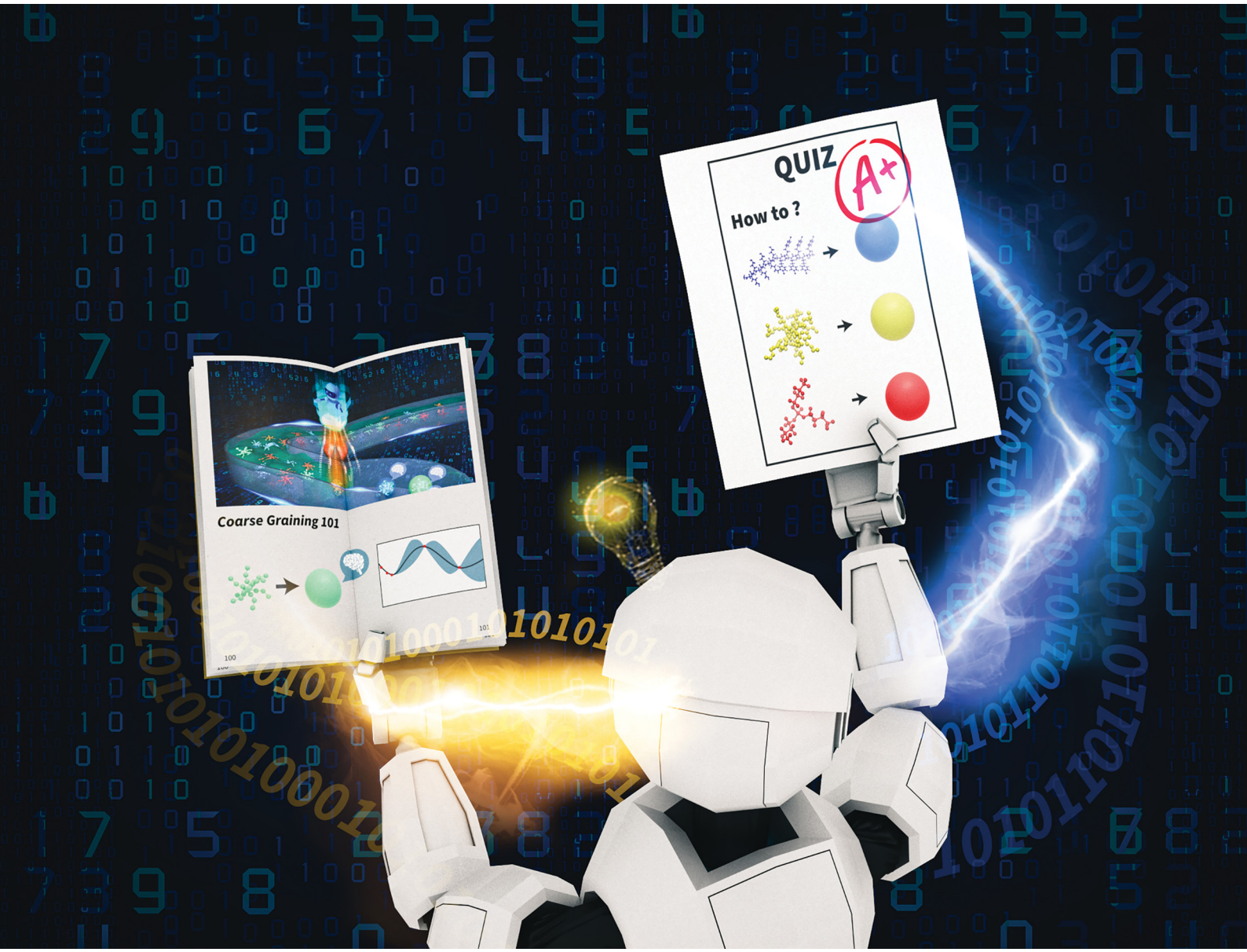
Volume 17
Number 24
28 June 2021
Pages 5841–6046

Soft Matter

rsc.li/soft-matter-journal



ISSN 1744-6848

ROYAL SOCIETY OF CHEMISTRY

PAPER
Wenxiao Pan *et al.*
Transfer learning of memory kernels for transferable coarse-graining of polymer dynamics

Soft Matter

PAPER

View Article Online
View Journal | View Issue

Cite this: *Soft Matter*, 2021, **17**, 5864

Transfer learning of memory kernels for transferable coarse-graining of polymer dynamics

Zhan Ma, [a] Shu Wang, [a] Minhee Kim, [b] Kaibo Liu, [b] Chun-Long Chen [c] and Wenxiao Pan *[a]

Received 9th March 2021,
Accepted 7th May 2021

DOI: 10.1039/d1sm00364j

rsc.li/soft-matter-journal

The present work concerns the transferability of coarse-grained (CG) modeling in reproducing the dynamic properties of the reference atomistic systems across a range of parameters. In particular, we focus on implicit-solvent CG modeling of polymer solutions. The CG model is based on the generalized Langevin equation, where the memory kernel plays the critical role in determining the dynamics in all time scales. Thus, we propose methods for transfer learning of memory kernels. The key ingredient of our methods is Gaussian process regression. By integration with the model order reduction *via* proper orthogonal decomposition and the active learning technique, the transfer learning can be practically efficient and requires minimum training data. Through two example polymer solution systems, we demonstrate the accuracy and efficiency of the proposed transfer learning methods in the construction of transferable memory kernels. The transferability allows for out-of-sample predictions, even in the extrapolated domain of parameters. Built on the transferable memory kernels, the CG models can reproduce the dynamic properties of polymers in all time scales at different thermodynamic conditions (such as temperature and solvent viscosity) and for different systems with varying concentrations and lengths of polymers.

1 Introduction

To study polymers or biomolecules in solution, coarse-grained (CG) modeling and simulations can be practically more efficient[1–8] compared with full atomistic simulations *via*, *e.g.*, all-atom molecular dynamics (MD). Instead of tracking individual atoms of molecules and solvent, the CG modeling averages out or eliminates certain degrees of freedom (DOFs) to reduce the system's dimensionality and captures the molecules' collective dynamics and properties. The removal of highly-fluctuating atomic DOFs and the larger characteristic length scale of CG coordinates allow us to employ larger time steps in CG simulations. In implicit-solvent CG modeling,[9–14] not only are the DOFs representing polymer molecules reduced but also the solvent DOFs are eliminated. Providing significantly reduced DOFs and larger time steps, CG modeling is computationally more efficient than full atomistic simulations and hence can grant larger accessible length scales and render tractable simulation of long-time effects in practical applications.[15–18]

However, to reap the benefit of CG modeling, two challenges must be addressed. The first one is to conserve both the structural and dynamic properties of polymers under coarse-graining. To conserve the structural properties (*e.g.*, radial and angular distribution functions), the CG potential (or potential of mean force) must be correctly constructed.[19–26] To conserve the dynamic properties (*e.g.*, diffusivity and the velocity auto-correlation function (VACF)), the kinetic effect of unresolved DOFs (including solvent) on the system must be properly accounted for in CG modeling. For that, a non-Markovian dynamics (*e.g.*, in the form of a generalized Langevin equation (GLE)) must be introduced in the CG model because the elimination of DOFs results in a non-Markovian memory in the dynamics of CG variables, as discussed in the literature[27–32] and also in our prior work.[13,14] The second challenge lies in the transferability of CG modeling, for instance, how the CG model constructed can be transferable across different thermodynamic conditions. Efforts to attain transferable CG potentials that preserve structural properties under coarse-graining have made substantial progress.[33–37] In contrast, the transferability of a CG model in conserving the dynamic properties has not been extensively discussed. Lyubimov *et al.* derived from the GLE an analytical factor for dynamical rescaling of the friction coefficient in CG modeling to correctly capture the long-time diffusion of polymers. The derived rescaling factor is transferable for different polymer systems and thermodynamic conditions,

[a] *Department of Mechanical Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA. E-mail: wpan9@wisc.edu*
[b] *Department of Industrial and Systems Engineering, University of Wisconsin-Madison, Madison, WI 53706, USA*
[c] *Physical Sciences Division, Pacific Northwest National Laboratory, Richland, WA 99352, USA*

with the temperature and radius-of-gyration as the input parameters.[38,39] However, the approach is only applicable to polymer melts and reproducing the long-time diffusion coefficient of polymers.[38,39] Other valuable attempts for polymer melts have employed the energy renormalization method.[40–42] By varying the cohesive interaction strength with a temperature-dependent factor in the CG models, the dynamic properties of polymers related to configurational entropy can be preserved over a wide temperature range. In the work by Dunbar and Keten,[42] this method has also been shown to be capable of capturing the effect of chemistry specificity and preserving mechanical properties. However, for polymers in solution and to reproduce the dynamic properties beyond the long-time normal diffusion, *e.g.*, super- and/or sub-diffusion and the VACF as a function of time, a new methodology to render transferability in CG modeling is required. Noting that the memory kernel plays a critical role for a CG model to reproduce the entire dynamics,[27,28,43,44] especially in implicit-solvent CG modeling with many solvent DOFs unresolved,[13,14,29,30] the present work hence focuses on the transferability of the memory kernel in CG modeling.

In particular, we propose two transfer learning methods to enable memory kernel transfer across different thermodynamic conditions (such as temperature and solvent viscosity) and across different systems with varying solute concentrations and lengths of polymer chains. The proposed transfer learning methods draw on Gaussian process regression (GPR). By integration with the model order reduction *via* proper orthogonal decomposition (POD) and the active learning technique, the transfer learning can be practically efficient and requires minimum training data. GPR is chosen because of its flexibility to represent nonlinear relationships between multi-dimensional outputs and inputs and its capability to quantify the uncertainty of prediction in both interpolation and extrapolation. The model order reduction enables the representation of the memory kernel (as a function of time and parameters) in a reduced temporal and parameter space, which in turn greatly reduces the training and prediction costs of GPR. The active learning technique allows for the efficient use of data with maximum information gain *via* adaptive sampling guided by the uncertainty quantified in GPR. Through two example polymer solution systems, we demonstrate the accuracy and efficiency of the proposed transfer learning methods in the construction of transferable memory kernels, from which the CG models can reproduce the dynamic properties of polymers across different thermodynamic conditions and systems.

The rest of the paper is organized as follows. In Section 2.1, we describe the framework of CG modeling, which is built upon the GLE and extended dynamics. Section 2.2 explains in detail the proposed transfer learning methods, whose key ingredients include GPR, POD, and active learning. We present the numerical results in Section 3, where two benchmark examples are used to assess the accuracy and computational cost of transfer learning. Finally, we conclude and summarize our main findings and contributions in Section 4.

2 Methodology

2.1 CG modeling

Without loss of generality, we consider an atomistic system consisting of n atoms in polymer molecules, with coordinates $\mathbf{r} = \{\mathbf{r}_i | i = 1,2,\ldots,n\}$ and momenta $\mathbf{p} = \{\mathbf{p}_i | i = 1,2,\ldots,n\}$. In CG modeling, n atoms are coarse-grained as N clusters (referred to as CG particles), and each cluster contains n_c atoms. To be consistent in notation, we use the lowercase m_i, $\mathbf{r}_i$, and $\mathbf{p}_i$ to represent the mass, position, and momentum of the i-th atom in the atomistic system; and the uppercase M_I, $\mathbf{R}_I$, and $\mathbf{P}_I$ denote the mass, position, and momentum of the I-th CG particle in the CG system. The variables of the atomistic and CG systems are related *via*:

$$M_I = \sum_{i=1}^{n_c} m_{Ii}, \quad \mathbf{R}_I = \frac{1}{M_I}\sum_{i=1}^{n_c} m_{Ii}\mathbf{r}_{Ii}, \quad \mathbf{P}_I = \sum_{i=1}^{n_c} \mathbf{p}_{Ii}, \tag{1}$$

where the subscript Ii denotes the i-th atom in the I-th CG particle; M_I, $\mathbf{R}_I$, and $\mathbf{P}_I$ are defined as the total mass, center-of-mass (COM) position, and total momentum of all atoms in the I-th CG particle, respectively.

The GLE given by the Mori–Zwanzig projection formalism[45–47] provides a theoretically sound framework for CG modeling, where the non-Markovian dynamics of the CG system are governed by:

$$\dot{\mathbf{P}}_I = \langle \mathbf{F}_I \rangle - \int_0^t K(t-t') M_I^{-1} \mathbf{P}_I(t') \mathrm{d}t' + \tilde{\mathbf{F}}_I. \tag{2}$$

On the right-hand side of eqn (2), the first term represents the mean force on the I-th CG particle. The third term $\tilde{\mathbf{F}}_I$ denotes the fluctuating force. The second term (referred to as the dissipative force) has a memory kernel $K(t-t')$, which is related to the fluctuating force by $K(t) = (1/k_B T)\ \langle[\tilde{\mathbf{F}}_I(t)][\tilde{\mathbf{F}}_I(0)]\rangle$ to satisfy the second fluctuation–dissipation theorem.[48] Here, k_B denotes the Boltzmann constant; T is the thermodynamic temperature. The kinetic effect of the lost atomic DOFs under coarse-graining is properly accounted in the CG dynamics by the memory kernel and the fluctuating force with colored noise in eqn (2).[46,47]

The mean force in eqn (2) is $\langle \mathbf{F}_I \rangle = \dfrac{1}{k_B T}\dfrac{\partial}{\partial \mathbf{R}_I} \ln \omega(\mathbf{R})$, where $\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_N\}$ is a point in the CG phase space; $\omega(\mathbf{R})$ represents a normalized partition function of all the atomistic configurations at phase point $\mathbf{R}$. Because the present work concerns the dynamic properties and does not consider the structural properties or free energy, $\langle \mathbf{F}_I \rangle$ is regarded as the average for the I-th CG particle over all phase points. Thus, without external force fields, the mean force exerted on a CG particle is approximated to be zero; *i.e.*, $\langle \mathbf{F}_I \rangle = 0$. Eqn (2) can hence be simplified to:

$$\dot{\mathbf{P}}_I(t) = -\int_0^t K(t-t') \mathbf{V}_I(t') \mathrm{d}t' + \tilde{\mathbf{F}}_I(t), \tag{3}$$

with the velocity $\mathbf{V}_I(t') = \dfrac{\mathbf{P}_I(t')}{M_I}$.

To conserve the dynamics and reproduce the dynamic properties of the underlying atomistic system, the memory kernel in eqn (3) must be directly linked to the atomistic system and can be computed from the atomistic data.

2.1.1 Memory kernel. To determine the memory kernel $K(t)$ in eqn (3) from the atomistic data, we rely on the property that the velocity $\mathbf{V}$ and fluctuating force $\tilde{\mathbf{F}}$ come from two orthogonal subspaces and hence are not correlated to each other, *i.e.*, $\langle \mathbf{V}^T\tilde{\mathbf{F}}\rangle = 0$. Thus, multiplying both sides of eqn (3) by $\mathbf{V}(0)^T$ leads to:

$$\langle \mathbf{V}(0)^T\dot{\mathbf{P}}(t)\rangle = -\int_0^t K(t-t')\langle \mathbf{V}(0)^T\mathbf{V}(t')\rangle \mathrm{d}t', \qquad (4)$$

where $C(t) = \langle \mathbf{V}(0)^T\mathbf{V}(t')\rangle$ is the VACF; $W(t) = \langle \mathbf{V}(0)^T\dot{\mathbf{P}}(t)\rangle$ defines the force–velocity correlation function (FVCF). While the data of the VACF can be directly obtained in the atomistic simulations, the FVCF can be evaluated from $W(t) = \dfrac{\mathrm{d}C(t)}{\mathrm{d}t}$ using numerical differentiation. The memory kernel $K(t)$ can then be solved *via* deconvolution of eqn (4) given the data of the VACF and FVCF. Note that the integral operator in eqn (4) is the Volterra operator.[49] Although the deconvolution can have a unique continuous solution, the solution does not depend continuously on the data; *i.e.*, the solution is unstable against data noise due to the nondegeneracy of $C(t)$. Thus, solving the deconvolution to determine the memory kernel is an ill-posed problem, and proper regularization must be enforced. What we have are the data of the VACF and FVCF at discrete times (*i.e.*, $t_i = (i-1)\Delta t$ with $i = 1,2,\ldots,N_t$). Hence, eqn (4) can be discretized, and the deconvolution is solved in the discrete setting. Note that discretization can regularize an ill-posed problem, known as the "self-regularization" property of discretization.[50] As a result, the deconvolution problem becomes well-posed when solved in the discrete setting. However, the linear system resulting from discretization could be ill-conditioned[51] and requires further regularization. Applying the midpoint quadrature rule,[49] eqn (4) can be discretized into the following linear system:

$$\mathbf{CK} = -\mathbf{W}, \qquad (5)$$

where $\mathbf{C} \in \mathbb{R}^{(N_t-1)\times(N_t-1)}$ with

$$\mathbf{C}_{i,j} = \begin{cases} \dfrac{\Delta t}{2}(C(t_{i-j+1}) + C(t_{i-j+2})) & i \geq j \\ 0 & i < j \end{cases};$$

$\mathbf{K} \in \mathbb{R}^{N_t-1}$ with $\mathbf{K}_i = K(t_{i+1/2})$; and $\mathbf{W} \in \mathbb{R}^{N_t-1}$ with $\mathbf{W}_i = W(t_{i+1})$. If the linear system in eqn (5) is ill-conditioned, which means its solution is unstable and sensitive to data noise, the Tikhonov regularization[52,53] is introduced and leads to the following regularized linear system:

$$(\mathbf{C}^T\mathbf{C} + \beta)\mathbf{K} = -\mathbf{C}^T\mathbf{W}, \qquad (6)$$

where $\mathbf{C}^T$ is the transpose of $\mathbf{C}$, and β is the regularization parameter. The value of β can be determined using the quasi-optimality criterion.[54,55] In addition to the deconvolution, the numerical differentiation used to obtain the data of the FVCF is also an ill-posed problem and requires regularization.

For the numerical examples considered in this work, the FVCF ($W(t)$) was obtained *via* numerical differentiation regularized by the Tikhonov regularization following the quasi optimality principle; the linear system in eqn (5) was well-conditioned and hence directly solved without regularization.

2.1.2 Extended dynamics. Given the memory kernel $K(t)$ determined, the GLE in eqn (3) can be solved to predict the dynamics of the CG system. However, directly solving this equation requires evaluation of the time convolution of the memory kernel and velocity and generation color noise for the fluctuating force, which needs to store the historical information and can be prohibitively expensive. Note that the solvent-mediated kinetics could result in a long-tailed memory kernel, making the computation even more expensive. To address the challenge of directly solving the GLE, $K(t)$ is first approximated by an asymptotic expansion as:

$$K(t) \approx \sum_{l=1}^{\mathcal{N}} \exp\left(-\frac{a_l}{2}t\right)[b_l\cos(q_l t) + c_l\sin(q_l t)], \qquad (7)$$

where the parameters $\{a_l,b_l,c_l,q_l\}_{l=1,\ldots,\mathcal{N}}$ can be determined *via* fitting.[13,27] Truncating the expansion with more terms (larger $\mathcal{N}$) leads to a more accurate approximation of $K(t)$. Approximating the memory kernel by a finite set of exponentially damped oscillators as in eqn (7) would allow the GLE to be replaced with Markovian dynamics extended in higher dimensions. By doing so, the expensive cost of solving the GLE can be significantly reduced, as has been evidenced in the literature.[13,27,56] To this end, eqn (7) is rewritten in a matrix form as:

$$K(t) \approx -\mathbf{A}_{ps}\mathrm{e}^{-t\mathbf{A}_{ss}}\mathbf{A}_{sp}, \qquad (8)$$

where $\mathbf{A}_{ps} = -\mathbf{A}_{sp}^T$. If we define the parameter matrix $\mathbf{A} = [0, \mathbf{A}_{ps}; \mathbf{A}_{sp}, \mathbf{A}_{ss}]$, it can be assembled from the parameters in eqn (7) by:

$$\mathbf{A}_l = \begin{bmatrix} 0 & \sqrt{\dfrac{b_l}{2} - \dfrac{q_l c_l}{a_l}} & \sqrt{\dfrac{b_l}{2} + \dfrac{q_l c_l}{a_l}} \\ -\sqrt{\dfrac{b_l}{2} - \dfrac{q_l c_l}{a_l}} & a_l & \dfrac{1}{2}\sqrt{4q_l^2 + a_l^2} \\ -\sqrt{\dfrac{b_l}{2} + \dfrac{q_l c_l}{a_l}} & -\dfrac{1}{2}\sqrt{4q_l^2 + a_l^2} & 0 \end{bmatrix}. \qquad (9)$$

In eqn (9), the top right block contributes to $\mathbf{A}_{ps}$; the bottom left contributes to $\mathbf{A}_{sp}$; and the block on the bottom right constitutes $\mathbf{A}_{ss}$, which is a block diagonal matrix consisting of 2×2 blocks.

Given eqn (8) and by introducing auxiliary variables $\mathbf{S}$, the extended Markovian dynamics is given by:

$$\begin{pmatrix} \dot{\mathbf{P}} \\ \dot{\mathbf{S}} \end{pmatrix} = -\begin{pmatrix} 0 & \mathbf{A}_{\mathrm{ps}} \\ \mathbf{A}_{\mathrm{sp}} & \mathbf{A}_{\mathrm{ss}} \end{pmatrix}\begin{pmatrix} M^{-1}\mathbf{P} \\ \mathbf{S} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & \mathbf{B}_{\mathrm{s}} \end{pmatrix}\begin{pmatrix} 0 \\ \boldsymbol{\xi} \end{pmatrix}. \qquad (10)$$

Here, $\boldsymbol{\xi}$ is a vector of uncorrelated Gaussian random variables with $\langle \boldsymbol{\xi}(t)\rangle = \mathbf{0}$ and $\langle \xi_{I,\mu}(t)\xi_{J,\nu}(0)\rangle = \delta_{IJ}\delta_{\mu\nu}\delta(t)$, where ξ_ν and ξ_μ denote the different elements of $\boldsymbol{\xi}$. To satisfy the second fluctuation–dissipation theorem,[48] $\mathbf{B}_s\mathbf{B}_s^T = k_{\mathrm{B}}T(\mathbf{A}_{ss} + \mathbf{A}_{ss}^T)$. We can write the parameter matrix $\mathbf{B} = \mathrm{diag}(0,\mathbf{B}_s)$. To ensure $\mathbf{A}$ and $\mathbf{B}$ are both real number matrices, the parameters in eqn (7) need to satisfy: $a_l \geq 0$, $b_l \geq 0$, and $|c_l| \leq \dfrac{a_l b_l}{2q_l}$.

The extended dynamics in eqn (10) are equivalent to the GLE in eqn (3)[56] with the fluctuating force:

$$\tilde{\mathbf{F}}(t) = -\int_0^t \mathbf{A}_{\mathrm{ps}} \mathrm{e}^{-(t-t')\mathbf{A}_{\mathrm{ss}}} \mathbf{B}_{\mathrm{s}} \boldsymbol{\xi}(t') \mathrm{d}t'. \tag{11}$$

The extended dynamics in eqn (10) circumvent the expensive time-convolution, only needs to sample white noise, and hence can be solved more efficiently than the GLE. In the present work, the implicit velocity-Verlet temporal integrator[57] was used to numerically solve eqn (10) in the CG simulations.

2.2 Transfer learning of the memory kernel

In this section, we explain the methodology proposed for transfer learning of the memory kernel, which enables the CG modeling to be transferable across a range of parameters to reproduce the dynamic properties of the underlying atomistic systems. The proposed methodology draws on the GPR and can be significantly accelerated by the model order reduction and active learning techniques.

2.2.1 Transfer learning based on GPR. The memory kernel is denoted as $K(t;\boldsymbol{\mu})$, where the vector $\boldsymbol{\mu}$ represents the parameters of interest such as temperature, concentration, and solvent viscosity; and $(t,\boldsymbol{\mu}) \in \mathscr{T} \times \mathscr{P}$ with $\mathscr{T} = [0,t_{\mathrm{f}}]$ and $\mathscr{P} \in \mathbb{R}^d$ representing the time domain and parameter space, respectively, where d is the dimension of the parameter space. Our goal is to predict the memory kernel $K(t;\boldsymbol{\mu}^\star)$ at any given parameter instance $\boldsymbol{\mu}^\star \in \mathscr{P}$ by using the data of $K(t;\boldsymbol{\mu})$ at N_p training parameter instances. To achieve this goal, we define $\mathbf{x} = [t,\boldsymbol{\mu}] \in \mathbb{R}^{d+1}$ and $y = K(\mathbf{x}) \in \mathbb{R}$ as the input and output of GPR, respectively. Recalling that the data of the memory kernel are obtained at N_t discrete times (Section 2.1.1), we have a total of $N_{\mathrm{train}} = N_t \times N_p$ training data for GPR. The dependence of the outputs (memory kernels) $\mathbf{Y} = [y_1,y_2,\ldots,y_{N_{\mathrm{train}}}]^T \in \mathbb{R}^{N_{\mathrm{train}}}$ on the inputs (time and parameters) $\mathbf{X} = [\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_{N_{\mathrm{train}}}]^T \in \mathbb{R}^{N_{\mathrm{train}}\times(d+1)}$ is modeled as a Gaussian process: $\mathbf{Y}(\mathbf{X}) \sim \mathscr{GP}(\mathbf{u}(\mathbf{X}),\boldsymbol{\Sigma}(\mathbf{X},\mathbf{X}))$. Here, $\mathbf{u}(\mathbf{X}) \in \mathbb{R}^{N_{\mathrm{train}}}$ is the mean function, and $\boldsymbol{\Sigma}(\mathbf{X},\mathbf{X}) \in \mathbb{R}^{N_{\mathrm{train}}\times N_{\mathrm{train}}}$ is the covariance matrix. In this work, the covariance function is assumed to be a squared exponential form, *i.e.*,

$$\boldsymbol{\Sigma}_{ij}(\mathbf{X},\mathbf{X};\mathbf{q}) = {\theta_{\mathrm{f}}}^2 \exp\left[-\frac{1}{2}\sum_{k=1}^{d+1} {\theta_{l_k}}^2 (x_{i,k} - x_{j,k})^2\right], \tag{12}$$

where $x_{i,k}$ and $x_{j,k}$ refer to the k-th element of $\mathbf{x}_i$ and $\mathbf{x}_j$, respectively; $\boldsymbol{\theta} = (\theta_{\mathrm{f}},\theta_{l_1},\theta_{l_2},\ldots,\theta_{l_{d+1}})$ denotes the hyper-parameters. Using the GPR model inferred from the training data, we can predict $K(t;\boldsymbol{\mu}^\star)$ at any given $\boldsymbol{\mu}^\star$. For that, the inputs are $\mathbf{X}^\star = [\mathbf{x}_1{}^\star,\mathbf{x}_2{}^\star,\ldots,\mathbf{x}_{N_t}{}^\star]$ with $\mathbf{x}_i{}^\star = [t_i,\boldsymbol{\mu}^\star]$; the outputs are $\mathbf{Y}^\star = [K(t_1;\boldsymbol{\mu}^\star),K(t_2;\boldsymbol{\mu}^\star),\ldots,K(t_{N_t};\boldsymbol{\mu}^\star)]^T$ and satisfy:

$$\mathbf{Y}^\star|\mathbf{Y} \sim \mathscr{GP}(\hat{\mathbf{u}},\hat{\boldsymbol{\Sigma}}), \tag{13}$$

where

$$\begin{aligned} \hat{\mathbf{u}} &= \boldsymbol{\Sigma}(\mathbf{X}^*,\mathbf{X})\left[\boldsymbol{\Sigma}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}\right]^{-1}[\mathbf{Y} - \mathbf{u}(\mathbf{X})] + \mathbf{u}(\mathbf{X}^*), \\ \hat{\boldsymbol{\Sigma}} &= \boldsymbol{\Sigma}(\mathbf{X}^*,\mathbf{X}^*) - \boldsymbol{\Sigma}(\mathbf{X}^*,\mathbf{X})\left[\boldsymbol{\Sigma}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}\right]^{-1}\boldsymbol{\Sigma}(\mathbf{X}^*,\mathbf{X})^T, \end{aligned} \tag{14}$$

and σ^2 is the variance of identically independent Gaussian noise (with zero mean) assumed in the GPR model. In eqn (14), $\hat{\mathbf{u}}$ is the mean value of the prediction at $\boldsymbol{\mu}^\star$; the diagonal elements of $\hat{\boldsymbol{\Sigma}}$ are the analytical uncertainty bounds, denoted as $\hat{\sigma}$. To determine σ^2 and the hyper-parameters $\boldsymbol{\theta}$, we minimize the negative log marginal likelihood:[58]

$$-\log p(\mathbf{Y}|\theta,\sigma^2) = \frac{1}{2}\mathbf{Y}^T\mathscr{C}^{-1}\mathbf{Y} + \frac{1}{2}\log|\mathscr{C}| + \frac{N_{\mathrm{train}}}{2}\log(2\pi), \tag{15}$$

using the Quasi-Newton optimizer L-BFGS,[59] where $\mathscr{C} = \boldsymbol{\Sigma}(\mathbf{X},\mathbf{X}) + \sigma^2\mathbf{I}$. Once σ^2 and the hyper-parameters are determined, the GPR model is complete and can be used to predict the memory kernel at any parameter instance of interest along with the uncertainty quantified for the prediction.

2.2.2 Transfer learning accelerated by model order reduction. The training cost of GPR exhibits a cubic scaling with respect to the number of training data, *i.e.* $\mathscr{O}({N_{\mathrm{train}}}^3)$. In practice, $N_{\mathrm{train}} = N_p \times N_t$ can be a large number due to the large N_t. As we have discussed above, the dynamics of a polymer in solution can display a long-tailed memory kernel. To fully capture the memory kernel's variations in all time scales, the data must be collected at many discrete times, resulting in a large N_t. In the numerical examples considered in this work, N_t is larger than N_p (the number of training parameter instances) by 2 or 3 orders. Hence, if the memory kernel is directly modeled by GPR as discussed in Section 2.2.1, the training cost can be expensive due to a large N_{train}. Thus, we further propose a strategy to greatly accelerate the transfer learning by combining the GPR with model order reduction. In particular, the memory kernel is first decomposed using POD into temporal and parameter modes, from which only the dominant modes are retained. GPR is then used to model the dominant parameter modes only.

As in Section 2.2.1, we need snapshot data of $K(t;\boldsymbol{\mu}_i)$ at $\boldsymbol{\mu}_i$ with $i = 1,2,\ldots,N_p$. A reduced order model (ROM) is established for the memory kernel by decomposing the snapshot data of $K(t;\boldsymbol{\mu}_i)$ as:

$$K(t;\boldsymbol{\mu}_i) = \bar{K}(t) + \hat{K}(t;\boldsymbol{\mu}_i) = \bar{K}(t) + \sum_{k=1}^{N_{\mathrm{p}}} \alpha_k(\boldsymbol{\mu})\phi_k(t), \tag{16}$$

where $\bar{K}(t) = \frac{1}{N_p}\sum_{i=1}^{N_p} K(t;\boldsymbol{\mu}_i)$ denotes the mean of all snapshots; the fluctuating part $\hat{K}$ is decomposed into the temporal bases $\phi_k(t)$ and the parameter modes $\alpha_k(\boldsymbol{\mu})$. Note that in the classical model order reduction for dynamical systems, the POD basis functions are spatial bases.[60] Here, we adapt the technique to our needs and replace the spatial bases with temporal bases. According to POD, the basis functions can be obtained by eigendecomposition of the correlation matrix $\mathbf{G} \in \mathbb{R}^{N_p\times N_p}$ of the fluctuating parts:

$$G_{ij} = \int_0^{t_{\mathrm{f}}} \hat{K}(t;\mu_i)\hat{K}(t;\mu_j)\mathrm{d}t, \tag{17}$$

where i and j refer to the i-th and j-th snapshots, respectively. Then, the temporal bases $\phi_k(t)$ (*i.e.*, POD basis functions) are given as:

$$\phi_k(t) = \frac{1}{\sqrt{\lambda_k}}\sum_{i=1}^{N_p} w_i^k \hat{K}(t;\mu_i), \tag{18}$$

where $\{\lambda_1,\lambda_2,\ldots,\lambda_M\}$ are the eigenvalues of $\mathbf{G}$ in descending order; w_i^k is the i-th component of $\mathbf{w}^k$, the eigenvector corresponding to the eigenvalue λ_k. The idea of POD is that the energy contributed by each basis can be reflected by its corresponding eigenvalue. Thus, if

the first $R \ll N_p$ eigenvalues are significantly larger than those remaining, we only need to retain the first R modes because they dominate the energy, *i.e.*,

$$\hat{K}(t;\mu_i) = \sum_{k=1}^{N_p} \alpha_k(\mu)\phi_k(t) \approx \sum_{k=1}^{R} \alpha_k(\mu_i)\phi_k(t) = \sum_{k=1}^{R} \sqrt{\lambda_k} w_k^i \phi_k(t), \tag{19}$$

where $\alpha_k(\mu_i) = \sqrt{\lambda_k} w_k^i$. The approximation (truncation) error of eqn (19) is $\varepsilon^{\mathrm{POD}} = \sqrt{\sum_{k=R+1}^{N_p} \lambda_k}$, which can be reduced by increasing R, *i.e.*, by keeping more POD modes. Based on it, we define the relative error of POD as:

$$\varepsilon_{\mathrm{r}}^{\mathrm{POD}} = \sqrt{\frac{\sum_{k=R+1}^{N_p} \lambda_k}{\sum_{k=1}^{N_p} \lambda_k}}. \tag{20}$$

For a target tolerance ζ_{POD}, by requiring $\varepsilon_{\mathrm{r}}^{\mathrm{POD}} \leq \zeta_{\mathrm{POD}}$, we can determine how many dominant modes to retain in the ROM, *i.e.*, the value of R.

Given eqn (19), we can establish the ROM for the memory kernel as:

$$K(t;\mu^*) \approx K^{\mathrm{ROM}}(t;\mu^*) = \bar{K}(t) + \sum_{k=1}^{R} \alpha_k(\mu^*)\phi_k(t), \tag{21}$$

by which the memory kernel can be effectively predicted at a given parameter instance $\boldsymbol{\mu}^*$ by the reduced temporal bases $\phi_k(t)$ and parameter modes $\alpha_k(\boldsymbol{\mu}^*)$. Thus, we only need to train and predict $\alpha_k(\boldsymbol{\mu}^*)$ instead of the entire memory kernel function $K(t;\boldsymbol{\mu}^*)$, resulting in greatly reduced training and prediction costs. Following the idea of data-driven nonintrusive reduced order modeling,[60] we still use GPR to model each parameter mode $\alpha_k(\boldsymbol{\mu})$, *i.e.*, $\alpha_k(\boldsymbol{\mu}) = y(\mathbf{x}) \sim \mathcal{GP}(u_k(\mathbf{x}),\boldsymbol{\Sigma}_k(\mathbf{x},\mathbf{x}^*))$, where the input and output of GPR become $\mathbf{x} = \boldsymbol{\mu} \in \mathbb{R}^d$ and $y = \alpha_k(\boldsymbol{\mu})$, respectively. Now the number of training data required in GPR is $N_{\mathrm{train}} = N_p$, which is much smaller than $N_{\mathrm{train}} = N_p \times N_t$ in Section 2.2.1. Once the GPR model of each parameter mode is constructed, we can predict $\alpha_k(\boldsymbol{\mu}^*)$ and in turn the memory kernel by eqn (21) at any given parameter instance $\boldsymbol{\mu}^*$. In addition, the uncertainty for the predicted $\alpha_k(\boldsymbol{\mu}^*)$ can be quantified by the standard deviation $\hat{\sigma}_k(\boldsymbol{\mu}^*)$ in the GPR model.

2.2.3 Active learning with minimum training data. To ensure the accuracy of the inferred GPR models, sufficient training data are required. However, attaining the data of memory kernels requires nontrivial efforts. Thus, minimizing the number of data required for training the GPR models becomes necessary. To this end, we propose an active learning strategy to adaptively sample the training data and to ensure maximum information gain from the data sampled. The key idea is to take advantage of the fact that the GPR can predict not only the mean $\hat{u}$ but also the standard deviation $\hat{\sigma}$ to quantify the uncertainty. The knowledge of uncertainty can guide the sampling of the next data. In particular, the next parameter instance $\boldsymbol{\mu}_{\mathrm{next}}$ to be sampled is where the uncertainty level reaches a maximum over the sampling space. In this work, active learning is only integrated with the transfer learning accelerated by model order reduction (Section 2.2.2) but not with the transfer learning based solely on GPR (Section 2.2.1). This is because in Section 2.2.1 the computational cost for updating the GPR model after each new sampling is significant, which can overshadow the benefit of active learning. The proposed procedure is described below.

Let $\mathscr{P}_{\mathrm{tr}} \in \mathbb{R}^d$ denote the training sampling space and note $\mathscr{P}_{\mathrm{tr}} \subseteq \mathscr{P}$, where $\mathscr{P}$, as mentioned before, represents the entire parameter space considered including the region beyond the range of training sampling space. The active learning starts with a small set of sampling points, denoted as $\mathscr{P}_{\mathrm{tr}}^{\mathrm{active}}$, which can be chosen randomly or uniformly over the training sampling space. The data of memory kernels $K(t;\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}^{\mathrm{active}}$ are generated following Section 2.1.1. The following steps are then taken successively: (1) build the ROM for the memory kernel as in eqn (21) and construct a GPR model for each $\alpha_k(\boldsymbol{\mu})$, $k = 1,2,\ldots,R$; (2) quantify the uncertainty $\bar{\sigma}(\boldsymbol{\mu})$ of the GPR models for each $\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}$ as:

$$\bar{\sigma}(\mu) = \frac{\sum_{k=1}^{R} \hat{\sigma}_k(\mu)}{\sum_{k=1}^{R} \| \hat{\mathbf{u}}_k \|_\infty}, \tag{22}$$

with $\hat{\sigma}_k(\boldsymbol{\mu})$ the standard deviation of the GPR model for $a_k(\boldsymbol{\mu})$ and $\|\hat{\mathbf{u}}_k\|_\infty$ the L_∞ norm of $\hat{\mathbf{u}}_k$, a vector consisting of $\hat{u}_k$ for each $\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}$; (3) determine the next sampling parameter instance: $\boldsymbol{\mu}_{\mathrm{next}} = \arg\max_{\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}} \bar{\sigma}(\boldsymbol{\mu})$; and (4) generate the data of the memory kernel $K(t;\boldsymbol{\mu}_{\mathrm{next}})$ at $\boldsymbol{\mu}_{\mathrm{next}}$ and augment the training data set with the new data. These four steps are repeated until the maximum uncertainty $\bar{\sigma}(\boldsymbol{\mu}_{\mathrm{next}})$ is less than the preset tolerance ζ_{AL}. As such, the active learning can start with a small set of training data and then adaptively add more data as necessary at locations that can maximize information gain. The entire procedure is also outlined in Algorithm 1.

Algorithm 1 Active learning
Generate data of $K(t;\boldsymbol{\mu})$ for all $\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}^{\mathrm{active}}$
Set ζ_{AL}, ζ_{POD}, and $\bar{\sigma}(\boldsymbol{\mu}_{\mathrm{next}}) = \infty$
While $\bar{\sigma}(\boldsymbol{\mu}_{\mathrm{next}}) > \zeta_{\mathrm{AL}}$ **do**
Build the ROM as in eqn (21) *via* POD with the tolerance ζ_{POD}
Construct a GPR model for each $\alpha_k(\boldsymbol{\mu})$, $k = 1,\ldots,R$
Quantify the uncertainty $\bar{\sigma}(\boldsymbol{\mu})$ of the GPR models for each $\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}$
Determine $\boldsymbol{\mu}_{\mathrm{next}} = \arg\max_{\boldsymbol{\mu} \in \mathscr{P}_{\mathrm{tr}}} \bar{\sigma}(\boldsymbol{\mu})$
Generate the data of $K(t;\boldsymbol{\mu}_{\mathrm{next}})$
Augment the training parameter instances with $\mathscr{P}_{\mathrm{tr}}^{\mathrm{active}} = \mathscr{P}_{\mathrm{tr}}^{\mathrm{active}} \cup \{\boldsymbol{\mu}_{\mathrm{next}}\}$
end while
Output the ROM (as in eqn (21)) with the temporal bases $\phi_k(t)$ and the GPR model for each $\alpha_k(\boldsymbol{\mu})$, $k = 1,\ldots,R$

3 Results

In this section, we assess the accuracy and efficiency of our proposed transfer learning methods in two example systems: a star polymer and a peptoid polymer. The dynamics of the star polymer or peptoid polymer in solution can be affected by parameters such as temperature, concentration, and solvent property. We aimed to establish transferable memory kernels so that the CG models can faithfully reproduce the dynamic properties of the reference atomistic systems across different values of the parameters. The dynamic properties include the VACF, diffusion coefficient, and mean square displacement (MSD) as functions of time. For convenience, we hereinafter denote the transfer learning method based solely on GPR (Section 2.2.1) as GPR transfer learning and the transfer learning method *via* both ROM and GPR (Section 2.2.2) as ROM–GPR transfer learning.

In CG modeling, each star polymer or peptoid polymer was coarse-grained as a single CG particle, *i.e.*, the CG coordinate is the COM of each polymer, and the CG variable is the total momentum of each polymer; and the solvent (or water) DOFs were all eliminated. The memory kernels computed from atomistic data were obtained following Section 2.1.1. To construct a CG model, the memory kernel predicted by transfer learning was approximated by eqn (7). The dynamics of the CG system were simulated by solving eqn (10) numerically using the implicit velocity-Verlet temporal integrator, for which in-house computer codes were developed under the framework of LAMMPS.[61]

3.1 Star-polymer solution

We first considered solutions of star polymers. In the atomistic representation, each star polymer consists of a core Lennard-Jones (LJ) bead and 10 identical arms with 3 LJ beads per arm, as illustrated in Fig. 1.

The core LJ bead and the LJ beads in each arm are identical and connected by the finitely extensible non-linear elastic (FENE) bonds. The solvent is also modeled by LJ beads. The dynamics of the atomistic system are governed by the Hamiltonian:

$$H = \sum_{i=1}^{n} \frac{p_i^2}{2m_i} + \sum_{i \neq j} E(r_{ij}), \tag{23}$$

where n is the total number of LJ beads in the atomistic system; $\mathbf{p}_i$ and m_i are the momentum and mass of the i-th LJ bead, respectively; $r_{ij} = |\mathbf{r}_{ij}| = |\mathbf{r}_i - \mathbf{r}_j|$ is the distance between two LJ beads; and E denotes the total potential energy contributed by the interatomic and bonded potentials.

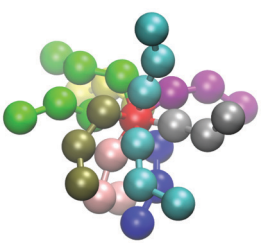


Fig. 1 Atomistic model of a star polymer consisting of 31 LJ beads: one core and 10 arms with 3 beads per arm.

The interatomic LJ potential between the i-th and j-th LJ beads adopts the purely repulsive Weeks–Chandler–Andersen (WCA) potential and is given by:

$$E_{\mathrm{WCA}}(r_{ij}) = \begin{cases} 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6} + \frac{1}{4}\right] & r_{ij} \leq r_{\mathrm{cut}} \\ 0 & r_{ij} > r_{\mathrm{cut}} \end{cases}, \tag{24}$$

where $r_{\mathrm{cut}} = 2^{1/6}\sigma_{ij}$ is the cutoff distance. We denote the WCA potential's parameters for the star polymers as $\varepsilon_{\mathrm{star}}$ and σ_{star} and for the solvent as $\varepsilon_{\mathrm{sol}}$ and σ_{sol}. In all MD simulations, while the values of $\varepsilon_{\mathrm{star}}$, σ_{star}, and $\varepsilon_{\mathrm{sol}}$ were fixed as $\varepsilon_{\mathrm{star}} = 1$, $\sigma_{\mathrm{star}} = 1$, and $\varepsilon_{\mathrm{sol}} = 1$, the value of σ_{sol} varied for different solution systems within the range $\sigma_{\mathrm{sol}} \in [1.0,2.2]$. For the interaction between a star-polymer LJ bead and a solvent LJ bead, the WCA potential's parameters were determined using the geometric mix rule.

The FENE potential for the bonded interaction between connected LJ beads in each star polymer is:

$$E_{\mathrm{FENE}}(r_{ij}) = \begin{cases} -\frac{1}{2}k_{\mathrm{b}}r_0^2 \ln\left[1 - \left(\frac{r_{ij}}{r_0}\right)^2\right] & r_{ij} \leq r_0 \\ \infty & r_{ij} > r_0 \end{cases}, \tag{25}$$

where $k_{\mathrm{b}} = 30$ is the spring constant, and $r_0 = 1.5$ is the maximum length of the FENE spring.

Here, the reduced LJ units were employed; the mass of all LJ beads was chosen to be unity. Each star-polymer solution consists of 620 000 LJ beads. The MD simulations were performed using LAMMPS.[61] In all simulations, a periodic cubic box of length 115.7295 was used, which is large enough to neglect the effect of the periodic box on the VACF. The Nose–Hoover thermostat under the canonical ensemble (*NVT*) was employed with the time step $\Delta t = 0.001$. In each MD simulation, the data after reaching thermal equilibrium were collected for computing the ensemble-averaged quantities of interest.

The concentration of star polymers in each solution was different and varied with $\gamma \in [0.2,0.8]$, where γ is defined as the number of star-polymer LJ beads divided by the total number of all LJ beads. In addition, the temperature varied with $T \in [0.5,2.0]$ for each system. Thus, there are three parameters that differentiate the star-polymer solution systems considered herein, *i.e.*, the concentration of star polymers γ, the temperature T, and the solvent viscosity characterized by σ_{sol}. We applied the proposed transfer learning methods to enable memory kernel to transfer across these parameters. The memory kernel was determined up to $t_{\mathrm{f}} = 20$ until the VACF ($C(t_{\mathrm{f}})$) decayed to $|C(t_{\mathrm{f}})|/|C(0)| \leq 10^{-2}$, which is long enough to capture the dynamics in all time scales.

To compare the two methods (GPR transfer learning and ROM–GPR transfer learning) and also to demonstrate how the proposed active learning technique can be implemented, we first limited the transfer learning to one parameter. Thereafter, we tackled the more challenging case over all three parameters

using the ROM–GPR transfer learning method along with active learning.

3.1.1 Transferable in one parameter. The temperature and solvent viscosity were fixed at $T = 1.0$ and $\sigma_{sol} = 1.0$, respectively. The concentration of star polymers γ was the only changing parameter, *i.e.*, $\boldsymbol{\mu} = \gamma \in [0.2,0.8]$. Hence, the parameter space is one-dimensional: $\mathscr{P} = [0.2,0.8]$. Here, the parameter instances used for training were from $\mathscr{P}_{tr} = [0.2,0.6]$. The proposed active learning technique allowed us to adaptively sample the training data and to minimize the data needed for training the GPR models given the target tolerance of accuracy.

The tolerances for POD and active learning were set as $\zeta_{POD} = 0.1$ and $\zeta_{AL} = 0.005$, respectively. The active learning process was initiated with two initial parameter instances randomly sampled as $\mathscr{P}_{tr}^{active} = \{0.2496,0.4528\}$. At these two parameter instances, we computed the memory kernels $K(t;\gamma = 0.2496)$ and $K(t;\gamma = 0.4528)$ from the MD simulation data as described in Section 2.1.1. Through POD we constructed the ROM as in eqn (21), where $R = 1$ and only the first POD basis needs to be retained for the target $\zeta_{POD} = 0.1$. The GPR model for the parameter mode $\alpha_1(\mu)$ inferred from the two training data instances, along with its uncertainty as defined in eqn (22), are shown in Fig. 2. With only two training data instances, the inferred GPR model exhibits large uncertainty, which indicates that more training data are needed. Because the uncertainty quantified by $\bar{\sigma}(\mu)$ reaches its maximum at $\mu = 0.6$, it was chosen as the next sampling point $\mu_{next} = 0.6$. The memory kernel $K(t;\gamma = 0.6)$ was computed from the MD simulation data and added to the training data.

Now the training sampling set became $\mathscr{P}_{tr}^{active} = \{0.2486,0.4528,0.6\}$. With the three training data instances, the GPR model inferred for $\alpha_1(\mu)$ along with its uncertainty is depicted in Fig. 3. Compared with Fig. 2, the uncertainty of the GPR model was greatly reduced by adding one more piece of training data. However, the maximum uncertainty is still larger than the preset tolerance $\zeta_{AL} = 0.005$, and hence, the next sampling point was chosen where the maximum uncertainty occurs, *i.e.*, $\mu_{next} = 0.2$. With that, the training sampling set became $\mathscr{P}_{tr}^{active} = \{0.2486,0.4528,0.6,0.2\}$. The GPR model for $\alpha_1(\mu)$ and its uncertainty inferred from the four training data is illustrated in Fig. 4. By including one more training data instance, the GPR model's uncertainty was further reduced, and the maximum uncertainty was less than the preset tolerance $\zeta_{AL} = 0.005$. Thus, the sampling process *via* active learning could be terminated. The GPR model for $\alpha_1(\mu)$ in Fig. 4

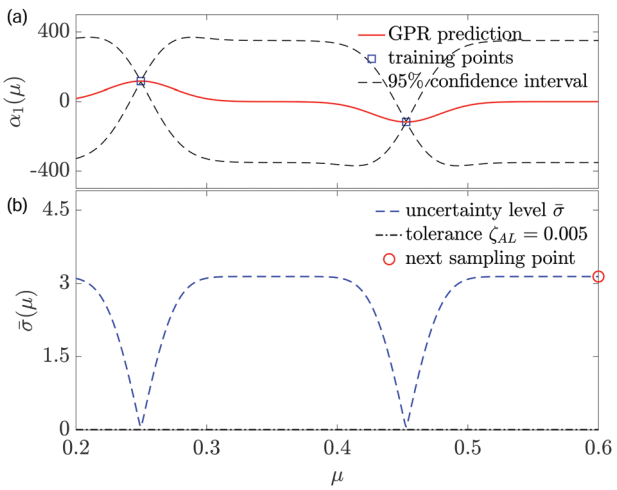


Fig. 2 The first step of the active learning process: (a) GPR model for the POD parameter mode $\alpha_1(\mu)$ inferred from 2 randomly initialized training data instances. (b) Uncertainty quantified by $\bar{\sigma}(\mu)$ of the inferred GPR model and the next sampling point ($\mu_{next} = 0.6$) determined by active learning.

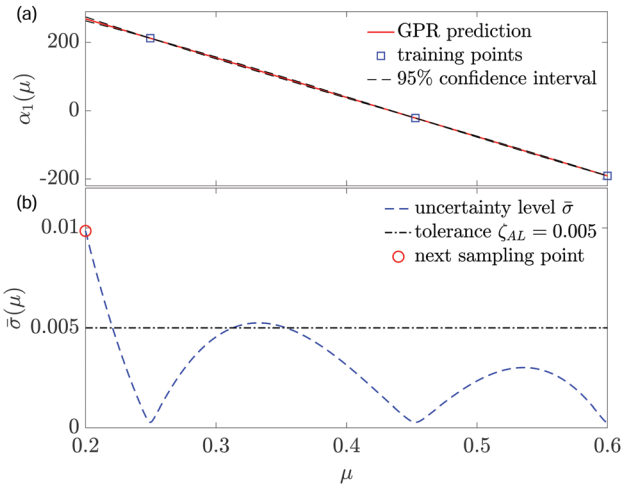


Fig. 3 The second step of the active learning process: (a) GPR model for the POD parameter mode $\alpha_1(\mu)$ inferred from 3 training data instances. Here the third training data instance is obtained from Fig. 2. (b) Uncertainty quantified by $\bar{\sigma}(\mu)$ of the inferred GPR model and the next sampling point ($\mu_{next} = 0.2$) determined by active learning.

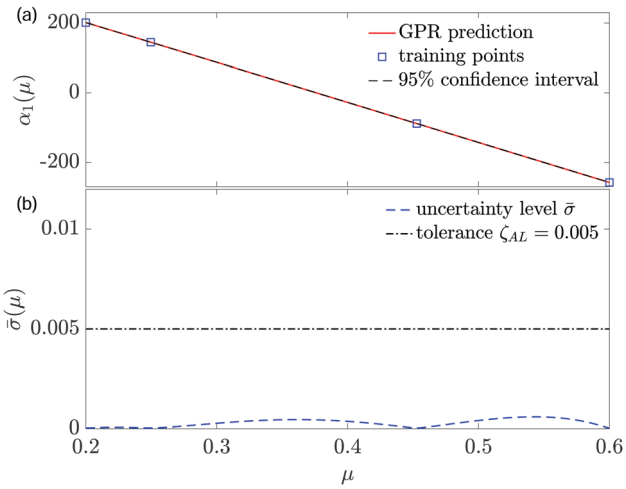


Fig. 4 The third step of the active learning process: (a) GPR model for the POD parameter mode $\alpha_1(\mu)$ inferred from 4 training data instances, where the fourth training data instance is obtained from Fig. 3. (b) Uncertainty quantified by $\bar{\sigma}(\mu)$ of the inferred GPR model. Here, the active learning process is terminated since the maximum of the uncertainty level is smaller than the preset tolerance.

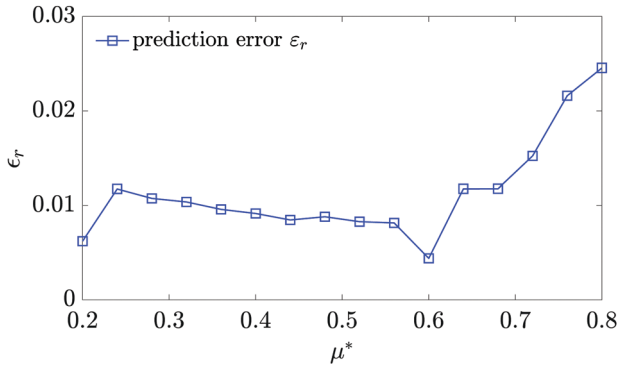


Fig. 5 The relative error $\varepsilon_r(\mu^*)$ of the memory kernels $K^{R-G}(t;\mu^*)$ predicted by ROM–GPR transfer learning at different testing parameter instances $\mu^* \in \mathscr{P}_{test}$.

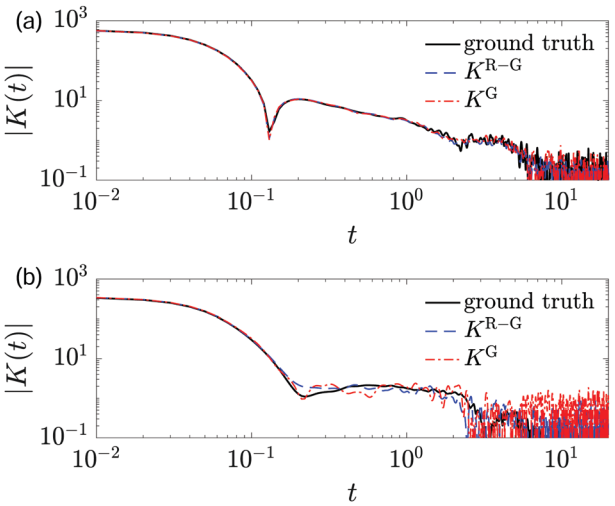


Fig. 6 Star polymer: the memory kernels $K(t)$ predicted at (a) $\mu_1^* = \gamma_1^* = 0.4$ (interpolation) and (b) $\mu_2^* = \gamma_2^* = 0.8$ (extrapolation) by the ROM–GPR (K^{R-G}) and GPR (K^G) transfer learning methods, respectively, compared with the memory kernels computed from the MD simulation data (ground truth). Here, γ is the concentration of star polymers; the same training parameter instances $\mathscr{P}_{tr}^{active}$ were used for both methods.

was then substituted into the ROM in eqn (21) with $R = 1$ to predict the memory kernel at any given parameter $\mu^* \in \mathscr{P}$.

To examine the accuracy of such a predicted memory kernel, we selected a set of testing parameter instances: $\mathscr{P}_{test} = \{0.2 + 0.04i\}_{i=0,1,\ldots,15}$. Note that while some of these testing parameter instances are within the range of the training sampling space $\mathscr{P}_{tr}$, others are beyond the range; *i.e.*, the predictions include not only interpolation but also extrapolation. We used the relative error to evaluate the accuracy of prediction, which is defined as: $\varepsilon_r = \frac{\| K^{R-G} - K \|_2}{\| K \|_2}$ with K^{R-G} denoting the memory kernel predicted by ROM–GPR transfer learning, K the memory kernel determined from MD simulation data, and $\|\cdot\|_2$ denoting the L_2 norm. The relative errors of the memory kernels predicted at all 16 testing parameter instances are depicted in Fig. 5. It can be seen that all predictions are accurate with the largest relative error of 2.5%; extrapolation outside the training sampling space $\mathscr{P}_{tr}$ is less accurate than interpolation inside $\mathscr{P}_{tr}$; and the error increases as the extrapolation goes further.

Next, we compare the ROM–GPR transfer learning method with the GPR transfer learning method in terms of the accuracy and computational cost. For that, we used the same training parameter instances determined by active learning, *i.e.*, $\mathscr{P}_{tr}^{active} = \{0.2486, 0.4528, 0.6, 0.2\}$; and two testing parameter instances were selected: $\mu_1^* = 0.4$ inside the training sampling space $\mathscr{P}_{tr}$ and $\mu_2^* = 0.8$ outside $\mathscr{P}_{tr}$. The memory kernels predicted at the two testing parameter instances by the two methods are compared in Fig. 6. By comparison with the memory kernel computed from the MD simulation data (regarded as the "ground truth"), we find that the predictions of these two methods both achieve good accuracy with the relative errors $\varepsilon_r(\mu_1^*) = 0.009$ and $\varepsilon_r(\mu_2^*) = 0.025$ for K^{R-G} and $\varepsilon_r(\mu_1^*) = 0.011$ and $\varepsilon_r(\mu_2^*) = 0.046$ for K^G, where K^G denotes the memory kernel predicted by the GPR transfer learning method. However, the computational costs of the two methods are significantly different. Specifically, it took the GPR transfer learning method 1086 s to construct the GPR model and 0.2832 s to predict the memory kernel at a testing parameter instance, but it only took the ROM–GPR transfer learning method 0.1127 s to construct the ROM and GPR model and 7.880×10^{-4} s to predict $K(t;\mu^*)$. Both methods were implemented in Matlab and run on a workstation with an Intel Core CPU i5-6500 3.20 GHz processor. We hence find evidence that the transfer learning based on GPR can be greatly accelerated by model order reduction.

Finally, from the predicted memory kernel at a given parameter instance, the CG model was constructed. For that, the memory kernel was first approximated by eqn (7). The dynamics of the CG system were then simulated by solving eqn (10). To examine whether the CG model could accurately reproduce the dynamic properties of the reference atomistic system in all time scales, we computed the VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD as functions of time. In Fig. 7 and 8, we present the results obtained from CG modeling at two different concentrations of star polymers ($\gamma_1^* = 0.4$ and $\gamma_2^* = 0.8$) compared with the MD simulation results. Overall good agreements can be found, and all stages of diffusion from super-diffusion to normal diffusion are correctly captured. We hence demonstrate that by transfer learning of 4 memory kernels at concentrations $\gamma = \{0.2486, 0.4528, 0.6, 0.2\}$, we were able to construct the CG model that could accurately reproduce the reference atomistic system's dynamic properties in all time scales across a range of concentrations ($\gamma^* \in (0.2, 0.8]$), unseen in the training data and even outside the range of training sampling space $\mathscr{P}_{tr} = [0.2, 0.6]$.

3.1.2 Transferable in multiple parameters. After verifying the transferability of the memory kernel in one parameter and engaging in a detailed discussion about the two transfer learning methods and the active learning strategy, we next demonstrate how transfer learning can enable the memory kernel to be transferable in multiple parameters. In particular, the parameter space comprises three parameters: the temperature T, the concentration of star polymers γ, and the solvent viscosity

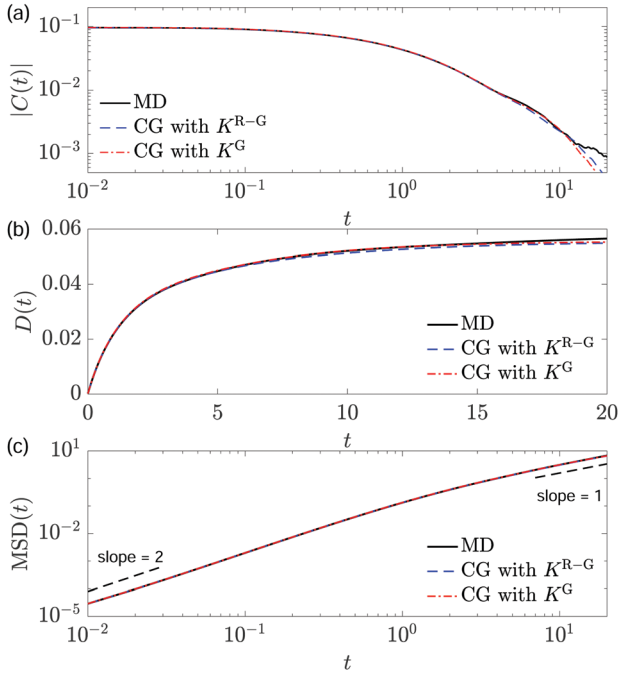


Fig. 7 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD of the star polymer predicted by the CG models at $\mu_1^* = \gamma_1^* = 0.4$ compared with the MD simulation results. Here, γ is the concentration of star polymers; the memory kernel predicted by the ROM–GPR (K^{R-G}) or GPR (K^{G}) transfer learning method was approximated by eqn (7) with $\mathcal{N} = 7$ terms.

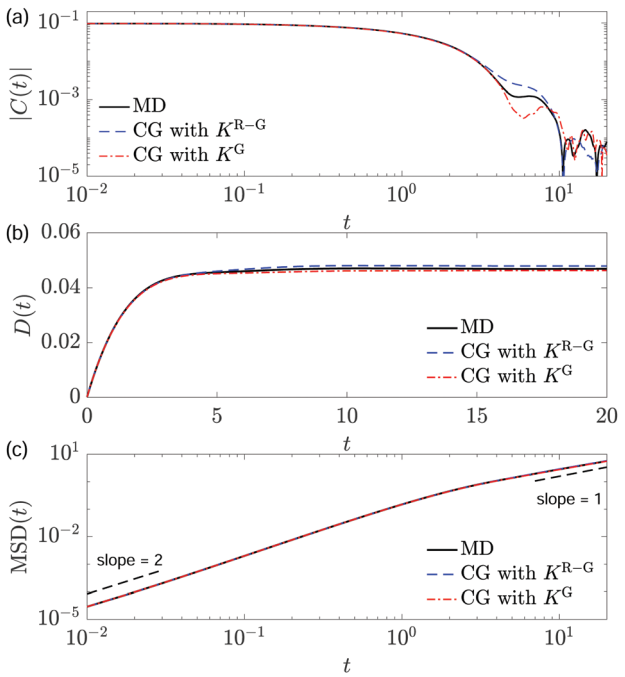


Fig. 8 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD of the star polymer predicted by the CG models at $\mu_2^* = \gamma_2^* = 0.8$ (outside the training sampling space), compared with the MD simulation results. Here, γ is the concentration of star polymers; the memory kernel predicted by the ROM–GPR (K^{R-G}) or GPR (K^{G}) transfer learning method was approximated by eqn (7) with $\mathcal{N} = 7$ terms.

characterized by σ_{sol}, and was specified as $\mathscr{P} = [0.5,2.0] \times [0.2,0.8] \times [1.0,2.2]$ in this case with the training sampling space set as $\mathscr{P}_{tr} = [0.75,1.75] \times [0.3,0.7] \times [1.1,2.1]$. In such a multi-dimensional, large parameter space, the memory kernels' magnitudes and time scales vary in orders, as depicted in Fig. 9, where 4 memory kernel samples are displayed at 4 parameter instances sampled in $\mathscr{P}_{tr}$. This makes transfer learning challenging and hence serves as a good problem to assess the effectiveness of our proposed methods.

The active learning process started with 8 initial training parameter instances chosen as $\mathscr{P}_{tr}^{active} = \{(0.75 + 1i_1, 0.3 + 0.4i_2, 1.1 + 1i_3)\}_{i_1=0,1;\ i_2=0,1;\ i_3=0,1}$. For the preset tolerance $\zeta_{POD} = 0.01$, 8 POD bases were retained in the ROM, *i.e.*, $R = 8$ in eqn (21). As a result of active learning, $N_p = 54$ sampling points was determined as the number of final training parameter instances, such that the uncertainty (defined in eqn (22)) of the GPR models for the parameter modes $\alpha_k(\boldsymbol{\mu})$ with $k = 1,\ldots,8$ was less than the preset tolerance $\zeta_{AL} = 0.025$. To illustrate the advantage of active learning, we set up a control group, where the training parameter instances of the same number ($N_p = 54$) were sampled uniformly as: $\mathscr{P}_{tr}^{uniform} = \{(0.75 + 0.5i_1, 0.3 + 0.2i_2, 1.1 + 0.2i_3)\}_{i_1=0,1,2;\ i_2=0,1,2;\ i_3=0,\ldots,5}$. Two testing parameter instances were selected: $\boldsymbol{\mu}_1^* = (1.75, 0.42, 1.74)$ inside the training sampling space $\mathscr{P}_{tr}$ but not in $\mathscr{P}_{tr}^{active}$ or $\mathscr{P}_{tr}^{uniform}$, and $\boldsymbol{\mu}_2^* = (0.5, 0.2, 1.0)$ outside $\mathscr{P}_{tr}$. Fig. 10 presents the memory kernels predicted by the ROM–GPR transfer learning method at the two testing parameter instances with different sets of training parameter instances, *i.e.*, $\mathscr{P}_{tr}^{active}$ *vs.* $\mathscr{P}_{tr}^{uniform}$. By comparing the results with the ground truth, it is obvious that using the same number of training data, the memory kernels predicted using the training parameter instances adaptively sampled *via* active learning are more accurate than using the uniformly sampled training parameter instances. While the relative errors for the former are $\varepsilon_r(\mu_1^*) = 0.026$ and $\varepsilon_r(\mu_2^*) = 0.068$, the relative errors for the latter are $\varepsilon_r(\mu_1^*) = 0.349$ and $\varepsilon_r(\mu_2^*) = 0.932$. Thus, to achieve the same target accuracy, transfer learning coupled with active learning may require much less training data, which thereby greatly reduces the training cost.

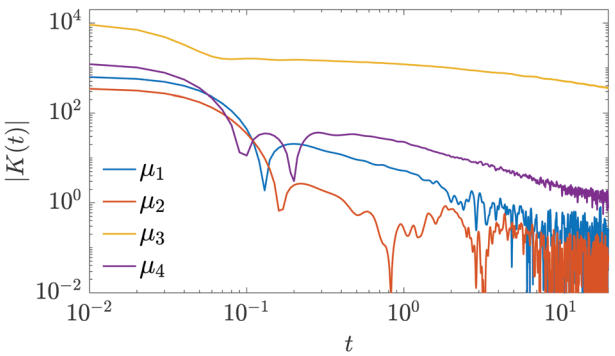


Fig. 9 Star polymer: sample memory kernels at 4 parameter instances: $\boldsymbol{\mu}_1 = (0.75, 0.3, 1.1)$, $\boldsymbol{\mu}_2 = (0.75, 0.7, 1.1)$, $\boldsymbol{\mu}_3 = (1.75, 0.3, 2.1)$, and $\boldsymbol{\mu}_4 = (1.75, 0.7, 2.1)$. Here, $\boldsymbol{\mu} = (T, \gamma, \sigma_{sol})$ with T the temperature, γ the concentration of star polymers, and σ_{sol} characterizing the solvent viscosity.

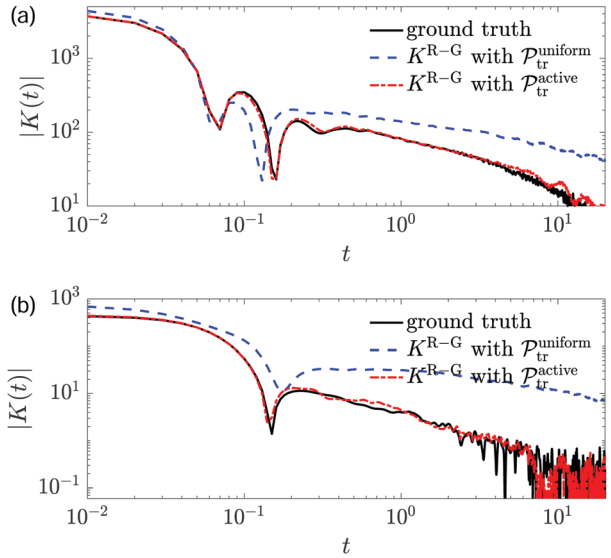


Fig. 10 Star polymer: the memory kernels $K^{R-G}(t)$ predicted at (a) $\boldsymbol{\mu}_1^*$ = (1.75,0.42,1.74) and (b) $\boldsymbol{\mu}_2^*$ = (0.5,0.2,1.0) by the ROM–GPR transfer learning method with $\mathcal{P}_{tr}^{active}$ *vs.* $\mathcal{P}_{tr}^{uniform}$, compared with the memory kernels directly computed from the MD simulation data (ground truth). Here, both $\mathcal{P}_{tr}^{active}$ and $\mathcal{P}_{tr}^{uniform}$ have 54 training parameter instances.

The CG model was then constructed from the predicted $K^{R-G}(t)$. Fig. 11 and 12 present the VACF, diffusion coefficient, and MSD predicted by the CG model. They agree well with the MD simulation results. All stages of diffusion – super-, sub-, and normal-diffusion (in Fig. 11) – are correctly captured. We hence verify that by transfer

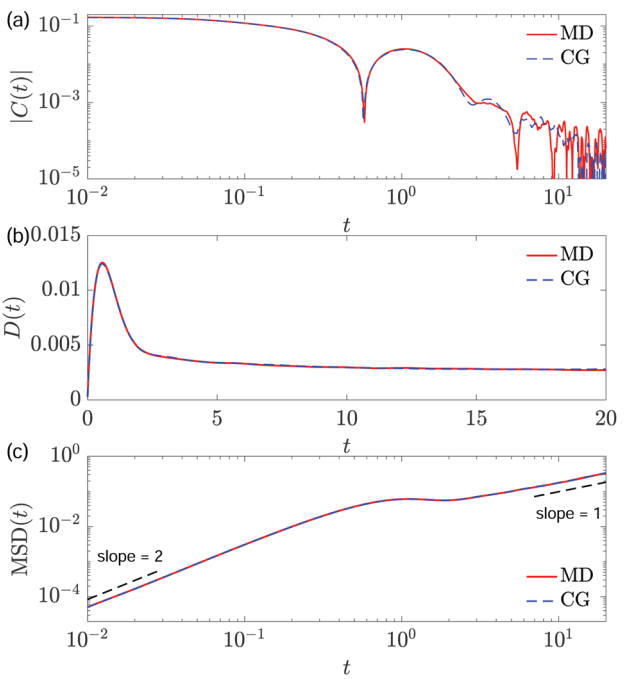


Fig. 11 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD of the star polymer predicted by the CG model at $\boldsymbol{\mu}_1^*$ = (1.75,0.42,1.74) compared with the MD simulation results. Here, the memory kernel predicted by the ROM–GPR transfer learning method coupled with active learning was approximated by eqn (7) with $\mathcal{N}$ = 5 terms.

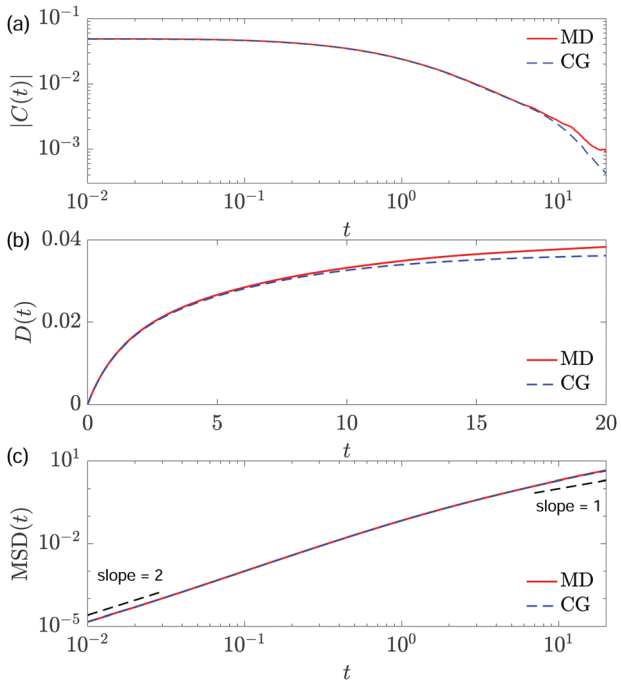


Fig. 12 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD of the star polymer predicted by the CG model at $\boldsymbol{\mu}_2^*$ = (0.5,0.2,1.0) (outside the training sampling space), compared with the MD simulation results. Here, the memory kernel predicted by the ROM–GPR transfer learning method coupled with active learning was approximated by eqn (7) with $\mathcal{N}$ = 6 terms.

learning, the memory kernel is transferable in multiple parameters, built on which the CG models can accurately reproduce the dynamic properties of the reference atomistic system in all time scales for different temperatures, concentrations of star polymers, and solvent viscosities.

3.2 Peptoid polymer

After assessing the proposed methods in a model polymer solution system, we next studied a real polymer system consisting of peptoid polymers, which are known as a class of highly configurable biopolymers.[62–68] The peptoid polymer studied in this work contains *N*-[2-(4-chlorophenyl)ethyl] glycine ($N_{4\text{-Cl}}$pe) and *N*-(2-carboxyethyl)glycine (Nce) groups,[63] whose chemical structure is described in Fig. 13. In different solution systems, a peptoid polymer can consist of different numbers of repeat units, denoted as z. The peptoid solutions were made of one peptoid polymer immersed in water with a fixed concentration of 1600 mg kg^{-1}. The all-atom representation of a peptoid polymer in MD simulations is shown in Fig. 14. Here, the AMBER03[69] force field with the parameters from the generalized AMBER force field[63,70,71] was adopted for the peptoid molecule; the SPC force field[72] was employed for modeling the water molecules. The MD simulations were performed using the GROMACS package.[73]

The appropriate size of the periodic cubic box was determined by running the MD simulation in the isothermal–isobaric ensemble (*NPT*) using the Parrinello–Rahman barostat[74] at 1 bar

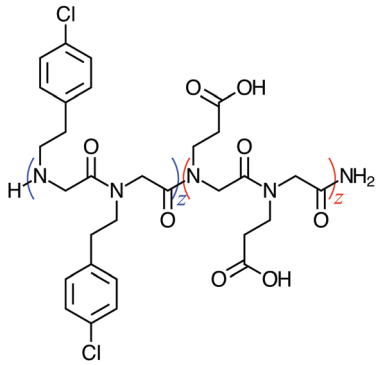


Fig. 13 Chemical structure of a peptoid polymer, where z denotes the number of repeat units: $N_{4\text{-Cl}}$pe and Nce.

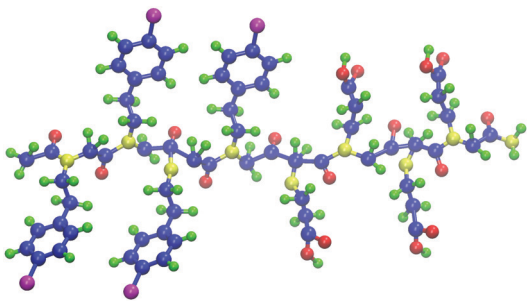


Fig. 14 Atomistic representation of a peptoid polymer with the number of repeat units z = 2.

with isotropic pressure coupling. The MD simulations were then performed in the *NVT* ensemble with the modified Berendsen thermostat.[75] To maintain the entire system in the NVT ensemble more effectively, the thermostat was enforced to the peptoid molecule and to the water molecules separately. The time step Δt was chosen as 2 fs. After the thermal equilibrium was reached (after 100 ps in our case), data were collected for computing the ensemble-averaged quantities of interest.

In this study, the temperature and the length (*i.e.*, the number of repeat units) of a peptoid polymer varied in different systems. Hence, they were the two parameters to practise transfer learning and to assess the transferability of the memory kernel. The parameter space was specified as $\mathscr{P} = [275,365] \times [1,6]$, *i.e.*, the temperature T varies from 275 K to 365 K, and the number of repeat units in a peptoid polymer z varies from 1 to 6. The training sampling space was set as $\mathscr{P}_{\text{tr}} = [275,315] \times [1,3]$. The 4 initial training parameter instances for active learning were sampled as $\mathscr{P}_{\text{tr}}^{\text{active}} = \{(275 + 40i_1, 1 + 2i_2)\}_{i_1=0,1;\ i_2=0,1}$. The tolerance for POD was set as $\zeta_{\text{POD}} = 0.1$, for which 2 POD bases were retained in the constructed ROM with $R = 2$ in eqn (21). After adding 6 more sampling points *via* active learning, the uncertainty (defined in eqn (22)) of the GPR models for the parameter modes $\alpha_k(\boldsymbol{\mu})$ with k = 1,2 was less than the preset tolerance $\zeta_{\text{AL}} = 0.01$. Thus, we used the memory kernels computed from the MD simulation data at the 10 sampled parameter instances as the training data in the transfer learning. To ensure that the memory kernel is long enough to capture the dynamics in all time scales, each was computed up to $t_{\text{f}} = 50$ ps until the VACF ($C(t_{\text{f}})$) decayed to $|C(t_{\text{f}})|/|C(0)| \leq 10^{-2}$.

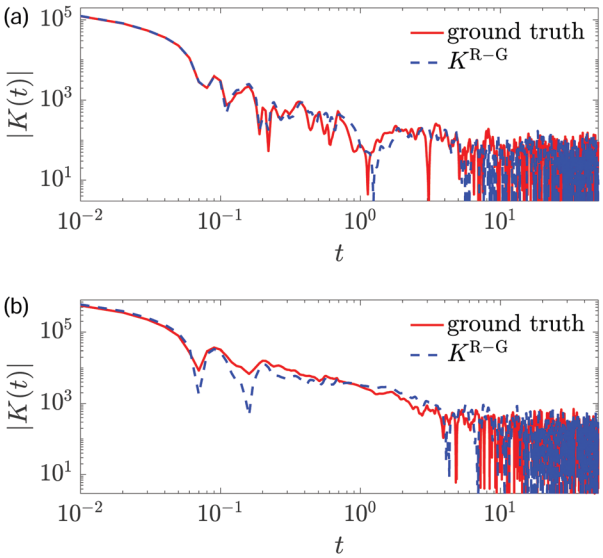


Fig. 15 Peptoid polymer: the memory kernels $K^{\text{R-G}}(t)$ predicted by the ROM–GPR transfer learning method at (a) $\boldsymbol{\mu}_1^*$ = (307,1) (interpolation), corresponding to T = 307 K and z = 1 and (b) $\boldsymbol{\mu}_2^*$ = (365,6) (extrapolation), corresponding to T = 365 K and z = 6, compared with the memory kernels directly computed from the MD simulation data (ground truth).

To examine the performance of transfer learning, two testing parameter instances were selected: $\boldsymbol{\mu}_1^*$ = (307,1), which is where the uncertainty of the GPR models for all parameter modes is

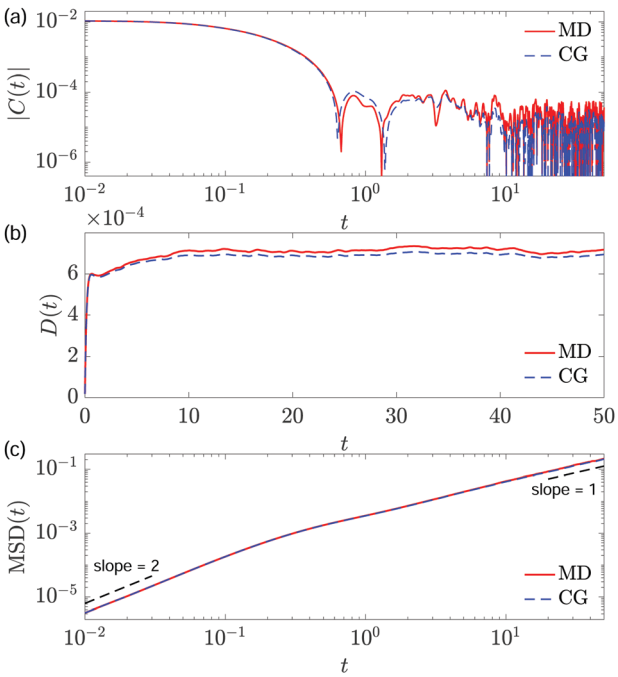


Fig. 16 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD predicted by the CG model at T = 307 K for the peptoid polymer with one $N_{4\text{-Cl}}$pe and Nce, compared with the MD simulation results. Here, to construct the CG model, $K^{\text{R-G}}(t;\boldsymbol{\mu}_1^*)$ in Fig. 15 (a) was approximated by eqn (7) with $\mathscr{N}$ = 7 terms.

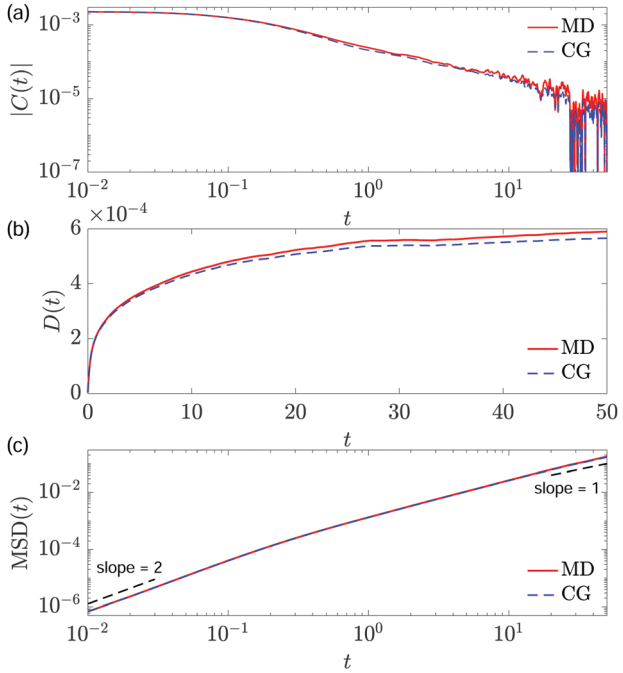


Fig. 17 VACF ($C(t)$), diffusion coefficient ($D(t)$), and MSD predicted by the CG model at T = 365 K for the peptoid polymer with 6 $N_{4\text{-}Cl}$pe and Nce (outside the training sampling space), compared with the MD simulation results. Here, to construct the CG model, $K^{R-G}(t;\boldsymbol{\mu}_2^*)$ in Fig. 15 (b) was approximated by eqn (7) with $\mathcal{N}$ = 8 terms.

maximized inside the training sampling space $\mathscr{P}_{tr}$; and $\boldsymbol{\mu}_2^*$ = (365,6), which is outside $\mathscr{P}_{tr}$ and represents the most extreme case in the parameter space considered. Fig. 15 shows the memory kernels predicted at the two selected testing parameter instances. The figure shows that the memory kernels predicted by the ROM–GPR transfer learning agree well with the ground truths with the relative errors of 0.028 and 0.086 for $\boldsymbol{\mu}_1^*$ and $\boldsymbol{\mu}_2^*$, respectively.

Using the predicted memory kernels $K^{R-G}(t;\boldsymbol{\mu}^*)$, we constructed the CG models. The VACF, diffusion coefficient, and MSD predicted by the CG models were compared with the MD simulation results, as depicted in Fig. 16 and 17. Good agreements were achieved in all time scales. Thus, in a real polymer solution system, we again verify that by transfer learning, the memory kernel can be transferable in multiple parameters and across a non-trivially large parameter space. Built on the transferable memory kernel, the CG model can accurately reproduce the dynamic properties of the peptoid polymer in all time scales for different temperatures and lengths of polymer.

4 Conclusion

We have introduced transfer learning of the memory kernel in CG modeling. Built on a transferable memory kernel, the CG model can reproduce the dynamic properties of the reference atomistic system under different thermodynamic conditions such as temperature and solvent viscosity and for different systems with varying solute concentrations and lengths of polymer chains. Notably, the transferability allows for out-of-sample predictions in the extrapolated domain of parameters. Through two example polymer solution systems, we have demonstrated the accuracy and efficiency of the proposed transfer learning. Because the memory kernels are accurately predicted, the CG models can reproduce the dynamic properties of polymers in all time scales, including the VACF, diffusion coefficient, and MSD. The predictions of the CG models agree well with the atomistic simulation results, and all stages of diffusion – super-, sub-, and normal-diffusion – are correctly captured. We expect that the proposed methodology is generally applicable to CG modeling of other kinds of polymer solutions and soft matter systems such as colloid suspensions. While the parameters considered in this work include the temperature, solvent viscosity, concentration of polymers, and the length of polymer chains, the proposed transfer learning can certainly be applied to a broader range of parameters beyond this list.

Two transfer learning methods have been proposed and compared. The GPR transfer learning method directly models memory kernels (functions of time) at different parameters as a multivariate Gaussian process. GPR is not only flexible for interpolation or extrapolation at any given input, but also can quantify the uncertainty of any prediction. The ROM–GPR transfer learning method integrates the GPR with model order reduction and active learning, by which it is more computationally efficient and requires minimum training data. In the ROM–GPR transfer learning method, a ROM is first constructed *via* POD for the memory kernel. By such, the memory kernel is represented in a reduced temporal and parameter space, and the GPR is only needed for the parameter modes. Thus, both the training and prediction costs of GPR are greatly reduced. Furthermore, guided by the uncertainty quantified in GPR, the active learning technique enables adaptive sampling of the training data. Compared with other sampling strategies, *e.g.*, uniform sampling, using the same number of training data, active learning leads to much more accurate transfer learning. Thus, to achieve the same accuracy, active learning requires less training data and thereby a lower training cost.

The present work has attempted to construct transferable memory kernels that preserve dynamic properties under coarse-graining. Potentially, it can be integrated with the efforts in the literature that focus on attaining transferable CG potentials,[33–37] so that the CG modeling can preserve both the structural and dynamic properties of the underlying atomistic system and is transferable across a range of parameters. This will be our future work.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

Z. M., W. P., M. K., and K. L. acknowledge the funding support from the Defense Established Program to Stimulate

Competitive Research (DEPSCoR) Grant No. FA9550-20-1-0072. S. W. and W. P. acknowledge the funding support from the National Science Foundation under Grant No. CMMI-1761068. Peptoid design and data interpretation were supported by the U.S. Department of Energy (DOE), Office of Basic Energy Sciences (OBES), Biomolecular Materials Program at Pacific Northwest National Laboratory (PNNL). PNNL is a multi-program national laboratory operated for DOE by Battelle under Contract No. DE-AC05-76RL01830.

References

1 M. G. Saunders and G. A. Voth, *Annu. Rev. Biophys.*, 2013, **42**, 73–93.
2 S. Kmiecik, D. Gront, M. Kolinski, L. Wieteska, A. E. Dawid and A. Kolinski, *Chem. Rev.*, 2016, **116**, 7898–7936.
3 M. Dinpajooh and M. G. Guenza, *Soft Matter*, 2018, **14**, 7126–7144.
4 K. M. Salerno, A. Agrawal, D. Perahia and G. S. Grest, *Phys. Rev. Lett.*, 2016, **116**, 058302.
5 A. Gooneie, S. Schuschnigg and C. Holzer, *Polymers*, 2017, **9**, 16.
6 M. Zhao, J. Sampath, S. Alamdari, G. Shen, C.-L. Chen, C. J. Mundy, J. Pfaendtner and A. L. Ferguson, *J. Phys. Chem. B*, 2020, **124**, 7745–7764.
7 T. K. Haxton, R. N. Zuckermann and S. Whitelam, *J. Chem. Theory Comput.*, 2016, **12**, 345–352.
8 P. Du, S. W. Rick and R. Kumar, *Phys. Chem. Chem. Phys.*, 2018, **20**, 23386–23396.
9 J. Kleinjung and F. Fraternali, *Curr. Opin. Struct. Biol.*, 2014, **25**, 126–134.
10 T. T. Pham, M. Bajaj and J. R. Prakash, *Soft Matter*, 2008, **4**, 1196–1207.
11 R. Chudoba, J. Heyda and J. Dzubiella, *Soft Matter*, 2018, **14**, 9631–9642.
12 G. Sevink and J. Fraaije, *Soft Matter*, 2014, **10**, 5129–5146.
13 S. Wang, Z. Li and W. Pan, *Soft Matter*, 2019, **15**, 7567–7582.
14 S. Wang, Z. Ma and W. Pan, *Soft Matter*, 2020, **16**, 8330–8344.
15 Z. G. Mills, W. Mao and A. Alexeev, *Trends Biotechnol.*, 2013, **31**, 426–434.
16 J. Mu, R. Motokawa, C. D. Williams, K. Akutsu, S. Nishitsuji and A. J. Masters, *J. Phys. Chem. B*, 2016, **120**, 5183–5193.
17 L. Rovigatti, N. Gnan, L. Tavagnacco, A. J. Moreno and E. Zaccarelli, *Soft Matter*, 2019, **15**, 1108–1119.
18 P. Bełdowski, P. Weber, A. Dėdinaitė, P. M. Claesson and A. Gadomski, *Soft Matter*, 2018, **14**, 8997–9004.
19 D. Reith, M. Pütz and F. Müller-Plathe, *J. Comput. Chem.*, 2003, **24**, 1624–1636.
20 A. P. Lyubartsev and A. Laaksonen, *Phys. Rev. E: Stat. Phys., Plasmas, Fluids, Relat. Interdiscip. Top.*, 1995, **52**, 3730.
21 S. Izvekov and G. A. Voth, *J. Phys. Chem. B*, 2005, **109**, 2469–2473.
22 M. S. Shell, *J. Chem. Phys.*, 2008, **129**, 144108.
23 T. Sanyal and M. S. Shell, *J. Chem. Phys.*, 2016, **145**, 034109.
24 S. T. John and G. Csányi, *J. Phys. Chem. B*, 2017, **121**, 10934–10949.
25 L. Zhang, J. Han, H. Wang, R. Car and W. E, *J. Chem. Phys.*, 2018, **149**, 034101.
26 J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. de Fabritiis, F. Noé and C. Clementi, *ACS Cent. Sci.*, 2019, **5**, 755–767.
27 Z. Li, H. S. Lee, E. Darve and G. E. Karniadakis, *J. Chem. Phys.*, 2017, **146**, 014104.
28 L. Ma, X. Li and C. Liu, *J. Chem. Phys.*, 2016, **145**, 204117.
29 H. Lei, N. A. Baker and X. Li, *Proc. Natl. Acad. Sci. U. S. A.*, 2016, **113**, 14183–14188.
30 G. Jung, M. Hanke and F. Schmid, *J. Chem. Theory Comput.*, 2017, **13**, 2481–2488.
31 G. Jung, M. Hanke and F. Schmid, *Soft Matter*, 2018, **14**, 9368–9382.
32 H. S. Lee, S.-H. Ahn and E. F. Darve, *J. Chem. Phys.*, 2019, **150**, 174113.
33 B. E. Husic, N. E. Charron, D. Lemm, J. Wang, A. Pérez, M. Majewski, A. Krämer, Y. Chen, S. Olsson, G. de Fabritiis, F. Noé and C. Clementi, *J. Chem. Phys.*, 2020, **153**, 194101.
34 J. Ruza, W. Wang, D. Schwalbe-Koda, S. Axelrod, W. H. Harris and R. Gómez-Bombarelli, *J. Chem. Phys.*, 2020, **153**, 164501.
35 J. Jin, A. Yu and G. A. Voth, *J. Chem. Theory Comput.*, 2020, **16**, 6823–6842.
36 J. Sauter and A. Grafmüller, *J. Chem. Theory Comput.*, 2017, **13**, 223–236.
37 K. Shen, N. Sherck, M. Nguyen, B. Yoo, S. Köhler, J. Speros, K. T. Delaney, G. H. Fredrickson and M. S. Shell, *J. Chem. Phys.*, 2020, **153**, 154116.
38 I. Y. Lyubimov, J. McCarty, A. Clark and M. G. Guenza, *J. Chem. Phys.*, 2010, **132**, 224903.
39 I. Lyubimov and M. G. Guenza, *Phys. Rev. E: Stat., Nonlinear, Soft Matter Phys.*, 2011, **84**, 031801.
40 W. Xia, J. Song, C. Jeong, D. D. Hsu, F. R. Phelan, J. F. Douglas and S. Keten, *Macromolecules*, 2017, **50**, 8787–8796.
41 W. Xia, N. K. Hansoge, W.-S. Xu, F. R. Phelan, S. Keten and J. F. Douglas, *Sci. Adv.*, 2019, **5**, eaav4683.
42 M. Dunbar and S. Keten, *Macromolecules*, 2020, **53**, 9397–9405.
43 X. Bian, C. Kim and G. E. Karniadakis, *Soft Matter*, 2016, **12**, 6331–6346.
44 Y. Yoshimoto, Z. Li, I. Kinefuchi and G. E. Karniadakis, *J. Chem. Phys.*, 2017, **147**, 244110.
45 H. Mori, *Prog. Theor. Phys.*, 1965, **33**, 423–455.
46 R. Zwanzig, *J. Stat. Phys.*, 1973, **9**, 215.
47 R. Zwanzig, *Nonequilibrium statistical mechanics*, Oxford University Press, 2001.
48 R. Kubo, *Rep. Prog. Phys.*, 1966, **29**, 255.
49 P. Linz, *Analytical and Numerical Methods for Volterra Equations*, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1985.
50 P. K. Lamm, in *Surveys on Solution Methods for Inverse Problems*, ed. D. Colton, H. W. Engl, A. K. Louis,

J. R. McLaughlin and W. Rundell, Springer, Vienna, 2000, pp. 53–82.
51 C. Groetsch, in *Handbook of Mathematical Methods in Imaging*, ed. O. Scherzer, Springer, New York, 2015, pp. 3–41.
52 A. N. Tikhonov, *Sov. Math. Dokl.*, 1963, **4**, 1624–1627.
53 A. N. Tikhonov, *Sov. Math. Dokl.*, 1963, **4**, 1035–1038.
54 A. N. Tikhonov and V. B. Glasko, *USSR Comput. Math. Math. Phys.*, 1964, **4**, 236–247.
55 A. N. Tikhonov and V. B. Glasko, *USSR Comput. Math. Math. Phys.*, 1965, **5**, 93–107.
56 M. Ceriotti, G. Bussi and M. Parrinello, *J. Chem. Theory Comput.*, 2010, **6**, 1170–1180.
57 A. Brünger, C. L. Brooks III and M. Karplus, *Chem. Phys. Lett.*, 1984, **105**, 495–500.
58 C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT Press, 2006.
59 D. C. Liu and J. Nocedal, *Math. Prog.*, 1989, **45**, 503–528.
60 Z. Ma and W. Pan, *Comput. Methods Appl. Mech. Eng.*, 2021, **373**, 113495.
61 S. Plimpton, *J. Comput. Phys.*, 1995, **117**, 1–19.
62 J. Sun and R. N. Zuckermann, *ACS Nano*, 2013, **7**, 4715–4732.
63 H. Jin, F. Jiao, M. D. Daily, Y. Chen, F. Yan, Y.-H. Ding, X. Zhang, E. J. Robertson, M. D. Baer and C.-L. Chen, *Nat. Commun.*, 2016, **7**, 1–8.
64 N. Gangloff, J. Ulbricht, T. Lorson, H. Schlaad and R. Luxenhofer, *Chem. Rev.*, 2016, **116**, 1753–1802.
65 X. Ma, S. Zhang, F. Jiao, C. Newcomb, Y. Zhang, A. Prakash, Z. Liao, M. Baer, C. Mundy, J. Pfaendtner, A. Noy, C. Chen and J. D. Yoreo, *Nat. Mater.*, 2017, **16**, 767–774.
66 F. Jiao, X. Wu, T. Jian, S. Zhang, H. Jin, P. He, C.-L. Chen and J. J. De Yoreo, *Angew. Chem., Int. Ed.*, 2019, **58**, 12223–12230.
67 J. H. Kim, E. M. Grzincic, L. Yun, R. K. Spencer, M. A. Kline and R. N. Zuckermann, *Soft Matter*, 2020, **16**, 907–913.
68 B. Cai, Z. Li and C.-L. Chen, *Acc. Chem. Res.*, 2021, **54**, 81–91.
69 Y. Duan, C. Wu, S. Chowdhury, M. C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo and T. Lee, *et al.*, *J. Comput. Chem.*, 2003, **24**, 1999–2012.
70 J. Wang, W. Wang, P. A. Kollman and D. A. Case, *J. Mol. Graphics Modell.*, 2006, **25**, 247–260.
71 J. Wang, R. M. Wolf, J. W. Caldwell, P. A. Kollman and D. A. Case, *J. Comput. Chem.*, 2004, **25**, 1157–1174.
72 H. Berendsen, J. Postma, W. Van Gunsteren and A. J. Hermans, *Intermolecular Forces*, 1981.
73 M. J. Abraham, T. Murtola, R. Schulz, S. Páll, J. C. Smith, B. Hess and E. Lindahl, *SoftwareX*, 2015, **1**, 19–25.
74 M. Parrinello and A. Rahman, *J. Appl. Phys.*, 1981, **52**, 7182–7190.
75 G. Bussi, T. Zykova-Timan and M. Parrinello, *J. Chem. Phys.*, 2009, **130**, 074101.