Fast simulation of polymer chains

Carsten Hartmann,[1,a)] Christof Schütte,[1] Galina Kalibaeva,[2] Michele Di Pierro,[2] and Giovanni Ciccotti[2]
[1]*Institut für Mathematik, Freie Universität Berlin, Arnimallee 6, 14195 Berlin, Germany*
[2]*Dipartimento di Fisica, Universita "La Sapienza," Piazzale Aldo Moro 2, 00185 Rome, Italy*

(Received 25 November 2008; accepted 10 March 2009; published online 8 April 2009)

We propose an algorithm for the fast and efficient simulation of polymers represented by chains of hard spheres. The particles are linked by holonomic bond constraints. While the motion of the polymers is free (i.e., no collisions occur) the equations of motion can be easily integrated using a collocation-based partitioned Gauss–Runge–Kutta method. The method is reversible, symplectic, and preserves energy. Moreover the numerical scheme allows the integration using much longer time steps than any explicit integrator such as the popular Verlet method. If polymers collide the point of impact can be determined to arbitrary precision by simple nested intervals. Once the collision point is known the impulsive contribution can be computed analytically. We illustrate our approach by means of a suitable numerical example. © *2009 American Institute of Physics*. [DOI: 10.1063/1.3110603]

I. INTRODUCTION

The dynamics of polymers covers processes on vastly different time scales[1] making their numerical solution a tedious and time-consuming issue. To extend simulations to more realistic times, fast and efficient algorithms are required. Computer simulation of polymers is often based on one of the two approaches: The polymer is either thought of as beads interacting by continuous confining potential or as a hard body with no internal degrees of freedom.[2,5] In this article, we represent a polymer as a chain of hard spheres linked by bond constraints with no additional restrictions on its internal degrees of freedom.

In case of a group of single hard spheres the motion of the free particles follows straight lines, and the solution of both inertial motion and collisions can be carried out analytically.[4] Until recently, it has been thought[3] that models involving hard spheres were incompatible with rigid bonds as the constraint force, except at the moment of a collision, is smooth and so it is difficult to combine algorithms for impulsive and continuous forces. In Ref. 6 it has been demonstrated how to include constraints in hard dimers, and an analytical solution for the free motion has been obtained. In this paper we will call the motion of a polymer *free*, if no collisions occur. If one tries to extend these polymer models to more than two hard spheres, an analytical solution for the free motion is no longer available, for the Lagrange multipliers become time-dependent functions. Therefore other strategies to simulate the free propagation of polymer chains need to be developed.

In this work we employ a partitioned Runge–Kutta scheme for the free propagation that is based on Gaussian collocation. The implicit Runge–Kutta scheme allows for using stable time steps that are about 400 times longer than in standard molecular dynamics (MD) simulation. The impulsive contributions due to the collisions lead to discontinuous jumps in the momenta of the interacting bodies, which can be computed analytically, provided the point of impact is known, cf. Ref. 6. Hence the problem of polymer dynamics can be split into as follows:

(1) propagation of the free polymer between two collisions and determination of the point of impact to prescribed accuracy, and
(2) solution of the collision problem between two particles (i.e., spheres).

It is interesting to note that similar problems are well known in celestial mechanics.[7,8] In contrast to MD the use of highly accurate implicit integration schemes is rather common in celestial mechanics. Moreover celestial mechanics problems require the numerical schemes to be symplectic as there is strong evidence that symplecticness goes hand by hand with good energy conservation properties and good long-time stability. This explains the popularity of symplectic implicit Runge–Kutta methods in this community.[9,10]

The outline of the article is as follows: Sec. II introduces the equations of motion of the hard spheres polymer model and explains how the equations are solved by the method of Gauss collocation. Section III is dedicated to the solution of the collision problem itself. Then, Sec. IV introduces the nested intervals method for the collision detection and its integration into the overall scheme. Finally, we illustrate the full procedure in Sec. V and illustrate it with two numerical examples.

II. FREE POLYMER MOTION

A. Hard spheres model: The freely linked chain

We are modeling a polymer as a chain of hard spheres moving in three dimensions $\mathbf{R}^3$ and linked by bond constraints. For this purpose, we denote by $\mathbf{r}_i \in \mathbf{R}^3, i=1,\ldots,N$

[a)]Author to whom correspondence should be addressed. Electronic mail: chartman@mi.fu-berlin.de.

the position of the ith particle, where N is the total number of particles. We suppose that all particles have the same mass m. If we let a be the uniform distance between two consecutive particles, the polymer configurations $\mathbf{r}=(\mathbf{r}_1,\ldots,\mathbf{r}_N)^T \in \mathbf{R}^{3N}$ are restricted to lie on the submanifold defined by

$$\sigma_k(\mathbf{r}) = |\mathbf{r}_{i_k} - \mathbf{r}_{j_k}|^2 - a^2 = 0, \quad k = 1, \ldots, M, \tag{1}$$

where $\mathcal{M}=\{(i_k,j_k)\in\{1,\ldots,N\}^2 : i_k \neq j_k, k=1,\ldots,M\}$ is the set of pairs of particle numbers that are subject to constraints. The equations of motion are as follows:

$$\begin{aligned}\dot{\mathbf{r}}(t) &= \mathbf{v}(t),\\ \dot{\mathbf{v}}(t) &= f(\mathbf{r}(t),\lambda(t)),\\ 0 &= \sigma(\mathbf{r}(t)).\end{aligned}\tag{2}$$

The acceleration can be split according to $f=f_{\mathrm{cstr}}+f_{\mathrm{imp}}$, where f_{imp} denotes the local impulsive contribution due to the collisions. It can be considered as the (generalized) gradient of the hard core potential

$$\phi(\mathbf{r}) \equiv \begin{cases} 0, & \text{if there are some } i \neq j \text{ s.t. } |\mathbf{r}_i - \mathbf{r}_j| \geq d \\ \infty, & \text{otherwise,}\end{cases}$$

where d is the diameter of the hard spheres. The part of the force acting perpendicular to the submanifold of accessible polymer configurations, defined by Eq. (1), is

$$f_{\mathrm{cstr}} = -\frac{1}{m}\nabla\sigma(\mathbf{r})^T\lambda, \tag{3}$$

and labels the acceleration due to the constraint force, where $\lambda \in \mathbf{R}^M$ is the vector of undetermined Lagrange multipliers. Note that the constraint force is not a smooth function at the collision points since the multipliers are discontinuous. Between the collisions, instead, there are no impulsive forces so that $f=f_{\mathrm{cstr}}$ is smooth.

B. Collocation method

We seek a *continuous* numerical approximation of the smooth polymer motion between two collisions, say, in the time interval $[t_0, t_0+\tau]$ with τ being sufficiently small. To this end we shall approximate the solution $(\mathbf{r},\mathbf{v},\lambda)$ of the equations of motion (2) by polynomials $\mathbf{x},\mathbf{u},\Lambda$ of degree s with vectorial coefficients that will have to be determined by the numerical scheme to be presented. That is, in the following $(\mathbf{r},\mathbf{v},\lambda)$ will always refer to the solution while $(\mathbf{x},\mathbf{u},\Lambda)$ refers to its polynomial approximation. Let us write $\mathbf{P}^s(\mathbf{R}^d)$ for the space of all polynomials of degree s with coefficients in $\mathbf{R}^d$, thus $\mathbf{x},\mathbf{u}\in\mathbf{P}^s(\mathbf{R}^{3N})$ and $\Lambda\in\mathbf{P}^s(\mathbf{R}^M)$. In order to determine the coefficients of $\mathbf{x},\mathbf{u},\Lambda$ uniquely, we need $(6N+M)\times(s+1)$ independent conditions. To specify them, collocation methods first introduce s distinct points in time,

$$t_j = t_0 + c_j\tau, \quad j = 1, \ldots, s,$$

denoting the so-called collocation points, where the nodes $0 < c_1 < \ldots < c_s \leq 1$ will be specified later on. General collocation theory[11] now asserts that the first $s(6N+M)$ unknowns are to be determined by requiring that the polynomials $\mathbf{x},\mathbf{u},\Lambda$ have to satisfy the equations of motion (2) at each collocation point t_j, i.e.,

$$\begin{aligned}\dot{\mathbf{x}}(t_0+c_j\tau) &= \mathbf{u}(t_0+c_j\tau),\\ \dot{\mathbf{u}}(t_0+c_j\tau) &= f_{\mathrm{cstr}}(\mathbf{x}(t_0+c_j\tau),\Lambda(t_0+c_j\tau)),\\ 0 &= \sigma(\mathbf{x}(t_0+c_j\tau)),\end{aligned}\tag{4}$$

with j running from 1 to s. The missing $6N+M$ conditions are then determined by the initial conditions

$$\begin{aligned}\mathbf{x}(t_0) &= \mathbf{r}(t_0),\\ \mathbf{u}(t_0) &= \mathbf{v}(t_0),\\ \Lambda(t_0) &= \lambda(t_0).\end{aligned}\tag{5}$$

Typically, the initial conditions will be the values of $(\mathbf{r},\mathbf{v})$ and λ right after the last collision. We shall expand the polynomials $\mathbf{x},\mathbf{u},\Lambda$ in the Lagrange polynomials

$$l_j(t) = \prod_{i=1,i\neq j}^{s}\frac{t-c_i}{c_j-c_i}.$$

Obviously, we have $l_j(c_k)=\delta_{jk}$ and $l_j\in\mathbf{P}^{s-1}(\mathbf{R})$, i.e., the l_j are polynomials of degree $s-1$. In fact, $\{l_j, j=1,\ldots,s\}$ is a *basis* of $\mathbf{P}^{s-1}(\mathbf{R})$. Accordingly, we can expand any polynomial of degree less than s uniquely in terms of the Lagrange basis. Since $d\mathbf{x}/dt, d\mathbf{u}/dt\in\mathbf{P}^{s-1}(\mathbf{R}^{3N})$, we thus may write

$$\dot{\mathbf{x}}(t_0+\theta\tau) = \sum_{j=1}^{s}\dot{\mathbf{x}}(t_0+c_j\tau)l_j(\theta),$$

$$\dot{\mathbf{u}}(t_0+\theta\tau) = \sum_{j=1}^{s}\dot{\mathbf{u}}(t_0+c_j\tau)l_j(\theta).$$

This is true for all $\theta\in\mathbf{R}$. Integration with respect to time yields another general formula

$$\mathbf{x}(t_0+\theta\tau) = \mathbf{x}(t_0) + \tau\int_0^\theta \dot{\mathbf{x}}(t_0+\eta\tau)d\eta,$$

$$\mathbf{u}(t_0+\theta\tau) = \mathbf{x}(t_0) + \tau\int_0^\theta \dot{\mathbf{u}}(t_0+\eta\tau)d\eta.$$

Setting $\theta=c_i$ for any $i=1,\ldots,s$, the latter expressions can be recast as

$$\begin{aligned}\mathbf{x}(t_0+c_i\tau) &= \mathbf{x}(t_0) + \tau\sum_{j=1}^{s}a_{ij}\dot{\mathbf{x}}(t_0+c_j\tau),\\ \mathbf{u}(t_0+c_i\tau) &= \mathbf{u}(t_0) + \tau\sum_{j=1}^{s}a_{ij}\dot{\mathbf{u}}(t_0+c_j\tau),\end{aligned}\tag{6}$$

where we have used the shorthand

$$a_{ij} = \int_0^{c_i} l_j(\eta) d\eta$$

for the scalar coefficients that can be computed in advance once the nodes c_j are specified. If we insert conditions (4) and (5) into Eq. (6), we obtain the following equations for the coefficients of the polynomials $\mathbf{x}, \mathbf{u}, \Lambda$:

$$X_i = \mathbf{r}(t_0) + \tau \sum_{j=1}^{s} a_{ij} U_j,$$

$$U_i = \mathbf{v}(t_0) - \tau \sum_{j=1}^{s} a_{ij} \frac{1}{m} \nabla \sigma(X_j)^T \cdot \chi_j, \tag{7}$$

$$0 = \sigma(X_j),$$

with the abbreviations $(j=1,\ldots,s)$

$$X_j = \mathbf{x}(t_0 + c_j\tau) \in \mathbf{R}^{3N},$$

$$U_j = \mathbf{u}(t_0 + c_j\tau) \in \mathbf{R}^{3N}, \tag{8}$$

$$\chi_j = \Lambda(t_0 + c_j\tau) \in \mathbf{R}^{M}.$$

The system of Eq. (7) forms a set a $s(6N+M)$ nonlinear equations for the unknowns (X_j, U_j, χ_j), $j=1,\ldots,s$. Since the first equation in Eq. (7) is linear, we can eliminate the U_j and reduce the task to solving $s(3N+M)$ nonlinear equations with $s(3N+M)$ unknowns.

As there is no analytical solution available, we have to apply an iterative method in order to solve the equations numerically; details of how this is done are given in Appendix B. For the ease of the presentation, we confine ourselves to constraints of the form

$$\sigma_i(\mathbf{r}) = |\mathbf{r}_i - \mathbf{r}_{i+1}|^2 - a^2 = 0, \quad i = 1, \ldots, N-1, \tag{9}$$

but we emphasize that our general scheme can be generalized in a straightforward fashion.

We have presented a collocation method of degree s, using s internal nodes. General collocation theory[11,12] tells us that the accuracy of the methods depends on the choice of the internal nodes c_i, $i=1,\ldots,s$. For the problem at hand (mechanical systems with constraints), the maximally achievable order of accuracy is $2s-2$ as has been proved in Ref. 13. The maximal order of accuracy is obtained if the collocation nodes are chosen to be *Gauss nodes*, and literature contains tables or algorithms for computing Gauss nodes for arbitrary s.[11,13] In this case the local error in approximating $(\mathbf{r}, \mathbf{v}, \lambda)$ by the polynomial $(\mathbf{x}, \mathbf{u}, \Lambda)$ is of the order τ^{2s-1}.

Collocation methods are contained in the much larger class of implicit Runge–Kutta methods. In application to constrained mechanical systems, as it is the case here, they are known by the name of specialized partitioned additive Runge–Kutta methods.[13] Gaussian collocation methods furthermore belong to the class of variational integrators.[14] This is important for the present work since it implies that the associated discrete flow maps are volume preserving (in fact symplectic) and reversible, and that the method presented above preserves quadratic invariants; hence both kinetic energy and the constraints are exactly preserved along the nodes (provided the nonlinear equations are solved exactly). The literature furthermore contains alternative methods that certainly are more efficient or more stable than the technique presented herein; see, e.g., Ref. 15. Such methods, however, are rather difficult to implement and have not been generalized to include collisions. Therefore, we prefer to stick to the relatively simple collocation method in order to allow for simple inclusion of collisions and to keep the following discussion transparent.

III. ANALYTICAL SOLUTION TO THE COLLISION PROBLEM

We now want to study what happens in the moment a collision takes place. To this end, let us recall the equations of motion for the velocity, viz.,

$$m\dot{\mathbf{v}} = -\nabla\sigma(\mathbf{r})^T\lambda,$$

and let us split the constraint force acting on the ith hard sphere that is involved in a collision according to

$$\nabla\sigma(\mathbf{r})^T\lambda(t) \equiv \nabla\sigma(\mathbf{r})^T(\nu(t) + \eta\delta(t)).$$

Here the rightmost term is the discontinuous part acting only during the collision. In what follows we shall use the subscripts "−" and "+" to denote discontinuous quantities immediately before and after the collision. Letting ∇_i denotes the derivative with respect to $\mathbf{r}_i$, the relation between the impulse due to the collision and the change in the linear momentum of the sphere i reads

$$m\mathbf{v}_{i,+} - m\mathbf{v}_{i,-} = -\int_{0-}^{0+} \nabla_i\sigma(\mathbf{r})\lambda_i(t)dt = -\int_{0-}^{0+} \nabla_i\sigma(\mathbf{r})\,\eta_i\delta(t)dt.$$

The smooth part does clearly not contribute to the change in momentum, while the impulsive part yields

$$\mathbf{v}_{i,+} - \mathbf{v}_{i,-} = \frac{\eta}{m}\nabla_i\sigma(\mathbf{r}). \tag{10}$$

For a single constraint $\sigma(\mathbf{r}) = |\mathbf{r}_i - \mathbf{r}_j|^2 - d^2$, we have

$$\nabla_i\sigma(\mathbf{r}) = 2(\mathbf{r}_i - \mathbf{r}_j). \tag{11}$$

Hence the generic collision contribution to the velocity of the ith particle having a rigid bond with particle j is

$$\mathbf{v}_{i,+} - \mathbf{v}_{i,-} = \mu\mathbf{r}_{ij}, \tag{12}$$

where we used the shorthand notations

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j \quad \text{and} \quad \mu = 2\eta/m$$

that will be used throughout the article. If the hard sphere is connected to more than one hard sphere by further bond constraints, we have to sum up the contributions coming from every single constraint.

A. Velocity update due to polymer collisions

Next we derive computable expressions for the velocity update of colliding hard sphere particles of polymers. Consider polymer chains consisting of N hard spheres. Let us suppose that the collision occurs between the spheres n and

m, belonging either to the same or to different polymers. Using the $\pm$ notation from above, the conservation law for the total linear momentum reads

$$m\sum_{i=1}^{N_T}\mathbf{v}_{i,+}=m\sum_{i=1}^{N_T}\mathbf{v}_{i,-}, \tag{13}$$

where N_T is equal to N in the case of intramolecular collisions or equals $2N$ in case of intermolecular collisions. Recall that all particles have the same mass; therefore the momentum law reduces to the conservation of total velocity. In the spirit of Eqs. (11) and (12) above, we obtain a linear system of equations per polymer for the postcollision velocities of the noncolliding hard spheres,

$$\begin{aligned}
\mathbf{v}_{1,+}&=\mathbf{v}_{1,-}+\mu_1\mathbf{r}_{12}\\
\mathbf{v}_{2,+}&=\mathbf{v}_{2,-}-\mu_1\mathbf{r}_{12}+\mu_2\mathbf{r}_{23}\\
&\vdots\\
\mathbf{v}_{i,+}&=\mathbf{v}_{i,-}-\mu_{i-1}\mathbf{r}_{i-1,i}+\mu_i\mathbf{r}_{i,i+1}\\
&\vdots\\
\mathbf{v}_{N-1,+}&=\mathbf{v}_{N-1,-}-\mu_{N-2}\mathbf{r}_{N-2,N-1}+\mu_{N-1}\mathbf{r}_{N-1,N}\\
\mathbf{v}_{N,+}&=\mathbf{v}_{N,-}-\mu_{N-1}\mathbf{r}_{N-1,N}.
\end{aligned} \tag{14}$$

The μ's in the system of equations are the Lagrange multipliers that are to be determined by the constraints (see below). We can write the solution in this form since all the particles (except the one colliding) feel the collision only through the constraints, and each particle is only subjected to one (extremities) or two constraints.

We also have to account for the momentum exchanged between the colliding atoms n and m, $\Delta\mathbf{V}$, as the hard sphere colliding bears the contribution from the collision, equal and opposite to the one received by the other colliding sphere, namely,

$$\begin{aligned}
\mathbf{v}_{n,+}&=\mathbf{v}_{n,-}-\mu_{n-1}\mathbf{r}_{n-1,n}+\mu_n\mathbf{r}_{n,n+1}+\Delta\mathbf{V},\\
\mathbf{v}_{m,+}&=\mathbf{v}_{m,-}-\mu_{m-1}\mathbf{r}_{m-1,m}+\mu_m\mathbf{r}_{m,m+1}-\Delta\mathbf{V},
\end{aligned} \tag{15}$$

where the spheres m and n may belong to different polymers in case of intermolecular collision or to the same polymer in case of an intramolecular collision (the net effect on the other particles is zero). Depending on the type of collision, we have either one or two sets of equations. Note that the conservation of total linear momentum, Eq. (13), is entailed by Eqs. (14) and (15).

The conservation law for the total angular momentum is

$$m\sum_{i=1}^{N_T}\mathbf{r}_i\times(\mathbf{v}_{i,+}-\mathbf{v}_{i,-})=0. \tag{16}$$

After substituting the final velocities (14) and (15) in Eq. (16), we obtain

$$0=\sum_{i=1}^{N_T}\mathbf{r}_i\times(\mathbf{v}_{i,+}-\mathbf{v}_{i,-})=(\mathbf{r}_n-\mathbf{r}_m)\times\Delta\mathbf{V},$$

which implies that $\Delta\mathbf{V}$ is parallel to the vector $(\mathbf{r}_n-\mathbf{r}_m)$,

$$\Delta\mathbf{V}=\Delta v\frac{\mathbf{r}_n-\mathbf{r}_m}{|\mathbf{r}_n-\mathbf{r}_m|}.$$

The yet unknown factor Δv will be determined later on by requiring that the energy be conserved.

Since $|\mathbf{r}_n-\mathbf{r}_m|=d$ at the moment of collision, the last equation can be equivalently expressed as

$$\Delta\mathbf{V}=\frac{\Delta v}{d}(\mathbf{r}_n-\mathbf{r}_m). \tag{17}$$

Let us now determine the μ's. The constraint $\sigma(\mathbf{r})=0$ implies the condition $\nabla\sigma(\mathbf{r})\cdot\mathbf{v}=0$ on the polymer velocity. Hence, by Eq. (11), the condition

$$(\mathbf{v}_{i,+}-\mathbf{v}_{i+1,+})\cdot\mathbf{r}_{i,i+1}=0 \tag{18}$$

must hold for all $i=1,\ldots,N-1$. Substituting Eq. (14) into Eq. (18), we obtain the following equation for the velocity constraint between the spheres i and $i+1$:

$$(\mathbf{v}_{i,-}-\mu_{i-1}\mathbf{r}_{i-1,i}+2\mu_i\,\mathbf{r}_{i,i+1}-\mu_{i+1}\mathbf{r}_{i+1,i+2}-\mathbf{v}_{i+1,-})\cdot\mathbf{r}_{i,i+1}=0.$$

A similar equation is obtained upon inserting Eq. (15) that holds for the colliding spheres. Noting that $|\mathbf{r}_{i,i+1}|^2=a^2$ and that the velocities before the collision satisfy the constraint, the last equation simplifies according to

$$\mu_{i-1}\mathbf{r}_{i-1,i}\cdot\mathbf{r}_{i,i+1}-2\mu_i a^2+\mu_{i+1}\mathbf{r}_{i+1,i+2}\cdot\mathbf{r}_{i,i+1}=0. \tag{19}$$

Going through all $N-1$ constraints in this way, we obtain a system of $N-1$ equations for $\mu=(\mu_1,\ldots,\mu_{N-1})$ and for each polymer participating in the collision (one in case of a intramolecular collisions and two otherwise). The equation for μ can be written in the form

$$K\mu=\frac{\Delta v}{d}b, \tag{20}$$

with a yet unknown proportionality factor Δv that is given in Eq. (25) below. The matrix $K\in\mathbf{R}^{(N-1)\times(N-1)}$ is symmetric tridiagonal; for systems of moderate size, its inverse can be analytically computed;[16] but also for large systems, tridiagonal systems such as Eq. (20) can be easily solved by backward substitution employing the Thomas algorithm.[17] The matrix K has the form

$$K=\begin{pmatrix} -2a^2 & \mathbf{r}_{12}\cdot\mathbf{r}_{23} & 0 & 0 & 0 & \ldots & 0\\ \mathbf{r}_{12}\cdot\mathbf{r}_{23} & -2a^2 & \mathbf{r}_{23}\cdot\mathbf{r}_{34} & 0 & 0 & \ldots & 0\\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0\\ 0 & \ldots & \mathbf{r}_{i-1,i}\cdot\mathbf{r}_{i,i+1} & -2a^2 & \mathbf{r}_{i,i+1}\cdot\mathbf{r}_{i+1,i+2} & \ldots & 0\\ 0 & \ldots & \ldots & \ldots & \ldots & \ldots & 0\\ 0 & \ldots & 0 & 0 & \mathbf{r}_{N-3,N-2}\cdot\mathbf{r}_{N-2,N-1} & -2a^2 & \mathbf{r}_{N-2,N-1}\cdot\mathbf{r}_{N-1,N}\\ 0 & \ldots & 0 & 0 & 0 & \mathbf{r}_{N-2,N-1}\cdot\mathbf{r}_{N-1,N} & -2a^2 \end{pmatrix}. \tag{21}$$

The entries of the vector $b\in\mathbf{R}^{N-1}$ are zero except for

$$b_{n-1}=(\mathbf{r}_m-\mathbf{r}_n)\cdot\mathbf{r}_{n-1,n},\qquad b_n=(\mathbf{r}_n-\mathbf{r}_m)\cdot\mathbf{r}_{n,n+1}, \tag{22}$$

and

$$b_{m-1}=(\mathbf{r}_n-\mathbf{r}_m)\cdot\mathbf{r}_{m-1,m},\qquad b_m=(\mathbf{r}_m-\mathbf{r}_n)\cdot\mathbf{r}_{m,m+1}. \tag{23}$$

Clearly, for n,m being 1 or $N-1$ only one term appears. Note that in case of intermolecular collision the elements b_{n-1} and b_n belong to the system of equations which correspond to the first polymer, whereas the elements b_{m-1} and b_m belong to the system of equations for the second polymer. In case of an intramolecular collision, all elements in Eqs. (22) and (23) are included in the same vector b in a single system of equations.

B. Conservation of energy

The unknown factor Δv in Eq. (20) can be determined by the scalar conservation law for the kinetic energy during the collision, i.e.,

$$\frac{m}{2}\sum_{i=1}^{N_T}|\mathbf{v}_{i,+}|^2=\frac{m}{2}\sum_{i=1}^{N_T}|\mathbf{v}_{i,-}|^2. \tag{24}$$

Inserting the Eqs. (14), (15), and (17) into the energy balance (24) with the Lagrange multiplier given by $\mu=\xi\Delta v/d$, where, for each polymer participating in the collision, the vector $\xi\in\mathbf{R}^{N-1}$ is the solution of

$$K\xi=b,$$

we obtain a quadratic equation for Δv. Factorizing out the trivial solution $\Delta v=0$ leaves a linear equation. Lengthy but straightforward manipulations reveal that the solution to the linear equation is given by

$$\Delta v=\frac{d}{R+d^2}(\mathbf{v}_{n,-}-\mathbf{v}_{m,-})\cdot(\mathbf{r}_m-\mathbf{r}_n), \tag{25}$$

with

$$R=a^2\sum_i\xi_i^2-\sum_i\xi_i\xi_{i+1}\mathbf{r}_{i,i+1}\cdot\mathbf{r}_{i+1,i+2}+\mathbf{r}_{nm}\cdot(\xi_n\mathbf{r}_{n,n+1}-\xi_{n-1}\mathbf{r}_{n-1,n})+\mathbf{r}_{nm}\cdot(\xi_{m-1}\mathbf{r}_{m-1,m}-\xi_m\mathbf{r}_{m,m+1}),$$

where we have taken advantage of the fact that the precollision velocities satisfy the constraints. The summations in the expression for R are over all indices i that correspond to a ξ_i (of which there are $N-1$ for an intramolecular and $2N-2$ for intermolecular collisions: the second sum has one term less). Thus, we have explicitly computed all expressions that are needed to compute all polymer velocities after a collision.

IV. NUMERICAL SOLUTION INCLUDING COLLISIONS

We now are going to generalize the collocation method introduced in Sec. II B such that collisions can be included. We approach the problem in the following way: Suppose we have computed the collocation solution $(\mathbf{x},\mathbf{u},\Lambda)$ in the interval $[t_0,t_0+\tau]$ and we ask whether a collision has occurred. If not, we proceed with the time integration. If a collision has occurred, the first collision point t_* is identified by a nested interval method based on the polynomial solution $(\mathbf{x},\mathbf{u},\Lambda)$, a new collocation solution on $[t_0,t_*]$ is computed, velocities are updated according to the previous section, and we continue with the integration in $[t_*,t_*+\tau]$.

Our approach links our specific collocation and nested intervals scheme with an analytical treatment of the velocity update at the collision points. The literature contains alternative approaches that address the entire problem by direct discretization without handling the velocity update analytically, see, e.g., Refs. 18 and 19. However, most of these approaches have not been algorithmically designed to include constraints. Moreover typical fixed time step integration schemes suffer from energy dissipation at the point of impact;[20,21] cf. also Ref. 22. We will study below as to whether this is also the case for the scheme proposed in this article.

A. Detection of collision points

Solving the collocation Eqs. (7) and (8) we have the polynomials $(\mathbf{x},\mathbf{u})$ as continuous functions in time. As the exact solution $(\mathbf{r},\mathbf{v})$ is not available, we have to identify the collision points based on its polynomial approximation. Again, letting $d>0$ denote the spheres' diameter, we call the motion of a single polymer free whenever

TABLE I. Global integration error $\|\Delta x\|_\infty$ between the reference and collocation trajectories, CPU run time T_{CPU} (arbitrary units), and the energy error $\|\Delta E\|_\infty$ for various combinations of the collocation parameters τ and s.

τ	$s=4$	$s=5$	$s=6$	$s=7$	$s=8$	$s=9$	$s=10$
0.1	1.7×10^{-4}	1.5×10^{-5}	4.1×10^{-6}	4.2×10^{-6}	4.2×10^{-6}	4.2×10^{-6}	4.2×10^{-6}
	$T_{\mathrm{CPU}}=35$	$T_{\mathrm{CPU}}=50$	$T_{\mathrm{CPU}}=60$	$T_{\mathrm{CPU}}=76$	$T_{\mathrm{CPU}}=90$	$T_{\mathrm{CPU}}=115$	$T_{\mathrm{CPU}}=135$
	6.7×10^{-6}	$4.5\cdot10^{-7}$	$2.5\cdot10^{-9}$	$3.1\cdot10^{-10}$	$3.7\cdot10^{-11}$	$6.5\cdot10^{-13}$	$3.8\cdot10^{-13}$
0.2	$4.4\cdot10^{-3}$	$1.9\cdot10^{-4}$	$4.6\cdot10^{-6}$	$4.1\cdot10^{-6}$	$4.4\cdot10^{-6}$	$4.2\cdot10^{-6}$	$4.2\cdot10^{-6}$
	$T_{\mathrm{CPU}}=23$	$T_{\mathrm{CPU}}=28$	$T_{\mathrm{CPU}}=35$	$T_{\mathrm{CPU}}=45$	$T_{\mathrm{CPU}}=54$	$T_{\mathrm{CPU}}=62$	$T_{\mathrm{CPU}}=72$
	$1.7\cdot10^{-4}$	$8.2\cdot10^{-6}$	$3.1\cdot10^{-7}$	$3.6\cdot10^{-8}$	$1.3\cdot10^{-8}$	$3.9\cdot10^{-10}$	$5\cdot10^{-11}$
0.3	$2.5\cdot10^{-2}$	$2.5\cdot10^{-4}$	$4.6\cdot10^{-5}$	$1.2\cdot10^{-5}$	$9.6\cdot10^{-6}$	$4\cdot10^{-6}$	$4.1\cdot10^{-6}$
	$T_{\mathrm{CPU}}=19$	$T_{\mathrm{CPU}}=25$	$T_{\mathrm{CPU}}=27$	$T_{\mathrm{CPU}}=35$	$T_{\mathrm{CPU}}=38$	$T_{\mathrm{CPU}}=45$	$T_{\mathrm{CPU}}=52$
	$9.3\cdot10^{-4}$	$3.4\cdot10^{-5}$	$5.9\cdot10^{-6}$	$1\cdot10^{-6}$	$4.1\cdot10^{-7}$	$2.6\cdot10^{-8}$	$6.7\cdot10^{-9}$
0.4	$7\cdot10^{-2}$	$7\cdot10^{-3}$	$3.9\cdot10^{-4}$	$1.2\cdot10^{-4}$	$1.3\cdot10^{-5}$	$3.6\cdot10^{-6}$	$2.4\cdot10^{-6}$
	$T_{\mathrm{CPU}}=18$	$T_{\mathrm{CPU}}=23$	$T_{\mathrm{CPU}}=26$	$T_{\mathrm{CPU}}=28$	$T_{\mathrm{CPU}}=32$	$T_{\mathrm{CPU}}=40$	$T_{\mathrm{CPU}}=45$
	$2.8\cdot10^{-3}$	$3.4\cdot10^{-4}$	$3.9\cdot10^{-5}$	$8.9\cdot10^{-6}$	$2.1\cdot10^{-6}$	$2.8\cdot10^{-7}$	$9.5\cdot10^{-8}$
0.5	0.105	$1.2\cdot10^{-2}$	$1.5\cdot10^{-3}$	$1.9\cdot10^{-4}$	$8.7\cdot10^{-4}$	$1.6\cdot10^{-5}$	$5.4\cdot10^{-6}$
	$T_{\mathrm{CPU}}=17$	$T_{\mathrm{CPU}}=19$	$T_{\mathrm{CPU}}=23$	$T_{\mathrm{CPU}}=26$	$T_{\mathrm{CPU}}=29$	$T_{\mathrm{CPU}}=35$	$T_{\mathrm{CPU}}=40$
	$3.1\cdot10^{-3}$	$9.3\cdot10^{-4}$	$1.5\cdot10^{-4}$	$3.3\cdot10^{-5}$	$2.7\cdot10^{-5}$	$2\cdot10^{-6}$	$4.8\cdot10^{-7}$
0.6	0.186	$3.5\cdot10^{-2}$	$2.9\cdot10^{-3}$	$7.3\cdot10^{-4}$	$3.5\cdot10^{-3}$	$1.2\cdot10^{-4}$	$5.3\cdot10^{-5}$
	$T_{\mathrm{CPU}}=17$	$T_{\mathrm{CPU}}=20$	$T_{\mathrm{CPU}}=24$	$T_{\mathrm{CPU}}=25$	$T_{\mathrm{CPU}}=29$	$T_{\mathrm{CPU}}=32$	$T_{\mathrm{CPU}}=38$
	$8.6\cdot10^{-3}$	$2.4\cdot10^{-3}$	$3.9\cdot10^{-4}$	$1\cdot10^{-4}$	$1.8\cdot10^{-4}$	$9.4\cdot10^{-6}$	$4\cdot10^{-6}$
0.7	0.256	$8.4\cdot10^{-2}$	$1.9\cdot10^{-3}$	$3.2\cdot10^{-3}$	$1.4\cdot10^{-2}$	$3.2\cdot10^{-4}$	$2.4\cdot10^{-4}$
	$T_{\mathrm{CPU}}=15$	$T_{\mathrm{CPU}}=18$	$T_{\mathrm{CPU}}=19$	$T_{\mathrm{CPU}}=25$	$T_{\mathrm{CPU}}=26$	$T_{\mathrm{CPU}}=29$	$T_{\mathrm{CPU}}=33$
	$1.1\cdot10^{-2}$	$4.9\cdot10^{-3}$	$7.6\cdot10^{-4}$	$3\cdot10^{-4}$	$7.4\cdot10^{-4}$	$3.4\cdot10^{-5}$	$2.2\cdot10^{-5}$
0.8	0.441	0.209	$7.1\cdot10^{-3}$	$4\cdot10^{-3}$	$1.5\cdot10^{-2}$	$1\cdot10^{-4}$	$6.5\cdot10^{-5}$
	$T_{\mathrm{CPU}}=14$	$T_{\mathrm{CPU}}=16$	$T_{\mathrm{CPU}}=18$	$T_{\mathrm{CPU}}=21$	$T_{\mathrm{CPU}}=25$	$T_{\mathrm{CPU}}=29$	$T_{\mathrm{CPU}}=32$
	$1.9\cdot10^{-2}$	$1.1\cdot10^{-2}$	$1.6\cdot10^{-3}$	$7.7\cdot10^{-4}$	$9\cdot10^{-4}$	$1\cdot10^{-4}$	$3.7\cdot10^{-5}$
0.9	1.73	0.516	$1.5\cdot10^{-2}$	$1.8\cdot10^{-2}$	$1.3\cdot10^{-2}$	$3.1\cdot10^{-4}$	$3.3\cdot10^{-4}$
	$T_{\mathrm{CPU}}=16$	$T_{\mathrm{CPU}}=17$	$T_{\mathrm{CPU}}=18$	$T_{\mathrm{CPU}}=23$	$T_{\mathrm{CPU}}=24$	$T_{\mathrm{CPU}}=26$	$T_{\mathrm{CPU}}=31$
	$6.3\cdot10^{-2}$	$2.6\cdot10^{-2}$	$3.7\cdot10^{-3}$	$2.1\cdot10^{-3}$	$1.9\cdot10^{-3}$	$2.9\cdot10^{-4}$	$1.2\cdot10^{-4}$
1.0	1.24	0.83	$2.8\cdot10^{-2}$	$3.4\cdot10^{-2}$		$9.5\cdot10^{-4}$	0.255
	$T_{\mathrm{CPU}}=15$	$T_{\mathrm{CPU}}=16$	$T_{\mathrm{CPU}}=18$	$T_{\mathrm{CPU}}=19$	Unstable	$T_{\mathrm{CPU}}=29$	$T_{\mathrm{CPU}}=27$
	$5.5\cdot10^{-2}$	$3.5\cdot10^{-2}$	$5.6\cdot10^{-3}$	$3.6\cdot10^{-3}$		$3.8\cdot10^{-4}$	$1.4\cdot10^{-2}$

$$|\mathbf{x}_i(t)-\mathbf{x}_j(t)| \geq d, \qquad \forall\, t\in[t_0,t_0+\tau],\ \forall\,(i,j)\in\mathcal{M}. \tag{26}$$

If this condition is violated, we employ the continuous polynomial $\mathbf{x}$ to determine the first time of collision, where we shall take advantage of the fact that both the constraints and the energy are quadratic and therefore are exactly preserved at the discrete collocation points. We adopt the following simple nesting strategy: Let t_{k+1} be the first collocation point at which Eq. (26) is violated. Now we refine the time interval either by setting $t_0=t_k$ and $\tau=t_{k+1}-t_k$ or by keeping t_0 fixed, setting $\tau=t_{k+1}-t_0$, and computing a new trajectory. By repeating these steps until the present iterate τ falls below a prescribed tolerance, we eventually determine the point of impact t_* (within the prescribed accuracy). It is advisable to decrease the polynomial degree s in the course of refinement in order to avoid condition problems (see Appendix B and Table I below).

B. Error due to collisions

In principle, the collocation method preserves quadratic invariants exactly at the collocation points. Since the collision point t_* is determined on the basis of the polynomial approximation instead of the exact solution and only up to some tolerance, it can only be an approximation of the exact collision point. However even if we had the exact collision point we would have to compute the collision update using the polynomial solution, thereby introducing an error. Now let ΔX and ΔU be the local error in the positions and the velocities at $t=t_*$. These errors then give rise to errors in the velocity updates after a collision. Due to the finite precision in the nested interval iteration, t_* will moreover not coincide with a collision point (at which, e.g., energy is conserved). As a consequence the error propagates into the (kinetic) energy. Furthermore the scheme is not symplectic at intermediate collocation points so the energy error will not be controlled by the symplecticness of the method. In the numerical examples below, we will study to what extend this lack of symplecticness introduces a drift in the energy.

V. NUMERICAL EXAMPLES

A. Self-intersecting trimer

In order to test our collocation method we simulate the simplest possible polymer chain: a trimer whose particles are allowed to self-intersect; we do not include collisions at this stage, as we shall first study the long-term stability of the scheme. For a detailed description of the collocation algorithm, we refer to Appendixes A and B.

The parameters for the polymer chain to be used in the following are $m=1$ (particle's mass), $d=1$ (particle's diameter), and $a=1$ (particle distance). The initial conditions

$$\mathbf{r}(0) = (1,0,0,0,0,0,0,1,0),$$

$$\mathbf{v}(0) = (1/2,-1/2,0,1/2,0,0,-1/2,0,0)$$

were chosen such that the constraints are satisfied. As a reference solution, we calculate a (rather expensive) trajectory employing an ordinary velocity Verlet/SHAKE algorithm. The reference solution is then compared to the collocation solution for different polynomial orders $s=4,\ldots,10$ and different time steps $\tau=0.1,\ldots,1.0$, where $\tau=1$ corresponds to 2000 integration steps of the Verlet algorithm at step size $h=5\times10^{-4}$; the Verlet time step is chosen sufficiently small so as to obtain a rather accurate reference trajectory. All times given in the following are in polymer units. The total length of the trajectory for collocation and Verlet simulation is $T=10^5$; this means that we do 2×10^8 Verlet steps.

We observe that the collocation scheme is numerically stable for all possible combinations of $\tau\leq0.8$ (1600 Verlet integration steps per collocation step) and $s\geq6$; the constraints are preserved to machine precision (the Newton tolerance was set to $\varepsilon_{\text{tol}}=10^{-12}$; see Appendix B).

Given the reference positions $\hat{x}(t_l)$ at all collocation points t_l in the integration interval $[0,T]$ from the Verlet trajectory (by interpolation if necessary), we introduce the numerical error of the collocation trajectory $\mathbf{x}$ by

$$\|\Delta\mathbf{x}\|_\infty = \max_l\{|x(t_l) - \hat{x}(t_l)|\}. \tag{27}$$

Additionally, we compute the energy error

$$\|\Delta E\|_\infty = \max_l\{|E(t_l) - E_0|\} \tag{28}$$

as the maximum deviation from the initial (kinetic) energy that is given by $2E_0=|\mathbf{v}(0)|^2$. The simulation results are presented in Table I. As the table indicates, energy is reasonably well preserved up to $\tau=0.5$ (corresponding to 1000 Verlet steps). A segment of a collocation trajectory for $\tau=0.5$ and $s=10$ is shown in Fig. 1 and compared to the reference solution. Figure 2 illustrates the conservation of kinetic energy for the same trajectory. The energy shows fluctuations around the exact value as is typical for a symplectic method. Although the collocation scheme preserves quadratic invariants in principle, we cannot expect any actual implementation to inherit this property as we use a Newton solver with

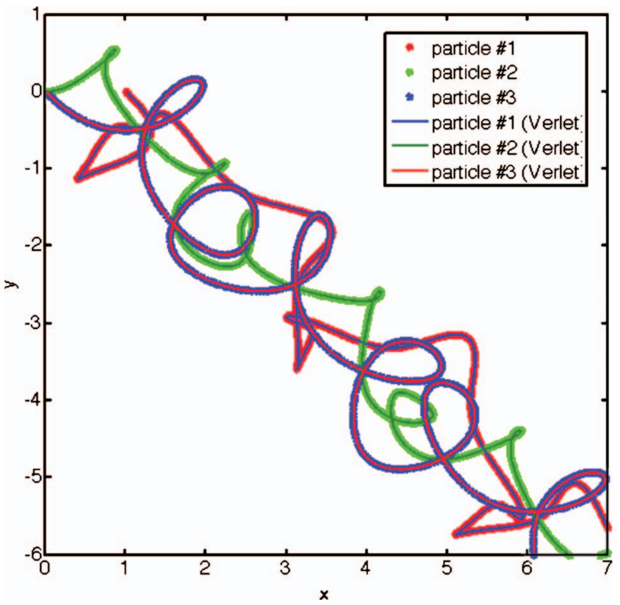

FIG. 1. (Color) Short segment of a trimer trajectory, time 0–500, with self-intersections simulated by Gauss collocation for $\tau=0.5$ and $s=10$ (stars) compared to reference trajectory obtained by the velocity Verlet algorithm (lines).

finite precision to solve the nonlinear equations. If the time step τ is chosen too large for a given collocation order s the energy starts to drift slowly.

In fact the energy criterion suggests that a safe choice are all combinations of τ and s lying strictly above the diagonal that goes from ($\tau=0.1$, $s=4$) to ($\tau=0.7$, $s=10$) in Table I. It is interesting to note that the CPU run time increases if the number of nodes s is kept fixed, while the time step τ is decreased; see the rightmost column in Table I. We suspect that this effect is due to degrading numerical condition of the Jacobi matrix in the Newton iteration since its

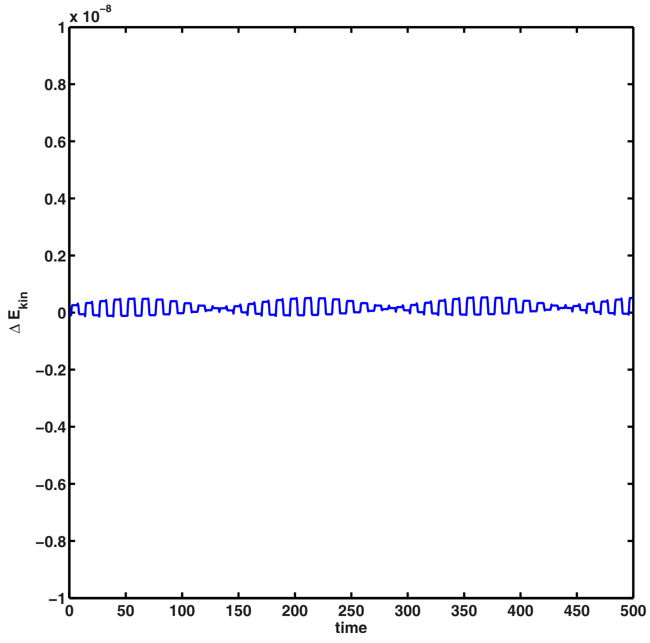

FIG. 2. (Color online) Error $\Delta E_{\text{kin}}=E-E_0$ of the total energy of a single trimer simulated by the Gauss collocation method using $s=10$ collocation points with $\tau=0.5$ (1000 Verlet steps).

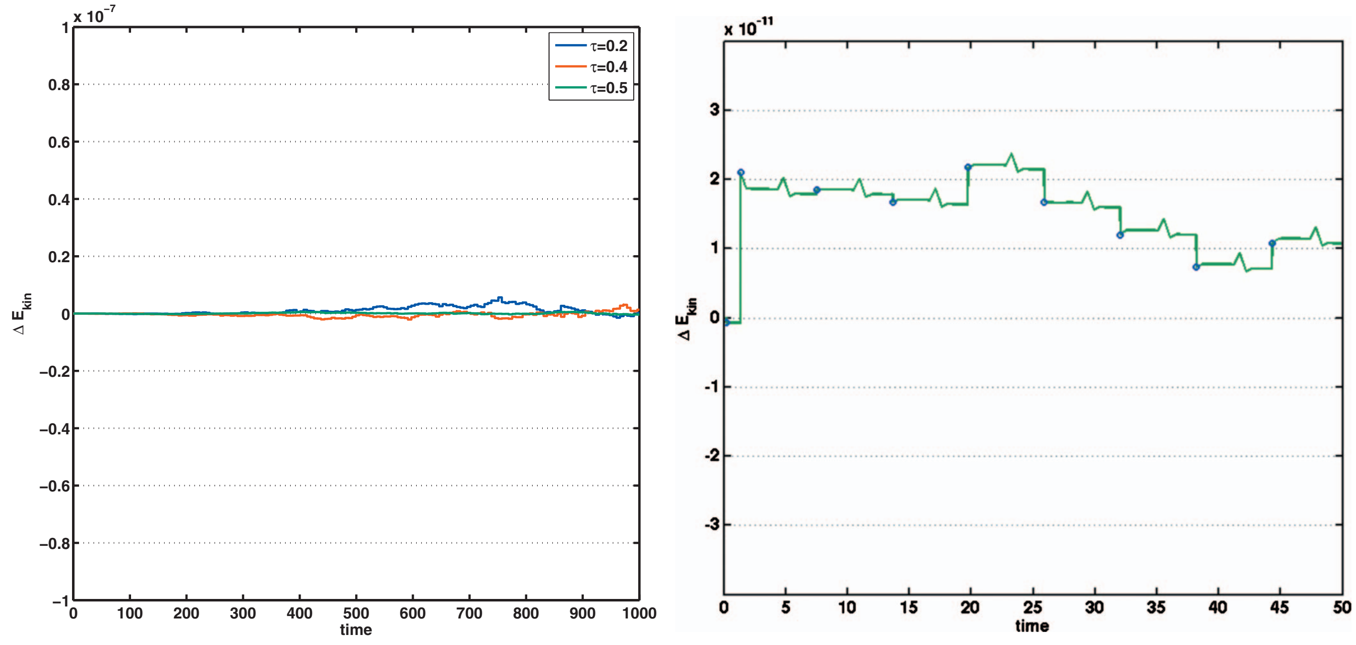

FIG. 3. (Color) Total energy. The left panel shows the energy for $s=10$ at various step sizes τ. The right panel shows a small initial segment of the entire simulation, from $t=0$ to $t=50$. It demonstrates that the sudden jumps in the energy are in fact related to the collisions (blue circles).

columns become nearly linearly dependent as the Gauss nodes become closer to each other (see Appendix B 2). Finally, the best efficiency among all simulations (in terms of energy and trajectory error compared to CPU time) is achieved by the simulations with $\tau=0.4$ and $s=9$ or $\tau=0.5$ and $s=10$.

B. Colliding trimers

To test the collision routine we simulate two colliding trimers with the parameters $a=d=1$. The collisions involve both self-collisions and collisions between the trimers. In order to verify the stability of the algorithm we check conservation of energy, angular momentum, and constraints. The linear momentum remains conserved within the machine precision. It is therefore omitted in the following. We run the algorithm with the initial conditions

$$\mathbf{r}(0)^{(1)} = (1,0,0,0,0,0,0,-1,0),$$

$$\mathbf{r}(0)^{(2)} = (2,1/2,0,3,1/2,0,4,1/2,0),$$

$$\mathbf{v}(0)^{(1)} = (1/2,0,0,1/2,0,0,0,0,0),$$

$$\mathbf{v}(0)^{(2)} = (-1/2,0,0,-1/2,0,0,-1/2,0,0).$$

Times are again given in polymer units. Given our particular choice of $s=10$, we observe the onset of drift in the angular momenta for step sizes $\tau>0.5$; the total energy does not drift though—even for step sizes beyond $\tau=0.5$. We confine our attention to step sizes $\tau\leq 0.5$ and terminate the nested intervals when $t_{k+1}-t_k\leq 10^{-4}$; further iterations are not advisable unless the number of collocation points is reduced in the course of the iterations. All other parameters for the Newton solver were left as before. The collocation trajectory is computed in the time interval $[0,T]$ with $T=10^3$ which corresponds to 2×10^6 Verlet steps at step size $h=5\times 10^{-4}$ and about 200 collisions during the simulation.

The energy behavior is shown in Fig. 3. Looking at the left panel we observe that the total energy remains stable up to $\tau=0.5$; there is no noticeable drift, yet the variance of the energy error increases linearly (but slowly) with time. Additionally the right panel shows a short initial segment of the entire simulation for $\tau=0.5$. It can be clearly seen that the energy does not oscillate but rather exhibits discrete jumps that are located at the collision points (see the discussion in Sec. IV B). A similar behavior can be observed in Fig. 4,

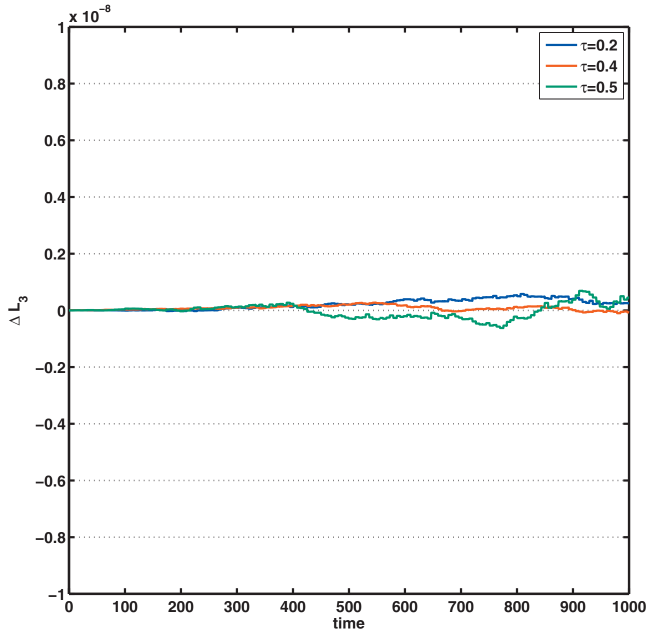

FIG. 4. (Color) Angular momentum (z-component) for different step sizes τ. Notice the staircaselike behavior of the angular momentum component; each jump occurs at a collision.

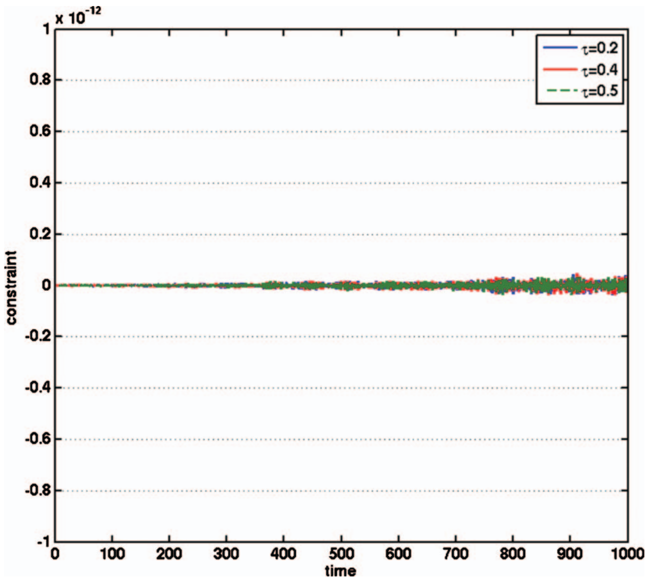

FIG. 5. (Color) Constraint $|\mathbf{r}_3-\mathbf{r}_2|^2-1=0$ for various time steps.

where the collisions lead to discrete jumps in the angular momentum. In contrast to that, the constraints show no collision-related behavior as Fig. 5 indicates for the bond constraint $|\mathbf{r}_2-\mathbf{r}_1|^2-1=0$ of polymer number one. All constraints are well within the prescribed numerical accuracy of the Newton method ($\varepsilon_{\mathrm{TOL}}=10^{-12}$).

C. Comparison with Verlet

To conclude, we compare the efficiency of our collocation method with Verlet, simulating a single trimer undergoing self-collisions. For each algorithm we determine the maximum stable step size that turns out to be $h=0.01$ for Verlet and $\tau=0.5$ for the collocation algorithm. The initial values were set to

$$\mathbf{r}(0)=(1,0,0,0,0,0,0,-1,0),$$

$$\mathbf{v}(0)=(0,1/2,0,0,0,0,-1/2,0,0).$$

A comparison of the energies is made in the left panel of Fig. 6 where we observe the typical oscillatory energy behavior of the Verlet scheme in contrast to almost perfect energy conservation of the collocation method; strictly speaking, the collocation method exhibits random energy fluctuations of the order 10^{-9}, hence about three orders of magnitude smaller than that of the Verlet method; the erratic energy jumps have already been discussed in Section IV B. In order to suppress the effect, the tolerances have to be reduced (of course, only if collisions are detected). However this will render the algorithm inefficient if collisions are frequent, thus showing a limitation of the present approach.

The right panel of Fig. 6 shows the 1–3 distance,

$$\mathrm{dist}(\mathbf{r}_1,\mathbf{r}_3)=|\mathbf{r}_1-\mathbf{r}_3|,$$

of the polymer for both Verlet and collocation as a function of time. The motion is clearly periodic, but after a few turns the trajectories show a slight phase drift that is probably caused by the finite tolerance $t_{k+1}-t_k\leq 10^{-4}$ in determining the moments of collision. Intriguingly the period of energy oscillations of the Verlet method is different from that of the collisions which suggests that the energy error is dominated by the integrator rather than by the collision algorithm (most notably, given the fact that we use the same refinement strategy for collision detection as is used in the collocation algorithm).

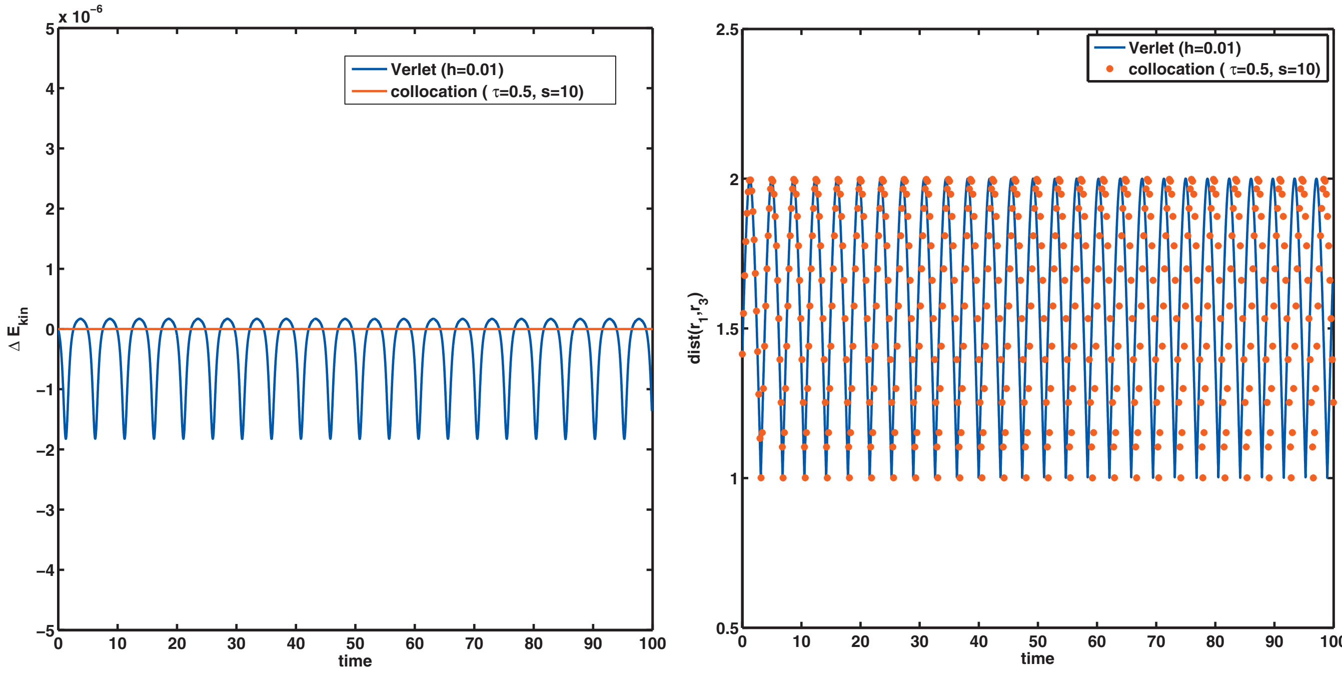

FIG. 6. (Color) Comparison between Verlet and collocation trajectories in a small initial segment of the entire simulation. Left panel: total energy. Right panel: distance between the first and the third atom.

TABLE II. CPU run time in seconds for various integration methods and different total trajectory lengths.

Total length	Collocation	Analytic	Sweeping
$T=10$	$T_{\mathrm{CPU}}=1.94$	$T_{\mathrm{CPU}}=5.76$	$T_{\mathrm{CPU}}=9.38$
$T=100$	$T_{\mathrm{CPU}}=16.55$	$T_{\mathrm{CPU}}=40.18$	$T_{\mathrm{CPU}}=78.99$
$T=1000$	$T_{\mathrm{CPU}}=151.88$	$T_{\mathrm{CPU}}=537.98$	$T_{\mathrm{CPU}}=939.61$

For the efficiency comparison between collocation and Verlet we have implemented two different variants of the Verlet (more precisely: SHAKE/RATTLE) integrator: in the traditional "sweeping" method that is due to[23] the unknown Lagrange multipliers are determined iteratively by updating the constraints one by one, whereas in the "analytic" method all Lagrange multipliers (here only 2) are updated simultaneously, employing the analytic inverse of the constraints' Jacobian; for our little toy example latter method is about two times faster than the sweeping method, but, clearly, computing the inverse is not the method of choice if the number of constraints is large. The tolerance of the SHAKE/RATTLE iteration was set to 10^{-12}, i.e., equal to the Newton tolerance $\varepsilon_{\mathrm{tol}}$. All simulations were carried out in MATLAB using a laptop computer (2.5 GHz dual core processor, 4Gbytes memory) with the results being shown in Table II below. For the single trimer, collocation clearly beats Verlet by a factor of 3–6. Interestingly enough, collocation in MATLAB automatically uses both of the available processor cores (most probably by parallelizing the matrix inversion in the Newton iteration), whereas Verlet runs only on a single core. Generally speaking, when the average size of a single polymer grows large, the implicit collocation method offers many possibilities for efficient implementations by parallelizing the chosen nonlinear solver. However even without the additional speed-up of the two processor cores, the efficiency gain for our small benchmark system would still be a factor of 1.5–3.

VI. CONCLUSIONS

We have presented a collision algorithm for the fast numerical integration of polymer chains. The scheme is a collocation-based partitioned Gauss–Runge–Kutta method with a nested intervals method to determine when an inter- or intramolecular collision takes place. For a test system of two colliding trimers, the algorithm turns out to be stable on the typical time scales of MD simulations when the time steps is up to 50 times the maximum Verlet time step. Notice that the maximum stable time step without collision was found to be about two times larger. Beyond their stability limits both Verlet and collocation method show a slight drift of the integral invariants total energy and angular momentum. Instead of the rather naive nested intervals strategy, a genuinely event-driven algorithm would require to treat the collision points as additional unknowns that are determined within the Newton iteration while using a larger integration time step. Work in this direction is currently undertaken by the authors.

APPENDIX A: COLLOCATION ALGORITHM

We propose the following collocation algorithm.

(1) Pick initial values $(\mathbf{x}_0,\mathbf{u}_0)=(\mathbf{r}(t_0),\mathbf{v}(t_0))$ that satisfy the constraints

$$\sigma(\mathbf{x}_0)=0 \quad \text{and} \quad \nabla\sigma(\mathbf{x}_0)\cdot\mathbf{u}_0=0,$$

and $\chi_0=\lambda(t_0)$ that is consistent with

$$\left.\frac{d}{dt}\right|_{t=t_0}\nabla\sigma(\mathbf{x}(t))\cdot\mathbf{u}(t)=0,$$

where $\dot{\mathbf{u}}(t)=-\nabla\sigma(\mathbf{x}(t))^T\lambda(t)$. Given $s\in\mathbf{N}$ and $\tau>0$, compute $(X_j,U_j,\chi_j)=(\mathbf{x}(t_j),\mathbf{u}(t_j),\Lambda(t_j))$ according to Eqs. (B1)–(B3) below at all Gauss nodes $t_j=t_0+c_j\tau$ with $j=1,\ldots,s$.

(2) Repeat the last steps with $t_0:=c_s\tau$ while condition (26) is met (i.e., no collisions occur).

(3) If a collision occurs between, say, nodes t_i and t_{i+1} start the integration at $t_0:=t_i$ with the new time step $\tau:=t_{i+1}-t_i$, and refine until convergence toward the point of impact t_* is reached.

(4) Compute the impulsive Lagrange multiplier μ using Eq. (20) and update the velocity $\mathbf{u}(t_*)$ according to Eqs. (14) and (15).

(5) Restart the integration from $t_0:=t_*$ with initial values $(\mathbf{x}_0,\mathbf{u}_0)=(\mathbf{x}(t_*),\mathbf{u}(t_*))$ and the corresponding $\chi_0(t_*)$.

APPENDIX B: COMPUTATIONAL ISSUES

For the numerical implementation it is convenient to recast the collocation methods (7) and (8) as an equation in the unknown positions X and the Lagrange multipliers χ only. This is done by inserting the second equation of Eq. (7) into the first one. We obtain

$$X_i=\mathbf{r}_0+\mathbf{v}_0\tau\sum_{j=1}^{s}a_{ij}-\frac{2\tau^2}{m}\sum_{j,k=1}^{s}a_{ij}a_{jk}G^TX_k\chi_k,$$

where $(\mathbf{r}_0,\mathbf{v}_0)=(\mathbf{r}(t_0),\mathbf{v}(t_0))$, and we have replaced $f=f_{\mathrm{cstr}}$, by expression (3) for the constraint acceleration upon noting that

$$\nabla\sigma(X_k)=2X_k^TG.$$

We employ a vectorial notation and understand the last expression as a shorthand for

$$X_k^TG=(X_k^TG_1,\ldots,X_k^TG_{N-1})\in\mathbf{R}^{(N-1)\times 3N},$$

with $X_k\in\mathbf{R}^{3N}$ and $G=(G_1,\ldots,G_{N-1})^T$. The G_l are $3N\times 3N$ matrices corresponding to σ_l, namely,

$$G_l = \begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & \dots & 1 & -1 & \dots & 0 \\ 0 & \dots & -1 & 1 & \dots & 0 \\ \vdots & & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix},$$

with 0 and 1 denoting 3×3 zero and unit matrices; the nonzero entries of G_l correspond to the $(l,l)-(l+1,l+1)$ blocks. Note that the collocation coefficients c_j and a_{ij} depend only on the degree s of the interpolants, and have to be computed only once. Hence we are left with the system of equations

$$0 = \underbrace{X_i - \mathbf{r}_0 - \mathbf{v}_0\tau\sum_{j=1}^{s} a_{ij} + \frac{2\tau^2}{m}\sum_{j,k=1}^{s} a_{ij}a_{jk}G^T X_k\chi_k}_{F_i^1}, \qquad 0 = \underbrace{X_i^T G X_i - a(1,\dots,1)^T}_{F_i^2}, \tag{B1}$$

with i running from 1 to s. For simplicity we abbreviate our nonlinear system of $3Ns+(N-1)s$ equations by $F(z)=0$ with $F=(F_1^1,\dots,F_s^1,F_1^2,\dots,F_s^2)$ and $z=(X,\chi)$. The equations are easily solved by means of Newton's method: Given a starting guess $z^{(0)}$, we solve the equation $F(z)=0$ by successive linearization[24]

$$\nabla F(z^{(k)})\Delta z^{(k)} = -F(z^{(k)}), \tag{B2}$$

where $k=0,1,\dots$ and

$$z^{(k+1)} = z^{(k)} + \Delta z^{(k)}. \tag{B3}$$

The analytic expression for ∇F, also called the Jacobian matrix of F, is given below. The iteration is terminated, if either the increments Δz or the function F have become sufficiently small. Here we use the most simple Newton strategy without damping. Clearly, F is not strictly convex, but as the nonlinearity is only quadratic we expect that $|F|\to 0$ (or $|\Delta z|\to 0$, respectively) in some appropriate norm $|\cdot|$ and for almost all starting values.[25] Regarding our numerical examples we found that the collocation method performs best when we use the increments Δz as termination criterion in the Newton iteration (in the maximum norm $|\Delta z|_\infty = \max_i|\Delta z_i|$) rather than the residuum $|F|_\infty$.

1. Jacobian matrix

The Jacobian of the right hand side of the $3Ns+(N-1)s$ Eq. (B1) has the following block structure:

$$\nabla F = \begin{pmatrix} I & L \\ K & 0 \end{pmatrix}.$$

Here $I=(I_{ij})\in\mathbf{R}^{3N\cdot s\times 3N\cdot s}$, $L=(L_{ij})\in\mathbf{R}^{3N\cdot s\times(N-1)\cdot s}$, and $K=(K_{ij})\in\mathbf{R}^{(N-1)\cdot s\times(N-1)\cdot s}$. The respective entries $I_{ij}\in\mathbf{R}^{3N\times 3N}$, $L_{ij}\in\mathbf{R}^{3N\times(N-1)}$ and $K_{ij}\in\mathbf{R}^{(N-1)\times 3N}$ are again matrices which read

$$\frac{\partial F_i^1}{\partial X_j} = \delta_{ij}1_{3N\times 3N} + \frac{2\tau^2}{m}\sum_{l=1}^{s} a_{il}a_{lj}G^T\chi,$$

$$\frac{\partial F_i^1}{\partial \chi_j} = \frac{2\tau^2}{m}\sum_{l=1}^{s} a_{il}a_{lj}G^T X_j,$$

$$\frac{\partial F_i^2}{\partial X_j} = 2X_j^T G\delta_{ij}.$$

2. Condition problems

In each step of the Newton method we will have to solve linear systems of equations of the form $Jz=b$. The Jacobian $J=\nabla F$ and the right hand side vector $b=F$ are functions of the approximate solutions $\mathbf{x},\mathbf{u},\Lambda$. Thus J and b contain errors, say, δJ and δb. For the solution of the linear equations, these errors induce an error[26]

$$|\delta z| \le \mathrm{cond}(J)\left(\frac{\|\delta J\|}{\|J\|} + \frac{|\delta b|}{|b|}\right)|z|$$

with $\mathrm{cond}(J)=\|J\|\|J^{-1}\|$ denoting the condition number of the matrix J. Here the norm $\|A\|$ of a matrix is defined in terms of the associated vector norm, i.e., by $\|A\|=\max_z\|Az\|$, where the maximum goes over all vectors of unit length. General collocation theory tells us that cond(J) can be large (the so-called condition problem) if the collocation points $t_0+c_i\tau$, $i=1,\dots,s$ get too dense. Thus, when the value of τ is rather small, one should choose s not too large.

[1] M. Doi and S. F. Edwards, *The Theory of Polymer Dynamics* (Clarendon, Oxford, 1986).
[2] L. H. de la Pena, R. van Zon, J. Schofield, and S. B. Opps, J. Chem. Phys. **126**, 074106 (2007).
[3] A. Scala, Th. Voigtmann, and C. De Michele, J. Chem. Phys. **126**, 134109 (2007).
[4] B. J. Alder and T. E. Wainwright, J. Chem. Phys. **31**, 459 (1959).
[5] L. H. de la Pena, R. van Zon, J. Schofield, and S. B. Opps, J. Chem. Phys. **126**, 074105 (2007).
[6] G. Ciccotti and G. Kalibaeva, J. Stat. Phys. **115**, 701 (2004).
[7] M. McNett, "Geometric integration of the collisional N-body problem," M.S. thesis, University of Kansas, 1999.
[8] I. P. Omelyan, Phys. Rev. E **74**, 036703 (2006).
[9] J. Wisdom and M. Holman, Astron. J. **102**, 1528 (1991).
[10] H. Yoshida, Celest. Mech. Dyn. Astron. **56**, 27 (1993).
[11] E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration* (Springer, Berlin, 2002).
[12] Ch. Burnton and R. Scherer, BIT **38**, 12 (1998).
[13] L. O. Jay, J. Comput. Appl. Math. **204**, 56 (2007).
[14] J. E. Marsden and M. West, Acta Numerica **9**, 357 (2001).
[15] A. Steinbrecher, "Numerical solution of quasi-linear differential- algebraic equations and industrial simulation of multibody systems," Ph.D. thesis, Technische Universität Berlin, 2006.
[16] P. Schlegel, Math. Comput. **24**, 665 (1970).
[17] L. N. Trefethen and D. Bau, *Numerical Linear Algebra* (SIAM, Philadelphia, 1997).
[18] M. Mabrouk, Eur. J. Mech. A/Solids **17**, 819 (1998).
[19] R. C. Fetecau, J. E. Marsden, M. Ortiz, and M. West, SIAM J. Appl. Dyn. Syst. **2**, 381 (2003).
[20] C. Kane, E. A. Repetto, M. Ortiz, and J. E. Marsden, Comput. Methods Appl. Mech. Eng. **180**, 1 (1999).
[21] L. Demkowicz and A. Bajer, Comput. Methods Appl. Mech. Eng. **190**, 1903 (2001).
[22] P. Deuflhard, R. Krause, and S. Ertel, Int. J. Numer. Methods Eng. **73**, 1274 (2007).

[23] J.-P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen, J. Comput. Phys. **23**, 327 (1977).
[24] P. Deuflhard, *Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms* (Springer, Berlin, 2004).
[25] M. Drexler, "Towards a global convergence theory for Newton's method," Stanford University Technical Report No. SCCM-98-06, 1998.
[26] G. Golub and C. van Loan, *Matrix Computations* (The Johns Hopkins University Press, Oxford, 1996).