Supplementary Information for:

Adhesive dynamics simulations of highly polyvalent DNA motors

Aaron T. Blanchard[1,*,†], *Selma Piranej*[2], *Victor Pan*[1,§], *& Khalid Salaita*[1,2*]

[1] Department of Biomedical Engineering, Georgia Institute of Technology and Emory University, Atlanta, Georgia 30322, USA.
[2]Department of Chemistry, Emory University, Atlanta, Georgia 30322, USA.
[*]Address correspondence to: ATBlanchard@ymail.com and k.salaita@emory.edu
[†] Current address: Department of Biomedical Engineering, Duke University, Durham, North Carolina, 27709, USA
[§]Current address: Intellia Therapeutics, Cambridge, Massachusetts 02139, USA

Contents

Supplemental User Guide for RoloSim v1.20

The following section proceeds in three parts. **Section I** contains a list of all input parameters, with descriptions. **Section II** describes the data structures used to track and store data in RoloSim. **Section III** contains a list of all function files that are included in the core RoloSim package, a diagram of dependencies (See **Figure S1**), and a brief description of each file. **Section IV** describes additional files used to prepare and interpret RoloSim simulations. Finally, **Section V** describes some of the key physical and mathematical assumptions made in order to make RoloSim computationally tractable, as well as limitations of RoloSim that arise from those approximations.

Note, there are two common acronyms used throughout this documentation: the potential pairs list (PPL) and the RNA indices cell array (RICA). These two data arrays are central to the computational efficiency of RoloSim, so they are described here:

> RICA – RNA indices cell array. In RoloSim, the RICA is called v.fIndCell. One of the most computationally expensive tasks in adhesive dynamics is determining which guide-fuel pairs are close enough to each other to associate. For example, if there are 10^6 fuel strands on the surface, then it would require 10^6 distance calculations to determine which strands are within binding range of a single guide strand. This expensive and wasteful calculation would then need to be repeated for each DNA guide strand in interaction range, which would then need to be repeated after every event. To circumvent this issue, the RICA, which is generated during initialization (**Fig. S2**) is used as a data array that is organized in a manner that mimics the physical geometry of the RNA fuel surface. To create the RICA, all RNA strands are divided into 10×10 nm^2 grid points, and the indices of all RNA strands within a grid point are stored within the corresponding cell in the RICA. For example, the cell in the 46th row and 38th column in the RICA contains RNA fuel strands with 455 nm $\leq$ y-coordinate < 465 nm and 375 nm $\leq$ x-coordinate < 385 nm.
>
> PPL – potential pairs list. In RoloSim, the PPL is called v.unbound_pairs (see below). This is another data structure that is utilized to decrease the number of computations of distances between pairs. Because individual guide strands do not move very much between successive timepoints, it is not necessary to determine which strands are within interaction range of each other at every timepoint. Instead, the PPL is constructed to contain a running list of potential pairs, and this list is referenced at each timepoint to determine whether strands stochastically interact. The PPL is updated periodically when the HPDM has moved sufficiently far or when DNA guide strands enter or leave interaction range.

The PPL and RICA are closely linked; when constructing the PPL, potential pairs can be determined by extracting RNA fuel indices from cells in the RICA that are close to the grid point of the DNA guide strand only, thus greatly reducing the number of computations.

I – Input parameters

Settings are initialized in the first several lines of MainRollsim.m with fields in the “Settings” struct (these are later transferred into “s”).

FuelDensity (Default: 0.05)

Denotes the number of RNA fuel strands (per nm^2) on the planar surface.

ParticleStrandDensity (Default: .091)

Denotes the number of DNA guide strands (per nm^2) on the surface of the HPDM.

BodyType (Default: 1)

This is an index used to specify the “type” of HPDM geometry. 1=sphere, 2=dimer, 3=rod with spherical caps, 4=polygon with rounded edges, 5=continuous track HPDM, 6=continuous track HPDM geometry without slip (4-6 will be described in future work).

Diameter (Default: 5,000)

HPDM diameter in nm. For types 1 and 2, denotes the spherical diameter of the particle(s). For type 3, denotes the diameter of the rod and spherical caps.

SepDist (Default: 1,000)

Denotes “separation distance” between sphere centroids for types 2 and 3. Denotes motor width for types 4, 5, and 6.

CTlength (Default: 100)

For type 5 only, this parameter denotes the length of a continuous-track HPDM along the translocation direction.

Npoly (Default: 3)

Order of the polygon prism (e.g. the number of edges around the prism cross-section), for type 4 only.

RadCurv (Default: 4)

Radius of curvature of the edges of the polygonal prisms, for type 4 only.

t_max (Default: 1,800)

Denotes the total duration, in simulation time, of the simulation.

dt (Default: .03)

Denotes the simulation time step duration in seconds.

kRH (Default: 2.1658)

Denotes the RNase H cleavage rate in 1/s.

DisplayFlag (Default: 1)

This parameter determines whether or not the DisplayStatus.m function is called to periodically show the simulation progress. This flag is set to zero when many simulations are run on a computing cluster.

DisplayInterval (Default: 1)

Denotes the duration, in simulation seconds, between successive calls to DisplayStatus.m.

RandStepSize (Default: 1)

Denotes the step size, in nm, of random displacements applied per timestep during Brownian motion when polyvalency drops to zero.

LockPercent (Default: 0)

Denotes the percent of RNA fuel strands that are non-cleavable strands (thus mimicking DNA analogues of the RNA fuel.

beta (Default: 1)
Scaling factor for transition state tether stiffness. This parameter should be obsolete.

BiasForce (Default: 0)
Force in pN, applied in positive x-direction. Non-zero force will bias motion (**Fig. 8**).

kb0 (Default: 0.0091333)
Guide-fuel hybridization rate at zero distance, in 1/s.

ktstar (Default: 0.8769)
The κ_t^* coefficient that defines the tensile stiffness of a tether's transition state for hybridization.

kt (Default: 3.663)
The κ_t coefficient that defines the tensile stiffness of a tether.

K (Default: 0.0045)
The K coefficient used to calculate the energy-extension relationship of a tether.

Kstar (Default: 0.00356)
The K^* coefficient used to calculate the energy-distance relationship of a tether's transition state for hybridization.

SaveFile
This is the name of the ".mat" file that the output is saved as at the end of the simulation.

II – Variables used to pass information between functions

All information in RoloSim is passed between functions using three "struct" arrays, denoted s, v, and o.

s – which stands for "Settings" – contains information related to the input parameters. Information stored within s does not change throughout the simulation. In addition to all of the setting parameters described above, s, which is initialized at the beginning of MainRollSim.m and in InitializeSettings.m, contains the following fields:

LowMidThresh
This is the cutoff height that separates the interaction range from the intermediate range (**Fig. S3**). This parameter is calculated automatically by determining the distance, in nm at which the guide-fuel association rate is equal to (0.00001)$k_{hyb,0}$.

MidHighThresh
This is the cutoff height that separates the intermediate range from the non-interacting range (**Fig. S3**). This parameter is calculated automatically as 30 nm + LowMidThresh.

FOrup

This is a “cfit” function handle that describes the relationship between a tether’s squared-extension (r^2, in nm^2) and its rupture rate (k_{rup}, in 1/s). By default, FOrup is set equal to the “FOrup15bp” cfit that is saved in the file RupRateVsDistSquaredv2.mat, which itself is generated by running the function “WLCtetherCalc.m”.

DisplayTime

If s.DisplayFlag is nonzero, this parameter updates during every call to s.DisplayStatus as the current time plus s.DisplayInterval. Subsequent calls to DisplayStatus will check if the simulation time is greater than DisplayTime to determine whether to output display items to the figure.

SurfaceWidth

This parameter, which is used for initialization calculations, is the width of the initial RNA surface square. It is automatically calculated as 1,000 nm + s.SepDist.

t_all

This is a vector, generated and used during initialization, of all timepoints that will be modeled during the simulation. It is automatically calculated from s.dt and s.t_max.

dTheta

This is the angle, in radians, that a spherical HPDM can roll before it is possible for strands in the non-interacting range to transition into interaction range. This parameter is automatically calculated from s.MidHighThresh, s.LowMidThresh, and HPDM diameter.

RollLimit

This is calculated simply as cos(s.dTheta), and represents the change in the z-height of a unit vector with (initial orientation [0,0,0]) rotated by dTheta away from the z-axis. In practice, s.RollLimit is directly compared with the bottom right position of a cumulative rotation matrix to determine whether or not to call UpdateHighMid.m in UpdateSystem.m.

f

This is a figure handle that is generated if s.DispFlag is nonzero. Display items will be displayed to this figure.

opt

This is a SimulannealbndOptions object that specifies the settings for simulated annealing during UpdateSystem.m. It is generated in InitializeSettings.m via a call to the “optimoptions” function.

cutoff

This parameter is referenced by CheckEdge.m so that the surface can be expanded if the HPDM is within s.cutoff of one of the surface’s edges. It is calculated in a BodyType-dependent manner. This parameter is generally hard-coded to be 500 nm + s.SepDist/2 in InitializeSettings.m.

TileSize

This parameter, which determines how far to expand a surface boundary in CheckEdge.m if the HPDM's x-y center-of-mass position comes within s.Cutoff of an edge, is hard coded as 500 nm in InitializeSettings.m.

BindTimeListLength

This parameter, which determines the number of timesteps represented in v.BindEventList (see description of ApplyStochasticChanges.m), is hard-coded as 2,000 steps in InitializeSettings.m.

BindTimeListTime

This parameter, which determines the duration of simulation time represented in v.BindEventList (see description of ApplyStochasticChanges.m), is calculated at s.BindTimeListLength multiplied by s.dt.

EulerInit

This parameter, which defines the initial orientation in Z-Y-Z Euler angles, consists of three randomly generated numbers between 0 and 2π. This vector is generated in InitializeSettings.m and subsequently used to calculate the initial rotation matrix using CreateR.m.

TransWithinLowThresh

This parameter, which is generated in GenerateAssociationIndices.m and used in BindingDistances.m, is a monotonically-increasing vector that denotes the z-heights at which the number of grid cells accessible by a guide strand changes. When generating entries in the PPL for a given guide strand, the PPL is populated with RNA fuel indices from grid cells in the RICA in a manner that depends on the z-height of the guide strand: If the guide strand's z-height is less than the first entry of s.TransWithinLowThresh, then the first entry of CellIndOffset is referenced; if the z-height is between the first and second entries of s.TransWithinLowThresh, then the second entry of CellIndOffset is referenced; and so on.

CellIndOffset

This cell array, which is generated in GenerateAssociationIndices.m and used in BindingDistances.m, contains information necessary to reference appropriate cells from the RICA when generating entries for the PPL. Each cell contains a matrix with two columns; for each row, the entry in the first column denotes the offset in x-grid points and the second entry denotes the offset in y-grid points, both with respect to the guide strand's grid cell, that must be referenced. When generating a list of PPL entries (in BindingDistances.m) for a given guide strand, the guide strand's height is calculated and checked against s.TransWithinLowThresh to determine which entry in s.CellIndOffset should be referenced. Then, all cells with offsets denoted in the rows of the selected cell of s.CellIndOffset are extracted.

TimeIntFun

This function handle denotes the smallest type of integer that can be used to index all timepoints uniquely, as calculated by MinIntType.m.

LockDivisor

This parameter, which is generated in InitializeSettings.m as 100/s.LockPercent (rounded to the nearest integer), is used in ApplyStochasticChanges.m to determine whether an RNA fuel strand should be treated as a non-cleavable DNA lock. Specifically, a fuel strand is treated as a DNA lock strand if the remainder of its index divided by s.LockDivisor is equal to zero.

v – which stands for "Variables" – contains information used to describe the current state of the HPDM, including the position and orientation of the HPDM, a list of guide molecules within interaction range of the surface, and the distances between guide-fuel pairs. Information stored within v is highly dynamic and is not saved during or after the simulation. Specifically, v contains the following fields.

Step

This integer indicates which timestep the simulation is currently on, and is incremented at the end of each call to UpdateSystem.m (or in the jump-ahead section of ApplyStochasticChanges.m) to move the simulation forward to the next timestep.

ClvCount

This integer indicates the total number of cleavages that have occurred during the simulation. It is incremented with each call to CleaveBond.m and output to the command prompt during each call to DisplayStatus.m.

RupCount

This integer indicates the total number of ruptures that have occurred during the simulation. It is incremented with each call to BreakBond.m and output to the command prompt during each call to DisplayStatus.m.

BindCount

This integer indicates the total number of tether association events that have occurred during the simulation. It is incremented with each call to CreateBond.m.

R

This 3×3 rotation matrix is used to determine whether or not UpdateHighMid.m needs to be called in UpdateSystem.m. It describes the cumulative HPDM rotation that has occurred since the last call to UpdateHighMid.m. It is reset to the 3×3 identity matrix following each call to UpdateHighMid.m and is updated in UpdateR.m, which is called in each call to UpdateSystem.m.

x_move

This scaler denotes the cumulative displacement in the x-direction that has occurred since the BindEventList was last updated in ApplyStochasticChanges.m.

y_move

This scaler denotes the cumulative displacement in the y-direction that has occurred since the BindEventList was last updated in ApplyStochasticChanges.m.

EdgePos

This is a 1×4 vector that describes the positions of the boundaries of the RNA fuel surface. The first and third entries of the vector should always be set to zero. The second and fourth entries denote the width of the surface in the x- and y-dimensions, respectively, and are updated whenever the surfaces is expanded in CheckEdge.m.

Euler

This is a 1×3 vector containing the Z-Y-Z Euler angles that describe the HPDM's current orientation. It is updated after each energy minimization step in UpdateSystem.m.

Px

This is a vector that contains the x-positions of the HPDM at all current and former timesteps. A new entry is added to it after energy minimization in UpdateSystem.m (or with each skipped timepoint in the jump-ahead section of ApplyStochasticChanges.m).

Py

This is a vector that contains the y-positions of the HPDM at all current and former timesteps. A new entry is added to it after energy minimization in UpdateSystem.m (or with each skipped timepoint in the jump-ahead section of ApplyStochasticChanges.m).

Pz

This is a vector that contains the z-positions of the HPDM at all current and former timesteps. A new entry is added to it after energy minimization in UpdateSystem.m (or with each skipped timepoint in the jump-ahead section of ApplyStochasticChanges.m).

fCoords

Which stands for "fuel strand coordinates", is a matrix with 3 columns corresponding to x-, y-, and z-coordinates. Each row in the matrix corresponds to a single RNA fuel strand. The matrix is created during initialization and expanded when necessary during calls to CheckEdge.m. These values are referenced frequently when calculating inter-strand distances in UBdistCalc.m. The x- and y- coordinates of RNA strands do not change during the simulation, but a strand's z-coordinate is switched from 0 to $-\infty$ when it forms a tether with a guide strand. If that tether is ruptured (but not cleaved), then the fuel strand's z-coordinate is set back to 0. The "fuel strand index" discussed elsewhere describes a fuel strand's row in this matrix. This matrix is analogous to o.fCoords, with the only difference being that it contains z-coordinates.

gCoords

Which stands for "guide strand coordinates", is a matrix with 3 columns corresponding to the x-, y-, and z-coordinates of all guide strands. Each row in the matrix corresponds to a single DNA guide strand's coordinates. Coordinates are updated at each timestep for strands in interaction or intermediate range, but coordinates for strands in non-interaction range are centered on the origin. The "guide strand index" discussed elsewhere describes a guide strand's row in this matrix. This matrix is analogous to o.gCoords, with the main difference being that coordinates are updated frequently throughout the simulation.

gIndHigh

This vector contains a list of indices for all guide strands that are in non-interacting range. It is updated during calls to UpdateHighMid.m. See **Fig. S3** for a schematic depiction.

gIndMid

This vector contains a list of indices for all guide strands that are in interaction range. It is updated during calls to UpdateHighMid.m and UpdateMidLow.m. See **Fig. S3** for a schematic depiction.

gIndLow

This matrix contains a dynamic list of all DNA guide strands in interaction range, including associated information necessary to speed up potential-pairs' distance calculations. Each row in this matrix corresponds to one DNA guide strand in interaction range. The first column contains the guide strand's index (it's row in the coordinate list v.gCoords). The second column contains the position of the first row in the PPL with a potential pair involving this guide strand (this is updated in calls to the RefreshUBpairs sub-function in UBdistCalc.m). The third column contains an integer denoting the number of potential pairs that are listed continuously for this guide strand in the PPL. The values in the second and third columns are used for inter-strand distance calculations in UBdistCalc.m. The fourth and fifth columns contain the x- and y-coordinates (respectively) of the guide strand rounded to the nearest 10 nm. These latter two values are used to determine which grid cell the strand is in when generating PPL entries in AddToUBpairs.m, and to determine whether the guide strand has shifted into a new grid cell in UpdateCoords.m. The fifth column contains the guide strand's z-coordinate, which is used to determine which grid cells are within binding range in BindingDistances.m. Rows are added to this list in calls to CreateBond.m (**Fig. S5**) and removed in calls to BreakBond.m and CleaveBond.m (**Fig. S4**). See **Fig. S3** for a schematic depiction of this matrix.

fIndCell

This cell array, which is referred to elsewhere in this document as the RNA indices cell array (RICA), speeds up PPL creation by storing RNA fuel indices in a manner that mimics the physical geometry of the surface; each cell contains a list of indices corresponding to RNA fuel strands that are located within a 10×10 nm^2 bin corresponding to the position of the cell in the array. For example, the cell in the 46th row and 38th column in the RICA contains RNA fuel strands with 455 nm $\leq$ y-coordinate < 465 nm and 375 nm $\leq$ x-coordinate < 385 nm. See a schematic depiction of the RICA in **Fig. S2**. This cell array is generated during initialization and extended when necessary upon calls to CheckEdge.m. When strands are irreversibly cleaved, their indices are removed from the RICA to prevent needless future computations. When a DNA guide strand enters interaction range, the grid cells that can contain RNA fuel strands within a cutoff distance of the guide strand are determined as a function of guide strand z-height in BindingDistances.m. The RNA indices in these cells are then extracted and paired with the guide strand index for addition to the PPL.

pairs

This matrix, which has two columns, contains a dynamic list of tethers bound to the HPDM. Each row corresponds to one tether. For each row, the first column contains the index of the guide strand in the tether, while the second column contains the index of the fuel strand. Tethers are added to and removed from this list during calls to CleaveBond.m, CreateBond.m, and BreakBond.m.

unbound_pairs

This variable is referred to elsewhere as the potential pairs list (PPL). It is a matrix that contains three columns, and each row either 1) corresponds to a potential guide-fuel pair or 2) contains dummy / padding values that occupy pre-allocated space. For rows that correspond to potential pairs, the first and second columns contain the indices of the DNA guide and RNA fuel strands, respectively, involved in the potential pairs. The third column contains the guide-fuel squared-distance values that were calculated during the last call to UBdistCalc.m. This matrix contains many rows of un-used pre-allocated space, which new potential pairs are inserted into whenever new guide strands are added to interaction range (**Fig. S4**). The first row of pre-allocated space is pointed to by ubp_idx. When new tethers are added to v.pairs, the potential pairs corresponding to guide strand in the PPL are over-written with dummy values (see **Fig. S5**). When the PPL runs out of padding, RefreshUBpairs (a sub-function of AddToUBpairs.m) is called to remove all dummy rows and add new padding.

ubp_idx

This integer is an index that points to the first padded position in the PPL (see **Fig. S4**), thus denoting the position at which entries should be added to the PPL. It is updated in AddToUBpairs.m.

BindEventList

This matrix contains a list of potential pairs that are expected to form within a short number of timesteps, specified by s.BindTimeListLength. This list is constructed from the PPL in ApplyStochasticChange.m following calculations of expected association times by UBdistCalc.m. Because the PPL is very large, it is computationally inefficient to calculate new stochastic association times and determine which events occur within the current timestep for every single timestep. Instead, association times are calculated and events within the next s.BindTimeListLength timesteps are extracted, sorted, and stored in v.BindEventList. In subsequent timesteps, association events are drawn from the top of this list. This matrix contains three columns: the first column contains an integer with the number of timesteps until association occurs, and is decremented at the end of each timestep; the second and third columns contains the indices of the guide and fuel strands involved in the association event. v.BindEventList is refreshed whenever the HPDM moves more than 1 nm relative to the x-y plane.

LockInd

This vector contains a list of indices to entries in o.LockData.PairList that still represent active pairs. It is used to determine which entries in o.LockData.TimeData must be updated in each timepoint.

t

This scalar denotes the current time (in simulation time) of the simulation.

o – which stands for “Output” – contains information that is saved at the end of the simulation, including a list of the HPDM’s position and orientation at each timestep and a full list of tether formation, cleavage, and rupture events. The information contained within o is sufficient to re-create the HPDM’s state at any timepoint during the simulation, but requires much less memory than v. Specifically, o contains the following fields:

fCoords

Which stands for “fuel strand coordinates”, is a matrix with 2 columns corresponding to x- and y-coordinates (all z-coordinates are zero, so they are not saved). This output is generated during initialization and is updated when the surface is expanded during CheckEdge.m. The “fuel strand index” discussed elsewhere describes a fuel strand’s row in this matrix. This matrix is analogous to v.fCoords, but only for the initial timepoint.

gCoords

Which stands for “guide strand coordinates”, is a matrix with 3 columns corresponding to the initial x-, y-, and z-coordinates of all guide strands. This output is generated during initialization and does not change during the simulation. The “guide strand index” discussed elsewhere describes a guide strand’s row in this matrix. This matrix is analogous to v.gCoords, but only for the initial timepoint.

TimeData

This is a matrix with a number of rows equal to the number of timesteps and six columns. The first three columns of the i^{th} row correspond to the x-, y-, and z-position of the HPDM at the i^{th} timestep, while the fourth, fifth, and sixth columns correspond to the Z-Y-Z Euler angles that describe the HPDM’s orientation at the i^{th} timestep. For continuous track HPDMs (s.BodyType=5), a seventh column denotes the roll angle. This matrix is pre-allocated during initialization, and its rows are filled in throughout the simulation.

EventData

This is a table with four columns. Each row in the table describes an event (tether formation, cleavage, or rupture). The first column, called “EventType”, describes what type of event occurred (1=association, 2=cleavage, 3=rupture). The second column, called “TimeIndex”, is an integer that describes what timestep the event occurred in. The third column, gInd, describes the index of the guide strand involved in the event, while the fourth column, fInd, describes the index of the fuel strand involved in the event. This table is generated during initialization and is extended throughout the simulation as events occur.

pairs

This is a two-colum matrix listing the set of tethers at the initial timepoint. It is analogous to v.pairs, but reflects the state of the tether set at the initial timepoint only. For each row, the first column denotes the index of the DNA guide strand in the tether, while the second column denotes the index of the RNA fuel strand in the tether,

r2

This vector contains a growing list of the distance-squared between strands immediately before association for every association event that occurs.

dz

This vector contains a growing list of the z-heights of the guide strands involved in every association event that occurs.

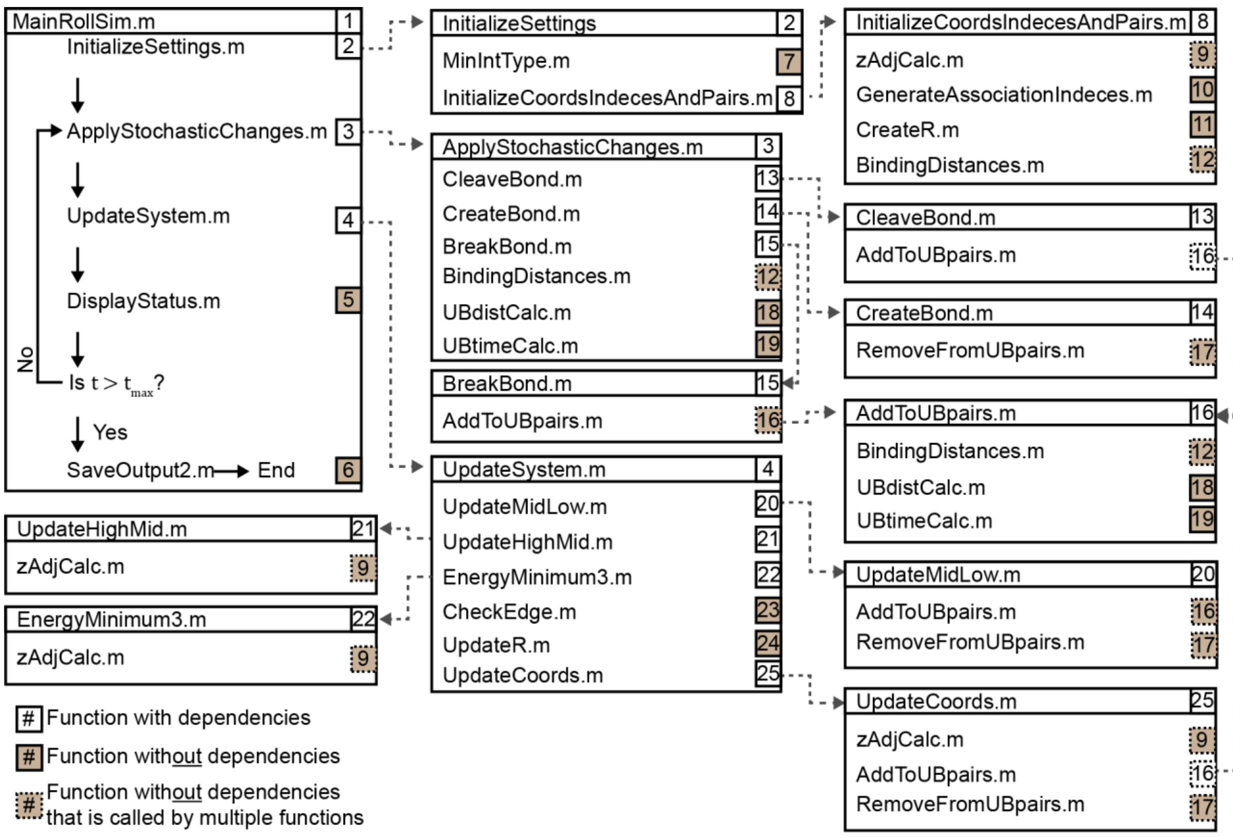


Figure S1 RoloSim V1.20 flowchart and depiction of dependencies
The core of RoloSim V1.20 is a package containing 25 MATLAB files (extension: .m). The manner in which the files relate to each other is shown in the above figure. Each file is listed next to a number label in a white or brown box. Files are described in the text below in the order denoted by their number labels. White label boxes indicate files that call additional files (i.e. dependencies). Brown label boxes indicate files with no dependencies, and brown boxes with dashed borders are files that have no dependencies that are called by multiple files. All files with dependencies are depicted in the figure in a block with a list of dependencies. The main file, MainRollSim.m, contains a flowchart that shows the iterative process used to simulate HPDM motion.

III – Descriptions of core RoloSim files

1) MainRollSim.m

This function is RoloSim's main script. The first large block of code contains definitions of input parameters. The simulaiton is initialized with a call to "InitializeSettings.m", and then the functions "ApplyStochasticChanges.m", "UpdateSystem.m", and (periodically) "DisplayStatus.m" are called iteratively until the simulation is complete. The output of the simulation is then saved with a call to "SaveOutput2.m", and, following a 1-minute pause, the function "CallMRSmeasureForce" is called to begin the process of estimating the force generated by the simulated HPDM (see next section – note this line is commented out). MainRollSim.m has two inputs, both integers: i, which is used to determine a parameter of interest in parameter sweep studies (e.g. DiameterAll=500:500:6000;Settings.Diameter=DiameterAll(i);); and j, which is used to initialize the random number generator (e.g. rng(j)) and defines the iteration number when multiple iterations at a single condition are performed. Note that j must be changed between iterations to produce different stochastic outputs, if all setting are the same.

2) InitializeSettings.m

This function, which is called at the beginning of the simulation, performs several operations necessary to initialize the simulation including the calculation of:

- The width of the RNA fuel surface (s.SurfaceWidth).
- The cutoff distance (s.cutoff), which is used by the function “CheckEdge.m”.
- The angle, in radians, that an HPDM can roll (s.dTheta) before it is possible for strands in the non-interacting range to transition into interaction range (**Fig. S3**). This parameter is used to determine whether “UpdateHighMid.m” must be called.

In addition, the function performs the following tasks:

- The parameter s.TileSize (which is the distance by which the RNA fuel surface boundary is shifted within CheckEdge.m) is set to 500 nm.
- The bind time list is initialized.
- A figure (s.figure) is opened for display if s.DisplayFlag is nonzero.
- Settings for simulated annealing are initialized (s.opt).
- The HPDM rotation, saved in Z-Y-Z Euler angles, is initialized as [0,0,0].
- Additional data initialization, including initialization of most of the field in the “v” struct is performed by calling “InitializeCoordsIndicesAndPairs.m”
- The x- and y-coordinates of the first DNA guide strand and the first RNA fuel strand are set to -∞ and ∞, respectively. (At the time of this writing, I don’t remember why this is done, but I remember it was important at one point in the development so I’ve kept it this way).
- The smallest type of unsigned integer that can be used to save time index information in o.EventData is calculated using MinIntType.m.
- The “o” struct is initialized by creating the table o.EventData, initializing o.TimeData, and duplicating the pair list, RNA fuel coordinates, and DNA guide coordinates.

3) ApplyStochasticChanges.m

This function, which is called in every timestep by MainRollSim.m, performs the following steps:

- Calculates the time until each tether is cleaved ($ClvgTime = -\log(rand)/k_{clvg}$) or ruptured ($RupTime = -\log(rand)/k_{rup}$) where $rand$ is a random number between zero and one. A list of events that occur in the current timesteps is then assembled, with rupture events taking precedence over cleavage if both occur for the same tether.
- Calculates the time until binding events happen in a fashion similar to the above step. However, because there are a large number of potential binding interactions, it would be computationally costly and inefficient to perform binding time calculations during every timestep. To overcome this limitation, association times are calculated via calls to UBdistCalc.m and UBtimeCalc.m and all binding events that occur within a time specified by s.BindTimeListTime are added to v.BindEventList. After creation of v.BindEventList, events are cross checked to eliminate events that contain fuel or guide indices found in earlier-occuring events in the list. In subsequent timesteps, association

times are only re-calculated if 1) v.BindEventList is empty, and/or 2) the HPDM has translocated by more than 1 nm (in the x-y plane) since v.BindEventList was last updated. Association events for the current timestep are then calculated by referencing v.BindEventList to find all binding events that occur in the current timestep.

- If no events occur during the current timestep, and if pairs exist, then there is no point in calling UpdateSystem.m because the HPDMs position and orientation will remain the same. Therefore, ApplyStochasticChanges.m has a code to jump ahead to the next timestep at which events do occur. This code updates the simulation time and places repeat values in v.Px, v.Py, v.Pz, and o.TimeData, to denote that no movement occurs during the timesteps in which no events occur.
- For all binding events that occur in the current timestep, tethers are created by calling CreateBond.m. For any new DNA-DNA lock tethers, entries in o.LockData are created.
- For all cleavage events and rupture events that occur, tethers are removed with calls to CleaveBond.m and BreakBond.m, respectively. For any DNA-DNA lock tether ruptures, the time of rupture is recorded in o.LockData.
- All cleavage, rupture, and association events are added to o.EventData.

4) UpdateSystem.m

This function, which is called in every timestep by MainRollSim.m, updates the HPDM's position and orientation and performs other data management following alterations to the tether set performed in ApplyStochasticChanges.m. The bulk of the code is skipped if no events (e.g. tether formation, cleavage, or rupture) occurred during the last call to ApplyStochasticChange.m (in which case the HPDM's position and orientation do not change). Otherwise, a series of steps are performed:

- The HPDM's position and orientation is updated. If there are one or more tethers, this is accomplished by using the function "simulannealbnd" to perform energy minimization, with the function EnergyMinimum3.m used as the objective function and an initial guess of [0, 0, 0, 0, 0, 0]. If there are no tethers, the HPDM's position is updated by displacing the HPDM in a random direction parallel to the x-y plane by an amount specified by s.RandStepSize.
- The positions of guide strands are updated by calling UpdateCoords.m.
- The position and orientation of the HPDM are updated in the "v" struct and stored in the "o" struct.
- Unpaired DNA guide strands are swapped between interaction and intermediate ranges by calling UpdateMidLow.m.
- The HPDM's rotation since the last call to UpdateHighMid.m is checked against s.RollLimit and, if sufficient rotation has occurred, UpdateHighMid.m is called.
- The function CheckEdge.m is called to determine whether the HPDM is approaching the edge of the RNA fuel surface and expand the surface if necessary.
- If locks (non-cleavable DNA analogues of RNA fuel) exist, information about their extensions are stored in o.LockData.

5) DisplayStatus.m

This function is only called if, 1) the setting "s.DisplayFlag" is not set to "0", and 2) a duration, in simulation time, longer than the setting "s.DisplayInterval" has passed since the last call to

DisplayStatus.m. The function displays a 2D histogram of un-consumed RNA fuel (thus simulating a typical fluorescence microscopy image of Cy3-labeled RNA during HPDM translocation) with a 30 nm bin size, as well as a circular outline of the HPDM to show its current position. The function also outputs some system parameters to the MATLAB command prompt; specifically, the polyvalency, the duration in the simulation and reality since the initiation of the simulation, and the total number of cleavage and rupture events that have occurred in the simulation.

6) SaveOutput2.m

This function, which is called at the end of the simulation, extracts all fields within the "o" and "s" structs and saves them as variables within a .mat file (this approach was selected to improve the robustness of file storage, as structs can sometimes become corrupted while or after being saved to .mat files). This function then pauses the simulation for 5 minutes to ensure that the file has time to finish saving before the code terminates and the compute cluster core is switched off.

7) MinIntType.m

This is a simple function that counts how many guide strands are generated and determines the smallest type of unsigned integer (i.e., uint8, uint16, uint32) that can be used to give a unique index to each one.

8) InitializeCoordsIndicesAndPairs.m

This function, which is called by "InitializeSettings.m", initialized data in the "v" struct via several steps:

- The HPDM coordinates are initialized with v.Px and v.Py set at the center of the RNA fuel surface and with v.Pz=5 nm.
- The sub-function "GenerateStrandCoordinates" is then called. This sub-function generates v.fCoords by randomly patterning fuel strands across a square with a width defined by s.SurfaceWidth at a surface density defined by s.FuelDensity. The function then generates v.gCoords by randomly patterning guide strands on a simulated HPDM at a surface density defined by s.ParticleStrandDensity (this step depends on s.BodyType).
- The sub-function "DivideGuideStrands" is then called. This sub-function separates strands into interaction range, intermediate range, and non-interacting range (referred to as low, mid, and high, respectively, see **Fig. S3**) according to height cutoffs defined by s.LowMidThresh and s.MidHighThresh. The sub function then generates lists for the three range (v.gIndHigh, v.gIndMid, and v.gIndLow, see **Fig. S3**).
- The sub-function "CellularizeCoords" is then called. This function generates the RICA, wherein each cell contains a list of fuel indices within a 10×10 nm^2 bin.
- An initial list of fuel-guide pairs (v.pairs) is then generated by creating a list of all pairs within 5 nm of each other and removing duplicates.
- The parameters s.TransWithinLowThresh and s.CellIndOffset, which are used to determine which cells in the RICA to extract fuel indices from as a function of guide strand height, are then calculated by calling "GenerateAssociationIndices.m".
- A rotation matrix is then created from a random triplet of Z-Y-Z Euler angles by calling "CreateR.m", and the HPDM's height is calculated by zAdjCalc.

- Finally, the PPL (v.unbound_pairs) is created by calling BindingDistances.m and the PPL is padded with 100,000 empty rows. The PPL pointer (v.ubp_idx) is then initialized as the index of the first padded row.

9) zAdjCalc.m

This function, which is called by multiple other functions, calculates the relationship between HPDM rotation and z-displacement of the HPDM's center-of-mass. For spherical HPDMs, rotation does not influence z-displacement. However, the center-of-mass of rod-shaped HPDMs, for example, must be displaced upwards with off-axis rotations to prevent collision with the surface.

10) GenerateAssociationIndices.m

This function generates two settings variables – s.TransWithinLowThresh and s.CellIndOffset – to speed up calls to the RICA. This code determines which 10×10 nm^2 grid-cells around a guide strand's grid cell can contain strands within a binding cutoff distance. (The binding distance cutoff is set equal to the parameter s.LowMidThresh). The two variables s.TransWithinLowThresh and s.CellIndOffset are then used subsequently during additions to the PPL to determine which fuel strands can serve as potential pairs.

11) CreateR.m

This is a simple function that creates a rotation matrix from a set of 3 Z-Y-Z Euler angles. It's much quicker than built-in functions that do the same thing.

12) BindingDistances.m

This function, which is called by multiple other functions, determines which RNA fuel strands are within binding range of a DNA guide strand (see description of "GenerateAssociationIndices.m") and returns a list of all potential interactions to be added to the PPL. This calculation is sped up by the use of the RICA; the guide strand's "bin" is calculated by rounding its x- and y-coordinates to the nearest 10 nm. The guide strand's distance from the surface is then used to determine which bins in the RICA could contain RNA fuel strands within binding range. A list of potential pairs is then created by pairing the guide strand's index with each of the indices of RNA strands within the determined grid cells. This function can accept a list of multiple DNA guide strands as an input, and will perform this process for each guide strand and produce a single list of potential pairs as an output. For each guide strand input, this function also formats and outputs a row in the list of DNA strands in interaction range, which can then be appended to the full list.

13) CleaveBond.m

This function is called by "ApplyStochasticChanges.m" whenever a tether is cleaved due to irreversible degradation of the RNA by RNase H. The function 1) removes the tether from the "pairs" list, 2) permanently removes the fuel molecule from the surface by removing it from the RNA indices cell array (RICA – defined in the description of "InitializeCoordsIndicesAndPairs.m" above), and 3) calls "AddToUBpairs.m" to perform all operations necessary to properly returns the DNA strand to the list of DNA strands in interaction range.

14) CreateBond.m

This function is called by “ApplyStochasticChanges.m” whenever one or more tethers is formed. For each tether, the function 1) adds the tether to the “pairs” list, 2) removes the fuel molecule from interaction range by setting its position to -∞, and 3) calls “RemoveFromUBpairs.m” to perform all operations necessary to properly remove the DNA strand to the list of DNA strands in interaction range.

15) BreakBond.m

This function is called by “ApplyStochasticChanges.m” whenever a tether is ruptured. The function 1) removes the tether from the “pairs” list, 2) returns the fuel molecule to the surface (from its temporary z-height of -∞), and 3) calls “AddToUBpairs.m” to perform all operations necessary to properly returns the DNA strand to the list of DNA strands in interaction range.

16) AddToUBpairs.m

This function, which is called by multiple other functions, performs all operations necessary to add one or more guide strands (and all of their potential RNA fuel binding partners) to the PPL. This function calls BindingDistances.m, UBdistCalc.m, and UBtimeCalc.m to perform this data management, and also updates the PPL index. In addition, if the PPL has run out of padding, the sub-function “RefreshUBpairs” is called to remove all dummy-rows from and re-pad the PPL. For a schematic depiction of data manipulation, see **Fig. S4**, which shows the operations performed to add a guide strand to the PPL following the cleavage of its RNA binding partner.

17) RemoveFromUBpairs.m

This function is called to remove a guide strand the list of guide strands in interaction range and overwrite the strand’s rows in the PPL with dummy values. For a schematic depiction of data manipulation, see **Fig. S5**, which shows the operations performed after a DNA strand pairs with an RNA strand.

18) UBdistCalc.m

This is a simple function that calculates the squared-distances between pairs of DNA guide and RNA fuel strands in the PPL.

19) UBtimeCalc.m

This is a simple function that calculates the time until a guide-fuel pair (or multiple pairs) hybridize. This calculation is performed by generating random number(s): $BindTime = -\log{(rand)}/k_{on}$, where $rand$ is a randomly generated number between zero and one. k_{on} is calculated from the squared-distance between the two strands using eqns. 6-7 from the main text.

20) UpdateMidLow.m

This function, which is called in every timestep by “UpdateSystem.m” determines which DNA guide strands have transitioned from “interaction range” to “intermediate range” (see **Fig. S3**) and calls “RemoveFromUBpairs.m” to remove them from the list of DNA strands in interaction range. Similarly, the function also determines which DNA guide strands have moved from “intermediate range” to “interaction range” and calls “AddToUBpairs.m” to add them to the list of DNA strands in interaction range.

21) UpdateHighMid.m

This function, which is called by "UpdateSystem.m" periodically (specifically, it is called whenever the HPDM has rotated enough for any point in non-interacting range to transition into interaction range since the last call of UpdateHighMid.m) determines which DNA strands in non-interacting range have transitioned into intermediate range and vice-versa (see **Fig. S3**). The function applies all necessary data management steps to transition these strands.

22) EnergyMinimum3.m

This function, which is called by "UpdateSystem.m", calculates the energy of the system (see the main text for details on energy calculations). This function is called within the "simmulannealbnd" simulated annealing step for the purpose of calculating the energetic minimum HPDM position and orientation. This function calls "zAdjCalc.m" to determine how an HPDM's rotational state influences its z-height. Note that the number "3" in the file's title is an arbitrary notation of this file's version number prior to release.

23) CheckEdge.m

This function, which is called by "UpdateSystem.m", checks to see if the HPDM is within a cutoff distance (s.cutoff) of a boundary (or two boundaries) of the existing RNA surface. If so, the function also extends the boundary (or boundaries) by a distance specified by s.TileSize (set to 500 nm by default) by adding additional fuel strands to the surface between the existing boundary and the new boundary. The addition of new fuel strands also requires the expansion of the RNA indices cell array (RICA – defined in the description of "InitializeCoordsIndicesAndPairs.m" above). If expansion occurs at the lower x-axis and/or the lower y-axis (e.g. left or bottom) boundary, then existing coordinates of all RNA and DNA strands, as well as the HPDM coordinates, are adjusted (in both the o and v structs) to ensure that the full surface is situated in the x>0, y>0 quadrant of the Cartesian plane.

24) UpdateR.m

This is a simple function that accepts as its inputs 1) a rotation vector, which it then converts to a rotation matrix, and 2) an initial rotation matrix. The two rotation matrices are then multiplied together. In other words, this function updates the orientation defined by the initial rotation matrix to include the additional rotation defined by the rotation vector.

25) UpdateCoords.m

This function updates the coordinates of paired guide strands and unpaired guide strands in interacting or intermediate range. The function then determines which unpaired guide strands have changed grid cells since the last timestep and, for each, refreshes their entries in the PPL by calling RemoveFromUBpairs.m and AddToUBpairs.m.

IV – Descriptions of additional files

26) CallMRSmeasureForce.m

This function has two inputs, the first of which is the name of a RoloSim output file and the second of which is "N". This function performs N iterations of stall force measurement by calling MRSmeasureForce N times, each time saving the save output (oLock) in a new file. The function also has code that outputs details of any error that may cause an iteration of MRSmeasureForce to terminate prematurely.

27) MRSmeasureForce.m

This function performs a single iteration of HPDM stall force measurement from an output file specified by the input “OutputFileName”. The function first calls loadRestruct.m to load the Output and Setting structs and re-format them in the manner that is used by RoloSim. The function then selects a random timestep between the 1,000th timestep and the last timestep at which an event occurred, and then calls ReInitialize.m to re-create the “v” struct and finalize re-creation of the “o” and “s” structs. A random tether is then selected and designated as an indestructible lock tether that will be used to stall the HPDM. A RoloSim simulation is then allowed to run in a manner similar to MainRollSim.m, with the following key differences:

- ApplyStochasticChangesPin.m, a slightly-modified version of ApplyStochasticChanges.m, is called rather than ApplyStochasticChanges.m.
- Rather than running until s.t_max is reached, the simulation runs until the number of tether drops below a cutoff polyvalency, which is currently hard-coded as PolyvalencyCutoff=5.
- Rather than saving the final output, this function outputs “oLock”, which contains fields “Extension” (the extension, in nm, of the indestructible lock tether at all timepoints), “dt” (from s.dt), “LockPair” (a two-component vector with the indices of the guide and fuel strands that composed the indestructible tether), and “Tpoint” (The timepoint at which this iteration of the simulation began).

The second input, “j”, is an index used to initialize the random number generator (e.g. rng(j)). Note that different results will only be obtained from iterations that start with different j input values.

28) loadRestruct.m

This function loads all variables from the output file and re-structures them within the “Output” and “Settings” (which are similar to “o” and “s”) structs.

29) ReInitialize.m

This function performs all tasks necessary to re-create the “v” struct based on “s” and “o” inputs at a timepoint given by the third input “TimePoint”. The code contains a complicated block of text that fixes an issue with orientation storage, which should be addressed in a future version of RoloSim. The function also calls upon sub-functions “CellularizeCoords” and “DivideGuideStrands”, as well as the function BindingDistances.m, as part of the v re-creation (see **Figs. S1-2**).

30) ApplyStochasticChangePin.m

This function, which is called in each timestep by MRSmeasureForce.m, is nearly identical to ApplyStochasticChanges.m, with the key difference being that it includes code that prevents an indestructible tether from being cleaved or ruptured.

31) EnergyMinimumCT.m

This is a variant of EnergyMinimum3.m that is specific to continuous track (s.BodyType=5) HPDMs. It was not used in the present study but will be used in future work.

32) CreatePBSfiles.m

This script, which is located in the “CreatePBS” sub-directory (to prevent the main directory from getting cluttered), was used to create batches of PBS files necessary to run many simulations in parallel on a computing cluster. PBS files contain commands that are specifically formatted for the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology. Running this script currently generates several files for performing a parameter sweep of calculations with various HPDM diameters. In the current example, there are 10 different diameters being tested. The “i" and “j” parameters in the for loops are the same “i" and “j” indices used in MainRollSim.m. Therefore, the current code creates 5 PBS files (j ranging from 1 to 5) each for 10 different HPDM diameters, each denoted by the index i. The script also creates a batch submission file “BatchSubFile.txt” at the end, which can be used to initiate all simulations at once. Note that, in this example, RoloSim would need to be run on the server with the lines that start with “SaveFile=['~/scratch/…” and “CallMRSmeasureForce(…)” in MainRollSim.m un-commented.

33) WLCtetherCalc.m

This code can be used to re-create the function handle “FOrup15bp” and save it as “RupRateVsDistSquaredv2.mat”. It calls the function WLCapprox.m to perform worm-like-chain force-extension calculations.

34) WLCapprox.m

This code applies the approximation developed by Petrosyan (Rheologica Acta volume 56, pages 21–26, 2017) to calculate extension as a function of force, the number of residues in a worm-like-chain, the chain’s contour length per residue, the polymer’s persistence length, and thermal energy at room temperature.

V – RoloSim Assumptions and Limitations

RoloSim is not suitable for the simulation of very small HPDMs (e.g. nanomotors 25 nm diameter). RoloSim is not able to calculate association distances that include wrapping of the transition state tether around the body of an HPDM; all distances and extension calculations are simple end-to-end calculations. This is not an issue for HPDMs with dimensions much larger than the typical tether length. However, if the boundary between the interaction range and the intermediate range is higher than the HPDM diameter, the interaction range is automatically adjusted to be set at the HPDM radius during “InitializeSettings.m”. This setting is implemented to prevent associations of tethers through the body of a small (e.g. 25 nm diameter) HPDM, as only strands within interaction range can form tethers.

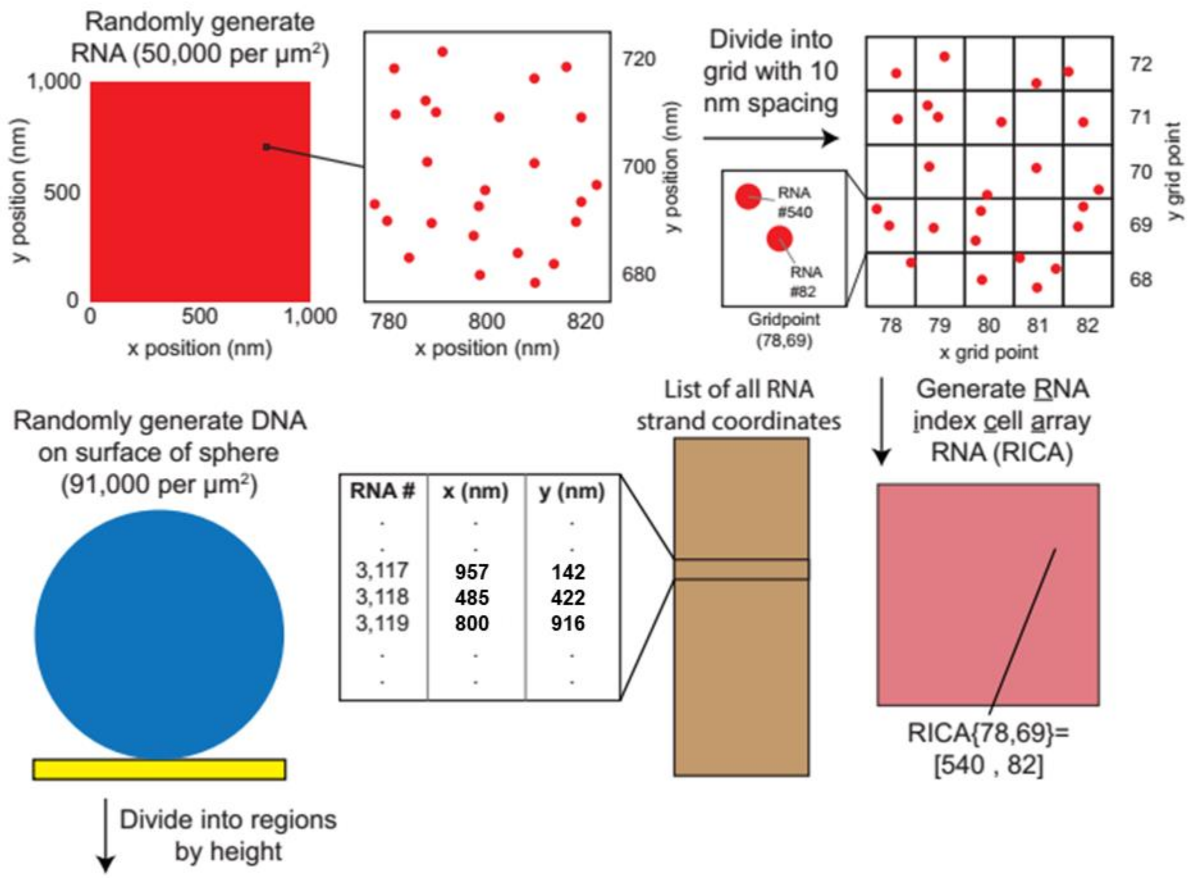
RoloSim outputs are not saved periodically during the simulation, and are instead saved only once at the very end. An earlier version of RoloSim implemented periodic saving (e.g. every 30 minutes in real-time), but this resulted in output files becoming corrupted on computing clusters. We were unable to determine why file corruption occurred, so we removed periodic saving and ensured that the wall-time used for cluster computing was much longer than necessary (generally, wall-time was set to 1-week).

As discussed in the description of “ApplyStochasticChanges.m“, stochastic association times are not calculated in every single timestep. Instead, association times for specific guide-fuel pairs are calculated only after the HPDM has translocated by >1 nm in the x-y plane since the previous association time calculation. Displacements below 1 nm were assumed to be small enough that they would not meaningfully affect k_{on}. However, technically it would be possible for large z-displacements to occur (e.g. moving closer to the substrate by 5 nm) without surpassing the 1 nm x-y displacement threshold. It is also possible for the HPDM to rotate substantially, thus causing inter-strand distance changes for some pairs to change by much more than 1 nm, without shifting its center of mass. These specific scenarios could potentially cause inaccurate binding time calculations, but were assumed to not occur with meaningful frequency because 1) z-height fluctuations are generally very small (**Fig. S6a-b**), and 2) the tight coupling between translation and rotation (**Fig. S6c-e**) make large rotations with very small associated translations very unlikely. However, future versions of RoloSim could close this gap by including calculations of 1) z-displacements and 2) maximum possible rotational displacements for strands within interaction range. In addition, in future work the cutoff displacement (currently 1 nm) and s.BindTimeListTime could potentially be optimized to balance tradeoffs between computational efficiency and accuracy. Finally, when DNA guide or RNA fuel strands go from bound to unbound through cleavage or rupture, their pairing events are not added immediately to v.BindEventList. Accordingly, there is a lag between when a guide strand becomes available for pairing and when the simulation allows it to form new tethers via v.BindEventList (specifically, this lag is the duration between the strand becoming available and the next trigger of association times calculations). Future versions of RoloSim could remedy this limitation.

Many initialization steps and checks were optimized for spherical HPDMs. When additional geometries were added, they were generally run and evaluated for proper function. However, the entire parameter space with non-spherical geometries was not explored, so users may encounter unforeseen issues when simulating non-spherical geometries. Here are some potential sources of errors that may arise when tinkering with non-sphreical geometries:

- The calculations of RollLimit and dTheta, as well as checks to determine whether to call UpdateHighMid.m, have not been tailored to non-spherical body types, which may perturb some functions. However, because the parameter s.Diameter is also used to describe type 2 and 3 HPDMs (dimers and rods with spherical caps), these calculations should coincidentally by okay. However, type 4 HPDMs (polygonal prisms) do not use the parameter s.Diameter, so if s.RadCurve is larger than the (otherwise arbitrary) s.Diameter parameter, then there may be issues wherein strands aren't properly exchanged between ranges when they need to be.
- The time-jump function in ApplyStochasticChanges could potentially cause the simulation to jump out beyond the prescribed end time, which could create additional problems elsewhere.

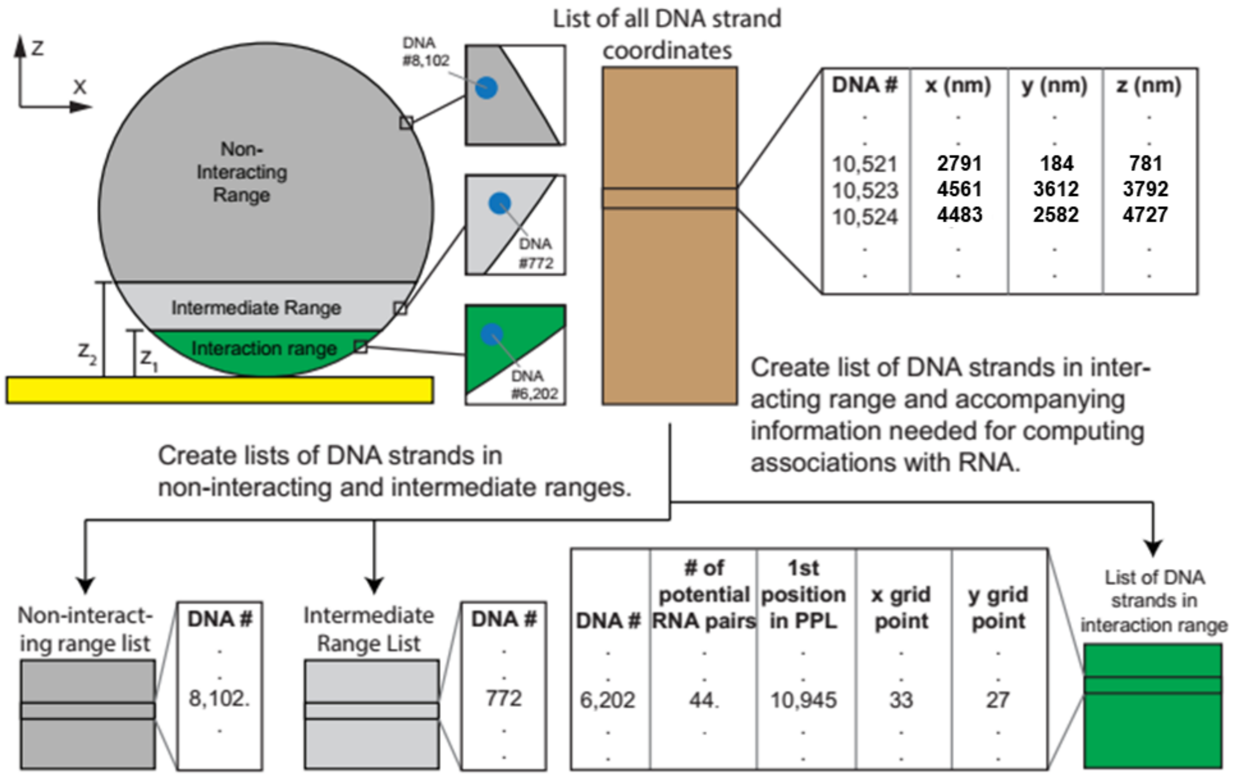
Supplemental Figures to Accompany User Guide



Randomly generate RNA (50,000 per μm²)
1,000
500
0
y position (nm)
0
500
1,000
x position (nm)
720
700
680
y position (nm)
780
800
820
x position (nm)
Divide into grid with 10 nm spacing
RNA #540
RNA #82
Gridpoint (78,69)
72
71
70
69
68
y grid point
78
79
80
81
82
x grid point
Randomly generate DNA on surface of sphere (91,000 per μm²)
Divide into regions by height
List of all RNA strand coordinates
Generate RNA index cell array RNA (RICA)
RICA{78,69}= [540 , 82]

RNA #	x (nm)	y (nm)
.	.	.
.	.	.
3,117	957	142
3,118	485	422
3,119	800	916
.	.	.
.	.	.

Figure S2: Initialization of RNA and DNA coordinates (part 1).
First, RNA fuel coordinates are randomly generated at a density of 50,000 molecules/μm^2. Each RNA strand is assigned a unique integer index. Two separate arrays are used to track RNA molecules. The first is the RNA strand coordinate list (denoted by the brown rectangle), which has the x-, y-, and z-coordinates of each RNA molecule. A fuel strand's position in this list corresponds to its index. The second is the RNA index cell array (RICA, denoted by the pink square). The RICA, which is used to determine which RNA molecules are within a certain area, is constructed by dividing all fuel molecules into 10 nm bins and storing their indices in corresponding cells. For a 1000 nm × 1000 nm surface, the RICA is a 100 × 100 bin cell array and each cell in the array contains zero or a few indices corresponding to the RNA strands that are within the corresponding 10 nm × 10 nm gridpoint. When the simulated HPDM comes within a set distance of a boundary, the surface is expanded by randomly generating new coordinates in the neighboring region, appending those coordinates to the to the strand coordinate list, and expanding the RICA accordingly. Next, DNA strands are randomly generated on a spherical surface – see **Fig. S3** below for initialization of DNA strand-related arrays.


z
x
Non-Interacting Range
Intermediate Range
Interaction range
z2
z1
DNA #8,102
DNA #772
DNA #6,202
List of all DNA strand coordinates
Create list of DNA strands in interacting range and accompanying information needed for computing associations with RNA.
Create lists of DNA strands in non-interacting and intermediate ranges.
Non-interacting range list
Intermediate Range List
List of DNA strands in interaction range

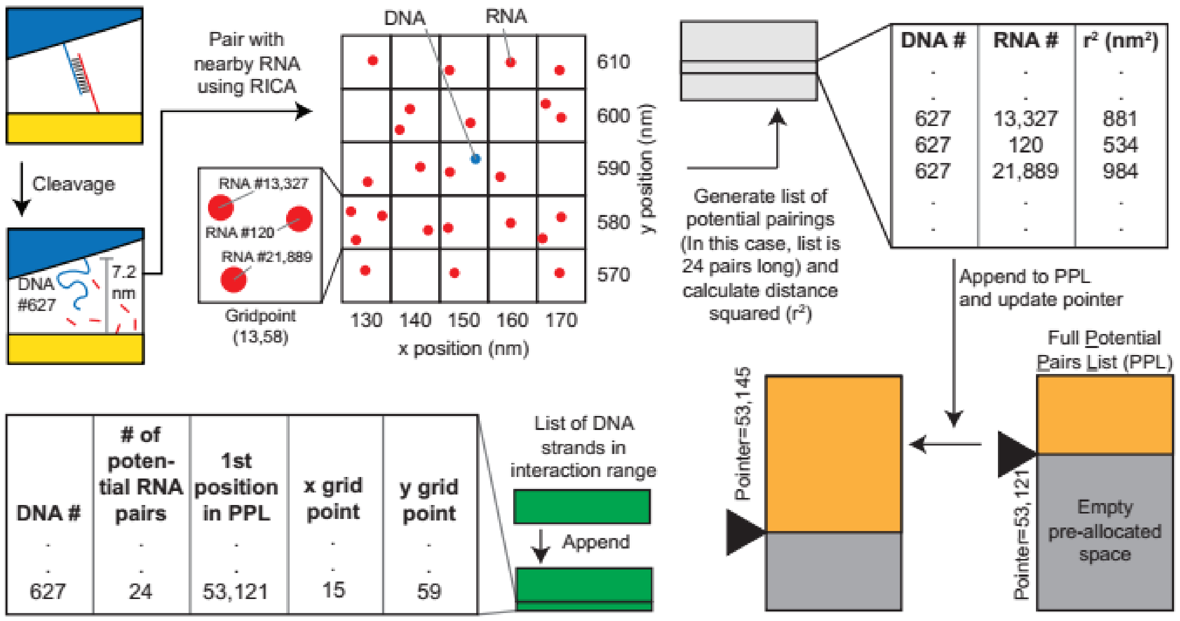
DNA #	x (nm)	y (nm)	z (nm)
.	.	.	.
.	.	.	.
10,521	2791	184	781
10,523	4561	3612	3792
10,524	4483	2582	4727
.	.	.	.
.	.	.	.

DNA #
.
.
8,102.
.

DNA #
.
.
772
.
.

DNA #	# of potential RNA pairs	1st position in PPL	x grid point	y grid point
.	.	.	.	.
.	.	.	.	.
6,202	44.	10,945	33	27
.	.	.	.	.
.		.	.	.

Figure S3: Initialization of RNA and DNA coordinates (part 2 – continued from previous supplemental figure).
Each DNA strand is assigned a unique integer index. DNA strand x-, y-, and z-coordinates are stored in a single list (denoted by brown rectangle) and a strand's position in this list corresponds to its index. Furthermore, DNA strand coordinates are divided into three groups according to height. Coordinates with height above z_2 are in the non-interacting range (dark gray zone on particle), while strands with height between z_1 and z_2 are in the intermediate range (light gray zone on particle). The indices of strands in each of these zones is kept in a corresponding list (denoted by dark and light gray rectangles). Coordinates with height below z_1 are classified as in the "interacting range" (green zone on particle). Indices of strands in the interacting range are also stored in a list, along with their x- and y- grid points (corresponding to the grid with 10 nm spacing developed for the RICA) and information on where to find data on that strand's potential interactions with RNA strands in the PPL (see the next figure for more information).


Cleavage
DNA #627
7.2 nm
Pair with nearby RNA using RICA
RNA #13,327
RNA #120
RNA #21,889
Gridpoint (13,58)
DNA
RNA
130
140
150
160
170
x position (nm)
610
600
590
580
570
y position (nm)
Generate list of potential pairings (In this case, list is 24 pairs long) and calculate distance squared (r^2)
Append to PPL and update pointer
Full Potential Pairs List (PPL)
Pointer=53,145
Pointer=53,121
Empty pre-allocated space
List of DNA strands in interaction range
Append

DNA #	RNA #	r^2 (nm^2)
.	.	.
.	.	.
627	13,327	881
627	120	534
627	21,889	984
.	.	.
.	.	.

DNA #	# of poten-tial RNA pairs	1st position in PPL	x grid point	y grid point
.	.	.	.	.
.	.	.	.	.
627	24	53,121	15	59

Figure S4: Keeping track of potential DNA-RNA pairs (part 1).
There are a very large number of RNA molecules that a single DNA strand can potentially form a tether with, and vice-versa. To make the calculation of association rates computationally tractable, we utilize a cutoff system and only consider potential DNA-RNA pairs that are within a certain distance of each other. However, even with cutoffs there can be millions of potential pairs. Here, we illustrate how we manage all of these calculations. For simplicity, we consider a single DNA molecule which has just been released from a tether via RNase H-mediated cleavage. First, the DNA strand's grid point is determined. Next, the RICA is used to obtain a list of all RNA molecules within two grid points of the DNA strand. Then, the squared-distance (r^2) between the DNA strand and each of the RNA strands in this list is calculated. An Nx3 list (where N is the number of RNA molecules within interaction range) containing DNA and RNA indices and r^2 is then constructed (denoted by light gray rectangle). Finally, this list is inserted into the full potential pairs list (PPL, denoted by gold and dark gray rectangle) at a position which is determined by a pointer value. Because the PPL is so large, we pre-allocate a significant amount of storage for the PPL, and the constructed list is inserted into the first empty space in the PPL. Following insertion of the constructed list, the pointer is then updated to point to the next empty space in the PPL so that this process can be repeated for the next guide DNA strand that enters interaction range. Finally, an entry is appended to the list of DNA strands in interaction range, which includes the DNA strands index, its x- and y- grid points, the position of the strand's segment of the PPL list, and the number of potential RNA pairing partners (these last two are used to extract information from the PPL). A similar treatment is applied towards 1) all guide strands in the interaction range during initialization, 2) guide strands following rupture, and 3) guide strands that enter the interaction range from the intermediate range.

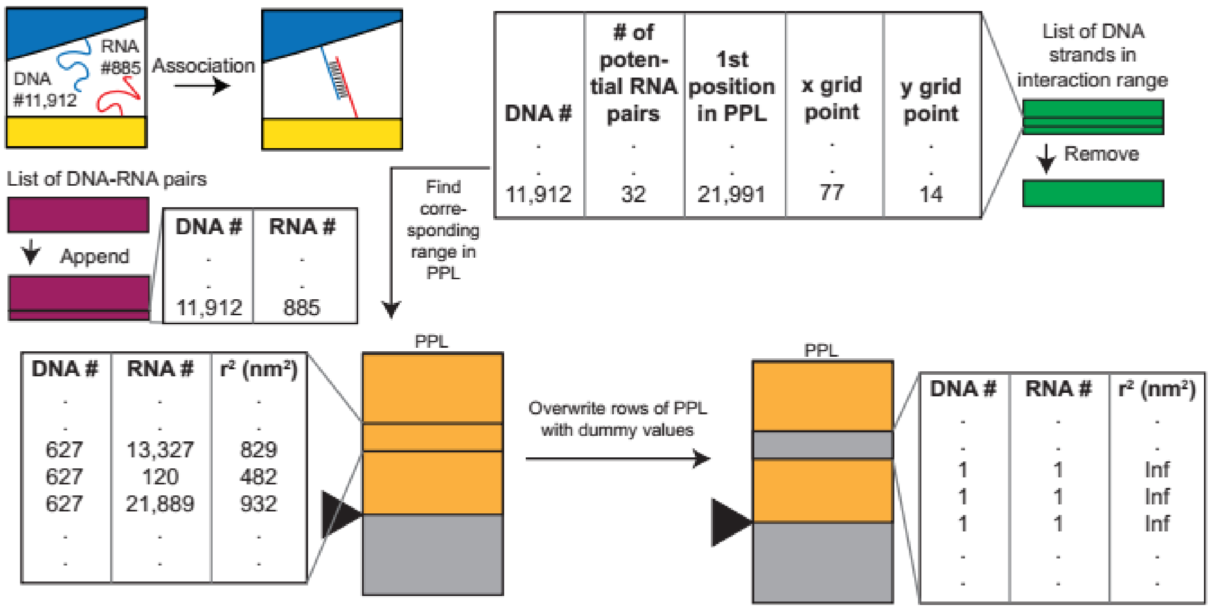


Figure S5: Keeping track of potential DNA-RNA pairs (part 2).
Here we show the treatment of variables after a DNA strand is paired. First, the indices of the newly paired DNA and RNA strands are added to the end of the list of DNA-RNA pairs (denoted by purple rectangle). Next, the guide strand's index is found in the list of interaction-range DNA. The guide strand's PPL list position and number of entries (which are stored in the list of DNA strands in interaction range) are used to overwrite all of the guide strand's pairing information in the PPL with dummy values. The guide strand's information in the list of DNA strands in interaction range is then removed.

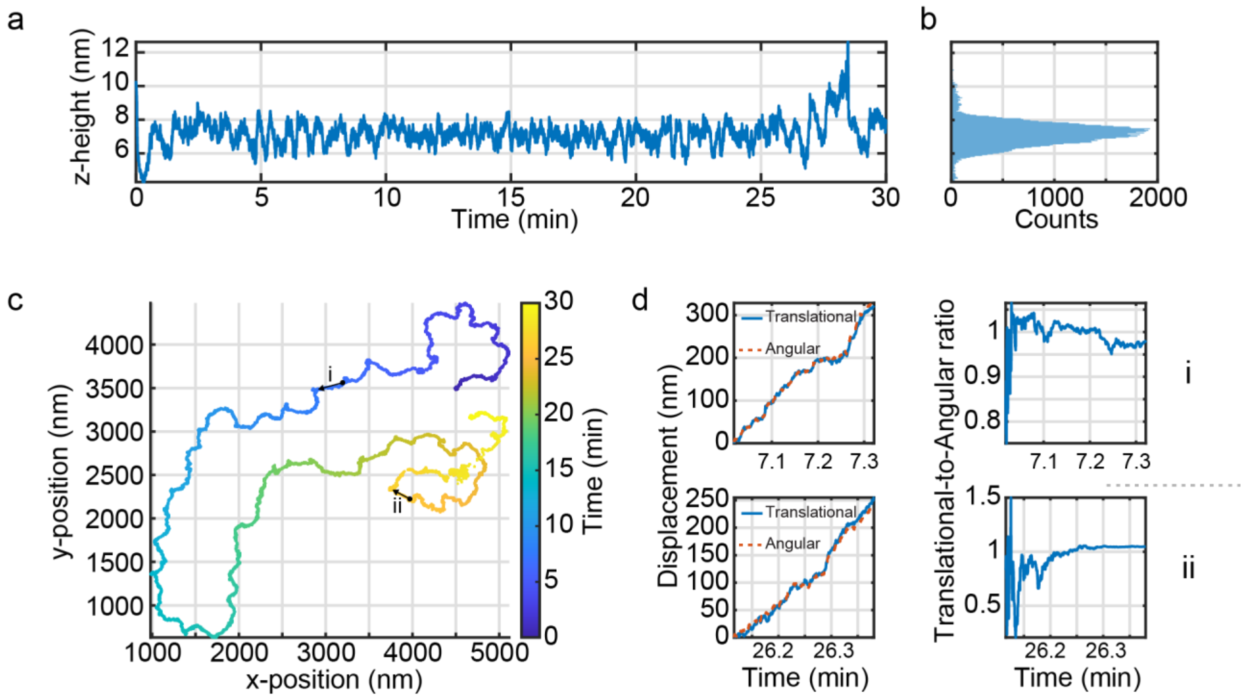


Figure S6: Z-height and rolling behavior of HPDMs simulated with RoloSim

a) A plot of z-height ($P_z - R$) over time from a RoloSim simulation at default conditions. The plot shows small fluctuations within a narrow range between ~6 nm and ~ 8 nm. The exceptions to this behavior occur at 1) the beginning of the simulation, when the HPDM comes into closer contact because the polyvalency increases to a very large value due to initial adhesion, and 2) towards the end of the simulation, when the HPDM reaches heights exceeding 10 nm because of detachment caused by depletion track-entrapment. **b)** A histogram of all z-height values shown in **a**. **c)** A scatterplot of all HPDM positions during the simulation, with color denoting simulation time. Two black arrows (marked i and ii) denote two stretches of near-linear motion, which were analyzed for rolling behavior. **d)** Four plots show that the HPDM undergoes pure rolling (i.e. no-slip rolling) during the periods marked in **c** as i (top two plots) and ii (bottom two plots). For each of the two rows, the left plot shows both the translational displacement (blue, solid line) and the angular displacement (orange dashed line – measured by calculating the angle of HPDM rotation around a horizontal vector perpendicular to the arrows marked i or ii and multiplying that angle by the radius of the HPDM). For each of the two rows, the right plot shows the ratio of the translational and angular displacement. Both plots converge to ~1, which suggests that motion is, on average, no-slip.

Supplemental Note 1: RoloSim Engineering Objectives

We had four specific goals that we sought to use RoloSim to accomplish in this work. The first one is broad while the second, third, and fourth focus on specific potential applications of HPDMs.

1) **To understand the scaling properties of autochemophoretic HPDM motion.** HPDMs have potential to be versatile tools for molecular recognition, nanopatterning, and force generation. In order to engineer HPDMs to serve specific roles, it is important to know how controllable HPDM properties – such as diameter, RNase H concentration, guide and fuel strand surface density, and tether duplex and spacer length – affect HPDM translocation properties such as force and velocity. RoloSim offers the opportunity to measure these scaling properties *in silico*. While there is no guarantee that RoloSim results are experimentally accurate, RoloSim does have value in that it offers a pure environment that is free of experimental error, heterogeneity between experiments, and measurement noise. RoloSim also allows for the exploration of HPDM properties in high-throughput (i.e. the testing of large numbers of conditions in parallel), which can generate hypotheses that can later be tested experimentally.

2) **To determine under what conditions HPDMs can be used for single molecule sensing.** The ability to perform molecular detection in a rapid, reliable, manner in resource limited settings is critical for responding to disease outbreaks, assessment of food and water quality, and for medical decision making. However, molecular detection often require samples to be transported to dedicated laboratories with specialized equipment for testing. Conversely, most field-based sensors do not provide appropriate molecular sensitivity. HPDMs have potential to address these issues, by serving as single molecule biomarker sensors. In principle, if F_{HPDM} can be reduced while preserving HPDMs' processivity and speed, they could be stalled by individual biomarkers. This would allow HPDMs to serve as tools for rapid, sensitive, on-site molecule detection using smart phone-based readers[1]. We sought to use RoloSim to understand how to reduce F_{HPDM} to levels that would make HPDMs suitable for single molecule sensing.

3) **To measure force generation by non-spherical HPDMs.** HPDMs have significant potential to serve as motors powering nano- and micromachines. However, the large size of existing HPDMs is a hindrance that will prevent future nanoscale applications. The specific importance of pN-scale force generation at the nanoscale is clearly illustrated by biological systems; molecular motors such as myosin and kinesin power countless tasks by generating and sustaining 1-10 pN of force via ATP-fueled powerstrokes. Currently, we do not have the means by which to engineer similar functionalities from the bottom-up, and so nanotechnological tools are largely viewed from a chemical rather than mechanical paradigm. The field of nanorobotics is still relatively new and there is a large need for demonstrations of technology that can perform fundamental tasks such as force generation. By miniaturizing force-generating HPDMs to the nanoscale, we can push the field forward and pave the way for the next generation of rationally-designed nanoscale mechanical systems. By designing RoloSim to allow for non-spherical HPDM geometries, we can understand how rod-shaped HPDMs – which can have the same surface contact area as spherical HPDMs with only a small fraction of the volume – can be used to scale the large force-generating potential of HPDMs down to the nanoscale (**Fig. S7**).

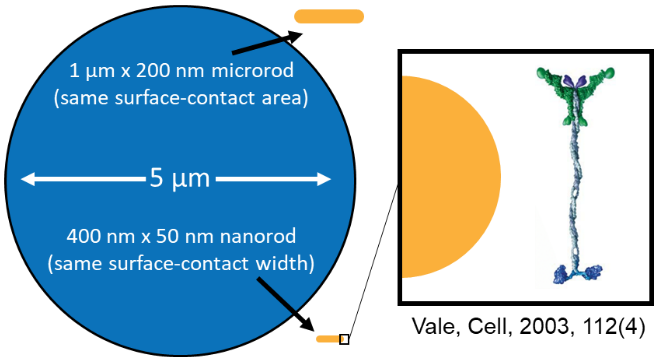


Figure S7: Depiction of the size difference between rod-shaped and spherical HPDMS
A crystal structure of a kinesin motor[2] is shown for scale.

4) **To understand how to "steer" HPDMs in a massively parallel fashion:** The HPDM-substrate contact zone is restricted to a width of <400 nm and this could potentially be reduced. This narrow, sharply bounded footprint suggests that HPDMs can serve as highly selective tools for "writing" nanoscale patterns with high precision. Furthermore, the ability to deposit streptavidin into depletion tracks using mechanical bond rupture as a selection mechanism[3] also demonstrates that HPDMs can perform additive, in addition to subtractive, lithography. If HPDM motion can be controlled using an externally-controlled force field, then all HPDMs can, in principle, be directed to follow the same path in parallel. Given the small size of an individual depletion track, HPDMs may be capable of performing relatively high-resolution nanolithography. By designing RoloSim to allow for the incorporation of externally-applied forces, we can thus understand how to steer HPDMs for massively parallel nanolithography.

Supplemental Note 2: Calculation of gravitational and electrostatic repulsion forces

The electrostatic repulsion between a DNA-coated microsphere and a nucleic acid-coated planar surface has been described previously[4]. The electrostatic repulsion potential energy (E_{el}) vs. height (h, measured from the lowest point of the microsphere) can be described when h is much larger than the Debye screening length (λ_D) using a two-parameter exponential function:

$$E_{el} = E_{el,0} \exp\left(-\frac{h}{\lambda_D}\right) \tag{1}.$$

The λ_D parameter is defined by the equation:

$$\lambda_D = \sqrt{\frac{\varepsilon_r \varepsilon_0 k_B T}{\sum_{all\ ions} (z_i\ e^-)^2 C_i \left(10^3 \frac{m}{L}\right) N_A}} \tag{2}$$

Where $\varepsilon_r = 78.5$ is the dielectric constant for water, $\varepsilon = 8.85 \times 10^{-12} \frac{C^2}{J\,m}$ is the permittivity of free space, $k_B = 1.38 \times 10^{-23} \frac{J}{K}$ is Boltzmann's constant, and $T = 298\ K$ is room temperature, $e^- = 1.6 \times 10^{-19}\ C$ is the charge of an electron, N_A is Avogadro's number, and z_i and C_i are the charge and molar concentration (respectively) of the ith ion. Plugging in values with 25 mM Potassium Phosphate, 37.5 Tris HCl, and 1.5 mM $MgCl_2$ and simplifying yields:

$$\lambda_D = 1.18\ nm \tag{3}$$

The $E_{el,0}$ parameter is calculated as:

$$E_{el,0} = 64\pi R \varepsilon_r \varepsilon_0 \tanh\left(\frac{e^- \Psi_{sphere}}{4 k_B T}\right) \tanh\left(\frac{e^- \Psi_{substrate}}{4 k_B T}\right) \left(\frac{k_B T}{e^-}\right) \tag{4}$$

Where the surface potential, Ψ, can be calculated from the surface charge density, σ, and the ionic strength, I:

$$\Psi_{\text{sphere}} = \sinh^{-1}\left(\frac{\sigma_{sphere}}{2\sqrt{\varepsilon_r \varepsilon_0 k_B T I N_A}}\right)\left(\frac{2 \mathrm{k_B T}}{\mathrm{e^-}}\right) \tag{5a}$$

$$\Psi_{\text{substrate}} = \sinh^{-1}\left(\frac{\sigma_{substrate}}{2\sqrt{\varepsilon_r \varepsilon_0 k_B T I N_A}}\right)\left(\frac{2 \mathrm{k_B T}}{\mathrm{e^-}}\right) \tag{5b}$$

where

$$\sigma_{sphere} = e_g \rho_{guide} \tag{6a}$$

$$\sigma_{substrate} = e_f \rho_{fuel} + e_a \rho_{anch} \tag{6b}$$

and

$$I = \sum_{all\ ions} \frac{z_i^2 C_i}{2} \tag{7}$$

where $e_g = 30e^-$ is the charge per guide strand, $\rho_{guide} = 91{,}000\ \mu m^{-2}$ is the HPDM surface density of guide strands, $e_f = 35e^-$ is the charge per fuel strand, $\rho_{fuel} = 50{,}000\ \mu m^{-2}$ is the substrate surface density of RNA fuel strands, $e_a = 30$ is the charge per DNA anchor strand (anchor strands are used to attach fuel strands to the substrate), and $\rho_a = 100{,}000\ \mu m^{-2}$ is the substrate surface density of anchor strands. Plugging in parameters and solving yields:

$$E_{el,0} = R\left(1.32 \times 10^{-11}\frac{J}{m}\right) \tag{8}$$

The gravitational force acting upon a sphere (minus the force of buoyancy) can be described as a function of h using the energy potential (U_g):

$$E_g = h\Delta\rho g\frac{4}{3}\pi R^3 \tag{9}$$

where $\Delta\rho = 995\ kg/m^3$ is the difference in density between the microsphere (silica) and surrounding media (water) and $g = 9.81 m/s^2$ is the gravitational constant. Plugging in parameters and simplifying produces

$$E_g = \left(40{,}886\frac{J}{m^4}\right)R^3 h \qquad (10).$$

The default HPDM setting is $R = 2.5 \times 10^{-6} m$. Plugging this in, adding the E_{el} and E_g terms, and rescaling energy to multiples of k_BT yields:

$$E_g + E_{el} = 8{,}024\exp\left(-\frac{h}{1.18\ nm}\right) + 0.155\frac{h}{nm} \tag{11}$$

Supplemental Note 3: Worm like chain (WLC) calculations of tether mechanics

To model the relationship between tether extension (r) and energy (E_{teth}), we first adopted a model from our previous work[3]. Specifically, we used an approximation of the worm-like chain (WLC) model[5] to calculate the relationship between force (F) and r:

$$\mathrm{r_{WLC}(F, L_0, P)} = \mathrm{L_0}\left(\frac{4}{3} + \frac{4}{3\sqrt{\frac{\mathrm{FP}}{\mathrm{k_B T}}+1}} - \frac{10e^{\sqrt[4]{900\frac{\mathrm{k_B T}}{\mathrm{FP}}}}}{\sqrt{\frac{\mathrm{FP}}{\mathrm{k_B T}}}\left(e^{\sqrt[4]{900\frac{\mathrm{k_B T}}{\mathrm{FP}}}}-1\right)^2} + \frac{\left(\frac{\mathrm{FP}}{\mathrm{k_B T}}\right)^{1.62}}{3.55 + 3.8\left(\frac{\mathrm{FP}}{\mathrm{k_B T}}\right)^{2.2}}\right) \quad (12)$$

where P is the persistence length, $\mathrm{L_0}$ is the chain contour length, and $\mathrm{k_B T} = 4.114\ \mathrm{pN\ nm}$ is thermal energy at room temperature. A tether is composed of double stranded DNA (dsDNA), single stranded DNA (ssDNA), and a double stranded DNA-RNA hybrid duplex. P and $\mathrm{L_0}$ (per base) are summarized in table S1. Therefore, the relationship between F and r for a tether composed of ssDNA, dsDNA, and DNA-RNA hybrid duplexes can be calculated using the equation:

$$\mathrm{r} = \mathrm{r_{WLC}(F, L_{0,ssDNA}, P_{ssDNA})} + \mathrm{r_{WLC}(F, L_{0,dsDNA}, P_{dsDNA})} + \mathrm{r_{WLC}(F, L_{0,DNA-RNA}, P_{DNA-RNA})} \quad (13)$$

and the energy, in units of pN $*$ nm can be calculated by numerically integrating the force-extension curve from 0 to r (**Fig. S8**):

$$\mathrm{E_{teth}} = \int_0^{\mathrm{r}} \mathrm{F(r')}\, d\mathrm{r'} \quad (14)$$

Table S1 – Worm-like-chain modeling parameters and segments*

Polymer Type	Persistence Length	Length per base	Segment Name	Segment length (nt or bp)
DNA-DNA duplex	53 nm	0.34 nm	Anchor	15
DNA-RNA duplex	100 nm	0.33 nm	Pairing region	15
Single-stranded DNA	1.07 nm	0.60 nm	Surface spacer	15
			Particle spacer	15
			Inter-duplex spacer	5
DNA-RNA duplex – transition state	2.7 nm	0.53 nm	Pairing region – tst	15

*Parameter estimates were taken directly from reference [6]. When multiple estimates were available, we used the estimate for the highest ionic strength condition available.

The tether energy-extension curve calculated using this method, as well as energy-extension cure calculated for 9-15 bp lock duplexes (which were not simulated for this study but are nonetheless possible to simulate with RoloSim). In addition, an accurate, computationally efficient is shown:

$$E_{teth} = (0.91\ k_B T)(\exp(0.0045r^2) - 1)$$

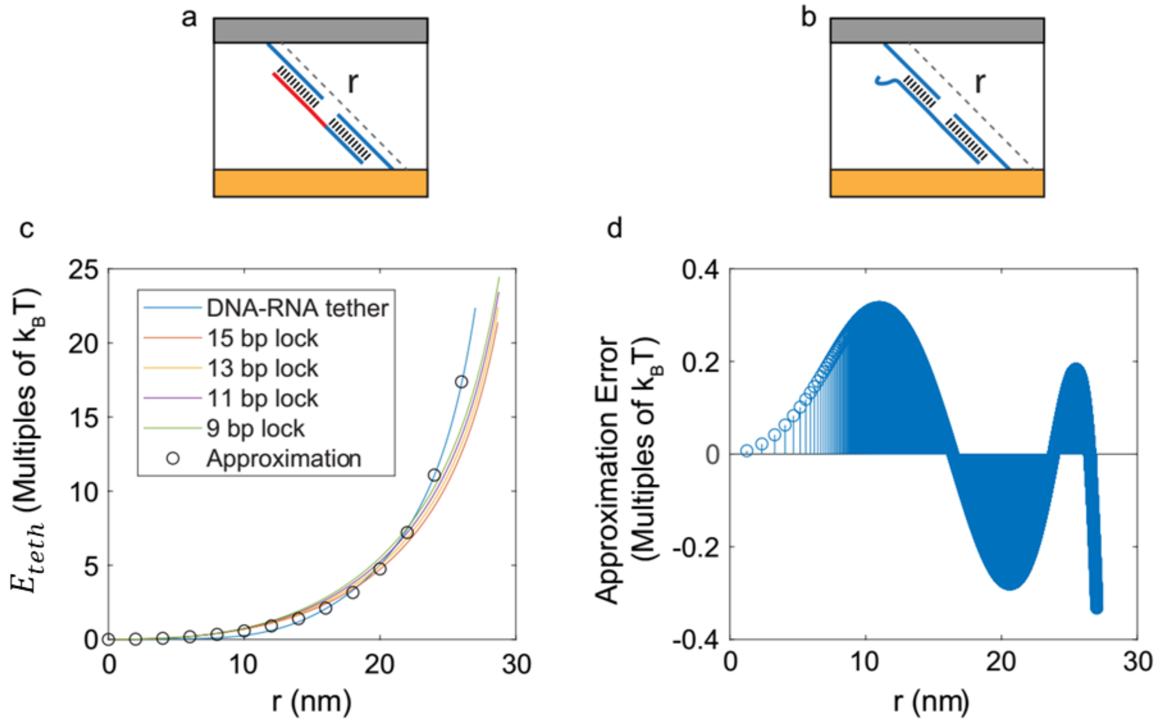


Figure S8: Tether WLC calculation and approximation
a) Depiction of a tether with end-to-end extension, r. **b)** Depiction of a DNA-DNA lock with a shorter duplex. **c)** Tether energy (E_{teth}) as a function of r. Calculations for various DNA-DNA locks are also shown. Calculations were performed using the Petrosyan approximation[5]. Circles show simple approximation used in this work. **d)** Absolute error of the simple approximation (with respect to the Petrosyan approximation), showing that the error does not exceed $0.35\ k_B T$ at relevant extensions of $r \leq 25\ nm$.

This representation can accurately represent tether mechanics as large extensions, but monotonically increases and thus fails to report resistance to compression (due to confinement within a small volume) at very small extension. To compensate for this issue, we adapted a Monte Carlo simulation method presented by Becker, Rosa, and Everaers[7] to understand E_{teth} with both extension and compression.

In this method, each ssDNA nucleotide (nt) or dsDNA basepair (bp) was represented as a point at fixed distance from its neighbors (the length per base shown in Table S1), and then a series of 10^7

iterations were applied to the construct via the Metropolis-Hastings algorithm. Each iteration consisted of either a crankshaft or pivot attempt (randomly selected). A pivot entails selection of a random point in the tether, followed by a counter-clockwise rotation of all downstream points (where upstream means closer to the point at which the tether is anchored to the substrate) around a random axis by an angle randomly sampled from the range $\pm 50°$. A crankshaft entails selection of two points between two and four bases away from eachother and rotation of all points between them by an angle randomly sampled from the range $\pm 50°$. The construct's post-attempt free energy was calculated as the sum of the bending energy of all non-terminal points. The bending energy for the i^{th} non-terminal point (e.g. a point that is bound to at least two additional points), $E_{bend,i}$, with 3D coordinate vector $\boldsymbol{r_i}$ is:

$$E_{bend,i} = -k_{s,i}\frac{(\boldsymbol{r}_i - \boldsymbol{r}_{i\leftarrow}) \cdot (\boldsymbol{r}_i - \boldsymbol{r}_{i\rightarrow})}{|\boldsymbol{r}_i - \boldsymbol{r}_{i\leftarrow}||\boldsymbol{r}_i - \boldsymbol{r}_{i\rightarrow}|} \tag{15}$$

where $k_{s,i}$ is the point's bending spring constant, which is related to the persistence length, L_p via the relation:

$$L_p = \frac{-|\boldsymbol{r}_i - \boldsymbol{r}_{i\leftarrow}|}{\ln\left(coth(k_s) - \frac{1}{k_s}\right)} \tag{16}$$

and $\boldsymbol{r}_{i\leftarrow}$ and $\boldsymbol{r}_{i\rightarrow}$ are the 3D coordinate vectors for the nearest upstream and downstream points, respectively, and $|\boldsymbol{r}_i - \boldsymbol{r}_{i\leftarrow}|$ is the length per base. Next, the bending energy was summed along the tether and the change in energy from the last iteration, ΔE_{bend}, was used to determine whether the pivot or crankshaft was accepted. Specifically, the pivot/crankshaft was accepted if $\Delta E_{bend}, < 0$ or, in the scenario that $\Delta E_{bend} > 0$, if $\exp(-\Delta E_{bend})$ was greater than a randomly-generated number sampled from the range of 0 to 1. To reflect attachment of the tether to the substrate, ΔE_{bend} was set to ∞ if any point in the tether exhibited a z-position below 0. Finally, to calculate E_{teth} from the Monte Carlo simulation result, the 10^7 iterations were binned into a 2-D histogram of end-to-end tether extension (r) and the z-height of the end of HPDM-bound end of the tether (r_z) (**Fig. S9, left**). Finally, we took the logarithm of the number of iterations in each bin to obtain E_{teth} as a function of r and r_z. Finally, we developed a simple, accurate approximation that accurately fits the Monte Carlo simulation data (**Fig. S9, right**) for E_{teth} as a function of the tether's end-to-end extension, d:

$$E_{teth} \approx \kappa_t \exp(\mathrm{K}\, r^2) + \frac{\kappa_c}{r_z} \tag{17a}$$

where κ_t and κ_c are spring constants for tension and compression, respectively, K (in units of nm^{-2}) is a fit parameter that is related to the persistence and contour lengths of the tether, and r_z is the tether's extension in the z-direction. While the first term reflects an energetic cost for extending the tether, the second term reflects an energetic cost for compressing the tether into a small volume between the particle and the substrate. In both cases, the energetic cost is entropic in nature because extension and compression both reduce the tether's conformational mobility.

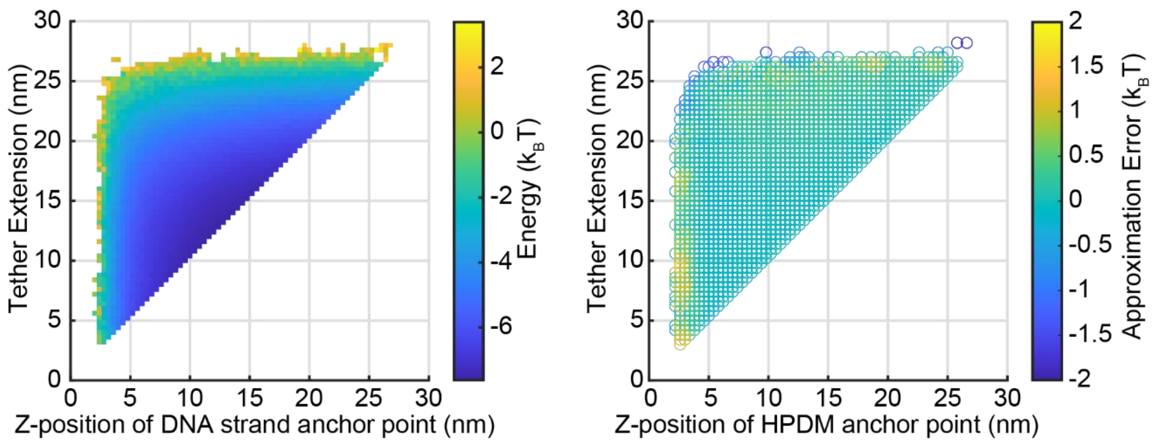


Figure S9: E_{teth} measured using WLC Monte Carlo simulations
Monte Carlo simulations[7] were performed with 100,000 timesteps of the Metropolis-Hastings algorithm described above. At each timestep, tether extension (r) and the z-position of the DNA-strand's anchor point on the particle (d_z) were recorded at each timepoint. Datapoints were grouped into $1\ nm \times 1\ nm$ bins and the energy of each bin was calculated as $E = -\log(n/r^2)$, where r^2 accounts for the increasing surface area with respect to r (note that the r^2 denominator allows for $E_{teth} > 0$ to be represented on the graph). We found that our result (left) fit accurately to a simple computationally-efficient approximation:

$$E_{teth} \approx (0.36\ k_B T)(\exp(0.0045r^2) - 1) + \frac{(1.4k_B T\ nm)}{d_z} \tag{17b}$$

The energy of the transition state for tether formation, E_{tst}, was calculated using a similar WLC-based approach (the numerical approximation, not the Monte Carlo simulation approach – see previous work[3]) by calculating the force-extension curve for a tether with a DNA-RNA transition state duplex (which has WLC parameters from single- or double-stranded oliconucleotids[6]).

$$\mathrm{r} = \mathrm{r}_{\mathrm{WLC}}(\mathrm{F}, \mathrm{L}_{0,\mathrm{ssDNA}}, \mathrm{P}_{\mathrm{ssDNA}}) + \mathrm{r}_{\mathrm{WLC}}(\mathrm{F}, \mathrm{L}_{0,\mathrm{dsDNA}}, \mathrm{P}_{\mathrm{dsDNA}}) + \mathrm{r}_{\mathrm{WLC}}(\mathrm{F}, \mathrm{L}^*_{0,\mathrm{DNA-RNA}}, \mathrm{P}^*_{\mathrm{DNA-RNA}}) \tag{18}$$

where transition state WLC parameters are shown in **Table S1**. Again, the energy, in units of pN nm can be calculated by numerically integrating the force-extension curve from 0 to d:

$$\mathrm{E}_{\mathrm{tst}} = \int_0^{\mathrm{d}} \mathrm{F}(\mathrm{r})\, d\mathrm{r} \tag{19}$$

Finally, the association rate, k_{hyb}, can be calculated as a function of the inter-strand distance, d, by using the Bell model (**Fig. S10**):

$$k_{hyb} = k_{hyb,0} \exp\left(\frac{-E_{tst}}{k_B T}\right) \tag{20}$$

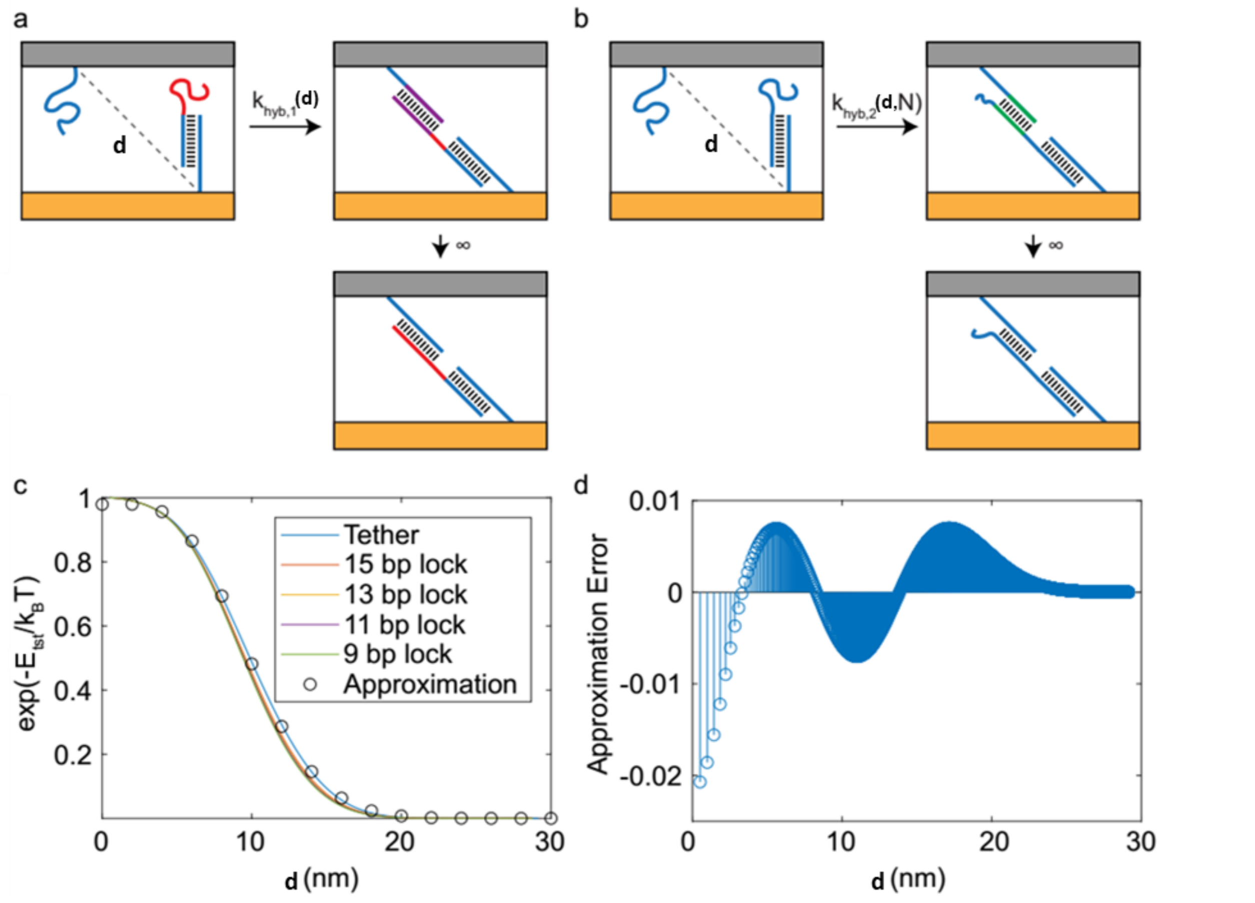


Figure S10: Depiction and calculation of association
a,b) Depiction of strand association, with the transition state duplex shown in purple (for DNA-RNA) or green (for DNA-DNA). **c)** WLC calculation of the exponent of tether energy (which is proportional to the tether association rate) as a function of the distance between strands. The calculation is shown for tethers of various lengths. A simple approximation described above is shown with black circles. **d)** Error of approximation as a function of d, showing less than 1% error at all relevant distances.

We found that, like with the tether calculation, E_{tst} could be accurately represented using a computationally efficient approximation:

$$E_{tst} \approx \kappa_t^* \exp(\mathrm{K}^* d^2) \tag{21}$$

where $\kappa_t^* = 0.8769$ and $\mathrm{K}^* = 0.00356$ are analogous to κ_t and K, but are specific to the transition state.

Supplemental Note 4: Optimization of RoloSim

We initially sought to obtain optimal parameters that would create the greatest resemblance between the RoloSim result and experimental observations. To summarize the process, we first sought to obtain a set of mechanical parameters (k_c, k_t, and k_t^* - we always used $K = 0.0045$) as well as the $k_{on,0}/k_{clvg}$ ratio in order to reproduce "equilibrium" quantities of HPDM translocation. We then scaled $k_{on,0}$ and k_{clvg} together to accurately reproduce the dynamic properties of HPDM translocation.

Metrics evaluated during optimization

We sought to optimize our RoloSim parameters to accurately reproduce the following observable quantities:

1. Depletion track width: depletion track width was previously quantified[1] by taking linescans of depletion tracks and quantifying the full-width half maximum (FWHM) of the tracks. Previously, we utilized structured illumination microscopy (SIM, a superresolution imaging technique) to obtain an estimate of $FWHM = 380\ nm$. As a best-attempt to replicate this scenario, we used an automated process in which simulated data is used to create a mimic SIM image and then linescans are taken at defined intervals along the simulated HPDM's trajectory (**Fig. S11**).

2. Depletion track depth: we generally find that roughly 50% of the fuel strands beneath the HPDM are cleaved[1,3,8]. While this estimate is not as precise as the FWHM measurement, the finding that a substantial fraction (~50%) of fuel strands in depletion tracks remain uncleaved is experimentally reproducible. We quantify width and depth together by fitting linescans of simulated SIM images to inverted Gaussian functions. We then calculate the FWHM and depth directly from the parameters of the best-fit Gaussian (**Fig. S11**).

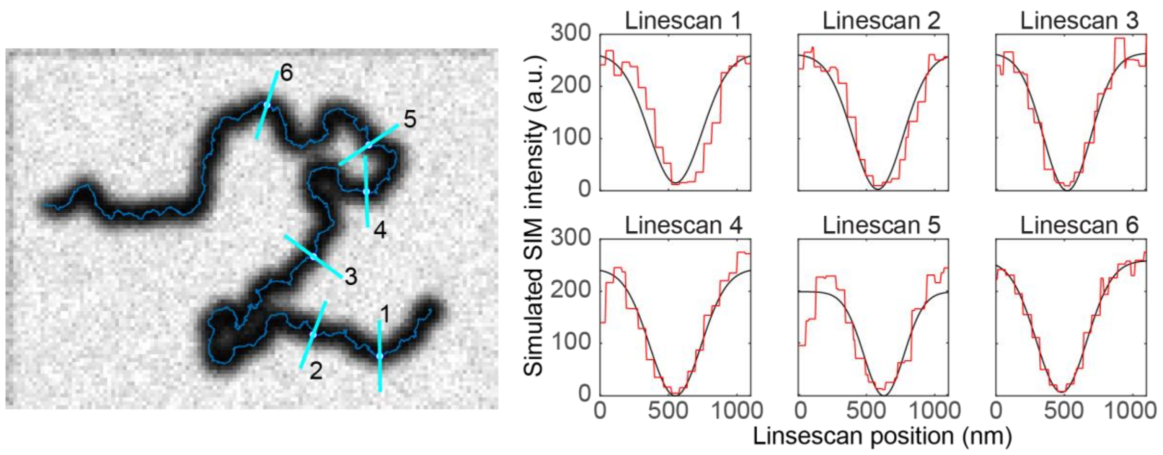


Figure S11: Automated analysis of simulated depletion tracks

An automated analysis method was used to analyze depletion track properties. Briefly, The trajectory was queried at an interval of every 2,000 timepoints. The first and last query points were then excluded to avoid sampling the track at the ending or beginning of the image. For each query point, all trajectory points within a 550 nm radius were collected. From these trajectory points, two durations were recorded: the time between the first and the last point, and duration of the continuous set of trajectory points that the query point belongs to. If the ratio of these durations is greater than 1.1, the query point is discarded. This process is intended to exclude query points where the track switched back on itself (linescans of such query points could artificially bias the width measurement upwards). Each of the remaining query points was then used to draw a linescan 733 nm in length and with an orientation that is perpendicular to the major axis of the set of trajectory points within the 550 nm radius cutoff. This linescan was then fit to a Gaussian function, which was then analyzed to extract the track depth (depth of the Gaussian) and width (FWHM of the Gaussian).

3. Path persistence length: we previously found that HPDMs exhibit persistent motion at intermediate timescales (i.e. HPDMs generally appear to move in a straight line when observed for durations on the order of tens of seconds)[3]. At longer timescales of several minutes or more, the directionality of spherical HPDMs is randomized. Our simulated trajectory should also exhibit these properties. This intermediate timescale persistence can be quantified using the path persistence length (L_{traj}) metric[9], which is analogous to the persistence length parameter of the WLC model and denotes the distance across which a trajectory's directionality remains correlated. Experimental measurements of L_{traj} are generally on the order of 1-2 μm (**Fig. S12**). However, because this quantification of L_{traj} includes contributions from some HPDMs that are stalled due to nonspecific interactions between the HPDM and the surface, we take 2 μm as our lower-bound cutoff.

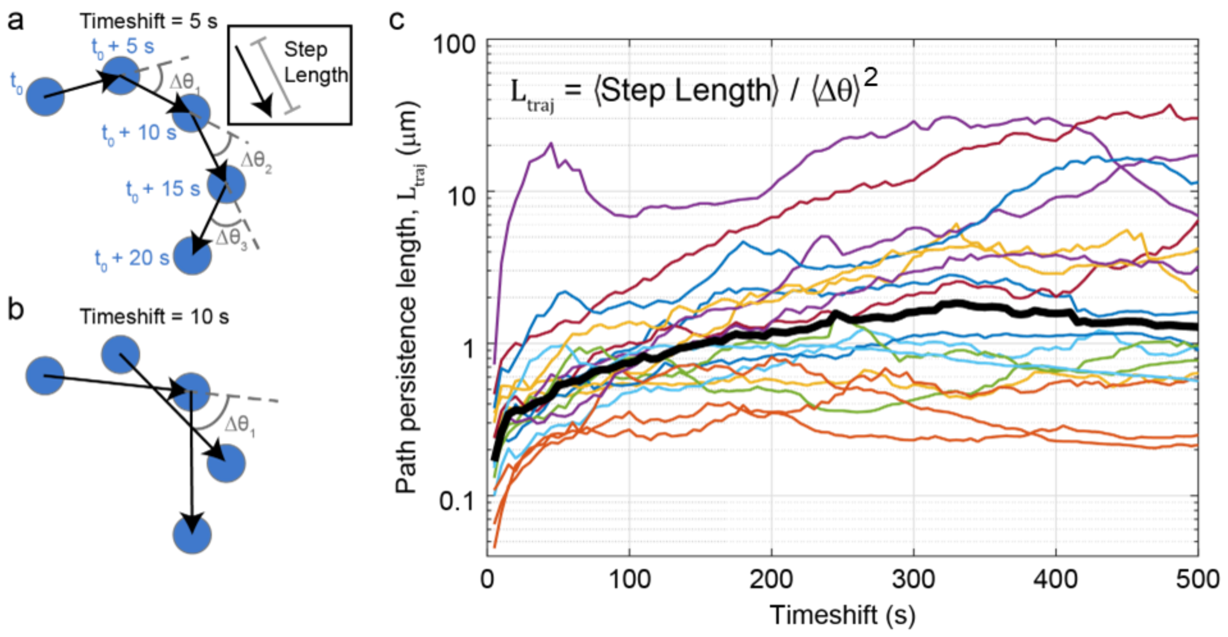


Figure S12: Experimental measurement of path persistence (L_{traj}).

a) Depiction of $\Delta\theta$ and step length (inset) for a timeshift of 5 seconds when the HPDMs are imaged at a 5-second interval. Five representative HPDM positions are shown, along with four "steps" and denotations of $\Delta\theta$ measurements. **b)** Same as **a**, but for a timeshift of 10 seconds. **c)** L_{traj} – which is calculated as the average step length divided by $\langle\Delta\theta\rangle^2$ – as a function of timeshift for 18 individual HPDMs, as well as the median of the 18 HPDMs (thick black curve). At short timescales, L_{traj} appears to be artificially low, which is an expected result of localization error. At intermediate timescales (~1-5 minutes) the median L_{traj} value converges to a range of $\sim 1-2\ \mu m$. Trajectories were calculated by re-analyzing supplemental movie 1 from ref. 1 using an improved particle tracking code presented in ref. 3 (and used in this work).

4. Stall force: in previous experiments[3], we quantified the force generated by HPDMs (F_{HPDM}) as $F_{HPDM} \approx 150\,pN$. This measurement was obtained using high throughput particle tracking experiments involving hundreds of individual HPDMs. Because this level of throughput is not practical for RoloSim, we adopted an alternative approach. Briefly, we restart a completed RoloSim simulation at a random timepoint and "pin" the HPDM by making one random tether unbreakable. The simulation is then allowed to continue until the HPDM "stalls". Stalling is determined to have occurred when the polyavalency (the number of tethers) drops to five, which we have found is too low to ever occur under normal conditions. A 1-second smoothing average is then applied to the force vs. time curve, and the maximum force of this smoothed curve is then determined to be the stall force. This process is repeated several dozen times to obtain an average estimate of F_{HPDM} (**Fig. S13**).

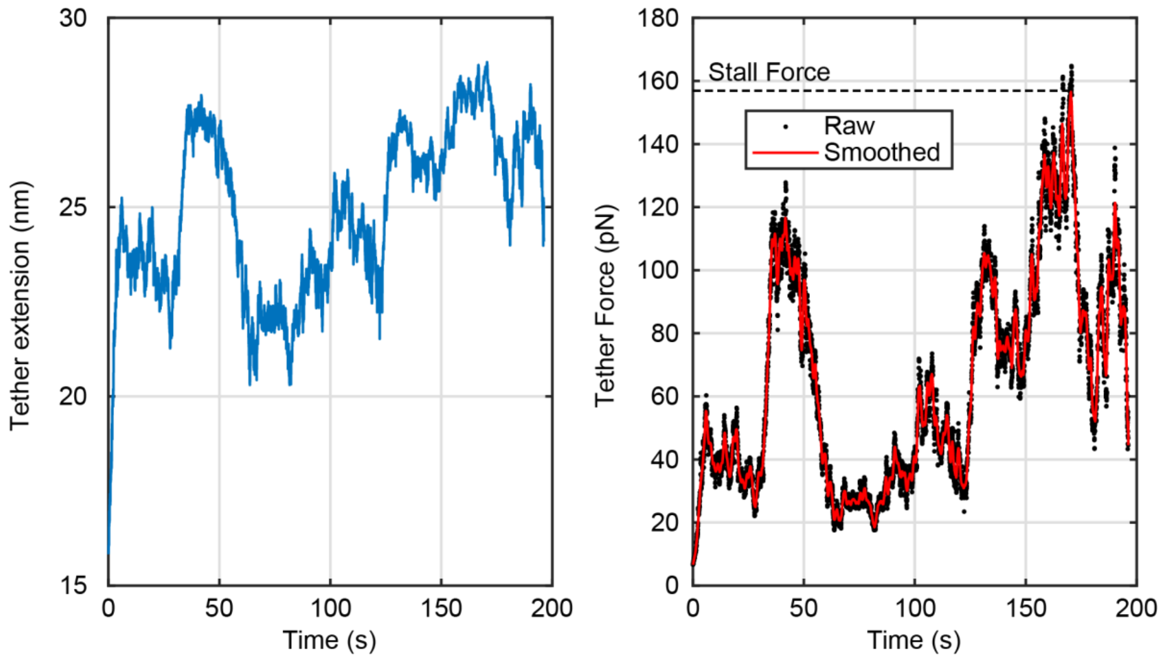


Figure S13: RoloSim-based estimation of F_{HPDM}
Because the previous experimental methods for F_{HPDM} estimation require very high throughput, we developed an alternative method to estimate F_{HPDM} using RoloSim. In this method, a completed simulation is restarted at a random timepoint, and then a random tether is selected from the set of existing tethers at that time point. That tether is set as an indestructible tether, that can be neither cleaved nor ruptured. The simulation is then allowed to continue until the polyvalency drops below five. The tether's extension vs. time curve is stored (left). This curve is then used to calculate a force vs. time curve (right), which is then smoothed with a 1-second rolling average. The maximum value of the smoothed curve is then recorded as the stall force. This process is repeated ~100 times per condition to obtain a robust estimate of F_{HPDM}.

5. Average velocity (v_{avg}): we generally find that the average HPDM velocity is roughly 1.3 $\mu m/min$. This estimate lumps contributions from mobile HPDMs as well as low-mobility HPDMs that have become "entrapped" within their own depletion tracks (**Fig. S14**).

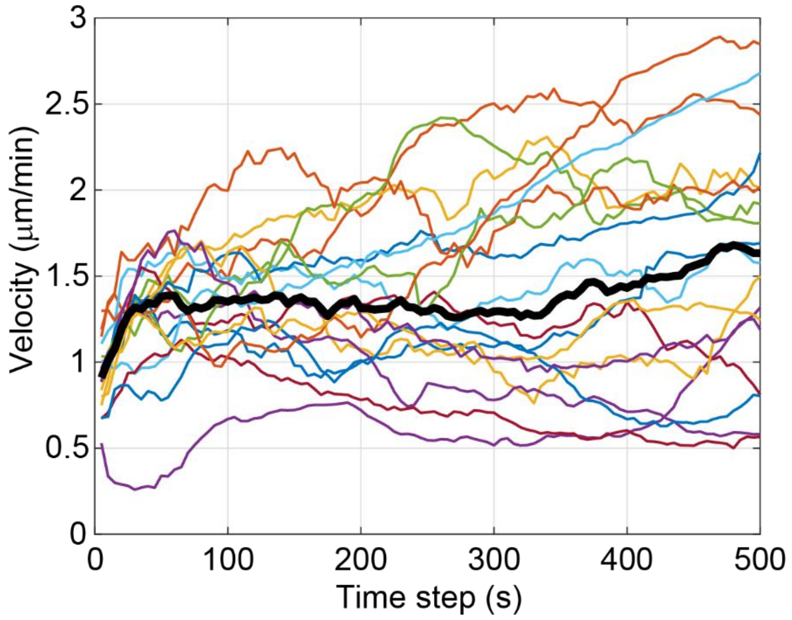


Figure S14: Experimental measurement of average velocity, v_{avg}
Average HPDM velocity (v_{avg}) as a function of timeshift for 18 individual HPDMs, as well as the median of the 18 HPDMs (thick black curve). At short timescales, v_{avg} appears to be artificially low, which is an expected result of localization error. At intermediate timscales (~1 to 5 minutes) the median v_{avg} value converges to $\sim 1.3\ \mu m/min$. Trajectories were calculated by re-analyzing supplemental movie 1 from ref. 1 using an improved particle tracking presented in ref. 3.

6. The kinetic rate constant for RNase H-mediated hydrolysis (k_{clvg}): through experiments, we previously estimated $k_{cat} = 25\ \mathrm{min}^{-1} = 0.4\ s^{-1}$, where k_{cat} is the maximal kinetic rate constant for RNase H-mediated cleavage according to the Michalis Menten kinetic framework (i.e. $k_{clvg} = k_{cat}$ when $[RNase\ H] = \infty$). Accordingly, the k_{clvg} value obtained from our optimization process should not exceed this value.

These six metrics can be divided into three categories: depletion track width and depth are timescale-independent, meaning that they are equilibrium properties that are calculated without any consideration of the dynamics of HPDM translocation; path persistence length and F_{HPDM} both exhibit a light timescale-dependence but can be assumed to be largely timescale independent (L_{traj} depends slightly on the selected step length, **Fig. S12**, and the simulated F_{HPDM} calculation depends on a 1-second smoothing average); and v_{avg} and k_{clvg} are completely timescale-dependent quantities.

Optimization of mechanical parameters
We started the optimization process by using parameters calculated according to the WLC model. Specifically, we found that $K = 0.0045$ allowed for an accurate representation of the tether's force-extension curve. While the Petrosyan approximation yielded best-fit values of $\kappa_t = 0.91\ k_BT$ and $\kappa_c = 0\ k_BT$, our Monte Carlo WLC simulations produced best-fit values of $\kappa_t = 0.36\ k_BT$ and $\kappa_c =$

$1.4\ k_BT\ nm$. This result suggested that tethers, when bound on either end to two surfaces, are easier to extend than would be expected from the conventional WLC model. The conventional WLC model calculation for the association transition state also yielded a best-fit of $k_t^* = 0.88\ k_BT$ and $K^* = 0.0036$. The disparity between the conventional and Monte Carlo WLC results suggested that a similar disparity may exist for the transition state tether. Therefore, we varied κ_t^* across two orders of magnitude (from 0.088 to 8.8). We also varied $k_{on,0}/k_{clvg}$ across two orders of magnitude. In total, we ran one 60 minute simulation for most combinations of 13 κ_t^* values and 13 $k_{on,0}/k_{clvg}$ values (several combinations were not simulated according to user choice, particularly in the extremes of the parameter space). Of these simulations, a total of 110 finished successfully. For each condition that finished successfully, we quantified track width and depth, L_{traj}, and F_{HPDM} and then adjusted the timescale of the simulation (by adjusting $k_{on,0}$ and k_{clvg}, and Δt post-hoc) to obtain $v_{avg} = 1.3\ \mu m/min$ (we initially ran simulations with an arbitrarily-selected $k_{clvg} = 60\ min^{-1} = 1\ s^{-1}$). Following timescale adjustment, we also quantified k_{clvg}. Finally, we quantified the steady-state polyvalency of each HPDM by measuring the number of tethers at 1-second intervals throughout the simulation.

The results of these simulations, shown in **Fig. S15** and **Fig. S16**, show three general behaviors. At low $k_{on,0}/k_{clvg}$ and high κ_t^*, tether association is much slower than cleavage, which results in a low steady-state polyvalency, very low force ($F_{HPDM} < 1\ pN$) diffusive motion (i.e. $L_p < 0.4\ \mu m$) and a lack of a discernable depletion track. At high $k_{on,0}/k_{clvg}$ and low κ_t^*, association is much faster than cleavage, resulting in a wide, deep depletion track (with 100% consumption of all strands in the HPDM's path), high force ($F_{HPDM} > 10\ pN$), and high persistence ($L_p > 2\ \mu m$). The third category exhibited intermediate properties, with depletion track depths between 10% and 90%, $FWHM$s close to 400 nm, and L_{traj} within the ideal range of $1\ \mu m < L_{traj} < 2 \mu m$.

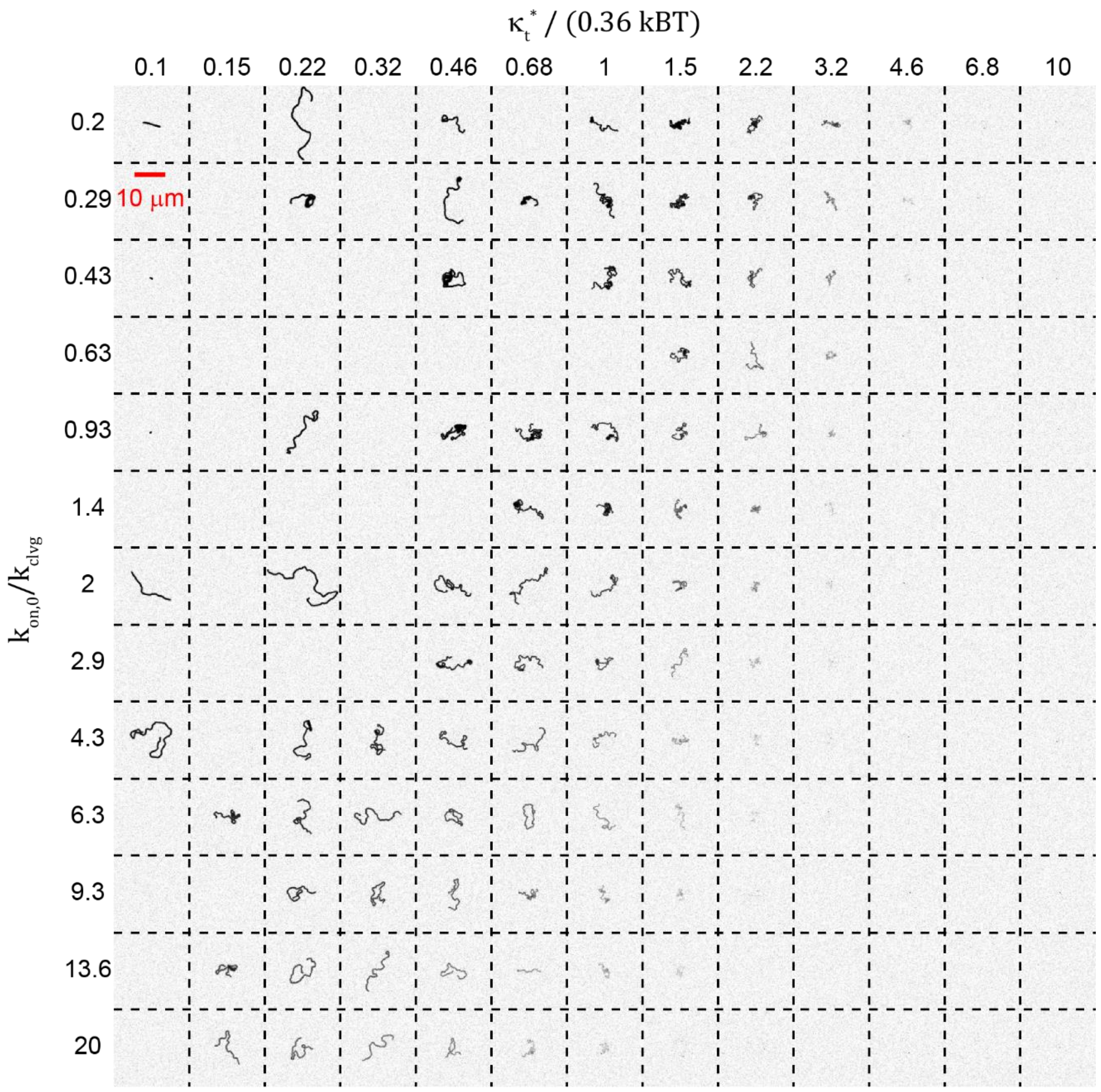
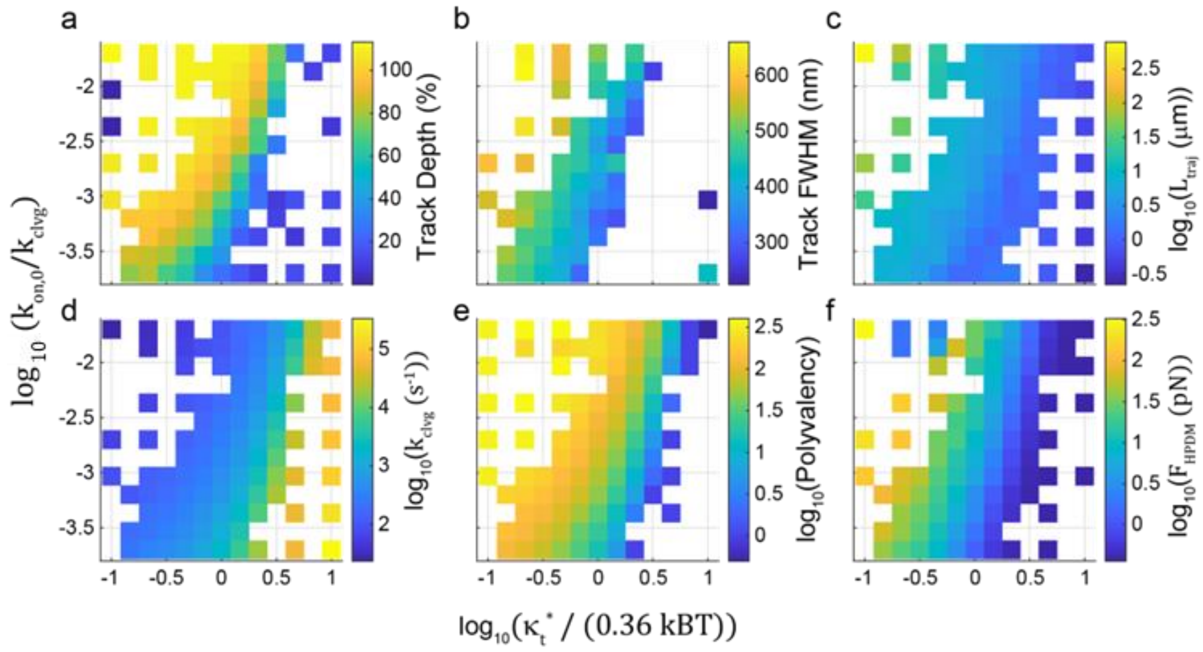


Figure S15: Depletion tracks from simulations run with varying κ_t^* and $k_{on,0}/k_{clvg}$ values

A 13×13 grid of simulated depletion track images at different parameter combinations as denoted on the top and left. Not all conditions are represented due a combination of 1) early termination of a small number of simulations due to various computational errors (some of which were corrected following this round of optimization), and 2) efficient selection of conditions to conserve computational resources.



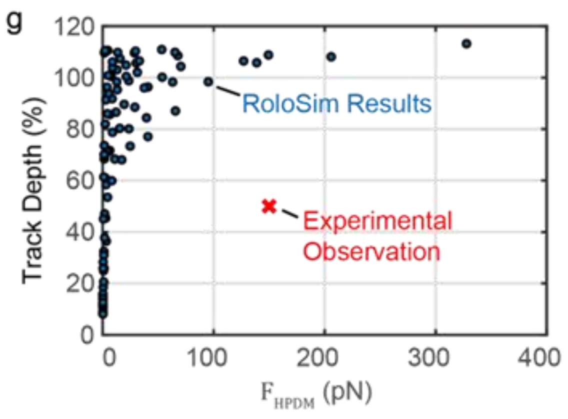


Figure S16: Metrics for first round of optimization
a-f) Various metrics quantified from the results shown in **Fig. S15**. The key takeaway for the purposes of optimization is **g)** that no pair of parameters results in the experimental observation of a 50% track depth and $F_{HPDM} = 150\ pN$ (or even $F_{HPDM} > 10pN$).

Unfortunately, no condition produced a depletion track depth below 100% and $F_{HPDM} > 10\ pN$ (**Fig. S16g**). This finding contrasts starkly with experimental observations, wherein the track depth is ~50% and $F_{HPDM} \approx 150\ pN$. Instead, conditions that produced a depletion track depth of between 10% and 90% resulted in low force magnitudes that would not be capable of rupturing even a single DNA-DNA duplex, as previously observed in extensive and unambiguous detail[3].

While the reason for this disparity from experimental observation is unclear, we think the primary cause of this issue may be RoloSim's failure to account for potential heterogeneity of the activation energy of tether association; if strands have different levels of accessibility and/or nonspecific association with their underlying surfaces, then different strands could have different hybridization rates. We believe that a method that allows for heterogeneity in association activation energy could allow for simulated HPDMs to achieve higher steady-state polyvalency (and thus higher F_{HPDM} measurements) while still producing a track depth of ~50%. However, the exploration of such hypotheses will be the subject of future work.

Having failed to obtain a pair of κ_t^* and $k_{on,0}/k_{clvg}$ values that accurately reproduces all of our experimentally-observed metrics simultaneously, we sought to obtain a reasonable set of parameters that could be used for this initial study. Specifically, we relaxed our constraint on track dimensions and sought a condition in which $F_{HPDM} \approx 150\, pN$ and $L_p > 1\ \mu m$. We started by increasing κ_c and κ_t ten-fold, with the reasoning that increasing tether stiffness would result in increased force generation. (Ideally, future versions of RoloSim will include more rigorous parameterization). We then varied $k_{on,0}/k_{clvg}$ across two order of magnitudes, as before. We ran seven independent simulations for each of seventeen $k_{on,0}/k_{clvg}$ values ranging from 0.0001 to 0.01 and quantified L_{traj} and F_{HPDM} for each (**Fig. S17**). We found that $L_{traj} > 2\mu m$ at all conditions (**Fig. S17d**). At no condition was $FWHM \leq 380\, nm$ (this may potentially be an artifact of the automated track analysis method, which down not always result in linescans being taken perpendicular to the depletion track, leading to over-estimation of $FWHM$) (**Fig. S17b**). Across the range of conditions tested, F_{HPDM} increased from ~10 pN to ~300 pN (**Fig. S17e**). Similarly, track depth increased from ~10% to 100% (**Fig. S17c**). Finally, at no condition was k_{clvg} lower than the previously measured $k_{cat} = 0.4\ s^{-1}$ (**Fig. S17e**). This result likely stems from the same issues that require the 10-fold scaling of κ_t and κ_c – which we hope to resolve with future iterations of RoloSim. Ultimately, we chose the condition of $k_{on,0}/k_{clvg} = 0.0042$, which produced $F_{HPDM} \approx 170\, pN$ and $L_{traj} \approx 7\ \mu m$. Temporal scaling to obtain $v_{avg} = 1.3\ \mu m/min$ resulted in. $k_{on,0} = .0091\ s^{-1}$ and $k_{clvg} = 2.2\ s^{-1}$.

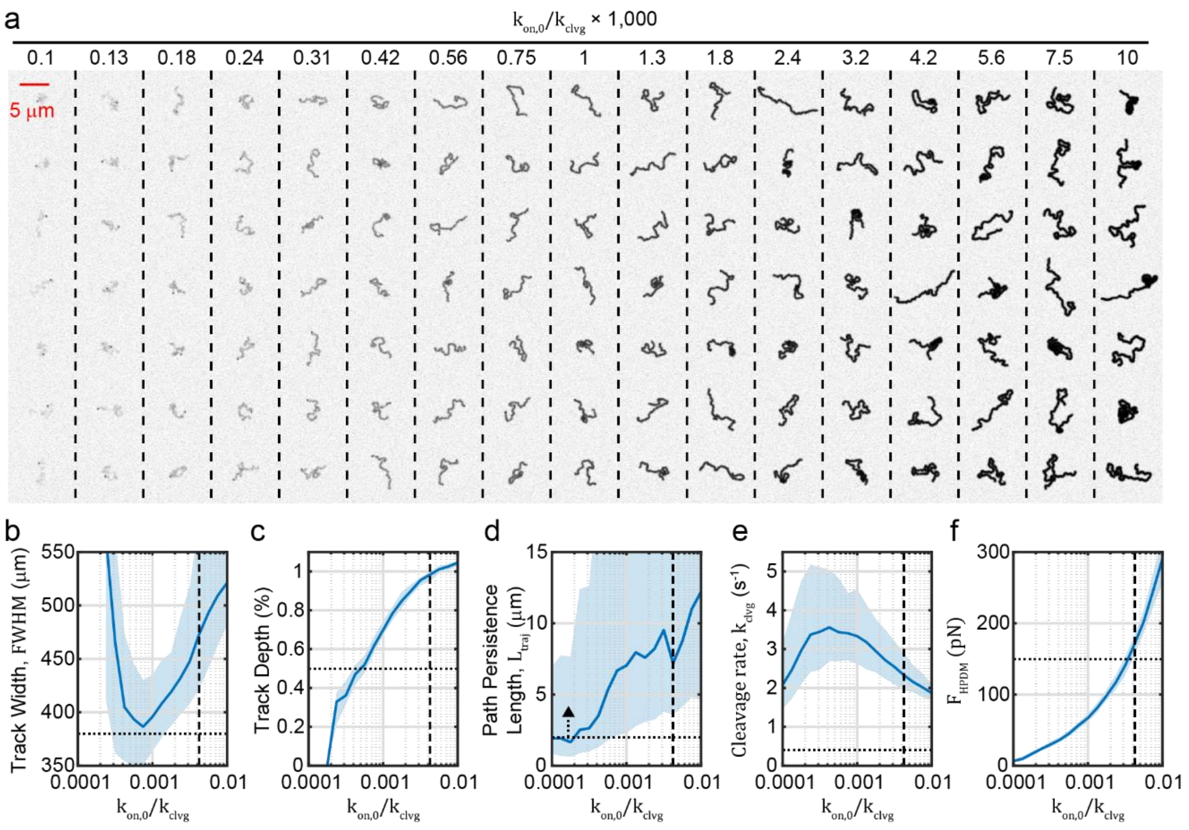


Figure S17: RoloSim optimization results, round 2

a) Montage of depletion tracks depicting results of seven simulations at each of seventeen $k_{on,0}/k_{clvg}$ values. Image brightness is scaled to 0% to 100% RNA fuel depletion. **b-f)** Median +/- inter-quartile range of several quantifiable metrics shown as a function of $k_{on,0}/k_{clvg}$. The horizontal dotted line denotes experimental observation, while the vertical dashed line denotes the selected condition.

Supplemental Note 5: Estimating HPDM translocation dynamics following the formation of a single tether

To determine whether microparticle position and orientation could be expected to equilibrate between timesteps (30 ms), we calculated HPDM relaxation dynamics on sub-timestep timescales. We used the kinematic equation for a sphere experiencing force applied by a newly-formed tether ($F_{lateral}$) and the force of drag from the surrounding fluid (F_{drag}) to calculate an HPDM's theoretical lateral displacement (x_{relax}) after tether formation:

$$\ddot{x}_{relax} = m\left(F_{lateral} - F_{drag}\right) \tag{22}$$

where $\ddot{x}_{relax}$ is the acceleration of the HPDM and m is the HPDM's mass. In the absence of a nearby surface, F_{drag} can be calculated for a sphere using Stoke's law (i.e. $F_{drag} = 6\pi\eta\mathrm{R}\dot{\mathrm{x}}_{relax}$, where η is the dynamic viscosity - 0.001 Pa s for water – and $\dot{\mathrm{x}}_{relax}$ is the velocity). However, because the HPDM is very close to a surface, Faxen's law, a modification to Stoke's law that accounts for the increased drag experienced by a microsphere moving near a surface[10], was used instead:

$$F_{drag} = \frac{6\pi\eta\mathrm{R}\dot{\mathrm{x}}_{relax}}{1 - \frac{9}{16}q + \frac{1}{8}q^3 - \frac{45}{256}q^4 - \frac{1}{16}q^5} \tag{23}$$

where $q = R/P_z$. This modification results in a ~3x increase in F_{drag} relative to what would be calculated using Stoke's law. To reflect equilibration of the system as the HPDM approaches its new energetic minimum, we approximated $F_{lateral}$ as decreasing linearly from the initial force-imbalance immediately after tether formation ($F_{lateral,0}$) to zero as x_{relax} approaches the maximum displacement ($\Delta\mathrm{x}_{\mathrm{relax}}$):

$$F_{lateral} = F_{lateral,0}\left(1 - \frac{x_{relax}}{\Delta x_{relax}}\right) \tag{24}$$

We next parameterized this model using estimates from a representative RoloSim simulation with default parameter settings. The lateral component of the tensile force on a tether immediately after association ($F_{lateral,0}$) was, on average, 1.56 pN (the 5th and 95th percentiles were 0.11 pN and 8.58 pN, respectively). The median lateral HPDM displacement after each timestep (Δx_{relax}) was 1.50 nm (5th and 95th percentiles: 0.27 nm and 4.97 nm). The median distance between the bottom of the HPDM (e.g. $P_z - R$) and the surface was 7.20 nm (5th and 95th percentiles: 6.02 nm and 8.29 nm). We next used all 27 combinations these median and outer percentile estimates to calculate relaxation dynamics. Specifically We used numerical integration to calculate x_{relax} vs time via Euler's method with a timestep of 10^{-7} s from 0 s to 0.03 s (the duration of one RoloSim timestep). For each timecourse, we calculated the equilibration time as the time taken for the HPDM to move 90% of the total distance (e.g. the time when $x_{relax}/\Delta x_{relax} = 0.90$). We also calculated the Reynold's number at all timepoints ($Re = 2R\dot{\mathrm{x}}_{relax}\rho/\eta$ where $\rho = 1{,}000\ kg/m^3$ is the density of water) to ensure that fluid flow was laminar, a condition – which is only satisfied at $Re < 10$ – that must be met for Faxen's law to be accurate.

Note that the equation for Faxen's law only applied to motion parallel to the surface; motion perpendicular to the surfaces, which experiences a much greater drag force, was not considered here because changes in P_z were very small between timesteps (median: 0.030 nm, 5th and 95th percentiles: 0.002 and 0.113 nm).

Our results (**Fig. S18**) revealed that, under all conditions tested, the particle's position equilibrated well before the 30 ms timestep length; the equilibration time ranged from ~0.009 ms to 15 ms, with the average of the 27 simulations being 2.4 ms. Re never surpassed 3×10^{-4} for any condition. The slowest relaxations occurred at the lowest P_z and the highest $F_{lateral}$. Similar relaxation dynamics could be expected for HPDM rotation. These results suggest that our approach, wherein the position and orientation of the HPDM is set to the energetic optimum at each timestep, is satisfactory because the actual relaxation dynamics should occur on a much faster timescale than the RoloSim timestep.

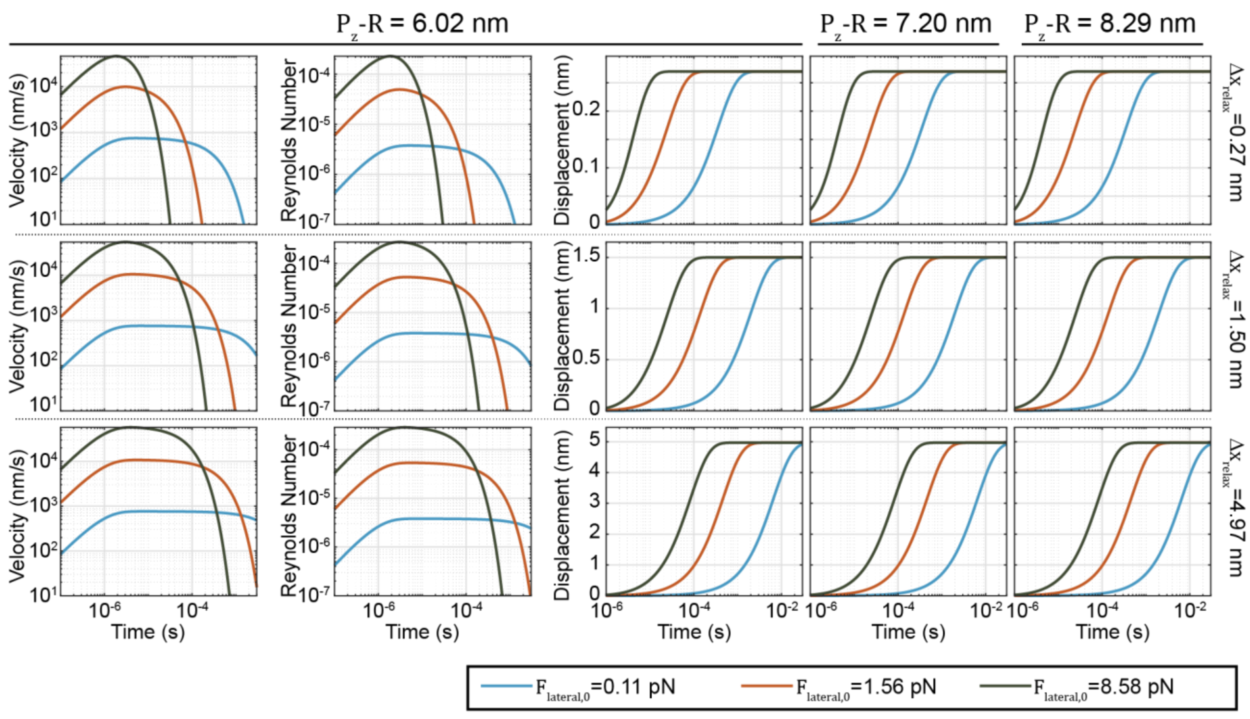


Figure S18: Relaxation dynamics calculations

Plots show the velocity ($\dot{x}_{relax}$), Reynold's number (Re), and displacement (x_{relax}) as a function of time (note the logarithmic-scale axes) at various $P_z - R$, Δx_{relax}, and $F_{lateral,0}$ combinations as denoted via the legend and labels on the top and right side of the figure.

Supplemental Note 6: Assessing the accuracy of the energetic minimum approximation

To assess the potential effect of thermal fluctuations on HPDM position and orientation, we performed simulations of an HPDM undergoing thermal fluctuations according to the Metropolis-Hastings method. To accomplish this, we carried out the following algorithm:

1. Restart a completed RoloSim simulation at a random timepoint, with some exceptions as described in the section on F_{HPDM} measurement (e.g., timepoints at the beginning of the simulation or during periods of depletion track-entrapment were excluded).
2. Select a random subset of a pre-defined number of tethers (i.e. a pre-defined polyvalency).
3. Calculate the energy of the system (composed of the HPDM body and the selected subset of tethers) at the current HPDM position and orientation using equation 2 from the main text.
4. Generate a random rigid-body transformation of the HPDM. Specifically, normally distributed random values of x-, y-, and z-displacements and rotation vector components ρ_x, ρ_y, and ρ_z were generated. The six random numbers were scaled such that they would have standard deviations of 1 nm, 1 nm, 0.1 nm, 1/R, 1/R, and 1/R respectively.
5. Calculate the energy of the system after application of the randomly-generated transformation in step 4.
6. Apply the transformation if one of the following two conditions is met: 1) If the change in energy following the transformation (ΔE) is negative, or 2) if $\exp(-\Delta E) < rand$, where $rand$ is a randomly generated number uniformly distributed on the interval 0 to 1.
7. Repeat steps 3-6 for a total of 10^5 iterations.

In this manner, thermal fluctuations of the HPDM body were simulated (**Fig. S19a-d**). This entire algorithm was repeated seven times each for polyvalency values ranging from 1 to 100. The energetic optimum for one iteration was calculated as the position and orientation that resulted in the lowest energy conformation (**Fig. S19e**). The deviation of the HPDM was calculated as the distance between the HPDM's center of mass at each timepoint and the HPDM's center of mass at the energetic optimum (**Fig. S19f**). The positions of the strand's anchor points for each tether were also used to calculate sample k_{on} values at each iteration. Two sample k_{on} plots – one for a pair of strands that is well within interaction range, and one for a pair that is just on the outer cusp of interaction range – are shown in **Fig. S19g** and **h** respectively. The k_{on} value at the energetic optimum ($k_{on,opt}$, black lines in **Fig. S19g-h**) was generally very similar to the average k_{on} value from across the entire iteration ($\langle k_{on} \rangle$ green dashed lines in **Fig. S19g-h**). This is exemplified by the ratio of the two metrics being clustered tightly on 1 (**Fig. S19i**). However, when the inter-strand distance at the genetic optimum (r_{opt}) is above a certain cutoff, the ratio diverges above one and becomes systematically biased (**Fig. S19i**). We ascribe this phenomenon to an increasing role of nonlinearities in the k_{on} vs. r curve at large r. Notably, the role of Brownian fluctuations decreases with increasing polyvalency, as can be viewed by an average deviation that decreases in a manner that is proportional to the inverse square-root of the polyvalency (**Fig. S19j**). Accordingly, the cutoff r_{opt} value increases with the polyvalency (**Fig. S19k**). The fraction of total association events in a RoloSim simulation that falls below this cutoff thus decreases with increasing polyvalency (**Fig. S19l**).

At a modest polyvalency of 20, only 5% of association events lie above this cutoff, while a polyvalency of 40 brings that measure down to 1% (**Fig. S19l**). Typical RoloSim simulations occur with polyvalencies in the range of ~80+, where this fraction is below 0.01%. Accordingly, only a very small fraction of association events have their on-rates affected by ignoring thermal fluctuations. We thus conclude that ignoring thermal fluctuations is a reasonable approximation with minimal effect on the prediction of HPDM dynamics.

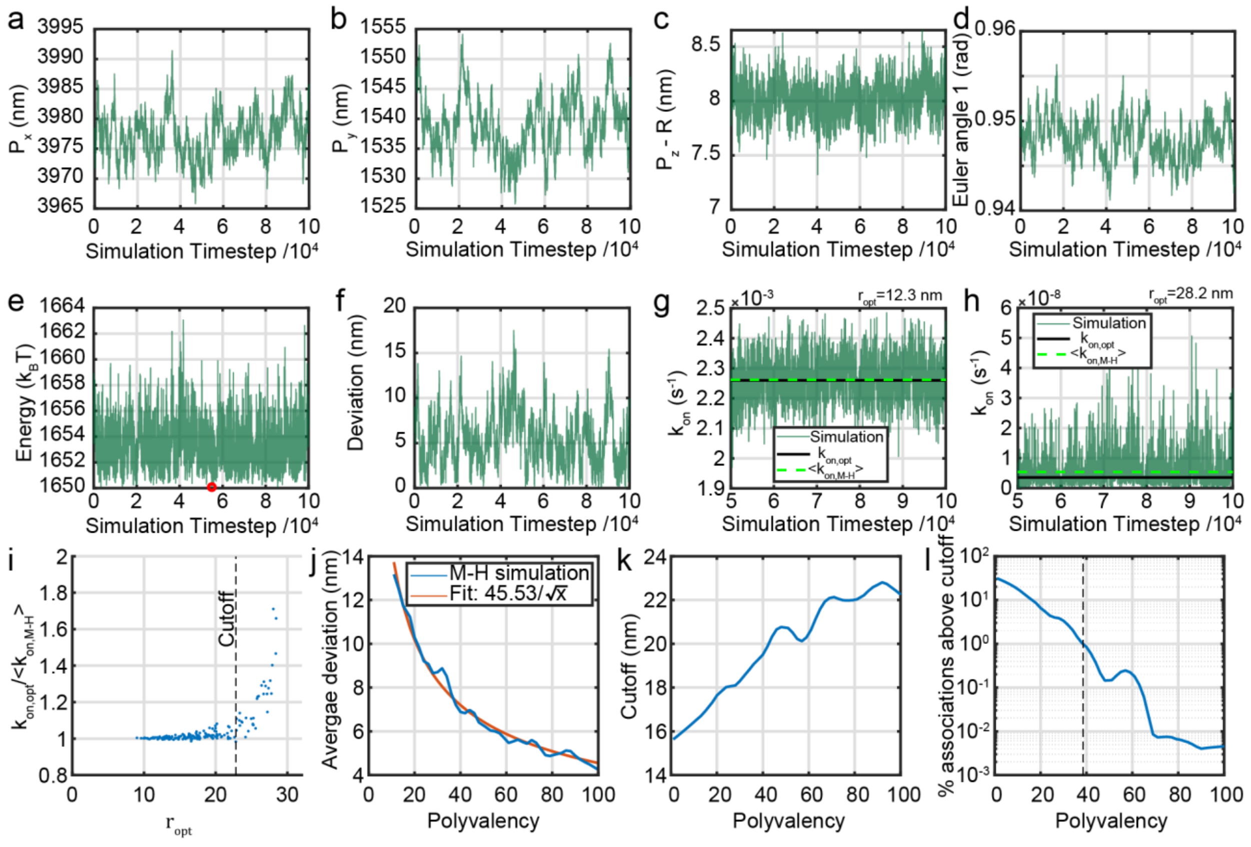


Figure S19: Consideration of thermal fluctuations with Metropolis-Hastings dynamics

a-h) Metrics from a representative Metropolis-Hastings simulation of thermal fluctuations with a polyvalency of 80 tethers. “Euler angle1” in **d** is the first of the three rotation angles from the Z-Y-Z Euler set used to store HPDM orientation information. The red circle in **e** denotes the energetic optimum. The solid black and dashed green lines in **g** and **h** represent $\langle k_{on} \rangle$ and $k_{on,opt}$, respectively. **i)** The ratio $k_{on,opt}/\langle k_{on} \rangle$ from DNA-RNA pairs in the simulation shown in **a-h**. The cutoff was calculated as the highest r_{opt} value that exhibited a ratio below 1, averaged across the seven iterations of the thermal fluctuation simulations. **j-l** metrics averaged across the seven iterations of the thermal fluctuation simulations for each polyvalency (blue). The orange curve in **j** shows a single-parameter fit to a 1/sqrt(polyvalency) relationship. The y-axis value in **l** was calculated by counting the number of association events that occurred with r greater than the cutoff (which is depicted in **k**) in the original RoloSim simulation that all iterations of the thermal fluctuation simulations were restarted from.

Supplemental Note 7: Calculation of HPDM contact zone width

The width of the HPDM contact zone, $w_{contact}$ was calculated as the maximum width of the contact zone. The contact zone is defined as the full area, on the RNA fuel surface, within which the closest point of the HPDM is within a distance such that $k_{on} \geq k_{on,0}/10$. In other words, all RNA fuel strands outside of the contact zone will, by definition, have an association rate $\leq k_{on,0}/10$. This cutoff distance, $r_{contact}$ can be calculated from equations (20) and (21)

$$10 = \exp\left(\frac{-\kappa_t^* \exp(\mathrm{K}^* \, {r_{contact}}^2)}{k_B T}\right) \tag{25}$$

Which re-arranges to

$$r_{contact} = \sqrt{\frac{\log\left(\frac{-\log(10)\, k_B T}{\kappa_t^*}\right)}{\mathrm{K}^*}} \tag{26}$$

Once $r_{contact}$ is calculated, $w_{contact}$ for spherical HPDMs can be calculated using an adaptation of a previously-derived[1] equation:

$$w_{contact} = D_{HPDM} \sin\left(\cos^{-1}\left(1 - \frac{r_{contact}2}{D_{HPDM}}\right)\right) \qquad (27).$$

For rod-shaped HPDMs, $w_{contact}$ is calculated with a similar equation that accounts for the discorectangular shape of the contact area:

$$w_{contact} = D_{rod} \sin\left(\cos^{-1}\left(1 - \frac{r_{contact}2}{D_{HPDM}}\right)\right) + L_{rod} \qquad (28).$$

References

1 Yehl, K. *et al.* High-speed DNA-based rolling motors powered by RNase H. *Nat. Nanotechnol.* **11**, 184, doi:10.1038/nnano.2015.259 (2015).

2 Vale, R. D. The Molecular Motor Toolbox for Intracellular Transport. *Cell* **112**, 467-480, doi:10.1016/S0092-8674(03)00111-9 (2003).

3 Blanchard, A. T. *et al.* Highly polyvalent DNA motors generate 100+ piconewtons of force via autochemophoresis. *Nano Lett.*, doi:10.1021/acs.nanolett.9b02311 (2019).

4 Clack, N. G., Salaita, K. & Groves, J. T. Electrostatic readout of DNA microarrays with charged microspheres. *Nat. Biotechnol.* **26**, 825-830, doi:10.1038/nbt1416 (2008).

5 Petrosyan, R. Improved approximations for some polymer extension models. *Rheol. Acta* **56**, 21-26, doi:10.1007/s00397-016-0977-9 (2017).

6 Whitley, K. D., Comstock, M. J. & Chemla, Y. R. Elasticity of the transition state for oligonucleotide hybridization. *Nucleic Acids Res.* **45**, 547-555, doi:10.1093/nar/gkw1173 (2017).

7 Becker, N. B., Rosa, A. & Everaers, R. The radial distribution function of worm-like chains. *The European Physical Journal E* **32**, 53-69, doi:10.1140/epje/i2010-10596-0 (2010).

8 Bazrafshan, A. *et al.* Tunable DNA Origami Motors Translocate Ballistically Over μm Distances at nm/s Speeds. *Angew. Chem. Int. Ed.* **59**, 9514-9521, doi:10.1002/anie.201916281 (2020).

9 Nitta, T. & Hess, H. Effect of Path Persistence Length of Molecular Shuttles on Two-stage Analyte Capture in Biosensors. *Cell. Mol. Bioeng.* **6**, 109-115, doi:10.1007/s12195-012-0262-7 (2013).

10 Schäffer, E., Nørrelykke, S. F. & Howard, J. Surface Forces and Drag Coefficients of Microspheres near a Plane Surface Measured with Optical Tweezers. *Langmuir* **23**, 3654-3665, doi:10.1021/la0622368 (2007).