Soft Matter

ARTICLE TYPE

Cite this: DOI: 00.0000/xxxxxxxxxx

Using Reactive Dissipative Particle Dynamics to Understand Local Shape Manipulation of Polymer Vesicles

Qinyu Zhu, Timothy R. Scott, and Douglas R. Tree*

Received Date
Accepted Date

DOI: 00.0000/xxxxxxxxxx

Biological cells have long been of interest to researchers due to their capacity to actively control their shape. Accordingly, there is significant interest in generating simplified synthetic protocells that can alter their shape based on an externally or internally generated stimulus. To date, most progress has been made towards controlling the *global* shape of a protocell, whereas less is known about generating a *local* shape change. Here, we seek to better understand the possible mechanisms for producing local morphological changes in a popular protocell system, the block copolymer vesicle. Accordingly, we have combined Dissipative Particle Dynamics (DPD) and the Split Reactive Brownian Dynamics algorithm (SRBD) to produce a simulation tool that is capable of modeling the dynamics of self-assembled polymer structures as they undergo chemical reactions. Using this *Reactive DPD or RDPD* method, we investigate local morphological change driven by either the microinjection of a stimulus or an enzymatically-produced stimulus. We find that sub-vesicle-scale morphological change can be induced by either a solvent stimulus that swells the vesicle membrane, or by a reactant stimulus that alters the chemistry of the block polymer in the membrane corona. Notably, the latter method results in a more persistent local deformation than the former, which we attribute to the slower diffusion of polymer chains relative to the solvent. We quantify this deformation and show that it can be modulated by altering the interaction parameter of the parts of the polymer chain that are affected by the stimulus.

Introduction

From a soft materials perspective, living cells are a technological wonder. Each cell is an active material with metabolic processes and an internal instruction set that gives it control over its shape and motion.[1–3] Collectively, cells communicate and engage in far-from-equilibrium assembly to form remarkable higher-order structures such as tissues, organs, and complete organisms.[4] Clearly, there would be numerous scholarly and practical benefits if one could create synthetic analogues to the cell, i.e. a synthetic protocell.

Given the extensive overlap between the fields of biology and soft matter,[5] researchers in both fields are vigorously pursuing the creation of protocells that perform functions similar to biological cells.[6] The polymer vesicle—a solvent-filled spherical bilayer composed of either natural lipids (liposome) or synthetic amphiphilic block polymers (polymersome)—is a popular platform for creating protocells.[7,8] However, even with recent attention, the most advanced polymer vesicles still lack basic functions that are essential to biological cells.[9]

Chemical Engineering Department, Brigham Young University, Provo, Utah, United States. E-mail: tree.doug@byu.edu

† Electronic Supplementary Information (ESI) available. See DOI: 10.1039/cXsm00000x/

In particular, we focus here on the challenge of actively controlling the shape and morphology of a polymer vesicle. We distinguish between two different scales of control that biological systems exert over shape. First, cells are able to control their *global* shape, by which we mean differences on the scale of the entire cell (e.g. the difference between rod-like, spherical, or cup-shaped). Even setting aside the drastic differences between the different types of specialized cells (e.g. nerve cells are incredibly complicated compared to the spheres and rods that are the most common cell shapes in animals[10]), individual cells can actively change their global morphology. For example, the familiar biconcave shape of red blood cells becomes cup-like when traveling in the narrowest blood vessels.[5] Because the shape of a cell is a balance between the forces exerted on the cell membrane from both intracellular components and the external environment,[1] it can be actively manipulated by osmotic forces that swell or shrink the cell, or by the introduction of membrane disrupting agents that cause the total dissolution of the cell leading to cell death.[11] It is interesting to note that the latter mechanisms can be utilized beneficially by the immune system[11] or hijacked to cause diseases such as Alzheimer's.[12]

Second, cells exert control over their *local* shape, through localized osmotic swelling and processes such as cell division, exocytosis, and endocytosis.[13] In contrast with global shape changes,

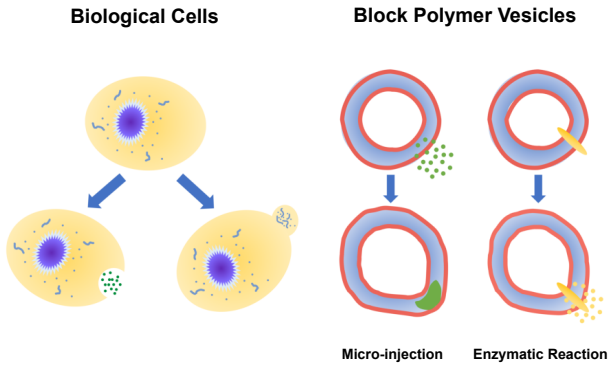

Fig. 1 (Left) Biological cells regulate their local shape in response to a variety of signals. (Right) Protocells, such as diblock copolymer vesicles, can also experience local morphological responses to microinjected or enzymatically-produced chemical signals.

these processes occur on the sub-cellular scale, affecting for example only one side of the cell or cell membrane. For example, fine-tuned local protrusions assist cellular motility when a neutrophil (i.e. white blood cell) navigates through red blood cells to chase a bacterium or travel directly to a wound site.[14,15] Such local deformation also happens within the cells, where mitochondria change the structure of their inner cristae membrane in response to metabolic needs, adjusting ATP production through shape control.[16]

Biological cells rely heavily on signaling and feedback processes to actively control both local and global shape.[2,5,14] Concomitantly, researchers have spent considerable effort on achieving stimuli-responsive shape manipulation of vesicles. Most of the reported progress has been on achieving stimuli-responsive manipulation of the *global* shape. To cite a few examples, Huo et al. utilized liquid crystalline block copolymers to change ellipsoidal assemblies to spheres or faceted spheres, programming them through temperature control.[17] Eloi et al. created fully reversible assembly and disassembly of spherical and rod-like micelles with redox reactive polymers.[18] Lagzi et al. used a pH oscillator to reversibly alternate between spherical micelles and vesicles assembled from fatty acids.[19] Finally, by changing solvent selectivity, concentration, or osmotic pressure, several authors have collapsed vesicles into cup-shaped stomatocytes, which have potential as micro-reactors, and when loaded with platinum can be used in catalytic hydrogen peroxide reactions to create motors.[20–23] While each of these cases use a different stimulus to achieve distinct morphological features, the general paradigm of using a stimuli-responsive amphiphilic polymer to induce global shape change is the same.

By contrast, less is known about the *local* shape control of synthetic vesicles, but several approaches have recently emerged. One paradigm focuses on manipulating pH gradients near a pH-responsive lipid membrane. A straightforward technique for doing so is to "microinject" an acid or base into a localized region near the membrane.[24–26] Alternatively, a pH gradient can be created by synthesizing or re-hydrating a vesicle in a basic solution, and then decreasing the environmental pH using an acid, thereby creating local pH gradients at the membrane.[27] Bitbol et al. used the former method to locally change the pH near a lipid vesicle, resulting in a reversible local membrane deformation.[24–26] Khalifat et al. also performed similar experiments and observed membrane invaginations that resemble those found in the cristae membrane of mitochondria mentioned earlier.[28,29] Though Bibtol et al. studied effects on the outer membrane and Khalifat et al. studied the inner, in both cases, the authors attributed the spontaneous change in curvature to a chemical modification and subsequent dynamic redistribution of the lipids in the respective monolayers.

Other paradigms for achieving local morphological control focus on the use of either a hybrid membrane or in-situ chemical reactions. Passos Gibson et al. used a hybrid membrane composed of both pH-inert block copolymer amphiphiles and pH-responsive lipids to trigger a variety of local conformational changes including tubular protrusions, membrane fluctuations, and internalized vesicles.[30] Using the latter approach, Miele et al. reported that the urease-catalyzed hydrolysis of urea inside the lumen of a vesicle led to spontaneous vesicle fission, again driven by pH gradients.[31] Other reaction-driven approaches are also possible, including methods that directly modify the molecular weight of the membrane polymer including polymerization[32] and chain scission.[33]

The paradigms that employ pH gradients and reaction-driven morphological manipulation are especially exciting developments, as they enable autonomous far-from-equilibrium processes that echo Nature's use of reaction-driven signalling pathways. As shown in Figure 1, in the present manuscript, we seek to give these approaches a more firm theoretical footing by adopting polymer vesicles as protocells that can mimic the cellular local morphological changes in responses to microinjected or enzymatically-produced chemical signals. In particular, we focus on modeling a polymersome composed of a coarse-grained representation of a diblock copolymer vesicle with solvophobic and solvophilic blocks.

Computer simulations have been used extensively in the past two decades to study the behavior of block polymer vesicles with both generic coarse-grained models,[34–36] and models whose parameters mimic a specific experimental system.[33,37,38] There have been many studies focused on the equilibrium (or metastable) morphology of block polymer micelles and vesicles and connecting molecular properties (e.g. block length, solvent/polymer interaction parameter) to the final self-assembled structure (e.g. micelle core radius).[36,39–46]

More interesting for our present purposes, several researchers have recently simulated kinetically-driven morphology changes in polymer vesicles.[33,47] Wright et al. performed coarse grained DPD simulations investigating enzyme-induced kinetic control over the self-assembly behavior of a polymer-peptide diblock in solution.[33] They studied the morphological evolution of spherical micelles following the addition of a protolytic enzyme that cleaves the peptide block, reducing the fraction of the hydrophilic block. In quite a different system, Gumus et al. studied the effects of "fast-quenching" or multi-stage quenching simulations on the morphology of micellar structures of bottlebrush polymers in solution.[47] They found a series of non-equilibrium nanostructures

that could potentially be realized in experiments by inducing a change in solvent quality in response to an external stimulus. Though different physical systems, both approaches found that kinetics could indeed induce morphological transformation of micellar structures. Additionally, both approaches relied on an instantaneously applied stimulus effected by a change of a simulation parameter (e.g. block length or interaction parameter) at a certain time in the simulation to induce morphological change.

By contrast, in order to study the far-from-equilibrium regulation of local structure inherent to living systems, one must properly account for the kinetics as part of the simulation. Indeed, one of the primary challenges to mimicking the above experimental systems is the need to incorporate chemical kinetics, as there are few established simulation methods that are able to capture both solution-phase amphiphilic polymer self-assembly and the type of stochastic chemical kinetics that are appropriate for these systems. Accordingly, we introduce a method below that combines dissipative particle dynamics (DPD) with a model for stochastic chemical kinetics that mirrors the Split Reactive Brownian Dynamics (SRBD) method by Donev et al.[48]. We call this combined method reactive DPD (RDPD). Following its description, we use RDPD to explore morphological change driven by (i) microinjection of a stimulus and (ii) a stimulus generated via chemical reactions at membrane embedded-catalysts. We subsequently describe and quantify the resulting morphological change and speculate on the implications for experiments. Notably, we believe this is the first theoretical or simulation approach that explicitly accounts for the effects of chemical kinetics on polymer vesicle assembly and morphology dynamics. Subsequently, this is the first simulation to demonstrate local morphological manipulation via these mechanisms.

Methods: Reactive Dissipative Particle Dynamics

Simulating reaction-driven local morphology change in a polymer vesicle is not an easy task. The first key challenge is to properly model block polymer self-assembly and dynamics. Models of block polymer self-assembly are notoriously subtle, and a vast array of approaches for simulating polymer vesicles spanning length and times scale from atoms[49–52] to continuous fields[53–55] have been used. The timescale of atomistic simulations limit their applicability here, and though there are emerging field-based approaches of interest,[55–58] field theory-based simulations are typically challenging due to the need to capture both dynamics and the strong composition fluctuations present in micellar systems.[53] A mesoscopic particle model such as DPD is a practical alternative as it splits the difference between molecular dynamics (MD) and field-based methods, allowing for relatively fast relaxation times compared to MD and a facile incorporation of dynamics and fluctuations compared to field-based approaches.[41,59]

The second key challenge is to model chemical kinetics appropriately. Models of chemical kinetics can be categorized as either (i) nonspatial or spatial and (ii) deterministic or stochastic.[60–63] Nonspatial models are valid for a chemically homogeneous (i.e. well-mixed) system, and deterministic models are valid when the concentration is large enough for concentration fluctuations to have a negligible effect on the kinetics.[64] Neither is the case for reaction-driven morphology change in polymer vesicles, where inhomogeneities are inherent, and the small concentration of reactants and catalysts can lead to important stochastic effects. One pragmatic approach is to use a particle-based chemical reaction model. This approach offers spatially resolved, non-deterministic chemical kinetics based on collision theory,[65] and is compatible with a particle-based model of polymer self-assembly such as DPD.

Accordingly, we report here a DPD method for simulating local morphology changes in polymer vesicles that incorporates a collision-based model of chemical kinetics that we term Reactive Dissipative Particle Dynamics or *RDPD*. A classical DPD algorithm is used to simulate the self-assembly thermodynamics and transport behavior. An efficient, event-driven algorithm called Split Reactive Brownian Dynamics (SRBD) is used to model the stochastic chemical reactions.[48] While there are multiple open source software packages for performing classical DPD simulation (e.g. HOOMD-Blue[66,67]), we are not aware of any that have the capacity to simultaneously perform stochastic reaction kinetics. As such, we describe in detail our approach below, which we have subsequently implemented in a custom developed GPU-accelerated Python code.

Dissipative Particle Dynamics

Our DPD model is based on the classic work by Groot and Warren,[68] which models N_t particles driven by Newton's equations of motion[68]

$$\frac{d\boldsymbol{r}_i}{dt} = \boldsymbol{v}_i \tag{1}$$

$$\frac{d\boldsymbol{v}_i}{dt} = \boldsymbol{f}_i \tag{2}$$

where $\boldsymbol{r}_i$ is the position, $\boldsymbol{v}_i$ is the velocity, and $\boldsymbol{f}_i$ is the total force on the i^{th} particle. We assume the mass of each particle to be the same. Eqs. 1 and 2 and those below are expressed in dimensionless units with the cut-off distance r_c scaling length scales, the Boltzmann factor k_BT scaling energy scales, and the bead mass m scaling the mass. The total force is the sum of four terms,

$$\boldsymbol{f}_i = \sum_{j\neq i}\left(\boldsymbol{f}_{ij}^C + \boldsymbol{f}_{ij}^D + \boldsymbol{f}_{ij}^R + \boldsymbol{f}_{ij}^S\right) \tag{3}$$

a conservative force, a dissipative force, a random force[69], and a spring force[68] respectively. The forces are given by

$$\boldsymbol{f}_{ij}^C = a_{ij}\omega^C(\boldsymbol{r}_{ij})\boldsymbol{e}_{ij} \tag{4}$$

$$\boldsymbol{f}_{ij}^D = -\gamma\omega^D(\boldsymbol{r}_{ij})(\boldsymbol{e}_{ij}\cdot\boldsymbol{v}_{ij})\boldsymbol{e}_{ij} \tag{5}$$

$$\boldsymbol{f}_{ij}^R = \sigma\omega^R(\boldsymbol{r}_{ij})\zeta_{ij}\Delta t^{-0.5}\boldsymbol{e}_{ij} \tag{6}$$

$$\boldsymbol{f}_{ij}^S = C_{\text{spring}}(|\boldsymbol{r}_{ij}|)\boldsymbol{e}_{ij} \tag{7}$$

where $\boldsymbol{r}_{ij}$ and $\boldsymbol{e}_{ij}$ are the position vector and unit vector between i and j, and $\boldsymbol{v}_{ij}$ is the relative velocity between i and j. The coefficients a_{ij} are binary repulsive interaction parameters between particles of type i and j. σ and γ are coefficients of the dissipative

Table 1 Value of the model parameters in the specified DPD model. The number of beads correspond to a simulation box of size $L_x = L_y = L_z = 100r_c$.

parameter	value	description
Δt	0.05	time step
σ	3.0	coefficient of random force
γ	4.5	coefficient of dissipative force
C_{spring}	100.0	spring constant
N_t	3×10^6	total number of beads
x_p	0.08376	mole fraction of polymer beads
f_a	0.2	block fraction of A
N_b	60	diblock chain length
ρ	3	bead number density

and random forces and are related via the fluctuation–dissipation theorem

$$\sigma^2 = 2\gamma k_B T \tag{8}$$

where k_B is Boltzmann's constant, and T is the absolute temperature. ω^C, ω^D, and ω^R are weight functions defined as,

$$\omega^C(\boldsymbol{r}_{ij}) = \omega^R(\boldsymbol{r}_{ij}) = (\omega^D(\boldsymbol{r}_{ij}))^2 = \begin{cases} 1 - |\boldsymbol{r}_{ij}|, & |\boldsymbol{r}_{ij}| < r_c, \\ 0, & |\boldsymbol{r}_{ij}| \geq r_c. \end{cases} \tag{9}$$

ζ_{ij} is a Gaussian-distributed random number with zero mean and unit variance and is chosen independently for each interacting pair at each time step. Δt is the time step and the parameters C_{spring} is the spring constant between bonded beads along the polymer chain. Note that the equilibrium length of the bonds along a polymer chain is determined by a balance of the spring force in Eq. 7 and the repulsive interactions between bonded neighbors from Eq. 4. A list of the values of the parameters used for our simulations is given in Table 1.

Our block polymer solution consists of N_s solvent beads and N_p polymer beads, the latter consisting of N_c chains of an A_mB_n diblock copolymer with block lengths m and n respectively. In our convention, A represents a solvophilic block and B represents a solvophobic block. The chain length of the diblock is given by $N_b = m + n$, and the block fraction of the solvophilic block is defined as $f_A = \frac{m}{N_b}$. Additionally, we define the mole fraction of the total number of monomers in polymer chains as $x_p = N_p/N_t$ and the total solvent mole fraction as $x_s = N_s/N_t$. Typical simulation parameters associated with the solvent and polymer model are again specified in Table 1 unless otherwise noted in the text below.

In addition to solvent (S_A) beads, and beads for the AB diblock (A, B), we introduce three additional bead types. Enzyme (E) beads are separate particles which are completely compatible with A beads and S_A beads, but act as reaction catalysts. We introduce a second solvent type (S_B) which are equivalent to B beads. We also have a third solvent type ($S_{A'}$) which act as the external stimulus to change polymer properties. We set the repulsive parameter between beads of the same type to $a_{ii} = 25$. The repulsive parameter between compatible beads is also set to $a_{ij} = 25$, and the parameter between incompatible beads is set to $a_{ij} = 160$, unless otherwise noted. A summary of the matrix of

Table 2 Summary of the DPD interaction parameters between bead types: A (solvophilic block), B (solvophobic block), S_A (solvent), $S_{A'}$ (reactive solvent as external stimulus), S_B (B-compatible solvent), and E (enzyme). Dashes in the table indicate that the two species never appear in the same simulation, and therefore don't interact.

	A	B	S_A	$S_{A'}$	S_B	E
A	25	160	25	25	160	25
B	160	25	160	160	25	160
S_A	25	160	25	25	160	25
$S_{A'}$	25	160	25	25	–	–
S_B	160	25	160	–	25	160
E	25	160	25	–	160	25

interaction parameters between the different beads is given in Table 2. The cutoff radius is the same for all simulation particles, r_c. Following Groot and Warren, we set the number density of beads in the simulation box to $\rho = 3$. At this number density, an effective Flory-Huggins interaction parameter can be estimated from the DPD interaction parameter,

$$\chi = (0.286 \pm 0.002)(a_{ij} - a_{ii}) \tag{10}$$

Using Eq. 10, a typical interaction parameter between incompatible beads is $\chi \approx 38.6$. Though useful, Eq. 10 should be viewed as an estimate, given that there is subtlety in correctly estimating effective chi parameters from simulations.[70]

We numerically integrate Eqs. 1 and 2 using a modified Velocity-Verlet algorithm[71]

$$\boldsymbol{r}_i(t + \Delta t) = \boldsymbol{r}_i(t) + \Delta t \boldsymbol{v}_i(t) + \frac{(\Delta t)^2}{2} \boldsymbol{f}_i(t) \tag{11}$$

$$\tilde{\boldsymbol{v}}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \lambda \Delta t \boldsymbol{f}_i(t) \tag{12}$$

$$\boldsymbol{f}_i(t + \Delta t) = \boldsymbol{f}_i(\boldsymbol{r}(t + \Delta t), \tilde{\boldsymbol{v}}(t + \Delta t)) \tag{13}$$

$$\boldsymbol{v}_i(t + \Delta t) = \boldsymbol{v}_i(t) + \frac{\Delta t}{2} [\boldsymbol{f}_i(t) + \boldsymbol{f}_i(t + \Delta t)] \tag{14}$$

with $\Delta t = 0.05$ and $\lambda = 0.65$. All simulations were carried out in a cubic simulation box of $100r_c \times 100r_c \times 100r_c$ with period boundary conditions applied in all directions, unless otherwise stated. The 3D figures shown below were generated with VMD.[72]

We use a cell list to efficiently calculate the pairwise interactions between particles. We chose a cell list, rather than another technique such as a Verlet list, because it is compatible with the SRBD technique described below. In our approach, we use a traditional algorithm that divides the simulation domain into uniform cells of size r_c and stores particle indices in a linked list.[71] One must pay particular attention to the algorithm for creating and populating this linked list in order to achieve efficient parallelism and to preserve $O(N_t)$ scaling when running on a GPU. We provide details for this procedure in the Supplementary Information.

Stochastic Chemical Kinetics

To the classical DPD simulation described above, we add a model of spatially heterogeneous, stochastic reaction kinetics. There are a variety of alogrithms for simulating particle-based stochastic re-

action kinetics including Meredys,[73] Green's function reaction dynamics,[74] hybrid smoothed dissipative particle dynamics (SDPD), the spatial stochastic simulation algorithm (SSA),[75] first passage kinetic Monte Carlo (FPKMC),[76,77] and split reactive Brownian dynamics (SRBD).[48] The SRBD[48] algorithm stands out amongst these methods for a number of reasons, including its compatibility with DPD and its ability to simulate reversible reactions.

All of the above methods are based on collision theory, which postulates that reactive particles need to approach one another within some distance in order for a reaction to occur.[65] Doi added an important model onto collision theory, making a reaction a probabilistic event (a Poisson process) that may or may not occur even when particles are inside the reaction radius.[78] Consequently, simulations based on Doi's model makes it easier to achieve detailed balance (reversibility), which is rather important for real systems. Doi's model also provides a means for calculating an expected reaction time for a given reaction rate parameter.

It is easy to imagine a naive algorithm for simulating chemical kinetics based on collision theory and Doi's model. Namely, one searches the simulation box for pairs of beads within some reaction radius, and a decision is made whether or not to execute a reaction event based on the reaction rate parameter and a draw of a random time increment. (This time increment needs to be less than the diffusion time step Δt, or the reaction will not happen.) While certainly correct, this method is computationally expensive, because one needs to repeatedly search for reactive pairs throughout the entire simulation box following each reaction.

The SRBD algorithm[48] solves this problem by dividing the system into reactive subvolumes where each reaction is processed based on the Doi model.[78] This approach makes SRBD similar to solving a local reaction–diffusion master equation in each subvolume.[63,79] Notably, such a cell-based method is also compatible with efficient implementations of DPD. On top of the spatial discretization, SRBD implements an event-driven method for processing the reactions in each cell, increasing the efficiency of the algorithm. In our implementation of the method, we have replaced the uncorrelated Brownian particles discussed in Ref. 48 with DPD particles, making it more suitable for simulating a liquid system.

To illustrate the method more specifically, we consider here the SRBD method with the binary reaction,

$$\mathrm{A}+\mathrm{B} \underset{k_r}{\overset{k_f}{\rightleftharpoons}} \mathrm{C}+\mathrm{D} \tag{15}$$

where k_f is the forward rate constant and k_r is the rate constant of the reverse reaction. Eq. 15 is quite general, as A and B can be assumed to be the same species, and simulations with unimolecular reactions are a trivial simplification. Note also that the original SRBD algorithm for Brownian dynamics allows the total number of particles to be increased or decreased during reactions. This is undesirable for a DPD system, as it changes the system density, so we do not consider such reactions here.

SRBD is implemented as shown in the schematic in Figure 2. First, the simulation box is divided into reactive cells of size r_c (see Figure 2a), and a reaction time is calculated for each cell based on its current condition. This reaction time is the scheduled time for the next reaction that will happen in the cell, and is calculated using a propensity function based on the local concentration of reactants in the cell (see Figure 2b). The propensity function for the forward reaction in Eq. 15 in cell i is given by,

$$\alpha_{if} = \frac{k_f}{2}\left[n_\mathrm{A}\tilde{n}_\mathrm{B} + \tilde{n}_\mathrm{A} n_\mathrm{B}\right] \tag{16}$$

where n_A and n_B are respectively the number of particles of A and B in cell i, and $\tilde{n}_\mathrm{A}$ and $\tilde{n}_\mathrm{B}$ are respectively the number of particles of A and B in both cell i and *all of the cells immediately neighboring cell i*. Note that the propensity function is a sum of two "ordered half-reactions", $\mathrm{A}+\mathrm{B}\to\ldots$ and $\mathrm{B}+\mathrm{A}\to\ldots$, thus explaining its particular form. Analogously, the propensity function for the reverse reaction is given by,

$$\alpha_{ir} = \frac{k_r}{2}\left[n_\mathrm{C}\tilde{n}_\mathrm{D} + \tilde{n}_\mathrm{C} n_\mathrm{D}\right] \tag{17}$$

where n_C and n_D are respectively the number of particles of C and D in cell i, and $\tilde{n}_\mathrm{C}$ and $\tilde{n}_\mathrm{D}$ are respectively the number of particles of C and D in both cell i and its neighbors. The total propensity function for cell i is given by the sum

$$\alpha_i = \sum_j \alpha_{ij} \tag{18}$$

where $j \in [f,r]$.

As is characteristic of a Poisson process, the reaction time δt_i for cell i is obtained by a draw from an exponential distribution with a rate parameter α_i.[80] Once the δt_i are calculated for each cell, they are arranged into an event queue in ascending order, as illustrated in Figure 2c. The cell with the smallest δt_i (i.e. the top of the event queue) is chosen first, and either the forward or reverse reaction is chosen according to the SSA algorithm.[64,80,81] In the latter algorithm, a uniformly distributed random number between 0 and 1, ξ, is generated and the forward reaction is selected if $\xi < \alpha_{if}/\alpha_i$. Otherwise, the reverse reaction is selected.

Once a reaction is chosen, beads of the appropriate reactive species are randomly chosen inside cell i. If the interbead distance r_{ij} between the reactive species is less than the reaction radius r_c, the particles undergo the selected reaction and their identities are converted to those of the products. If the chosen particles are too far apart, then no reaction takes place. Regardless of whether the reaction occurs or not, the global time t is advanced to $t+\delta t_i$, where δt_i is the reaction time of the first cell in the event queue. We then proceed to deal with the next shortest reaction time, as suggested at in Figure 2c. The reactions are processed by iterating this procedure until the global time reaches $t+\Delta t$ (the length of one time step in the velocity Verlet algorithm) or until the event queue is empty.

The simple "one event per cell" procedure is complicated by the fact that a reaction changes the state of a cell and its neighbors, meaning the next event that will take place in the reacting cell and its neighbors has changed. Thus, when a reaction takes place, a new reaction time $\delta t_i'$ is calculated for both the reacting cell and its neighboring cells by again evaluating the propensity function using Equations 16–18 and generating a random number from

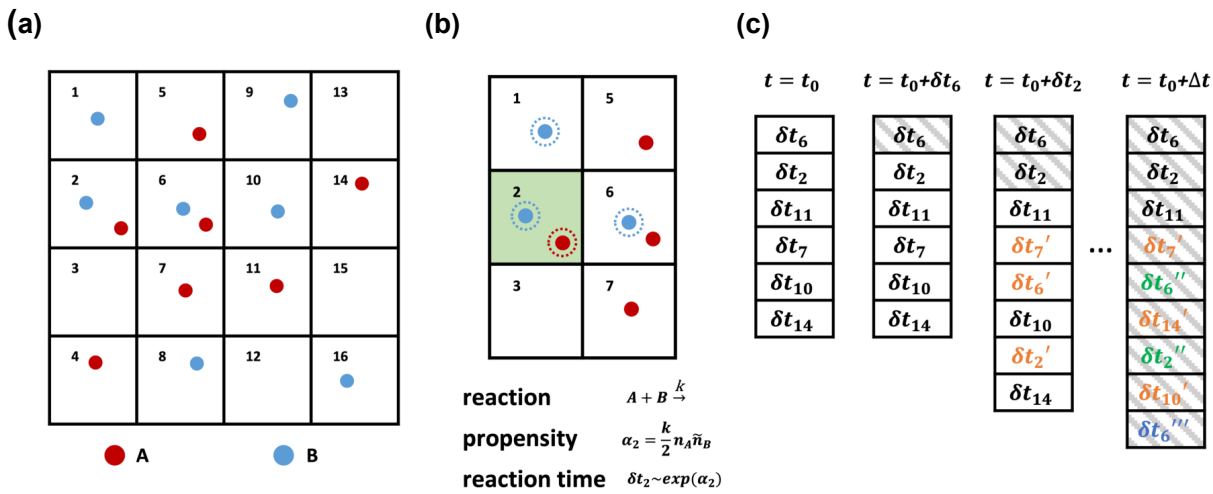

Fig. 2 A schematic diagram of the SRBD algorithm: (a) The simulation box is divided into small reactive cells. (b) Reaction times are calculated for each cell based on the local number of reactive particles in the cell and its neighbors. For example, the propensity function in cell 2 is calculated based on the particles with dashed outlines and is used to obtain δt_2. (c) An event queue containting the reaction times is sorted from the shortest to the longest times. After each event, the time is updated to $t_0 + \delta t_i$ where δt_i is the reaction time of the last event. Additionally, After each successful reaction takes place, the queue is updated with new reaction times $\delta t_i'$ for cells altered by the reaction, where the number of primes indicates the number of updates.

an exponential distribution rate α_i. Note that the new reaction times $\delta t_i'$ may update the reaction time for a cell that has not yet processed an event, or it may re-insert a cell into the queue that has previously experienced an event (whether or not the event resulted in a successful reaction). The event queue is then re-sorted with the new reaction times, as visualized in Figure 2c.

We implemented the above version of SRBD in our custom GPU-accelerated DPD code described above. In the original algorithm, Donev et al. used a second-order Strang splitting method to integrate their Brownian dynamics code.[48] That is, they executed a diffusive half-step, then processed the reactions over a full time step Δt, and then completed another half-step. For our DPD algorithm, we found that we required a time step size at least one order of magnitude smaller than Donev et al., making the second-order scheme unnecessary for accuracy. Accordingly, in our approach, we first diffuse for a full time step Δt, and then react over Δt. This process results in fewer evaluations of the interbead forces, increasing our efficiency relative to the second order method.

Additionally, analogous to our cell list for evaluating pairwise interactions between DPD particles, we parallelized the reactive cell calculations of δt_i on the GPU. This does not represent a full parallelization of SRBD, because the event queue is still processed sequentially. As noted by Donev et al., parallelizing the event queue remains an open methodological challenge.[48]

Results and Discussion: Validation of RDPD

Vesicle Self-Assembly via DPD

While numerous DPD simulations of polymer vesicles have been achieved,[34,37,82] and much has been said about the equilibrium morphology of micellar polymer solutions,[35,83] the literature does not provide simple, universal relationships between model parameters and vesicle morphology. Indeed such an endeavor is complicated by the formation of neighboring metastable morphologies and defective states that have similar free energies. As such, achieving a polymer vesicle of a desired size at a given chain length and block fraction remains somewhat of an art form.

Additionally, in the present work, we require a polymer vesicle that is large enough to sustain meaningful gradients in the simulation box for times that are long enough to drive local morphological change. As we show below, such a simulation requires a large simulation box, requiring an efficient, parallelized DPD simulation. Here we show some basic results demonstrating the formation of a large polymer vesicle in our DPD model. These results provide the primary validation that our model is (i) fast enough to reach these large scales and (ii) capable of capturing the relevant polymer phase behavior.

To demonstrate that our code is capable of running large scale jobs, Figure 3 compares the simulation time of our code with that of a popular MD package, HOOMD-Blue.[66] These simulations were performed in a diblock copolymer solution with $x_p = 0.2$ and $x_s = 0.8$, as the box size is increased from $L = L_x = L_y = L_z = 30 r_c$ to $L = 80 r_c$ with the other DPD parameters given in Table 1. The computational time scales linearly with N_t, demonstrating that the cell list appropriately accounts for pairwise interactions.

Our DPD code is between 1.58 and 1.75 times slower than the HOOMD-Blue code under the same conditions. This is not an insignificant slowdown, but the result is in fact remarkable given our code platform and development path. HOOMD-Blue is a highly optimized CUDA/C++ code developed and maintained over several years with many users. Our code is developed in native Python and is optimized with the Numba just-in-time (JIT) compiler with the accompanying CUDA Toolkit that enables GPU-accelerated calculations.[84] Numba is an open source compiler for

Python numerical calculations that generates optimized machine code from pure Python, and its CUDA Toolkit provides the ability to develop GPU-accelerated code with speeds that are competitive with C++. With these tools, our code development process took a single graduate student less than three months.

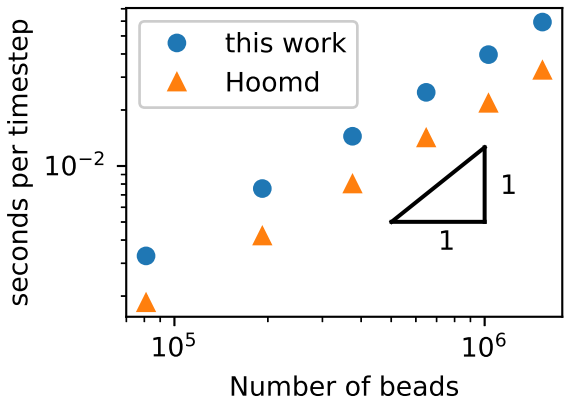

Fig. 3 Simulation time of a GPU-accelerated DPD model (without chemical reactions) as a function of the number of beads for the RDPD code and HOOMD-Blue.

We now turn our attention to equilibrating a large polymer vesicle. Since our simulations are carried out in a large simulation box, it is computationally costly to relax and equilibrate a vesicle from random initial conditions. Additionally, because of nearby metastable states the final morphology of a given simulation is sensitive to the initial condition and is therefore kinetically determined, making it even harder to obtain our desired vesicle structure. Accordingly, we used an external field to guide the formation of vesicle structure.[85] The guiding field is in the shape of the vesicle morphology and consists of ghost particles at fixed positions that only interact with the solvophilic blocks through a Gaussian potential,

$$U_{\text{Guass}} = -ae^{-b|\boldsymbol{r}_{ij}|^2} \tag{19}$$

for $|\boldsymbol{r}_{ij}| < r_c$, where $a = 5$ and $b = 2$. During simulations with a guiding field, Eq. 19 is differentiated and used as an additional force in Eq. 3.

Using the guiding field, we equilibrated the vesicle in Figure 4 using the following procedure. We first performed a DPD simulation in a $L = 60r_c$ box with the guiding field to get a roughly spherical vesicle morphology. We then expanded the box size to $L = 100r_c$ using the previous morphology as an initial condition (filling the rest of the space with solvent particles), and ran another DPD simulation without the guiding field for 10^6 steps to ensure that the system reached equilibrium. As the system equilibrated, some polymer chains were ejected from the vesicle. At the conclusion of the simulation, these chains were converted to solvent particles, and the final system was again relaxed for another 10^6 steps. These stray chains were converted into solvent to ensure that no extra chains in the solution could interfere with the vesicle and to keep the system density fixed at $\rho = 3$. Further details of the relaxation time of our system are provided in the Supporting Information. The final polymer vesicle was obtained using $N_c = 60$, $f_A = 0.2$ and $x_p = 0.08376$. The average size of the vesicle is 56.8 r_c in diameter, and the lumen is approximately 17.2 r_c in diameter.

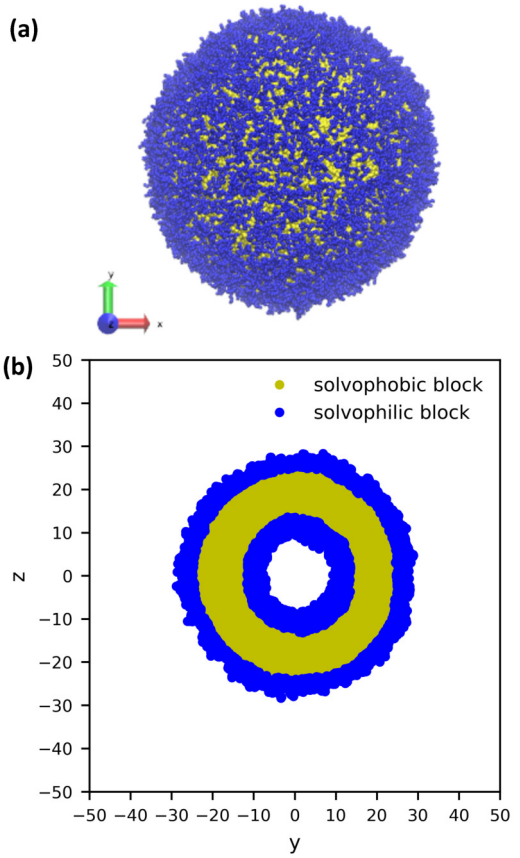

Fig. 4 (a) A 3D image and (b) cross-sectional view of a large, equilibrated vesicle morphology with diameter $56.8r_c$. The solvophilic A-block is shown in blue and the solvophobic B-block is shown in yellow. For clarity, the solvent particles are not shown.

Stochastic Reaction Diffusion Model

As discussed above, the SRBD algorithm enables the simulation of chemical reaction kinetics in DPD. To validate the reaction kinetics, we performed simulations of the binary chemical reaction of the conversion of solvent A into solvent B mediated by catalyst particles E,

$$\mathrm{S_A} + \mathrm{E} \rightleftharpoons \mathrm{S_B} + \mathrm{E}. \tag{20}$$

We performed two different types of simulations of Eq. 20. In the first, we set $k_f = 0.1$ and $k_r = 0$, making the reaction irreversible to study the kinetics of the complete conversion of S_A to S_B. In the second, we set $k_f = 0.1$ and $k_r = 0.05$ to study both the kinetics and the equilibrium reached as $t \to \infty$. In both cases, the reactions were performed with single DPD solvent beads randomly placed in a box of size $L = 15r_c$ with initial mole fraction of $x_{S_\mathrm{A}} = 0.993$, $x_{S_\mathrm{B}} = 0$, and $x_\mathrm{E} = 0.007$. Additionally, we ran 20 replicates of each simulation in order to obtain smooth averages for comparison to non-stochastic rate theory.

The results from the irreversible reaction are shown in Fig. 5a. To interpret the RDPD results, it is instructive to compare to a model of a non-stochastic first order rate equation,

$$\frac{dx_{S_A}}{dt} = -kx_{S_A}x_E = -k'x_{S_A} \tag{21}$$

where we lump the reaction rate k and the catalyst mole fraction ϕ_E into k' since they remain constant throughout the simulation. The analytical solution for ϕ_{S_A} is given as,

$$x_{S_A}(t) = x_{S_A}(0)\exp(-k't). \tag{22}$$

As expected, $\langle x_{S_A}\rangle$ obtained from the RDPD simulation decays exponentially in time starting from the initial mole fraction of $x_{S_A}(0) = 0.993$ as predicted by Eq. 22. However, a non-linear fit to the the rate of decay gives a value of $k' = 0.0029$, which does not match the microscopic reaction rate of $k_f x_E = 7\times10^{-4}$. This apparent contradiction is resolved by more carefully mapping the microscopic reaction rate onto the effective macroscopic rate k'. In the SRBD algorithm, the macroscopic reaction rate is determined by the propensity function, i.e. Eq. 16 and Eq. 17. The propensity function depends on the number of particles within the reactive distance, r_c, and therefore the effective macroscopic reaction rate is given by,

$$k' = \frac{4}{3}\pi {r_c}^3 k_f x_E \tag{23}$$

where the factor $4/3\pi r_c^3$ is the volume of a sphere with a reaction radius of r_c. Using Eq. 23 gives $k' = 0.00293$, which is in excellent agreement well with the value of 0.0029 obtained from the fit. Fig. 5a presents the averaged simulation data obtained from RDPD alongside $x_{S_A}(t)$ from Eq. 22 using the predicted value of $k' = 0.00293$. As shown in the figure, the data is in good agreement with the theoretical prediction.

Turning our attention to the case of the reversible reaction, Fig. 5b shows the mole fraction of S_A as a function of time obtained from RDPD, again averaged over 20 replicates. At long times we expect this system to reach an equilibrium value of $x_{S_A} = 0.333$. We observe that the value of $\langle x_{S_A}\rangle$ fluctuates around an equilibrium mole fraction of 0.33 after $t = 1500$. This is of course in excellent agreement with the prediction, suggesting that we are reaching the equilibrium value determined by the forward and reverse rate constants as expected.

It is also informative to compare the performance of the various algorithms discussed in the previous section for simulating chemical kinetics. Fig. 6 shows the run-time of classical DPD (our code), the "naive" algorithm, and the SRBD algorithm in both serial and parallel (i.e. GPU) schemes. These simulations were performed using the same parameters as those above, namely for monomers in a box of size $L = 15r_c$, though here each simulation is only run for 2×10^4 time step. The particles were randomly placed in the simulation box, and were subjected to the catalytic conversion reaction in Eq. 20.

We draw several conclusions from the results in Fig. 6. First, as expected, the parallel versions are considerably faster than the serial versions, even when executing chemical reactions. In fact, the

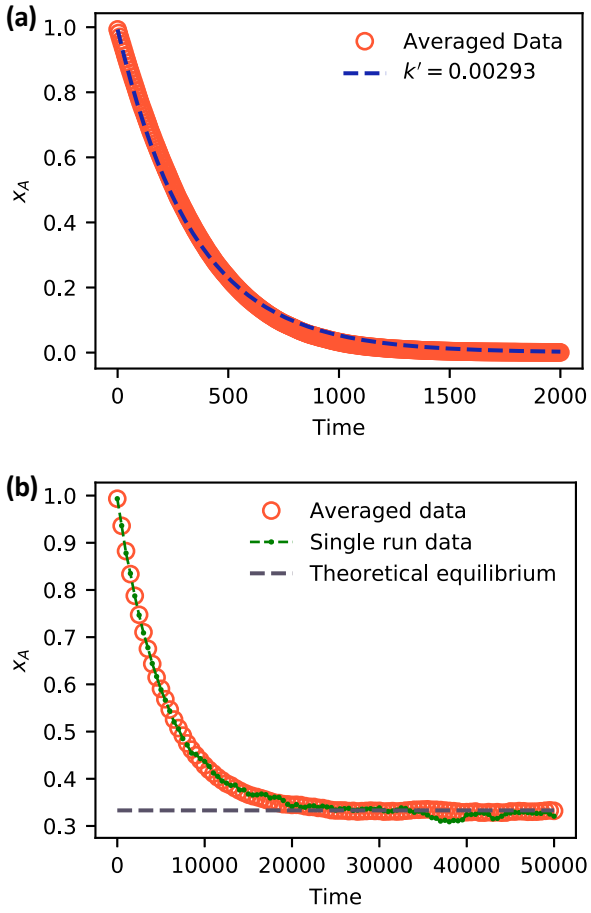

Fig. 5 Simulation data of x_A as a function of time for (a) an irreversible catalytic conversion, and (b) a reversible catalytic conversion. The average of 20 replicates are plotted for both simulations. Additionally, the blue dashed line in panel (a) shows Eq. 22 using the predicted value $k' = 0.00293$, and the gray dashed line in panel (b) shows the theoretical equilibrium mole fraction of 0.333. The green points in panel (b) show a single simulation, illustrating the stochastic nature of the reaction.

SRBD algorithm is over 22× faster when using the GPU. Second, including chemical reaction kinetics slows down the simulation, which is also expected. When executing the serial version code, the *SRBD* code runs 32.2% slower than the basic DPD code, and in parallel the *SRBD* code is only 12.6% slower. Finally, we see that SRBD brings modest performance gains over the naive algorithm. SRBD is 20% faster than the naive algorithm in serial and 12% faster in parallel. Again we note that we did not parallelize the SRBD event queue, but this could bring additional performance gains.

In addition to characterizing the rate of chemical reactions, it is useful to know the species diffusivity in chemically reacting systems. Accordingly, we performed a calculation of the tracer diffusion by calculating the mean squared displacement (MSD) of DPD solvent beads as a function of time and fitting the data points to

$$MSD = 6D_{S_A}t \tag{24}$$

where D_{S_A} is the tracer diffusion of the solvent S_A. In our diffusiv-

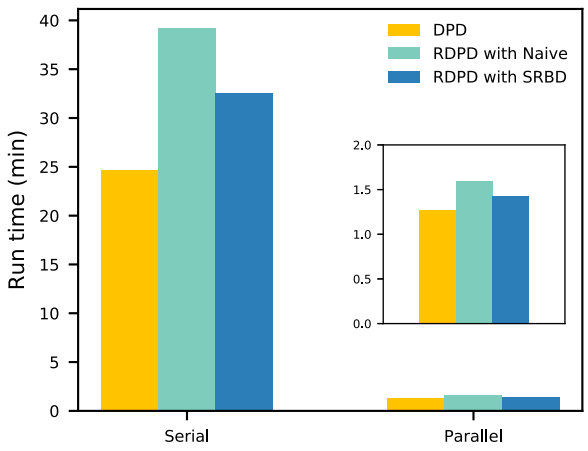

Fig. 6 Comparison of the run time of three algorithms in both serial and parallel: bare DPD, RDPD using the "naive" algorithm, and RDPD using SRBD. The inset shows the parallel results on a re-scaled axis. Numerical values of these run times are provided in the Supporting Information.

ity calculations, we used a system composed of initially randomly placed S_A beads in a box size of $L = 25r_c$ that was first relaxed for 10^4 timesteps. Following the relaxation period, the MSD of all particles from $t = 0$ to $t = 35$ is shown in Fig. 7. As expected, the MSD is linear and a fit gives $D_{S_A} = 0.2296$ in simulation units.

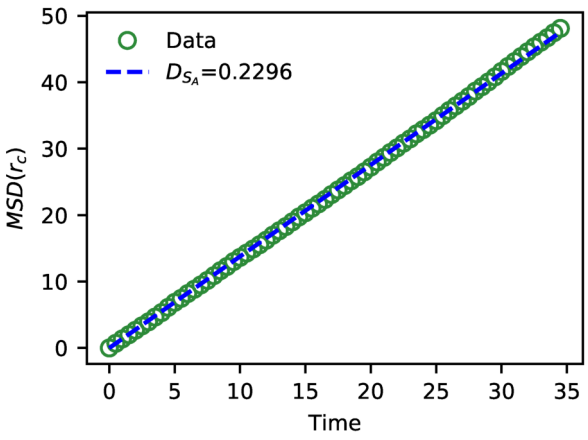

Fig. 7 The MSD of solvent beads as a function of time. The green circles are the MSD obtained from the simulation, and the blue dashed line is a fit to Eq. 24 with $D_{S_A} = 0.2296$.

The value of D_{S_A} obtained above is a simple calculation of the bare tracer diffusion and does not include interactions between solvents that must be accounted for in the mutual diffusivity when considering a mixture that contains beads with disparate values of a_{ij}. Additionally, in their paper on SRBD,[48] Donev et al. reported a curious enhancement of diffusion during reversible reactions where the number of particles are not conserved, such as

$$A + B \rightleftharpoons C. \tag{25}$$

Because of our need to keep the particle density constant in DPD, we did not directly test this mechanism. We did generate estimates of the diffusivity in simulations containing both irreversible and reversible reactions where the particle number is conserved, but the results were inconclusive. A more careful study of any coupling between these effects would be welcome future work.

In addition to the solvent diffusivity, we also calculated the diffusivity of polymer chains inside the vesicle membrane to be $D_{\text{polymer}} = 0.0073$. As expected, chains move significantly slower than solvent, especially when co-located in the vesicle membrane. Additional details related to these latter results are given in the Supplementary Information.

Results and Discussion: Stimulus-Responsive Local Morphology

We highlighted above two generic experimental paradigms for manipulating the local shape of vesicles: microinjection of a solvent and local chemical reactions. These paradigms are not mutually exclusive, since a local chemical reaction can proceed from either a microinjected reactant or from a localized catalyst, such as an embedded enzyme.

In this section, we explore two mechanisms using both microinjection and local chemical reactions that can induce morphological change in a vesicle. The first mechanism is local morphology change due to solvent swelling. Here a solvent can either be microinjected or can be produced by a local chemical reaction. If this solvent interacts favorably or unfavorably with the monomers that compose the polymer vesicle, this can result in swelling or deswelling respectively. The second mechanism of morphology change comes from alteration of the chemistry of the polymer blocks inside the vesicle, leading to a local change in the "shape parameter" of these blocks.[86] Here a reactant that is microinjected near the vesicle or produced locally via enzymatic reaction can react with a polymer block, resulting in a new local block chemistry or molecular weight that can subsequently alter the local morphology. We explore both of these mechanisms here, starting with solvent swelling and then discussing changes to the polymer shape parameter.

For both mechanisms, we explore a case with an instantaneous change in either solvent or polymer properties and a case with finite reaction kinetics. The simulations of the latter case are performed with a significantly shorter timescale than those of the former. We keep these simulations short because the fast reaction kinetics lead to a rapid conversion of solvent or polymer respectively that can obscure the localized deformation at longer times as the reaction proceeds.

Local Morphology Change by Solvent Swelling

To understand how local solvent gradients can alter the morphology of a polymer vesicle, we performed two different sets of *in silico* experiments, one mimicking solvent microinjection and the other simulating solvent production by a membrane-embedded enzyme. Both sets of calculations were preceded by the creation and equilibration of a large AB polymer vesicle with $N_b = 60$, $f_A = 0.2$ in the solvent S_A as described above.

We first mimicked solvent microinjection by "instantaneously injecting" a droplet of B-selective solvent S_B near the vesicle

membrane at the beginning of the simulation. Recall that B is the majority block that comprises the bilayer of the vesicle membrane. As shown in Fig. 8a-b, microinjection was achieved by converting a portion of S_A to a droplet of S_B near the surface of the vesicle. We varied the size of this droplet, ranging from 1.65% of the vesicle volume (2058 beads) to 8.07% (15584 Beads) of the vesicle volume and examined the swelling behavior as a function of time up to $t = 2.5 \times 10^4$.

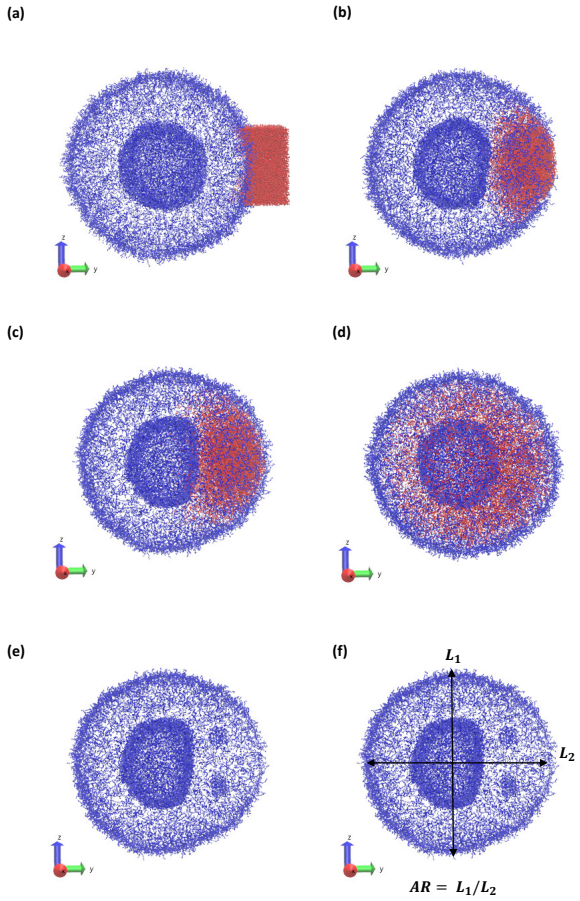

Fig. 8 Microinjection of a droplet of S_B that is 8.07% of the vesicle volume (15584 beads) near the surface of the vesicle at (a) $t = 0$, (b) $t = 200$, (c) $t = 500$, and (d) $t = 5000$. The polymer A block is shown in blue, S_B is colored red, and the polymer B block and S_A beads are not shown. Panel (e) shows $t = 500$ again, omitting S_B to highlight the flattening of the lumen wall and the formation of a micelle in the inner membrane. Panel (f) defines the aspect ratio (AR) as the ratio between the spans in z direction and y direction.

The vesicle dynamics following microinjection followed a similar pattern for all of the simulated cases, and therefore in Fig. 8 we highlight a single example. Fig. 8a shows the initial state of the simulation immediately following the microinjection event, where a droplet of S_B that is 8.07% (15584 beads) of the vesicle volume is injected. Following their introduction, the beads of S_B are all systematically drawn into the vesicle, with no S_B escaping into the bulk solvent, as shown in Fig. 8b. These beads segregate into the vesicle membrane where they associate with the polymer B-blocks. As shown in Fig. 8c, this leads to a noticeable flattening of the internal compartment of the vesicle co-localized with a swelling of the outer vesicle wall. At long times, the S_B beads diffuse more evenly throughout the vesicle membrane, and as shown in Fig. 8d, the vesicle returns to its spherical shape with slightly increased size due to the solvent swelling.

The different droplet sizes all follow the pattern in Fig. 8, but there is some noticeable variation. At large enough droplet sizes (3.3% of the vesicle volume and greater) small portions of the outer vesicle wall are carried with the droplet into the vesicle and form micelles within the vesicle membrane as can be seen in Fig. 8e. For intermediate droplet sizes (3.3% to 4.73% of the vesicle volume), these micelles are transient and merge with the internal wall of the vesicle at long times. However, for the largest droplet sizes (6.46% and 8.07% of the vesicle volume), these micelles appear to be metastable, and persist through the end of our longest simulations ($t = 2.5 \times 10^4$).

Additionally, the degree of local swelling and its effect on the shape of the vesicle also varies with the size of the microinjected droplet. To characterize this swelling, we calculate the aspect ratio of the outer vesicle wall as a function of time for each size of S_B droplet. Fig. 9a shows this calculation for the S_B droplet that is 8.08% the size of the vesicle, where the blue points represent the calculated aspect ratio at each each time point, and the curve applies the Savitzky–Golay filter to smooth the data points and better demonstrate the tendency.[87] As expected from Fig. 8, the aspect ratio initially decreases in time as the vesicle anisotropically swells. After a short period, the aspect ratio then increases, as the microinjected solvent diffuses throughout the vesicle membrane.

There is a curve analogous to Fig. 9a for each droplet size, and the minimum of this curve represents the maximum degree of local swelling. The aspect ratio at $t = 800$, at which the vesicles display the maximum degree of local swelling, is shown in Fig. 9b as a function of droplet size. The larger injected droplets produce smaller aspect ratios, indicating more dramatic changes to the vesicle. This is supported by our qualitative observation that the outer wall increasingly swells and the inner wall increasingly flattens as the droplet size increases.

In our second set of *in silico* experiments, we observed local solvent swelling driven by membrane-embedded "enzyme" particles E that catalyze the conversion of S_A into S_B according to

$$S_A + E \rightleftharpoons S_B + E. \tag{26}$$

These enzymes are introduced by switching the identities of the A-block of 50 polymer chains co-located in a region on the outer vesicle wall to be E beads. (Recall from Table 2 that E particles have the same DPD interaction parameters as A particles, so this does not change the shape of the vesicle.) The forward and reverse rate constants were set to 10 and 1 respectively, mimicking an enzyme that has a relatively high turnover number. With these constants, we estimate the Damkohler number to be,

$$\mathrm{Da} = \frac{k' L^2}{D_A} \approx 365 \tag{27}$$

where k' is the effective macroscopic rate estimated using Eq. 23,

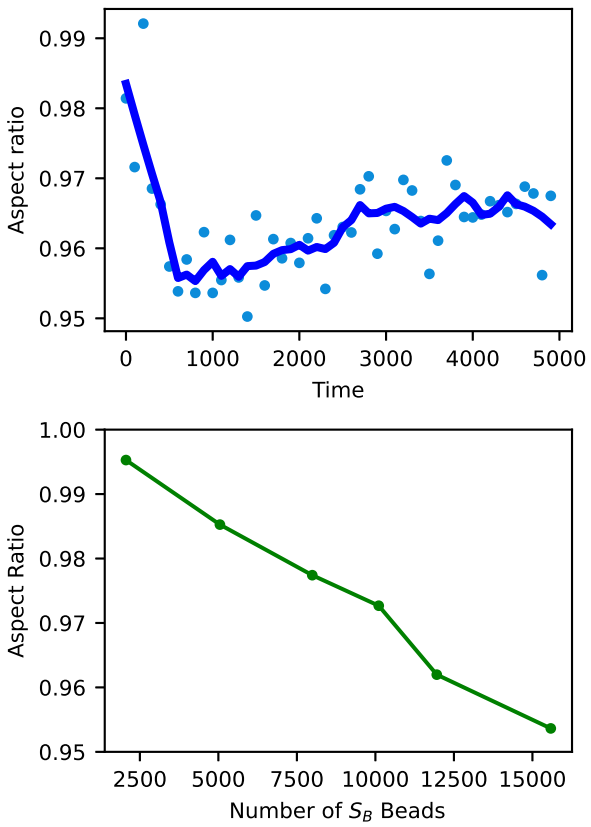

Fig. 9 (a) Aspect ratio as a function of time of a vesicle swollen with S_B solvent beads equal to 8.07% (15584 beads) of the vesicle volume. (b) The aspect ratio at $t = 800$ of the vesicle outer wall as a function of the size of the microinjected S_B droplet.

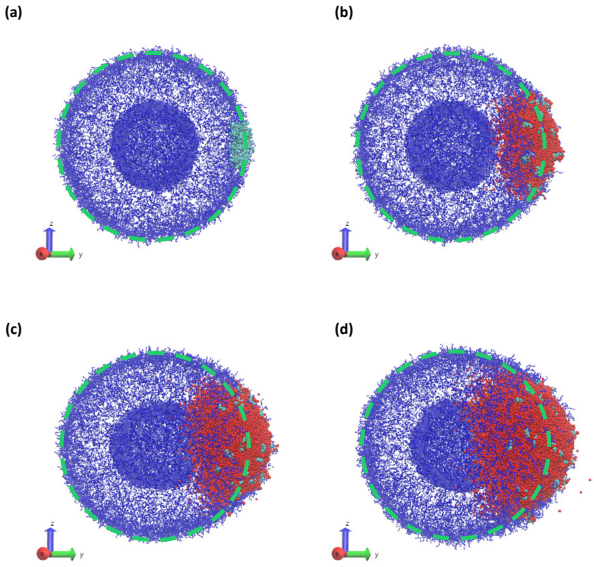

Fig. 10 Vesicle morphology at (a) $t = 0$, (b) $t = 75$, (c) $t = 150$, and (d) $t = 250$ while a chemical reaction produces S_B that swells the vesicle. Blue particles represent the monomers in the A-block of the polymer, cyan particles represent the enzyme (E) beads, and red particles represent S_B solvent particles. Monomers in the B-block and other solvent particles are not shown for clarity. The dashed green circle indicates the initial vesicle circumference as a guide to the eye.

and D_A is the tracer diffusivity estimated in Fig. 7. Note that the local value of Da may be significantly higher, because (i) this calculation assumes an evenly distributed mole fraction resulting in an underestimated rate constant, and (ii) the diffusivity was obtained for ideal solvent beads likely giving an overestimate. Regardless of the exact local value of Da, the rate of reaction is fast relative to diffusion, and the system is clearly operating in a diffusion limited regime.

Fig. 10 gives a time-series of the local morphology change as S_B is produced. Similar to the microinjection case, the vesicle quickly absorbs nearly all of the S_B, and the vesicle begins to swell locally. Due to the rapid production of S_B, the accumulation and swelling happens quickly within the vesicle membrane that contains the B-blocks and is already apparent at $t = 75$ in Figure 10b. Furthermore, due to the rapid production of S_B, the degree of deformation increases with time as more particles of S_B are produced before they could diffuse out, as shown in Fig. 10c-d. This can be seen from the dashed circle which outlines the spherical shape of the initial vesicle in each panel. Finally, note that particles of S_B largely remain near the B-blocks in the vesicle membrane, despite the deceptive appearance of Fig. 10d. The figure shows the 2D projections of a 3D object, and the solvent particles are also diffusing into the foreground obscuring one's view of the vesicle interior.

It is interesting to consider the similarities and differences between the microinjection and embedded-enzyme cases. In both cases, the S_B particles prefer to aggregate in the vesicle membrane, and there is a sequence where S_B is first absorbed into the vesicle before distributing throughout. However, in the enzymatically-driven case, the S_B particles remain "bunched up" throughout the entire simulation because the reaction rate is producing them faster than diffusion can disperse them. Additionally, there are also more S_B particles generated in the enzymatically-driven case than the pre-allocated droplets in the mircroinjection case. These two effects combine to give a larger degree of swelling for the enzyme-driven case relative to microinjection. However, we expect that if we stop the reaction at a given time and let the the S_B particles diffuse, they would distribute uniformly in within the vesicle membrane and would produce global swelling at long times similar to the microinjection case.

Local Morphology Change by Altering Polymer Packing Parameter

In addition to modifying the local vesicle morphology by solvent swelling, another mode for inducing changes to the vesicle structure is to directly alter the block polymer that makes up the vesicle. This modality has precedence for example in experiments that use pH-responsive polymers in polymersomes.[88,89] A simplistic way to conceptualize the local morphology change is via modification of the so-called packing parameter of the polymer, which has been widely used in explaining the self-assembly behavior of amphiphilic molecules.[90] The packing parameter of an

amphiphile is given by

$$p = \frac{v_0}{a l_0} \tag{28}$$

where v_0 and l_0 are the volume and the length of the amphiphile tail, and a is the equilibrium area per amphiphilic molecule at the head-tail interface.[90] Generally, amphphilic molecules with $p < 1/3$ form spherical micelles, those with $1/3 < p < 1/2$ form cylindrical micelles, and those with $1/2 < p < 1$ form bilayers or vesicles.

For block polymers, it has been argued that p is a function of the block fraction f_A, i.e. the ratio of the size of the hydrophilic block to the hydrophobic block.[91] For a fixed molecular weight, the a parameter increases with f_A, meaning that small f_A corresponds to lower curvature structures such as vesicles, while a larger f_A corresponds to higher curvature structures such as micelles. Accordingly, we hypothesize that it is possible to alter the curvature in a localized region of a vesicle by modifying the block fraction of a polymer in that region.

To test this hypothesis, we performed two different types of simulations. First, starting from an equilibrated vesicle obtained using the procedure described previously ($N_b = 60$, $f_A = 0.2$ and $x_p = 0.08376$), we instantaneously changed the block fraction of a series of vesicles. Though unphysical, this instantaneous change allows us to study the infinitely fast kinetic limit without the complication of finite reaction kinetics. Second, we simulate a more realistic scenario where we "microinject" a chemical stimulant $S_{A'}$ that converts beads in the solvophobic B-block in the polymer chain into beads of A according to

$$\mathrm{S_{A'}} + \mathrm{B} \rightleftharpoons \mathrm{S_A} + \mathrm{A}. \tag{29}$$

In this latter calculation, the conversion process results in a copolymer that is no longer strictly blocky and is better classified as an asymmetric random copolymer.

Figure 11 summarizes the results of the first class of simulations, where we instantaneously vary f_A of several chains on one side of the outer corona of the polymer vesicle. In these calculations we chose 174 co-localized chains, which is about 4% of the total number of chains in the vesicle structure, and varied f_A from 0.1 to 1.0 while retaining the chain position and orientation. In other words, the bead positions remain the same and the A block is still located in the outer corona. Recall that the original block fraction of the vesicle was $f_A = 0.2$. All simulations were performed for 5×10^5 steps in order to fully observe any changes in morphology.

We observed four qualitatively different behaviors as a function of f_A. For $0.1 < f_\mathrm{A} < 0.3$, there is little or no change in the morphology, and as shown in Figure 11a the vesicle retains its spherical shape. For $0.4 < f_\mathrm{A} < 0.7$, the vesicle remains largely spherical, but as seen in Fig. 11b there is an increase in curvature of the outer corona in the region where the chain block fraction was altered. For $0.7 < f_A < 1.0$, local swelling does not occur, and the chains are only weakly attached to the vesicle. Indeed, as is apparent in Figure 11c, some chains do not remain bound to the vesicle, but escape into the bulk solvent. Finally, for $f_A = 1.0$ (i.e. a homopolymer of A) there is no longer a thermodynamic force

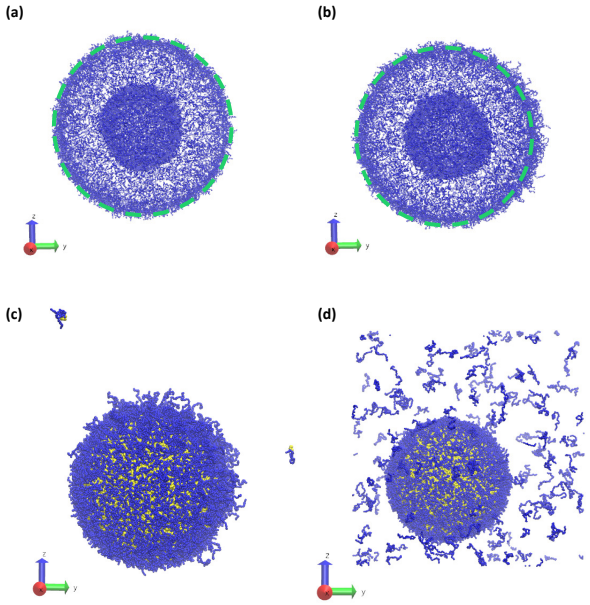

Fig. 11 Vesicle morphology at $t = 2.5 \times 10^4$ following an instantaneous change at $t = 0$ of the block fraction of a localized portion on the right side (positive y-direction) of the vesicle from $f_\mathrm{A} = 0.2$ to (a) $f_A = 0.1$, (b) $f_A = 0.4$, (c) $f_A = 0.8$, and (d) $f_A = 1.0$. As above blue beads show A-type monomers, and in panels (c) and (d) yellow beads represent B-type monomers. Solvent beads are not shown. The green circle in panels (a) and (b) shows the circumfrence of the original vesicle as a guide to the eye.

holding these fully solvophilic chains inside the vesicle, and as shown in Fig. 11d they eventually completely escape the structure. Note however that this does not destabilize the rest of the vesicle, and the unconverted chains remain.

Notably, the change to the polymer chemistry results in a longer-lived local deformation than the solvent swelling case. Local deformations persist in Fig. 11d until at least $t = 2.5 \times 10^4$, which is our longest run time for these simulations. By contrast, the local deformation in the solvent swelling case in Fig. 8d has completely disappeared by $t = 7.5 \times 10^3$. We attribute this difference in time to the relative rates of diffusion between solvent inside the vesicle membrane and the polymer chains that compose the vesicle.

In a more realistic scenario, Fig. 12 shows the results of the microinjection of a solvent stimulus followed by the chemical conversion of monomers on the B-block of the polymers inside the vesicle according to Eq. 29. We performed this calculation by instantaneously converting a droplet containing 11343 $\mathrm{S_A}$ beads (equivalent to about 5.54% of the volume of the vesicle) into beads of $\mathrm{S'_A}$ near one side of the vesicle surface at $t = 0$. We then ran the simulation with the reaction given in Eq. 29 where $k_f = 5$ and $k_r = 0.2$ giving a diffusion limited process with $\mathrm{Da} \approx 3450$ similar to the reactions performed previously. An additional plot characterizing the number of monomers that are converted as a function of time is given in Fig. S5 in the Supplementary Information.

Fig. 12 gives snapshots of a typical simulation as a function of time. Following the initial state in Fig. 12a, Fig. 12b-d show a

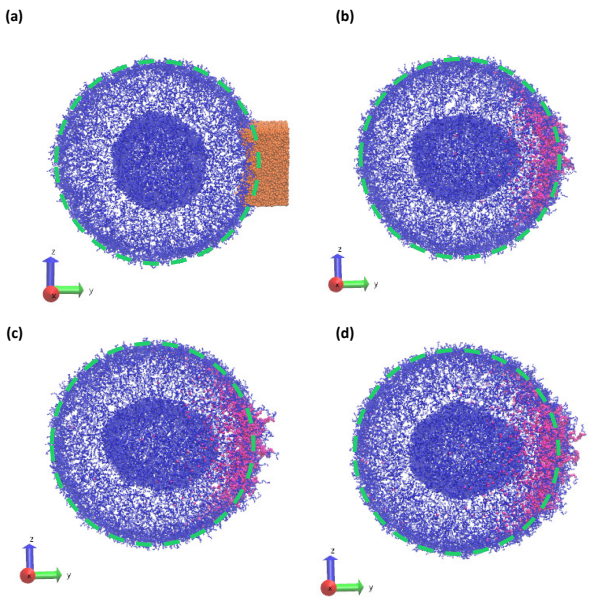

Fig. 12 Vesicle morphologies as a function of time as a stimulus converts solvophobic monomers near the vesicle outer wall to solvophilic monomers: (a) $t = 0$, (b) $t = 250$, (c) $t = 375$, and (d) $t = 500$. Orange particles in panel (a) represents the initially placed $S_{A'}$ particles. Blue particles represent A monomers, mauve particles represent monomers that were converted from A to B, and unconverted B monomers and solvent beads are not shown for better visualization of the vesicle structure. The dashed circles outlines the circumference of the original vesicle to highlight the local deformation. Other solvent particles were not shown for clarity.

locally swollen vesicle where the local curvature increases as a function of time. Notably, the change in polymer chemistry results in an increased positive curvature for both the outer vesicle wall and the boundary separating the lumen and the membrane. This is in contrast with the solvent swelling seen for example in Fig. 8, where the boundary between the lumen and membrane was flattened as it was pushed by excess solvent.

We claimed above that it was possible to alter the local curvature of a vesicle and create a more persistent local deformation by altering the polymer chemistry. To justify this claim more quantitatively, we calculate the curvature of a 2D projection of the vesicle before and after reaction to demonstrate the localized shape change. We do so by calculating a local concentration of monomers of B as a function of space, $x_B(\boldsymbol{r})$, using a grid. After smoothing this concentration function, we define the vesicle wall as the contour $x_B(\boldsymbol{r}) = C$ and project it onto the $y-z$ plane to define a two-dimensional space curve $\gamma(s)$. Parameterizing this curve as $\gamma(s) = (y(s), z(s))$ permits us to define a curvature[92]

$$\kappa = \left| \frac{y'z'' - z'y''}{\left[(y')^2 + (z')^2\right]^{3/2}} \right| \tag{30}$$

where κ is the absolute curvature, $y' = dy/ds$, $z' = dz/ds$, $y'' = d^2y/ds^2$, and $z'' = d^2z/ds^2$. Additional details related to the smoothing procedure and curvature calculation can be found in the Supporting Information.

Fig. 13 shows an analysis of the curvature of the initial and final vesicles from Fig. 12. Fig. 13a-b shows a projection of the initial vesicle shape and the absolute curvature as a function of an index that traces the circumference of the shape. Clearly, the projection is circular, and the curvature fluctuates about 0.035, the curvature for a circle with average diameter $56.8r_c$. There are significant fluctuations about $\kappa = 0.035$ despite our smoothing techniques because of (i) thermal fluctuations that make the vesicle an imperfect sphere, (ii) discrete DPD beads and a grid mapping that yields noisy concentration fields, and (iii) an amplification of noise due to the numerical calculation of first and second derivatives in Eq. 30.

Fig. 13c shows the projection of the vesicle at $t = 500$ after the reaction has occurred and clearly shows a local deformation of the vesicle structure in the positive y-direction. Concomitantly, the curvature in Fig. 13d shows a sharp increase at these circumferential indices, but is otherwise similar to the original vesicle in other locations. Interestingly, the curvature does not increase smoothly, but shows significant fluctuations in the region where the reaction occured. This heterogeneity may be simply a consequence of noise or of the scale of the curvature calculation. Alternatively, because the reaction forms random copolymers, these fluctuations may be inherent to the reaction-driven change and may have important physical effects. We leave the latter hypothesis to future investigation.

To better understand how to modulate the extent of deformation, we also performed a series of simulations with a generalization of Eq. 29,

$$S_{A'} + B \rightleftharpoons S_A + C \tag{31}$$

where a stimulus converts monomers in the solvophobic B-block into new type-C monomers. We let the interaction parameter between A and C (a_{AC}) vary from from 25 to 150 while keeping the sum of a_{AC} and a_{BC} constant at 185. This modulates the compatibility of C with both the solvophobic and solvophilic blocks. When $a_{AC} = 25$ ($\chi_{AC} = 0$), C beads are equivalent to A beads, and as above, this leads to additional curvature as the local packing parameter is decreased. When $a_{AC} = 150$ ($\chi_{AC} = 35.75$), C beads are almost completely compatible with B beads, and the local packing parameter is nearly unperturbed from the neat vesicle.

Fig. 14 shows the average aspect ratio of 10 replicate runs at $t = 500$ after the initiation of the reaction versus the Flory–Huggins parameter χ_{AC}. The aspect ratios are calculated by sweeping orthogonal directions with an increment of $\pi/10$ and locating the direction with the smallest aspect ratio. The values of χ_{AC} were obtained from the a_{ij} parameters using Eq. 10.

As expected, the aspect ratio is the smallest, i.e. the vesicles are the most deformed, when χ_{AC} approaches 0. As χ_{AC} increases, the aspect ratio also increases and then levels off at the aspect ratio of the original vesicle, $AR_0 = 0.975$. Interestingly, the aspect ratio first reaches AR_0 near the point of neutral interaction where $\chi_{AC} = \chi_{BC} = 19.3$. It seems plausible that this value of the interaction parameter marks the point where a newly created monomer of C no longer creates a significant driving force for expulsion from the solvophobic vesicle membrane to create extra curvature. Additionally, we fit the aspect ratio data in Fig. 14 to a

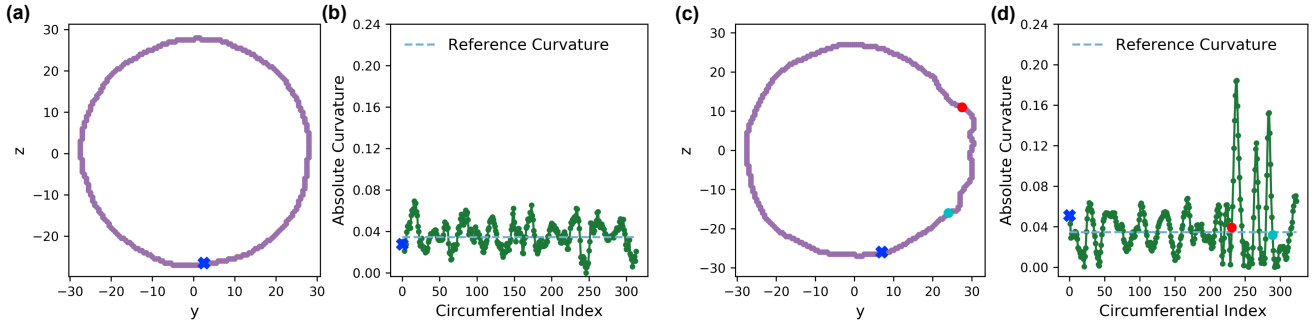

Fig. 13 (a) Projection of the vesicle boundary in the $y-z$ plane at $t=0$. (b) Absolute curvature as a function of an index of the circumference of the vesicle at $t=0$. The 0 of the index starts at the blue x and proceeds clockwise. (c) Projection of the vesicle boundary in the $y-z$ plane at $t=500$ after chemical reactions alter the solvophobicity of chains in the vesicle wall. (d) Absolute curvature as a function of circumferential index at $t=500$. The red and blue points in panels (c) and (d) correspond to the same points along the circumference. The cyan dashed line in panels (b) and (d) provides a reference curvature for a circle with a diameter of 56.8 r_c, which is the average size of the initial vesicle.

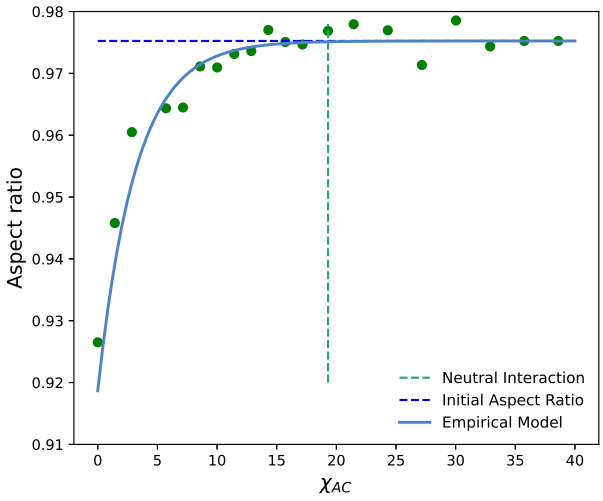

Fig. 14 Aspect ratio (AR) as a function of the interaction parameter, χ_{AC}, of newly created monomers on the solvophobic block. The neutral interaction point occurs when C particles can no longer distinguish between A and B beads, i.e. when $\chi_{AC}=\chi_{BC}$.

three-parameter empirical model,

$$AR = AR_0 - ae^{b(\chi_{AC}+c)} \tag{32}$$

where AR is the predicted aspect ratio, and $a=0.119$, $b=-0.315$ and $c=2.361$ are the parameter estimates. The empirical fit provides a useful summary of the data that may be useful for future comparison.

Conclusion

We have developed and validated a GPU-accelerated *RDPD* code that combines a DPD model and an SRBD reaction kinetics model to simulate the dynamics of self-assembled polymer solutions due to chemical reactions. Notably, our use of just-in-time compiled Python tools has allowed us to create this tool in a relatively short time with speeds that approach those of the much more sophisticated HOOMD-Blue.[66]

After validating the code, we performed a series of *RDPD* simulations to study the manipulation of local shape change of polymer vesicles. We first investigate the local morphology change due to solvent swelling. We performed two different sets of *in silico* experiments, one mimicking solvent microinjection and the other simulating solvent production by a membrane-embedded enzyme. Our results suggest that the generated solvophobic S_B particles tend to aggregate within the B blocks, causing a local swelling at the injection site or reactive site, and that the extent of deformation increases with the number of injected or converted S_B particles. However, introducing S_B particles does not result in a persistent local deformation, since the S_B particles rapidly disperse throughout the solvophobic membrane layer, resulting in a globally swollen morphology.

We also demonstrated local morphology change in the vesicle structure by altering the solvophobicity of the block polymer either instantaneously or with the introduction of external stimuli. Similar to the solvent swelling case, the polymer vesicle also displays an obvious localized swelling due to the decrease of solvophobicity of the B blocks. Here the local deformation has a longer time-scale that we attribute to the relatively long diffusion time for a polymer chain in the outer vesicle corona. We also showed that this deformation is tunable, based on the interaction parameter of the newly created monomer. These latter results imply that changing the polymer solvophobicity is a more practical approach for creating a persistent local deformation.

Even with the current progress, much remains to be studied in the present system. Experiments show more extensive local deformations such as large protrusions and vesicle fission, and the related parameter space for simulations has yet to be explored. For example, diblock copolymers of different molecular weight and block fraction may exhibit different membrane elasticities or diffusivities that could be highly relevant for such processes. Additionally, the results presented here are qualitative, and could benefit enormously from a more rigorous attempt to connect to experimental values. Finally, more complex systems such as vesicles that interact and communicate via chemical signals present many additional opportunities.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

We gratefully acknowledge financial support from Brigham Young University and the Simmons Research Endowment at Brigham Young University. We also acknowledge computational resources from the BYU Office of Research Computing.

Notes and references

1 E. Paluch and C. P. Heisenberg, *Curr Biol*, 2009, **19**, R790–R799.
2 S. M. Saporito-Irwin, C. E. Birse, P. S. Sypherd and W. A. Fonzi, *Mol Cell Biol*, 1995, **15**, 601–613.
3 S. J. Vainio, P. V. Itaranta, J. P. Perasaari and M. S. Uusitalo, *Int J Dev Biol*, 2003, **43**, 419–423.
4 H. Weinans and P. J. Prendergast, *Bone*, 1996, **19**, 143–149.
5 I. Hamley and V. Castelletto, *Angew Chem Int Edit*, 2007, **46**, 4442–4455.
6 R. V. Solé, A. Munteanu, C. Rodriguez-Caso and J. Macía, *Philos T R Soc B*, 2007, **362**, 1727–1739.
7 E. Rideau, R. Dimova, P. Schwille, F. R. Wurm and K. Landfester, *Chem Soc Rev*, 2018, **47**, 8572–8610.
8 N. P. Kamat, J. S. Katz and D. A. Hammer, *J Phys Chem Lett*, 2011, **2**, 1612–1623.
9 B.-Y. Xu, J. Xu and T. Yomo, *Biochem Soc T*, 2019, **47**, 1909–1919.
10 P. C. Letourneau, *Nerve cell shape*, Academic Press, Inc New York, 1989.
11 D. R. Green and F. Llambi, *CSH Perspect Biol*, 2015, **7**, a006080.
12 M. F. M. Sciacca, S. A. Kotler, J. R. Brender, J. Chen, D.-k. Lee and A. Ramamoorthy, *Biophys J*, 2012, **103**, 702–710.
13 Q. F. Ahkong and J. A. Lucy, *J Cell Sci*, 1988, **91**, 597–601.
14 A. F. M. Marée, V. A. Grieneisen and L. Edelstein-Keshet, *PLoS Comput Biol*, 2012, **8**, e1002402.
15 S. K. Yoo, Q. Deng, P. J. Cavnar, Y. I. Wu, K. M. Hahn and A. Huttenlocher, *Dev Cell*, 2010, **18**, 226–236.
16 H. M. Heath-Engel and G. C. Shore, *BBA-Mol Cell Res*, 2006, **1763**, 549–560.
17 M. Huo, G. Song, J. Zhang, Y. Wei and J. Yuan, *ACS Macro Lett*, 2018, **7**, 956–961.
18 J. C. Eloi, D. A. Rider, G. Cambridge, G. R. Whittell, M. A. Winnik and I. Manners, *J Am Chem Soc*, 2011, **133**, 8903–8913.
19 I. Lagzi, D. Wang, B. Kowalczyk and B. A. Grzybowski, *Langmuir*, 2010, **26**, 13770–13772.
20 I. A. B. Pijpers, L. K. E. A. Abdelmohsen, D. S. Williams and J. C. M. van Hest, *ACS Macro Lett*, 2017, **6**, 1217–1222.
21 K. T. Kim, J. Zhu, S. A. Meeuwissen, J. J. L. M. Cornelissen, D. J. Pochan, R. J. M. Nolte and J. C. M. van Hest, *J Am Chem Soc*, 2010, **132**, 12522–12524.
22 S. A. Meeuwissen, K. T. Kim, Y. Chen, D. J. Pochan and J. C. van Hest, *Angew Chme-Ger Edit*, 2011, **123**, 7208–7211.
23 D. A. Wilson, R. J. M. Nolte and J. C. M. van Hest, *Nat Chem*, 2012, **4**, 268–274.
24 A. F. Bitbol, J. B. Fournier, M. I. Angelova and N. Puff, *J Phys-Condens Mat*, 2011, **23**, 284102.
25 A.-F. Bitbol, N. Puff, Y. Sakuma, M. Imai, J.-B. Fournier and M. I. Angelova, *Biophys J*, 2012, **102**, 33a.
26 A. F. Bitbol and J. B. Fournier, *BBA-Biomembranes*, 2013, **1828**, 1241–1249.
27 W. F. Paxton, D. Price and N. J. Richardson, *Soft Matter*, 2013, **9**, 11295–11302.
28 N. Khalifat, N. Puff, S. Bonneau, J. B. Fournier and M. I. Angelova, *Biophys J*, 2008, **95**, 4924–4933.
29 N. Khalifat, J. B. Fournier, M. I. Angelova and N. Puff, *BBA-Biomembranes*, 2011, **1808**, 2724–2733.
30 V. Passos Gibson, M. Fauquignon, E. Ibarboure, J. Leblond Chain and J.-F. Le Meins, *Polymers-Basel*, 2020, **12**, 637.
31 Y. Miele, Z. Medveczky, G. Holló, B. Tegze, I. Derényi, Z. Hórvölgyi, E. Altamura, I. Lagzi and F. Rossi, *Chem Sci*, 2020, **11**, 3228–3235.
32 H. Maemichi, K. Shikinaka, A. Kakugo, H. Furukawa, Y. Osada and J. P. Gong, *Langmuir*, 2008, **24**, 11975–11981.
33 D. B. Wright, A. Ramírez-Hernández, M. A. Touve, A. S. Carlini, M. P. Thompson, J. P. Patterson, J. J. De Pablo and N. C. Gianneschi, *ACS Macro Lett*, 2019, **8**, 676–681.
34 X. Li, I. V. Pivkin, H. Liang and G. E. Karniadakis, *Macromolecules*, 2009, **42**, 3195–3200.
35 S. H. Chou, D. T. Wu, H. K. Tsao and Y. J. Sheng, *Soft Matter*, 2011, **7**, 9119–9129.
36 T. L. Rodgers, J. E. Magee, T. Amure and F. R. Siperstein, *Soft Matter*, 2016, **12**, 9014–9024.
37 X. Li, *Soft Matter*, 2013, **9**, 11663–11670.
38 Z. Luo, Y. Li, B. Wang and J. Jiang, *Macromolecules*, 2016, **49**, 6084–6094.
39 P. Sun, Y. Yin, B. Li, Q. Jin and D. Ding, *Int J Mod Phys B*, 2003, **17**, 241–244.
40 X. He, H. Liang, L. Huang and C. Pan, *J Phys Chem B*, 2004, **108**, 1731–1735.
41 V. Ortiz, S. O. Nielsen, M. L. Klein and D. E. Discher, *J Polym Sci Pol Phys*, 2006, **44**, 1907–1918.
42 N. K. Li, W. H. Fuss, L. Tang, R. Gu, A. Chilkoti, S. Zauscher and Y. G. Yingling, *Soft Matter*, 2015, **11**, 8236–8245.
43 J. A. Mysona, A. V. McCormick and D. C. Morse, *Phys Rev E*, 2019, **100**, 012602.
44 J. A. Mysona, A. V. McCormick and D. C. Morse, *Phys Rev E*, 2019, **100**, 012603.
45 M. Xiao, J. Liu, J. Yang, R. Wang and D. Xie, *Soft Matter*, 2013, **9**, 2434–2442.
46 S. Ma, M. Xiao and R. Wang, *Langmuir*, 2013, **29**, 16010–16017.
47 B. Gumus, M. Herrera-Alonso and A. Ramírez-Hernández, *Soft Matter*, 2020, **16**, 4969–4979.
48 A. Donev, C.-Y. Yang and C. Kim, *J Chem Phys*, 2018, **148**,

034103.
49 R. M. Elder, T. Emrick and A. Jayaraman, *Biomacromolecules*, 2011, **12**, 3870–3879.
50 R. M. Elder and A. Jayaraman, *J Phys Chem B*, 2013, **117**, 11988–11999.
51 A. H. de Vries, A. E. Mark and S. J. Marrink, *J Am Chem Soc*, 2004, **126**, 4488–4489.
52 G. Srinivas, D. E. Discher and M. L. Klein, *Nat Mater*, 2004, **3**, 638–644.
53 G. H. Fredrickson, *The Equilibrium Theory of Inhomogeneous Polymers*, Oxford University Press, 2006.
54 Y. Jiang, S. Li and J. Z. Chen, *Eur Phys J E*, 2016, **39**, 91.
55 L. Zhang, A. Sevink and F. Schmid, *Macromolecules*, 2011, **44**, 9434–9447.
56 D. R. Tree, K. T. Delaney, H. D. Ceniceros, T. Iwama and G. H. Fredrickson, *Soft Matter*, 2017, **13**, 3013–3030.
57 S. Qi and F. Schmid, *Macromolecules*, 2017, **50**, 9831–9845.
58 H. Chao, J. Koski and R. A. Riggleman, *Soft Matter*, 2017, **13**, 239–249.
59 G. D. Smith and D. Bedrov, *J Polym Sci Pol Phys*, 2007, **45**, 627–643.
60 A. Marciniak-Czochra and M. Kimmel, *Math Model Nat Pheno*, 2008, **3**, 90–114.
61 A. Mahmutovic, D. Fange, O. G. Berg and J. Elf, *Nat Methods*, 2012, **9**, 1163.
62 K. Takahashi, S. N. V. Arjunan and M. Tomita, *FEBS lett*, 2005, **579**, 1783–1788.
63 N. G. Van Kampen, *Stochastic processes in physics and chemistry*, Elsevier, Amsterdam, 1992, vol. 1.
64 R. Erban and S. J. Chapman, *Phys Biol*, 2009, **6**, 046001.
65 M. Davis and R. Davis, *Fundamentals of Chemical Reaction Engineering*, Dover Publications, 2013.
66 J. A. Anderson, J. Glaser and S. C. Glotzer, *Comp Mater Sci*, 2020, **173**, 109363.
67 C. L. Phillips, J. A. Anderson and S. C. Glotzer, *J. Comput. Phys.*, 2011, **230**, 7191–7201.
68 R. D. Groot and P. B. Warren, *J Chem Phys*, 1997, **107**, 4423–4435.
69 P. Español and P. Warren, *Europhys. Lett.*, 1995, **30**, 191–196.
70 P. Medapuram, J. Glaser and D. C. Morse, *Macromolecules*, 2015, **48**, 819–839.
71 M. Allen and D. Tildesley, *Computer Simulation of Liquids*, Clarendon Press, 1987.
72 W. Humphrey, A. Dalke, K. Schulten *et al.*, *J Mol Graphics*, 1996, **14**, 33–38.
73 D. P. Tolle and N. Le Novère, *BMC Syst Biol*, 2010, **4**, 24.
74 L. Sbailò and F. Noé, *J Chem Phys*, 2017, **147**, 184106.
75 B. Drawert, B. Jacob, Z. Li, T. M. Yi and L. Petzold, *J. Comput. Phys.*, 2019, **378**, 1–17.
76 T. Opplestrup, V. V. Bulatov, G. H. Gilmer, M. H. Kalos and B. Sadigh, *Phys. Rev. Lett.*, 2006, **97**, 230602.
77 T. Oppelstrup, V. V. Bulatov, A. Donev, M. H. Kalos, G. H. Gilmer and B. Sadigh, *Phys. Rev. E*, 2009, **80**, 066701.
78 M. Doi, *J. Phys. A. Math. Gen.*, 1976, **9**, 1479–1495.
79 M. Malek-Mansour and G. Nicolis, *J Stat Phys*, 1975, **13**, 197–217.
80 D. T. Gillespie, *Annu Rev Phys Chem*, 2007, **58**, 35–55.
81 D. T. Gillespie, A. Hellander and L. R. Petzold, *J. Chem. Phys.*, 2013, **138**, 170901.
82 S. Yamamoto, Y. Maruyama and S.-a. Hyodo, *J chem phys*, 2002, **116**, 5842–5849.
83 H. Tan, C. Yu, Z. Lu, Y. Zhou and D. Yan, *Soft Matter*, 2017, **13**, 6178–6188.
84 S. K. Lam, A. Pitrou and S. Seibert, Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC, 2015, pp. 1–6.
85 C. Nowak and F. A. Escobedo, *J Chem Theory Comput*, 2018, **14**, 5984–5991.
86 P. C. Hiemenz and R. Rajagopalan, *Principles of Colloid and Surface Chemistry, 3rd Edition*, CRC Press, 1997.
87 A. Savitzky and M. J. Golay, *Anal Chem*, 1964, **36**, 1627–1639.
88 S. H. R. Shin, P. T. McAninch, I. M. Henderson, A. Gomez, A. C. Greene, E. C. Carnes and W. F. Paxton, *Chemical Communications*, 2018, **54**, 9043–9046.
89 G. Kocak, C. Tuncer and V. Bütün, *Polym Chem-UK*, 2017, **8**, 144–176.
90 J. N. Israelachvili, *Intermolecular and Surface Forces*, Academic Press, San Diego, 3rd edn, 2011, ch. 20, pp. 535 – 576.
91 M. Antonietti and S. Förster, *Adv. Mater.*, 2003, **15**, 1323–1333.
92 S. Kobayashi and K. Nomizu, *Foundations of differential geometry*, New York, London, 1963, vol. 1.

Supporting Information: Using Reactive Dissipative Particle Dynamics to Understand Local Shape Manipulation of Polymer Vesicles

Qinyu Zhu, Timothy R. Scott, and Douglas R. Tree**

Chemical Engineering Department, Brigham Young University, Provo, Utah

E-mail: tree.doug@byu.edu

Description of Parallel Cell List Algorithm

Instead of using a Verlet list of neighboring particles within a certain cut-off radius, we divide the simulation space into cells of uniform size r_c and use a cell list to sort the particles and count the pairwise interactions.[S1] Here we briefly explain the basic concept of cell list structure using a 2D system, as shown in Figure S1. The cell list structure consists of a head array and a linked-list array. The system in Figure S1 is divided into nine uniform cells. Each element in the head array corresponds to one cell and stores the first particle's index that belongs to the cell. The elements in the head array also point to the address of the next particle index in the linked-list array. The elements in the linked-list represent the particle indices, and also points to the next particle in the same cell, and so on. If we follow the trace of the linked-list array, we will reach an element of -1, which means that we have iterated through all the particles in this cell.

When calculating the pairwise interactions for a particle, one only needs to check the particles in the same cell and half of the neighboring cells. For example, if we are calculating the pairwise interaction for particle 7 in Figure S1, we only need to check the particles in cells 0, 1, 2, and 5. Other pairs that involve particle 7 will be taken into account when we deal with particles in the other half of the neighboring cells. The cell list needs to be updated at every time step. The algorithm is efficient in large systems with a computational time that scales linearly with the number of particles.

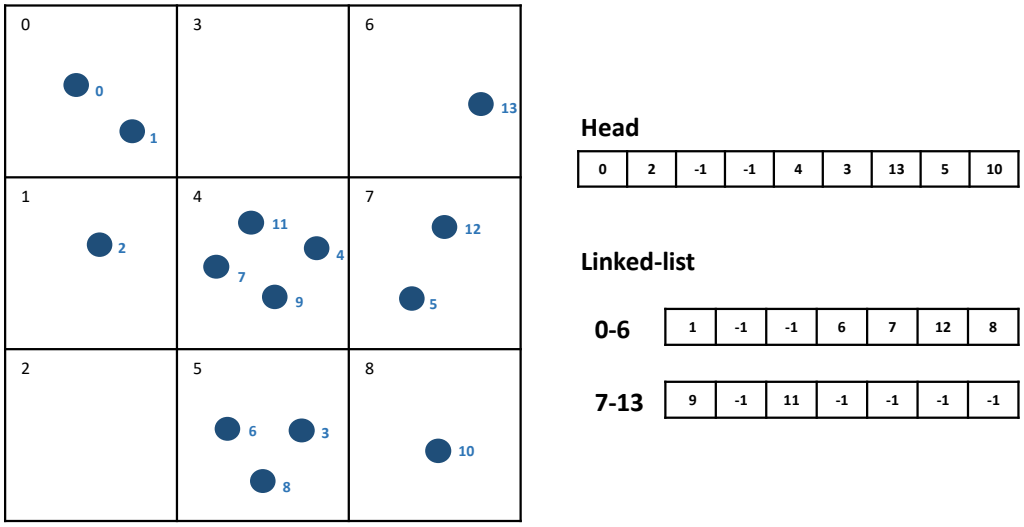

Figure S1: Demonstration of cell list structure in a 2D system.

However, constructing the cell list using the algorithm above is intrinsically a serial process, since we need to successively loop through each and every particle in the system to place them in the right location. Constructing the cell list on the CPU and then transferring it to the GPU is one less desirable option. This memory transfer process can be the rate-limiting step for a parallel computation. To avoid this, we used the the compare_and_swap function, which is an atomic operation in CUDA,[S2] to set up the cell list on GPU. The pseudocode for this procedure is found in Algorithm 1. To better interpret the algorithm, we define several functions here that we later refer to in Algorithm 1,

1. **Initialize**($head, clist$): initialize the head array ($head$) and linked-list array ($clist$) and set all the elements in both arrays to -1.

2. **Identify_Cell(**idx**)**: identify the cell index of a corresponding particle index idx.

3. **compare_and_swap(**$array[n], old\_val, new\_val$**)**: check if element $array[n]$ has the value of old_val. If so, then the new_val is assigned to this element; otherwise, the element remain unchanged.

We first initialize the head array and linked-list array on GPU and obtain the thread id from the GPU, as suggested in the first two lines. Each thread is then assigned to a single particle, and the cell index that the particle belongs to is identified. In line 5, we use the **compare_and_swap** function to attempt to swap the corresponding element in the head array with the particle index. If the index is successfully swapped, this thread returns and proceeds to process the next particle in the queue. Otherwise, we locate the next address in the linked-list and repeat the *compare_and_swap* function until a swap is successful, as suggested in the while loop from line 10 to line 18.

Algorithm 1 Parallel Cell List Construction

```
1: Initialize(head, clist)
2: Set idx = threadIdx.x + blockDim.x * blockIdx.x  #Thread id from GPU, also referred
   to as the particle index that the thread is processing
3: Set idcell = Indentify_Cell(idx)
4: compare_and_swap(head[idcell], −1, idx)
5: if head[idcel] = idx then
6:    return and proceed to next particle
7: else
8:    Set swap = True
9:    while swap = True do
10:      nextid ← head[idcell]
11:      compare_and_swap(clist[nextid], −1, idx)
12:      if clist[nextid]) = idx then
13:         swap = False
14:      else
15:         nextid ← clist[nextid]
16:      end if
17:   end while
18: end if
```

Relaxation Time of the Diblock Copolymer Solution

To characterize the relaxation time, we performed a DPD simulation using the same parameters as our polymer vesicle system described in the main text starting from random initial conditions. We ran the simulation for 5×10^5 steps, and the average radius of gyration (R_g) was obtained every 50 steps. The normalized autocorrelation function is plotted in Figure S2, where it shows an initially exponential decay before it becomes noisy due to insufficient statistics. Accordingly, we fit the data from $\tau = 0$ to $\tau = 20000$ using the following model

$$\ln E(\tau) = -\frac{\tau}{\theta} \tag{1}$$

where E is the normalized autocorrelation function of Rg, and θ is the autocorrelation time. The autocorrelation time obtained from the fit is $\theta \approx 1.43 \times 10^4$ timesteps. As mentioned in the main text, we typically ran for 1×10^6 timesteps for the final equilibration of the vesicle structure, which is approximately 70× the autocorrelation time. Thus, it is reasonable to conclude that our final system is effectively relaxed.

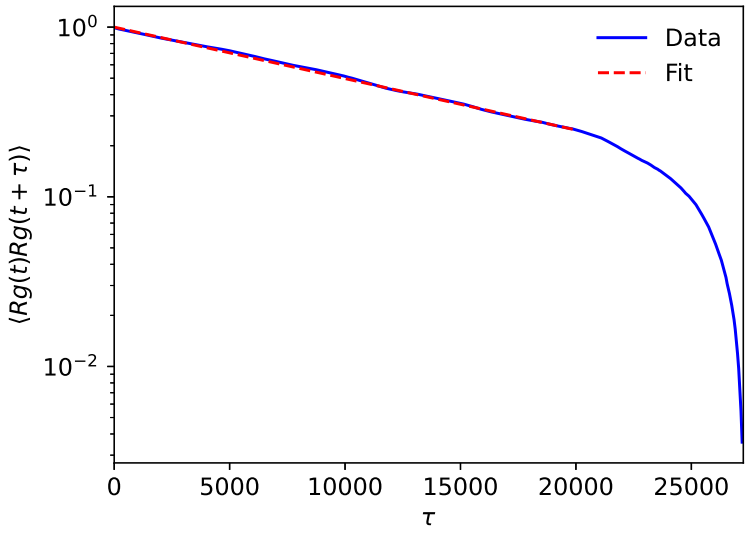

Figure S2: Normalized autocorrelation function of R_g.

Lateral Diffusion Coefficient of Polymer Chains in the Vesicle

In the main manuscript, we presented the result of mean squared displacement (MSD) of DPD solvent beads as a function of time and fitting the data points obtain tracer diffusion of the solvent S_A, $D_{S_A} = 0.2296$ in DPD unit. Here, We performed a similar calculation to obtain the lateral diffusion coefficient of polymer chains within the vesicle by tracking and averaging the MSD of center of mass of the polymer chains as a function of time and fitting the data point to

$$MSD = 6D_{\text{Lateral}}t \tag{2}$$

where D_{Lateral} is the lateral diffusion coefficient of the polymer chains in DPD unit. The MSD is plotted against time in Fig. S3. The fitted lateral diffusion coefficient is $D_{\text{Lateral}} = 0.0073$, which is approximately 30 times slower than that of the individual solvent bead.

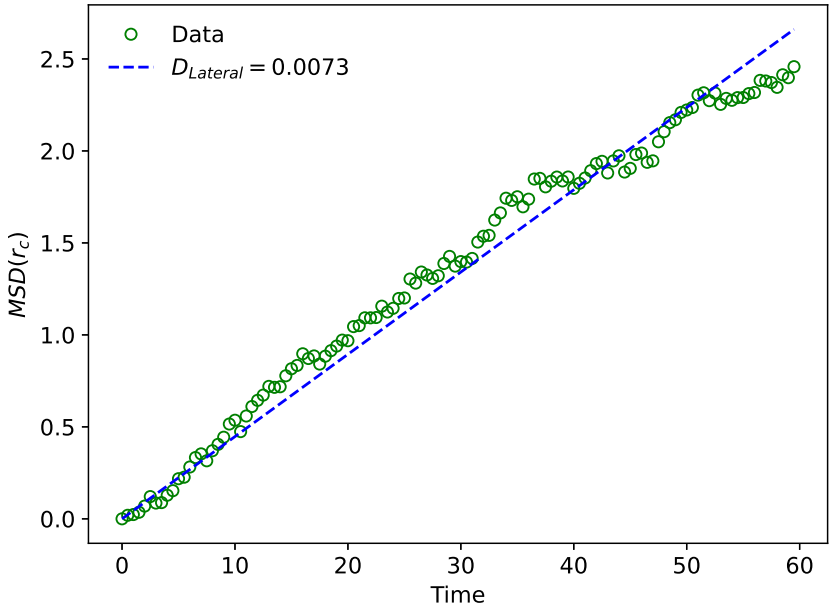

Figure S3: The MSD of polymer chains as a function of time.

Run-Time Comparison of DPD and *RDPD* Codes

In the main text, we show a bar chart that compares the run time of serial and parallel versions of DPD and *RDPD* codes. Here we attach the data we used to create the bar chart.

Table S1: Time comparison

Code Version	Run Time (min)
Serial DPD	24.62
Serial *RDPD* with Naive Algorithm	39.21
Serial *RDPD* with SRBD	32.54
Parallel DPD	1.27
Parallel *RDPD* with Naive Algorithm	1.60
Parallel *RDPD* with SRBD	1.43

Validation of SRBD Equilibria at Other Reaction Rates

In the main text, we validate the SRBD equilibrium by simulating the following catalytic conversion reaction,

$$S_{\mathrm{A}} + \mathrm{E} \rightleftharpoons S_{\mathrm{B}} + \mathrm{E}. \tag{3}$$

where $k_f = 0.1$ and $k_r = 0.05$. Here we show additional sets of forward and reverse reaction rates that also demonstrate similar equilibria. The simulations started with a random initial condition of equal mole fractions of A and B. All the simulations have the same forward reaction rate of $k_f = 0.1$. The reverse reaction rate in Figure S4 a, b, c and d are $k_r = 0.02$, 0.1, 0.2, and 0.5 respectively. The red lines represent the theoretical equilibrium mole fraction x_{A} for the corresponding reaction rate parameters. In Figure S4, the value of $x_{S_{\mathrm{A}}}$ fluctuates around the equilibrium mole fraction after $t \approx 1500$, in agreement with the expected values of equilibrium concentration with respect to the various sets of forward and reverse reaction rates.

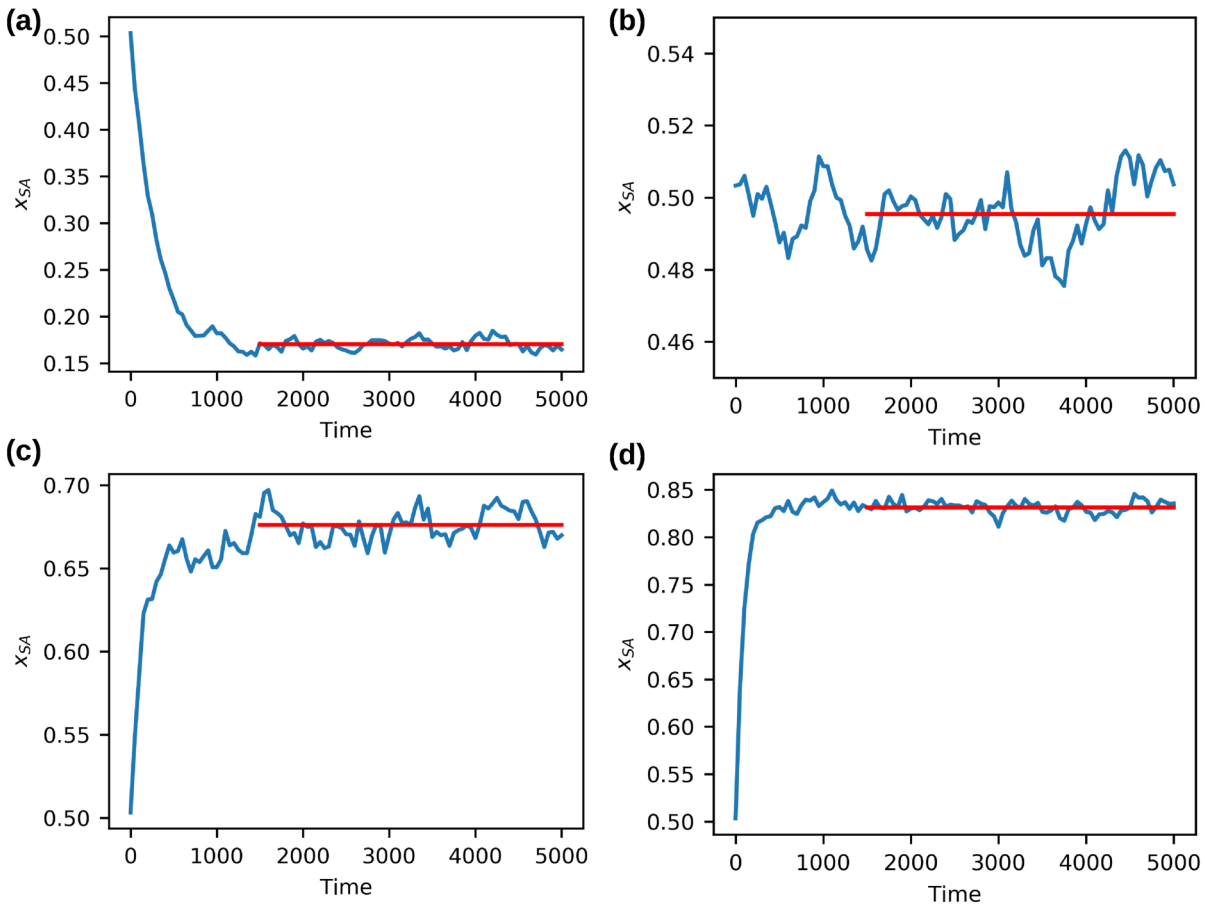

Figure S4: Mole fraction x_{S_A} as a function of time for reversible catalytic reactions at $k_f = 0.1$ and (a) $k_r = 0.02$, (b) $k_r = 0.1$, (c) $k_r = 0.2$, (d) $k_r = 0.5$.

Characterizing the monomer conversion in Fig. 12

To better demonstrate the process of dynamically changing polymer chemistry, as shown in Fig. 12 in the main text, we ran 20 replicates of the corresponding RDPD simulation, and plotted the number of converted B beads against the simulation time in Fig. S5. As suggested by the trend of the curve, the conversion rate slowly decreases as the reaction proceeds, which is reasonable because the consumption of reactants would decrease the probability of successful collision that leads to the chemical reactions. On average, 3681 B particles were converted at the end of the reaction (at $t = 500$).

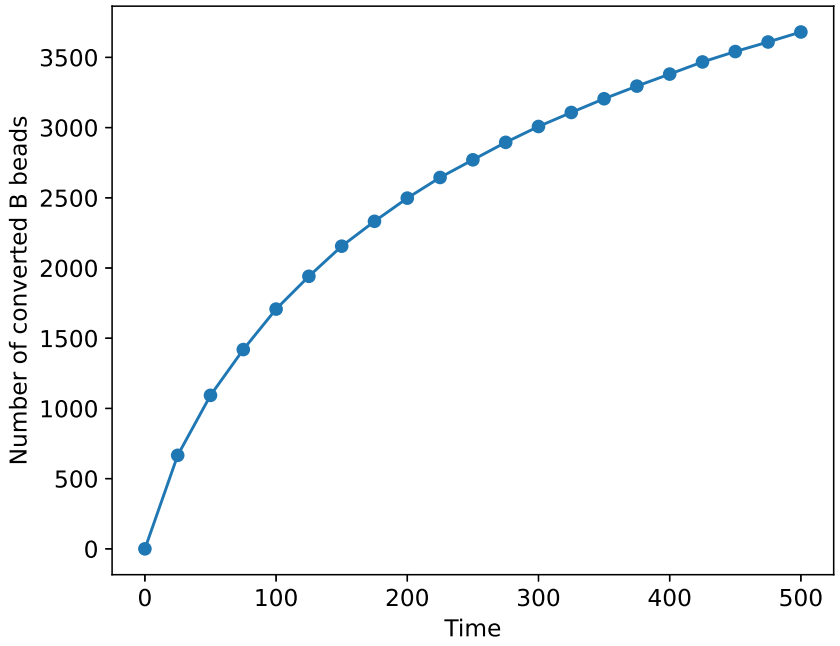

Figure S5: Number of B beads converted as a function of time.

Details of the Calculation of the Vesicle Curvature

As discussed in the main text, we estimated the vesicle curvature to show that the deformation occurs locally. The snapshots of the vesicle structure before and after dynamically changing the solvophobicity of B blocks were projected onto the Y-Z plane since the $S_{A'}$ particles were placed symmetrically with respect to the X and Z axes. The 2D space was divided into equally spaced grid points with a bin size of 0.5 in both directions. To get a smoother surface for a more accurate curvature estimation, we removed the A blocks in the outer corona, and we averaged the coordinates over 100 snapshots. Additionally, to further reduce the noise amplitude in the density calculation each particle was treated as a Gaussian distributed density pulse

$$f(y,z) = A\exp(-(\frac{(y-y_0)^2}{2{\sigma_y}^2} + \frac{(z-z_0)^2}{2{\sigma_z}^2})) \tag{4}$$

where y and z represent the grid point, and y_0 and z_0 are the coordinates of the particles. The amplitude was set as $A = 1$, and $\sigma_y = \sigma_z = 1$. A threshold between 0.5 and 75 was applied to obtain the grid points constituting the outer layer of the projection. The alpha shapes, which describe the boundary that envelops a set of points, were then defined with $\alpha = 2.0$ to obtain the outlines of the projection for curvature estimation.[S3] The curvatures of the 2D projection were estimated after parameterizing this curve as $\gamma(s) = (y(s), z(s))$ where s is a circumferential index. We then define a curvature as[S4]

$$\kappa = \left| \frac{y'z'' - z'y''}{\left[(y')^2 + (z')^2\right]^{3/2}} \right| \tag{5}$$

where κ is the absolute curvature, $y' = dy/ds$, $z' = dz/ds$, $y'' = d^2y/ds^2$, and $z'' = d^2z/ds^2$, respectively. The first and second derivatives in Eq. 5 were calculated based on a third order polynomial that was fit locally using a Savitzky–Golay filter with a window length of 23.[S5]

References

(S1) Allen, M.; Tildesley, D. *Computer Simulation of Liquids*; Oxford science publications; Clarendon Press, 1987.

(S2) Nickolls, J.; Buck, I.; Garland, M.; Skadron, K. Scalable parallel programming with CUDA. *Queue* **2008**, *6*, 40–53, DOI: 10.1145/1365490.1365500.

(S3) Edelsbrunner, H.; Kirkpatrick, D.; Seidel, R. On the shape of a set of points in the plane. *IEEE T Inform Theory* **1983**, *29*, 551–559, DOI: 10.1109/TIT.1983.1056714.

(S4) Kobayashi, S.; Nomizu, K. *Foundations of differential geometry*; New York, London, 1963; Vol. 1.

(S5) Savitzky, A.; Golay, M. J. Smoothing and differentiation of data by simplified least squares procedures. *Anal Chem* **1964**, *36*, 1627–1639, DOI: 10.1021/ac60214a047.