DNA electrophoresis studied with the cage model

A. van Heukelum and G. T. Barkema

Institute for Theoretical Physics, Utrecht University,
Princetonplein 5, 3584 CC Utrecht, The Netherlands

and

R. H. Bisseling

Mathematical Institute, Utrecht University,
PO Box 80010, 3508 TA Utrecht, The Netherlands

Key Words: DNA, electrophoresis, cage model, reptation, sparse matrix, BSP, parallel matrix-vector multiplication, band collapse, diffusion coefficient

The cage model for polymer reptation, proposed by Evans and Edwards, and its recent extension to model DNA electrophoresis, are studied by numerically exact computation of the drift velocities for polymers with a length L of up to 15 monomers. In agreement with De Gennes' reptation arguments, we find that asymptotically for large polymers the diffusion coefficient D decreases quadratically with polymer length; for the cage model, the proportionality coefficient is $DL^2 = 0.175(2)$. Additionally we find that the leading correction term for finite polymer lengths scales as $N^{-1/2}$, where $N = L - 1$ is the number of bonds. If an electric field is applied with strength E, the polymer drift velocity v scales initially as $v \sim E/L$, and then changes into a regime where $v \sim E^2$, as in the Duke-Rubinstein model. We find confirmation for the existence of a third regime, still in the physically relevant range of field strengths, where the drift velocity decreases exponentially.

1. INTRODUCTION

In the rapidly-growing fields of molecular genetics and genetic engineering, gel electrophoresis is a technique of great importance. One reason is that it enables efficient separation of polymer strands by length. In DNA electrophoresis, strands of DNA with a variety of lengths are injected into a gel composed of agarose and a buffer solution. Since DNA is acidic, it becomes negatively charged. Next, an electric field is applied which causes the DNA to migrate in one direction. Since shorter strands travel faster than longer ones, the initial mixture of strands will become separated, allowing the measurement of the relative concentrations of strands of different lengths, or the isolation of strands with a particular length. Given the great practical importance of DNA electrophoresis, there is much interest in gaining an understanding of precisely what the mechanisms of gel electrophoresis are and how the migration rate depends on strand length, applied electric field, and the properties of the agarose gel.

It is known that in the gel, agarose forms long strands which cross-link and impede movement of the polymer transverse to its length; its movement is dominated by a mechanism which De Gennes [1] has dubbed *reptation*: movement of a polymer along its own length by diffusion of stored length.

A commonly used lattice model to simulate the dynamics of reptation is the so-called "repton model", introduced by Rubinstein in 1987 [2]. Rubinstein already conjectured that the diffusion coefficient D as a function of polymer length L for long polymers is given by $DL^2 = 1/3$ (with large finite-size effects); this conjecture was further corroborated by van Leeuwen and Kooiman [3, 4, 5], and finally proven by Prähofer and Spohn [6]. The repton model has been extended to study electrophoresis by Duke [7, 8, 9], and the resulting model — known as the Duke-Rubinstein model — has been studied numerically and analytically by several groups [10, 11, 12, 13, 14] and compared to experiments [15].

The main findings of these studies are that the property $DL^2 = 1/3$ (with large finite-size effects) in combination with the fluctuation-dissipation theorem results in a drift velocity $v \sim E/L$ for small electric field strength E; and that for some value of $E \sim 1/L$ this regime crosses over in a regime where the drift velocity ceases to be length-dependent and is given by $v \sim E^2$, the so-called *band collapse*. Furthermore, it was found a property of the model in the limit of large E, that the drift velocity decays exponentially, $v \sim e^{(2-L)E/2}$ or $v \sim e^{(3-L)E/2}$ for even and odd L respectively [16].

Before the introduction of the repton model, Evans and Edwards had introduced the so-called "cage model" to simulate the dynamics of reptation [17]. Also in this model, DL^2 approaches a constant in the limit of large chains [18], which in combination with the fluctuation-dissipation theorem leads to $v \sim E/L$ for small electric field strengths. This model has recently been extended to electrophoresis and studied with Monte Carlo simulations [19]. Besides the expected fluctuation-dissipation regime, these simulations also featured the band-collapse regime where $v \sim E^2$; these are the two regimes that were identified for the Duke-Rubinstein model. Additionally, a third regime was reported where v decreases with increasing E, but still in the physical range $E \ll 1$. The identification of this regime might possibly shed more light on the hitherto not well understood phenomenon of *band inversion* [20], where longer polymers travel faster than shorter ones.

This article presents numerically exact computations on the cage model, extended for electrophoresis as in Ref. [19]. As in most models, numerically exact results can only be obtained for relatively small systems (here, for polymers up to a length of $L = 15$), but they do not have the inherently large statistical errors of Monte Carlo results. Thus, they allow for a different class of analysis techniques, for instance those exploiting numerical differentiation. The combination of numerically exact results for short chains with the Monte Carlo results for larger chains reported in Refs. [18, 19] provides a more complete picture of the model, and allows us to explore the nature of the third regime mentioned above (see section 5).

The calculations done in this work, with a chain length of up to $L = 15$, are computationally challenging, and could only be obtained by the exploitation of symmetries in the model, combined with the application of parallel processing. The state vector of the cage model for electrophoresis has 6^{L-1} components, and the original transition matrix which represents the transition probabilities between polymer configurations is of size $6^{L-1} \times 6^{L-1}$ (see section 2). We show that many components of the steady state vector are equal, because the configurations they belong to are equivalent. By using those equivalences, the original transition matrix could be reduced significantly (see section 3). The parallel implementation of the computation is done by spreading the nonzeros of the sparse transition matrix over the processors. Interprocessor communication is reduced by exploiting the specific sparsity structure of the matrix (see section 4). The combined effect of the decrease in matrix size and the application of parallel processing accelerates the computation by more than a factor of a million, allowing us to reach larger values of L; in this paper we present numerically exact values for the drift velocities and the diffusion coefficients for polymers up to length $L = 15$ (see section 5). We also present computation times and parallel efficiency results for up to 64 processors of a Cray T3E computer (see section 6). The conclusions are summarized in section 7.

2. CAGE MODEL FOR REPTATION

The cage model, introduced by Evans and Edwards [17], describes a polymer that moves through a gel. The polymer is modeled as a chain of “monomers”, connected by *bonds*. Two monomers connected by a bond must reside in adjacent sites of a cubic lattice. No other excluded volume interactions are enforced, so each lattice site may contain many monomers. Figure 1 shows an impression of the model. The configuration of a cage polymer is easiest defined by the set of directions of all the bonds. The bond representations shown in Figure 2 are **-y +x +y +y +y -y -y +x -x -y +x** and **-y +x -x +x -y -y -x +y +x +x -x**. A part of a configuration consisting of a monomer with two opposite bonds is called a *kink*. In the left part of Figure 2, the configurations feature kinks at monomers 5 and 8, and in the right part at 2, 3, and 10.

The gel is modeled by the edges of a cubic lattice, translated by a vector $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2})$ relative to the lattice on which the polymer resides. The dynamics of a cage polymer consists of those single monomer moves for which the polymer does not cut gel strands. This leaves two classes of allowed moves: (i) a kink is randomly replaced by a kink in one of the six possible directions; (ii) a bond at an end monomer is randomly replaced by a bond in one of the six possible directions. Every other single monomer move is forbidden because it would cause the polymer to cross a gel strand. The time increment associated with an attempted move is $\delta t = (6L)^{-1}$, such that each move is tried, on average, once per unit of time. The diffusion coefficient can be computed from the displacement of the center of mass.

The cage model has been extended to include the effects of an electric field on the motion of (charged) polymers [19]. The possible transitions are the same, but the rates are different. The electric field is $\vec{E} = (E, E, E)$, such that replacing a kink by one of the three forward pointing kinks (along the electric field) occurs with rate e^{qE}, and by one of the three backward pointing kinks (against the electric field) occurs with rate e^{-qE}, where q is the dimensionless charge of the moved monomer.

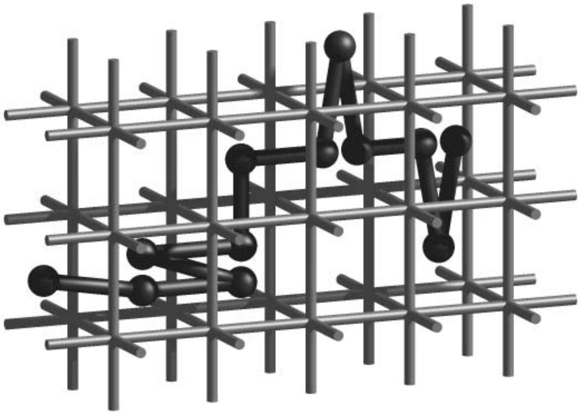

FIG. 1. An impression of the cage model for reptation. The polymer consists of a sequence of monomers, connected by unit-length bonds.

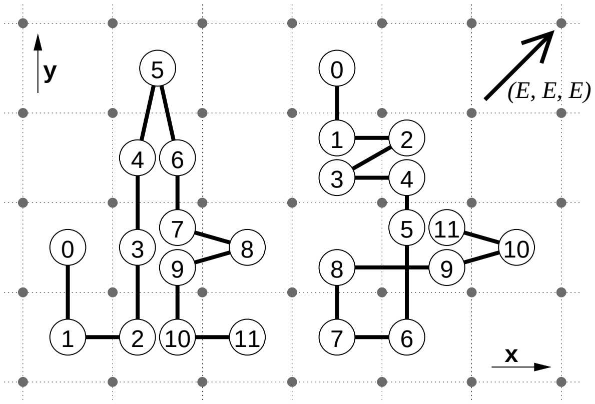

FIG. 2. The cage model. The dotted lines denote the gel strands in the x-y plane, and the large grey dots are the gel strands in the z direction. The space between the gel strands represents the pores of the gel. The polymer is modeled as a chain of monomers; two adjacent monomers reside in nearest neighbor pores. We denote bonds that are going right, left, up, down, out of the paper and into the paper by `+x`, `-x`, `+y`, `-y`, `+z`, and `-z` respectively. The two example configurations were chosen to be planar, for clarity. The electric field vector points diagonally out of the paper.

In the remainder of this paper, we will assume that $q = 1$. The time increment associated with an attempted move is $\delta t = \big(3L(e^{E} + e^{-E})\big)^{-1}$.

The set of all probabilities of the 6^{L-1} possible configurations can be represented by a 6^{L-1}-dimensional vector $\vec{a}$. The dynamics of the model is then specified by a sparse $6^{L-1} \times 6^{L-1}$ matrix T. The transition matrix T has $\big[5(\frac{L-2}{6} + 2) + 1\big]6^{L-1}$ nonzero elements: each polymer has $L-2$ inner monomers that can move if their bonds are in opposite directions, and two end monomers that can always move; a monomer that can move goes either to one of five new positions or the polymer stays unaltered. The steady state vector $\vec{a}_{\mathrm{steady}}$ is computed from T as the eigenvector with eigenvalue 1, and the drift velocity of the polymer along one of the principal axes is

$$v = \frac{2}{3}\sum_{i} a_i \big(b_i e^{E} - f_i e^{-E}\big), \tag{1}$$

where b_i is the number of kinks and endpoints of polymer configuration i pointing backward (which can move forward with a probability e^{E}), and f_i the number of kinks and endpoints pointing forward. The factor of 2/3 appears because moves occur along each of the three principal axes, and because each kink move increases or decreases the sum of the coordinates of a configuration by two.

3. EXPLOITING SYMMETRIES OF THE MODEL

In the model that we study here, the electric field is chosen in the $(1,1,1)$ direction, and consequently polymer configurations that are related through rotation around the direction $(1,1,1)$ are equivalent, i.e., their probability is the same, irrespective of the field strength. Moreover, in many cases it is possible to rotate *part* of the polymer around this direction while preserving this equivalence. If polymer configurations are grouped into classes containing only equivalent polymers, it is sufficient to determine the probability for one representative per class rather than for all polymer configurations, since by definition the probabilities are equal within a class.

Rather than working in the state space of all polymer configurations, we work in the state space of all equivalence classes. Since equivalence classes can easily contain thousands of configurations, the state space is thus reduced by several orders of magnitude, and a tremendous speed-up is obtained. Next, we will discuss how physical quantities such as the velocity can be computed within this reduced state space of equivalence classes, and how to identify whether two polymer configurations are equivalent.

Suppose that equivalence class i contains n_i polymer configurations, each with b'_i backward pointing kinks and end monomers, f'_i forward pointing kinks and end monomers; and that the probability for each of the configurations within class i is given by a'_i. The average drift velocity is then given by

$$v = \frac{2}{3}\sum_{i} n_i a'_i \big(b'_i e^{E} - f'_i e^{-E}\big). \tag{2}$$

To identify which polymer configurations are equivalent, we construct a representation that puts equivalent configurations in the same class. We call part of a

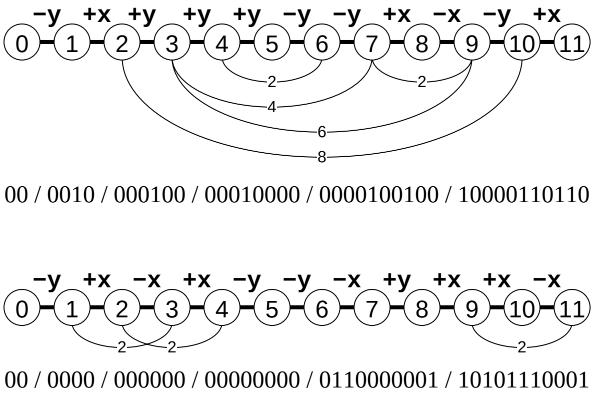

FIG. 3. Kink representations for the two examples from Figure 2. The arcs show which parts can be removed by repeatedly removing kinks. The kink representations are also given as a binary value; the slashes separate the removable parts of length 10, 8, 6, 4, 2, and the bond directions. A bit 1 at position r for part length l means that the part between monomers r and $r + l$ can be removed.

configuration between two monomers *removable*, if all monomers in between them can be removed by repeatedly removing kinks. A kink is removed by deleting the central monomer, the two bonds connected to it, and merging the two monomers adjacent to the central monomer. In the left part of Figure 2, monomers 4 and 6 are merged when the kink at monomer 5 is removed. If two polymers have the same sequence of forward and backward bonds, and the same set of removable pairs of monomers, then their probabilities are the same. The construction of such a representation is illustrated in Figure 3. Furthermore, the forward/backward symmetry was removed by also computing the kink representation starting at the other end of the polymer, and then using only the one with the lower binary value. The kink representation gives a unique number to each symmetry class. We have not found a rigorous proof yet, but we have checked explicitly up to $L \leq 9$ that all polymers with the same kink representation indeed have the same probability.

The reduced state space is constructed by computing the kink representation for each polymer configuration and removing the duplicates (in our implementation, by using hashing). During this phase some additional information is stored about each kink representation: each bond representation that introduces a new kink representation is stored along with the kink representation, and the total number of bond representations for each kink representation is recorded. Table 1 shows the reduction of the configuration space obtained by removing the symmetries. The kink representations are enumerated by sorting them based on their binary value, with the rightmost bit the least significant. This ordering has the property that in most cases moves cause only small changes in binary values, e.g. replacing a kink **+x -x** by **-y +y** swaps two bond-direction bits; replacing **+x -x** by **+y -y** even keeps them the same; the removable-parts bits can be affected as well, but this becomes less likely for increasing part length.

The stored bond representations, one per kink representation, are used to generate the transition matrix for the reduced state space. Each column j of this matrix is computed by generating all the moves of the corresponding bond representation,

TABLE 1
The number of kink representations for polymer lengths $L = 3$–15, the reduction factor of the state space, the number of nonzero elements for the matrix in the kink representation, and the reduction factor of the number of nonzero elements.

L	kink representations	reduction factor	nonzero elements	reduction factor
3	5	7	19	22
4	9	24	49	56
5	37	35	233	75
6	93	84	785	142
7	340	137	3 084	229
8	1 015	276	11 003	407
9	3 534	475	41 594	680
10	11 397	884	150 645	1 182
11	39 082	1 547	559 722	1 999
12	130 228	2 786	2 032 536	3 451
13	445 315	4 888	7 479 343	5 869
14	1 505 785	8 674	27 130 349	10 110
15	5 154 859	15 202	99 199 551	17 248

computing the kink representations for each of the possibilities. If the polymer is moved along the applied field, $\delta t \cdot e^{E}$ is added to the transition matrix element T'_{ij}, where j and i denote the kink representations before and after the move respectively, or $\delta t \cdot e^{-E}$, if it is moved against the electric field. The diagonal element T'_{ii} is such that the sum of each column is exactly one. Thus, T_{ij} is the probability to move from kink representation j to i. The reduction of the number of nonzero matrix elements is also shown in Table 1.

Our choice of time increment guarantees that $\sum_{j \neq i} T_{ij} \leq 1$, so that all matrix elements are between 0 and 1, inclusive. The steady state vector is equal to the eigenvector of this matrix belonging to the largest eigenvalue, which by definition equals unity. Repeatedly multiplying a starting vector by the transition matrix leaves only this eigenvector. This iterative method is well known as the power method. We can compute the same eigenvector by applying the power method to a slightly modified matrix, $T' = I + \omega(T - I)$, which has eigenvalues $\lambda'_i = 1 + \omega(\lambda_i - 1)$, where the λ_i are the eigenvalues of T. The matrix T' has the same eigenvectors as T. We used $\omega = 2$, which is optimal for long polymers and only slightly suboptimal for shorter polymers. This modification spreads the eigenvalues over the interval (-1,1], thereby accelerating the convergence.

4. PARALLEL PROCESSING APPROACH

The reduced transition matrix for $L = 15$ contains about 10^8 elements, so both computational cost (10 Tflops for 50 000 iterations), and memory requirements (1.6 Gbyte) are too high for regular workstations or PCs. We used the parallel programming library BSPlib [21] to obtain our results on a Cray T3E supercomputer, using 64 processors. Within the Bulk Synchronous Parallel (*BSP*) computing model [22], computations and interprocessor communications are separated by global synchronizations. BSPlib supports two types of communication: Direct Remote Memory Access (*DRMA*) and Bulk Synchronous Message Passing (*BSMP*). The DRMA operation **put** copies data into the memory space of a remote process at the next synchronization, and **get** retrieves data from a remote process at the next synchronization. The BSMP operation **send** sends a packet to a queue on a remote processor, which, after the next synchronization, can be accessed there with the **move** operation. In total, the BSP library has 20 primitives. We use the most efficient primitive, **put**. This can be done because the matrix remains constant during all the iterations, so that it becomes worthwhile to analyze the communication pattern beforehand and store a list of memory addresses to be used as the target of **put**s.

In our problem, for $L > 12$, we cannot afford to store the complete matrix on a single processor, so we need to distribute it over a number of processors. The traditional way to do this is to distribute blocks of rows of the matrix over the processors (even though for dense matrices and certain sparse matrices it has been shown that this is not the most efficient way for communication [23]). In principle, we use a more general, two-dimensional matrix distribution, which we will tailor to our problem. The general computation of a matrix-vector product $\vec{x}' = A\vec{x}$ with communication is as follows. The matrix and vector are distributed over the processors: the nonzero matrix elements A_{ij} and the vector components x_i are each assigned to a processor. The matrix-vector product is given by $x'_i = \sum_j A_{ij} x_j$. The first step is to communicate the components x_j to the processors with the corresponding A_{ij}. Now, each processor q computes the partial row sums $s_{iq} = \sum'_j A_{ij} x_j$, where $\sum'_j$ denotes a summation that runs only over indices j for which A_{ij} has been assigned to processor q. The partial row sums are then communicated to the processor containing x'_i, and finally they are accumulated into the components x'_i.

The matrix we have to deal with is sparse and we exploit this in our computations, since we only handle nonzero elements T_{ij}. In addition, the nonzero structure shows 'patches' with many nonzero elements. We can exploit this to make our communications faster. Consider a rectangular patch (i.e., a contiguous submatrix). A value x_j must already be sent to the owner of the patch if one element T_{ij} in column j of the patch is nonzero. It is likely that most columns have at least one nonzero, so that we might as well send all x_j for that patch. This makes it possible to send a contiguous subvector of $\vec{x}$, which is more efficient than sending separate components; this comes at the expense of a few unnecessary communications. The trade-off can be shifted by increasing or decreasing the patch size.

To find suitable patches, we first divide the state vector into contiguous subvectors. We use a heuristic to partition the matrix into blocks of rows with approximately the same number of nonzeros. If we use P processors, and we want each processor to have K subvectors, we have to divide the vector into KP subvectors. (The factor K is the overpartitioning factor.) This initial division tries to minimize the computation time. Next, we adjust the divisions to reduce communication: a suitable patch in the matrix corresponds to an input subvector of kink representations where only the last few bits differ, and also to an output subvector with that property. Therefore, we search for a pair of adjacent kink representations that has

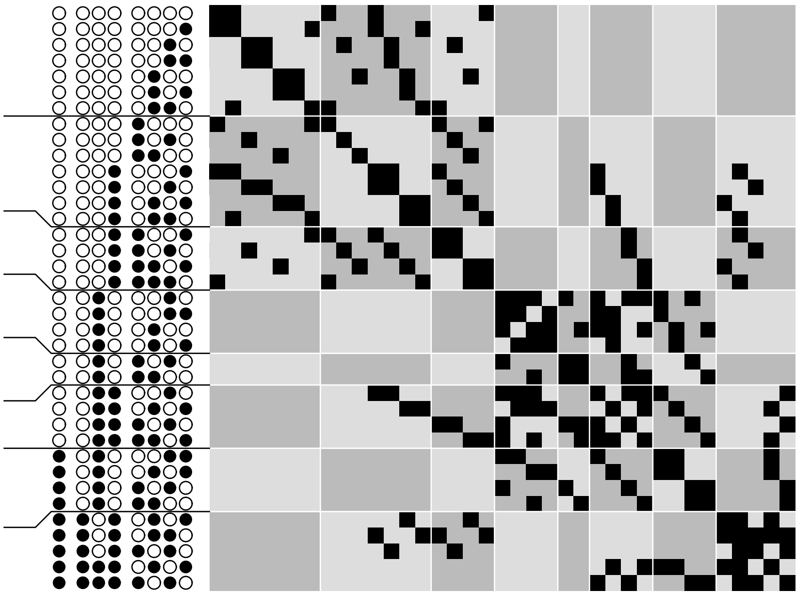

FIG. 4. Reduced transition matrix for polymer length $L = 5$. The size of the matrix is 37×37 and it has 233 nonzero elements, shown as black squares. To the left of each row is the corresponding kink representation written as a binary number, with black circles denoting 1 and open ones 0. The horizontal lines on the left show the initial division of the reduced state vector into eight contiguous parts, optimized to balance the number of nonzeros in the corresponding matrix rows. The jumps of these lines indicate slight adjustments to make the division fit the nonzero structure of the matrix. The resulting vector division induces a division of the rows and columns of the matrix, and hence a partitioning into 64 submatrices, shown by the gray checkerboard pattern. Complete submatrices are now assigned to the processors of a parallel computer.

a different bit as much as possible to the left. This is a suitable place to split. We try to keep the distance from the starting point as small as possible.

As an example of the structure of the reduced transition matrices and the division into submatrices, we show the nonzero structure of the matrix for $L = 5$ in Figure 4 and its corresponding communication matrix in Figure 5 (left). The communication matrix is built from the partitioned transition matrix, by considering each submatrix as a single element. It is a sparse matrix of much smaller size, which determines the communication requirements. Our communication matrix for $L = 13$ is given in Figure 5 (right).

5. RESULTS: DRIFT VELOCITIES AND DIFFUSION COEFFICIENTS

Figure 6 shows the numerically exact values for the drift velocity of the cage polymers up to length $L = 15$. As expected, initially the drift velocity increases linearly with field strength, and eventually it reaches a maximum drift velocity, after which it decreases exponentially with field strength. Clearly visible in Monte Carlo data [19] is a regime just before the maximum velocity where the drift velocity increases quadratically with the field strength; in the numerically exact data presented here, for relatively short chains, this regime is hardly visible. Only for the largest chains that we can handle, this regime starts to become noticeable: ex-

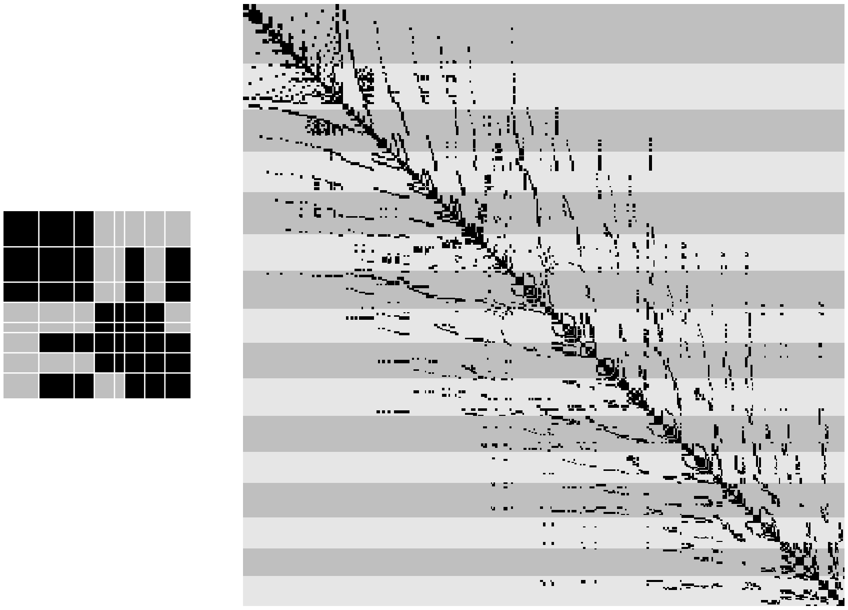

FIG. 5. Communication matrix for $L = 5$ (left) and $L = 13$ (right). Note that the matrix for $L = 5$ can be obtained by replacing each nonempty submatrix in Figure 4 by a single nonzero element. The communication matrix for $L = 13$, of size 320×320, is distributed over 16 processors in a row distribution.

amining the ratio of the drift velocities for $L = 14$ and $L = 15$, we find a trend towards unity just before the maximum velocity is reached.

It is clear from Figure 6, that for longer polymers, the maximum velocity sets in at lower electric fields; for $L = 15$ the highest drift velocity is attained at $E \approx 0.248$. This trend was also observed in the Monte Carlo simulations, where for $L = 200$, this regime starts already at $E \approx 0.04$. Since the exponential decrease in the drift velocity as a function of field strength sets in earlier for longer polymers, the newly reported third regime does not provide an explanation for the phenomenon known as "band inversion", where under certain conditions, longer polymers move faster than shorter ones (see Ref. [20]).

The diffusion coefficient is computed from the drift velocities using the Nernst-Einstein relation $v = qLED$, which holds for vanishing E. We used the velocity found at $E = 10^{-6}$ to compute the diffusion coefficient (see Table 2).

It is known that asymptotically for large polymers the diffusion coefficient behaves as $D \sim L^{-2}$, but with large finite-size corrections for usual polymer lengths. The nature of these finite-size corrections is the second important topic of this work. As the polymers are modeled as a random walk of $N = L-1$ steps, finite-size corrections of the order $N^{-1/2}$ are expected. Let us call $d(N) = D \cdot (N+1)^2 = DL^2$, and $d_\infty = (DL^2)_{L\to\infty}$; we expect that for large but finite polymers $d(N) = d_\infty + aN^{-x}$. The parameters a and x can be found from this equation by differentiation: $\frac{\partial d}{\partial N} = -axN^{-x-1} \approx \frac{1}{2}\big(d(N+1) - d(N-1)\big)$. A least-squares fit of the derivative of the new data against N for $N = 8$–13 gives $a = 2.469(5)$ and $x = 0.512(6)$, strongly suggesting finite-size corrections with an exponent $\frac{1}{2}$. This shows an advantage of

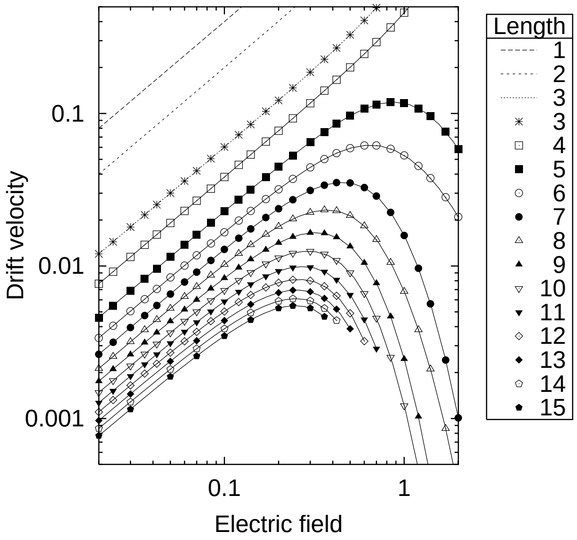

FIG. 6. The graphs show the computed drift velocities of the cage polymers as a function of electric field strength E. For $E \leq 1$, the relative error is less than 10^{-10}; all other points have a relative error less than 10^{-3}. The graphs for lengths 1, 2, and 3 are $v_1 = 2(e^E - e^{-E})$, $v_2 = e^E - e^{-E}$, and $v_3 = 4(e^{3E} - e^{-3E})/(18 + 11(e^{2E} + e^{-2E}))$ respectively; for $L > 3$, the computed points are connected by straight lines.

TABLE 2

Diffusion coefficients for cage polymers up to length $L = 15$. All values have a relative error of less than 10^{-10}, except for $L = 15$, which has relative error 10^{-6}. It is expected for large L, that the diffusion coefficient is $D \sim L^{-2}$ [18].

L	D	L^2D
3	0.200 000 000 00	1.800 000 000 00
4	0.095 541 401 267	1.528 662 420 27
5	0.045 892 037 844	1.147 300 946 10
6	0.028 134 332 038	1.012 835 953 37
7	0.018 844 680 456	0.923 389 342 34
8	0.013 302 014 726	0.851 328 942 46
9	0.009 776 090 803 8	0.791 863 355 11
10	0.007 424 928 046 4	0.742 492 804 64
11	0.005 790 292 327 1	0.700 625 371 58
12	0.004 615 107 026 0	0.664 575 411 74
13	0.003 746 569 185 6	0.633 170 192 36
14	0.003 089 624 048 9	0.605 566 313 58
15	0.002 582 785 982 6	0.581 126 846 09

the numerically exact computations over Monte Carlo simulations in that we can compute the derivative of the data reliably.

We used our new diffusion coefficients, combined with data from Barkema and Krenzlin, and data from Van Heukelum and Beljaars, to find the length dependence of the diffusion coefficient. A least-squares fit with $d(N) = a + bN^{-1/2} + cN^{-1}$ gives

TABLE 3

BSP cost, time, efficiency, and speedup for one matrix-vector multiplication.

L	P	BSP cost	time (ms)	efficiency	speedup
12	8	$545156 + 64716g + 2l$	47 + 4.3	85%	6.8
13	16	$1002824 + 187347g + 2l$	89 + 13	81%	13.0
14	32	$1836920 + 425152g + 2l$	169 + 44	73%	23.4
15	64	$3452776 + 1380415g + 2l$	330 +112	67%	42.9

$d(N) = 0.172(6) + 0.63(8)N^{-1/2} + 3.3(2)N^{-1}$, and a least-squares fit with $d(N) = (a' + b'N^{-1/2} + c'N^{-1})^{-1}$ gives $d(N) = (5.67(5) - 22.2(5)N^{-1/2} + 28(2)N^{-1})^{-1}$. Both of these expansions converge, within the error margins, to the same value for large N. The first expansion converges to 0.172(6), and the second expansion converges to $1/5.67(5) = 0.176(2)$. Combining these results, we conclude that for large L the diffusion coefficient is $D = 0.175(2)L^{-2}$. Our diffusion coefficient agrees with that of Barkema and Krenzlin [18], but they found a different finite-size scaling: $DN^2 = 0.173 + 1.9N^{-2/3}$.

6. RESULTS: COMPUTATION TIME AND EFFICIENCY

Our computations were performed on a Cray T3E computer. The peak performance of a single node of the Cray T3E is 600 Mflop/s for computations. The bsp_probe benchmark shows a performance of 47 Mflop/s per node [21]. The peak interprocessor bandwidth is 500 Mbyte/s (bi-directional). The bsp_probe benchmark shows a sustained bi-directional performance of 94 Mbyte/s per processor when all 64 processors communicate at the same time. This is equivalent to a BSP parameter $g = 3.8$, where g is the cost in flop time units of one 64-bit word leaving or entering a processor. The measured global synchronization time for 64 processors is 48 μs, which is equivalent to $l = 2259$ flop time units.

Table 3 presents the execution time of one iteration of the algorithm in two forms: the BSP cost $a+bg+cl$ counts the flops and the communications and thus gives the time on an arbitrary computer with BSP parameters g and l, whereas the time in ms gives the measured time on this particular architecture, split into computation and communication time. (The total measured synchronization time is negligible.) The BSP cost can be used to predict the run time of our algorithm on different architectures. Table 3 also gives the efficiency and speedup relative to a sequential program.

Peak computation performance is often only reached for dense matrix-matrix multiplication; the performance for sparse matrix-vector multiplication is always much lower. Comparing the flop count and the measured computation time for the largest problem $L = 15$, we see that we achieve about 10.5 Mflop/s. Comparing the communication count with the measured communication time, we obtain a g-value of 8.1 μs, (or $g = 3.8$ flop units, see above). This means that we attain the maximum sustainable communication speed. This is due to the design of our algorithm, which communicates contiguous subvectors, instead of single components. Furthermore, the results show that our choice to optimize mainly the computation (by choosing a row distribution) is justified for this architecture: the communication time is always less than a third of the total time. For a different machine, with a higher value of

g, more emphasis must be placed on optimizing the communication, leading to a two-dimensional distribution.

Each iteration of our computation contains one matrix-vector multiplication. The number of iterations needed for convergence depends on the length of the polymer, and on the applied electric field. The iteration was stopped when either the accuracy was better than 10^{-10}, or the number of iterations exceeded 100 000. In the latter case, the accuracy was computed at termination. Typically, for $L = 15$ and a low electric field strength, 50 000 iterations are needed. Only computed values with accuracy 10^{-4} or better are shown in Figure 6. For $L = 12$, we compared the output for the parallel program with that of the sequential program and found the difference to be within rounding errors. The total speedup for $L = 15$, compared to a naive implementation (for which one would need 38.5 Tbyte of memory), is a factor $1.5 \cdot 10^6$: a factor of $\omega = 2$ by increasing the time step, a factor of 17 248 by using a reduced state space, and another factor 42.9 by using a parallel program on 64 processors.

7. CONCLUSIONS

In numerically exact computations on the cage model, extended for the study of DNA electrophoresis, we exploited symmetries of the model and applied parallel processing. This has resulted in a computational speedup factor of over a million.

Regarding the cage model, we conclude that the polymer diffusion coefficient D scales asymptotically for large polymers as $DL^2 = 0.175(2)$, in qualitative agreement with De Gennes' reptation arguments, and in quantitative agreement with earlier recent simulation reports [18]. The finite-size corrections are found to be a combination of $N^{-1/2}$ (which asymptotically is the dominant correction) and N^{-1}, and probably higher-order corrections; this is in disagreement with earlier reports [11, 18] where the leading corrections were reported to be $N^{-2/3}$, but in agreement with recent density matrix renormalisation group computations by Carlon *et al.* [24].

As a function of increasing electric field strength E, the drift velocity first increases linearly as $v \sim E/L$, then (for longer polymers) it increases quadratically as $v \sim E^2$ [19], to reach its maximum value at some value of E which decreases with polymer length L; a further increase in field strength makes the drift velocity decline exponentially. The fact that this maximum velocity occurs at a decreasing field strength — for polymers with length $L = 15$ already at $E = 0.248$ — indicates that, in the range of electric field strengths where the model is supposed to be realistic ($E \ll 1$), there is a third regime besides the two mentioned before. This third regime, however, cannot explain the phenomenon known as "band inversion".

ACKNOWLEDGMENT

We would like to thank the Netherlands Computer Facilities foundation and the High Performance Applied Computing center at Delft University of Technology for providing access to a Cray T3E.

REFERENCES

1. P. G. de Gennes, J. Chem. Phys. **55**, 572 (1971).
2. M. Rubinstein, Phys. Rev. Lett. **59**, 1946 (1987).
3. J. M. J. van Leeuwen and A. Kooiman, Physica A **184**, 79 (1992).
4. A. Kooiman and J. M. J. van Leeuwen, Physica A **194**, 163 (1993).
5. A. Kooiman and J. M. J. van Leeuwen, J. Chem. Phys. **99**, 2247 (1993).
6. M. Prähofer and H. Spohn, Physica A **233**, 191 (1996).
7. T. A. J. Duke, Phys. Rev. Lett. **62**, 2877 (1989).
8. T. A. J. Duke, J. Chem. Phys. **93**, 9049 (1990).
9. T. A. J. Duke, J. Chem. Phys. **93**, 9055 (1990).
10. B. Widom, J. L. Viovy, and A. D. Defontaines, J. Phys. I France **1**, 1759 (1991).
11. G. T. Barkema, J. F. Marko, and B. Widom, Phys. Rev. E **49**, 5303 (1994).
12. T. A. J. Duke, A. N. Semenov, and J. L. Viovy, Phys. Rev. Lett. **69**, 3260 (1992).
13. G. T. Barkema and M. E. J. Newman, Physica A **244**, 25 (1997).
14. M. E. J. Newman and G. T. Barkema, Phys. Rev. E **56**, 3468 (1997).
15. G. T. Barkema, C. Caron and J. F. Marko, Biopolymers **38**, 665 (1996).
16. A. Kolomeiski, Cornell Ph.D. Thesis (1998).
17. K. E. Evans and S. F. Edwards, J. Chem. Soc. Faraday Trans. 2 **77**, 1891 (1981).
18. G. T. Barkema and H. M. Krenzlin, J. Chem. Phys. **109**, 6486 (1998).
19. A. van Heukelum and H. R. W. Beljaars, J. Chem. Phys. **113**, 3909 (2000).
20. J. Noolandi, J. Rousseau, G. W. Slater, C. Turmel, and M. Lalande, Phys. Rev. Lett. **58**, 2428 (1987).
21. J. M. D. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. H. Bisseling, Parallel Computing **24**, 1947 (1998).
22. L. G. Valiant, Comm. of the ACM **33**, 103 (1990).
23. R. H. Bisseling and W. F. McColl, Proc. IFIP 13th World Computer Congress, Vol. I, North-Holland, 509 (1994).
24. E. Carlon, A. Drzewiński and J. M. J. van Leeuwen, cond-mat/0010177.