Lattice Kinetic Monte-Carlo Method for Simulating Chromosomal Dynamics and other (Non-)Equilibrium Bio-Assemblies

Christiaan A. Miermans and Chase P. Broedersz

Arnold-Sommerfeld-Center for Theoretical Physics and Center for NanoScience, Ludwig-Maximilians-Universität München, D-80333 München, Germany. Corresponding author: CPB (c.broedersz@lmu.de). November 12, 2019

S1. Updating Time in Gillespie Algorithms

Below, we present a derivation for updating time in Gillespie simulations. Similar derivations can be found in e.g. [1, 2, 3, 4, 5, 6]. Consider two processes 1,2 with exponential waiting-time distributions with rates k_1, k_2. The probability distribution for an event to occur at time t is $p(1;t) = k_1 e^{-k_1 t}$, so that the probability for any event to occur in the time-interval $[0, \Delta t\rangle$ is $P(1;\Delta t) = \int_0^{\Delta t} \mathrm{d}t p(1;t) = 1 - \exp(-k_1 \Delta t)$. The probability for *either* process 1 or 2 to have occurred in the time-interval $[0, \Delta t\rangle$ is

$$\begin{aligned} P(1 \vee 2; \Delta t) &= P(\overline{\bar{1} \vee \bar{2}}; \Delta t) \\ &= 1 - P(\bar{1} \vee \bar{2}; \Delta t) \\ &= 1 - P(\bar{1}; \Delta t) P(\bar{2}; \Delta t), \end{aligned}$$

where the overline is the "not-operator". Thus, we find

$$\begin{aligned} P(1 \vee 2 \Delta t) &= 1 - e^{-k_1 \Delta t} e^{-k_2 \Delta t} \\ &= 1 - e^{-(k_1 + k_2) t}, \end{aligned}$$

i.e. the probability for either process 1 or 2 to occur is again exponentially distributed, but now with rate $K = k_1 + k_2$. The proof is analogous for more than two processes.

Given that we can sample uniformly random numbers $r \in [0, 1\rangle$, how do we generate samples that are exponentially distributed with rate K? The probability distribution for an event to occur at time t is $p(t) = K e^{-Kt}$. To transform this distribution to a distribution for $p(r)$, we use the fact that probability must be conserved in both the r or t coordinates; this implies that $p(t)|\mathrm{d}t| = p(r)|\mathrm{d}r|$. Since $p(r) = 1$ on the corresponding interval, we find $p(t) = |\mathrm{d}r/\mathrm{d}t|$ or

$$\begin{aligned} r &= \pm \int \mathrm{d}t p(t) \\ &= \mp e^{-Kt}, \end{aligned}$$

but, since r must be positive, we find $r = e^{-Kt}$. Inverting this, we find $t = -K^{-1} \log r$.

S2. Choosing the Maximal Interaction Range for the Minimal Local Update

The minimal local update (see main text) involves a set of M coordinates that are updated after each LKMC iteration. If a microscopic transition T_i displaces a particle at coordinate $\mathbf{r}$ to a coordinate $\mathbf{r}'$, then $M = M_1 \cup M_2$, where $M_1 = \{\mathbf{r} + \mathbf{v}\}_{\forall |\mathbf{v}| \leq \delta r(\mathbf{r})}$ and $M_2 = \{\mathbf{r}' + \mathbf{v}\}_{\forall |\mathbf{v}| \leq \delta r(\mathbf{r})}$, i.e. M contains all points within the *maximal interaction ranges* $\delta r(\mathbf{r}), \delta r(\mathbf{r}')$ of the coordinates $\mathbf{r}, \mathbf{r}'$. The maximal interaction ranges $\delta r(\mathbf{r}), \delta r(\mathbf{r}')$ are not necessarily equal to the interaction range of the particle displaced by T_i, but rather by the largest possible interaction range of *all* particles in the system that are affected by T_i. In our slip-link model system for example, a slip-link only ever displaces one lattice site, but the crank-shaft move can displace a monomer by two lattice sites; hence, $\delta r = 1$ if a particular move only affects slip-links but $\delta r = 2$ if a monomer can be affected by the move.

Concretely, the maximal interaction ranges before and after the move are $\delta r(\mathbf{r}) = \max\{\delta r_j \cdot \theta(|\mathbf{r} - \mathbf{r}_j| \leq \delta r_j)\}$ and $\delta r(\mathbf{r}') = \max\{\delta r_j \cdot \theta(|\mathbf{r}' - \mathbf{r}_j| \leq \delta r_j)\}$, where $\theta(\mathbf{x})$ is the three-dimensional Heaviside theta function, the $\mathbf{r}_j$ are the coordinates of all particles in the system and δr_j their corresponding interaction ranges. However, to avoid having to compute the $\delta r(\mathbf{r}), \delta r(\mathbf{r}')$ after every transition T_i in this costly manner, and for the sake of simplicity, we always use $\delta r = \max\{\delta r_j\}$. In particular, for our slip-link model system we have $\max\{\delta r_j\} = 2$ corresponding to the crank-shaft move in the Verdier-Stockmayer move-set that we use for the simulation of polymer dynamics (Figure 6a).

S3. Kinetics of Slip-Link Movement

1 Diffusion constant of slip-link with single-DOF kinetics

Suppose the elastic slip-link is composed of two sides, A, B; each side can move in two directions $+, -$. We consider a slip-link with an elastic internal energy of extension, and consider the spring constant so high that the slip-link will–at most–be extended by one lattice site. With this assumption, there are four possible ways of having a net displacement after two Monte-Carlo moves, i.e. moves such as A_+B_- are neglected. These four possible pathways are: $A_+B_+, B_+A_+, A_-B_-, B_-A_-$ (the ordering of the operators denotes the ordering of the MC moves). There are four other pathways that result in the same energy gain ΔE, namely $A_+A_-, A_-A_+, B_+B_-, B_-B_+$.

Since there are four MC pathways that produce a net displacement and four that do not, on average, we have to perform twice two MC moves to achieve one displacement ℓ_0–either to the left or to the right. The average waiting time for each of these pathways at unit temperature is $T_0 = \tau_0(1 + e^{\Delta E})$. To find the diffusion coefficient, we solve

$\langle r^2\rangle = 2Dt$ for D. As explained above, the dimer makes a displacement of size ℓ_0 after at time $t = 2T_0$. We thus find $\ell_0^2 = 4DT_0$. In sum, the diffusion constant using single-DOF kinetics is $D = \ell_0^2/[4\tau_0(1+e^{\Delta E})]$.

2 Diffusion constant of slip-link with single-DOF kinetics

Consider two particles $\alpha = a, b$ with position, velocity $\mathbf{r}_\alpha = \hat{\mathbf{x}}x_\alpha + \hat{\mathbf{y}}y_\alpha, \mathbf{v}_\alpha = \partial_t \mathbf{r}_\alpha$ experiencing overdamped kinetics and subject to a harmonic potential $V(\mathbf{r}_a, \mathbf{r}_b)$. The Langevin equations are $\gamma \mathbf{v}_\alpha = -\nabla V(\mathbf{r}_a, \mathbf{r}_b) + \boldsymbol{\eta}_\alpha$ where γ is a damping coefficient and $\boldsymbol{\eta}_\alpha$ is delta-correlated noise, $\langle \boldsymbol{\eta}_\alpha(t) \cdot \boldsymbol{\eta}_\alpha(t') = 2\gamma k_B T d\delta(t-t')$.

The forces on the particle due to this potential are $\mathbf{f}_\alpha = -\nabla_\alpha V = \pm k[\hat{\mathbf{x}}](x_b - x_a) + \hat{\mathbf{y}}(y_b - y_a)]$ with the plus, minus signs corresponding to respectively $\alpha = a, b$. We thus find $\mathbf{f}_a = -\mathbf{f}_b$, so that we can sum the two Langevin equations together to find for the center-of-mass velocity

$$\begin{aligned}\mathbf{v}_{CM} &= \frac{1}{2}(\mathbf{v}_a + \mathbf{v}_b) \\ &= \frac{1}{2\gamma}(\boldsymbol{\eta}_a + \boldsymbol{\eta}_b).\end{aligned}$$

The total displacement of the center of mass is $\mathbf{r}_{CM} = \int_0^t \mathrm{d}\tau \mathbf{v}_{CM}(\tau)$, so the mean-squared-displacement is

$$\begin{aligned}\langle \mathbf{r}_{CM}(t)^2\rangle &= \int_0^t \int_0^t \mathrm{d}\tau \mathrm{d}\tau' \langle \mathbf{v}_{CM}(\tau) \cdot \mathbf{v}_{CM}(\tau')\rangle \\ &= \frac{2}{(2\gamma)^2}\int_0^t \int_0^t \mathrm{d}\tau \mathrm{d}\tau' \langle \boldsymbol{\eta}_\alpha(\tau) \cdot \boldsymbol{\eta}_\alpha(\tau')\rangle \\ &= \frac{dk_BT}{\gamma}t.\end{aligned}$$

Combined with the Einstein relation, $\gamma = D/k_BT$ [7], this gives $\langle \mathbf{r}_{CM}^2(t)\rangle = 2dD_{CM}t$ with $D_{CM} = D/2$. Our stochastic argument based on two independent oscillators that move over a discrete lattice produces a slightly different value for the center-of-mass diffusion coefficient D_{++}, namely $D_{++} = D\log 2 \approx 0.69D$ (see main text).

3 Velocity of particle moving in potential ramp

Consider a particle moving diffusively (microscopic attempt rate k_0) inside a potential ramp with a (dimensionless) potential energy gain of $\beta\Delta E_{\mathrm{MH}}$ per step (Figure 1). We wish to calculate the relationship between velocity $\langle v\rangle$ and the steepness set by $\beta\Delta E_{\mathrm{MH}}$.

If we denote the probability of a forward, backward step by respectively p_+, p_-, then the average velocity $\langle v\rangle$ is

$$\langle v\rangle = \ell\langle k\rangle(p_+ - p_-), \tag{S1}$$

where the probabilities in Metropolis-Hastings kinetics are defined as $p_\pm = \min(1, e^{\mp\beta\Delta E_{\mathrm{MH}}})$ and where the average stepping rate is

$$\langle k\rangle = \int \mathrm{d}k\, kp(k) = K, \tag{S2}$$

where we used $p(k) = K^{-1}\exp(-k/K)$. The total rate depends on ΔE_{MH} as $K = k_0 p_- + k_0 p_+$, for which we can find a closed expression using the $p_\pm$ as defined before:

$$\begin{aligned}K =& k_0(\min(1, e^{-\beta\Delta E_{\mathrm{MH}}}) + \min(1, e^{+\beta\Delta E_{\mathrm{MH}}})) && \text{(S3)}\\ =& k_0(1 + e^{-\beta|\Delta E_{\mathrm{MH}}|}). && \text{(S4)}\end{aligned}$$

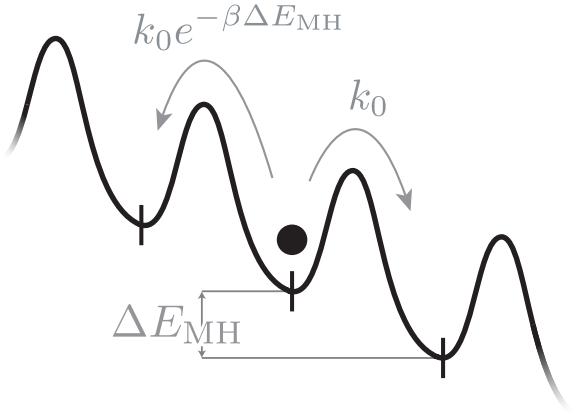

Fig. S1. Particle moving in a potential ramp according to Metropolis-Hastings kinetics. Schematically indicated is the potential ramp with jump size ΔE_{MH} per step. The microscopic attempt rate is k_0, with limits the maximum velocity towards the right, even in the limit $\beta\Delta E_{\mathrm{MH}} \to \infty$ (Equation S7).

Combining the previous results, we find the average velocity to be

$$\begin{aligned}\langle v\rangle =& \ell k_0(1 + e^{-\beta|\Delta E_{\mathrm{MH}}|})\frac{\min(1, e^{-\beta\Delta E_{\mathrm{MH}}}) - \min(1, e^{+\beta\Delta E_{\mathrm{MH}}})}{1 + e^{-\beta|\Delta E_{\mathrm{MH}}|}} && \text{(S5)}\\ =& \ell k_0\left(\min(1, e^{-\beta\Delta E_{\mathrm{MH}}}) - \min(1, e^{+\beta\Delta E_{\mathrm{MH}}})\right) && \text{(S6)}\\ =& \ell k_0 \mathrm{sgn}(\Delta E_{\mathrm{MH}})(e^{-\beta|\Delta E_{\mathrm{MH}}|} - 1). && \text{(S7)}\end{aligned}$$

We find that the maximum velocity in Metropolis-Hastings kinetics is ℓk_0, and is therefore fundamentally limited by the microscopic attempt rate k_0.

For small external driving $\beta|\Delta E_{\mathrm{MH}}| \ll 1$ we have $\exp(-\beta\Delta E_{\mathrm{MH}}) \approx 1 - \beta\Delta E_{\mathrm{MH}}$. This gives $\langle v\rangle \approx -\mathrm{sgn}(\Delta E_{\mathrm{MH}})\ell k_0 \beta|\Delta E_{\mathrm{MH}}| = -\ell k_0 \beta\Delta E_{\mathrm{MH}}$. Furthermore, we have for the diffusion coefficient $D = k_0\ell^2$ and $\Delta E_{\mathrm{MH}} = -\phi_{\mathrm{MH}}\ell$ with ϕ_{MH} the force ϕ_{MH} generated by the potential ramp. We combine these last results to find

$$\langle v\rangle \approx \beta\phi_{\mathrm{MH}} D, \tag{S8}$$

Importantly, we find that Metropolis-Hastings kinetics is consistent with the fluctuation-dissipation theorem, but only in the limit of a potential ramp with small slope $\beta|\Delta E_{\mathrm{MH}}| \ll 1$.

4 Residence time of motor stalling

Assume that the motor has two possible states: ballistic movement with velocity v and stalled motion at the end of the polymer (position $N/2$). For the sake of simplicity, we work in a regime where the motor movement attempt rate is much higher than the monomer diffusion attempt rate k_0. This means that the motor velocity is rate-limited by polymer dynamics. We have previously estimated (figure 8 in [8]) that the motor velocity in such a regime is $v \approx C^2 k_0$ with $C \approx 3/16$ the probability of two monomer bonds to be aligned. Moreover, we measured the prior probability of a motor slip-link to stall once it reaches the end of the polymer to be $p_0 \approx 0.2$ for $N = 200$ (note that p_0 may have an implicit $N-$dependency).

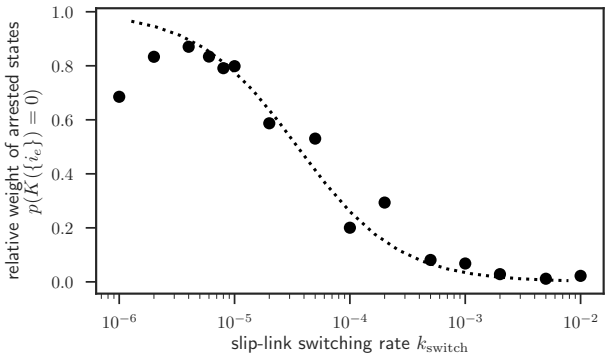

Fig. S2. The relative time that a motor slip-link spends in a stalled position depends on a balance between the direction switching time and the motor velocity. Shown are the fractional time spent in a stalled state (black circles) for various rates k_{switch} of the motor-slip switching its direction of movement. A stalled state is defined as $K(i_e) = 0$, where $K(i_e)$ is the total rate of the loop (sites i_e) that surrounds a slip-link. Dashed line: mean-field approximation (S9).

With the above assumptions, the time spent in the ballistic regime is $\langle t_{\mathrm{ballistic}}\rangle \approx N/v$. Once the motor slip-link stalls, the time spent in that stalled position is $\approx 1/k_{\mathrm{switch}}$, so the average time spent in a stalled state is $\langle t_{\mathrm{stalled}}\rangle \approx p_0/k_{\mathrm{switch}}$. An estimate for the probability of a motor to be in a stalled state θ is then simply the weighted average of these two lifetimes:

$$\theta \approx \frac{\langle t_{\mathrm{stalled}}\rangle}{\langle t_{\mathrm{ballistic}}\rangle + \langle t_{\mathrm{stalled}}\rangle} \approx \frac{p_0 v}{p_0 v + k_{\mathrm{switch}} N}. \qquad \text{(S9)}$$

This estimate fits the data quite well, except for very small k_{switch} (Figure 2). The systematic deviation stems from the way that we prepare the system: The motor slip-link is bound to the polymer without any extruded loop. This slightly biases the data.

5 Detailed Balance of Loop Kinetics

For a stochastic variable to obey detailed detailed balance, we must have $p(\Delta,\tau;\Delta',0) = p(\Delta',\tau;\Delta,0)\forall\tau$, where $p(\Delta,\tau;\Delta',0)$ is the joint probability density of transitioning from Δ' into Δ over a time-interval τ.[9] Specifically, for our slip-link model system we monitor the loop-size $\Delta(\tau)$ trapped by a slip-link (Figure 3a). We collected statistics of loop-sizes $\Delta = 1$ and $\Delta = 3$ (Figure 3b) and generated histograms of $p(1,\tau;3,0), p(3,\tau;\tau,0)$. As can be seen in Figure 3c, these distributions match within statistical error, indicating that detailed balance is obeyed.

S4. Linear Response Theory

We consider a linear polymer with end-to-end distance projected along the $z-$axis R. In equilibrium, we can define a partition function conditioned on end-to-end distance R, which in turn defines a free energy F_R. Using this free energy, we will now compute the force-extension relation for small externally applied forces (along the $z-$axis) f.

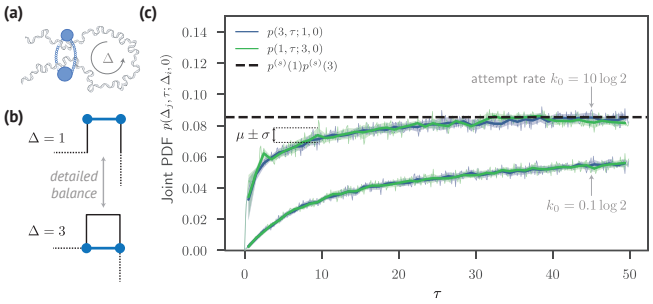

Fig. S3. Coarse-grained variables can be used to verify that our LKMC satisfies detailed balance in thermal equilibrium. (a) We monitor the size Δ of a DNA loop trapped by a slip-link. **(b)** We collected statistics of loop-sizes $\Delta = 1$ and $\Delta = 3$. **(c)** Detailed balance is obeyed if $p(1,\tau;3,0) = p(3,\tau;1,0)$ for all τ, which is indeed the case for our LKMC data within one standard deviation (blue and green curves). A lower slip-link diffusion attempt rate k_0 (compare upper and lower curves) simply increases the decorrelation time. For large times, the steady-state distributions satisfy $p(\Delta,\tau;\Delta',0) \to p^{(s)}(\Delta)p^{(s)}(\Delta')$ (dashed line) since the system is characterized by a finite amount of memory.

The conditional free energy satisfies $\beta F_R = \log Q_R$. The conditional probability distribution can be found from Q_R as $p(R) = e^{-\beta F_R}/Q_R$, so $\partial_R p(R) = -\beta\partial_R F_R \cdot p(R)$. If we now assume $p(R)$ is Gaussian distributed in R in the unperturbed ensemble, we find $\beta\partial_R F_R = R/\langle R^2\rangle$. Since $f = \langle\partial_R F_R\rangle$, $\beta f = \langle R\rangle/\langle R^2\rangle$, where $\langle R^2\rangle$ is the unperturbed variance. After plugging in $\langle R^2\rangle = \ell_0^2 N^{2\nu}$, we have an explicit relation between $f, \langle R\rangle$. For more information on the out-of-equilibrium response of polymers, we refer to the existing literature [10, 11].

We assumed that R is Gaussian distributed in the unperturbed case. This assumption is not always valid. To have linear response between the conjugate variables f, R for small R, we must have $F_R = a + bR^2 + \mathcal{O}(R)^3$. From our above calcuation, we then see that this implies that $p(R) \sim \exp(cR^2 + \mathcal{O}(R)^3)$. Normalization then sets a, b, c so that $p(R)$ is a Gaussian. In sum, linear response between f, R implies that F_R is harmonic for small f, which in turn implies that $p(R)$ is Gaussian for small R.

S5. Additional Checks on Polymer Dynamics

1 Rotational Invariance

The particles in our LKMC live on a lattice, thereby strictly speaking introducing a breaking of rotational symmetry. In order for the lattice simulation to simulate dynamics of an off-latice system (e.g. a polymer), the breaking of rotational symmetry should be negligible in the thermodynamic limit.

To test whether angular symmetry is broken, we applied a force along the $z-$axis to both ends of a linear polymer (end-to-end vector $\mathbf{R}$) and measured the azimuthal angle θ defined implicitly as $\tan\theta = (\mathbf{R}\cdot\hat{\mathbf{y}})/(\mathbf{R}\cdot\hat{\mathbf{x}})$. Indeed, we find that the histogram of θ is closely matched by a uniform distribution

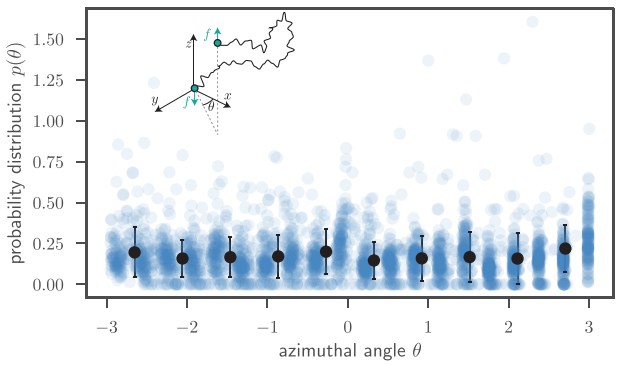

Fig. S4. Our LKMC obeys rotational invariance. We measured the azimuthal angle θ as indicated in the figure. Data was collected for polymer length $N_m = 12 \ldots 192$ and monomer overlap energy $J = 0 \ldots 20$ and generated individual histograms for each parameter set. All histograms were then combined (blue), and the average histogram of all parameter sets was computed (black, error bars: standard deviation).

(Figure 4), indicating that rotational symmetry is not broken by the lattice geometry.

2 Dependence of Polymer Relaxation Time-Scale on External Force

Linear response theory assumes that the timescales of regression of spontaneous and forced fluctuations are identical close to equilibrium, also known as *Onsager's regression hypothesis* [7]. This implies that the time-scale of relaxation from out-of-equilibrium back to equilibrium should be independent of the applied force f for small forces. Thus, if the total rate of the system is denoted $K = \sum_i k_i$, this means that K should not depend on f. We already verified that our LKMC satisfies the *fluctuation-dissipation theorem*, but we now wish to verify this independence of K on f in a more direct way.

We applied a force along the $z-$axis to linear polymers of length N and with monomer overlap energy J. The dimensionless force is $\tilde{f} = \beta|f|\ell_0 N^\nu$ with ν the fractal scaling exponent of the polymer. For very small ($N \lesssim 16$) polymers, there is a moderate depedency of K on $\tilde{f}$, but for $N \gtrsim 16$, K is no longer dependent on $\tilde{f}$ (Figure 5). This establishes that the dynamic regression of fluctuations close-to-equilibrium is identical to that in equilibrium.

3 Rouse-Like Dynamics of Self-Avoiding Chains

We assume an overdamped limit, so that we have for the hydrodynamic force $f = \zeta_N v$ onto a polymer of length N moving at a velocity v, where the friction coefficient of a chain of size N by virtue of extensivity obeys $\zeta_N \approx \zeta_1 N$. Combined with the Einstein relation, this gives $D_N = D_1/N$, where D_N, D_1 are respectively the polymer and monomer diffusion coefficients. Thus, the quantity $N\langle \mathbf{R}_{CM}(t)^2 \rangle$ should be independent of N, which is indeed the case for our LKMC (Figure 6).

Dynamics of circular self-avoiding polymers with local kinetics display Rouse dynamics with adapted static and dynamic scaling exponents as compared to dynamics of phantom chains. We already verified that the monomeric diffusion is in agreement with the these Rouse-like dynamics (see main text and Figure 7). We will now verify that not only a single monomer, but also subchains of the self-avoiding walk displays the correct dynamic scaling exponents. The intermonomeric vector $\mathbf{r}_\Delta$ of a subchain of size Δ changes over time as $\mathrm{MSD}_\Delta(t) = \langle \Delta \mathbf{r}_\Delta(t)^2 \rangle \sim N^{2\nu}$ and where the time-dependency has a longest relaxation time $\tau_N \sim N^{1+2\nu}$ [12]. For times $t \gg \tau_N$, we simply recover the static subchain scaling $\langle \mathbf{r}_\Delta^2 \rangle \sim N^{2\nu}$. Thus, a plot of $\langle \Delta \mathbf{r}(t)^2 \rangle / N^{2\nu}$ versus t/τ_N should approximately collapse all data onto a single master-curve, which is indeed the case for our LKMC (Figure 7).

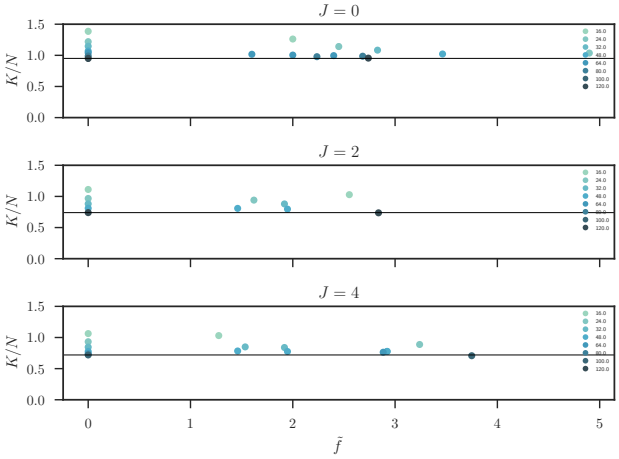

Fig. S5. Regression of fluctuations is independent of the applied force close to equilibrium. We applied a constant force f along the $z-$axis to a linear polymer of length N (indicated in legend) and measured the total rate K of the polymer dynamics. The dimensionless force is $\tilde{f} = \beta|f|\ell_0 N^\nu$. Horizontal lines were added at $\tilde{f} = 0$ for the maximum value of N used for each J to indicate that K/N does not vary with $\tilde{f}$.

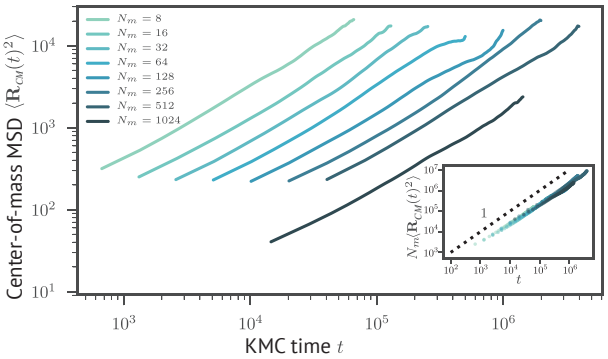

Fig. S6. Diffusion of the center of mass of a circular polymer. Center of mass $\mathbf{R}_{CM}$ for polymers of various sizes. Inset: Collapse onto universal master curve by a rescaling $D_N = D_1/N$.

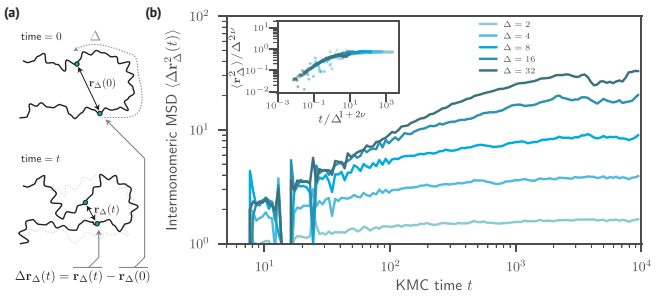

Fig. S7. Rouse dynamics of mean-squared displacement of the intermonomeric distance in a circular polymer. (a) Schematic of quantity that we measure. **(b)** Intermonomeric distance dynamics for various chain sizes Δ. **Inset:** Data collapse onto a single master-curve by a rescaling $\langle \Delta \mathbf{r}(t)^2 \rangle / N^{2\nu}$ versus t/τ_N, where τ_N is the whole-polymer relaxation time [13].

4 Statics of Chains with Partial Self-Avoidance

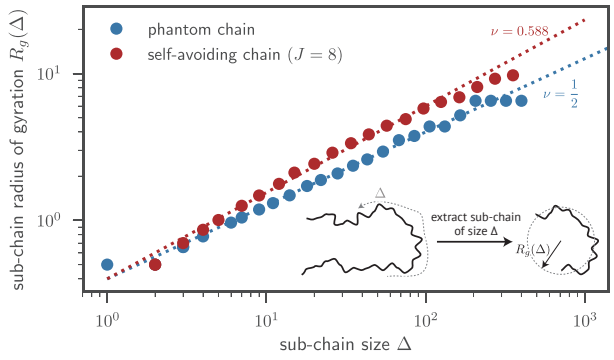

Fig. S8. Ensemble-averaged polymer statistics of polymers with partial self-avoidance are in agreement with theory. Two polymers were simulated: a phantom chain (blue) and a polymer with monomer overlap energy $J = 8$. The radius of gyration scaling of subchains within such the phantom and self-avoiding walk obey a scaling $R_g \sim \Delta^\nu$ with respectively $\nu = 1/2$ and $\nu \approx 0.588$.

We implemented monomer-monomer interactions by allowing for multiple monomers on the same lattice point $\mathbf{r}_i$. Monomer overlap was penalized with the Hamiltonian $H = \frac{1}{2} \sum_{i,j} J \delta_{\mathbf{r}_i \mathbf{r}_j}$, where $\delta_{\mathbf{a},\mathbf{b}}$ is a Kronecker delta. For $J = 0$, we recover a random walk, whereas for any $J > 0$, the polymer should self-avoiding walk statistics [14], as verified by our simulations (Figure 8).

References

[1] Arthur F. Voter. INTRODUCTION TO THE KINETIC MONTE CARLO METHOD. In *Radiation Effects in Solids*, pages 1–23. Springer Netherlands, Dordrecht, 2007.

[2] Peter Kratzer. Monte carlo and kinetic monte carlo methods – a tutorial. Technical report, 2009.

[3] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[4] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403 – 434, 1976.

[5] Kristen A. Fichthorn and W. H. Weinberg. Theoretical foundations of dynamical Monte Carlo simulations. *The Journal of Chemical Physics*, 95(2):1090–1096, jul 1991.

[6] Daniel T. Gillespie, Andreas Hellander, and Linda R. Petzold. Perspective: Stochastic algorithms for chemical kinetics. *The Journal of Chemical Physics*, 138(17):170901, 2013.

[7] L. E. Reichl. *A Modern Course in Statistical Physics*, volume 67. Wiley, 4 edition, 2016.

[8] Christiaan A. Miermans and Chase P. Broedersz. Bacterial chromosome organization by collective dynamics of SMC condensins. *Journal of The Royal Society Interface*, 15(147):20180495, oct 2018.

[9] Crispin Gardiner. *Stochastic Methods: A Handbook for the Natural and Social Sciences*. Springer, Berlin, 2009.

[10] Michael Rubinstein and Ralph H. Colby. *Polymer Physics*. Oxford University Press, 2003.

[11] M. Doi and S. F. Edwards. *The Theory of Polymer Dynamics*. Clarendon Press, Oxford, 1986.

[12] Debabrata Panja and Gerard T Barkema. Passage Times for Polymer Translocation Pulled through a Narrow Pore. *Biophysical Journal*, 94, 2008.

[13] Debabrata Panja and Gerard T. Barkema. Rouse Modes of Self-avoiding Flexible Polymers. *The Journal of Chemical Physics*, 131:154903, aug 2009.

[14] Somendra M Bhattacharjee, Achille Giacometti, and Amos Maritan. Flory theory for Polymers. *Journal of Physics: Condensed Matter*, 25:503101, 2013.

S6. Implementation Details

`RateCatalog` stores all possible transitions in the Kinetic Monte-Carlo algorithm. This is done by using an *unordered map*, a data-structure with a key (that uniquely defines a transition) and a *value* (we use the result of the transition as the value).

```
class RateCatalog
        // variables
        unordered_map catalog
                // Unordered map variable that is the actual rate catalog.
                // Key to the map: (move_type, position, direction),
                // where "position" is an identifier that specifies on
                // which particle the move is to be performed
                // and "direction" specifies in which direction (up, down, etc.)
                // the move is to be performed.

        bool isStored(key)
                return true if key is in the rate catalog

        void remove(key)
                removes key from rate catalog
```

`Transformer` performs most of the work in our LKMC. It contains methods to add transitions for a given particle to the rate catalog, remove tansitions, and perform transitions (including the necessary updating of the microstate). A member variable of the class is `system`, which contains the details of the *microstate*. Another member variable is `rate_catalog`, which contains details about the possible *transitions*.

```
class Transformer
        // variables
        float total_rate = 0
        unordered_map rate_catalog
        float time = 0
        System system

        void initiate()
                // instantiates the rate catalog for the very first KMC step
                loop over all particles with particle ID = site
                        addMoves(site)

        void addMoves(site)
                // adds all transitions at particle ID = site
                loop over all move types
                        if move type has a rate > 0
                                loop over all possible moves of this type at this site
                                        X = compute result of move
                                        if move_result != null // checks whether move is allowed
                                                k = computeMoveRate(X)
                                                insert (X,k) in rate catalog
                                                total_rate += k

        void removeMoves(site)
                // remvoes all transitions at particle ID = site
                affected_moves = getMoves(site)
                for key in affected_moves
                        (X, k) = rate_catalog[key]
                        total_rate -= k
                        rate_catalog.remove(key)

        void removeMoves(coord)
                // removes all transitions at a 3D-coordinate on the lattice
                site = lattice[coord]
                removeMoves(site)

        void addMoves(coord)
                // adds all transitions at a 3D-coordinate on the lattice
                site = lattice[coord]
                addMoves(site)

        vector<key> getMoves(site)
                // returns all transitions at particle ID = site
                result = {} // empty container
                for key in rate_catalog // loop over all KMC moves
                        if key.position == site
```

```
                        result = {result, key}
        return result

    key selectMove()
        // selects move using tower sampling
        r = uniformly sampled number in [0, 1>
        R = total_rate * r
        cum = 0
        for key in rate_catalog// loop over all KMC moves
                rate = key.rate
                if (cum <= R) and (R < cum + rate)
                        return key
                else
                        cum += rate

    void doMove()
        // performs move and updates state of the system
        r = uniformly sampled number in [0, 1>
        delta_t = - log(r) / total_rate  // sample exponentially distributed random number
        time += delta_t
        key = selectMove()
        move_class = key.move_type // instance of an overload of BaseMoveClass
        move_result = rate_catalog[key].result // returns all information about transition
        aff_coords = move_class.computeAffectedCoordinates(move_result.coordinate)

        for coord in aff_coords
                removeMoves(coord)

        move_class.effectMove(result)

        for coord in aff_coords
                addMoves(coord)
```

`System` is a class with the details of the microstate and methods to manipulate the microstate, i.e. where the particles are, methods to displace particles, etc.

```
class System // contains all details of the microstate
    // variables
    parameters // example {{"n_monomers", 100}, ... }
    coordinates = {coord_1, coord_2, ...}
    lattice

    bool isOccupied(coord)
        return true if lattice[coord] is occupied

    bool areNeighboring(coord_1, coord_2)
        return (coord_2 - coord_1).euclideanNorm == 1

    bool areOverlapping(coord_1, coord_2)
        return coord_2 == coord_1

    void moveMonomer(site, new_coord)
        old_coord = lattice[
        lattice.move(site, old_coord, new_coord)
        coordinates[site] = new_coord

    vector<int> getAllNeighbors(site) // returns all occupied particles neighboring to particle ID = site
        X = coordinates[site]
        vector<int> result = {}

        for direction in {-1,+1}
                for j in 0...dimension
                        e_ij = direction * e_j // e_j[j]=1, e_j[k!=j]=0
                        if isOccupied(X+e_ij)
                                result = {result, lattice[X+e_ij]
        return result
```

We created a *virtual class* called `BaseMoveClass` that can be overloaded by a specific implementation of a transition. Such a transition will have the same methods as this virtual class, but with details that depend on the type of move. An example are the classes for the Verdier–Stockmayer move-set [11] that displace one or two monomers. We have additional overloads of this class for slip-link diffusion, binding and unbinding, etc.

Most of the *physics* is contained in the method `computeEnergyDifference`, which returns a floating point number that corresponds to $\Delta E = H(\omega') - H(\omega)$, where $H(\omega)$ is the Hamiltonian before the transition and $H(\omega')$ is the Hamiltonian after it. For a polymer with

self-overlap we considered the Hamiltonian $H = \frac{1}{2} J \sum_{i,j} \delta_{\mathbf{r}_i,\mathbf{r}_j}$, where the sum runs over all pairs of monomers and J is the monomer overlap energy. Another example would be the Hamiltonian for a semiflexible polymer: $H = 1/2K \sum_i (\mathbf{t}_{i+1} - \mathbf{t}_i)^2$, where $\mathbf{t}_i = \mathbf{r}_i - \mathbf{r}_{i+1}$ is a tangent vector and K is a bending rigidity.

```
class BaseMoveClass (virtual class, not strictly necessary, but useful for making sure overloaded classes have
                        all the necessary methods)

        // variables
        attempt_rate

        virtual vector<coord> computeAffectedCoordinates(move_result)
        // returns all coordinates possibly affected by the transition using the 'minimal local update'

        virtual vector<move_selection> computeMoveSelections(site)
        // returns all possible transitions that can be performed on particle with ID = site

        virtual move_result computeMoveResult(move_selection)
        // returns the result of the transition "move_selection"

        virtual void effectMove(move_result)
        // performs the transition, changing the lattice, updating coordinates, etc.

        double computeMoveRate(move_result)
                // returns the rate of the transiton based on Metropolis-Hastings kinetics
                delta_E = computeEnergyDifference(move_result)
                if delta_E < 0
                        return attempt_rate
                else
                        return attempt_rate * exp(- delta_E )

        virtual float computeEnergyDifference(move_result)
        // returns energy difference of the transition
```