Electronic Supplementary Material (ESI) for Soft Matter.

Supporting information:

Multiscale Simulations of Polyzwitterions in Aqueous Bulk Solutions and Brush Array Configurations

Aristotelis P. Sgouros,[1] *Stefan Knippenberg,*[2] *Maxime Guillaume,*[2†] *and Doros N. Theodorou*[1*]

[1]*School of Chemical Engineering, National Technical University of Athens, 9 Heroon Polytechniou Street, Zografou Campus, GR-15780 Athens, Greece.*

[2]*Solid State Battery Applicability Laboratory, Solvay SA, 310 rue de Ransbeek, B-1120 Brussels, Belgium*

[†] *E-mail: maxime.guillaume@solvay.com*
[*] *to whom correspondence should be addressed. E-mail doros@central.ntua.gr, Tel. +30 210 772 3157, fax +30 210 772 3112*

Contents

S1. OPLS coefficients

The present section contains the modified OPLS coefficients derived from references 1,2. Any unspecified coefficient corresponds to the original OPLS parameters.[3]

Tables **S1** and **S2** depict the coefficients of the harmonic bonds [$\mathcal{V}_{\text{bond}}(r) = k_{\text{bond}}(r - l_0)^2$], and harmonic bond angles [$\mathcal{V}_{\text{angle}}(r) = k_{\text{angle}}(\theta - \theta_0)^2$].

Table S1. Coefficients of the harmonic bonds.

class1	class2	l_0 (nm)	k_{bond} (kj/mol/nm^2)	reference
S2	O2	0.144	585760	2
S2	CT	0.181	185770	2

Table S2. Coefficients of the harmonic angles.

class1	class2	class3	θ_0 (rad)	k_{angle} (kj/mol/rad^2)	reference
C	S2	O2	2.076942	870.27	2
C	S2	CT	1.900664	619.23	2
C	CT	S2	1.895428	418.4	2
C_2	CT	S2	1.911136	292.18	2

Table **S3** illustrates the coefficients of the dihedral angles, as described by the Ryckaert-Bellemans potential:

$$\mathcal{V}_{\text{dih}}\left(\varphi_{ijkl}\right) = \sum_{n=0}^{5} c_n \left[\cos\left(\psi_{ijkl}\right)\right]^n \qquad \text{(S1)}$$

using the convention, $\psi = \varphi - 180°$.

Table S3. Coefficients of the dihedrals. c_i are in units kJ/mol.

class1	class2	class3	class4	c_0	c_1	c_2	c_3	c_4	c_5	reference
CT	CT	CT	S2	3.9246	–3.925	0	0	0	0	2
CT	CT	CT	HC	0.7322	2.1966	0	0	0	0	2

Alternatively, the dihedral coefficients can be expressed through the Fourier description of the Ryckaert-Bellemans function:[4]

$$\mathcal{V}_{\mathrm{rb}} = \frac{1}{2}\left[K_1\left(1+\cos\left(\varphi\right)\right) + K_2\left(1-\cos\left(2\varphi\right)\right) + K_3\left(1+\cos\left(3\varphi\right)\right) + K_4\left(1-\cos\left(4\varphi\right)\right)\right] \tag{S2}$$

The conversion among c_i and K_i has as follows [see ref 5, p. 361]:

$$\begin{aligned} c_0 &= K_2 + \frac{1}{2}\left(K_1 + K_3\right) \\ c_1 &= \frac{1}{2}\left(-K_1 + 3K_3\right) \\ c_2 &= -K_2 + 4K_4 \\ c_3 &= -2K_3 \\ c_4 &= -4K_4 \\ c_5 &= 0 \end{aligned} \tag{S3}$$

Finally, Table **S4** depicts the coefficients of the nonbonded dispersive Lennard-Jones interactions, $V_{\mathrm{LJ}}\left(r\right) = 4\varepsilon[(\sigma/r)^{12} - (\sigma/r)^{6}]$, and the charges that tune the electrostatic interactions, $E_{\mathrm{coul}} = Cq_iq_j/\epsilon r$, where C is an energy conversion constant and ϵ is the dielectric constant.

Table S4. Coefficients of the nonbonded interactions.

Type name	class	element	m (g/mol)	q (e)	σ (nm)	ε (kj/mol)	reference
opls_O2_SPE	O2	O	15.9994	–0.755	0.296	0.7113	q from ref 1
opls_S2_SPE	S2	S	32.06	1.3193	0.355	1.046	q from ref 1
opls_288_SPE	opls_288_SPE	N	14.0067	0.1057	0.325	0.71128	q modified slightly to conserve charge for SPE

S2. Details about the mesoscopic polymer builder PyMeso

S2.1. Introduction

The mesoscopic configurations were generated using the PyMeso polymer builder, which was developed for the purposes of the present work. The builder generates the topology of the chains based on repeating units that have been defined by the user. The chains are reconstructed across a user-specified domain (either periodic or aperiodic) via a Monte-Carlo reverse sampling scheme. The polymer builder is generic, has been designed under the object-oriented programming paradigm in Python3.8, and supports polymeric chains of arbitrary chemical constitution and architecture (with the exception of loop topologies such as ring polymers). The final topology is extracted in the format of LAMMPS data files.

S2.2. Setup

Initially, the program requires the definition of the bead, strand and angle types, and their corresponding coefficients:

- **Beads.** id, mass, charge
- **Strands.** pair of bead types it connects, coefficients (stiffness and equilibrium length).
- **Angles.** triplet of bead types, coefficients (stiffness and equilibrium angle).

Using the bead and strand types defined previously, the user can create one or multiple kinds of repeating units in terms of the constituent bead/strand types, the internal topology (e.g., pairs of connected beads), and the head/tail of the repeating unit; i.e., the first/last bead. Note that, in case the repeating unit does not have a head/tail, it is treated as a single molecule; e.g., a water bead.

The repeating units can then be connected together and form “groups of repeating units”; i.e., chains. The connection of two repeating units implies that the head of the former is connected to the tail of the latter, and that the ids of the beads included in the new repeating unit are renumbered accordingly. By taking advantage of the operator overloading function, the connection of one or more repeating units with each other can be performed seamlessly via the “+” and “*” operators. For example, let there be two simple repeating units denoted as “A” and “B”, which comprise a single bead that is marked both as the head and tail of the repeating unit. In order to construct an “AAAABABA” group (we’ll call it

chain_4A2BA) we could either connect the aforementioned repeating units sequentially via the "+" operation:

chain_4A2BA = A + A + A + A + B + A + B + A

or we could take advantage of the "*" operator:

chain_4A2BA = 4 * A + 2 * (B + A)

Last, having specified all the required chains/molecules, the code requires the number of each chain kind to be included in the system, as well as the system dimensions and periodicity.

S2.3. Chain reconstruction

As soon as the required parameters have been specified, the algorithm will iterate over all groups (chains or molecules) included in the system and attempt to insert them one at a time. The builder utilizes a three-state reaction-like process to reconstruct the chains, in which the beads are categorized into the following states:

i) *reacted*, beads that have already been inserted in the system
ii) *react_next*, beads that are going to be inserted in the next iteration
iii) *unreacted*, beads that have not been inserted yet

At each iteration, the beads that have been labeled as *react_next* are inserted in the system and their state changes to *reacted*. In addition, the algorithm searches for the neighboring *unreacted* beads and changes their state to *react_next*. The aforementioned process is repeated until all beads of the group have been inserted to the system (i.e., they have been assigned the *reacted* state). Figure **S1** illustrates schematically the reaction process regarding the insertion of an A**(**B)-C**(**DE)-FG chain.

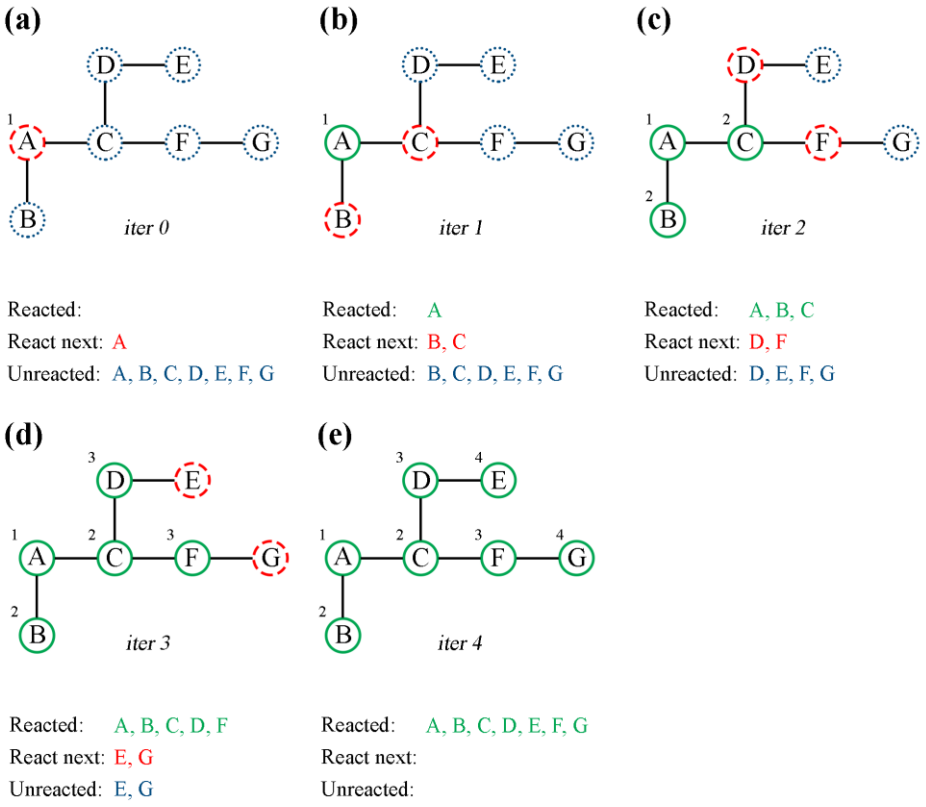

Figure S1. Schematic illustration of the insertion of an A(B)-C(DE)-FG chain. Solid circles depict beads that have been inserted in the box, circles with dashed contours indicate beads that will be inserted in the next iteration, and circles with dotted contours denote beads that have not been considered for insertion yet.

The placement of the beads across the simulation domain is performed as follows:

- The first reacted bead is placed at a random point, $\mathbf{r}_1$, within the specified domain of the simulation box.
- The second reacted bead is placed randomly on the surface of a sphere which is centered at $\mathbf{r}_2$ and has radius r_{12}. The radius of the sphere, r_{12}, is obtained by reverse sampling from the theoretical strand-length distribution corresponding to a type-t_1t_2 strand, with t_a denoting the type of bead with id a. Details about the reverse sampling procedure are illustrated in section **S2.4**.
- The insertion of the remaining beads requires a more complicated procedure. Suppose that the bead to be inserted has been assigned the id i, and that the ids of its first and second reacted neighbors are j and k, respectively. Based on the types of these beads we can obtain the length $r_{ji} = ||\mathbf{r}_{ji}|| = ||\mathbf{r}_i - \mathbf{r}_j||$ of the new t_jt_i-type strand, and the value, θ_{ijk}, of the new $t_it_jt_k$-type angle via reverse sampling. Therefore, we have to place the new bead in the system in a way that forms a strand with length r_{ji} and an angle θ_{ijk} with respect to its neighboring reacted beads, j and k. The new vector $\mathbf{r}_{ji}$

can be formed by rotating the unit vector $\mathbf{e}_{ij} = \mathbf{r}_{jk}/||\mathbf{r}_{jk}||$ with respect to a random rotation axis that is orthogonal to vector $\mathbf{r}_{jk}$ and passes from $\mathbf{r}_j$; the present implementation does not support dihedral angles and thus is dihedral angle is set to a random value. Subsequently, the rotated unit vector can be scaled with r_{ji}, and thus form the new strand. The rotation of the vector is performed using the quaternion multiplication method which is very efficient and avoids problematic gimbal lock situations as happens with the conventional Euler angle method.[6]

In situations where the coordinates of the new bead lie outside the accessible domain (i.e., due to enforcing aperiodic boundary conditions) the move is rejected and the insertion is reattempted.

S2.4. Reverse sampling

The strand-length distribution, $f(r)$, of a strand with free energy $U(r)$ (e.g., $U(r) = \lambda(r - r_0)^2$ for harmonic strands) can be calculated via Boltzmann inversion as follows:

$$f\left(r\right) = C \cdot 4\pi r^2 \cdot \exp\left(-\frac{U(r)}{k_{\mathrm{B}}T}\right) \tag{S4}$$

with C being a normalization constant. By integrating eq S4 we can calculate the cumulative probability distribution function,

$$F(r) = \int_0^r \mathrm{d}u\, f\left(u\right) \tag{S5}$$

The inverse cumulative distribution function, $F^{-1}(x)$, $x \forall [0,1]$, can be calculated by inverting F, either numerically or analytically. In order to reverse sample from the aforementioned distribution we have to evaluate $F^{-1}(u)$ with u being a random variable which is sampled from the homogeneous distribution across the interval [0, 1].

The exact same procedure can be repeated for performing reverse sampling from angle distributions, with the difference that,

$$f_{\mathrm{angle}}(\theta) = C' \cdot \frac{1}{\sin\left(\theta\right)} \cdot \exp\left(-\frac{U(\theta)}{k_{\mathrm{B}}T}\right) \tag{S6}$$

S3. Autocorrelation function of the tangential components of the end-to-end vector

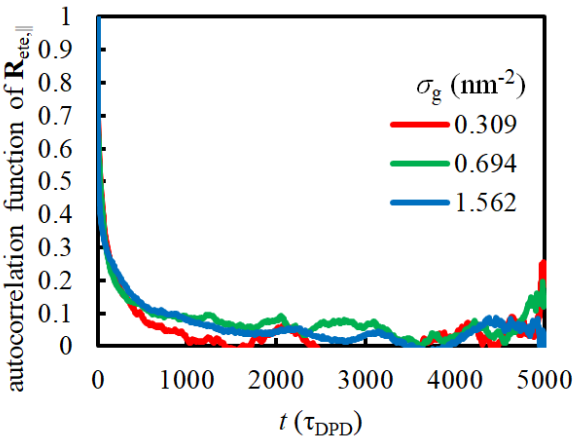

Figure S2. Time-autocorrelation function of the tangential components of the end-to-end vector, $\mathbf{R}_{ete,||} = \left(x_{end} - x_{graft}\right)\hat{x} + \left(y_{end} - y_{graft}\right)\hat{y}$, with $\alpha_{end}/\alpha_{graft}$ denoting the coordinate of the bead at the end/start of the grafted SPE chain along direction $\alpha=\{x, y\}$. In all cases the length of the grafted chains is $N_{SPE} = 5$.

S4. Brush-thickness scaling arguments for finitely extensible chains in a theta solvent

We consider a brush of finitely extensible chains of surface grafting density σ_g. Each chain is N statistical segments long. The segment size will be denoted as a. In this simple analysis, a is taken as a measure of both the statistical segment length (Kuhn length) and the diameter of a segment. The temperature is T. The brush is surrounded by a solvent, which is assumed to be a Θ solvent under the prevailing conditions. We wish to derive a scaling relation for the brush height, h_g, as a function of N and σ_g.

We adopt a simplified argument based on the work of Kuznetsov and Chen[7] for the brush in a theta solvent. The Helmholtz energy per chain, A, is envisioned as consisting of a (solvent mediated) interaction and an elastic contribution:

$$A = A_{\text{int}} + A_{\text{el}} \tag{S7}$$

For the interaction contribution we invoke the Alexander picture of a homogeneous gas of segments in the brush region. The volume of fraction of segments is (see also p. 681 in ref 8):

$$\phi = \frac{\text{volume of chain segments}}{\text{volume per chain}} = \frac{Na^3}{\frac{1}{\sigma_g} h_g} = \frac{Na^3\sigma_g}{h_g} \tag{S8}$$

The interaction free energy is expressed through a virial expansion. Under Θ conditions the second virial coefficient is zero and the leading term in the expansion is the third virial term. In the following, all equalities should be understood as rough, omitting multiplicative factors of order 1.

$$\frac{A_{\text{int}}}{k_B T} = \frac{\left(\text{Free energy density}\right)(\text{volume per chain})}{k_B T} = \frac{w}{a^3}\phi^3 \frac{h_g}{\sigma_g} \tag{S9}$$

with w being a dimensionless third virial coefficient for solvent-mediated interactions among the segments. In the Kuznetsov-Chen paper,[7] w is called υ.

Combining eqs S8 and S9:

$$\frac{A_{\text{int}}}{k_{\text{B}}T} = \frac{w}{a^3}\left(\frac{Na^3\sigma_{\text{g}}}{h_{\text{g}}}\right)^3 \frac{h_{\text{g}}}{\sigma_{\text{g}}} = \frac{wa^6N^3\sigma_{\text{g}}{}^2}{h_{\text{g}}{}^2} \tag{S10}$$

The elastic (entropy spring) contribution due to stretching is typically expressed based on a Gaussian chain picture (Hookean spring), $\frac{A_{\text{el}}}{k_{\text{B}}T} = \frac{h_{\text{g}}{}^2}{Na^2}$. Kuznetsov and Chen have modified this expression to include the finite extensibility of the chain in the following manner:[7]

$$\frac{A_{\text{el}}}{k_{\text{B}}T} = \frac{h_{\text{g}}{}^2}{N^2a^2}\frac{N}{1-\frac{h_{\text{g}}}{Na}} = \frac{h_{\text{g}}{}^2}{Na^2}\frac{1}{1-\frac{h_{\text{g}}}{Na}} \tag{S11}$$

For collapsed chains in a poor solvent, Kuznetsov and Chen append to eq S11 an additional, confinement entropy term of the form $\frac{Na^2}{h_{\text{g}}{}^2}$. We will not include that term here.

From (S7, S10, S11) the total reduced Helmholtz energy per chain is

$$\frac{A}{k_{\text{B}}T} = \frac{wa^6N^3\sigma_{\text{g}}{}^2}{h_{\text{g}}{}^2} + \frac{h_{\text{g}}{}^2}{Na^2}\frac{1}{1-\frac{h_{\text{g}}}{Na}} \tag{S12}$$

At equilibrium, this should be minimized with respect to h_{g}:

$$0 = \frac{\partial}{\partial h}\left(\frac{A}{k_B T}\right) \Rightarrow$$

$$0 = -2\frac{wa^6 N^3 {\sigma_g}^2}{{h_g}^3} + 2\frac{h_g}{Na^2}\frac{1}{1-\frac{h_g}{Na}} + \frac{{h_g}^2}{Na^2}\frac{1}{Na}\frac{1}{\left(1-\frac{h_g}{Na}\right)^2} \Leftrightarrow$$

$$0 = -2\frac{wa^6 N^3 {\sigma_g}^2}{{h_g}^3} + 2\frac{h_g}{Na^2}\frac{1}{1-\frac{h_g}{Na}} + \frac{{h_g}^2}{N^2 a^3}\frac{1}{\left(1-\frac{h_g}{Na}\right)^2} \Leftrightarrow$$

$$0 = -2\frac{wa^7 N^3 {\sigma_g}^2}{{h_g}^3} + 2\frac{h_g}{Na}\frac{1}{1-\frac{h_g}{Na}} + \frac{{h_g}^2}{N^2 a^2}\frac{1}{\left(1-\frac{h_g}{Na}\right)^2} \Leftrightarrow$$

$$0 = -2wa^4 {\sigma_g}^2\left(\frac{Na}{h_g}\right)^3 + 2\left(\frac{h_g}{Na}\right)\frac{1}{1-\left(\frac{h_g}{Na}\right)} + \left(\frac{h_g}{Na}\right)^2\frac{1}{\left[1-\left(\frac{h_g}{Na}\right)\right]^2} \quad \text{(S13)}$$

Setting, $\beta = \frac{h_g}{Na}$, eq S13 is rewritten as,

$$0 = -2wa^4 {\sigma_g}^2\frac{1}{\beta^3} + 2\beta\frac{1}{1-\beta} + \beta^2\frac{1}{(1-\beta)^2} \Leftrightarrow$$

$$0 = -2wa^4 {\sigma_g}^2\frac{1}{\beta^3} + \frac{2\beta - 2\beta^2 + \beta^2}{(1-\beta)^2} \Leftrightarrow$$

$$0 = -2wa^4 {\sigma_g}^2\frac{1}{\beta^3} + \frac{2\beta - \beta^2}{(1-\beta)^2} \Leftrightarrow \quad \text{(S14)}$$

$$wa^4 {\sigma_g}^2 = \frac{\beta^4(1-\beta/2)}{(1-\beta)^2}$$

So, the scaling analysis leads to

$$wa^4 {\sigma_g}^2 = \frac{\beta^4(1-\beta/2)}{(1-\beta)^2} \quad \text{(S15)}$$

For weak stretching, $\beta \ll 1$, eq S15 leads to:

$$\begin{aligned} w a^4 {\sigma_g}^2 &= \beta^4 \Leftrightarrow \\ \beta &= w^{1/4} a {\sigma_g}^{1/2} \Leftrightarrow \\ \frac{h_g}{Na} &= w^{1/4} a {\sigma_g}^{1/2} \end{aligned} \tag{S16}$$

or

$$h_g = w^{1/4} a^2 N {\sigma_g}^{1/2} \tag{S17}$$

which is the classical scaling result for a Θ solvent, $h_g \sim N^1 {\sigma_g}^{1/2}$.

When considering stretched brushes, (large β), the full right hand side of eq S15 is retained, and we have a functional relation between $w^{1/2} a^2 \sigma_g$ and β, which is plotted in Figure **S3**.

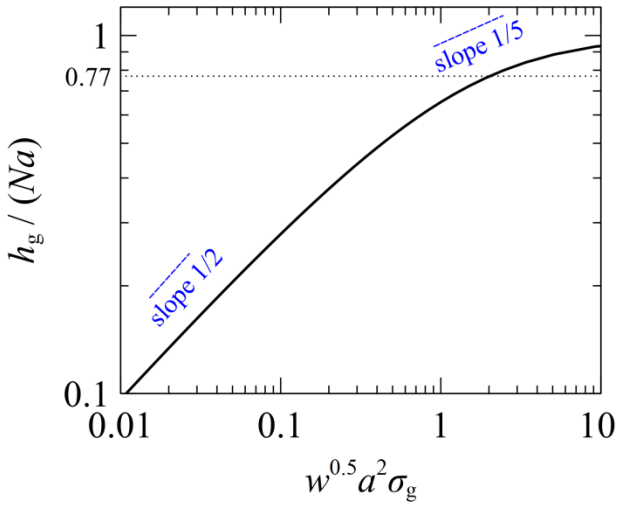

Figure S3. The functional dependence between $w^{1/2} a^2 \sigma_g$ and $\beta = h_g/(Na)$ according to eq S15. The dotted line depicts the value of h_g/Na for which the scaling exponent of σ_g becomes 1/5.

According to this simple model, the scaling exponent for the dependence of h_g on N is always 1. The scaling exponent for the dependence of h_g on σ_g, however, varies. At low stretching of the brush (weak

third virial coefficient interaction) it is 1/2. At high stretching, however, it can be 1/5 or lower. Similar response has been reported for good solvents as well, wherein the scaling exponent of the grafting density can decrease significantly relative to its classical value of 1/3 in situations with enhanced orientation and chain stretching.[99,101]

It is worth mentioning that the aforementioned model breaks down in the limit of low σ_g and N (mushroom regime), where the chains cannot interact with each other and the brush thickness becomes independent of σ_g. Therefore, the behavior of the brush in terms of the scaling exponent of σ_g (let's call it n) can be classified into three regimes: i) mushroom regime, $n = 0$; ii) intermediate regime, $n = 1/2$; iii) stretching regime, $0 \leq n < 1/2$.

The effective segment size can be estimated for our SPE brushes in terms of the following equation:

$$a = l \sin(\theta / 2) \tag{S18}$$

with θ being the angle among three consecutive backbone beads, and l being the length of a backbone bond. Setting $\theta = \theta_{\text{SB-SB-SB}} = 2.47$ rad and $l = l_{\text{SB-SB}} = 0.3$ nm (see Fig. **10** in the main text) we get a rough approximation for the effective segment length along the direction of the chain contour, $a_{\text{SPE}} = 0.28$ nm. Note that the actual SPE chains feature large side groups that would lead to anisotropic a_{SPE} along the tangential directions with respect to the chain contour; these complications will be not taken into account here. For the dense brushes considered here we get $h_g/(N_{\text{SPE}}\, a_{\text{SPE}}) \sim 0.6$, which is certainly in the domain $n < 1/2$, in reasonable agreement with the analytical scaling model considered here, given its simplicity.

S5. Nonergodic angle distributions

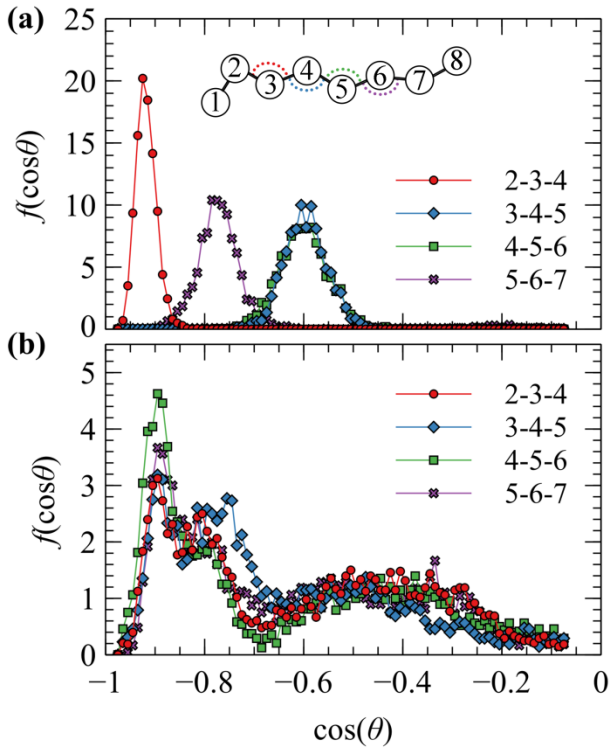

Figure S4. Strand-angle distributions of the inner SB-SB-SB triplets of SPE_8 chains; the angles at the chain ends are omitted. **(a)** Distributions from single MD simulations at T = 300 K. **(b)** Distributions based on the 120 temperature-annealed configurations. The inset in (a) depicts the backbone beads of an SPE_8 chain.

S6. Calibration of the bonded coefficients

Before discussing the calibration procedure, we first have to define the *total objective function* that quantifies the performance of the input set of bonded coefficients (strand or angle) as well as the *partial objective functions* that quantify the performance of individual sets of bonded coefficients in terms of reproducing the target bonded distributions. Since we deal with Gaussian strands and angles, the input set of coefficients can be defined as $\left[\left(k_\alpha, \mu_\alpha\right), \alpha \in \left\{\text{strand and angle types}\right\}\right]$ where (k_α, μ_α) = ($k_{\text{strand},\alpha}$, $l_{0,\alpha}$) when referring to kind-α Gaussian strands, and (k_α, μ_α) = ($k_{\text{angle},\alpha}$, $\theta_{0,\alpha}$) when referring to kind-α Gaussian angles.

The total objective function takes as an input the full list of the bonded coefficients of the strands and angles we wish to optimize, and outputs the overall performance of the full set of input coefficients in terms of the *total R*-square measure, which is defined as follows:

$$R^2_{\text{total}} = \sum_{a \in (\text{strand + angle types})} w_\alpha R^2_\alpha \tag{S19}$$

where R^2_α constitutes the *partial* objective function; i.e., an *R*-square measure that quantifies the goodness of the fit of *individual*, kind-α bonded coefficients with their corresponding target distributions, with α denoting a specific strand or angle type. w_α is the weight ascribed to R^2_α for a specific type; in our implementation $w_\alpha = 1$, thus, all strands and angles were weighted equally.

The partial objective function, R^2_α is defined as follows:

$$R^2_\alpha = 1 - \frac{\text{SSE}_\alpha}{\text{SST}_\alpha} \tag{S20}$$

with SSE_α being the sum of squared estimate of errors:

$$\text{SSE}_\alpha = \sum_{i=0}^{i_{\max}} \left\{ f_\alpha^{\text{sim}}[i] - f_\alpha^{\text{target}}[i] \right\}^2 \tag{S21}$$

SST being the sum of squares about the mean, $\overline{f}_\alpha^{\text{sim}}$:

$$\mathrm{SST}_\alpha = \sum_{i=0}^{i_{\max}} \left\{ f_\alpha^{\mathrm{sim}}[i] - \overline{f}_\alpha^{\mathrm{sim}}[i] \right\}^2 \tag{S22}$$

where $\overline{f}_\alpha^{\mathrm{sim}}$ is the mean of the distribution, and $f_\alpha^{\mathrm{sim}}[i]$ and $f_\alpha^{\mathrm{target}}[i]$ denote the value of the simulated and target bonded distributions at the i^{th} discrete bin that denotes a specific strand-length/angle when referring to strand/angle distributions. Note that, the way R_α^2 has been defined, a value close to unity corresponds to a good fit between the simulated distribution and the target one, whereas lower values indicate bad fit.

The algorithm requires the following inputs to run:

1. The parameters of the target distributions to be replicated. Since we deal with Gaussian distributions, their corresponding mean, μ_α, and standard deviation, σ_α, are required.
2. Initial guesses for the bond and angle coefficients to be optimized. These can be set to the coefficients derived via the initial Boltzmann inversion.

During the initialization phase, the algorithm performs an *evaluation* using the initial set of bonded coefficients and stores the corresponding R_α^2 and R_{total}^2, as well as the set of the initial coefficients to a list, which we will call $\mathbf{L}_{\mathrm{ext}}$.

Note that, each "evaluation" performed by the algorithm entails the following steps: i) a fresh DPD simulation based on the input set of bonded coefficients ($\mathbf{L}_{\mathrm{ext}}$ or $\mathbf{L}_{\mathrm{int}}$) and in the presence of the full nonbonded interactions, for a duration of 1500 τ_{DPD}; ii) post-processing of the trajectories in order to derive the bonded distributions of individual strands/angles; iii) calculation of the total and the partial objective functions via eqs S19 and S20.

Subsequently, the bonded coefficients are calibrated based on a two-level Monte Carlo (MC) optimization procedure:

Internal level. Across this level, the algorithm attempts to optimize the individual bonded coefficients for n_{internal} internal MC iterations in a way that maximizes their individual performance of the strands/angles, as quantified by the partial R_α^2. In other words, the individual strands and angles are treated as being decoupled; i.e., it is assumed that a perturbation to a kind-α bonded distribution does not affect the distributions of the other kinds of bonded distributions. Note that, during the internal iterations

the external list, $\mathbf{L}_{\text{ext}}$, remains unaffected, whereas any adjustments are stored in a copy of this list, which we will call $\mathbf{L}_{\text{int}}$ from here on. At the start of each internal iteration, the coefficients of the bonded interactions are adjusted simultaneously based on the following relations:

$$k'_\alpha = k_\alpha \left(1 + \lambda_\alpha \Delta k_{\max} U\left(-1,1\right)\right) \tag{S23}$$

$$\mu'_\alpha = \mu_\alpha \left(1 + \lambda_\alpha \Delta \mu_{\max} U\left(-1,1\right)\right) \tag{S24}$$

and

$$\lambda_\alpha = 1 - R_\alpha^2 \tag{S25}$$

where $\Delta k_{\max}$ and $\Delta \mu_{\max}$ denote the maximum change in strand/angle stiffness and mean, $U(-1,1)$ is a function that samples from the uniform distribution across the interval (–1, 1), and λ_α is a relaxation quantity whose magnitude depends on the accuracy of the fit. Note that, as the fit improves, $R_\alpha^2 \to 1$, $\lambda_\alpha \to 0$, and as a result the perturbations in k_α and μ_α become minor; i.e., convergence is achieved Subsequently, the algorithm performs an *evaluation* based on the set of the adjusted bonded coefficients, $\left[\left(k'_\alpha, \mu'_\alpha\right), \alpha \in \{\text{strand and angle types}\}\right]$, and it computes the new partial objective functions, R'^2_α, via eq S20. At the end of each internal iteration, the adjustments to individual bonded coefficients (internal MC step) are accepted/rejected based on the following criterion:

- If $R'^2_\alpha > R^2_\alpha$, the adjusted coefficients are accepted and the corresponding coefficients in list $\mathbf{L}_{\text{int}}$ are updated: $\left(k_\alpha, \mu_\alpha\right) \to \left(k'_\alpha, \mu'_\alpha\right)$
- If $R'^2_\alpha \le R^2_\alpha$, the adjusted coefficients are rejected and thus corresponding coefficients in list $\mathbf{L}_{\text{int}}$ are maintained.

External level. After n_{internal} iterations, the algorithm performs an *evaluation* based on the list $\mathbf{L}_{\text{int}}$, and based on the new value of the total objective function, R'^2_{total}, it accepts/rejects the series of internal MC moves as follows:

- If $R'^2_{\text{total}} > R^2_{\text{total}}$, the series of internal MC moves is accepted and the external list of coefficients is replaced by the internal one; $\mathbf{L}_{\text{ext}} \to \mathbf{L}_{\text{int}}$.

- If $R'^2_{\text{total}} \leq R^2_{\text{total}}$, the series of internal MC moves is rejected, and the external list of coefficients remains unaffected.

The algorithm cycles between the internal and external levels until convergence is achieved. The parameters of the optimizer were set to $\Delta k_{\max} = \Delta \mu_{\max} = 0.5$, (in DPD units), and $n_{\text{internal}} = 5$.

S7. Optimization of the nonbonded coefficients

The objective function takes as input the list of DPD nonbonded coefficients, a_{ij}, and outputs the mean sum square error (MSSE) between the DPD and MD partial pair distribution functions. In detail, the workflow inside the objective function is the following:

1. Generation of a LAMMPS script based in the input list of a_{ij}.
2. Generation of a fresh initial configuration
3. Simulation of the sample with an initial equilibration of $t_{\text{equil}} = 500\ \tau_{\text{DPD}}$, and a production phase which lasts for $t_{\text{prod}} = 10000\ \tau_{\text{DPD}}$, during which the trajectories of the DPD beads are recorded at time intervals of $\Delta t = 2\ \tau_{\text{DPD}}$.
4. Calculation of the partial pair distribution functions, g_{ij}, from the corresponding trajectories.
5. Calculation of the sum square error between the g_{ij} from MD and DPD and averaging of it over all pairs (excluding water-segment g_{ij}):

$$\text{MSSE} = \frac{1}{n_{\text{pairs}}} \sum_{ij \in \text{pairs}} \int_{r=0}^{r_{\max}} \mathrm{d}r \left(g_{ij}^{\text{DPD}}(r) - g_{ij}^{\text{MD}}(r) \right)^2 \tag{S26}$$

The following Figure **S5** depicts ~8000 evaluations (~5000 uniformly random bounded evaluations + ~3000 Bayesian interpretations) of MSSE sorted from higher to lower. MSSE appears to be quite sensitive to the input list of nonbonded coefficients and spans several orders of magnitude.

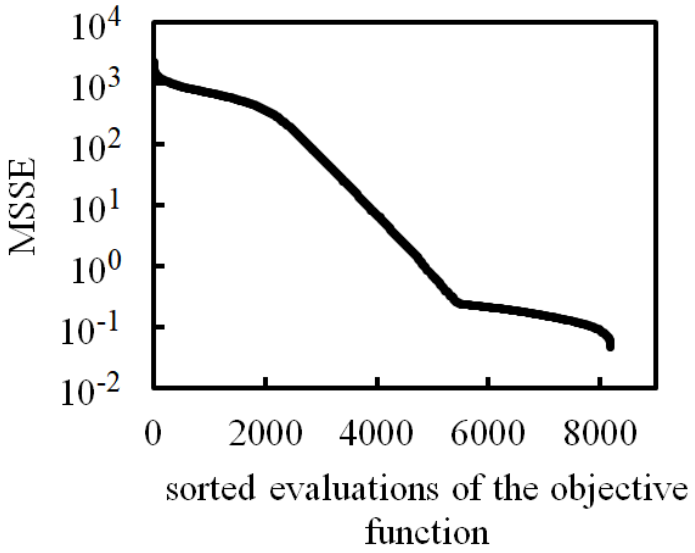

Figure S5. Evaluations of the objective function sorted from higher to lower values.

S8. Details about the Bayesian optimization

Bayesian optimization (BO) is a machine-learning-based optimization procedure which can be used to trace the global minimum/maximum of a (preferably) continuous objective function, $f(\mathbf{x})$, $x \forall \mathcal{R}^d$, such as the one described in section **S7**. BO offers distinct advantages as compared to conventional optimization algorithms:[9,10]

- It supports black box functions; i.e., functions with opaque implementations and unknown inner structure.
- It does not require access to the derivative of *f*; i.e., it is a derivative-free method.
- Efficient optimization of expensive objective functions (time- / resource-wise).
- It traces the global minimum/maximum by exploring regions far from local minima of *f*.
- It supports multi-dimensional objective functions; albeit the algorithm becomes inefficient when the number of dimensions becomes greater than 20.[9]
- It can be tuned to tolerate stochastic noise.

In practical terms, the algorithm performs "educated guesses" for where a minimum/maximum of the objective function exists, based on the previous function evaluations.

The main components of a Bayesian optimization algorithm are the following:

1. *Surrogate function*. Estimates the value and the uncertainty of the objective function across the parameter space.
2. *Acquisition function*. Interrogates the surrogate function in order to sample the next point to be evaluated by the objective function.

The surrogate function is modeled via Gaussian Process ($\mathcal{GP}$) regression and describes the objective function at each point $\mathbf{x} \forall \mathcal{R}^d$ by attributing a mean value $\mu(\mathbf{x})$ and a variance $\sigma^2(\mathbf{x})$. According to Ref 9 we can construct the surrogate function as follows:

Let there be a vector $[f(\mathbf{x}_1), f(\mathbf{x}_2), .., f(\mathbf{x}_n)]$ which contains the evaluations of the objective function at points $(\mathbf{x}_1, \mathbf{x}_2, .., \mathbf{x}_n)$; using a more compact notation we will refer to the aforementioned quantities as $f(\mathbf{x}_{1:n})$ and $\mathbf{x}_{1:n}$, respectively. Assuming that the evaluations $f(\mathbf{x}_{1:k})$ were drawn randomly from some *prior* probability distribution, $f(\mathbf{x}_{1:k})$ can be described by a $\mathcal{GP}$ as follows:

$$f(\mathbf{x}_{1:k}) \sim \mathcal{GP}\left(\mu_0(\mathbf{x}_{1:k}), K(\mathbf{x}_{1:k}, \mathbf{x}_{1:k})\right) \tag{S27}$$

with mean

$$\mu_0\left(\mathbf{x}_{1:k}\right)=\mathcal{E}\left[\,f\left(\mathbf{x}_{1:k}\right)\right] \tag{S28}$$

and covariance matrix

$$K\left(\mathbf{x}_{1:k},\mathbf{x}_{1:k}\right)=\mathcal{E}\left[\left(f\left(\mathbf{x}_{1:k}\right)-\mu_0\left(\mathbf{x}_{1:k}\right)\right)\left(f\left(\mathbf{x}_{1:k}\right)-\mu_0\left(\mathbf{x}_{1:k}\right)\right)\right] \tag{S29}$$

with $\mathcal{E}$ denoting the expectation function. Note that $K\left(\mathbf{x}_{1:k},\mathbf{x}_{1:k}\right)$ has been expressed in compact notation as well and describes the $k \times k$ matrix:

$$K\left(\mathbf{x}_{1:k},\mathbf{x}_{1:k}\right)=\left[K\left(\mathbf{x}_1,\mathbf{x}_1\right),\ldots,K\left(\mathbf{x}_1,\mathbf{x}_k\right);\ \ldots;K\left(\mathbf{x}_k,\mathbf{x}_1\right),\ldots,K\left(\mathbf{x}_k,\mathbf{x}_k\right)\right] \tag{S30}$$

The covariance is estimated by evaluating a *kernel* for each pair of observations. *Kernels* are positive semi-definite functions and indicate the correlations among neighboring observations across the parameter space; $K\left(\mathbf{x}_i,\mathbf{x}_j\right)$ increases/decreases between inputs at small/large distances, $x_{ij}=\left\|\mathbf{x}_j-\mathbf{x}_i\right\|$, across the input space.

Having defined a *prior* distribution based on the previous—supposedly noiseless—evaluations, we can perform a Bayesian inference at any point, $\mathbf{x}\forall\mathcal{R}^d$, using Bayes rule:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \tag{S31}$$

Following the derivations in Refs 9,10, the surrogate function (i.e., the Bayesian posterior probability distribution) can be estimated as the product, likelihood × prior, by the following $\mathcal{GP}$:[9,10]

$$f\left(\mathbf{x}\right)\middle|\, f\left(\mathbf{x}_{1,n}\right)\sim\mathcal{GP}\left(\mu_n\left(\mathbf{x}\right),\sigma_n^2\left(\mathbf{x}\right)\right) \tag{S32}$$

where

$$\begin{aligned}\mu_n\left(\mathbf{x}\right)&=K\left(\mathbf{x},\mathbf{x}_{1:n}\right)K\left(\mathbf{x}_{1:n},\mathbf{x}_{1:n}\right)^{-1}\left[\,f\left(\mathbf{x}_{1:n}\right)-\mu_0\left(\mathbf{x}_{1:n}\right)\right]+\mu_0\left(\mathbf{x}\right)\\ \sigma_n^2\left(\mathbf{x}\right)&=K\left(\mathbf{x},\mathbf{x}\right)-K\left(\mathbf{x},\mathbf{x}_{1:n}\right)K\left(\mathbf{x}_{1:n},\mathbf{x}_{1:n}\right)^{-1}K\left(\mathbf{x}_{1:n},\mathbf{x}\right)\end{aligned} \tag{S33}$$

A commonly used kernel is the Gaussian one:

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\sigma_{\mathrm{f}}{}^2\exp\left(-\frac{\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2}{2l^2}\right) \tag{S34}$$

with l being the length scale of the process, and σ_{f} controlling the overall variance. Another common kernel (which is used in the implementation by Nogueira[11]) is the Matérn kernel:

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\alpha_0\frac{2^{1-\nu}}{\Gamma\left(\nu\right)}\left(\sqrt{2\nu}\left\|\mathbf{x}_i-\mathbf{x}_j\right\|\right)^{\nu}K_{\nu}\left(\sqrt{2\nu}\left\|\mathbf{x}_i-\mathbf{x}_j\right\|\right) \tag{S35}$$

with Γ denoting the gamma function, K_ν being the modified Bessel function, and ν and α_0 being the parameters of the kernel.

There are several implementations of the acquisition function, each one characterized by an "exploration vs. exploitation tradeoff".[12] A common implementation for the acquisition function is the Expected Improvement (EI) scheme, which—as its name suggests—returns the point which yields the maximum expected improvement. The expected improvement at points lying in unexplored regions can be high, since the uncertainty is high in these regions as well. In addition, the expected improvement *near* quality points can be high as well; though the expected improvement right at points that have been already evaluated is zero, by definition (assuming we are dealing with a noise-free objective function). The above highlights the advantages of Bayesian optimization as compared to conventional optimization methods, in that it attempts to reach a global optimum by improving the quality of the solution while at the same time enhancing our knowledge about the objective function by favoring the exploration of unexplored regions.

The flowchart in **Figure S6** depicts the basic structure of a Bayesian optimization algorithm. During the initial warm-up phase of the optimization, the objective function is evaluated n_{init} times in a predefined domain (e.g., a grid across the $\mathcal{R}^d$ parameter space) or uniformly randomly across the imposed bounds. Subsequently, the algorithm enters the cycle of the Bayesian optimization procedure. At each step, the surrogate function is updated based on the previous evaluations of the objective function. Subsequently, the surrogate function (which is relatively cheap to evaluate) is interrogated by the acquisition function based on certain criteria depending on exploration vs exploitation tradeoffs[12] in order to retrieve the next point to be evaluated by the objective function. As soon as the maximum

iterations have been achieved the algorithm returns the maximizer of the objective function, $f(\mathbf{x}_{\max})$, or that of the posterior mean, $\mu_n(\mathbf{x}_{\max})$.

For a more detailed explanation about the Bayesian optimization algorithms the reader is referred to references 9,10. Further details regarding the implementation used in the present work can be found in ref 11.

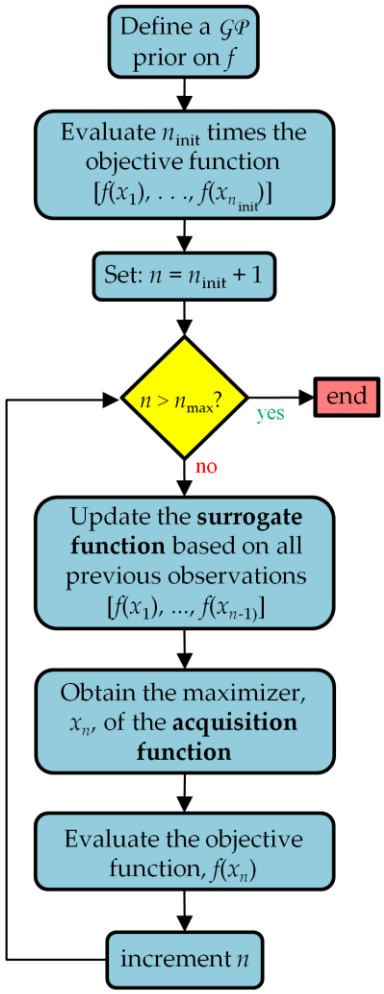

Figure S6. Flowchart of a basic Bayesian optimization algorithm

REFERENCES

(1) Shao, Q.; He, Y.; White, A. D.; Jiang, S. Difference in Hydration between Carboxybetaine and Sulfobetaine. *J. Phys. Chem. B* **2010**, *114*, 16625–16631.

(2) Yamanaka, T.; De Nicola, A.; Munaò, G.; Soares, T. A.; Milano, G. Effect of the Ligand's Bulkiness on the Shape of Functionalized Gold Nanoparticles in Aqueous Solutions: A Molecular Dynamics Study. *Chem. Phys. Lett.* **2019**, *731*, 136576.

(3) Klein, C.; Summers, A. Z.; Thompson, M. W.; Gilmer, J. B.; McCabe, C.; Cummings, P. T.; Sallai, J.; Iacovella, C. R. Formalizing Atom-Typing and the Dissemination of Force Fields with Foyer. *Comput. Mater. Sci.* **2019**, *167*, 215–227.

(4) Ryckaert, J.-P.; Bellemans, A. Molecular Dynamics of Liquid Alkanes. *Faraday Discuss. Chem. Soc.* **1978**, *66*, 95–106.

(5) Lindahl; Abraham; Hess; Spoel, van der. *GROMACS 2021.1 Manual Https://Doi.Org/10.5281/Zenodo.5053220*; 2021.

(6) Mukundan, R. Quaternions : From Classical Mechanics to Computer Graphics, and Beyond. In *Proceedings of the 7th Asian Technology Conference in Mathematics 2002.*; 2002; pp 97–106.

(7) Kuznetsov, D. V.; Chen, Z. Y. Semiflexible Polymer Brushes: A Scaling Theory. *J. Chem. Phys.* **1998**, *109*, 7017–7027.

(8) Zhao, B.; Brittain, W. J. Polymer Brushes: Surface-Immobilized Macromolecules. *Prog. Polym. Sci.* **2000**, *25*, 677–710.

(9) Frazier, P. A Tutorial on Bayesian Optimization. *arXiv:1807.02811* **2018**, *abs/1807.0*.

(10) Rasmussen, C. E.; Williams, K. I. *Gaussian Processes for Machine Learning*; the MIT Press: Massachusetts Institute of Technology, 2006.

(11) Nogueira, F. *Bayesian Optimization: Open Source Constrained Global Optimization Tool for Python*; https://github.com/fmfn/BayesianOptimization, 2014.

(12) Kaelbling, L. P.; Littman, M. L.; Moore, A. W. Reinforcement Learning: A Survey. *J. Artif. Int. Res.* **1996**, *4*, 237–285.