Fast Monte Carlo algorithms for knotted polymers

Stephen R. Quake*
Theoretical Physics, Department of Physics, University of Oxford, 1 Keble Road, Oxford OX1 3NP, United Kingdom
(Received 23 March 1995)

Fast implementations of the kink-jump-crankshaft and Berg-Foerster-Aragao de Carvalho-Caracciolo-Froehlich (BFACF) algorithms are discussed and applied to the study of topologically knotted polymers. The effect of knots on the size scaling laws of polymers is investigated. Finally, a comparison between the size scaling of the kink-jump-crankshaft and BFACF alogorithms is made

PACS number(s): 02.70.Lq, 02.40.−k, 83.10.Nn, 87.10.+e

INTRODUCTION

The statistical behavior of knots in polymers is a fascinating issue that appeals on many levels. It has been recognized for some time that topological entanglements and constraints affect the behavior of polymers [1–3], while knots in DNA have important biological functions [4]. Knots occur with nontrivial probability [5] and measurable effects in long polymers [6–8]. Since it is difficult to incorporate topological constraints into analytic calculations, computer simulations have played an important role in the study of knotted polymers. In this paper, we describe fast implementations of two topology-preserving polymer Monte Carlo algorithms, which we then use to study static and dynamic scaling properties of knotted polymers.

One of the oldest and conceptually simplest algorithms for polymer Monte Carlo is the Verdier-Stockmayer kink-jump algorithm combined with an out of plane crankshaft move (KJC) [9]. Although it has been shown that the KJC method is nonergodic [10] and thus inappropriate for studying static quantities, little is known about the size of the effects of the nonergodicity. KJC is still used to study dynamic properties, since it is thought to mimic the intrinsic dynamics of a polymer in solution [9]. In fact, it has been proven that the long wavelength dynamics of KJC are equivalent to the Rouse model of polymer dynamics [11]. The algorithm is defined as follows: the polymer is embedded in a simple cubic lattice; at each time step the algorithm chooses a monomer and attempts to change its position according to one of two moves. The moves are a kink-jump and a crankshaft motion, shown in Fig. 1. The success of a move depends on the availability of empty lattice sites in the vicinity of the original monomer. The nonergodicity comes from the fact that there are certain highly compact configurations that cannot relax; an example in two dimensions is shown in Fig. 1.

While it has been proven that all local length-conserving algorithms display this kind of nonergodicity [10], the Berg-Foerster-Aragao de Carvalho-Caracciolo-Froehlich algorithm (BFACF) [12–14] is a simple extension to the KJC method that has been shown to be ergodic within a fixed knot type [15]. It is thus an ideal tool with which to study the effects of knots. The difference between the BFACF algorithm and the KJC method is that in BFACF the crankshaft move is allowed to collapse, as shown in Fig. 1. The BFACF algorithm can be biased in order to prefer a particular length range [17]. In this scheme, a move increasing or decreasing the length of the polymer is accepted only with a particular probability β. If β depends on N, then the algorithm is biased toward producing certain lengths, but still provides a uniform distribution for each particular length. Finally, we note that the BFACF algorithm preserves the topology of the polymer [15], and it is clear by analogy that the proof also applies to KJC.

The pivot algorithm [16] has generally been favored over KJC and BFACF for high precision numerical studies of polymer statics, since pivot only costs $O(N)$ for a global change, while previous implementations of KJC and BFACF were of order $O(N^2)$. However, the pivot algorithm does not preserve the topology of the polymer. Other workers using the pivot algorithm to study topology have checked potential changes in knot type by calculating a topological invariant such as the Alexander polynomial at every iteration of the algorithm [17]. This is

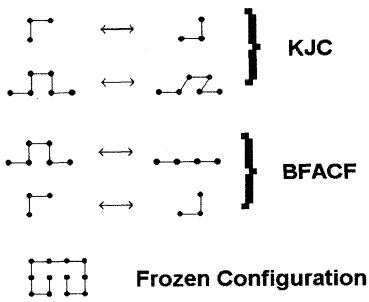

FIG. 1. Illustration of the allowed moves for the kink-jump-crankshaft (KJC) and BFACF algorithms. Also shown is a configuration that is "frozen" for the two dimensional KJC algorithm. Similar configurations exist in three dimensions.

*Present address: Department of Physics, Stanford University, Stanford, CA 94305.

computationally costly and may offset the advantage gained from using the pivot algorithm. It also has the caveat that no perfectly discriminating topological invariant has been discovered for knots, so an uncontrolled error is introduced.

In this paper, we describe an $O(N)$ implementation of the BFACF and KJC algorithms. This implementation was used previously to study the static and dynamic behaviors of knotted polymers as a function of knot type [18]. Here, we investigate the size scaling laws of fixed knot types and the nonergodicity of the KJC algorithm. The former question was first addressed by Janse van Rensburg and Whittington [17]; we present the first confirmation of that work and have been able to extend their results. They used a hybrid pivot-BFACF method, and their work was done on a large, specialized computer. The fast implementation of the biased BFACF algorithm described here produces comparable results on a workstation with a small ($\sim$10 h/knot) amount of CPU time. This implementation was also used for the KJC algorithm to study the dynamic relaxation time of knotted and unknotted polymers.

IMPLEMENTATION

Most previous implementations of KJC and BFACF used a linked list data structure and a hash table to keep track of the polymer and check for self-intersections, respectively [19]. The drawback to this is that the hash table checks are of order $O(N)$, where N is the length of the polymer. While this is not a problem for algorithms that make large changes, such as the pivot algorithm, for incremental local changes it becomes costly; the cost of making a global change with local moves becomes $O(N^2)$. An alternative solution is to keep the entire lattice stored in memory: then checking self-intersection is a local $O(1)$ procedure. This has generally been rejected because one expects that the memory costs go as the lattice volume, $O(N^3)$. However, that is actually a large overestimate, since the polymer crumples up into a ball whose radius of gyration goes as $N^{0.6}$. Then the volume of the ball is $N^{1.8}$, which means that the lattice only requires $N^{1.8}$ memory. Thus, by using a lattice, one reduces CPU cost by $O(N)$ at the expense of increasing memory demand by $O(N^{0.8})$. Memory is cheap in modern workstations, and studying a quite long polymer of length $N=10\,000$ requires only $O(10$ megabytes) of memory, a not unreasonable amount. In this paper we consider polymers up to length $N=800$, which requires $O(100$ kilobytes).

The last issue to resolve is the cost of choosing a monomer. Since the lattice method does not have the advantage of having all of the locations of the polymer in a list, it is not trivial to choose a random monomer. Choosing a random lattice location is not efficient, since the probability of finding an occupied location is $N/N^{1.8}=N^{-0.8}$. The cost of randomly finding a monomer is thus $O(N^{0.8})$. However, one can "walk along" the polymer in $O(1)$ time. One starts at an occupied lattice location, then walks a small random number (e.g., between 3 and 5) of monomers further along the polymer to find the next location with which to attempt a KJC or BFACF move. This costs $O(1)$, and the entire algorithm becomes $O(1)$ in CPU time to perform a local move.

Since it is a difficult task to try to visualize the configuration of a complicated knot in three dimensions, an automated system to implement the initial condition was developed. To create a three dimensional embedding of a knot, it is enough to know the projection of the knot into a plane and the ordering of each self crossing. In fact, the minimum number of such self crossings is a topological invariant and forms the basis for the knot classification system: 4_1 is the first knot with four self-crossings. Pictorial representations of projections of all knots up to and including ten crossings are found in Rolfsen [20]. It is easy to transcribe the projections onto a two dimensional lattice, and to make a list of the ordering of the crossings. With this data, a computer program can follow the projection and add in the appropriate z coordinate. It simply follows the knot segments, adding in a fixed z coordinate. The projection is self-avoiding except for crossings, so the embedding will be as well. At a crossing, the algorithm checks the list of crossings to see whether it should go over or under the other segment. If the z value of the intersecting segment has not been assigned yet, the algorithm inserts the previous segment's z value. If the z value has been assigned, the algorithm inserts the requisite number of z segments to insure that the next xy segment goes over or under the intersection, as appropriate. This system allows relatively painless embedding of knots in three dimensions.

The greatest drawback to the unbiased BFACF has been considered the long relaxation time, which leads to large statistical errors. As the relaxation time of the knot is of intrinsic importance in the KJC method as well, a brief summary of error calculation is presented here. Since repeated measurements of an observable A (such as the radius of gyration) are generally correlated, the error in the measurement is affected by the relaxation time of the algorithm. This is usually defined in terms of the relaxation of the autocorrelation function:

$$C_{AA}(t)=\langle A_s A_{s+t}\rangle-\langle A_s\rangle^2 . \tag{1}$$

The standard definition of the integrated autocorrelation time is

$$\tau(A)=\frac{1}{2}\sum_{t=-\infty}^{\infty}\frac{C_{AA}(t)}{C_{AA}(0)} . \tag{2}$$

If the autocorrelation function is exponential, Eq. (2) will give the decay constant. More generally, if the long time behavior is exponential, τ will be on the order of the longest relaxation mode. In the present application, the time resolution eliminated higher order modes and we simply fit $C_{AA}(t)$ to an exponential to determine τ.

The statistical uncertainty in a measurement of A is

$$\sigma^2(A)=\frac{2\tau(A)}{n}C_{AA}(0) . \tag{3}$$

We see that the role of the relaxation time is to reduce the number of statistically independent measurements from n to $n/2\tau$ [16].

Calculating the statistical errors in the BFACF results is subtle. Since we are, in generally, concerned with averaging a quantity over a fixed length, and the length of the polymer fluctuates during the algorithm, we are not making consecutive fixed-length measurements in time. The issue is further complicated by the fact that the distribution in time of polymers with a fixed length is not uniform. Hence, to count the effective number of independent measurements at a fixed length, we counted the number of occurrences of a particular length that were separated by at least 2τ time steps. To do this we counted the first occurrence of a particular length, then counted the next occurrence that was at least 2τ later, and so on. This number was used in place of $n/2\tau$ in Eq. (3).

Thermalization time relates closely to the issue of relaxation time; since, in general, one starts the algorithm with the polymer in a state far from equilibrium, how long must one iterate before correlations due to the initial state are gone? Madras and Sokol present a good explanation [16], in which the exponential thermalization time τ_{exp} is distinguished from the statistical relaxation τ defined above. For some algorithms $\tau_{\text{exp}} \gg \tau$; for the unbiased BFACF it has been shown that $\tau_{\text{exp}} = \infty$ [21]. However, we argue that an extremely biased BFACF is equivalent to KJC: the allowed moves are essentially the same if the length of the polymer is not allowed to fluctuate greatly. The relationship between τ_{exp} and τ for KJC has not been rigorously investigated, but there is indirect evidence that they are equal. It has been proved that an analytic model by Rouse is equivalent to the long wavelength modes of KJC [11]. Numerical integration of the Rouse equations shows that the thermalization of a polymer from an extended distribution has the same relaxation time as fluctuations about equilibrium [22], i.e., $\tau_{\text{exp}} \approx \tau$. Hence $\tau_{\text{exp}} \approx \tau$ for an extremely biased BFACF as well. Thus, to avoid the impractical thermalization times of BFACF, one can thermalize a configuration at an effectively fixed length with an extremely biased BFACF, and then open up the bias to take data on a wide range of lengths.

As a practical check on the previous argument, we compared the thermalization of an extended distribution to τ. Specifically, we started the unknot 0_1 in a square shape in the xy plane, and measured the radius of gyration as it crumples up into a three dimensional ball under the action of the extremely biased BFACF algorithm. The resulting graph of R_g versus time was fit to an exponential; the fitted relaxation time is an estimate of τ_{exp}. We find that in this particular example τ_{exp} and τ agree to within 10%, providing strong evidence that $\tau_{\text{exp}} \approx \tau$.

Since the above arguments are not a rigorous proof, we chose conservative thermalization times longer than the standard 20τ. In the case of KJC we iterated for $\sim 100\tau$ before taking data. For the pure BFACF algorithm we iterated for 500τ before taking data. To prevent this from being costly, we set the bias of the BFACF to prefer length 100 polymers during thermalization. This reduced τ_{exp} as discussed above and let us iterate quickly. After thermalization, we raised the ceiling on β to 800 and lowered the basement to 80, which allowed us to take data over an order of magnitude different lengths.

All simulations were performed on DEC Alpha and Sun workstations, and programmed in *C*. The software was prototyped on a 486 based PC. For the BFACF multilength algorithm, we iterated for $(1-3)\times 10^9$ BFACF attempts per knot after thermalization. For the KJC algorithm, we iterated for up to 10^9 KJC attempts per knot after thermalization.

RESULTS

The quantities of interest are both measures of the spatial extent of the equilibrium distribution. The first is the radius of gyration, defined as

$$R_g^2 = \frac{1}{N}\sum_{i=1}^{N} (\mathbf{R}_i - \overline{\mathbf{R}})^2 , \tag{4}$$

where $\mathbf{R}_i = (X_i, Y_i, Z_i)$ are the vertices of the polymer and $\overline{\mathbf{R}}$ is the center of mass. The second is the span, defined as

$$S = \tfrac{1}{3}(\max_{ij} |X_i - X_j| + \max_{ij} |Y_i - Y_j| + \max_{ij} |Z_i - Z_j|) . \tag{5}$$

Both should scale according to the single length scale in the problem: the length of the polymer. Furthermore, LeGuillou and Zinn-Justin [23] have worked out the confluent corrections to scaling, showing that the two quantities scale as

$$\begin{aligned} R_g^2 &= N^{2\nu}(C_K + b_K N^{-\Delta}) , \\ S &= N^{\nu}(D_K + f_K N^{-\Delta}) , \end{aligned} \tag{6}$$

where C_K, D_K, b_K, and f_K are constant coefficients that may depend on knot type. The estimates of the exponents from field theory by LeGuillou and Zinn-Justin are

$$\begin{aligned} \nu &= 0.5880(0.0015) , \\ \Delta &= 0.047(0.025) . \end{aligned} \tag{7}$$

It should be noted, however, that these are results from the n-vector model, which is an average over all knotted configurations, and may not strictly apply to individual knot classes. A three-parameter fit to the BFACF data for span and radius of gyration yields the values for ν as a function of knot type shown in Table I. All of the results

TABLE I. Fitted values and statistical errors of the scaling exponent ν for the observables radius of gyration and span. These values are derived from a three-parameter fit to Eq. (6). The agreement with the predicted theoretical value in Eq. (7) is good, although ν_{Rg} is systematically above the theoretical value.

Knot type	ν_{Rg}	σ_{Rg}	ν_s	σ_s
0_1	0.582	0.013	0.572	0.010
3_1	0.603	0.011	0.581	0.009
4_1	0.606	0.014	0.578	0.012
5_1	0.625	0.012	0.603	0.100
10_1	0.640	0.020	0.557	0.019
20_1	0.609	0.020	0.553	0.018

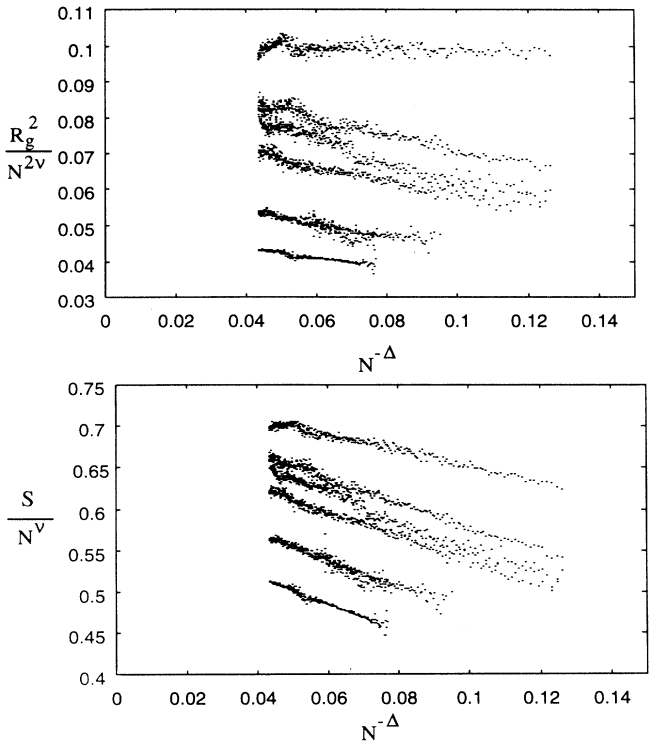

FIG. 2. The radius and gyration and span for several knot types, plotted to show the effects of the confluent corrections to scaling. The confluent correction is demonstrated by the dependence on $N^{-\Delta}$ after factoring out the excluded volume size scaling. The knots are 0_1, 3_1, 4_1, 5_1, 10_1, and 20_1.

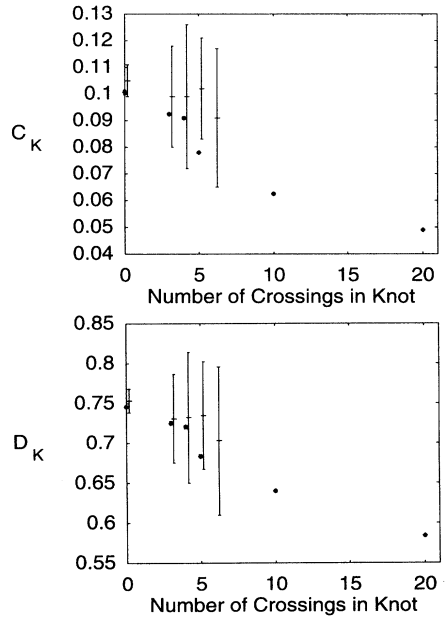

FIG. 3. The leading coefficient of size scaling for radius of gyration (top) and span (bottom). Diamonds are the data from this paper; pluses are the data from Janse van Rensburg and Whittington [17]. The coefficients were extracted from the three-parameter fits to Eq. (6), which takes into account confluent corrections to scaling. There is a clear systematic dependence on knot type. Knots used in the present study are 0_1, 3_1, 4_1, 5_1, 10_1, and 20_1.

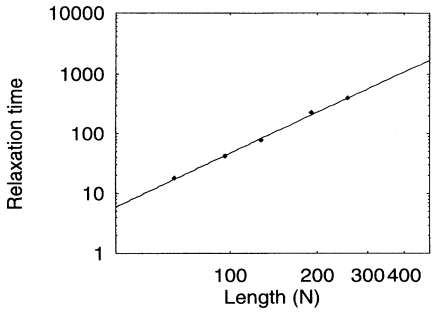

FIG. 4. Relaxation time as a function of knot length for the unknot (diamonds). The line is a two-parameter fit whose fitted scaling exponent $\nu=2.26(0.06)$ is in excellent agreement with $\nu=2.20$ predicted by the Rouse model.

for the exponent ν are close to the field theory value. The fits held Δ fixed to its field theory value; it is possible that Δ is knot dependent and this may cause systematic errors in the calculation. However, the results are in reasonable agreement with the expected values. Letting Δ be a free parameter in the fits led to large dependencies between the coefficients, indicating that the data do not have enough features to fit four free parameters.

After verifying the scaling exponent, we can study the leading coefficient; this is best examined through the confluent corrections. Janse van Rensburg and Whittington found that the coefficients C_K and D_K were independent of knot type K within the errors of their measurement [17]. However, it must be noted that there appears to be a weak systematic dependence in their coefficients. We have duplicated some of their results, but with smaller error bars, and have extended the measurement to more complex knots 10_1 and 20_1. Figure 2 shows the raw data, while Fig. 3 shows a comparison between the results. The error bars overlap in almost all cases, showing consistency between the two studies, and from the present data a clear systematic dependence upon knot complexity is evident. Hence the leading coefficient of scaling depends on the knot type.

We now turn to the dynamic behavior, and study the

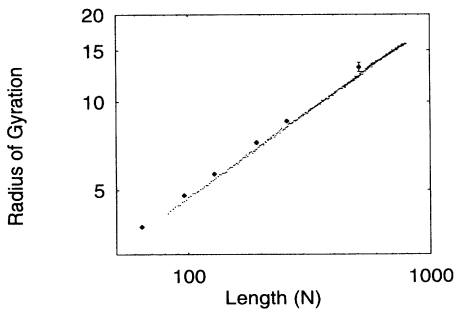

FIG. 5. Comparison between BFACF (dots) and KJC (diamonds) of the radius of gyration of the unknot 0_1. One sees a systematic deviation of KJC above BFACF; this is due to nonergodic swelling of the KJC configurations. However, within statistical error the scaling exponents agree between the two algorithms.

relaxation of the autocorrelation function via the KJC method. We studied the scaling of relaxation time versus length for the unknot, as shown in Fig. 4. The scaling exponent of τ with respect to polymer length was 2.27(0.06), in excellent agreement with the predicted Rouse value of 2.20. Previous studies [9] have also found exponents slightly higher than the predicted value; this is most likely a finite size artifact of the KJC algorithm.

Finally, it is possible to estimate the effects of nonergodicity in KJC by comparing the static quantities computed by KJC to those of BFACF (Fig. 5). The values of the radius of gyration are consistent within the error bars (not shown for BFACF) but we see that in both cases KJC is systematically above BFACF. The scaling law, however, is still close to the predicted value: the same fit that was applied to BFACF gives $\nu=0.60(0.04)$ for KJC, agreeing with both the BFACF and the field theory values within its error. Hence, it may be that the KJC system still obeys the scaling laws, but that the frozen states can be "renormalized" into a new "effective" excluded volume parameter. The fact that the algorithm is on a lattice is itself a finite size effect, since any frozen configuration can be unfrozen by rescaling into a finer grain lattice.

CONCLUSION

We have developed fast implementations of the BFACF and KJC algorithms for Monte Carlo simulation of polymers. These implementations are $O(N)$ faster than previous implementations, and are as fast as the pivot algorithm. We used the fact that BFACF and KJC preserve topology to study static and dynamic properties of knotted polymers. For size scaling as a function of length we find scaling exponents consistent with, but not in total agreement with, field theory predictions. It is possible that the confluent correction exponent depends on knot type, a case which is not accounted for in the field theory predictions and may explain the small discrepancies. The leading constant of scaling does appear to depend on knot type, but without resolving the previous issue it is difficult to draw an unambiguous conclusion. We also measured nonergodicity in the KJC algorithm. We find that the sizes of the polymers are systematically larger than found with BFACF, but that the scaling exponent is the same. Furthermore, the dynamical exponent agrees well with the Rouse model. Thus, although the nonergodicity of the algorithm prevents its use in high precision determinations of static scaling exponents, KJC remains a useful and self-consistent tool for studying polymer dynamics.

ACKNOWLEDGMENTS

The author would like to thank Robin Stinchcombe and Stu Whittington for helpful discussions and comments. He acknowledges financial support from the British Marshall Commission.

[1] S. F. Edwards, Proc. Phys. Soc. London **92**, 9 (1967).
[2] P. G. de Gennes, J. Chem. Phys. **55**, 572 (1971).
[3] M. Doi and S. F. Edwards, The Theory of Polymer Dynamics (Oxford University Press, Oxford, 1992).
[4] S. A. Wasserman and N. R. Cozzarelli, Science **232**, 951 (1986).
[5] D. W. Sumners and S. G. Whittington, J. Phys. A **21**, 1689 (1988).
[6] S. Y. Shaw and J. C. Wang, Science **260**, 533 (1993).
[7] V. V. Rybenkov, N. R. Cozzarelli, and A. V. Vologodskii, Proc. Nat'l. Acad. Sci. U.S.A. **90**, 5307 (1993).
[8] M. C. Tesi, E. J. Janse van Rensburg, E. Orlandini, D. W. Sumners, and S. G. Whittington, Phys. Rev. E **49**, 868 (1994).
[9] K. Kremer and K. Binder, Comput. Phys. Rep. **7**, 259 (1988).
[10] N. Madras and A. E. Sokal, J. Stat. Phys. **47**, 573 (1987).
[11] See. Ref. [3], p. 93 for a proof.
[12] B. Berg and D. Foerster, Phys. Lett. B **106**, 323 (1983).
[13] C. Aragao de Carvalho and S. Caracciolo, J. Phys. **44**, 323 (1983).
[14] C. Aragao de Carvalho, S. Caracciolo, and J. Froehlich, Nucl. Phys. B **215**, 209 (1983).
[15] E. J. Janse van Rensburg and S. G. Whittington, J. Phys. A **24**, 5553 (1991).
[16] N. Madras and A. E. Sokal, J. Stat. Phys. **50**, 109 (1988).
[17] E. J. Janse van Rensburg and S. G. Whittington, J. Phys. A **24**, 3935 (1991).
[18] S. R. Quake, Phys. Rev. Lett. **73**, 3317 (1994).
[19] D. E. Knuth, The Art of Computer Programming (Addison-Wesley, Reading, MA 1973), Vol. 3.
[20] D. Rolfsen, Knot and Links (Publish or Perish, Inc., Houston, 1990).
[21] A. D. Sokal and L. E. Thomas, J. Stat. Phys. **51**, 907 (1988).
[22] S. R. Quake, J. Chem. Phys. **101**, 4307 (1994).
[23] J. C. LeGuillou and J. Zinn-Justin, Phys. Rev. B **21**, 3976 (1980).