Parallel O(N) Stokes' solver towards scalable Brownian dynamics of hydrodynamically interacting objects in general geometries

Xujun Zhao, Jiyuan Li, Xikai Jiang, Dmitry Karpeev, Olle Heinonen, Barry Smith, Juan P. Hernandez-Ortiz, and Juan J. de Pablo

Citation: The Journal of Chemical Physics **146**, 244114 (2017); doi: 10.1063/1.4989545
View online: http://dx.doi.org/10.1063/1.4989545
View Table of Contents: http://aip.scitation.org/toc/jcp/146/24
Published by the American Institute of Physics

Parallel *O*(*N*) Stokes' solver towards scalable Brownian dynamics of hydrodynamically interacting objects in general geometries

Xujun Zhao,[1,2,a)] Jiyuan Li,[2,a)] Xikai Jiang,[2,a)] Dmitry Karpeev,[1] Olle Heinonen,[3,4] Barry Smith,[1] Juan P. Hernandez-Ortiz,[2,5,b)] and Juan J. de Pablo[2,3,c)]
[1]*Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, Illinois 60439, USA*
[2]*Institute for Molecular Engineering, University of Chicago, Chicago, Illinois 60637, USA*
[3]*Materials Science Division, Argonne National Laboratory, Lemont, Illinois 60439, USA*
[4]*Northwestern-Argonne Institute for Science and Engineering, Evanston, Illinois 60208, USA*
[5]*Departmento de Materiales, Universidad Nacional de Colombia, Sede Medellin, Colombia*

(Received 27 March 2017; accepted 6 June 2017; published online 29 June 2017)

An efficient parallel Stokes' solver has been developed for complete description of hydrodynamic interactions between Brownian particles in bulk and confined geometries. A Langevin description of the particle dynamics is adopted, where the long-range interactions are included using a Green's function formalism. A scalable parallel computational approach is presented, where the general geometry Stokeslet is calculated following a matrix-free algorithm using the general geometry Ewald-like method. Our approach employs a highly efficient iterative finite-element Stokes' solver for the accurate treatment of long-range hydrodynamic interactions in arbitrary confined geometries. A combination of mid-point time integration of the Brownian stochastic differential equation, the parallel Stokes' solver, and a Chebyshev polynomial approximation for the fluctuation-dissipation theorem leads to an $O(N)$ parallel algorithm. We illustrate the new algorithm in the context of the dynamics of confined polymer solutions under equilibrium and non-equilibrium conditions. The method is then extended to treat suspended finite size particles of arbitrary shape in any geometry using an immersed boundary approach. [http://dx.doi.org/10.1063/1.4989545]

I. INTRODUCTION

The dynamics of suspended objects in confined geometries, such as colloidal particles or polymeric molecules, is important for a wide range of applications, including enhanced oil recovery, biology, materials design, and medicine. For example, in modern genomic technologies, such as DNA sequencing or optical mapping, individual DNA molecules can be directly manipulated in microfluidic and nanofluidic devices, where hydrodynamic effects play a key role.[1–5] Another set of applications, involving lab-on-a-chip devices, aim to manipulate cell and particle suspensions through the precise design of flow conduits and accurate control of flow rates and hydrodynamic forces.[6,7] From a physical point of view, many intriguing rheological behaviors in suspensions, including foam formation,[8] shear thickening,[9] solidification,[10] and shear induced migration,[11–13] have been traced back to hydrodynamic effects. Similarly, in the context of intracellular motion, the effects of crowding are of considerable interest and are believed to be caused by hydrodynamic interactions.[14–21] The precise mechanisms that underpin such phenomena, however, remain poorly understood, particularly when they arise at high concentrations. Efficient methods and algorithms capable of accurately capturing hydrodynamic interactions and their influence on microstructure evolution are necessary to describe the dynamics of concentrated particle suspensions and for the design of fluidic devices.

Brownian Dynamics (BD) is generally used for problems that are characterized by a broad separation of length and time scales—on the order of several orders of magnitude, between the relaxation times of suspended particles or molecules and those of the solvent. The system is evolved through the integration of the Fokker-Planck equation for the probability distribution function, which incorporates a force balance for the suspended particles and the fluid, coupled through the fluctuation-dissipation theorem.[22,23] A representative system is provided by a 21 μm DNA molecule suspended in an aqueous solution. The characteristic size of the molecule is 18×10^{-10} m, and it has a relaxation time of 0.1 s. In contrast, the fluid has a characteristic diffusion time, i.e., the time required for the molecule to diffuse its own size, of 10^{-15} s. Simulations that include a solvent explicitly would be impractical. In BD, the fluid is represented as a continuum governed by the Stokes momentum equations.[23–26] In such a system, the motion of each particle is influenced by that of all other particles, and it is mediated by hydrodynamic interactions (HIs), which are long-ranged. Explicit expressions to calculate HI are only available in free space and for simple geometries; for the general cases encountered in micro- or nano-fluidic devices, hydrodynamic forces must be obtained numerically.

[a)]X. Zhao, J. Li, and X. Jiang contributed equally to this work.
[b)]Electronic mail: jphernandezo@unal.edu.co
[c)]Electronic mail: depablo@uchicago.edu

BD simulations have been used extensively to study the dynamics of macromolecules and colloids,[27–30] the transport of DNA,[3,31–35] and the flow behavior of colloidal dispersions.[36–38] Such simulations are computationally demanding, and are often limited to small systems in unbounded domains or in simple geometries. Over the last two decades, increasingly efficient methods and algorithms have been developed to evaluate HI. These include Stokes solvers,[39,40] Stokesian Dynamics (SD),[41–43] the Lattice Boltzmann method (LBM),[44–47] dissipative particle dynamics (DPD),[48,49] and Green's function based methods.[29,30,39,50–53] Fluid particle methods, like DPD, and fluid mesh methods, like LBM, are widely used and preferred by numerous authors. In DPD, the fluid/solvent molecules are represented by coarse-grained particles, which reduce the number of solvent particles and increase time intervals when compared to molecular dynamics (MD). However, correct implementation of confining walls, i.e., no-slip boundary conditions, is not a trivial task in DPD. On the other hand, the LBM provides approximate solutions to the incompressible Navier-Stokes equations and requires finite Reynolds, *Re*, and Mach, *Ma*, numbers. Complementary methods are also needed in the LBM to properly model complex geometries and eliminate viscosity-dependent errors.[54,55] Similar to other fluid element/mesh methods, LBM can become mesh-dependent when high concentrations of the suspended particles or molecules are considered, and care must be exercised in its implementation. The General geometry Ewald-like method (GgEm)[53] is of particular interest to our work because it directly solves the Stokes equations ($Re = 0$) and is able to handle arbitrary geometries and no slip boundary conditions efficiently. GgEm uses the linear character of the momentum equations to split the contributions of long-range interactions into a local and a global calculation, following the general philosophy of the Ewald split. It resolves short-range particle-particle interactions precisely, while far-field interactions are evaluated on a mesh using a computational fluid dynamics approach. GgEm has been used for simulations of confined DNAs,[3] DNA pore translocation,[30,56] charged dipoles[57] and polyelectrolytes,[58] suspensions of rigid and deformable particles,[59,60] and active suspensions.[61–63]

As helpful as the GgEm method has been, its implementation is challenging, particularly in highly parallel algorithms. The purpose of this work is to introduce an efficient and scalable computational implementation and to publicly release the corresponding software. To do so, we rely on a finite element formulation for the far-field contributions, using distributed memory parallelization methods that include a direct lower-upper (LU) decomposition analytic solver for systems with less than one million degrees of freedom, and a fast parallel iterative solver, with hybrid preconditioning, for larger systems. Our parallel Stokes' solver is combined with the GgEm, a mid-point integration scheme, and a Chebyshev polynomial approximation for the fluctuation-dissipation theorem, to arrive at an efficient $O(N)$ and completely scalable parallel BD algorithm. In what follows, we offer a detailed explanation of our proposed approach, including a discussion of important numerical issues and general aspects of the algorithm. We then illustrate its use in the context of confined polymer solutions and finite-size particle dynamics. More specifically, we calculate the diffusion of polymers in a slit geometry and demonstrate that the correct Zimm scaling is obtained.[26,64] We then simulate flowing polymers in a cross-channel geometry and show how the method can be used in arbitrary domain shapes. We also show results for the dynamics of finite-sized particles using an immersed boundary-GgEm formulation.

II. THE FOKKER-PLANCK AND THE BROWNIAN DYNAMICS EQUATIONS

The stochastic differential equation that governs the dynamics of N suspended beads in a viscous solvent is obtained from the force balance

$$\frac{\partial}{\partial t}\mathbf{M}\cdot\mathbf{U} = \mathbf{f}_\nu^H + \mathbf{f}_\nu^B + \mathbf{f}_\nu^{EV} + \mathbf{f}_\nu^C + \mathbf{f}_\nu^{\text{other}}, \tag{1}$$

for each bead $\nu = 1, \ldots, N$, where $\mathbf{f}_\nu^H$ is the hydrodynamic force, $\mathbf{f}_\nu^B$ is the Brownian force, $\mathbf{f}_\nu^{EV}$ are excluded volume forces (bead-bead and bead-wall), $\mathbf{f}_\nu^C$ are configurational forces (springs), $\mathbf{f}_\nu^{\text{other}}$ are other external forces that may apply (e.g., forces of electrostatic or magnetostatic character), $\mathbf{M}$ is the mass tensor, and $\mathbf{U}$ is the translational velocity. *Re* is assumed to be zero and inertial effects are neglected; therefore, the left-hand side of Eq. (1) is equal to zero.

In a Lagrangian frame of reference, the evolution equation for the probability distribution function, $\psi(\mathbf{x}, t)$, for the bead positions is a convection-diffusion equation of the Fokker-Planck type,[22]

$$\frac{\partial\psi}{\partial t} = -\frac{\partial}{\partial\mathbf{R}}\cdot\left[\left(\boldsymbol{\kappa}\cdot\mathbf{R} + \frac{1}{k_BT}\mathbf{D}\cdot\mathbf{F}\right)\psi\right] + \frac{\partial}{\partial\mathbf{R}}\cdot\mathbf{D}\cdot\frac{\partial}{\partial\mathbf{R}}\psi, \tag{2}$$

where $\mathbf{R} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$ is a $3N$ vector containing the spatial coordinates of the beads, $\boldsymbol{\kappa}$ is a diagonal $3N \times 3N$ tensor with the diagonal components of the imposed velocity gradient $\nabla\mathbf{U}_0$, $\mathbf{D}$ is the $3N \times 3N$ diffusion tensor, $\mathbf{F}$ is a $3N$ vector with the non-Brownian and non-hydrodynamic components of the force, k_B is Boltzmann's constant, and T is the temperature.

The diffusion tensor is given by $\mathbf{D} = k_BT\mathbf{M}$, where $\mathbf{M}$ is the mobility tensor that includes Stokes' drag and the pairwise Stokeslets that account for the hydrodynamic interactions between beads. Note that the time evolution of ψ, according to the Fokker-Planck equation, is due to convection, first term on the right-hand side, and diffusion, second term on the right-hand side, of the probability density. This is particularly important because once this equation is transformed to produce an equivalent stochastic differential equation, convective and diffusive terms will be generated. Each of these terms imposes particular challenges for their efficient numerical integration.

Assuming a continuous probability density, using the Chapman-Kolmogorov equation with a Wiener process, an equivalent stochastic differential equation for the motion of the beads is obtained as follows:[23]

$$d\mathbf{R} = \left[\mathbf{U_0} + \mathbf{M}\cdot\mathbf{F} + \frac{\partial}{\partial\mathbf{R}}\cdot\mathbf{D}\right]dt + \sqrt{2}\mathbf{B}\cdot d\mathbf{W}, \tag{3}$$

where $\mathbf{U_0}$ denotes a $3N$ vector with the unperturbed fluid velocity at the bead's position and $\mathbf{U} = \mathbf{M}\cdot\mathbf{F}$ contains the fluctuating velocities from the hydrodynamic interactions; this is a convective term. The divergence of the diffusion tensor $\frac{\partial}{\partial \mathbf{R}}\cdot\mathbf{D}$ is the drift resulting from the configuration-dependent mobility of the confined particles; this is the first diffusive term. Finally, $d\mathbf{W}$ is a random vector, the components of which are obtained from a real-valued Gaussian distribution with zero mean and variance dt. It is coupled to the diffusion tensor through the fluctuation-dissipation theorem: $\mathbf{D} = \mathbf{B}\cdot\mathbf{B}^T$. The term $\sqrt{2}\mathbf{B}\cdot d\mathbf{W}$—the second diffusive term—represents the Brownian displacement, which results from collisions between the beads and the surrounding molecules in the fluid.

Three issues must be addressed in order to arrive at an efficient numerical algorithm for Eq. (3). First, the mobility/diffusion tensor cannot be constructed explicitly; this would result in an $O(N^2)$ algorithm. This implies that the fluctuating velocity, $\mathbf{U}$, the divergence of the diffusion tensor, $\nabla\cdot\mathbf{D}$, and the diffusion tensor decomposition, $\mathbf{B}$, must be implemented in a matrix-free scheme. In our proposed algorithm, we use (i) the General geometry Ewald-like method (GgEm) for a matrix-free product of the mobility tensor with any vector, $\mathbf{M}\cdot\mathbf{F}$; (ii) a mid-point algorithm, proposed by Fixman,[65] that avoids the explicit calculation of $\nabla\cdot\mathbf{D}$; and (iii) a Chebyshev polynomial approximation for the $\mathbf{B}\cdot d\mathbf{W}$ product, also proposed by Fixman,[66] that uses the GgEm to avoid the explicit calculation of $\mathbf{D}$. The entire algorithm scales as $O(N)$ and enables one to handle arbitrarily shaped systems by relying on the GgEm.[53] In addition, we also rely on parallel finite element methods and parallel libraries to achieve the above tasks.

A. General geometry Ewald-like method (GgEm)

A velocity field, $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N) = \mathbf{M}\cdot\mathbf{F}$, driven by a set of forces at the bead positions, $\mathbf{F} = (\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_N)$, can be calculated directly by recognizing that, for zero Re, the force balance on the fluid is reduced to Stokes' momentum equations,

$$-\nabla p + \mu\nabla^2\mathbf{u} = -\boldsymbol{\rho}^f, \tag{4}$$

$$\nabla\cdot\mathbf{u} = \mathbf{0}, \tag{5}$$

where μ is the fluid viscosity, $\mathbf{u}(\mathbf{x})$ is the fluid velocity field generated by a force density $\boldsymbol{\rho}^f(\mathbf{x}) = \sum_{\nu=1}^{N}\mathbf{f}_\nu\delta(\mathbf{x}-\mathbf{x}_\nu)$, and p is the corresponding pressure. The inclusion of the Dirac delta function, $\delta(\mathbf{x})$, in the force density implies that we are assuming, for the moment being, that the forces exerted by the beads on the fluid are singular point-forces.

Similar to conventional particle-mesh Ewald methods,[67] GgEm uses the linear character of Stokes' equation to re-define the force density in Eq. (4),

$$\boldsymbol{\rho}^f(\mathbf{x}) = \boldsymbol{\rho}_l^f(\mathbf{x}) + \boldsymbol{\rho}_g^f(\mathbf{x}), \tag{6}$$

where the "local" density is

$$\boldsymbol{\rho}_l^f(\mathbf{x}) = \sum_{\nu}^{N}[\delta(\mathbf{x}-\mathbf{x}_\nu) - g(\mathbf{x}-\mathbf{x}_\nu)]\mathbf{f}_\nu(\mathbf{x}), \tag{7}$$

while the "global" density is given by

$$\boldsymbol{\rho}_g^f(\mathbf{x}) = \sum_{\nu} g(\mathbf{x}-\mathbf{x}_\nu)\mathbf{f}_\nu(\mathbf{x}). \tag{8}$$

The "screening" function, $g(\mathbf{x})$, must satisfy $\int_{\text{all space}} g(\mathbf{x}) = 1$, and it is chosen in a way that the local density contribution to the velocity field decays exponentially. We have found that[53] a modified Gaussian function of the type,

$$g(\mathbf{r}) = \frac{\alpha^3}{\pi^{3/2}}e^{-\alpha^2 r^2}\left[\frac{5}{2} - \alpha^2 r^2\right], \tag{9}$$

results in an exponentially decaying "local" velocity field over a length scale α^{-1}.

Ignoring the confining walls, a Green's function calculation can be carried out to obtain the local contribution according to

$$\mathbf{u}_l(\mathbf{x}) = \sum_{\nu}\mathbf{G}_l(\mathbf{x}-\mathbf{x}_\nu)\cdot\mathbf{f}_\nu, \tag{10}$$

where $\mathbf{G}_l(\mathbf{x})$ is a screened Green's function given by

$$\mathbf{G}_l(\mathbf{x}) = \frac{1}{8\pi\mu}\left[\boldsymbol{\delta} + \frac{\mathbf{xx}}{r^2}\right]\left[\frac{\operatorname{erfc}(\alpha r)}{r}\right] - \frac{1}{8\pi\mu}\left[\boldsymbol{\delta} - \frac{\mathbf{xx}}{r^2}\right]\left[\frac{2\alpha}{\pi^{1/2}}e^{-\alpha^2 r^2}\right], \tag{11}$$

where $r = |\mathbf{x}|$.

Note that the aim here is to calculate the product of the mobility/diffusion tensor and a forcing vector. In doing so, the physical characteristics of such a tensor, its positive definiteness and self-adjoint form, must be preserved. As the concentration of beads is increased, using a point-force model does not ensure such limits. We can, however, introduce a point-force regularization using the same form of the modified Gaussian over a new length scale ξ^{-1}. For $\xi^{-1} = 3a/\pi^{1/2}$, the maximum fluid velocity is equal to that of a particle with radius a and the pair mobility remains positive-definite. Introducing this regularization in the local contribution is trivial because it only modifies the screened Green's function as follows:

$$\mathbf{G}_l^R(\mathbf{x}) = \frac{1}{8\pi\mu}\left[\boldsymbol{\delta} + \frac{\mathbf{xx}}{r^2}\right]\left[\frac{\operatorname{erf}(\xi r)}{r} - \frac{\operatorname{erf}(\alpha r)}{r}\right] + \frac{1}{8\pi\mu}\left[\boldsymbol{\delta} - \frac{\mathbf{xx}}{r^2}\right]\left[\frac{2\xi}{\pi^{1/2}}e^{-\xi^2 r^2} - \frac{2\alpha}{\pi^{1/2}}e^{-\alpha^2 r^2}\right]. \tag{12}$$

The functions $\mathbf{G}_l^R$ and $\mathbf{G}_l$ decay exponentially on the length scale α^{-1}. In practice, the local velocity field can be computed, as is done in Ewald-like methods, by only considering the neighbors of each particle. On the other hand, the "global" contribution to the velocity field, $\mathbf{u}_g(\mathbf{x})$, which is driven by the force density $\boldsymbol{\rho}_g^f(\mathbf{x})$, is obtained numerically by solving the Stokes equations,

$$-\nabla p_g + \mu\nabla^2\mathbf{u}_g = -\boldsymbol{\rho}_g^f, \tag{13}$$

$$\nabla\cdot\mathbf{u}_g = \mathbf{0}. \tag{14}$$

This requires that $\mathbf{u}_l + \mathbf{u}_g$ satisfy the appropriate boundary conditions at the domain walls and boundaries. For example, a Dirichlet boundary condition, where the wall velocity is given by $\overline{\mathbf{u}}(\mathbf{x_W})$, results in a boundary condition for the global contribution $\mathbf{u}_g(\mathbf{x}) = \overline{\mathbf{u}}(\mathbf{x_W}) - \mathbf{u}_l(\mathbf{x})$, for $\mathbf{x}\in\mathbf{x_W}$.

The global solution is obtained on a set of M discrete points on a mesh; in this regard, the GgEm resembles a Particle-Particle-Particle-Mesh (PPPM) method, where the assignment function is replaced by the delta function. Several techniques (finite differences, finite elements, and spectral methods) can be used to find the global contribution; after the mesh is resolved, interpolation is used to get the value of the global velocity at position $\mathbf{x}$. As mentioned earlier, in the GgEm, $\mathbf{M}\cdot\mathbf{F}$ is determined implicitly, requiring $O(N)$ operations. For the point-force version, $\mathbf{M} = \mathbf{M}^T$, as required by the self-adjoint character of Stokes' equations. For the regularized version, $\mathbf{M} = \mathbf{M}^T$ if the boundary condition on $\mathbf{u}_g$ remains the same as for the point-force version, at the cost of violating the no-slip boundary condition for points within $\sim\xi^{-1}$ from the wall (which would normally be prevented by excluded volume interactions).

B. Mid-point integration algorithm

To avoid the explicit calculation of $\nabla\cdot\mathbf{D}$, we adopt Fixman's mid-point algorithm,[24] where the time evolution from time t to time $t+dt$ requires an intermediate step t^*,

$$\mathbf{R}^* = \mathbf{R}(t) + \frac{1}{2}\left[\mathbf{U}_0(\mathbf{R}) + \mathbf{M}(\mathbf{R})\cdot\mathbf{F}(\mathbf{R})\right]\Delta t + \frac{1}{2}\sqrt{2}\mathbf{D}(\mathbf{R})\mathbf{B}^{-1}(\mathbf{R})\cdot d\mathbf{W}(t), \tag{15}$$

$$\mathbf{R}(t+\Delta t) = \mathbf{R}(t) + \left[\mathbf{U}_0(\mathbf{R}^*) + \mathbf{M}(\mathbf{R}^*)\cdot\mathbf{F}(\mathbf{R}^*)\right]\Delta t + \sqrt{2}\mathbf{D}(\mathbf{R}^*)\mathbf{B}^{-1}(\mathbf{R})\cdot d\mathbf{W}(t). \tag{16}$$

C. Chebyshev polynomial approximation

Finally, the diffusion tensor decomposition is carried out using a Chebyshev polynomial approximation as proposed by Fixman,[65] which guarantees that the fluctuation-dissipation theorem is satisfied. When combined with the GgEm, it results in a matrix-free algorithm because it only requires matrix-vector products. This method has been widely used in previous studies with unbounded or periodic domains.[3,11,27,33,53] Here we only present a brief summary of the algorithm and highlight some of the key issues that arise in our scalable numerical implementation.

An operation over a matrix, $f(\mathbf{M})$, can be approximated by a linear combination of Chebyshev polynomials as follows:

$$f(\mathbf{M}) = \sum_{l=0}^{L} a_l\mathbf{C}_l, \tag{17}$$

where a_l and L are the coefficients and order of the approximation, respectively, and

$$\mathbf{C}_0 = \mathbf{I}, \tag{18}$$

$$\mathbf{C}_1 = d_a\mathbf{M} + d_b\mathbf{I}, \tag{19}$$

$$\mathbf{C}_{1+1} = 2(d_a\mathbf{M} + d_b\mathbf{I})\mathbf{C}_l - \mathbf{C}_{l-1}. \tag{20}$$

The parameters d_a and d_b depend on the span of the approximation and are defined by

$$d_a = \frac{2}{\lambda_{\max} - \lambda_{\min}}, \tag{21}$$

$$d_b = -\frac{\lambda_{\max} + \lambda_{\min}}{\lambda_{\max} - \lambda_{\min}}, \tag{22}$$

where $[\lambda_{min}, \lambda_{max}]$ is the range of eigenvalues of $\mathbf{M}$ where the approximation takes effect.

For the mid-point algorithm, we require calculation of $\mathbf{y} = f(\mathbf{M})\cdot d\mathbf{w}$. This can then be expressed in terms of a series of matrix-vector products given by

$$\mathbf{y} = f(\mathbf{M})\cdot d\mathbf{w} = \sum_{l=0}^{L} a_l\hat{\mathbf{x}}_l, \tag{23}$$

where

$$\hat{\mathbf{x}}_0 = d\mathbf{w}, \tag{24}$$

$$\hat{\mathbf{x}}_1 = (d_a\mathbf{M} + d_b\mathbf{I})\cdot d\mathbf{w}, \tag{25}$$

$$\hat{\mathbf{x}}_{1+1} = 2(d_a\mathbf{M} + d_b\mathbf{I})\hat{\mathbf{x}}_l - \hat{\mathbf{x}}_{l-1}. \tag{26}$$

The convergence rate of this polynomial approximation depends on both the condition number of the matrix and the accuracy of the evaluation of its eigenvalues. The well-conditioned character of the diffusion tensor is related to its positive-definite form, which is guaranteed by using the GgEm and a point-force regularization. More important is the proper calculation (or estimation) of the limiting eigenvalues. This procedure cannot rely on an explicit construction of the tensor, and it should be scalable and parallelizable. In our present numerical implementation, the eigenvalues, $\lambda_{\min}$ and $\lambda_{\max}$, are calculated by using the open-source software package SLEPc.[68] It is capable of rapidly solving upper and lower bounds of eigenvalues for large systems on parallel computers. SLEPc also provides a variety of solution methods in a matrix-free way. Therefore, combining GgEm with SLEPc results in a $O(N)$ scalable parallel algorithm.

III. PARALLEL FEM STOKES FLOW SOLVER

To be able to accommodate complex confining geometries, we discretize the Stokes problem, Eq. (13), using the $P^2 - P^1$ Taylor-Hood mixed element for $\mathbf{u} - p$.[69] This is a stable element that is a staple of Stokes discretizations. The discrete Stokes equation in this basis has the following form:

$$\underbrace{\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix}}_{Q}\begin{pmatrix} \hat{\mathbf{u}} \\ \hat{p} \end{pmatrix} = \begin{pmatrix} \hat{\boldsymbol{\rho}}_g^f \\ 0 \end{pmatrix}, \tag{27}$$

where $\hat{\mathbf{u}}$, $\hat{p}$, and $\hat{\boldsymbol{\rho}}_g^f$ are the vectors of the finite element method (FEM) coefficients for the velocity, pressure, and global forcing, respectively; A is the Laplacian operator and B is the divergence operator discretized using the FEM basis, both incorporating whichever boundary conditions are imposed on the velocity and pressure, respectively.

Once the bead positions are determined and $\hat{\boldsymbol{\rho}}_g^f$ is formed, the Stokes solver determines $\hat{\mathbf{u}}$ and evaluates it at the bead positions to obtain $\mathbf{U}$. This procedure implicitly defines the

mobility tensor $\mathbf{M}$. Since $\mathbf{M}$ is applied repeatedly in the time-stepper and the Chebyshev approximations, it is crucial to have an efficient solver for Eq. (27) that can handle sufficiently fine discretizations, necessary for resolving complicated confining geometries of large-size containers.

A. Direct solver

For meshes with moderate numbers of degrees of freedom N_h, the best approach to solve Eq. (27) is to use a direct solver, such as the LU factorization $Q = L\Lambda U$, where L and U are lower- and upper-triangular, respectively, and Λ is diagonal. The advantage of this approach is that the matrix has to be factored only once, and the factors can be reused repeatedly to compute the action of M, including inside the Chebyshev operator expansion, using back- and forward-solves U and L, respectively.

The main limitations of this approach are that the L, U factors have to be explicitly computed and stored, consuming $\mathcal{O}(N_h^3)$ time and $\mathcal{O}(N_h^2)$ storage, respectively. The solves with L and U can also consume $\mathcal{O}(N_h)$ time and can introduce substantial serialization among the processors in a parallel setting. While parallel direct solver packages, such as SuperLU_Dist[70] allow for moderately large systems to be solved in reasonable time by partitioning the factor matrices among processor memories, the direct approach is fundamentally unscalable: storage requirements with $N_h \approx 5.0 \times 10^5$ exceed 16 GB, a typical RAM size for many commodity machines.

B. Iterative solver

Modern scalable parallel solvers for sparse linear systems such as Eq. (27) are usually implemented using Krylov subspace methods (KSPs). These are a family of algorithms that rely solely on the availability of a subroutine implementing the matrix-vector product Qw to solve $Qw = h$.[71] Because of this, KSP are highly parallelizable, provided an efficient *pre-conditioner* is available. Popular members of the KSP family are the conjugate-gradient (CG) method, which is applicable to symmetric positive-(in)definite systems only, and the more flexible generalized minimal residual (GMRES) method. It is useful, albeit not exactly accurate, to think of a pre-conditioner as an operator P that approximates the inverse to Q so that $PQ \approx I$ at least in the sense of a better clustering of the resulting spectrum. The Portable, Extensive Toolkit for Scientific Computation (PETSc) library implements a wide variety of KSP methods as well as a number of general-purpose pre-conditioners or makes pre-conditioners from other packages available through a uniform Application Program Interface (API). This allows the user to quickly experiment with different combinations of solvers and fine-tune the solver parameters without changing the code.

Since Q in (27) is not positive definite, CG is inapplicable, but the more flexible GMRES can work well, given a good pre-conditioner for Q. Multigrid methods and other general-purpose pre-conditioners are not very effective for Q due to its indefinite nature and a zero diagonal pressure block. However, PETSc provides effective tools for building pre-conditioners out of block methods based on decompositions such as the natural velocity-pressure split.

Generally, the most effective pre-conditioners for the discrete Stokes system[72] exploit the *Schur complement* matrix $S = -BA^{-1}B^T$, obtained from a factorization of Q,

$$\begin{pmatrix} \boldsymbol{A} & B^T \\ B & \end{pmatrix} = \begin{pmatrix} \boldsymbol{A} & \\ B\boldsymbol{A}^{-1} & I \end{pmatrix} \begin{pmatrix} \boldsymbol{A} & \\ & S \end{pmatrix} \begin{pmatrix} I & \boldsymbol{A}^{-1}B^T \\ & I \end{pmatrix}.$$

Indeed, inverting the block matrices left to right leads to an exact solution $\begin{pmatrix} x \\ y \end{pmatrix}$ to $Q\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = \begin{pmatrix} \boldsymbol{f} \\ g \end{pmatrix}$ in terms of solutions to the subsystems,

$$\boldsymbol{A}\boldsymbol{z} = \boldsymbol{f}, \quad Sy = g - B\boldsymbol{z}, \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{f} - B^T y, \tag{28}$$

defining the action of operator $\begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix} = P\begin{pmatrix} \boldsymbol{f} \\ g \end{pmatrix}$. The first and the last systems involve the solves with the discrete Laplacian, which can be effectively preconditioned by a multigrid method or, even more simply, using $ILU(0)$—an incomplete LU factorization with zero fill applied locally on each processor.[71] The middle (negatively definite) system also responds well to GMRES, provided that S can be applied efficiently without forming the matrix, and that an effective pre-conditioner for S can be obtained.

Many common pre-conditioners for Stokes are approximate solution operators P obtained by solving the three systems above *inexactly* using Krylov solvers κ on the blocks of Q. This is done by replacing $\boldsymbol{A}^{-1}$ and S^{-1} in the three systems of Eq. (28) by applications of $\kappa(\boldsymbol{A})$ and $\kappa(\hat{S})$, respectively. The Schur pre-conditioner itself is applied only approximately using $\kappa(\boldsymbol{A})$ to define

$$\hat{S}y = -B\kappa(\boldsymbol{A})B^T y.$$

Varying degrees of accuracy are required of the different κ, and frequently only a few iterations (or even one) are necessary for an effective pre-conditioner; in particular, each occurrence of $\kappa(\boldsymbol{A})$ potentially denotes a different solver.

The main difficulty is in preconditioning $\hat{S}$ in such a way that the overall solver performance does not degrade with mesh refinement. A classical approach to preconditioning S or $\hat{S}$ consists of using the *pressure mass matrix* M_p, which is simply the classical mass matrix for the pressure finite-element basis; in our case, it is the mass matrix in the P^1 basis of piecewise linear continuous functions. Mass matrices have many desirable properties; in particular, they are symmetric, positive-definite, and have positive entries. Many Stokes solvers use $-M_p$ to precondition $\hat{S}$ very successfully, since $-M_p^{-1}\hat{S}$ generally has very good spectral properties, making KSP converge rather well.

Since $-M_p$ is a good pre-conditioner for $\hat{S}$, it is spectrally similar to $\hat{S}$ and can in some sense replace $\hat{S}$. Using this observation, we were able to devise a particularly effective pre-conditioner for Eq. (27) that *replaces* $\hat{S}$ by $-M_p$. This solves the problem of preconditioning the middle system in Eq. (28), since M_p is well pre-conditioned by the diagonal lumped mass matrix $\hat{M}_p$, which contains the row sums of M_p on its diagonal. At the same time, the use of M_p eliminates the need for the relatively expensive application of $\kappa(\boldsymbol{A})$ in the definition of the action of $\hat{S}$ on vectors.

Whenever pressure boundary conditions are used, it is important to observe that the lower-right block in Eq. (27) is not

actually zero but contains rows of the negative identity matrix corresponding to the mesh nodes where this boundary condition is applied. In that case, we add those rows to $-M_p$, which remains negative-definite. In fact, we use a penalty formulation of all Dirichlet boundary conditions, which adds large negative diagonal entries to $-M_p$ and, incidentally, preserves the symmetry of $\boldsymbol{A}$. In particular, we can use CG with ILU(0) or its Cholesky variant ICC(0) for both $\kappa(\boldsymbol{A})$ and κM_p. While ILU(0) is not guaranteed to preserve the symmetry or the definiteness of the preconditioned operator, in our experience this does not cause a problem. In case CG fails to converge, the more robust GMRES can be used.

An additional simplification of P can be obtained by dropping the last of the three factors and inverting the remaining two using κ in place of inverses. This is a rather common approximation (see Ref. 72), which might increase the number of GMRES iterations needed for convergence, but makes each application of P considerably cheaper, since it omits the intermediate $\boldsymbol{z}$. Concretely, the action of our pre-conditioner P is defined as $\hat{P}\begin{pmatrix} f \\ g \end{pmatrix} = \begin{pmatrix} \boldsymbol{x} \\ y \end{pmatrix}$, where $\boldsymbol{x}$ and y are computed as follows:

$$\boldsymbol{x} = z - \kappa(\boldsymbol{A}), \quad y = -\kappa(M_p)(g - B\boldsymbol{x}).$$

The effectiveness of KSP methods depends on the particular right-hand side vector in the system being solved. For the specific case of $\begin{pmatrix} \hat{\rho}_g^f \\ 0 \end{pmatrix}$, GMRES preconditioned with the P described above proves to be extremely effective, outperforming all other iterative methods we are aware of.

Finally, we note that as defined above P is, in fact, a nonlinear operator since it contains possibly under-converged KSP applications in it. This might cause GMRES to diverge, although this has not happened in our experience. In that case, the more robust *flexible* GMRES (FGMRES) should be used (see Ref. 71).

C. Solver configuration and comparison

In our software, the selection of a direct solver or an iterative solver can be done by simply changing runtime parameters, depending on the specific demands of the problem—most commonly, the size of the computational mesh. Moreover, *all* of the iterative solver parameters can be controlled in this manner, allowing the user to fine-tune the solver to a specific problem. We provided defaults, however, that we found to be the most effective. These can be nonetheless overridden at runtime thanks to PETSc's flexible command-line options system. In particular, by default, $\kappa(\boldsymbol{A})$ above is CG preconditioned with ILU(0) on each processor's local block and solved to the relative tolerance 10^{-6} on the residual. CG is also used in $\kappa(M_p)$, likewise preconditioned with ILU(0) and to the same relative tolerance of 10^{-6}. These settings are used in all of the numerical experiments here whenever the iterative solver is employed. All of these settings can be easily overridden on the command line using the PETSc options database system.

We compared direct and iterative solvers using 128 CPU cores of Intel Xeon E5-2698v3 @ 2.3GHz on Blues at Argonne National Laboratory (ANL). We observe that, for problem sizes up to 1.2×10^6 degrees of freedom, the direct solver performs better than iterative solvers. For systems with 0.32, 0.64, and 1.2×10^6 degrees of freedom, the Stokes solution takes 0.083, 0.18, and 0.55 s, respectively, using a direct LU solver (in the preparation step, the LU decomposition takes 80, 225, and 1036 s, respectively). On the other hand, the iterative Stokes solution takes 6, 10, and 29 s, respectively, for the same system sizes. For a problem size with 2.4×10^6 degrees of freedom, the direct solver was unable to provide a one-step Stokes solve and hung in the LU decomposition process for at least 6 h. The iterative solver, for systems with 2.4, 9.7, and 16.1×10^6 degrees of freedom, took during one step Stokes solve 57, 218, and 625 s, respectively.

IV. RESULTS

Multiple levels of validations of our parallel finite element GgEm solver (pFE-GgEM) are in order. We start from the Stokes' solver precision and scalability and follow with the convergence and precision of the full BD code. We will include, however, two systems that illustrate the capability of these open-source and available routines.

A. Scalability and Stokes' flow validation

Accounting for the fact that the GgEm is a proven and validated method, we wish to verify that the parallel FEM solver is correct and that the scalability of the system will allow its implementation to large systems (even though we already pointed out that the pFE-GgEm method permits the study of confined, non-equilibrium systems).

We start by locating three point particles in an infinite cubic domain, the boundary of which is unconstrained, so that an analytic solution, using the free space Green's function or Oseen tensor, is accessible. In order to make the solutions comparable, we set the exact values on the domain boundaries evaluated from the analytic method as the boundary conditions of the numerical method. It also helps to identify whether the boundary conditions for the GgEm solution are solved appropriately. The point-forces are located in $\mathbf{x}_1 = (-5, 0, 0)$, $\mathbf{x}_2 = (0, 0, 0)$, and $\mathbf{x}_1 = (+5, 0, 0)$ in a $30 \times 30 \times 30$ cube.

We used the bead hydrodynamic radius a as the characteristic length scale. The bead diffusion time $a^2\zeta/k_BT$ is the characteristic time ($\zeta = 6\pi\mu a$ is the Stokes drag coefficient); they result in a characteristic force k_BT/a and velocity $k_BT/a\zeta$. The point-forces have a strength of $f_v = 1/3$ along the x-direction. Figure 1 shows the velocity in the x-direction along centers of the beads calculated with the GgEm and the analytical solution. The figure includes the GgEm local and global contributions for a solution with $\alpha = 0.1$ and a global mesh with a resolution of $1/\sqrt{2}\alpha$ ($15 \times 15 \times 15$ mesh). The inset shows the relative error between the solutions, where the maximum error is of order 10^{-2}.

The parallel scalability of the pFE-GgEM algorithm is tested by placing 100 beads at random in a $30 \times 30 \times 30$ domain, using $\alpha = 0.1$ and a $1/\sqrt{2}\alpha$ mesh resolution, i.e., $60 \times 60 \times 60$ mesh. The FEM mesh results in approximately 2.9×10^6 degrees of freedom from the $P^2 - P^1$ FEM formulation using hexahedron elements with 20 nodes (HEX20).[39]

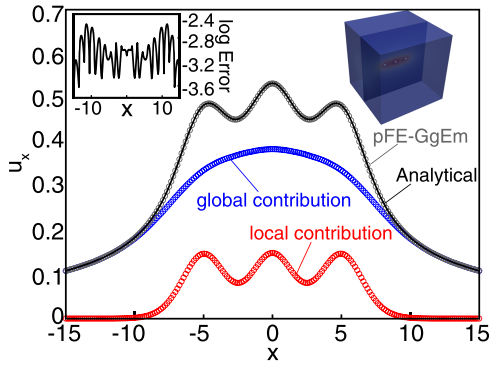

FIG. 1. Velocity in the x-direction due to three point particles driven by point forces in a $30 \times 30 \times 30$ domain. The point-forces are located in $\mathbf{x}_1 = (-5, 0, 0)$, $\mathbf{x}_2 = (0, 0, 0)$, and $\mathbf{x}_1 = (+5, 0, 0)$, with a strength of $f_v = 1/3$ along the x-direction. The GgEm solution is obtained using $\alpha = 0.1$ and a global mesh of $15 \times 15 \times 15$.

An iterative solver described in Sec. III—(F)GMRES with a custom Schur complement-based pre-conditioner—is used to solve Stokes' equations. Simulations are carried out at ANL's Laboratory Computing Resource Center (LCRC) Blues cluster, and the CPU time is measured as a function of the number of CPUs used for the calculation. Figure 2 shows the CPU time as a function of the number of CPUs for this system. As shown in the figure, the pFE-GgEm algorithm follows closely the ideal power law scaling of −1.

B. Confined DNA solutions

We now proceed to show how the pFE-GgEm algorithm translates into a full BD simulation. We start by calculating the diffusion coefficient of slit-confined DNA molecules. This simulation serves to validate the correct calculation of the diffusive terms in the stochastic differential equation. Both terms, the gradient of the diffusion tensor and the Chebyshev polynomial approximation, must be correct in order to obtain the proper diffusion coefficient and the Zimm scaling for the confined molecules.

The DNA model that we use was previously parametrized.[11,32,33,73] A DNA molecule is described by a bead-spring chain composed of N_b beads, with hydrodynamic radius a, that are connected by $N_s = N_b - 1$ worm-like springs.[74,75] The force between two connecting beads i and

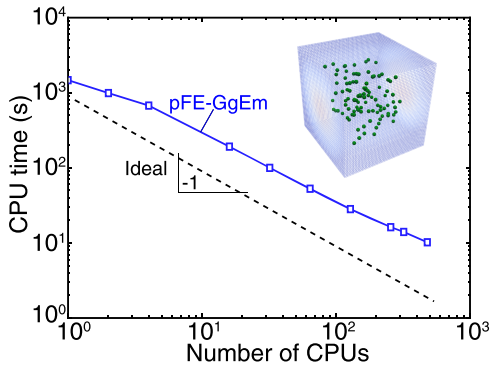

FIG. 2. CPU scalability test on ANL's LCRC Blues supercomputer. The pFE-GgEm algorithm is used to calculate the velocity field due to 100 particles that are randomly distributed in a $30 \times 30 \times 30$ domain. For the GgEm, $\alpha = 0.1$ and the mesh is $60 \times 60 \times 60$ with HEX20 elements. The total number of degrees of freedom is approximately 2.9×10^6.

j is given by

$$\mathbf{f}_{ij}^{\mathrm{wl}} = \frac{k_B T}{2b_k}\left[\left(1 - \frac{r_{ij}}{q_0}\right)^{-2} - 1 + \frac{4r_{ij}}{q_0}\right]\frac{\mathbf{x}_{ij}}{r_{ij}}, \tag{29}$$

where $\mathbf{x}_{ij} = \mathbf{x}_j - \mathbf{x}_i$, $r_{ij} = |\mathbf{x}_{ij}|$, b_k is the Kuhn length and $q_0 = N_{k,s} b_k$ is the maximum spring length ($N_{k,s}$ is the number of Kuhn segments per spring). Consequently, the contour length of the DNA molecule is $L = N_s q_0$. A DNA chain behaves as an ideal Gaussian chain at short scales (over the Kuhn segment). Therefore, a Gaussian bead-to-bead excluded volume is used as follows:[32,73]

$$U_{ij}^{\mathrm{ev}} = \frac{1}{2}\upsilon k_B T N_{k,s}^2 \left(\frac{3}{4\pi S_s^2}\right)^{3/2} \exp\left[-\frac{3r_{ij}^2}{4S_s^2}\right], \tag{30}$$

where υ is the excluded volume parameter, related to the type of solvent, and $S_s^2 = N_{k,s} b_k^2/6$ is the radius of gyration of an ideal chain consisting of $N_{k,s}$ Kuhn segments.

The confining walls impose excluded volume interactions to the DNA beads. We use an empirical bead-wall repulsive potential[11] of the type

$$U_i^{\mathrm{wall}} = \begin{cases} \dfrac{A_{\mathrm{wall}}}{3b_k \delta_{\mathrm{wall}}^2}(h - \delta_{\mathrm{wall}})^3 & \text{if } h < \delta_{\mathrm{wall}} \\ 0 & \text{if } h \geq \delta_{\mathrm{wall}} \end{cases}, \tag{31}$$

where h represents the perpendicular distance between bead i and the wall, $A_{\mathrm{wall}} = 25 k_B T$ is the strength of this repulsion, and $\delta_{\mathrm{wall}} = N_{k,s}^{1/2} b_k/2$.

Following past studies,[32] the model parameters used here are $a = 0.077$ μm, $b_k = 0.106$ μm, $\upsilon = 0.0012$ μm^3, and $N_{k,s} = 19.8$. Three different molecular weights are simulated, namely, a 21 μm ($N_s = 10$), a 42 μm ($N_s = 20$), and an 84 μm ($N_s = 40$) long DNA molecule. The molecules are confined between two parallel walls with a separation of $H = 2$ μm between them. The domain is periodic in the non-confined directions with a 20 μm period. Ten chains were randomly distributed within the domain with a volume fraction of $\phi < 0.02\%$, to ensure that the system is in the dilute regime. The GgEm parameters are $\alpha = 0.1$ and a mesh resolution of $1/\sqrt{2}\alpha$, resulting in a $37 \times 37 \times 4$ HEX20 mesh and 89 913 degrees of freedom.

We measure the center-of-mass in-plane mean squared displacement (MSD) in order to obtain the chain diffusion coefficient, i.e., $\langle \Delta x_{1,\mathrm{cm}}^2 \rangle + \langle \Delta x_{2,\mathrm{cm}}^2 \rangle = 4Dt$ (see Fig. 3). In the bulk, the diffusion coefficients of these DNA molecules are[76] $D_{\mathrm{bulk},21\,\mu\mathrm{m}} = 0.53$ μm$^2/s$, $D_{\mathrm{bulk},42\,\mu\mathrm{m}} = 0.37$ μm$^2/s$, and $D_{\mathrm{bulk},84\,\mu\mathrm{m}} = 0.22$ μm$^2/s$; the radii of gyration are $R_{g,\mathrm{bulk},21\,\mu\mathrm{m}} = 0.594$ μm, $R_{g,\mathrm{bulk},42\,\mu\mathrm{m}} = 0.85$ μm, and $R_{g,\mathrm{bulk},84\,\mu\mathrm{m}} = 1.43$ μm. Confinement and HI affect the value of the diffusion coefficient: it decreases due to the decrease of the chain mobility that is induced by the walls, and it follows a $(R_{g,\mathrm{bulk}}/H)^{-2/3}$ [76] scaling. Figure 3(a) shows typical MSD curves for the different DNA chains; the corresponding diffusion coefficients are plotted as a function of the confinement in Fig. 3(b). The calculated diffusion coefficients are $D_{\mathrm{slit},21\,\mu\mathrm{m}} = 0.3395$ μm$^2/s$, $D_{\mathrm{slit},42\,\mu\mathrm{m}} = 0.2104$ μm$^2/s$, and $D_{\mathrm{slit},84\,\mu\mathrm{m}} = 0.0713$ μm$^2/s$, which are the same as those reported in previous reports.[32,76] They follow the correct

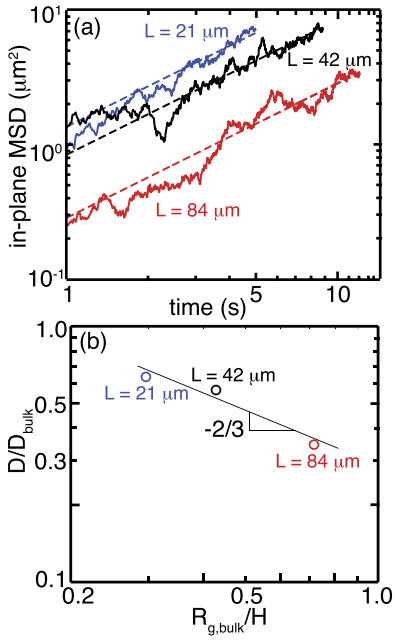

FIG. 3. (a) Typical in-plane MSDs for DNA molecules with contour length of 21 μm, 42 μm, and 84 μm confined in a slit, respectively. (b) Confined chain diffusion coefficient as a function of the confinement $R_{g,\text{bulk}}/H$.

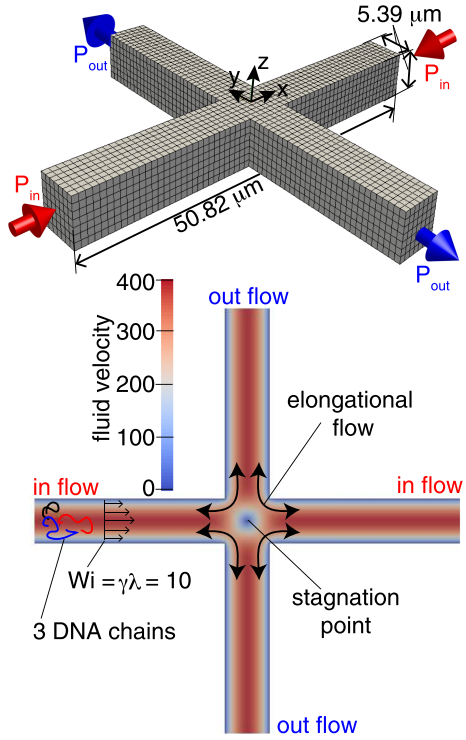

FIG. 4. Cross-channel geometry, where a Poiseuille driven flow is generated by imposing a pressure gradient between the inlet and outlet boundaries. An elongational flow, with to $Wi = \gamma\lambda = 10$, is applied to three 21 μm long DNA chains. The fluid system for the pFE-GgEm has 180 317 degrees of freedom with $\alpha = 0.1$ and HEX20 elements under a $P^2 - P^1$ scheme. We also include the contour of the magnitude of the imposed solvent velocity, $\mathbf{u}_0$, in a $z = 0$ plane, highlighting the stagnation point.

Zimm scaling,[26,64] ensuring that the "noise" and the HI are correctly calculated.

We now proceed to illustrate how the pFE-GgEm algorithm handles a complex geometry under non-equilibrium conditions. To do this, we study the behavior of 21 μm long DNA chains under an elongational flow within a cross-channel geometry. Figure 4 shows the geometry and contours of the magnitude of the imposed fluid velocity at a $z=0$ plane. To impose the elongation, a Poiseuille flow is generated by applying a pressure gradient $\Delta p = p_{\text{in}} - p_{\text{out}}$ between the inlet and outlet boundaries. A fluid viscosity $\mu = 1$ cP is used; the strength of the flow field is characterized by a Weissenberg number $Wi = \lambda\gamma$, where λ is the longest relaxation time of the DNA chain and γ is the characteristic shear rate.[30] Three individual molecules were located near an inlet boundary, and their center of mass was then followed.

We measure the instantaneous molecular "stretch" of chain v along the y-direction, $S_{y,v}$, according to

$$S_{y,\nu} = \max(\mathbf{Y}_\nu) - \min(\mathbf{Y}_\nu), \tag{32}$$

where $\mathbf{Y}_\nu$ is a N_b vector with the y-Cartesian coordinates of the beads of chain $\nu = 1, \ldots, 3$. We plot the molecular stretch as a function of time and a reaction coordinate ϕ, defined as follows:

$$\phi = \begin{cases} \dfrac{L_x}{2} + x_{\text{cm}} & \text{if} \quad x_{\text{cm}} \in \left[-\dfrac{L_x}{2}, -\dfrac{l_x}{2}\right], \\ \dfrac{L_x}{2} - x_{\text{cm}} & \text{if} \quad x_{\text{cm}} \in \left[\dfrac{l_x}{2}, \dfrac{L_x}{2}\right], \\ \dfrac{L_x}{2} - \dfrac{l_x}{2} + |y_{\text{cm}}| & \text{if} \quad x_{\text{cm}} \in \left[-\dfrac{l_x}{2}, \dfrac{l_x}{2}\right], \end{cases} \tag{33}$$

where $L_x = 50.82$ μm and $l_x = 5.39$ μm.

Figure 5 shows the instantaneous molecular stretch, along the y-direction, of the three DNA chains. Two main observations can be made from the figure. First, in the early stages ($t < 0.05$ s), the S_y magnitudes fluctuate around a value that corresponds to the initial configuration of the DNA chains. During this period, the chains are moving from the left inlet to the stagnation point under the elongational flow. Therefore, these chains are poorly stretched, and the fluctuations are mainly due to the Brownian motion and the HI. The stretch gradually increases to high values after a certain residency time. This maximum stretch, however, differs from chain to chain according to their positions along the flow lines. For instance, the chains represented by the black lines do not approach the stagnation point, and their elongation is low. On the other hand, blue and red chains manage to pass through the stagnation point, where the elongation is maximum, and their stretch is high. As a function of the reaction coordinate, the molecular stretch grows at approximately the same value of the reaction coordinate, namely $\phi = 23$ μm, which delimits the entrance of the "cross section." Within this section, these chains undergo a sharp velocity reduction and a transition from being elongated in the x-direction to being elongated in the y-direction.

C. Sedimentation of finite size particles: Immersed boundary-pFE-GgEm

Our pFE-GgEm algorithm is generalizable to treat finite size particles through the Immersed Boundary (IB) method

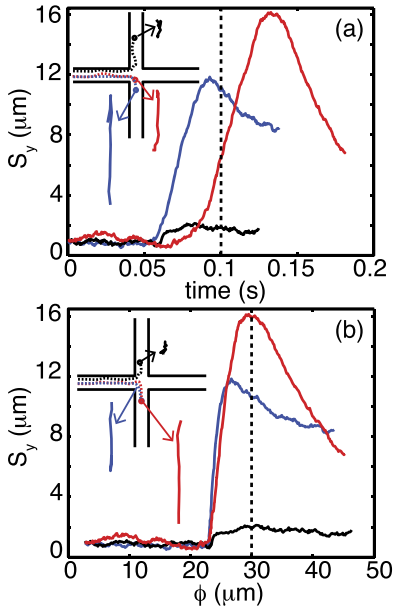

FIG. 5. Molecular stretch in the y-direction for the three DNA chains as a function of the time and the reaction coordinate ϕ. Snapshots of typical molecular arrangements are included in the inset with their corresponding locations along the cross-channel domain.

developed by Peskin.[1,77] This generalization was applied previously by Pranay *et al.*[60] for a GgEm implementation that uses finite differences and fast Fourier transforms. Here, we use a IB-pFE-GgEm that can handle confined, large-scale suspensions of finite-size particles of arbitrary shape in an arbitrary geometry.

In the IB method, the force distributions at moving solids, interfaces or membranes are discretized as distributions of regularized point-forces, where the length scale for "smoothing" the delta function scales as the grid spacing used to represent the moving entities. That is, if h is a characteristic length for the node spacing or element size on the surface, then we relate this scale with the regularization scale of the GgEm, i.e., $\xi_{\mathrm{IB}} \sim h^{-1}$. This ensures that the force density associated with each node, on the surface, is spread over the length scale of the associated elements, thereby preventing fluid from penetrating the membrane surface. Similar to conventional IB methods, in the IB-GgEm, the simulation results are insensitive to the choice of the regularization parameter if $\xi h = O(1)$. Details of the IB-GgEM can be found in the work of Pranay *et al.*[60]

As mentioned above, the forces exerted by a boundary immersed in a fluid are then modeled as a set of regularized point forces, just as is done in the regular GgEM. In particular,

$$\boldsymbol{\rho}^{f}_{\mathrm{IB}}(\mathbf{x}) = \sum_{\nu=1}^{N_{\mathrm{IB}}} \mathbf{f}^{\mathrm{C}}_{\nu}\delta_{\mathrm{IB}}(\mathbf{x}-\mathbf{x}_{\nu}), \tag{34}$$

where N_{IB} is the number of nodes (beads) that are used to represent the suspended solids, δ_{IB} is the regularization modified Gaussian that includes ξ_{IB}, and $\mathbf{f}^{\mathrm{C}}_{\nu}$ is the constitutive force that is used to describe the particles. The fluid velocity field comes from the numerical solution of Stokes' equation driven by this force density. We use the pFE-GgEm algorithm to efficiently calculate the dynamics of the particles. Recall that in traditional IB methods, the fluid mesh must be selected to resolve the ξ^{-1}_{IB} scale; using the GgEm prevents this, because the fluid mesh resolves the α^{-1} scale,[60] thereby increasing the performance of the algorithm.

We test the IB-pFE-GgEm approach by simulating the sedimentation of ten solid particles in a channel with rectangular symmetry. Seven cylinders and three spheres are initialized in a particular distribution, as shown in Fig. 6. The particles are sedimenting in a rectangular channel of size of $300\times100\times100$ due to a sedimenting force $\mathbf{g} = 0.1\boldsymbol{\delta}_1$ along the channel axis. The radius of the biggest sphere is 15. In this simulation, we kept the same "Brownian" characteristic variables for consistency: a for length, $a^2\zeta/k_BT$ for time, and k_BT/a for the force. The solids are modeled using 218 nodes for the spheres and 278 nodes for the cylinders, resulting in a total of 2600 tracking points. The pFE-GgEm mesh, for the global contribution, has 145 696 degrees of freedom, for a $\alpha = 0.1$. The node separation for the particles' surfaces is between $h_{\mathrm{min}} = 1.246\,247$

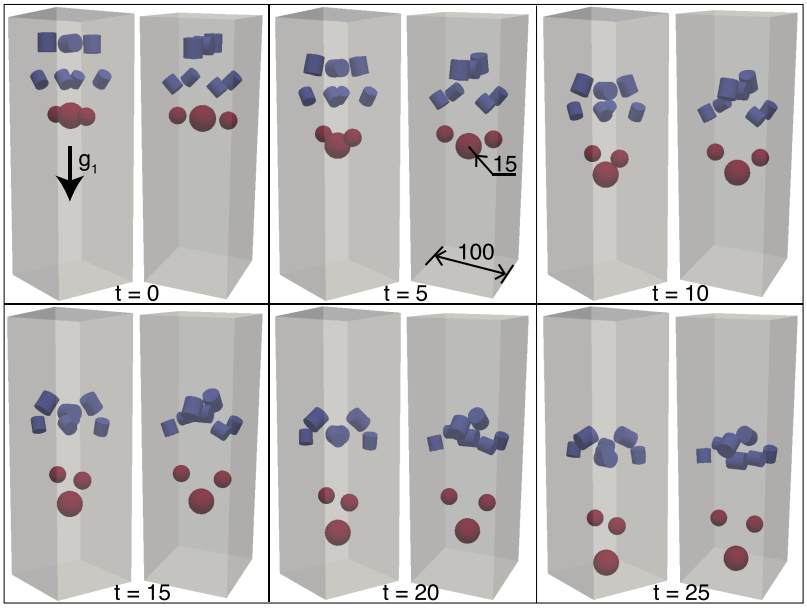

FIG. 6. Time snapshots of ten suspended finite-size particles sedimenting in a squared channel. The simulation is done using an IB-pFE-GgEm formalism. The FEM mesh, for the global contribution, has a 145 696 degrees of freedom, for a $\alpha = 0.1$ and domain size of $300\times100\times100$. The IB particle discretizations are 218 nodes for the spheres, while 278 nodes for the cylinders for a total of 2600 tracking points. The node separation for the particles' surfaces is between $h_{\mathrm{min}} = 1.246\,247$ and $h_{\mathrm{max}} = 5.402\,710$; the smoothing parameter of the IB is $1.0/0.75h_{\mathrm{min}}$.

and $h_{\max} = 5.402\,710$; the smoothing parameter of the IB is $\xi_{\mathrm{IB}} = 1.0/0.75 h_{\min}$.

We assume that the particles are "rigid," where each node point (tracking bead) on the particle is linked to its neighboring nodes by elastic springs with a prescribed large stiffness constant. In addition, all of the nodes are connected to its geometric center-of-mass point by an elastic spring. For simplicity, we assume that each link is a linear spring, and the force acting on the point i by the point j is given by

$$\mathbf{f}_{ij}^{\mathrm{C}} = k\left(r_{ij} - r_0\right)\frac{\mathbf{x}_{ij}}{r_{ij}}, \tag{35}$$

where k is the spring elastic constant and r_0 is the equilibrium spring size for each specific situation. For each particle, a spring network is formed, which generates the internal nodal force on each of the tracking beads to resist the deformation of the particles.

A time sequence of the particle's positions is shown in Fig. 6. Fluid velocity contours are also included in Fig. 7. One can appreciate that, as expected, particles located at the center of the channel sediment faster than those closer to the walls, due to the higher mobility, i.e., the fluid velocity is zero at the walls. The HI prevents the solids from sedimenting at the same rate and changes their relative orientation due to different fluid resistances. The induced fluid velocity includes regions where the velocity goes against the sedimenting direction (negative values in Fig. 7), which is a typical characteristic of sedimenting systems.[78]

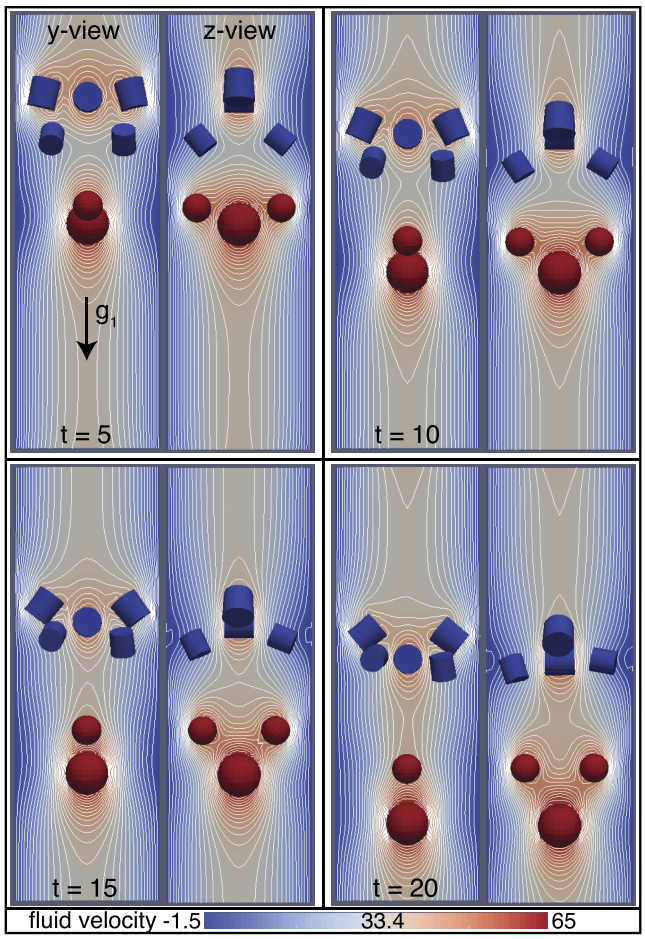

FIG. 7. Fluid velocity contours during the sedimentation of ten suspended finite-size particles in a squared channel. The contours represent the magnitude of the fluid velocity along the sedimentation direction. During the sedimentation, the hydrodynamic interactions induce fluid velocities in an opposite direction (dark blue). Solids near the center of the channel sediment at a faster rate than the solids near the walls.

V. pFE-GgEm ROUTINES AND LIBRARIES

The parallel finite element GgEm (pFE-GgEm) routines are built using open source libraries, thereby facilitating usage of our software. We list the required libraries and the repositories to download our pFE-GgEm routines:

- the finite element method is built using libMesh[79] and PETSc,[80–82]
- for systems with less than one million fluid degrees of freedom, we recommend a direct solver such as the distributed-memory SuperLU_Dist,[70,83,84]
- the scalable eigenvalue calculation uses SLEPc,[68]
- the pFE-GgEm routines can be downloaded at http://miccomcodes.org as part of the Continuum-Particle Simulation Suite (COPSS) from the Midwest Integrated Center for Computational Materials (MICCoM).

VI. CONCLUSIONS

We have developed an efficient $O(N)$ computational approach to model the dynamics of hydrodynamically interacting Brownian or micron-sized particles in arbitrary geometries. A parallel finite element Stokes' solver is the center of the algorithm. Once it is combined with the General geometry Ewald-like method (GgEm), a mid-point time integration scheme, and a Chebyshev polynomial approximation, for the fluctuation-dissipation theorem, results in a scalable algorithm—the pFE-GgEm algorithm. The approximations within these methods reduce the precision with which the fluctuation-dissipation theorem is observed. However, the Chebyshev approximation has been used extensively in confined and unconfined systems and its performance is excellent. On the other hand, the GgEm resolves Stokes equations with satisfactory precision. In particular, a 10% tolerance in the Chebyshev polynomial approximation and a proper selection of the GgEm alpha and mesh parameters result in the correct diffusion regime and diffusivities within a 5% error of the analytical or experimental values. In practice, precision is slightly sacrificed in the interest of computational performance: $O(N^3)$[85] vs. $O(N \log N)$ or $O(N)$. The finite element formulation in the pFE-GgEm has a distributed memory parallelization that may use a direct LU analytical solver or a fast parallel iterative solver with pre-conditioning.

The pFE-GgEm algorithm shows good parallel performance (linear) as a function of the number of CPUs. We validated the algorithm by comparing its solutions to theoretical results in a simple geometry (for Stokes' solver) and by calculating the diffusion coefficients of slit-confined DNA molecules (for the mid-point scheme, eigenvalues of the diffusion tensor, and the Chebyshev polynomial approximation).

It was also shown that the pFE-GgEm algorithm is capable of handling complex geometries under non-equilibrium

conditions by studying the behavior of DNA chains under an elongational flow in a cross-channel geometry. Finally, the proposed algorithm was combined with the immersed boundary method, the IB-pFE-GgEm approach, to simulate finite-size particles of arbitrary shape in any geometry. To illustrate the use of the IB-pFE-GgEm, we presented results for the sedimentation of solid particles in a rectangular channel.

We stress that the proposed algorithms are built on top of open-source libraries, such as libMesh (a framework for parallel mesh management and FEM discretization of Partial Differential Equations), PETSc (parallel linear and non-linear equation solvers), and SLEPc (eigenvalue calculations). We have provided convenient interfaces and building blocks for computational researchers that will facilitate implementations of our routines in available BD simulation codes.

ACKNOWLEDGMENTS

This work was supported by "MICCoM," as part of the Computational Materials Sciences Program funded by the U.S. Department of Energy, Office of Science, Basic Energy Sciences, Materials Sciences and Engineering Division and by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research, under Contract No. DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided on Blues and Fusion, high-performance computing clusters operated by the Laboratory Computing Resource Center at Argonne National Laboratory.

[1] T. T. Perkins, D. E. Smith, and S. Chu, Science **276**, 2016 (1997).
[2] D. E. Smith, H. P. Babcock, and S. Chu, Science **283**, 1724 (1999).
[3] K. L. Kounovsky-Shafer, J. P. Hernandez-Ortiz, K. Jo, T. Odijk, J. J. de Pablo, and D. C. Schwartz, Macromolecules **46**, 8356 (2013).
[4] C. Gupta, W. Liao, D. Gallego-Perez, C. Castro, and L. J. Lee, Biomicrofluidics **8**, 024114 (2014).
[5] C. Wang, R. L. Bruce, E. A. Duch, J. V. Patel, J. T. Smith, Y. Astier, B. H. Wunsch, S. Meshram, A. Galan, C. Scerbo, M. A. Pereira, D. Wang, E. G. Colgan, Q. Lin, and G. Stolovitzky, ACS Nano **9**, 1206 (2015).
[6] T. M. Squires and S. R. Quake, Rev. Mod. Phys. **77**, 977 (2005).
[7] A. Lenshof and T. Laurell, Chem. Soc. Rev. **39**, 1203 (2010).
[8] S. K. Bindal, G. Sethumadhavan, A. D. Nikolov, and D. T. Wasan, AIChE J. **48**, 2307 (2002).
[9] B. M. Guy, M. Hermes, and W. C. K. Poon, Phys. Rev. Lett. **115**, 088304 (2015).
[10] R. Pandey and J. C. Conrad, Soft Matter **9**, 10617 (2013).
[11] R. M. Jendrejack, D. C. Schwartz, J. J. de Pablo, and M. D. Graham, J. Chem. Phys. **120**, 2513 (2004).
[12] H. Ma and M. D. Graham, Phys. Fluids **17**, 083103 (2005).
[13] J. P. Hernandez-Ortiz, H. Ma, J. J. de Pablo, and M. D. Graham, Phys. Fluids **18**, 123101 (2006).
[14] T. Ando and J. Skolnick, Proc. Natl. Acad. Sci. U. S. A. **107**, 18457 (2010).
[15] R. J. Ellis, Trends Biochem. Sci. **26**, 597 (2001).
[16] F. Höfling and T. Franosch, Rep. Prog. Phys. **76**, 046602 (2013).
[17] H. Matsuda, G. G. Putzel, V. Backman, and I. Szleifer, Biophys. J. **106**, 1801 (2014).
[18] S. R. McGuffee and A. H. Elcock, PLoS Comput. Biol. **6**, e1000694 (2010).
[19] M. J. Morelli, R. J. Allen, and P. R. ten Wolde, Biophys. J. **101**, 2882 (2011).
[20] P. Polanowski and A. Sikorski, Soft Matter **10**, 3597 (2014).
[21] Q. Wang, K. C. Liang, A. Czader, M. N. Waxham, and M. S. Cheung, PLoS Comput. Biol. **7**, e1002114 (2011).
[22] H. Risken, *The Fokker-Planck Equation*, 2nd ed. (Springer, Berlin, Heidelberg, 1989), Vol. 18.
[23] H. C. Öttinger, *Stochastic Processes in Polymeric Fluids* (Springer-Verlag, Berlin, Heidelberg, 1996).
[24] M. Fixman, J. Chem. Phys. **69**, 1527 (1978).
[25] D. L. Ermak and J. A. McCammon, J. Chem. Phys. **69**, 1352 (1978).
[26] R. B. Bird, C. F. Curtiss, R. C. Armstrong, and O. Hassager, *Dynamics of Polymeric Liquids, Kinetic Theory*, 2nd ed. (John Wiley & Sons, Inc., New York, 1987), Vol. 2.
[27] R. M. Jendrejack, M. D. Graham, and J. J. de Pablo, J. Chem. Phys. **113**, 2894 (2000).
[28] M. Somasi, B. Khomami, N. J. Woo, J. S. Hur, and E. S. G. Shaqfeh, J. Non-Newtonian Fluid Mech. **108**, 227 (2002).
[29] J. P. Hernandez-Ortiz, J. J. de Pablo, and M. D. Graham, J. Chem. Phys. **125**, 164906 (2006).
[30] J. P. Hernandez-Ortiz, M. Chopra, S. Geier, and J. J. de Pablo, J. Chem. Phys. **131**, 044904 (2009).
[31] J. S. Hur, E. S. G. Shaqfeh, and R. G. Larson, J. Rheol. **44**, 713 (2000).
[32] R. M. Jendrejack, J. J. de Pablo, and M. D. Graham, J. Chem. Phys. **116**, 7752 (2002).
[33] R. M. Jendrejack, E. T. Dimalanta, D. C. Schwartz, M. D. Graham, and J. J. de Pablo, Phys. Rev. Lett. **91**, 038102 (2003).
[34] C. C. Hsieh, L. Li, and R. G. Larson, J. Non-Newtonian Fluid Mech. **113**, 147 (2003).
[35] A. Izmitli, D. C. Schwartz, M. D. Graham, and J. J. de Pablo, J. Chem. Phys. **128**, 085102 (2008).
[36] D. M. Heyes and J. R. Melrose, J. Non-Newtonian Fluid Mech. **46**, 1 (1993).
[37] M. Miyahara, S. Watanabe, and K. Higashitani, Chem. Eng. Sci. **61**, 2142 (2006).
[38] M. Schmidt, C. P. Royall, A. van Blaaderen, and J. Dzubiella, J. Phys.: Condens. Matter **20**, 494222 (2008).
[39] T. A. Osswald and J. P. Hernandez-Ortiz, *Polymer Processing: Modeling and Simulation* (Carl Hanser-Verlag, Munich, 2006).
[40] C. Pozrikidis, *Boundary Integral and Singularity Methods for Linearized Viscous Flow* (Cambridge University Press, Cambridge, 1992).
[41] A. J. Banchio and J. F. Brady, J. Chem. Phys. **118**, 10323 (2003).
[42] J. F. Brady and G. Bossis, Annu. Rev. Fluid Mech. **20**, 111 (1988).
[43] A. Sierou and J. F. Brady, J. Fluid Mech. **448**, 115 (2001).
[44] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond* (Oxford University Press, Oxford, 2001).
[45] S. Melchionna and S. Succi, J. Chem. Phys. **120**, 4492 (2004).
[46] R. Kekre, J. E. Butler, and A. J. C. Ladd, Phys. Rev. E **82**, 011802 (2010).
[47] P. Ahlrichs and B. Dünweg, J. Chem. Phys. **111**, 8225 (1999).
[48] M. Serrano, P. Espanol, and I. Zúñiga, Comput. Phys. Commun. **121**, 306 (1999).
[49] M. Ripoll, M. H. Ernst, and P. Espanol, J. Chem. Phys. **115**, 7271 (2001).
[50] M. R. Maxey and B. K. Patel, Int. J. Multiphase Flow **27**, 1603 (2001).
[51] R. Cortez, SIAM J. Sci. Comput. **23**, 1204 (2001).
[52] R. Cortez, L. Fauci, and A. Medovikov, Phys. Fluids **17**, 031504 (2005).
[53] J. P. Hernandez-Ortiz, J. J. de Pablo, and M. D. Graham, Phys. Rev. Lett. **98**, 140602 (2007).
[54] I. Ginzburg, F. Verhaeghe, and D. d'Humières, Commun. Comput. Phys. **3**, 427 (2008).
[55] Y. Hu, D. Li, S. Shu, and X. Niu, Int. Commun. Heat Mass Trans. **68**, 188 (2015).
[56] S. Geier, J. P. Hernandez-Ortiz, and J. J. de Pablo, Chem. Ing. Tech. **83**, 900 (2011).
[57] C. A. Miller, J. P. Hernandez-Ortiz, N. L. Abbott, S. H. Gellman, and J. J. de Pablo, J. Chem. Phys. **129**, 015102 (2008).
[58] J. P. Hernandez-Ortiz and J. J. de Pablo, J. Chem. Phys. **143**, 014108 (2015).
[59] A. Kumar and M. D. Graham, Phys. Rev. Lett. **109**, 108102 (2012).
[60] P. Pranay, S. G. Anekal, J. P. Hernandez-Ortiz, and M. D. Graham, Phys. Fluids **22**, 123103 (2010).
[61] J. P. Hernandez-Ortiz, C. G. Stoltz, and M. D. Graham, Phys. Rev. Lett. **95**, 204501 (2005).
[62] J. P. Hernandez-Ortiz, P. T. Underhill, and M. D. Graham, J. Phys.: Condens. Matter **21**, 204107 (2009).
[63] P. T. Underhill, J. P. Hernandez-Ortiz, and M. D. Graham, Phys. Rev. Lett. **100**, 248101 (2008).
[64] A. Balducci, P. Mao, J. Han, and P. S. Doyle, Macromolecules **39**, 6273 (2006).
[65] M. Fixman, Macromolecules **19**, 1195 (1986).
[66] M. Fixman, Macromolecules **19**, 1204 (1986).
[67] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (Taylor & Francis Group, New York, 1988).
[68] V. Hernandez, J. E. Roman, and V. Vidal, ACM Trans. Math. Software **31**, 351 (2005).

[69]F. Brezzi and M. Fortin, *Mixed and Hybrid Finite Element Methods* (Springer, New York, 1991).
[70]X. S. Li and J. W. Demmel, ACM Trans. Math. Software **29**, 110 (2003).
[71]Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed. (Society for Industrial and Applied Mathematics, 2003).
[72]H. C. Elman, D. Loghin, and A. J. Wathen, BIT Numer. Math. **43**, 961 (2003).
[73]R. M. Jendrejack, D. C. Schwartz, M. D. Graham, and J. J. de Pablo, J. Chem. Phys. **119**, 1165 (2003).
[74]J. F. Marko and E. D. Siggia, Macromolecules **27**, 981 (1994).
[75]J. F. Marko and E. D. Siggia, Macromolecules **28**, 8759 (1995).
[76]Y. L. Chen, M. D. Graham, J. J. de Pablo, G. C. Randall, M. Gupta, and P. S. Doyle, Phys. Rev. E **70**, 060901 (2004).
[77]C. S. Peskin, Acta Numer. **11**, 479 (2002).
[78]P. J. Mucha, S. Y. Tee, D. A. Weitz, B. I. Shraiman, and M. P. Brenner, J. Fluid Mech. **501**, 71 (2004).
[79]B. S. Kirk, J. W. Peterson, R. H. Stogner, and G. F. Carey, Eng. Comput. **22**, 237 (2006).
[80]S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, PETSc Web page, http://www.mcs.anl.gov/petsc, 2016.
[81]S. Balay, S. Abhyankar, M. F. Adams, J. Brown, P. Brune, K. Buschelman, L. Dalcin, V. Eijkhout, W. D. Gropp, D. Kaushik, M. G. Knepley, L. C. McInnes, K. Rupp, B. F. Smith, S. Zampini, H. Zhang, and H. Zhang, PETSc users manual, Technical Report No. ANL-95/11-Revision 3.7 (Argonne National Laboratory, 2016).
[82]S. Balay, W. D. Gropp, L. C. McInnes, and B. F. Smith, in *Modern Software Tools in Scientific Computing*, edited by E. Arge, A. M. Bruaset, and H. P. Langtangen (Birkhäuser Boston, 1997), pp. 163–202.
[83]J. W. Demmel, S. C. Eisenstat, J. R. Gilbert, X. S. Li, and J. W. H. Liu, SIAM J. Matrix Anal. Appl. **20**, 720 (1999).
[84]J. W. Demmel, J. R. Gilbert, and X. S. Li, SIAM J. Matrix Anal. Appl. **20**, 915 (1999).
[85]A. V. Lyulin, D. B. Adolf, and G. R. Davies, J. Chem. Phys. **111**, 758 (1999).