} would be required to bring its uncertainty to a level where one should start to consider directing effort instead at B_{10}.

V. CONCLUSIONS

We have obtained results for hard-sphere virial coefficients with precision 2 to 6 times better than previously established values. It may seem that such a calculation is overkill, but attempts to understand subtle features and limits of the virial equation of state can benefit from knowledge of the coefficients to the highest precision possible. Our ability to complete these calculations for B_9 and higher is made feasible by the breakthrough presented by Wheatley, whose recursive algorithm circumvents the explosion in the number of graphs with increasing coefficient order. While his technique has opened the door to calculation of hard-sphere virial coefficients up to B_{12}, our analysis suggests that practical limits will still be found, almost certainly at B_{14}, without further improvements in hardware or technique.

One potential route for improvement is the optimized binning method that we introduced and employed in this work. There is room to tweak this approach through the definition of the bin parameter $\boldsymbol{\alpha}$, finding a choice that makes f_B/π span a narrow range of values within each bin (small $\Sigma_{\boldsymbol{\alpha}}$). Another area for improvement is in the generation of sample configurations, which ideally are generated in proportion to their importance to the virial-coefficient average. This means, first, that they pass screening, which will leave more samples for the binning optimization to work on. But, more generally, the sampled configurations should yield values of f_B/π that show little variation across bins (small Σ_s). The ring template generally provides better configurations in this respect, but for larger n there is an increasing number of configurations that it cannot generate but that are required for the average, so it is de-emphasized relative to the chain template. As a remedy, it might be worthwhile to introduce multiple-ring templates: two (or more) rings sharing a common vertex or two (or more) rings sharing a bond (the latter would take some effort to implement, but it may be tractable).

The notion of the *difficulty*—defined formally here via Eq. (3)—as a measure of computational effort is largely an obvious construction, inasmuch as it is well understood that the group $t^{1/2}\sigma$ is invariant with t and σ for these calculations. Still, we have not seen routine use of this quantity in published work as a point of reference for characterizing the effort needed to obtain a precise result from a calculation. The relative difficulty in particular is useful for general comparisons, as it does not depend on the scale or units of the quantity being computed. The calculations performed here have relative difficulty ranging from 0.006 to 140 (CPU-seconds)$^{1/2}$ for $n = 5$ to $n = 11$. In this work, we have shown how the difficulty can be applied to estimate the effort required to complete calculations at new conditions (e.g., higher-order virial coefficients) and to direct the computational effort to minimize uncertainty in the final result.

ACKNOWLEDGMENTS

This work is supported by the U.S. National Science Foundation (NSF) Grant No. CHE-1027963. Computational resources were provided by the University at Buffalo Center for Computational Research.

[1] E. Mason and T. Spurling, *The Virial Equation of State* (Pergamon Press, Oxford, 1969).
[2] D. McQuarrie, *Statistical Mechanics* (University Science Books, South Orange, NJ, 2000).
[3] A. J. Masters, J. Phys.: Condens. Matter **20**, 283102 (2008).
[4] A. J. Schultz and D. A. Kofke, J. Chem. Phys. **130**, 224104 (2009).
[5] S. Caracciolo, B. Mognetti, and A. Pelissetto, J. Chem. Phys. **125**, 094903 (2006).
[6] S. Caracciolo, B. Mognetti, and A. Pelissetto, J. Chem. Phys. **125**, 094904 (2006).
[7] K. R. S. Shaul, A. J. Schultz, and D. A. Kofke, J. Chem. Phys. **135**, 124101 (2011).
[8] K. M. Benjamin, A. J. Schultz, and D. A. Kofke, J. Phys. Chem. C **111**, 16021 (2007).
[9] K. M. Benjamin, A. J. Schultz, and D. A. Kofke, J. Phys. Chem. B **113**, 7810 (2009).
[10] R. Hellmann and E. Bich, J. Chem. Phys. **135**, 084117 (2011).
[11] B. Jäger, R. Hellmann, E. Bich, and E. Vogel, J. Chem. Phys. **135**, 084308 (2011).
[12] H. M. Kim, A. J. Schultz, and D. A. Kofke, J. Phys. Chem. B **116**, 14078 (2012).
[13] A. Bellemans, Physica **28**, 617 (1962).
[14] A. Bellemans, Physica **28**, 493 (1962).
[15] A. Bellemans, Physica **29**, 548 (1963).
[16] F. H. Stillinger and F. P. Buff, J. Chem. Phys. **37**, 1 (1962).
[17] J. S. Rowlinson, Proc. R. Soc. Lond. A Math. Phys. Sci. **402**, 67 (1985).
[18] J. H. Yang, A. J. Schultz, J. R. Errington, and D. A. Kofke, J. Chem. Phys. **138**, 134706 (2013).
[19] G. Garberoglio and A. H. Harvey, J. Res. Natl. Inst. Stan. Technol. **114**, 249 (2009).
[20] G. Garberoglio and A. H. Harvey, J. Chem. Phys. **134**, 134106 (2011).

[21] G. Garberoglio, Chem. Phys. Lett. **525-526**, 19 (2012).
[22] K. R. S. Shaul, A. J. Schultz, D. A. Kofke, and M. R. Moldover, Chem. Phys. Lett. **531**, 11 (2012).
[23] K. R. S. Shaul, A. J. Schultz, and D. A. Kofke, J. Chem. Phys. **137**, 184101 (2012).
[24] A. Isihara and R. V. Hanks, J. Chem. Phys. **36**, 433 (1962).
[25] J. F. Douglas and E. J. Garboczi, Adv. Chem. Phys. **91**, 85 (1995).
[26] E. J. Garboczi and J. F. Douglas, Phys. Rev. E **53**, 6169 (1996).
[27] M. N. Rosenbluth and A. W. Rosenbluth, J. Chem. Phys. **22**, 881 (1954).
[28] J. Kolafa, S. Labik, and A. Malijevsky, Phys. Chem. Chem. Phys. **6**, 2335 (2004).
[29] S. Labik, J. Kolafa, and A. Malijevsky, Phys. Rev. E **71**, 021105 (2005).
[30] N. Clisby and B. McCoy, J. Stat. Phys. **122**, 15 (2006).
[31] R. J. Wheatley, Phys. Rev. Lett. **110**, 200601 (2013).
[32] C. Zhang and B. M. Pettitt, Mol. Phys. **112**, 1427 (2014).
[33] J. A. Barker and J. J. Monaghan, J. Chem. Phys. **36**, 2564 (1962).
[34] J. Barker, P. Leonard, and A. Pompe, J. Chem. Phys. **44**, 4206 (1966).
[35] K. Dyer, J. Perkyns, and B. Pettitt, Theor. Chem. Acc. **105**, 244 (2001).
[36] K. R. S. Shaul, A. J. Schultz, A. Perera, and D. A. Kofke, Mol. Phys. **109**, 2395 (2011).
[37] D. Frenkel and B. Smit, *Understanding Molecular Simulation: From Algorithms to Applications* (Academic Press, San Diego, 2002).
[38] J. K. Singh and D. A. Kofke, Phys. Rev. Lett. **92**, 220601 (2004).
[39] K. M. Benjamin, A. J. Schultz, and D. A. Kofke, Ind. Eng. Chem. Res. **45**, 5566 (2006).
[40] D. A. Kofke and P. T. Cummings, Mol. Phys. **92**, 973 (1997).
[41] D. A. Kofke, Mol. Phys. **102**, 405 (2004).
[42] D. A. Kofke, Fluid Phase Equil. **228-229**, 41 (2005).
[43] J. Kolafa and S. Labík, Mol. Phys. **104**, 1915 (2006).
[44] E. J. van Rensburg, J. Phys. A Math. Gen. **26**, 4805 (1993).
[45] L. Viererblova, J. Kolafa, S. Labik, and A. Malijevsky, Phys. Chem. Chem. Phys. **12**, 254 (2010).
[46] A. Vlasov, X. You, and A. Masters, Mol. Phys. **100**, 3313 (2002).
[47] R. Bellman, J. ACM **9**, 61 (1962).
[48] M. Held and R. M. Karp, J. Soc. Indust. Appl. Math. **10**, 196 (1962).
[49] D. L. Kreher and D. R. Stinson, *Combinatorial Algorithms: Generation, Enumeration, and Search*, Discrete Mathematics and Its Applications (CRC Press, Boca Raton, FL, 1988).
[50] S. Pemmaraju and S. Skiena, *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica* (Cambridge University Press, Cambridge, 2003).
[51] H. Prüfer, Arch. Math. Phys. **27**, 742 (1918).
[52] W. Kocay and D. L. Kreher, *Graphs, Algorithms, and Optimization 5.8: The Matrix-Tree Theorem*, Discrete Mathematics and Its Applications (CRC Press, Boca Raton, FL, 2004), pp. 111–116.
[53] R. E. Tarjan, Discret. Math. **55**, 221 (1985).
[54] F. H. Ree and W. G. Hoover, J. Chem. Phys. **46**, 4181 (1967).
[55] B. D. McKay, Congressus Numerantium **30**, 45 (1981).
[56] B. D. McKay and A. Piperno, J. Symbol. Computat. **60**, 94 (2014).
[57] M. Matsumoto and T. Nishimura, ACM Transact. Model. Comput. Simul. **8**, 3 (1998).
[58] J. Hu and Y.-X. Yu, Phys. Chem. Chem. Phys. **11**, 9382 (2009).
[59] N. F. Carnahan and K. E. Starling, J. Chem. Phys. **51**, 635 (1969).
[60] N. S. Barlow, A. J. Schultz, S. J. Weinstein, and D. A. Kofke, J. Chem. Phys. **137**, 204102 (2012).
[61] M. N. Bannerman, L. Lue, and L. V. Woodcock, J. Chem. Phys. **132**, 084507/1 (2010).
[62] M. Á. G. Maestre, A. Santos, M. Robles, and M. L. de Haro, J. Chem. Phys. **134**, 084502 (2011).
[63] M. Ončák, A. Malijevský, J. Kolafa, and S. Labík, Condens. Matter Phys. **15**, 23004 (2012).
[64] I. C. Sanchez and J. S. Lee, J. Phys. Chem. B **113**, 15572 (2009).
[65] I. C. Sanchez, J. Chem. Phys. **101**, 7003 (1994).
[66] A. Santos and M. López De Haro, J. Chem. Phys. **130**, 214104 (2009).
[67] J. Tian, Y. Gui, and A. Mulero, J. Phys. Chem. B **114**, 13399 (2010).
[68] L. V. Woodcock, Am. Inst. Chem. Eng. J. **58**, 1610 (2011).
[69] L. V. Yelash and T. Kraska, Phys. Chem. Chem. Phys. **3**, 3114 (2001).