Morphology and phase diagram of *ABC* linear triblock copolymers: Parallel real-space self-consistent-field-theory simulation

Mingzhu Sun,[1] Peng Wang,[2,3] Feng Qiu,[1,*] Ping Tang,[1] Hongdong Zhang,[1] and Yuliang Yang[1,†]

[1]*Department of Macromolecular Science, The Key Laboratory of Molecular Engineering of Polymers, Ministry of Education, Fudan University, Shanghai 200433, China*

[2]*Supercomputing Center, Fudan University, Shanghai 200433, China*

[3]*Parallel Processing Institute, Fudan University, Shanghai 200433, China*

(Received 21 June 2007; published 3 January 2008)

A parallel algorithm designed for widely used distributed computer clusters is developed for the real-space self-consistent-field theory for polymers. We adopt an efficient data partition method to ensure high performance of this parallel algorithm on clusters, which enables us to explore unknown phase structures of complex block polymers with high accuracy and efficiency. As a benchmark test, the algorithm is applied to an *ABC* linear triblock copolymer, in which the volume fractions of the two end blocks of the polymer chain are equal ($f_A = f_C$), to simplify the discussion. The three-dimensional microphases and phase diagram of this triblock copolymer are investigated in detail.

DOI: 10.1103/PhysRevE.77.016701 PACS number(s): 02.70.−c, 82.35.Jk, 82.20.Wt, 81.16.Dn

I. INTRODUCTION

Block copolymers, in which two or more chemically different subchains form a single molecule, are a fascinating class of soft materials with unique structural and mechanical properties. Interest in block copolymers has grown considerably in recent years because of their ability to self-assemble into a variety of ordered structures with domain sizes in the nanometer range. Current experiments and theories suggest many promising and fascinating possibilities for creating novel nanostructures and templates based on block copolymers [1–3].

A variety of theoretical methods have been developed to study phase behaviors of block copolymer systems. One of the most successful theoretical frameworks for block copolymers is the self-consistent-field theory (SCFT) that has its origin in work by Edwards in the 1960s [4] and was explicitly adapted to treat block copolymers by Helfand in 1975 [5,6]. Currently, there are three efficient numerical approaches to solving the SCFT. The first is the spectral method which was developed by Matsen and Schick in 1994 [7]. This method is numerically efficient for a precise computation of free energies and phase diagrams. However, it requires the space group of the ordered phase as an input. This information is typically lacking in exploratory studies of composite block copolymer material of different architectures. The second is the real-space method which was suggested by Drolet and Fredrickson [8]. Their method does not require *a priori* knowledge about the equilibrium morphologies. Hence it is more flexible than the spectral method and has a greater predictive capability. However, the real-space method is computationally intensive and requires large memory resources. The third is the pseudospectral method which was proposed by Tzeremes *et al.* [9]. This numerical algorithm is numerically superior in stability and performance.

In order to describe the equilibrium morphology of a block copolymer fully, large simulation cells are needed and numerical simulations must be implemented in the three-dimensional (3D) space. Moreover, a large simulation cell is also needed to eliminate the effect of boundary conditions usually adopted in the simulation of bulk phase separation. Hence, the computation intensity is enlarged greatly and enormous memory is needed, which exceeds what a PC can provide. So it is essential to develop parallel SCFT algorithms for parallel computers which can provide huge computation and memory resources. In recent years, computer clusters have become major parallel processing platforms because of their high performance-to-price ratio and good scalability. For computer clusters, Sides and Fredrickson developed an efficient parallel method using a pseudospectral algorithm in which one fast Fourier transform (FFT) and one inverse FFT (IFFT) are applied in each time step to solve a modified diffusion equation [10]. Because arrays are divided and distributed among computing nodes, a global array transpose incurring all-to-all communications must be performed for each FFT and IFFT. So the communication volume of this algorithm is large, which implies that it is important for this algorithm to run on a cluster with a high-performance network to achieve good performance. Besides, on account of the nature of the FFT (including the IFFT), the performance of any algorithm based on a FFT and/or IFFT is relatively poor when all dimension lengths of the array are not highly composite, especially for a length that has few positive divisors. Hence, for computer clusters, we determined to develop a different parallel algorithm for the real-space SCFT method, whose performance is not sensitive to any dimension length of the array. It is thus especially suitable for screening unknown structures of complex block copolymers, in which the system size has to be continuously varied to ensure that an equilibrium structure is obtained. In our previous papers [11,12], the morphologies and phase diagrams of linear and star *ABC* triblock copolymers were investigated systematically by a real-space implementation of SCFT in 2D. Our study provides guidance for the design of

*Corresponding author. fengqiu@fudan.edu.cn

†Corresponding author. yuliangyang@fudan.edu.cn

microstructures in complex block copolymers. However, some known microstructures are equivocal in 2D, for example lamellar phases with beads inside or at the interface. Hence, the implementation of SCFT in 3D is more suitable for the discovery of novel assembly structures in complex block copolymers. In this paper, our parallel algorithm is introduced in detail for the case of *ABC* linear triblock copolymers.

The paper is organized as follows. In the next section, the SCFT for triblock copolymers is described briefly. In Sec. III, the parallel algorithm is introduced in detail. The 3D phase structure and phase diagram of *ABC* linear triblock copolymers are presented in Sec. IV. To simplify the discussion, we have assumed that the volume fractions of the two end blocks of the polymer chain are equal ($f_A = f_C$). In the last section, we draw our conclusions and make some remarks.

II. SCFT FOR *ABC* LINEAR TRIBLOCK POLYMERS

We briefly outline the formulation of the SCFT for linear *ABC* triblock copolymers in this section. We use a canonical ensemble approach and consider n_C linear triblock copolymer chains in a volume V. Each copolymer chain is built from N monomers of species $\alpha = A, B, C$. The average volume fractions of segment α in the system are f_α with $\Sigma_\alpha f_\alpha = 1$. Following the standard procedure of the SCFT [7,8], by introducing external auxiliary fields $\omega_\alpha(\mathbf{r})$, which are self-consistent molecular fields conjugated to the collective densities $\phi_\alpha(\mathbf{r})$, and the Lagrangian multipliers $\xi(\mathbf{r})$ for the incompressibility of the system, the free energy of the system can be written as

$$\frac{\beta F}{n_C} = -\ln\left(\frac{Q}{V}\right) + \frac{N}{V}\int dr[\chi_{AB}\phi_A\phi_B + \chi_{BC}\phi_B\phi_C + \chi_{AC}\phi_A\phi_C - \omega_A\phi_A - \omega_B\phi_B - \omega_C\phi_C - \xi(1 - \phi_A - \phi_B - \phi_C)], \quad (1)$$

where $\beta = 1/k_BT$, and ϕ_A, ϕ_B, and ϕ_C are the monomer density fields normalized by the local fractions of A, B, and C, respectively. Q denotes the partition function of the single chain subject to the mean field $\omega_\alpha(\mathbf{r})$, and can be expressed as $Q = \int dr\, q(\mathbf{r},s)q^\dagger(\mathbf{r},s)$. The polymer segment probability distribution function $q(\mathbf{r},s)$ gives the probability of finding segment s at position $\mathbf{r}$. It satisfies a modified diffusion equation

$$\frac{\partial q(\mathbf{r},s)}{\partial s} = \frac{b^2}{6}\nabla^2 q(\mathbf{r},s) - \omega(\mathbf{r})q(\mathbf{r},s), \quad (2)$$

where b is the statistical segment length of the polymer, $\omega(\mathbf{r}) = \gamma_A(s)\omega_A(\mathbf{r}) + \gamma_B(s)\omega_B(\mathbf{r}) + \gamma_C(s)\omega_C(\mathbf{r})$, and $\gamma_\alpha(s)$ is 1 if s belongs to the block α and 0 otherwise. The initial condition is $q(\mathbf{r},0) = 1$. Similarly, a second distribution function $q^\dagger(\mathbf{r},s)$ satisfies Eq. (2) but with the right-hand side multiplied by -1, and the initial condition is $q^\dagger(\mathbf{r},N) = 1$.

Minimizing the free energy in Eq. (1) with respect to ϕ_A, ϕ_B, ϕ_C, ω_A, ω_B, ω_C, and ξ leads to the following self-consistent field equations that describe the equilibrium morphology:

$$\omega_A(\mathbf{r}) = \chi_{AB}\phi_B(\mathbf{r}) + \chi_{AC}\phi_C(\mathbf{r}) + \xi(\mathbf{r}), \quad (3)$$

$$\omega_B(\mathbf{r}) = \chi_{AB}\phi_A(\mathbf{r}) + \chi_{BC}\phi_C(\mathbf{r}) + \xi(\mathbf{r}), \quad (4)$$

$$\omega_C(\mathbf{r}) = \chi_{AC}\phi_A(\mathbf{r}) + \chi_{BC}\phi_B(\mathbf{r}) + \xi(\mathbf{r}), \quad (5)$$

$$\phi_A(\mathbf{r}) = \frac{V}{NQ}\int_0^{f_AN} ds\, q(\mathbf{r},s)q^\dagger(\mathbf{r},s), \quad (6)$$

$$\phi_B(\mathbf{r}) = \frac{V}{NQ}\int_{f_AN}^{(f_A+f_B)N} ds\, q(\mathbf{r},s)q^\dagger(\mathbf{r},s), \quad (7)$$

$$\phi_C(\mathbf{r}) = \frac{V}{NQ}\int_{(f_A+f_B)N}^{N} ds\, q(\mathbf{r},s)q^\dagger(\mathbf{r},s), \quad (8)$$

$$1 = \phi_A(\mathbf{r}) + \phi_B(\mathbf{r}) + \phi_C(\mathbf{r}), \quad (9)$$

where $\xi(\mathbf{r})$ is chosen to be

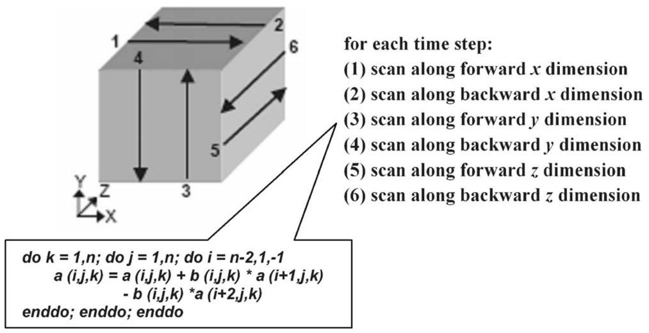
$$\xi(\mathbf{r}) = \lambda[1 - \phi_A(\mathbf{r}) - \phi_B(\mathbf{r}) - \phi_C(\mathbf{r})]. \quad (10)$$

Here λ is large enough to ensure the incompressibility of the system and the resulting density profiles and free energy should be independent of its particular value [13].

III. PARALLEL ALGORITHM

The numerical SCFT algorithm can be found in previous papers [10,14]. In the numerical implementation of real-space SCFT algorithms, the most time-consuming step is solving the modified diffusion equation to obtain the polymer segment probability distribution functions $q(\mathbf{r},s)$ and $q^\dagger(\mathbf{r},s)$. Hence, an efficient method for solving the modified diffusion equations is essential. We adopt the Douglas-Gunn scheme to solve Eq. (2) [15,16], because it is not only efficient but also unconditionally stable in 3D space. Moreover, our parallel algorithm is also suitable for other implicit schemes, such as the Crank-Nicholson and Dufort-Frankel schemes, because all these schemes share a similar program structure and our parallelizing technique can be applied directly.

Because the memory of a cluster is distributed physically among all computing nodes, the arrays must be divided and distributed among computing nodes to support large-scale simulation. Considering that message passing over the network is used to exchange information between nodes, which means that local memory access is much faster than remote memory access, arrays must be divided and distributed carefully so as to minimize expensive communications. The technology of dividing and distributing arrays is called "data partition" and is the key factor in achieving high efficiency for parallelizing programs on clusters. As far as the implicit schemes for SCFT, such as the Douglas-Gunn, Crank-Nicholson, and Dufort-Frankel schemes, are concerned, they use a similar data access pattern, which can be abstracted and illustrated as in Fig. 1. In each time step, the computation scans forward and/or backward over each spatial dimension



for each time step:
(1) scan along forward *x* dimension
(2) scan along backward *x* dimension
(3) scan along forward *y* dimension
(4) scan along backward *y* dimension
(5) scan along forward *z* dimension
(6) scan along backward *z* dimension

```
do k = 1,n; do j = 1,n; do i = n-2,1,-1
    a (i,j,k) = a (i,j,k) + b (i,j,k) * a (i+1,j,k)
              - b (i,j,k) *a (i+2,j,k)
enddo; enddo; enddo
```

FIG. 1. Sketch of 3D implicit schemes for SCFT.

of the arrays and this data access pattern is called "line sweeping computation." The inset in the figure shows a code fragment that scans backward along the first dimension of a, which is defined as the x dimension. In this fragment, the i loop iterates backward over the x dimension of a and a new value for each $a(i,j,k)$ element is computed from the value of its successors $a(i+1,j,k)$ and $a(i+2,j,k)$. Then, we say that there exist data dependences along the backward x dimension. Similarly, there exist data dependences along the forward x dimension when scanning forward along the first dimension of a.

Efficiently parallelizing a line sweeping computation on clusters is difficult because data dependences exist along all dimensions. Simple data partition methods cannot work efficiently. For example, the frequently used "static block partition" divides one or two dimensions of a 3D array into sections, and distributes data blocks evenly among processors. Under such a data partition, processor idleness will occur when scanning along at least one of the dimensions because of the lack of parallelism. It is possible to avoid such processor idleness by transposing arrays as necessary so that, in turn, each scan can be performed independently in parallel. But the time expended on array transposes can be considerable. Darte [17] has proposed a method to parallelize line sweeping computation, which can avoid processor idleness and array transpose. But his algorithm does not trade off well between computation cost and communication cost which affects its efficiency. Moreover, Darte's algorithm is very complicated in theory. So we develop here a data partition algorithm in which computation cost and communication cost are well balanced and higher performance can be achieved. In addition, our algorithm is much simpler than Darte's algorithm.

Our algorithm, like Darte's, always divides a 3D array along all its dimensions. In this manner, a 3D array is divided into a structured 3D grid. Then all data blocks are distributed among processors properly. In order to describe our data partition algorithm clearly, we first introduce some notations and assumptions used in this paper.

(1) The set of all positive integers is denoted as N. The set of all positive integers not more than t, where $t\in N$, is denoted as $N_t=\{1,2,\ldots,t\}$.

(2) p stands for the number of processors involved in parallel processing, where $p\in N$. So N_p denotes the set of all processors' ranks ranging from 1 to p.

(3) $A(n_1,n_2,n_3)$, $n_1,n_2,n_3\in N$, is the 3D array to be partitioned, where n_1, n_2, and n_3 represent the extents in the first, second, and third dimensions of A, respectively, and the array subscripts in every dimension of A begin with 1. The manner of dividing A into data blocks can be denoted by a 3D vector (r_1,r_2,r_3), where r_i, $i\in N_3$, represents the number of sections into which the array A is divided in the ith dimension and $r_i\in Nn_i$. We call (r_1,r_2,r_3) a "dividing vector." Then, obviously, the set $\bar{R}=\{(r_1,r_2,r_3)|r_i\in Nn_i,\ i\in N_3\}$ stands for the set of all possible dividing vectors for A. In this paper we never consider the effect of load imbalance that arises if n_i cannot be divided by r_i evenly.

(4) After being divided by $(r_1,r_2,r_3)\in\bar{R}$, A can be viewed as a structured 3D grid A'. Every data block in A' can be denoted by a vector (s_1,s_2,s_3) which identifies the data block's coordinate in A' with each $s_i\in Nr_i$, $i\in N_3$. So the set of all data blocks in A' can be expressed as the set $G=\{(s_1,s_2,s_3)|s_i\in Nr_i,\ i\in N_3\}$. Given $i\in N_3$ and $j\in Nr_i$, $H(i,j)=\{(s_1,s_2,s_3)|(s_1,s_2,s_3)\in G,\ s_i=j\}$, a subset of G, represents a set of data blocks in A' with the same ith coordinate, and we call $H(i,j)$ a slab of A'. From an inexact but intuitional view, $H(i,j)$ can be regarded as the jth slab along the ith dimension of A. It is obvious that there are no data dependences between any pair of data blocks in $H(i,j)$ as far as line sweeping computation is concerned, which means all data blocks in $H(i,j)$ can be processed in parallel.

(5) Given $i\in N_3$, $j\in Nr_i$, σ_{ij} is a function defined as follows: given any $(u_1,u_2,u_3)\in H(i,j)$, $\sigma_{ij}(u_1,u_2,u_3)=(v_1,v_2,v_3)$, where $\forall\ k\ (k\in N_3\wedge k\neq i\rightarrow v_k=u_k\wedge v_i=(u_i \bmod n_i)+1)$. Obviously, σ_{ij} is a bijection from $H(i,j)$ to $H(i,(j \bmod n_i)+1)$. Moreover, (u_1,u_2,u_3) and (v_1,v_2,v_3) are on a line perpendicular to $H(i,j)$ and $H(i,(j \bmod n_i)+1)$. So given $i\in N_3$, $j\in Nr_i$, $\bar{u}\in H(i,j)$, and $\bar{v}\in H(i,(j \bmod n_i)+1)$, there exist both forward and backward data dependences between $\bar{u}$ and $\bar{v}$ if and only if $\bar{v}=\sigma_{ij}(\bar{u})$. Considering that $H(i,j)$ and $H(i,(j \bmod n_i)+1)$ are two neighboring slabs, there will be communication if $\bar{u}$ and $\bar{v}$ are distributed to different processors.

(6) $B(r_1,r_2,r_3)$ is a 3D array used to record the owner of every data block in A', and array subscripts in every dimension of B begin with 1. For any data block $\bar{s}=(s_1,s_2,s_3)\in G$, the value of $B(\bar{s})$, an abbreviation of $B(s_1,s_2,s_3)$, indicates the rank of the processor owning $\bar{s}$. It is manifest that distributing data blocks in G among p processors is equivalent to assigning every element of B a value belonging to N_p.

(7) When scanning along the ith $(i\in N_3)$ dimension, no matter whether forward or backward, we suppose the number of successors that are used to compute an element of A is a constant, which is indicated as L_i.

(8) Let $V=n_1\times n_2\times n_3$, $V_i=n_1\times n_2\times n_3/n_i$, and $NB_i=r_1\times r_2\times r_3/r_i$, where $i\in N_3$; then V stands for the number of elements in A, V_i gives the number of elements in any plan of A vertical to the ith dimension, and NB_i gives the number of data blocks belonging to any $H(i,j)$ where $j\in Nr_i$.

(9) The computation cost spent on calculating every element of A during scanning along any dimension, no matter whether forward or backward, is assumed to be a constant K_{comp}.

(10) The communication cost spent on transmitting an element of A between two different processors is supposed to be a constant K_{comm}. This hypothesis is true for most modern high-performance cluster networks such as Myrinet, QsNet, and Infiniband when the data volume of every communication is not too small to ignore the overhead of additional computation associated with the communication.

(11) The equation $T=T_{\text{comp}}+T_{\text{comm}}$ is always true, where T, T_{comp}, and T_{comm}, respectively, represent the total time, the time spent on computation, and the time spent on communication, when scanning along every dimension of A once, no matter whether forward or backward.

Given any dividing vector $\bar{r}=(r_1,r_2,r_3)\in\bar{R}$, our algorithm distributes all data blocks once for all in a way ensuring that any slab $H(i,j)$, $i\in N_3$, $j\in Nr_i$, can be divided exactly into x subsets, denoted as $H_1(i,j)$, $H_2(i,j)$, …, and $H_x(i,j)$, which possess the following four characteristics:

(1) $\forall y,\ \forall \bar{u}_1,\ \forall \bar{u}_2\ [y\in N_x\wedge \bar{u}_1\in H_y(i,j)\wedge \bar{u}_2\in H_y(i,j)\wedge \bar{u}_1\neq\bar{u}_2\rightarrow B(\bar{u}_1)\neq B(\bar{u}_2)]$;

(2) $\forall y,\ \forall \bar{u}_1,\ \forall \bar{u}_2\ [y\in N_x\wedge \bar{u}_1\in H'_y(i,j)\wedge \bar{u}_2\in H'_y(i,j)\wedge \bar{u}_1\neq\bar{u}_2\rightarrow B(\bar{u}_1)\neq B(\bar{u}_2)]$, where for any $y\in N_x$, $H'_y(i,j)=\{\bar{v}\,|\,\bar{v}=\sigma_{ij}(\bar{u}),\ \bar{u}\in H_y(i,j)\}$;

(3) $\forall \bar{u}\ [\bar{u}\in H(i,j)\rightarrow B(\bar{u})\neq B(\sigma_{ij}(\bar{u}))]$;

(4) $\|H_1(i,j)\|=\|H_2(i,j)\|=\cdots=\|H_{x-1}(i,j)\|=p$, where for any set X, $\|X\|$ means the number of elements in X.

The characteristic 1 ensures that all data blocks in any of the x subsets of $H(i,j)$ are assigned to different processors, and characteristic 2 ensures the same thing for the corresponding x neighboring subsets in $H(i,(j \bmod n_i)+1)$. Characteristic 3 makes sure that any pair of neighboring data blocks along the ith dimension are assigned to different processors, so communications will occur between them when the program scans along the ith dimension, no matter whether forward or backward. These three characteristics together ensure two things. First, computations in both $H(i,j)$ and $H(i,(j \bmod n_i)+1)$ can be divided into x steps. During the kth step, $k\in N_x$, the data blocks in $H_k(i,j)$ or $H'_k(i,j)$ can be handled in parallel. Second, communications between $H(i,j)$ and $H(i,(j \bmod n_i)+1)$ can be divided into x steps too. During the kth step, $k\in N_x$, communications between $H_k(i,j)$ and $H'_k(i,j)$ are dealt with, and every processor sends and receives messages at most once each. This kind of communication scheme can disperse message passing and reduce network congestion as much as possible. Characteristic 4 makes sure that every $H_k(i,j)$ and its neighbor $H'_k(i,j)$, $k\in N_x$, contain as many data blocks as possible, which makes x equal to its minimum $\lceil NB_i/p\rceil$. The combination of these four characteristics has two properties. On the one hand, all data blocks are distributed as evenly as possible and every processor is kept as busy as possible. This property leads to minimized computing time, and we call it "load balance." On the other hand, the communications between any two adjacent slabs are accomplished in minimal steps, and in every step the number of processors involved in communication is maximized, while communications are dispersed as much as possible. This property can minimize communication time, and we call it "neighboring communication." In sum, the data distribution scheme satisfying these four characteristics can maximize parallelism and minimize communication simultaneously. In return, high parallel processing efficiency is guaranteed. The details of the method for distributing data blocks that our algorithm adopts will be discussed later. Taking a scan forward along the ith $(i\in N_3)$ dimension as an example, the scan could be evaluated by r_i computation phases and r_i communication phases in sequence. During the jth, $j\in Nr_i$, computation phase, data blocks in $H(i,j)$ are parallel processed in $\lceil NB_i/p\rceil$ steps as described before. Following the jth $(j\in Nr_i)$ computation phase is the jth $(j\in Nr_i)$ communication phase, in which only a boundary layer consisting of one or more planes of A is transmitted, and all communications are handled in $\lceil NB_i/p\rceil$ steps as described before. A similar situation occurs when scanning along the ith $(i\in N_3)$ dimension backward.

Now we describe how to find the optimal data partition in our algorithm. Suppose the dividing vector $\bar{r}$ is determined and a data distribution satisfying both the load balance and neighboring communication properties is applied, it is easy to deduce that T_{comp}, T_{comm}, and T can be calculated as

$$T_{\text{comp}} = 2K_{\text{comp}}V\sum_{i\in N_3}(\lceil NB_i/P\rceil/NB_i), \tag{11}$$

$$T_{\text{comm}} = 2K_{\text{comm}}\sum_{i\in N_3}[(V_iL_ir_i)\lceil NB_i/P\rceil/NB_i], \tag{12}$$

$$\begin{aligned}T &= T_{\text{comp}} + T_{\text{comm}}\\ &= 2\sum_{i\in N_3}[(K_{\text{comp}}V + K_{\text{comm}}V_iL_ir_i)\lceil NB_i/P\rceil/NB_i].\end{aligned} \tag{13}$$

Obviously, the optimal data partition is the one that can minimize T. In Eq. (13), K_{comp}, K_{comm}, V, V_i, and L_i are fixed, when A, p, and the cluster used for parallel processing are determined. Then $\bar{r}$ is the only factor that affects the performance, since NB_i and NB are both determined by $\bar{r}$. On the one hand, reducing ri, $i\in N_3$, can bring down communication cost and result in better performance. On the other hand, given any $i\in N_3$, the minimum of $\lceil NB_i/p\rceil/NB_i$, i.e., $1/p$, can be achieved when NB_i can be divided by p exactly, which means every processor is assigned the same number of data blocks for any slab along the ith dimension. Given any $i\in N_3$, if NB_i can be divided by p exactly, then we say there exists a "perfect load balance" along the ith dimension.

For a fixed p, it is subtle to find the optimal $\bar{r}$. From one point of view, suppose $i,j,k\in N_3\wedge i\neq j\wedge i\neq k\wedge j\neq k$; although reducing the values of r_j and r_k can lead to less communication cost so as to be helpful to performance, it can also bring down NB_i to a value less than p and make $\lceil NB_i/p\rceil/NB_i$ equal to $1/NB_i$, which is greater than the minimum, say, $1/P$; as another point of view, despite the fact that it is beneficial to performance to make sure that NB_1, NB_2, and NB_3 can all be divided by P evenly at the same time, this characteristic does not ensure a minimized communication. Thanks to the performance model expressed by Eq. (13), we can use a simple exhaustive enumeration method to find the optimal dividing vector. Suppose (r_1,r_2,r_3) is a dividing vector with $r_1\geq p\wedge r_2\geq p\wedge r_3\geq p$; it is easy to deduce that the performance of (r_1,r_2,r_3) is not superior to (p,p,p) according to Eq. (13). So the times of enumeration before finding the optimal dividing vector are not more than p^3, which is bearable considering that p is normally less than 10^4 nowadays. Actually, in our algorithm, we take only dividing vectors that ensure perfect load balance along two of three dimensions as candidates for the final dividing vector. This kind of searching method can be justified by the following three facts, even though it can shrink the searching space a lot so as to greatly reduce searching time. (1) Dividing vectors that can provide three-dimensional perfect load balance are good candidates undoubtedly. (2) Only considering dividing vectors with perfect load balance along all dimensions cannot ensure finding the optimal dividing vector. For example, suppose the processor number is 15 and A is a cube, we find that (3,5,15) is the best dividing vector among those possessing three-dimensional perfect load balance under the direction of Eq. (13). But as far as performance is concerned, (3,5,15) may not be superior to (3,5,3), which reduces communications dramatically at the cost of one-dimensional perfect load balance. (3) Dividing vectors with only one-dimensional perfect load balance or without any perfect load balance have poor parallelism. In a word, the dividing vectors with at least two-dimensional perfect load balance are good candidates for the final dividing vector.

When the final dividing vector is determined, we can assign data blocks shaped by the selected dividing vector to processors, which is equivalent to assigning every element of B a value belonging to N_p, as described previously. Given any (r_1,r_2,r_3) with $NB_i|p$ where $i\in N_3$, the algorithm (written in C programming language style; see Ref. [18]) can assign processors in a way satisfying the load balance and neighboring communication properties at the same time. The algorithm for any (r_1,r_2,r_3) with $NB_2|p$ or $NB_3|p$ is similar. The proof of the correctness of this algorithm is not difficult but lengthy, and we omit it due to space constraints. Taking the dividing vector (3,3,3) as an example, Fig. 2 shows the effect of processor assignment when the algorithm described in Ref. [18] is applied.

With the above data partition, a serial SCFT algorithm can be adapted to parallel processing. We now describe our parallel algorithm. First, an initial guess for the static field $\omega_\alpha(\mathbf{r})$ is obtained using a standard random number generator, and the initial value of the potential field $\xi(\mathbf{r})$ is set to $\xi(\mathbf{r}) =[\omega_A(\mathbf{r})+\omega_B(\mathbf{r})+\omega_C(\mathbf{r})]/3$. These processes of initializing fields are performed concurrently. Thereafter, using the Douglas-Gunn method and the alternating-direction implicit scheme [16], the modified diffusion equations of $q(\mathbf{r},s)$ and $q^\dagger(\mathbf{r},s)$ are solved on each processor with the given initial and boundary conditions. Certainly, interprocessor communication is required and the communication cost of this step occupies most of the whole communication time. But the communication of data between processors is only boundary data transfer. Then, the monomer densities $\phi_\alpha(\mathbf{r})$ conjugated to $\omega_\alpha(\mathbf{r})$ are evaluated with $q(\mathbf{r},s)$ and $q^\dagger(\mathbf{r},s)$. This step requires both local computation and interprocessor communication. Following the work of Drolet and Fredrickson [8,19], the potential fields $\omega_\alpha(\mathbf{r})$ and $\xi(\mathbf{r})$ are updated. Only local computation is implemented in this step. Through local computation and three reduction operations, the free energy of the system is obtained. With the new fields $\omega_\alpha(\mathbf{r})$ and $\xi(\mathbf{r})$, the procedure returns to calculate the polymer segment probability distribution function, and this is repeated until the free energy change at each iteration is reduced to 10^{-4}. To avoid any influence of the simulation cell size on the final morphology, each minimization of the free energy is further iterated with respect to the cell size to obtain the equilibrium structure.

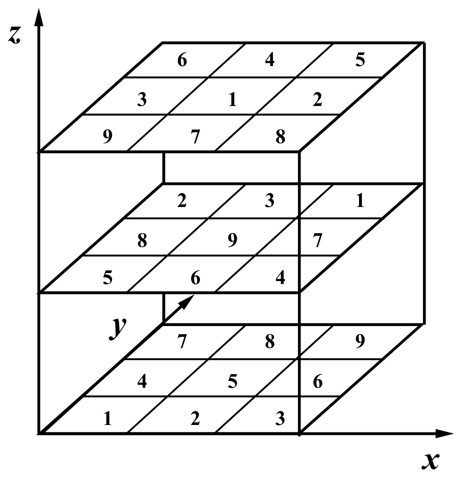


FIG. 2. Effect of processor assignment after the algorithm in Ref. [18] is applied. Each parallelogram indicates one data block, and the integer within each parallelogram indicates the rank of the processor that owns it. In this figure, x, y, and z dimension correspond to the first, second, and third dimension of the array to be partitioned.

In all the calculations, the chain length of the polymers is fixed to be $N=100$. The lattice units are chosen to be $dx =dy=a$, where a is the statistical length of the polymer segment and the discretization parameter along the chain contour length Δs is set to a.

IV. RESULTS AND DISCUSSION

A. Performance of the parallel algorithm

The validity and performance of the parallel algorithm were tested on a Beowulf cluster at the Supercomputing Center of Fudan University. After massive tests, we confirmed that the error of numerical results [i.e., $\phi_\alpha(\mathbf{r})$, $\omega_\alpha(\mathbf{r})$, $q(\mathbf{r},s)$, and $q^\dagger(\mathbf{r},s)$] between the serial and parallel algorithms is less than 10^{-9}. Hence, the parallel algorithm is valid and accurate. The performance of the parallel SCFT algorithm is illustrated in Fig. 3 for 3D systems. The speedup is the CPU time ratio of the parallel algorithm to the serial algorithm. CPU time includes both the computation and communication time. The tests are performed on four system sizes, 32^3, 48^3, 64^3, and 80^3. Ideally, the efficiency of a parallel algorithm is 1.0 and the speedup is equal to the number of CPUs used, which is denoted by the open stars in Fig. 3. The larger the size of a simulation cell, the better the efficiency of the parallel algorithm. For the larger simulation cells (64^3 and 80^3), our algorithm can obtain the ideal speedup state for eight CPUs. As shown in Fig. 3, the speedup increases continuously with

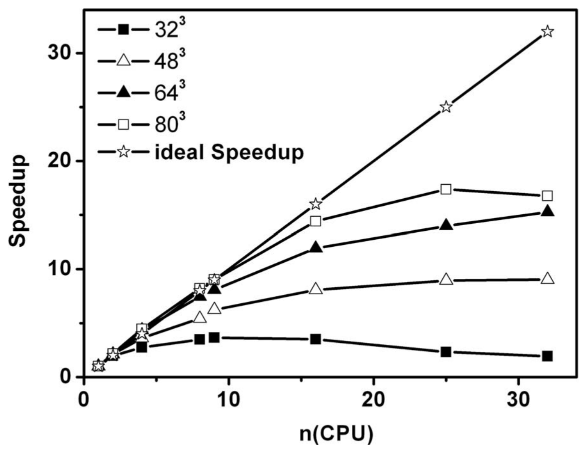


FIG. 3. Performance of parallel algorithm. Speedup is the CPU time ratio of the parallel to the serial algorithm. CPU time includes both the computation and communication time. The computation is performed in three-dimensional space and the system size is 32^3, 48^3, 64^3, or 80^3.

increasing CPU number. When the CPU number increases to a critical number, the speedup barely changes with further increase in the CPU number and the efficiency becomes worse. The critical number is different for different sizes of the simulation cell. Finally, the algorithm performs poorly in some cases. Meanwhile, the performance is also different with different sizes of the cell. For smaller systems (i.e., 32^3), the performance is degraded when the CPU number is more than 16. For the larger simulation cells (i.e., 64^3 and 80^3), our parallel algorithm is more efficient. The computation CPU time decreases with increasing CPU number. But the proportion of communication CPU time increases simultaneously. Hence, there exists an optimum ratio of communication-to-computation CPU time for a specific calculation system. As shown in Fig. 3, there is an optimum CPU number for a special simulation cell. For example, the optimum CPU number is about nine for the smaller system (32^3). In addition, our algorithm avoids the main drawback of an algorithm based on the FFT when applied to arrays where the lengths of all dimensions are not highly composite, because of the nature of the FFT. This characteristic is important to search for new microphase structures of block copolymers in a real-space simulation, since the system size has to be continuously varied to avoid its influence on the final morphology.

B. Morphology of *ABC* linear triblock copolymer with $f_A=f_C$

ABC triblock copolymers offer the potential to create nanoscale morphologies with interesting and useful chemical and physical properties. However, cataloging the expensive *ABC* parameter space, relative to *AB* diblocks, presents a daunting task for experimentalists and theoreticians alike. We hope this situation can be improved by the application of our parallel algorithm. As a benchmark test, in this paper, we report only a special case in which the volume fractions of the two end blocks of the *ABC* linear triblock copolymer are equal ($f_A=f_C$). We focus on the variation of morphology

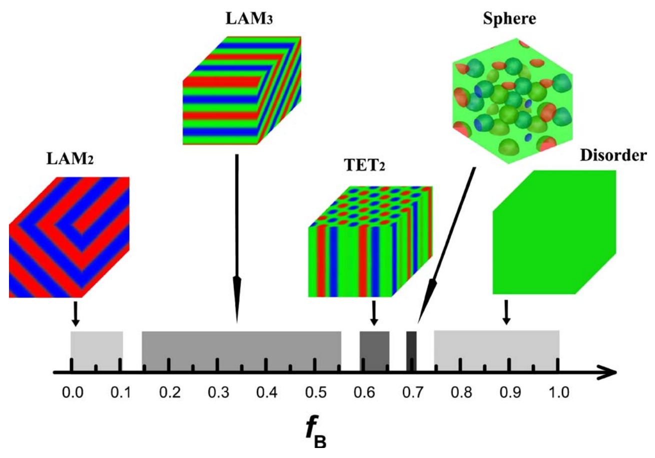


FIG. 4. (Color online) One-dimensional phase diagram for f_B. The chain length $N=100$ and the Flory-Huggins interaction parameters are set to be $\chi_{AB}N=\chi_{BC}N=\chi_{AC}N=35$, $f_A=f_C$. Three different colors, blue (black), green (light gray), and red (dark gray), are assigned to *A*, *B*, and *C* blocks, respectively.

with volume fraction of the middle block *B* (f_B). For large simulation cells, a globally uniform structure is difficult to obtain, and the defect forms easily and is difficult to remove. Hence, we choose smaller system sizes (26^3-32^3) to demonstrate the ability of the parallel algorithm.

First, we define two ratios of the interaction parameters as $R_1=\chi_{AB}/\chi_{AC}$ and $R_2=\chi_{BC}/\chi_{AC}$, respectively. Then the *ABC* linear triblock copolymers can be classified into four different classes according to the relative strengths of the interaction energies: (1) $R_1=R_2=1$, (2) $R_1>1$, $R_2\geq 1$, (3) $R_1<1$, $R_2\leq 1$, and (4) $R_1<1$, $R_2>1$. If the binary interaction parameters χ_{AB}, χ_{BC}, and χ_{AC} are different from each other, different sequences of blocks (*A-B-C*, *B-C-A*, *C-A-B*) will lead to different phase behaviors for the system, even with the same composition parameters. Once the relative strength of the interaction parameters is specified, the morphology is uniquely determined by the composition of the system. Hence, we can facilitate examining the influence of the sequences and relative strengths of the interaction energy using the ratios of the interaction parameters.

1. $R_1=R_2=1$

For this case, the morphology mainly depends on copolymer compositions due to equal binary interactions between each block. In our case, the volume fraction of the middle block (f_B) is a key factor in determining the morphology. We set $\chi_{AB}N=\chi_{AC}N=\chi_{BC}N=35$, which is in the experimentally interesting intermediate segregation regime. The one-dimensional phase diagram for varying f_B is shown in Fig. 4. In the calculation of the phase diagram, the increment of f_B is set to 0.01 near the phase boundary and is 0.05 in other cases.

As shown in Fig. 4, with the volume fraction of the middle block (f_B) increasing from 0.0 to 1.0, the following ordered morphologies appear successively: two-domain lamellae (LAM_2)→three-domain lamellae (LAM_3)→two interpenetrating tetragonally arranged cylinder phases (TET_2)

$\rightarrow$spherical domains in the CsCl-type structure (sphere) $\rightarrow$disorder. The same morphologies have been reported in previous experimental studies [20–22] and theoretical predictions [23,24]. When $f_B=0$, the triblock copolymer (ABC) is reduced to the diblock copolymer (AC) with $f_A=f_C$ in our case. The morphology is always LAM$_2$ with the same domain sizes of the two species ($D_A=D_C$). As the volume fraction f_B increases, but $f_B \leq 0.1$, most of the middle blocks B are enriched at the interfaces between the two end blocks (A and C). However, for $f_A \leq 0.1$ or $f_C \leq 0.1$, the minority component A or C dissolves in the lamellae of the middle block B in the same system conditions [11]. The two ends of the middle block B are connected with either end of the block A or C. Therefore, the middle block B can only distribute near the interfaces of A/C. When $0.1 < f_B < 0.5$, the LAM$_3$ phase is obtained. The middle block B forms a single domain with its size increasing with f_B. For a symmetric triblock copolymer ($f_A \approx f_B \approx f_C$), the LAM$_3$ phase with two characteristic lamellar widths is observed, i.e., $D_A \approx D_C \approx 2D_B$. Tang *et al.* also found the same morphology with the real-space method of the SCFT in 2D [11]. The same morphology was also obtained by Matsen using the Fourier space implementation of SCFT [24] and Zheng and Wang using the strong segregation theory [25]. When f_B increases to 0.60, the TET$_2$ morphology appears for the linear ABC block copolymer instead of the LAM$_3$. This stable phase is composed of A and C cylinders tetragonally arranged within the B matrix and there are no internal A/C interfaces. This structure arises from the special characteristics of triblock copolymers. A similar structure was found in experimental [20,22,27] and theoretical results [24]. As f_B further increases, the system exhibits the spherical phase which contains spherical domains in the CsCl-type structure. Similarly to the TET$_2$ phase, the A-rich and C-rich spheres must be placed close together because the B middle block has to bridge them. The same morphology was also predicted by Zheng and Wang using the strong segregation theory [26]. In addition, a hexagonal lattice phase (hex) is also a stable phase in Tang *et al.*'s work [11]. However, it is not found in our case; it may be a dimensional-dependent structure.

2. $R_1 > 1$, $R_2 \geq 1$

In this case, interactions between the end and middle blocks (A-B and B-C) are more unfavorable than those between the two end blocks (A-C), thus it is possible for A/C interfaces to form.

a. $\chi_{AB}N=50$, $\chi_{AC}N=20$, $\chi_{BC}N=50$. In Fig. 5, the one-dimensional phase diagram with varying f_B is shown for $\chi_{AB}N=50$, $\chi_{AC}N=20$, and $\chi_{BC}N=50$ and thus $R_1=2.5$ and $R_2=2.5$. As shown in Fig. 5, with the volume fraction of the middle block (f_B) increasing from 0.0 to 1.0, the following ordered morphologies appear successively: LAM$_2$$\rightarrow$perforated lamellae (PL)$\rightarrow$LAM$_3$$\rightarrow$network$\rightarrow$sphere$\rightarrow$disorder.

For middle blocks B forming the minority species ($f_B \leq 0.1$), most of them are enriched at the interfaces between the two majority components (A and C blocks). The LAM$_2$ phase is formed when $D_A=D_C$. When $f_B > 0.1$, B blocks form a single domain. With $\chi_{AB} \approx \chi_{BC} \gg \chi_{AC}$, it is possible for the A/C interface to be formed, although there is no

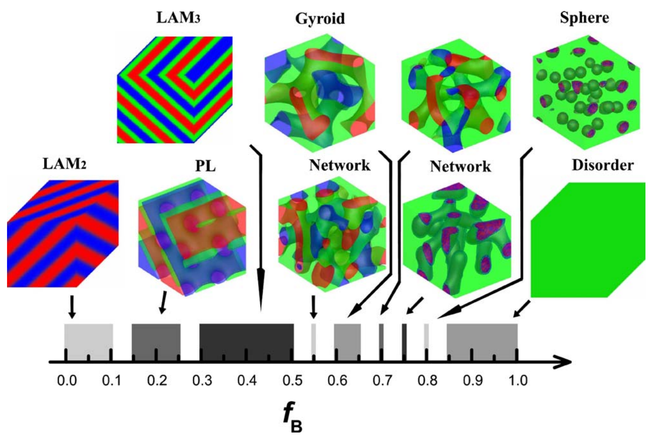


FIG. 5. (Color online) One-dimensional phase diagram for f_B. The chain length $N=100$ and the Flory-Huggins interaction parameters are set to be $\chi_{AB}N=50$, $\chi_{AC}N=20$, $\chi_{BC}N=50$, $f_A=f_C$. Three different colors, blue (black), green (light gray), and red (dark gray), are assigned to A, B, and C blocks, respectively.

chemical junction between A and C blocks. Therefore, perforated lamellae appear instead of LAM$_3$ within a certain range of f_B ($0.15 < f_B < 0.25$). As f_B further increases, the system is trapped in LAM$_3$. When $f_B > 0.55$, the end block (A or C) forms a network structure within the B matrix and thus the interpenetration network structure appears. As shown in Fig. 5, a double-gyroid network occurs about $f_B = 0.6$. In our case, the two end blocks (A and C) can mix together forming a domain in the B matrix when $f_B > 0.7$. A single network structure appears about $f_B=0.75$. Meanwhile, near $f_B=0.8$, the spherical phase is formed by mixed A and C blocks. When $f_B > 0.85$, the disordered state is stable.

b. $\chi_{AB}N=60$, $\chi_{AC}N=20$, $\chi_{BC}N=60$. The interaction energy χN is significant in determining the morphology of triblock copolymers in the weak and intermediate segregation regions [25]. In this section, both $\chi_{AB}N$ and $\chi_{BC}N$ are increased from 50 to 60 to study the influence of χN on the morphology.

As shown in Fig. 6, the morphology becomes more interesting and fascinating. With the volume fraction of the middle block (f_B) increased from 0.0 to 1.0, the following ordered morphologies appear successively: LAM$_2$$\rightarrow$lamellar phase with beads at the interface (LAM+BD)$\rightarrow$lamellar phase with cylinders at the interface (LAM+C)$\rightarrow$PL $\rightarrow$LAM$_3$$\rightarrow$network$\rightarrow$sphere$\rightarrow$disorder.

Compared to Fig. 5 (with lower $\chi_{AB}N$ and $\chi_{BC}N$), LAM +BD and LAM+C occur between the LAM$_2$ and PL phases. When f_B is about 0.1, the B block forms spherical domains with hexagonal arrangement located at the A/C interfaces. While f_B increases to 0.12, the system tends to exhibit another morphology, A and C lamellae with cylinder domains formed by B blocks located at the A/C interfaces. Stadler *et al.* found these two structures in triblock poly(styrene-*b*-butadiene-*b*-methylmethacrylate) (PS-PB-PMMA) [28–31]. In this case, the interactions between the end and middle blocks (PS-PB and PB-PMMA) are more unfavorable than that between the two end blocks (PS-PMMA), which falls into the class we investigated here. When the content of PB comes to

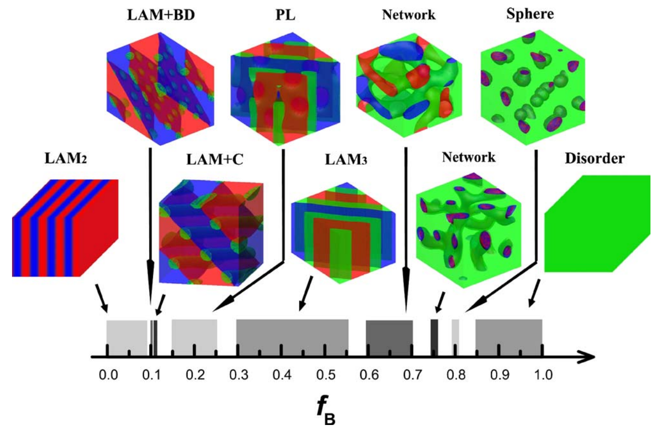


FIG. 6. (Color online) One-dimensional phase diagram for f_B. The chain length $N=100$ and the Flory-Huggins interaction parameters are set to be $\chi_{AB}N=60$, $\chi_{AC}N=20$, $\chi_{BC}N=60$, $f_A=f_C$. Three different colors, blue (black), green (light gray), and red (dark gray), are assigned to A, B, and C blocks, respectively.

6 wt %, the LAM+BD phase was obtained. As the content of PB increased to 17 wt %, the LAM+C phase was the stable structure. In addition, the two morphologies were also predicted by Ko *et al.* using molecular dynamics simulation [32]. In our case, LAM+BD and LAM+C exist in a very limited region. One can expect these two morphologies to exist in a larger region when $\chi_{AB}N$ and $\chi_{BC}N$ further increase; in addition, the range of the stable, well-separated LAM$_3$ phase is broadening.

3. $R_1<1$, $R_2\leq 1$

In this case, we set $\chi_{AB}N=20$, $\chi_{AC}N=50$, $\chi_{BC}N=50$. Therefore, $R_1=0.4$ and $R_2=1.0$. The interaction energy between the two blocks A and B is more favorable than that of A-C or B-C. Then it is unfavorable for the A/C interface to form.

As shown in Fig. 7, with the volume fraction of the middle block (f_B) increasing, the following ordered morphologies appear successively: LAM$_2\rightarrow$LAM$_3\rightarrow$PL$\rightarrow$TET$_2$ $\rightarrow$spherical domains in the CsCl-type structure (sphere) $\rightarrow$spherical domains formed by C blocks (sphere)$\rightarrow$disorder.

The triblock copolymer in Fig. 7 can be obtained by simply changing the block sequence of the triblock copolymer from A-B-C to A-C-B in Fig. 5. The different phase behaviors observed in Figs. 5 and 7 illustrate the effect of the block sequence in linear triblock copolymers when the interaction energies are not equal. Compared to Fig. 5, the LAM$_3$ morphology appears for the linear ABC block copolymer instead of the PL phase when the volume fraction of the B block is in the range of 0.15–0.25. As f_B further increases, B blocks tend to form a continuous phase. When f_B is about 0.55, a special perforated lamellae phase occurs, in which the cross section perpendicular to the lamellae is similar to the morphology of lamellae with beads inside (LAM+BD-I) in 2D predicted by Tang *et al.* [11]. The domain of the C block is perforated by those of A and B blocks. Because it is unfavorable for the A/C interface to form, the A domain is surrounded by the B

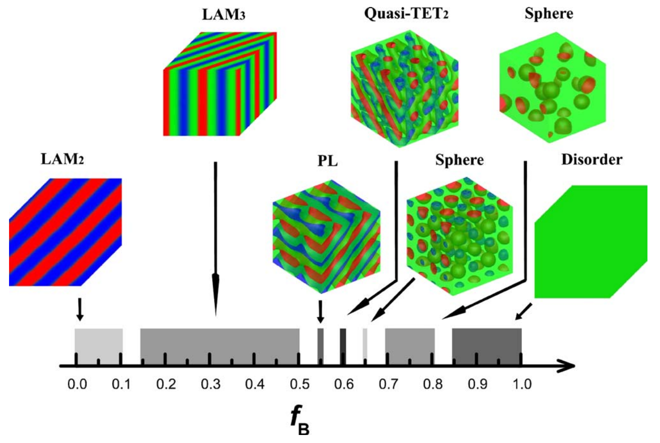


FIG. 7. (Color online) One-dimensional phase diagram for f_B. The chain length $N=100$ and the Flory-Huggins interaction parameters are set to be $\chi_{AB}N=20$, $\chi_{AC}N=50$, $\chi_{BC}N=50$, $f_A=f_C$. Three different colors, blue (black), green (light gray), and red (dark gray), are assigned to A, B, and C blocks, respectively.

domain. Zheng and Wang [26] and Tang *et al.* [11] also predicted a core-shell hexagonal (CSH) phase in a similar system. The CSH phase was observed in a poly(styrene-*b*-isoprene-*b*-2-vinyl-pyridine) melt [33]. However, this phase is not found in our case. In their cases, the A-C interaction is slightly larger than either A-B or B-C interactions and the system tends to form the CSH phase with no A/C interface because it is more favorable energetically. In our case, $\chi_{AC}N=\chi_{BC}N=50$ and our system is between F^0 and F^1 (F^0 and F^1 are defined by Tyler *et al.* in [34]). Hence, the CSH phase is replaced by the special perforated lamellae phase because the curvature energy is unfavorable.

When f_B is equal to 0.60, the quasi-TET$_2$ morphology is obtained. This structure arises from the special characteristics of triblock copolymers and avoids the A/C interface. The spherical domains in the CsCl-type structure occur when f_B increases to 0.65. As f_B further increases, the A block mixes into the B matrix and the C block forms spherical domains arranged in the B matrix. When $f_B>0.85$, a disordered phase is obtained.

4. $R_1<1$, $R_2>1$

Figure 8 presents the one-dimensional phase diagram for $\chi_{AB}N=20$, $\chi_{AC}N=40$, and $\chi_{BC}N=60$, i.e., $R_1=0.5$ and $R_2=1.5$. The system prefers the morphology that reduces the unfavorable contacts between the B and C blocks as well as increases the interfacial contacts between the A and B blocks. Compared to the case of $R_1<1$, $R_2\leq 1$, the values of the interaction energy of these two cases are close. Therefore, the sequence of the ordered morphologies in Fig. 8 is the same as that in Fig. 7; however, the same morphology occurs with different f_B. For example, the special PL phase appears near $f_B=0.5$ and TET$_2$ is formed in a larger region of f_B. In order to decrease the interfacial energy between the two end blocks (A and C), the special PL phase is favorable. Hence, we believe it is a kind of universal morphology in triblock copolymers with similar interaction parameters.

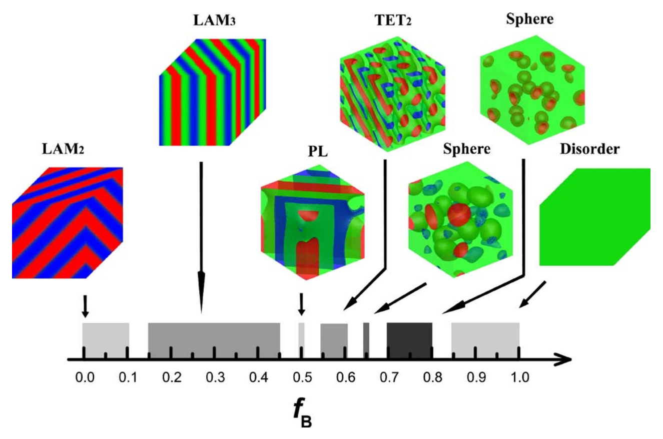


FIG. 8. (Color online) One-dimensional phase diagram for f_B. The chain length $N=100$ and the Flory-Huggins interaction parameters are set to be $\chi_{AB}N=20$, $\chi_{AC}N=40$, $\chi_{BC}N=60$, $f_A=f_C$. Three different colors, blue (black), green (light gray), and red (dark gray), are assigned to A, B, and C blocks, respectively.

V. CONCLUSIONS

We have developed a parallel algorithm for real-space SCFT simulation. The parallel algorithm is based on the Douglas-Gunn scheme, which is adopted in the numerical algorithm of SCFT to solve the modified diffusion equations. Our algorithm is also suitable for other implicit schemes, such as the Crank-Nicholson scheme, Dufort-Frankel scheme, etc. For the widely used parallel clusters with distributed memory, the parallel algorithm is efficient, especially for larger simulation cells. Our algorithm avoids the main drawback of an algorithm based on the FFT when applied to arrays where the lengths of all dimensions are not highly composite, because of the nature of the FFT. Therefore the simulation cell size can be continuously varied in the present simulation, which is important in searching for new microphase structures of complex block copolymers.

As a benchmark test, the special case of an *ABC* linear triblock copolymer with equal volume fractions of the two end blocks ($f_A=f_C$) was investigated by the present parallel SCFT simulation. The ordered phases of the *ABC* linear triblock copolymer depend not only on the composition and the degree of segregation, but also on the sequence of the blocks. Compared to previous serial SCFT algorithms, which are limited to studying 2D structures, the parallel SCFT gives rise to intrinsic 3D structures such as perforated lamellae and network phases. We expect the parallel algorithm developed in this paper to boost the discovery of novel assembly structures in complex block copolymers.

ACKNOWLEDGMENTS

We thank Dr. Kunkun Guo, Dr. Jianfeng Xia, Dr. Jiajing Xu, and Dr. Rongtang Fu for useful discussions. We gratefully acknowledge financial support from the National Basic Research Program of China (Grant No. 2005CB623800), the National Natural Science Foundation of China (Grants No. 20625413, No. 20221402, and No. 20490220), and the NCET Program of the Ministry of Education of China. We also acknowledge support from the Supercomputing Center of Fudan University.

[1] F. S. Bates and G. H. Fredrickson, Phys. Today **52**(2), 32 (1999).
[2] I. W. Hamley, *The Physics of Block Copolymers* (Oxford University Press, Oxford, 1998).
[3] I. W. Hamley, *Developments in Block Copolymer Science and Technology* (John Wiley & Sons, 2004).
[4] S. F. Edwards, Proc. Phys. Soc. London **85**, 613 (1965).
[5] E. Helfand, J. Chem. Phys. **62**, 999 (1975).
[6] E. Helfand, Macromolecules **8**, 552 (1975).
[7] M. W. Matsen and M. Schick, Phys. Rev. Lett. **72**, 2660 (1994).
[8] F. Drolet and G. H. Fredrickson, Phys. Rev. Lett. **83**, 4317 (1999).
[9] G. Tzeremes, K. O. Rasmussen, T. Lookman, and A. Saxena, Phys. Rev. E **65**, 041806 (2002).
[10] S. W. Sides and G. H. Fredrickson, Polymer **44**, 5859 (2003).
[11] P. Tang, F. Qiu, H. D. Zhang, and Y. Yang, Phys. Rev. E **69**, 031803 (2004).
[12] P. Tang *et al.*, J. Phys. Chem. B **108**, 8434 (2004).
[13] P. G. Ferreira, A. Ajdari, and L. Leibler, Macromolecules **31**, 3994 (1998).
[14] R. Wang *et al.*, J. Chem. Phys. **118**, 9447 (2003).
[15] J. W. Thomas, *Numerical Partial Differential Equations* (Springer, New York, 1995).
[16] W. H. Press *et al.*, *Numerical Recipes* (Cambridge University Press, Cambridge, England, 1989).
[17] A. Darte *et al.*, J. Parallel Distrib. Comput. **63**, 887 (2003).
[18] See EPAPS Document No. E-PLEEE8-76-064712 for program for processor assignment. For more information on EPAPS, see http://www.aip.org/pubservs/epaps.html.
[19] G. H. Fredrickson, V. Ganesan, and F. Drolet, Macromolecules **35**, 16 (2002).
[20] Y. Mogi *et al.*, Macromolecules **25**, 5408 (1992).
[21] Y. Mogi *et al.*, Macromolecules **25**, 5412 (1992).
[22] Y. Mogi *et al.*, Macromolecules **27**, 6755 (1994).
[23] F. Drolet and G. H. Fredrickson, Macromolecules **34**, 5317 (2001).
[24] H. Nakazawa and T. Ohta, Macromolecules **26**, 5503 (1993).
[25] M. W. Matsen, J. Chem. Phys. **108**, 785 (1998).
[26] W. Zheng and Z. G. Wang, Macromolecules **28**, 7215 (1995).
[27] K. Jung, V. Abetz, and R. Stadler, Macromolecules **29**, 1076 (1996).
[28] R. Stadler *et al.*, Macromolecules **28**, 3080 (1995).
[29] J. Beckmann, C. Auschra, and R. Stadler, Macromol. Rapid Commun. **15**, 67 (1994).
[30] U. Breiner *et al.*, Abstr. Pap. - Am. Chem. Soc. **210**, 57 (1995).
[31] U. Breiner, U. Krappe, and R. Stadler, Macromol. Rapid Commun. **17**, 567 (1996).
[32] M. J. Ko, S. H. Kim, and W. H. Jo, Macromol. Theory Simul. **10**, 381 (2001).
[33] S. P. Gido *et al.*, Macromolecules **26**, 2636 (1993).
[34] C. A. Tyler *et al.*, Macromolecules **40**, 4654 (2007).