UNIVERSITY OF CALIFORNIA

SANTA BARBARA

Scalable and Reliable Gaussian Processes for Uncovering Physical Patterns from Experiments and Simulations

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

STATISTICS AND APPLIED PROBABILITY

by

Xinyi Fang

Committee in charge:

Professor Mengyang Gu, Chair

Professor Yuedong Wang

Professor Michael Ludkovski

Professor M. Scott Shell

June 2026

The Dissertation of Xinyi Fang is approved:

Professor Yuedong Wang

Professor Michael Ludkovski

Professor M. Scott Shell

Professor Mengyang Gu, Chair

April 2026

To me, myself, and I,

to everything that brought us here,

and of course, Lychee.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Professor Mengyang Gu, for his mentorship, encouragement, and confidence in my potential throughout my doctoral studies. He devoted tremendous time and care to my growth as a researcher, guided me through every project we worked on together, and supported me during moments of self-doubt. I am truly grateful for everything I learned from him, both about research and beyond. My sincere thanks also go to my committee members, Professor Yuedong Wang, Professor Michael Ludkovski, and Professor Scott Shell, for their time, thoughtful feedback, and insightful discussions.

I am especially grateful to Professor Jianzhong Wu for the opportunity to work together on my first research project; his kindness and encouragement left a lasting impression on me. I also feel fortunate to work with many wonderful collaborators, including Professor Yimin Luo, Professor Javier Read de Alaniz, Dr. Saejin Oh, Elizabeth A. Murphy, Clayton Ellis, Runtong Pan, and others. Their insights have greatly enriched my research experience.

I would also like to thank Xubo Liu and Yizi Lin for the many conversations we shared throughout this journey, which were a great source of inspiration and comfort.

I gratefully acknowledge the support of the National Science Foundation under Awards No. OAC-2411043, DMR-2308708, and DMR-1933487; the UC Multicampus Research Programs and Initiatives (MRPI) under Award No. M23PL5990; the Hellman Fellowship; the Heeger Fellowship Award; the Graduate Division Dissertation Fellowship; and the OBayes Travel Award.

Above all, I am immensely thankful to my parents, Qinfang Li and Jincai Fang, and my grandmother, Fengxia Wang, for their unconditional love and support throughout my life. Without them, I would not be where I am today. I thank my best friend, Xinyi Niu,

for always believing in me and being there during the hardest moments. I am also grateful to Susan Farber for her invaluable perspective during difficult times. I am truly thankful to my boyfriend, Lang Wang, for his patience and understanding, and for always being by my side. To my cat, Lychee, thank you for the company and joy you brought me every day. Lastly, I thank myself for making it through this journey.

Curriculum Vitae

Xinyi Fang

Education

2026	Ph.D. in Statistics and Applied Probability, University of California, Santa Barbara.
2020	M.S. in Statistics, University of Southern California.
2019	B.S. in Applied and Computational Mathematics, University of Southern California.

Experience

06/2025–09/2025	Statistics Intern, GlaxoSmithKline.
2023–2025	Graduate Student Researcher, Department of Statistics and Applied Probability, University of California, Santa Barbara.

Publications

1. Oh, S., Fang, X., Lin, I. H., Dee, P., Dunham, C. S., Copp, S. M., Doyle, A. G., Read de Alaniz, J., & Gu, M. (2026). Synergizing chemical and AI communities for advancing laboratories of the future. *ACS Central Science*, 12(2), 157-173.
2. Ellis, C., Fang, X., Balzer, C., Quah, T., Shell, M. S., Fredrickson, G. H., & Gu, M. (2025). Fast phase prediction of charged polymer blends by white-box machine learning surrogates. *Macromolecules*, 59(1), 50-59.
3. Pan, R., Fang, X., Azizzadenesheli, K., Liu-Schiaffini, M., Gu, M., & Wu, J. (2025). Neural Operators for Forward and Inverse Potential-Density Mappings in Classical Density Functional Theory. *The Journal of Chemical Physics*, 163 (16), 164120.
4. Fang, X., & Gu, M. (2025). The inverse Kalman filter. *Biometrika*, 112(4), asaf054.
5. Fang, X., Murphy, E. A., Kohl, P. A., Li, Y., Hawker, C. J., Bates, C. M., & Gu, M. (2025). Universal Phase Identification of Block Copolymers From Physics-Informed Machine Learning. *Journal of Polymer Science*, 63(6), 1433-1440.
6. Gu, M., Fang, X., & Luo, Y. (2023). Data-driven model construction for anisotropic dynamics of active matter. *Physical Review X Life*, 1, 013009.
7. Luo, Y., Gu, M., Park, M., Fang, X., Kwon, Y., Uruena, J.M., Read de Alaniz, J., Helgeson, M., Marchetti, M.C., & Valentine, M. (2023). Molecular-scale substrate anisotropy and crowding drive long-range nematic order of cell monolayers. *Journal of the Royal Society Interface*. 20 (204), 20230160.
8. Fang, X., Gu, M., & Wu, J. (2022). Reliable emulation of complex functionals by active learning with error control. *The Journal of Chemical Physics*. 157(21), 214109.
9. Gu, M., Liu, X., Fang, X., & Tang, S. (2022). Scalable Marginalization of Correlated Latent Variables with Applications to Learning Particle Interaction Kernels. *The New England Journal of Statistics in Data Science*, 1(2), 172-186.

Academic Presentations

1. Machine learning for polymer phase discovery. BioPACIFIC All-Hands Meeting, UCSB, CA, May 2025. (Talk)
2. Machine learning for polymer phase discovery. MRSEC Site Visit, UCSB, CA, Apr 2025. (Talk)
3. The inverse Kalman filter. IMSI Workshop on Kernel Methods in Uncertainty Quantification and Experimental Design, Chicago, IL, Apr 2025. (Poster)
4. Predictive models for computational physics and materials science: Gaussian processes vs deep neural networks. MRSEC IRG-1 Seminar, UCSB, CA, Mar 2025. (Talk)
5. Reliable emulation of complex functionals by active learning with error control. UCSB Graduate Student Research Lightning Talks, UCSB, CA, Feb 2023, 2024, 2025. (Talk)
6. Universal phase identification of block copolymers from physics-informed machine learning. MRSEC IRG-1 Seminar, UCSB, CA, Nov 2024. (Talk)
7. Reliable emulation of complex functionals by active learning with error control. 2024 Joint Statistical Meetings (JSM), Portland, OR, Aug 2024. (Poster)
8. Reliable emulation of complex functionals by active learning with error control. American Physical Society (APS), Minneapolis, MN, Mar 2024. (Talk)
9. Universal phase identification of diblock copolymers by automated statistical learning of morphology. BioPACIFIC All-Hands Meeting, UCLA, CA, Dec 2023. (Talk)
10. Non-Gaussian and anisotropic fluctuations mediate the progression of global cellular order: a data-driven study. BioPACIFIC Monthly Meeting, UCSB, CA, Mar 2023. (Talk)
11. Reliable emulation of complex functionals by active learning with error control. O'Bayes 2022: Objective Bayes Methodology Conference, UCSC, CA, Sep 2022. (Poster)

Software

- Gu, M., Fang, X., and Lin, Y. *FastGaSP*: Fast and Exact Computation of Gaussian Stochastic Process, 2026. R package version 0.6.3.

Awards

2024	Graduate Division Dissertation Fellowship
2024	Heeger Fellowship Award
2023-2024	NSF BioPACIFIC Materials Innovation Platform Fellowship
2022	Objective Bayes Methodology Conference (OBayes) Travel Award
2022	Abraham Wald Memorial Prize
2016-2019	Dornsife Dean's List

Abstract

Scalable and Reliable Gaussian Processes for Uncovering Physical Patterns from Experiments and Simulations

by

Xinyi Fang

Gaussian processes (GPs) provide a fundamental tool for statistical inference and experimental design in scientific computing, which provides an important framework for quantifying uncertainty. However, two challenges remain for Gaussian process models of complex real-world data. First, computational and storage costs are prohibitively large when dealing with problems requiring massive amounts of data. Second, reliable predictions, inverse estimation, and feature selection for data with high-dimensional inputs or large noise. This thesis addresses these two challenges by developing novel, efficient statistical approaches and fast algorithms that address these problems.

Chapter 1 systematically reviews GPs and their theoretical connections to dynamic linear models and the Kalman filter, which form the foundation of the subsequent scalable inference methods, and active learning strategies for prediction and optimization, particularly for problems with a high-dimensional input space. Chapter 2 introduces the inverse Kalman filter, a novel algorithm that reduces the computational complexity of covariance matrix-vector multiplication from quadratic to linear order, for any covariance matrix induced by dynamic linear models, a broad class of models that include GPs having Matérn covariance with one-dimensional input and half-integer roughness parameter. By integrating the inverse Kalman filter with the conjugate gradient algorithm, we are able to compute the predictive distribution for GPs with a class of additive covariance matrix scalably, with applications in accurately estimating kernels between a large number of interacting particles over a

long period of time and interpolating incomplete lattices. Chapter 3 develops data-driven statistical methods for modeling collective cell motion, guiding feature selection in adaptive model construction through composite hypothesis testing to achieve effective parameterized modeling. Chapter 4 proposes a physics-informed machine learning method for automatically identifying the phase structure of block copolymers. The selected feature set is independent of the chemical composition of the materials, allowing the model to generalize to block copolymers with new monomers. Chapter 5 proposes an active learning framework with probabilistic error control, applicable to high-dimensional input space, providing a reliable solution for surrogate modeling of expensive simulations. Finally, Chapter 6 provides an outlook on future research directions.

Table of Contents

Dedication iv

Curriculum Vitae vii

Abstract ix

List of Figures xiv

List of Tables xxi

1 Introduction 1
- 1.1 Gaussian processes . . . 5
  - 1.1.1 Gaussian processes with scalar outputs . . . 5
  - 1.1.2 Gaussian processes with vectorized outputs . . . 10
- 1.2 Dynamic linear models and Kalman filter . . . 13
- 1.3 Outline . . . 16

2 The inverse Kalman filter 19
- 2.1 Introduction . . . 20
- 2.2 Inverse Kalman filter . . . 22
- 2.3 Nonparametric estimation of particle interaction functions . . . 26
- 2.4 Numerical results for particle interaction function estimation . . . 30
  - 2.4.1 Evaluation criteria . . . 30
  - 2.4.2 Unnormalized Vicsek model . . . 32
  - 2.4.3 A modified Vicsek model with multiple interactions . . . 35
  - 2.4.4 Estimating cell-cell interactions on anisotropic substrates . . . 37
- 2.5 Implementation of Inverse Kalman Filter in `FastGaSP` . . . 40
  - 2.5.1 Core IKF functions . . . 40
  - 2.5.2 Application to particle interaction estimation . . . 42

3 Data-driven model construction for anisotropic dynamics of active matter 45
- 3.1 Introduction . . . 46
- 3.2 Feature selection from experimental results . . . 52
  - 3.2.1 Experimental setting: In vitro cell alignment experiment . . . 52
  - 3.2.2 Permutation F-test on variances of the velocities . . . 55
  - 3.2.3 Shapiro-Wilk test for normality of velocity distribution . . . 57
  - 3.2.4 Neighboring feature construction for cellular interaction . . . 58

3.2.5 Reduced magnitude of local alignment . . . 60
3.2.6 Summary of data-driven findings and model development . . . 60
3.3 Model constructed from selected features and maximum likelihood estimation 61
3.3.1 Maximum likelihood estimator with Laplace fluctuation . . . 63
3.3.2 Maximum likelihood estimator with generalized Gaussian distribution 64
3.4 Results . . . 65
3.4.1 Model comparison by simulated orientational order parameter and velocity variability . . . 65
3.4.2 Sensitivity analysis of the simulation . . . 70

4 Universal phase identification of block copolymers from physics-informed machine learning 74
4.1 Introduction . . . 75
4.2 Results and discussion . . . 79
4.2.1 Generating block copolymer libraries . . . 79
4.2.2 Filtering and feature extraction . . . 80
4.2.3 Phase identification of new block copolymer chemistries . . . 83
4.2.4 Phase identification of mixed block copolymer chemistries . . . 87

5 Reliable emulation of complex functionals by active learning with error control 89
5.1 Introduction . . . 90
5.2 Classical density functional theory . . . 93
5.3 A reliable Gaussian process emulator by active learning with error control . 97
5.3.1 Parallel partial Gaussian process emulation for vectorized outputs . 97
5.3.2 ALEC: active learning with error control approach . . . 98
5.3.3 Sequential update of the ALEC emulator . . . 103
5.3.4 Algorithm and the computational cost . . . 104
5.4 Numerical Results . . . 106
5.4.1 Particle density and grand potential prediction for each group of external potential field separately . . . 108
5.4.2 Predicting particle densities and grand potentials for all groups of external potentials . . . 110
5.4.3 Predicting a new class of external potential . . . 113

6 Conclusion and future directions 116
6.1 Extensions of the inverse Kalman filter . . . 118
6.2 Future work on active learning . . . 119
6.2.1 Extensions of the Vecchia approximations in high dimensions . . . 120

A Appendix of Chapter 1 123
A.1 Derivation of the predictive t-distribution . . . 123
A.2 Proof . . . 126
A.3 Gaussian processes with half-integer Matérn covariances as dynamic linear models . . . 128

B Appendix of Chapter 2 132
B.1 Proofs . . . 132
B.2 The conjugate gradient method . . . 135
B.3 Scalable computation of the predictive distribution in particle dynamics . . 135

B.4 Application of predicting incomplete lattice for correlated data 140
B.5 Parameter estimation 141
B.6 Computational complexity 144
B.7 Additional numerical results for estimating interaction functions 147
B.7.1 Unnormalized Vicsek model 147
B.7.2 Modified Vicsek model 149
B.8 Numerical results of predicting incomplete lattice data 152
B.8.1 Evaluation criteria 152
B.8.2 Branin function 153
B.8.3 Interferometric synthetic aperture radar interferogram 155

C Appendix of Chapter 3 **158**
C.1 Maximum likelihood estimator with Laplace fluctuation 158
C.2 Maximum likelihood estimator with generalized Gaussian fluctuation 159
C.3 Details on simulation 162
C.4 Additional results on experimental data 163
C.5 Additional results on simulation using different neighbor radii 165

D Appendix of Chapter 4 **167**
D.1 Methods 167
D.1.1 Materials 167
D.1.2 Molecular Characterization 168
D.1.3 Procedure for the synthesis of 2-fluoroethyl acrylate 169
D.1.4 General procedure for the synthesis of poly(dodecyl acrylate) homopolymer 169
D.1.5 General procedure for the synthesis of fluorinated diblock copolymers 170
D.1.6 General procedure for fractionation of diblock copolymers via automated flash chromatography 171
D.1.7 Logarithmic space transformation 171
D.1.8 Noise filtering with Kalman filter 172
D.1.9 Peak detection and feature selection 173
D.1.10 Machine learning models for classification 175
D.2 More numerical results of machine learning models for classification 176
D.2.1 Phase identification of new chemical groups 176
D.2.2 Phase identification of mixed chemical groups 179
D.3 Further inspection of misclassified samples by quantified uncertainty 182

E Appendix of Chapter 5 **186**
E.1 Analytical forms in cDFT 186
E.2 External potentials and chemical potentials 186
E.3 Derivation of probabilistic upper bounds 188
E.4 Details of the D-optimality criterion 190
E.5 Out of sample prediction results of particle density and grand potential . . 191

Bibliography **194**

List of Figures

1.1 The dependent structure of the dynamic linear model. 14

2.1 The dependent structure of particle dynamics. 20
2.2 Panels (a) and (b) compare the IKF (blue dots) with direct computation (red triangles) for calculating $\mathbf{\Sigma u}$ in the original time scale and logarithmic scale (base 10), respectively. Panel (c) shows the predictive error comparison between the non-robust IKF (light blue squares) and the robust IKF (blue dots). 24
2.3 (a) Computational time of calculating predictive means using the IKF-CG algorithm (blue dots), CG method (purple squares), and direct computation (red triangles) for the unnormalized Vicsek model with different numbers of observations. (b), (c) NRMSE of predictive means for the interaction function and log-likelihood computed via direct computation (red triangles) and the IKF-CG algorithm (blue dots), with differences (yellow diamonds) of orders 10^{-7} to 10^{-5} and 10^{-5} to 10^{-4}, respectively. 32
2.4 Boxplots of NRMSE in (2.17) for estimating the latent interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.1^2$ based on 20 experiments. Panel (a) uses the Matérn covariance with roughness parameter 5/2, and panel (b) uses the exponential covariance. 33
2.5 Uncertainty assessment of predicted interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.1^2$ using (a) Matérn covariance function in (1.2) and (b) exponential covariance function. Bars represent average lengths of 95% posterior credible intervals (2.18). Dots indicate the proportions covered in the 95% intervals (2.19), with the optimal coverage (0.95) shown as the purple dashed lines. 33
2.6 (a) Predicted (blue dashed lines) and true (pink solid lines) interaction functions with the 95% posterior credible interval (grey shaded area) for the unnormalized Vicsek model with $n_\tau = 900$, $T = 10$, and $\sigma_0^2 = 0.1^2$, using Matérn covariance function with $\nu = 5/2$. (b) Trajectories of 45 randomly sampled particles over 50 time frames using estimated (blue dashed lines) and true (pink solid lines) interaction functions with identical initial positions and noise samples. 34
2.7 Predictions (blue dashed curves), truth (pink solid curves), and the 95% posterior credible interval (shaded regions) of particle interaction functions when $n_p = 900$, $n_\tau = 10$, and $\sigma_0^2 = 0.1^2$. 37

2.8 The blue dashed curves show the predicted interaction function for the horizontal and vertical directions. The grey shaded area is the 95% posterior credible interval. The purple line of slope 1 is the prediction using the unnormalized Vicsek model. The light pink histogram shows the velocity distribution of all training samples. 38

3.1 (a) Schematics of the two types of substrates (isotropic and nematic) tested in this study. On the nematic substrate, the molecular alignment (denoted by a red double-sided arrow) is parallel to x. (b) A stitched view of a snapshot of long-term cell imaging where the cell cytoplasm is labeled in red. (c) Correlation plots in exploratory data analysis, where x, y velocity components are plotted against those of the neighboring average in the previous step. The slope of the fit (red solid lines) is smaller than 1 (black dashed lines). (d) Parameter estimation of the slope parameters, ω, the slope value of the red solid line in (c), and their 95% confidence intervals (shaded area). The statistical tests yield four potential features (green blocks), representing new features added to the classical Vicsek model in fitting the current data set. (e) The grid-based system to search for the neighbors. In order to find the neighbor of the target cell i (circled in red), a radius of r will be searched. The computational procedure is simplified by only searching through neighboring squares connected to the square cell i is located in. This workflow generates various physical parameters via simulation to compare to quantities computed from the experiment. 50

3.2 Parameters estimated from the data set. (a) The standard deviation of the cell velocities is shown in yellow circles and green triangles. The light-colored histogram denotes the cell number count (b-c). The velocity distribution at a representative time = 20 hours. The dashed black lines denote the best fit normal distribution. The solid orange lines denote the best fit Laplace distribution. (d) The blue squares denote the polar order parameter, and the polar histograms of the velocity angle at two different time points are plotted in (e-f). 54

3.3 A permutation F-test shows that the variance of the fluctuation along x and y directions is indeed unequal at time =20 hrs. 56

3.4 Shapiro-Wilk test for normality. The figure shows the p-value of individual frames, which is a statistical measurement applied to validate whether or not the observed data follows a normal distribution. A small p-value indicates statistical significance. The inset shows a representative quantile-quantile (QQ) plot at time = 20 hours to compare the sample quantiles of normalized velocity with the theoretical quantiles from the standard normal distribution. The black curve indicates the linear curve with slope 1 (normal). 58

3.5 Effects of accounting for different neighbor ensembles. (a-c) Parameters and correlation plots using the neighbor ensemble of the classical Vicsek model for time = 20 hours. (d-f) The scenario where only cells traveling in the same direction are considered. [(b), (c), (e), and (f)] The correlation between the cell at t versus the mean of the velocity of its neighbors at $t-1$. (g) Correlations between velocity $v_{i,x}(t)$, $v_{i,y}(t)$ of any cell i and mean velocity of the neighboring cells from the previous step $\bar{v}_{i,x}(t-1)$, $\bar{v}_{i,y}(t-1)$, where the 95% confidence intervals, shown in fainted shades, are computed based on Fisher transformation of correlation coefficients. Two scenarios are shown, either by including all neighbors (bottom 2 curves), or only the same direction neighbors (top 2 curves). (h) The estimated weights, equivalent to the fitted slopes, $\hat{\omega}_x(t)$, $\hat{\omega}_y(t)$ from these two scenarios and their confidence intervals are plotted. 59
3.6 Reproducing orientational order parameters and mean absolute deviation of velocity with different simulation models at a representative radius $r = 75$ μm. The left, middle, right panels compare simulations using the Gaussian, Laplace, GGD fluctuation with experimental observations. For each type of fluctuation distributions, simulations of different interaction rules are presented. (a-c) show the orientational order parameter, (d-f) show the average absolute deviation of the velocities. 65
3.7 Distributions of random fluctuations between the observations and simulated models at a representative radius $r = 75$ μm and time = 20 hours with the distributions of the fluctuation in the simulation following Gaussian (a,d), Laplace (b,e) and GGD (c,f). 69
3.8 Change of order parameters due to the change of simulation parameters with $\tau_x = 0.02$ and $w_x = 0.7$. The color bar corresponds to the simulated order parameter at long times. The dashed lines and circles denote the actual path of the experimental results plotted against the phase diagram. The colors of the markers correspond to the order parameter as denoted by the color bar. Red and blue X's denote the beginning and the end of the trajectory, respectively. 72

4.1 A library of 364 well-ordered diblock copolymers derived from the synthesis and separation of only 16 parent copolymer samples was used to evaluate the machine-learning algorithms developed herein. Overall block copolymer molecular weight is denoted by M_{n}. Volume fraction of the semi-fluorinated block is denoted by f_{F}. Color indicates the morphology as determined by manual analysis of SAXS data. 79
4.2 Panels display a representative example of each morphology. Black and blue curves represent the original log intensity and the smoothed intensity curves, respectively. Detected peak locations are highlighted by red triangles and marked with orange dashed lines. 80
4.3 (a) T-distributed stochastic neighbor embedding (t-SNE) for visualization of original log intensity. (b) First peak location vs. the ratio between second and first peak. (c) First peak location vs. the ratio between third and second peak. 82
4.4 Maximum predicted probability for correctly and incorrectly predicted morphologies for (a) the D-9F block copolymer library; (b) the D-12F block copolymer library using PIMF-RF. 85

4.5 Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using the physics-informed features for (a) the original samples; (b) revised samples after correcting three mislabeled detected by our method. 88

5.1 (a) Functional learning in the context of cDFT for one-dimensional hard rods of uniform size a in an external field $V^{ext}(.)$ defined in a domain of size L. (b)-(c) t-distributed stochastic neighbor embedding (t-SNE) for visualization of input $\mu_{chem} - V^{ext}(.)$ (middle panel) and output $\rho(.)$ (right panel). For each group, 2000 density profiles are generated based on different combinations of μ_{chem} and parameters in $V^{ext}(.)$. 93
5.2 An overview of the active learning with error control framework. 98
5.3 Empirical CDF of absolute error of particle densities $|\rho_j(\mathbf{x}_i^*) - \hat{\rho}_j(\mathbf{x}_i^*)|$ for 4 classes of input functions at all grid points using the criterion (5.8) with different error bounds δ and probability tolerance thresholds α. The horizontal line and the vertical line are $1-\alpha$ and δ, respectively. On the x-axis, the $1-\alpha$ percentiles of the absolute errors are recorded, all of which are less than δ. 99
5.4 The bar plots and line plots show the out-of-sample RMSE of density ρ, and RMSE of grand potential Ω, respectively. The ALEC emulator is compared with other methods with three other designs: RS1, RS2, and D-opt (Active learning with D-optimality). For the ALEC emulator, we use $\delta = 0.01$ and $\alpha = 0.05$ as the threshold, and the augmented sizes for groups 1-4 are 0, 7, 21, and 504, respectively. In D-opt, the augmented sizes for groups 1-4 are 0, 7, 21, and 582, respectively. The number of training sample sizes from RS1 and RS2 are the same as the initial sample size and terminal sample size in ALEC. 108
5.5 A comparison of four surrogate models for predicting the group 4 functions: Active learning with error control (ALEC), random sampling with the number of training points as the initial number of training points in ALEC (RS1) and with the number of training points as the final number of training points in ALEC (RS2), and active learning with D-optimality (D-opt). (a) Boxplots of predictive RMSE of each predicted density in the testing dataset. (b) Predicted grand potential $\hat{\Omega}$ vs. the truth Ω. (c)-(d) True and predicted densities for two selected density profiles. 109
5.6 Out-of-sample RMSE of particle density ρ (bar plots) and grand potential Ω (line plots) for active learning with error control, RS1 design, RS2 design and active learning with D-optimality. The overall RMSE for combined all groups (solid bars and dots) and the RMSE of each individual group (shadow bars and dots) are both recorded. ALEC contains 80 inputs as initial training size, and the total number of augmented samples in the ALEC emulator ($\delta = 0.01$, $\alpha = 0.05$) is 873, with 0, 1, 58, and 814 augmented samples in groups 1 to 4, respectively. In D-opt, the total number of augmented samples is 935, and the augmented sizes for groups 1-4 are 59, 73, 393, and 410, respectively. RS1 contains a total of the first 80 training samples after uniformly mixing all four groups of density together. RS2 contains the first 953 training inputs, with 245, 221, 238, and 249 for groups 1-4, respectively. Since the ALEC emulator identifies the hardest region (group 4) to be predicted, thereby augmenting most samples from group 4, it has the smallest predictive error overall and predictive errors of all groups are controlled. 111

5.7 The running time for each function group using the ALEC algorithm and using numerical solver (NS). As the running time for building the GP model in ALEC is very small, the bars for GP are nearly invisible. Other approaches, such as RS2 and D-opt, have similar computational costs as ALEC as the training and augmented sample sizes are similar. 113
5.8 Prediction results from the ALEC emulator on a new class of external potentials where ALEC models are trained by the previous four functional groups separately and by all functional groups together. $\delta = 0.01$ and $\alpha = 0.05$ are used. Left panel: the empirical CDF of absolute error using the average variance selection criterion in Equation (5.8). The number of augmented samples using the model for groups 1-4 and all groups combined are 92, 122, 67, 48, and 15, respectively. Middle panel: the percentage that the absolute difference is greater than δ for each coordinate, i.e. $\sum_{i=1}^{n*} \mathbb{1}\{\rho_j(\mathbf{x}_i^*) - \hat{\rho}_j(\mathbf{x}_i^*) > \delta\}/n^*, j = 1, ..., k$. Right panel: the empirical CDF of absolute error using the maximum variance selection criterion in Equation (5.5). The number of augmented samples using the model for groups 1-4 and all groups combined are 408, 395, 384, 362, and 263, respectively. 114

B.1 (a) Illustration for preprocessing the particles into coarse-grained grids. The domain is divided into square grid cells with side lengths equal to or greater than the maximum interaction radius. For a given particle (red dot), as only particles in the current and eight adjacent grid cells (orange square region) need to be considered, it substantially accelerates the computation. Particles (blue dots) within the interaction radius (the dashed circle) and itself (red dot) can be identified as neighbors. (b) Visualization of velocity alignment in the unnormalized Vicsek model. Each red particle updates its velocity $v(\tau) = [v_1(\tau), v_2(\tau)]^T$ by aligning with all neighboring particles, including itself, within interaction radius r. While all particles simultaneously update their directions, this illustration focuses on the movement of the red particle. 137
B.2 Boxplots of NRMSE (2.17) for estimating the latent interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.2^2$. Results are based on 20 experiments for each scenario, using the Matérn covariance with roughness parameter 5/2 (Panel (a)) and exponential covariance (Panel (b)). 147
B.3 Uncertainty assessment of the predicted interaction function of the unnormalized Vicsek model using the Matérn covariance function with roughness parameter 5/2 (Panel (a)) and the exponential covariance function (Panel (b)), both with $\sigma_0^2 = 0.2^2$. The bars represent the mean length of the 95% posterior credible interval (2.18) over 20 experiments, and the dots represent the average percentage of test data covered by 95% posterior credible interval (2.19) across 20 experiments. The purple dashed line at 0.95 indicates the optimal coverage level for the dots. 148
B.4 Boxplots of the estimated radius distance $\hat{r}$ in the unnormalized Vicsek model. Panels (a) and (c) use the Matérn covariance with a roughness parameter 5/2 for $\sigma_0^2 = 0.1^2$ and $\sigma_0^2 = 0.2^2$, respectively, while panels (b) and (d) use the exponential covariance for the same noise levels. The purple dashed lines mark the true radius distance $r = 0.5$. 148

B.5 The 95% credible intervals for the estimated interaction radius in the unnormalized Vicsek model, obtained using the Matérn 5/2 covariance and $B = 100$ residual bootstrap samples. Results are based on 20 simulations for $\sigma_0^2 = 0.1^2$ and $\sigma_0^2 = 0.2^2$, with $n_p = 100$ and $n_\tau = 5$. The dots represent the estimated radius $\hat{r}$ from the original data, and the purple dashed lines represent the true radius $r = 0.5$. 149
B.6 (a) Observed data from a simulation of the Branin function with the first disk missing. (b) Latent mean of the Branin function. (c) Predictions obtained using the IKF-CG algorithm. 154
B.7 Violin plots illustrating the NRMSE of missing regions for different methods under 20% disk missing with the first center (panel (a)), 20% disk missing with the second center (panel (b)), and 20% random missing (panel (c)). The point on each violin represents the mean NRMSE for each method. 155
B.8 (a) Observed data from interferograms with the 25% random missing. (b) True values of the interferogram data. (c) Predictions obtained using the IKF-CG algorithm. 155
B.9 The NRMSE of the missing regions in disk missing (panel (a)) and random missing (panel (b)) scenarios for interferograms. 156
B.10 Total computational time for the missing regions in disk missing (panel (a)) and random missing (panel (b)) scenarios for interferograms. 156

C.1 Parameter estimation with interval (shaded) for a Generalized Gaussian distribution following Equation (3.5) along x-direction (yellow), and along y-direction (green). 161
C.2 (a-b) Change of order parameters due to the change of simulation parameters with $\tau_x = 0.01$ in (a) and $\tau_x = 0.04$ in (b), which cover the range of the observed τ's. $w_x = 0.7$ are assumed for both cases. Red and blue X's denote the beginning and the end of the trajectory. 162
C.3 Several typical simulation progressions, where the first 20 steps are regarded as burn-ins and left out in computing the averages in the phase diagram. . 163
C.4 Reproducing orientational order parameter (a-c) and average absolute deviation of velocities (d-f) from different simulation models at a representative radius $r = 75$ μm for experiments of cell migrating on a nematic substrate when starting at a higher initial density. 164
C.5 Reproducing orientational order parameter (a-c) and average absolute deviation of velocities (d-f) with different simulation models at a representative radius $r = 75$ μm for experiments of cells moving on an isotropic substrate, which do not align, and have isotropic velocity. 165
C.6 Reproducing orientational order parameters with different radii. The left, middle, and right panels compare simulations using the Gaussian, Laplace, and GGD fluctuation with experimental observations. Open square symbols show the order parameter calculated from experimental data, dark blue solid line, yellow dash-dotted line, and red dashed line represent simulation results of different radii 75, 50, and 25 μm. 166

D.1 Variable importance of the PIMF-RF model in predicting the third group (left) and the fourth group (right). 176
D.2 Maximum predicted probability for correctly and incorrectly predicted samples of D-9F group (left) and D-12F group (right) using PIMF-bagging model. . 177

D.3 Variable importance of PIMF-bagging in predicting the third group (left) and the fourth group (right). 178
D.4 Maximum predicted probability for correctly and incorrectly predicted samples of D-9F group (left) and D-12F group (right) using PIMF-XGBoost model. 178
D.5 Variable importance of PIMF-XGBoost in predicting the third group (left) and the fourth group (right). 179
D.6 Three mislabeled samples during manual assignment in the dataset. 179
D.7 Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using PIMF-bagging model. 181
D.8 Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using PIMF-XGBoost model. 181
D.9 (a) Two incorrectly predicted samples using PIMF-RF for the third group, D-9F. (b) Three incorrectly predicted samples for the fourth group, D-12F. (c) Two representative SAXS curves for the phases BCC, HEX, LAM, σ, and DIS. The black dots and blue curves represent the original log intensity and the smoothed intensity curves, respectively. Detected peak locations are marked with yellow dotted lines. 183
D.10 The value of each PIMF in the training and testing set when predicting the third group, D-9F. 184
D.11 The value of each PIMF in the training and testing set when predicting the last group, D-12F. 184

E.1 Each plot gives 5 examples of generated densities from external potential in E.4-E.7. 187

List of Tables

1.1 Commonly used kernel functions, with $c(x_i, x_j) = c(d)$ and $d = |x_i - x_j|$. In the Matérn kernel, $\Gamma(\cdot)$ is the gamma function and $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of the second kind. 6

2.1 The predictive accuracy and uncertainty assessment by (2.17)-(2.19) for the modified Vicsek model with $\sigma_0^2 = 0.1^2$ using Matérn covariance function and roughness parameter $\nu_j = 5/2$. 36

2.2 One-step-ahead prediction performance on the testing dataset. Here RMSE = $\{\sum_{\tau=1}^{n_\tau^*} \sum_{i'=1}^{n_p(\tau)} (\hat{v}_{i',l}(\tau) - v_{i',l}(\tau))^2 / \sum_{\tau=1}^{n_\tau^*} n_p(\tau)\}^{1/2}$, with n_τ^* denoting the number of testing time frames. L(95%) and P(95%) are computed similarly to those in (2.18) - (2.19), but with the predictive interval of z replaced by that of y, where y represents the velocity in each direction. 39

3.1 RMSE_S for the orientational order parameter estimates. All assumed the weights parameters from the fluctuation from the x and y directions are separately estimated. The standard deviation of the velocity orientational order parameter is 0.050 and an effective method (highlighted in bold) has RMSE_S smaller than this value. AN (= all neighbors) and SDN (= same direction neighbor) stand for methods with all neighboring cells within the radius and the same direction neighboring cells, respectively. Gau, Lap, and GGD are Gaussian, Laplace, and generalized Gaussian distributions of the fluctuation. 70

3.2 RMSE for the average absolute deviation of the velocity estimates $\tilde{\sigma}_x$ and $\tilde{\sigma}_y$. All assumed the weights parameters from the fluctuation from the x and y directions are separately estimated. The benchmark RMSE of the average absolute deviation (calculated by the standard deviation of the average absolute deviation) for $\tilde{\sigma}_x$, $\tilde{\sigma}_y$ are 5.7×10^{-4} and 5.3×10^{-4}, respectively. An effective method (highlighted in bold) has RMSE smaller than this value. AN (= all neighbors) and SDN (= same direction neighbor) stand for methods with all neighboring cells within the radius and the same direction neighboring cells, respectively. RMSE = (table value) $\times10^{-4}$. 71

4.1 Prediction results of random forest models on predicting D-9F block copolymer morphologies using PIMF-RF, CD-RF, and Curve-RF. 84

4.2 Results of random forest models on predicting D-12F block copolymer morphologies using PIMF-RF, CD-RF, and Curve-RF. 86

B.1 The computational complexity of parameter estimation with M iterations of numerical optimization of the parameters in learning particle interactions with $N_j^{all} = N_j + N_j^*$. The CG and direct computation algorithms involve forming an $N_j \times N_j$ covariance matrix, while the IKF-CG only requires computing the ordered inputs. We assume computing weights $\hat{\mathbf{u}}_j = \mathbf{A}_j^T \boldsymbol{\Sigma}_y^{-1} \mathbf{y}$ is a separate step with complexity shown in the third row of the table. 145
B.2 The predictive accuracy and uncertainty assessment by (2.17)-(2.19) for the modified Vicsek model with $\sigma_0^2 = 0.2^2$ using Matérn covariance function and roughness parameter 5/2. 150
B.3 The root mean squared error (RMSE) of the estimated radius for modified Vicsek model using Matérn covariance function with roughness parameter 5/2, computed as $(\sum_{i=1}^{20}(\hat{r}_i - r)^2/20)^{1/2}$ with $\hat{r}_i$ being the estimated radius of the i-th experiment for each scenario. 150
B.4 The prediction results of modified Vicsek model with exponential covariance function. 151
B.5 The RMSE of the estimated radius for modified Vicsek model using exponential covariance function. 151
B.6 The average computational time in seconds over 20 experiments for estimating the parameters and predicting Branin function. 154

D.1 The confusion matrix of PIMF-bagging in predicting the third group (D-9F) (top) and the fourth group (bottom). 177
D.2 The confusion matrix of PIMF-XGBoost in predicting the fourth group (D-12F) (top) and the fourth group (bottom). 178
D.3 The prediction results of the PIMF-RF, CD-RF [1], and Curve-RF on predicting mixed chemical groups with out-of-sample cross-validation. 180
D.4 The classification results of the bagging model with out-of-sample cross-validation using PIMF. 181
D.5 The prediction results of the XGBoost model with out-of-sample cross-validation using PIMF. 181

E.1 The prediction results for building models on four groups separately. In ALEC emulator, we ues $\delta = 0.01$ and $\alpha = 0.05$ as threshold. In D-opt, we adjust the threshold c_{th} such that it has the similar number of augmented samples in AL, and c_{th} for group 1-4 are, $30, 3.4, 2.4$ and 1.03, respectively. 192
E.2 The prediction results for building models on four groups together. We use $\delta = 0.01$ and $\alpha = 0.05$ in the ALEC emulator. In D-opt, the threshold $c_{th} = 1.12$. For each strategy, we evaluate the overall performance, as well as the performance for each subgroup. 193
E.3 The prediction results of the ALEC approach, using average-sense criterion and maximum-sense criterion, on a new functional form that is not in the training data set, starting with the final GP emulator developed for groups 1-4 and all groups combined. $\delta = 0.01$ and $\alpha = 0.05$ are used. 193

Chapter 1

Introduction

With the development of modern experimental techniques, vast amounts of experimental data can be generated in one experiment, such as a large microscopy video in imaging formation of microscopic patterns, but the number of experiments can be limited due to the time and cost. Thus, translating experimental observations into physical rules and knowledge faces two distinct challenges in practice. First, scalable computational tools are needed to process and analyze the massive data produced within each experiment, especially when the underlying physical or biological mechanisms can be unknown or too complex to model directly [2, 3, 4, 5, 6, 7]. Second, efficient inverse estimation approaches are required to extract and capture underlying patterns across a limited set of experiments [8, 9, 10]. This thesis aims to construct scalable statistical approaches for efficiently estimating system properties, automating model construction, and intelligently selecting informative training data for training surrogate models when physical simulation is available.

Gaussian processes (GPs) [9] are widely used surrogate models for predicting continuous outcomes from small tabular datasets due to its computational efficiency and internal uncertainty quantification [9, 11, 8]. Modern large language models, such as ChatGPT and Gemini, frequently recommend GP for such small-data regression tasks. In

particular, the predictive distribution of a GP provides more than just point predictions [12]. It also yields a principled measure of model confidence that guides the selection of the most informative input points for sampling [13, 14, 15, 16].

Training a GP surrogate for a larger number of observations is challenging due to the computational bottleneck of inverting the covariance matrix, which scales $\mathcal{O}(n^3)$ with the number of samples n, making exact inference prohibitive for large data. To address this, various approximation methods have been proposed. These include low-rank approximations that project GPs onto low-dimensional latent subspaces via inducing points [17, 18], Vecchia-based methods that exploit sparsity in the inverse Cholesky factor through conditional independence [19, 20, 21], and local GP (laGP) that performs inference on a locally optimal subset of observations [22]. Although these techniques accelerate computation, there is a trade-off between computational efficiency and predictive accuracy.

For processes with specific covariance structures, it is possible to achieve exact inference, prediction, and uncertainty quantification with high computational efficiency. When a GP employs certain kernel functions (e.g., the Matérn kernel with a half-integer roughness parameter [23] and one-dimensional inputs), it can be equivalently represented as a dynamic linear model (DLM) [24, 25]. Under this state-space representation, the Kalman filter (KF) algorithm enables exact inference and prediction with linear time complexity $\mathcal{O}(n)$ [26]. While this connection offers substantial advantages for one-dimensional inputs, it does not naturally extend to more complex systems motivated by physical models, where the covariance has an additive structure arising from multiple latent factors. Existing GP approximation methods are also not directly applicable to such cases. To bridge this gap, Chapter 2 introduces an accurate and scalable algorithm for GP inference under this broader class of covariance structures by combining state-space representations with iterative methods.

Beyond computational scalability, many scientific applications require not only accurate predictions but also an understanding of the underlying physical mechanisms. While non-parametric frameworks like GPs are powerful for discovering complex relationships, they often act as "black-box" predictors that do not explicitly account for specific physical or biological laws. In many contexts, such as the study of collective dynamics, parametric modeling remains essential for bridging the gap between empirical observations and physical theory [27, 28, 6]. By constructing models with defined mathematical structures, researchers can evaluate whether these representations align with experimentally observed behaviors. Such alignment provides a systematic way to interpret the underlying logic of a system and identify the key drivers behind complex phenomena. By integrating statistical learning and testing with physical models, Chapter 3 develops a statistically efficient framework for model construction that captures key features without exhaustive simulation-based model comparison.

The success of a statistical model depends not only on its architecture but also on how the training data is constructed, particularly when resources are limited. This "information limitation" can arise in two ways. First, although various types of measurement data are typically available, not all features exhibit good generalization capabilities. In materials characterization, for instance, models built on system-specific features like chemical composition or molecular weight are inherently restricted to training samples and fail to generalize across different chemical systems [29, 3]. In contrast, physical measurements such as scattering data capture structural information that is independent of specific chemical details, thus providing a basis for universal models [30]. In this case, extracting chemically independent yet informative features from high-dimensional data is crucial for achieving cross-system generalization. Chapter 4 addresses this challenge by developing a physics-informed approach to extract generalizable features from scattering data for polymer phase

identification, which is important for automating experiments in chemical laboratories [31].

Second, when physical simulations or experimental evaluations are expensive, researchers face the critical challenge of sample selection. Traditional experimental design often relies on static sampling schemes [32], such as space-filling designs including Latin hypercube sampling [33] and maximin distance designs [34]. These methods aim to distribute sample points across the input space to ensure sufficient training data near any test point [35]. While effective in low-dimensional problems and scenarios with sufficient computational resources, such space-filling strategy encounters the curse of dimensionality in complex physical systems. As the input dimension increases, the number of samples required to adequately cover the domain grows exponentially, leading to an extremely sparse input space under any realistic budget. This challenge is further amplified when inputs are functions (e.g., external potentials in density or quantum functional theory and initial conditions of partial differential equations) [36, 10, 37, 38], which are inherently infinite-dimensional. In these settings, the model essentially extrapolates across vast "gaps", making it difficult for static sampling schemes to capture abrupt changes. Active learning offers an alternative [39]: instead of aiming for uniform coverage of the entire space, it dynamically selects the most informative points for evaluation, focusing the limited budget on regions of high importance. Common strategies include minimizing the average variance of parameter estimates via the trace of the inverse Fisher information matrix (A-optimality) and maximizing the determinant of the Fisher information matrix (D-optimality) [40, 39]. In the context of GP modeling, approaches based on predictive variance reduction (e.g., ALM, ALC) have emerged [13, 14, 41]. However, these methods mostly focus on the geometric structure of the input domain [12] and rarely utilize the observed output responses to assess model reliability. A new active learning strategy, presented in Chapter 5, enables automatic error control through the quantified reliability of the surrogate model, ensuring high accuracy at feasible

cost.

The mathematical foundations of GPs and their connection to DLM and the KF are introduced in Sections 1.1 and 1.2, respectively. The outline of the subsequent chapters is presented in Section 1.3.

1.1 Gaussian processes

1.1.1 Gaussian processes with scalar outputs

When we model a real-valued unknown function $z(\cdot)$ using a Gaussian process (GP), it means that for any finite set of inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, where each $\mathbf{x}_i \in \mathbb{R}^p$, the corresponding output vector $\mathbf{z} = (z_1, \ldots, z_n)^T$ follows a multivariate normal distribution

$$\mathbf{z} \mid \boldsymbol{\mu}, \sigma^2, \mathbf{R} \sim \mathcal{MN}(\boldsymbol{\mu}, \sigma^2 \mathbf{R}), \tag{1.1}$$

where $\boldsymbol{\mu} = (\mu(\mathbf{x}_1), \ldots, \mu(\mathbf{x}_n))^T$ is a vector of mean, σ^2 is an unknown variance parameter, and the (i, j)-th entry of the correlation matrix $\mathbf{R}$ is defined by the kernel function $R_{ij} = c(\mathbf{x}_i, \mathbf{x}_j)$.

For any input $\mathbf{x}$, the output mean is typically modeled as $\mu(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta}$, where $\boldsymbol{\beta}$ is a p_μ-dimensional vector to be optimized, and $\mathbf{h}(\mathbf{x}) = (h_1(\mathbf{x}), \ldots, h_{p_\mu}(\mathbf{x}))$ is a predefined row vector of mean basis functions. In practice, the most common choice is to use only the intercept term, i.e., set $p_\mu = 1$ and $h(\mathbf{x}) = 1$.

There are two main classes of kernel functions whose structure directly determines the properties of the GP. Isotropic kernels characterize the correlation solely based on the Euclidean distance between two inputs $c(\mathbf{x}_i, \mathbf{x}_j) = c(\|\mathbf{x}_i - \mathbf{x}_j\|)$, where $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip})$ and $\mathbf{x}_j = (x_{j1}, \ldots, x_{jp})$. This type of kernel is often used for spatiotemporal data where each input dimension is on the same scale. When the scale of the input dimensions differs,

Table 1.1: Commonly used kernel functions, with $c(x_i, x_j) = c(d)$ and $d = |x_i - x_j|$. In the Matérn kernel, $\Gamma(\cdot)$ is the gamma function and $\mathcal{K}_\nu(\cdot)$ is the modified Bessel function of the second kind.

Kernel Type	Form	Range
Power Exponential	$\exp(-(d/\gamma)^\nu)$	$\nu \in (0, 2]$
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{d}{\gamma}\right) \mathcal{K}_\nu \left(\frac{d}{\gamma}\right)$	$\nu > 0$
Rational Quadratic	$\left(1 + \left(\frac{d}{\gamma}\right)^2\right)^{-\nu}$	$\nu > 0$

a product kernel is often used to incorporate different correlations for each coordinate, defined as $c(\mathbf{x}_i, \mathbf{x}_j) = c_1(|x_{i1} - x_{j1}|) \times \cdots \times c_p(|x_{ip} - x_{jp}|)$, where $c_l(\cdot)$ is the kernel for the l-th coordinate for $l = 1, \ldots, p$. In this case, the correlation matrix can be expressed as $\mathbf{R} = \mathbf{R}_1 \circ \cdots \circ \mathbf{R}_p$ with $\circ$ representing the Hadamard (element-wise) product.

Some commonly used kernel functions are shown in Table 1.1. The kernel $c_l(\cdot)$ typically involves a roughness parameter $\nu_l > 0$ and a range parameter γ_l that controls the rate at which the correlation decays with distance. For power exponential kernels, when $\nu_l = 2$, the kernel becomes the Gaussian correlation kernel, whose corresponding sample paths are infinitely differentiable. Although the Gaussian kernel offers good flexibility, its overly strong smoothness may be impractical when dealing with real-world physical systems. Thus, choosing $1 < \nu_l < 2$ is often a more suitable choice.

Another class of kernel functions highly favored in scientific computing is the Matérn kernel. When the roughness parameter ν_l takes half-integer values, i.e., $\nu_l = (2\tilde{q} - 1)/2$, with $\tilde{q} \in \mathbb{N}^+$ being a positive integer, the Matérn kernel has a closed-form expression. Furthermore, a GP with a Matérn kernel is $\lceil \nu_l - 1 \rceil$ times differentiable, an attractive property in practice as the differentiability of the process is directly controlled by the roughness parameter [9]. For example, when $\nu_l = 1/2$, it simplifies to the exponential correlation $c_l(d) = \exp(-d/\gamma_l)$, corresponding to the Ornstein-Uhlenbeck process, which is not mean-squared differentiable.

When $\nu_l = 5/2$, the Matérn kernel takes the form

$$c_l(d) = \left(1 + \frac{5^{1/2}d}{\gamma_l} + \frac{5d^2}{3\gamma_l^2}\right) \exp\left(-\frac{5^{1/2}d}{\gamma_l}\right), \tag{1.2}$$

which is twice differentiable. Due to the good balance between smoothness and flexibility, this Matérn kernel (1.2) has been widely adopted as the default covariance function in modern software of GP surrogate models or emulators [42, 43].

Predictive distribution and parameter estimation

Given observed data $\mathbf{z}$ and the corresponding inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the likelihood of the GP models is

$$\mathcal{L}(\boldsymbol{\beta}, \sigma^2, \boldsymbol{\gamma} \mid \mathbf{z}) = (2\pi\sigma^2)^{-n/2}|\mathbf{R}|^{-1/2} \exp\left(-\frac{(\mathbf{z} - \mathbf{H}\boldsymbol{\beta})^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\boldsymbol{\beta})}{2\sigma^2}\right), \tag{1.3}$$

where $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1)^T, \ldots, \mathbf{h}(\mathbf{x}_n)^T]^T$ is the $n \times p_\mu$ design matrix, $\boldsymbol{\beta}$ is the mean parameter vector, σ^2 is the variance parameter, and $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_p\}$ represents the range parameters for each input dimension. Conditional on $\boldsymbol{\gamma}$, the maximum likelihood estimates (MLE) is obtained as

$$\hat{\boldsymbol{\beta}}_{MLE} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}, \quad \hat{\sigma}^2_{MLE} = \frac{1}{n}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}}). \tag{1.4}$$

Plugging $\hat{\boldsymbol{\beta}}_{MLE}$ and $\hat{\sigma}^2_{MLE}$ into (1.3), the likelihood reduces to the profile likelihood

$$\mathcal{L}(\boldsymbol{\gamma} \mid \mathbf{z}, \hat{\boldsymbol{\beta}}_{MLE}, \hat{\sigma}^2_{MLE}) \propto |\mathbf{R}|^{-1/2} \left((\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})\right)^{-n/2}. \tag{1.5}$$

The range parameters $\boldsymbol{\gamma}$ are then estimated as the mode of the log profile likelihood (1.5), denoted as $\hat{\boldsymbol{\gamma}}_{MLE}$. The predictive distribution of a new input $\mathbf{x}^*$, conditional on all estimated parameters, is given by

$$z(\mathbf{x}^*) \mid \mathbf{z}, \hat{\boldsymbol{\beta}}_{MLE}, \hat{\sigma}^2_{MLE}, \hat{\boldsymbol{\gamma}}_{MLE} \sim \mathcal{N}(\hat{z}_{MLE}(\mathbf{x}^*), \hat{\sigma}^2_{MLE} c^*_{MLE}), \tag{1.6}$$

where

$$\hat{z}_{MLE}(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\beta}}_{MLE} + \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\left(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}}_{MLE}\right), \tag{1.7}$$

$$c^*_{MLE} = c(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*), \tag{1.8}$$

with $\mathbf{r}(\mathbf{x}^*) = [c(\mathbf{x}^*, \mathbf{x}_1), \ldots, c(\mathbf{x}^*, \mathbf{x}_n)]^T$, and $\mathbf{R}$ depends on $\hat{\gamma}_{MLE}$.

Another parameter estimation method is based on the robust marginal posterior. Within the Bayesian framework, we assign prior distributions to the mean and variance parameters. A common choice is the reference prior [44]: $\pi(\boldsymbol{\beta}, \sigma^2) \propto 1/\sigma^2$. Under this prior, the marginal likelihood is obtained by integrating out $\boldsymbol{\beta}$ and σ^2:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\gamma} \mid \mathbf{z}) &\propto \int\int p(\mathbf{z} \mid \boldsymbol{\beta}, \sigma^2, \mathbf{R})\pi(\boldsymbol{\beta}, \sigma^2) d\boldsymbol{\beta} d\sigma^2 \\ &\propto |\mathbf{R}|^{-1/2}|\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}|^{-1/2}\left((\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})\right)^{-(n-p_\mu)/2}.\end{aligned} \tag{1.9}$$

The maximum marginal posterior estimation (MMPE) of $\boldsymbol{\gamma}$ follows:

$$\hat{\boldsymbol{\gamma}}_{MMPE} = \text{argmax}_{\boldsymbol{\gamma}} \log(\mathcal{L}(\boldsymbol{\gamma} \mid \mathbf{z})\pi(\boldsymbol{\gamma})), \tag{1.10}$$

where $\pi(\boldsymbol{\gamma})$ is a prior of the range parameters or their transformation. The reference prior or jointly robust prior of $\boldsymbol{\gamma}$ are used in practice [45, 46].

Conditional on the estimated range parameter $\hat{\boldsymbol{\gamma}}_{MMPE}$, the predictive distribution of $z(\mathbf{x}^*)$ at a new input $\mathbf{x}^*$ follows a non-centered scaled Student's t-distribution with $n - p_\mu$ degrees of freedom:

$$z(\mathbf{x}^*) \mid \mathbf{z}, \hat{\boldsymbol{\gamma}}_{MMPE} \sim \mathcal{T}(\hat{z}(\mathbf{x}^*), \hat{\sigma}^2 c^{**}, n - p_\mu), \tag{1.11}$$

where, after replacing $\hat{\boldsymbol{\gamma}}_{MLE}$ by $\hat{\boldsymbol{\gamma}}_{MMPE}$, $\hat{z}(\mathbf{x}^*) = \hat{z}_{MLE}(\mathbf{x}^*)$, $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{MLE}$, $\hat{\sigma}^2 = n\hat{\sigma}^2_{MLE}/(n - p_\mu)$, and

$$c^{**} = c^*_{MLE} + \mathbf{h}^*(\mathbf{x}^*)^T \left(\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}\right)^{-1} \mathbf{h}^*(\mathbf{x}^*), \tag{1.12}$$

with $\mathbf{h}^*(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*) - \mathbf{H}^T \mathbf{R}^{-1} \mathbf{r}(\mathbf{x}^*)$. The derivation of Equation (1.11) is provided in Appendix A.

For computing either the profile likelihood or the marginal posterior distribution, the main computational bottleneck of GP lies in the inversion and determinant calculation of the correlation matrix $\mathbf{R}$. These operations are typically carried out via Cholesky decomposition, which has a computational complexity of $\mathcal{O}(n^3)$. Therefore, direct computation for GP can be computationally expensive for problems with large datasets.

Gaussian processes for noisy observations

In many practical applications, we cannot observe the underlying latent function, but only noisy observations:

$$y(\mathbf{x}) = z(\mathbf{x}) + \epsilon(\mathbf{x}), \tag{1.13}$$

where $z(\cdot)$ is modeled as a GP and $\epsilon(\mathbf{x}) \sim \mathcal{N}(0, \sigma_0^2)$ represents independent observation noise. This model is widely used in scenarios involving observation errors, such as measurement errors or random variability, or to improve numerical stability even if there is no noise. Let $\mathbf{y} = (y(\mathbf{x}_1), y(\mathbf{x}_2), \ldots, y(\mathbf{x}_n))^T$ be the noisy observations at the input set $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$, and define the nugget parameter as $\eta = \sigma_0^2/\sigma^2$. Under this model, the parameter estimation and predictive distribution remain identical to those described previously for both profile likelihood or marginal likelihood methods, except that all $\mathbf{R}$ in Equations (1.3)-(1.12) are replaced by $\tilde{\mathbf{R}} = \mathbf{R} + \eta \mathbf{I}_n$. The nugget parameter η is then estimated jointly with the range parameters $\boldsymbol{\gamma}$.

1.1.2 Gaussian processes with vectorized outputs

The output of interest in many simulations and experiments is often a vector rather than a scalar. Let $\mathbf{z}(\mathbf{x}) = (z_1(\mathbf{x}), \ldots, z_k(\mathbf{x}))^T$ be the k-dimensional output vector at the input $\mathbf{x}$, where $z_j(\mathbf{x})$ represents the response of the j-th output dimension. A straightforward modeling approach is to fit a GP model independently for each output dimension $z_j(\cdot)$. However, this method encounters significant computational bottlenecks when k is large. As discussed earlier, the main computational cost of a standard GP lies in the inversion and determinant calculation of the correlation matrix $\mathbf{R}$, with a complexity of $O(n^3)$. For a k-dimensional output, independent modeling requires parameter estimation and prediction for each dimension separately, resulting in a total computational complexity of $O(kn^3)$, which is computationally prohibitive when both k and n are large.

To reduce the computational cost while maintaining modeling flexibility, we adopt the partial parallel Gaussian process (PP-GP) model [47]. The core idea of this model is to share the range parameters $\boldsymbol{\gamma}$ and basis functions $\mathbf{h}(\mathbf{x})$ across all output dimensions while allowing the mean and variance parameters to vary. Specifically, for the j-th output

dimension, we assume $z_j(\cdot) \sim \mathcal{GP}(\mathbf{h}(\cdot)\boldsymbol{\beta}_j, \sigma_j^2 c(\cdot,\cdot))$, where $\boldsymbol{\beta}_j$ and σ_j^2 are the mean and variance parameters specific to the j-th dimension, respectively. The use of shared range parameters ensures that the correlation matrix $\mathbf{R}$ only needs to be computed once, while dimension-specific parameters σ_j^2 and $\boldsymbol{\beta}_j$ allow different outputs to have distinct mean structures and levels of variability.

The reference prior for this model is $\pi(\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_k, \sigma_1^2, \ldots, \sigma_k^2) \propto 1/\prod_{j=1}^k \sigma_j^2$. For a new input point $\mathbf{x}^*$, the predictive distribution for the j-th output dimension is a Student's t-distribution with $n - p_\mu$ degrees of freedom

$$z_j(\mathbf{x}^*) \mid \mathbf{z}_j, \boldsymbol{\gamma} \sim \mathcal{T}(\hat{z}_j(\mathbf{x}^*), \hat{\sigma}_j^2 c^{**}, n - p_\mu), \tag{1.14}$$

where

$$\hat{z}_j(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\beta}}_j + \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}(\mathbf{z}_j - \mathbf{H}\hat{\boldsymbol{\beta}}_j) \tag{1.15}$$

and c^{**} defined as in (1.12), with

$$\hat{\boldsymbol{\beta}}_j = \left(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}_j, \tag{1.16}$$

$$\hat{\sigma}_j^2 = (\mathbf{z}_j - \mathbf{H}\hat{\boldsymbol{\beta}}_j)^T\mathbf{R}^{-1}(\mathbf{z}_j - \mathbf{H}\hat{\boldsymbol{\beta}}_j)/(n - p_\mu). \tag{1.17}$$

In the PP-GP framework, the outputs are assumed to be conditionally independent across dimensions given the shared parameters. Thus, we have $p(z_j(\mathbf{x}^*) \mid \hat{\gamma}, \mathbf{z}) = p(z_j(\mathbf{x}^*) \mid \hat{\gamma}, \mathbf{z}_j)$, for any j and $\mathbf{x}^*$, which allows us to use Equation (1.14) for computing the posterior predictive distribution. The shared range parameters $\boldsymbol{\gamma}$ can be estimated by maximizing either the marginal likelihood, obtained after integrating out the mean and variance parameters $(\boldsymbol{\beta}_j, \sigma_j^2)$ for all $j = 1, \ldots, k$, or the marginal posterior density (MMPE), if we further incorporate a

prior distribution $\pi(\boldsymbol{\gamma})$ similar to the framework in (1.10). In practice, the marginal posterior mode is often preferred as it provides more robust parameter estimation.

In comparison with alternative methods for modeling multi-dimensional outputs, the PP-GP emulator discussed above has advantages in terms of computational scalability and internal uncertainty assessment. First, constructing the covariance matrix requires $O(n^2p)$ operations, while computing the predictive mean in Equation (1.15) only takes $\mathcal{O}(nk) + \mathcal{O}(n^3)$ operations, where the $\mathcal{O}(n^3)$ term stems from the Cholesky decomposition of the covariance matrix. Compared to the $\mathcal{O}(kn^3)$ cost of building independent emulators for each output dimension, this represents a significant reduction, as it requires only a single Cholesky decomposition for all output dimensions. Furthermore, approaches that explicitly model the $k \times k$ spatial covariance matrix generally incur a cost of $\mathcal{O}(k^3)$, which is computationally expensive when the number of grid points k is large. Second, predictive uncertainty can be directly assessed since the percentiles of the predictive distributions in (1.14) have closed-form expressions. This quantified uncertainty allows for the control of predictive errors, as discussed in Chapter 5.

One natural generalization is the multi-output emulator of [48], which models the k-dimensional output as a matrix-variate GP with separable covariance structure, where the joint covariance between outputs at two inputs is modeled as the product of an input correlation function and a $k \times k$ output covariance matrix that captures dependencies among output dimensions. Theorem 6.1 in [47] shows that under an objective prior, the posterior predictive mean is identical for any choice of this output covariance matrix, implying that the PP-GP achieves comparable predictive accuracy without explicitly modeling output dependencies.

An alternative for modeling multi-dimensional output is the linear model of coregionalization (LMC) [49]. It decomposes the output into a linear combination of independent

latent GP factors, with a factor loading matrix linking these latent factors to the observed output dimensions. A common way to estimate the loading matrix is to apply principal component analysis (PCA) to the simulator output [50], which turns the emulation problem into fitting a set of independent scalar GP emulators on the projected coordinates. Within this latent factor framework, the PCA solution coincides with the maximum marginal likelihood estimator of the factor loading matrix only when the latent factors are independent and normally distributed [51]. Once the factors are correlated, this PCA estimator can be suboptimal. To fix this, generalized probabilistic PCA (GPPCA) models each latent factor as a GP and derives a more accurate estimator of the factor loading matrix that accounts for factor correlation [51]. This framework was later extended to handle large datasets with irregular missing values via Gaussian orthogonal latent factor (GOLF) processes [52]. More recently, a new algorithm called FMOU was developed for high-dimensional dynamical systems [53]. By modeling each latent factor as an Ornstein-Uhlenbeck process with distinct correlation and variance parameters, this algorithm uses the orthogonal loading structure to achieve closed-form updates within an expectation-maximization framework. When these latent factor processes are equipped with specific covariances on a one-dimensional input, they admit an equivalent state-space representation as a dynamic linear model, as reviewed in Section 1.2.

1.2 Dynamic linear models and Kalman filter

Dynamic linear models (DLMs), also known as linear state space models, are widely used for modeling temporally correlated data [54, 55]. Each observation $y_t \in \mathbb{R}$ in DLMs is

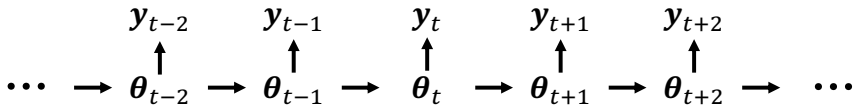

Figure 1.1: The dependent structure of the dynamic linear model.

associated with a latent state vector $\boldsymbol{\theta}_t \in \mathbb{R}^{\tilde{q}}$, defined as:

$$y_t = \mathbf{F}_t \boldsymbol{\theta}_t + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_{0,t}^2), \tag{1.18}$$

$$\boldsymbol{\theta}_t = \mathbf{G}_t \boldsymbol{\theta}_{t-1} + \mathbf{w}_t, \quad \mathbf{w}_t \sim \mathcal{MN}(\mathbf{0}, \mathbf{W}_t), \tag{1.19}$$

where $\mathbf{F}_t$ and $\mathbf{G}_t$ are matrices of dimensions $1 \times \tilde{q}$ and $\tilde{q} \times \tilde{q}$, respectively, $\mathbf{W}_t$ is a $\tilde{q} \times \tilde{q}$ covariance matrix for $t = 2, \ldots, n$, and the initial state vector follows a multivariate normal distribution $\boldsymbol{\theta}_1 \sim \mathcal{MN}(\mathbf{b}_1, \mathbf{W}_1)$. While the model allows for time-varying observation noise, in practice, it is often assumed that $\sigma_{0,t}^2 = \sigma_0^2$ for all t. The dependency structure of DLMs is illustrated in Figure 1.1.

The Kalman filter (KF) and Rauch-Tung-Striebel (RTS) smoother provide a fast and exact approach to estimate latent states and compute likelihood and predictive distributions in DLMs, which scales linearly with the number of observations [26], as reviewed in Lemma 1.1 and Lemma 1.2. In particular, the KF enables efficient computation of $\mathbf{L}^{-1}\mathbf{u}$ for any n-dimensional vector $\mathbf{u}$ in $\mathcal{O}(\tilde{q}^3 n)$ operations, where $\mathbf{L}$ is the Cholesky factor of the covariance matrix $\boldsymbol{\Sigma} = \mathrm{Cov}[\mathbf{y}_{1:n}] = \mathbf{L}\mathbf{L}^T$. The result is summarized in Lemma 1.3, with the detailed proof provided in Appendix A.2.

Lemma 1.1 (Kalman Filter)**.** *Let* $\boldsymbol{\theta}_{t-1} \mid \mathbf{y}_{1:t-1} \sim \mathcal{MN}(\mathbf{m}_{t-1}, \mathbf{C}_{t-1})$. *Recursively for* $t = 2, \ldots, n$*:*

(i) The one-step-ahead predictive distribution of $\boldsymbol{\theta}_t$ *given* $\mathbf{y}_{1:t-1}$ *is*

$$\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t-1} \sim \mathcal{MN}(\mathbf{b}_t, \mathbf{B}_t), \tag{1.20}$$

with $\mathbf{b}_t = \mathbf{G}_t\mathbf{m}_{t-1}$ *and* $\mathbf{B}_t = \mathbf{G}_t\mathbf{C}_{t-1}\mathbf{G}_t^T + \mathbf{W}_t$.

(ii) The one-step-ahead predictive distribution of y_t *given* $\mathbf{y}_{1:t-1}$ *is*

$$y_t \mid \mathbf{y}_{1:t-1} \sim \mathcal{N}(f_t, Q_t), \tag{1.21}$$

with $f_t = \mathbf{F}_t\mathbf{b}_t$, *and* $Q_t = \mathbf{F}_t\mathbf{B}_t\mathbf{F}_t^T + \sigma_{0,t}^2$.

(iii) The filtering distribution of $\boldsymbol{\theta}_t$ *given* $\mathbf{y}_{1:t}$ *follows*

$$\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t} \sim \mathcal{MN}(\mathbf{m}_t, \mathbf{C}_t), \tag{1.22}$$

with $\mathbf{m}_t = \mathbf{b}_t + \mathbf{B}_t\mathbf{F}_t^T Q_t^{-1}(y_t - f_t)$ *and* $\mathbf{C}_t = \mathbf{B}_t - \mathbf{B}_t\mathbf{F}_t^T Q_t^{-1}\mathbf{F}_t\mathbf{B}_t$, *where* $\mathbf{K}_t = \mathbf{B}_t\mathbf{F}_t^T Q_t^{-1}$ *is called the Kalman gain.*

Lemma 1.2 (RTS Smoother)**.** *Denote* $\boldsymbol{\theta}_{t+1} \mid \mathbf{y}_{1:n} \sim \mathcal{MN}(\mathbf{s}_{t+1}, \mathbf{S}_{t+1})$, *then recursively for* $t = n-1, ..., 1$,

$$\boldsymbol{\theta}_t \mid \mathbf{y}_{1:n} \sim \mathcal{MN}(\mathbf{s}_t, \mathbf{S}_t), \tag{1.23}$$

where

$$\mathbf{s}_t = \mathbf{m}_t + \mathbf{C}_t\mathbf{G}_{t+1}^T\mathbf{B}_{t+1}^{-1}(\mathbf{s}_{t+1} - \mathbf{b}_{t+1}),$$

$$\mathbf{S}_t = \mathbf{C}_t - \mathbf{C}_t\mathbf{G}_{t+1}^T\mathbf{B}_{t+1}^{-1}(\mathbf{B}_{t+1} - \mathbf{S}_{t+1})\mathbf{B}_{t+1}^{-1}\mathbf{G}_{t+1}\mathbf{C}_t.$$

Lemma 1.3. *Denote* $\tilde{\mathbf{y}} = (\tilde{y}_1, \ldots, \tilde{y}_n)^T = \mathbf{L}^{-1}\mathbf{y}$*, where* $\mathbf{L}$ *is the lower triangular factor in Cholesky decomposition of a positive definite covariance* $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$*, with* $\boldsymbol{\Sigma} = \mathrm{Cov}[\mathbf{y}_{1:n}]$ *with* y_t *defined as in (1.18). We have*

$$\tilde{y}_t = \frac{y_t - f_t}{Q_t^{1/2}}, \tag{1.24}$$

where f_t *and* Q_t *are defined in (1.21). Furthermore, the* t*-th diagonal term of* $\mathbf{L}$ *is* $L_{t,t} = Q_t^{1/2}$*.*

DLMs are a large class of models that include many widely used processes, such as autoregressive and moving average processes [56, 57]. When the input is one-dimensional and ordered (e.g., time or spatial series), GPs with commonly used covariance functions, including the Matérn kernel with a half-integer roughness parameter [23], can also be represented as DLMs [24, 25]. In such cases, there exist closed-form expressions for $\mathbf{F}_t$, $\mathbf{G}_t$ and $\mathbf{W}_t$. This connection, summarized in Appendix A, provides significant computational advantages for GPs, as we can achieve inference and prediction with linear complexity by utilizing KF algorithms, thereby overcoming the $\mathcal{O}(n^3)$ computational bottleneck of standard GP methods.

1.3 Outline

Chapter 2 introduces the inverse Kalman filter (IKF), which extends the computational scalability of the KF to more general and complex dependency structures. While standard DLMs assume a one-to-one mapping where each observation y_t depends only on its corresponding latent state $\boldsymbol{\theta}_t$ (Figure 1.1), many scientific problems involve dependencies where a single observation is influenced by multiple latent states simultaneously (Figure 2.1). The IKF addresses these complexities by enabling exact, linear-time multiplication of the covariance matrix with arbitrary vectors. Furthermore, when integrated with the conjugate

gradient algorithm, it enables fast GP inference and prediction for kernels that admit a DLM representation. This approach is validated through the nonparametric estimation of particle interaction functions and is implemented in the `FastGaSP` R package.

Chapter 3 explores the modeling of the same collective cell migration system from a different perspective. Building upon the Vicsek model framework [27], this chapter constructs a parameterized model through systematic statistical testing to describe cellular movement on anisotropic substrates. By using normality and significance tests to guide feature selection, we demonstrate how to capture key biophysical mechanisms while maintaining model simplicity and interpretability. This chapter uses the same cell trajectory data as Chapter 2, but the focus shifts from computational scalability in non-parametric prediction to the discovery of mechanistic rules.

Shifting from inference problems to design problems, Chapter 4 focuses on the extraction of features with cross-system generalization capabilities. This chapter proposes a physics-informed machine learning method for the automated identification of block copolymer phase structure from small-angle X-ray scattering data. By employing KF for curve denoising and extracting chemically independent physical features, the resulting random forest classification model demonstrates the ability to generalize to previously unseen chemical compositions.

Chapter 5 explores how to intelligently select samples when data acquisition costs are high. This chapter develops an active learning with error control (ALEC) framework to guide sequential sampling by quantifying the reliability of model predictions. Unlike some active learning methods that rely primarily on the geometric structure of the input space, ALEC incorporates observed output responses to dynamically assess model confidence and establishes probabilistic error bounds based on the predictive t-distribution of the GP. This effectiveness of the method is validated through surrogate modeling of classical density

functional theory and is compared with alternative sampling strategies.

Finally, Chapter 6 summarizes the main contributions of this thesis and outlines potential directions for future research. Specifically, we explore extensions of the IKF and active learning framework. We also discuss the integration of GP approximations, such as the Vecchia approximation, into our ALEC framework, and propose further extensions to the Vecchia method itself.

Chapter 2

The inverse Kalman filter

Building upon the connection between Gaussian processes and dynamic linear models established in Chapter 1, this chapter addresses the computational challenges of applying Gaussian process inference to large-scale datasets: the cubic computational complexity required for inverting the covariance matrix. We develop the inverse Kalman filter, which leverages the state-space representation of the Gaussian process to achieve exact multiplication of the covariance matrix with any vector in linear time. Unlike the Kalman filter, which is limited to the one-to-one mapping in traditional dynamic linear models where each observation relies on a single latent state, the IKF can be applied to handle complex dependency structures where a single observation depends on multiple latent states simultaneously. This is a common structure in many scientific problems, such as particle dynamics. Combined with the conjugate gradient algorithm, this method provides an accurate and scalable computational framework for inference in large-scale dynamic systems. We validate the performance of this method in nonparametric estimation of particle interaction functions and integrate its implementation into the `FastGaSP` R package.

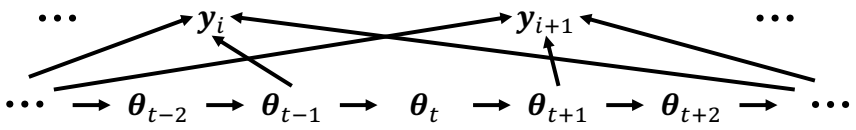

Figure 2.1: The dependent structure of particle dynamics.

2.1 Introduction

Dynamic linear models (DLMs) offer a powerful representation for modeling temporally correlated processes. As discussed in Section 1.2, certain Gaussian processes (GPs), such as those with Matérn covariance functions and half-integer smoothness parameters, admit exact DLM representations. This equivalence allows GP inference to be carried out using the Kalman filter (KF), leading to substantial computational advantages. In particular, as shown in Lemma 1.3 in Section 1.2, the KF enables the computation of $\mathbf{L}^{-1}\mathbf{u}$ for any vector $\mathbf{u}$ in $\mathcal{O}(\tilde{q}^3 N)$ operations, where $\tilde{q}$ is the latent state dimension, N is the number of observations. Here, $\mathbf{L}$ is the Cholesky factor of the covariance matrix $\mathbf{\Sigma} = \mathrm{Cov}(\mathbf{y}_{1:N}) = \mathbf{L}\mathbf{L}^T$, with $\mathbf{y}_{1:N}$ denoting the output observations.

Computational challenges remain for high-dimensional state space models and scenarios where the KF cannot be directly applied, such as the dependent structure shown in Figure 2.1, where each observation y_i can be associated with multiple latent states. This interaction structure, introduced in Section 2.3, is common in physical models, including molecular dynamics simulation [58]. A way to overcome these challenges is to efficiently compute $\mathbf{\Sigma}\mathbf{u}$ for any N-dimensional vector $\mathbf{u}$ and utilize optimization methods such as conjugate gradient (CG) algorithms for computing predictive distributions [59]. This strategy has been used to approximate the maximum likelihood estimator of parameters in GPs [60, 61]. Yet, each CG iteration requires matrix-vector multiplication, which involves $\mathcal{O}(N^2)$ operations and storage, making it prohibitive for large N.

To address this issue, we introduce the *inverse Kalman filter* (IKF), which computes $\mathbf{Lu}$ and $\mathbf{L}^T\mathbf{u}$ with $\mathcal{O}(\tilde{q}^3 N)$ operations without approximation, where $\mathbf{L}$ is the Cholesky factor of any $N \times N$ DLM-induced covariance $\mathbf{\Sigma}$ and $\mathbf{u}$ is an N-dimensional vector. The complexity is significantly smaller than $\mathcal{O}(N^2)$ in direct computation. The dimension of the latent states $\tilde{q}$ does not exceed 3 in our applications, and this choice includes many commonly used models such as twice differentiable GPs with Matérn covariance in (1.2), often assumed as default in GP models.

The IKF algorithm can be extended to accelerate matrix multiplication and inversion, a key computational bottleneck in practice. We integrate the IKF into the CG algorithm to scalably compute predictive distributions of observations with a general covariance structure:

$$\mathbf{\Sigma}_y = \sum_{j=1}^{J} \mathbf{A}_j \mathbf{\Sigma}_j \mathbf{A}_j^T + \mathbf{\Lambda}, \tag{2.1}$$

where $\mathbf{\Sigma}_j$ is a DLM-induced covariance matrix, $\mathbf{A}_j$ is sparse, and $\mathbf{\Lambda}$ is a diagonal matrix. This structure appears in nonparametric estimation of particle interactions, which will be introduced in Section 2.3. Both real and simulated studies involve numerous particles over a long time period, where conventional approximation methods [19, 21, 62, 63, 64, 22] are not applicable. This covariance structure also appears in other applications, including varying coefficient models [65] and estimating incomplete lattices of correlated data [11]. Though we focus on nonparametric estimation of particle interaction functions in this chapter, we also apply our approach to the latter in Appendix B.4 and provide numerical comparisons with various approximation methods in Appendix B.8.

2.2 Inverse Kalman filter

In this section, we introduce an exact algorithm for computing $\mathbf{\Sigma u}$ with $\mathcal{O}(\tilde{q}^3 N)$, where $\mathbf{u}$ is an N-dimensional real-valued vector and $\mathbf{\Sigma} = \mathrm{Cov}[\mathbf{y}_{1:N}]$ is an $N \times N$ covariance matrix of observations induced by a DLM with $\tilde{q}$ latent states, as defined in (1.18)-(1.19). This algorithm is applicable to any DLM specified in (1.18)-(1.19). Define $\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T$, where the Cholesky factor $\mathbf{L}$ does not need to be explicitly computed in our approach. We develop fast algorithms to compute $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$ and $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$ by Lemma 2.1 and Lemma 2.2 below, respectively, each with $\mathcal{O}(\tilde{q}^3 N)$ operations. Detailed proofs of these lemmas are available in the Appendix B.1. In the following, u_t, x_t, $\tilde{x}_t$ denote the t-th elements of the vectors $\mathbf{u}$, $\mathbf{x}$, and $\tilde{\mathbf{x}}$, respectively, and $L_{t',t}$ denotes the (t',t)-th element of $\mathbf{L}$ for t and $t' = 1, 2, \ldots, N$.

Lemma 2.1 (Compute $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$ with linear computational cost)**.** *For any N-dimensional vector $\mathbf{u}$, let $\tilde{x}_N = Q_N^{1/2} u_N$, $\tilde{x}_{N-1} = Q_{N-1}^{1/2}(\ell_{N,N-1}u_N + u_{N-1})$, and $\mathbf{g}_{N-1} = \mathbf{F}_N\mathbf{G}_N u_N$. For $t = N-2, \ldots, 1$, iteratively compute $\tilde{x}_t$ using*

$$\tilde{x}_t = Q_t^{\frac{1}{2}} \left(\tilde{\ell}_{t+1,t} + \ell_{t+1,t} u_{t+1} + u_t \right), \tag{2.2}$$

$$\mathbf{g}_t = \mathbf{g}_{t+1}\mathbf{G}_{t+1} + \mathbf{F}_{t+1}\mathbf{G}_{t+1}u_{t+1}, \tag{2.3}$$

with $\ell_{t+1,t} = \mathbf{F}_{t+1}\mathbf{G}_{t+1}\mathbf{K}_t$, $\tilde{\ell}_{t+1,t} = \mathbf{g}_{t+1}\mathbf{G}_{t+1}\mathbf{K}_t$, and the Kalman gain

$$\mathbf{K}_t = \mathbf{B}_t\mathbf{F}_t^T Q_t^{-1}, \tag{2.4}$$

where $\mathbf{B}_t = \mathrm{Cov}[\boldsymbol{\theta}_t \mid \mathbf{y}_{1:t-1}]$ and $Q_t = \mathrm{Var}[y_t \mid \mathbf{y}_{1:t-1}]$ are respectively defined in the KF in (1.20) and (1.21) in Section 1.2. Then $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$.

Lemma 2.2 (Compute $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$ with linear computational cost)**.** *For any N-dimensional vector $\tilde{\mathbf{x}}$, let $x_1 = \mathbf{F}_1\mathbf{b}_1 + {Q_1}^{1/2}\tilde{x}_1$ and $\tilde{\mathbf{m}}_1 = \mathbf{b}_1 + \mathbf{K}_1(x_1 - \mathbf{F}_1\mathbf{b}_1)$. For $t = 2, \ldots, N$,*

Algorithm 2.1 The IKF for computing $\boldsymbol{\Sigma}\mathbf{u}$

Input: $\mathbf{F}_t, \mathbf{G}_t, \mathbf{W}_t, t = 1, \ldots, N$, an N-dimensional vector $\mathbf{u}$.

1: Use KF from Lemma 1.1 to compute $\mathbf{B}_t$, Q_t, and $\mathbf{K}_t$ for $t = 1, \ldots, N$.

2: Use Lemma 2.1 to compute $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$.

3: Use Lemma 2.2 to compute $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$.

Output: $\mathbf{x}$.

iteratively compute x_t *using*

$$\mathbf{b}_t = \mathbf{G}_t\tilde{\mathbf{m}}_{t-1}, \tag{2.5}$$

$$x_t = \mathbf{F}_t\mathbf{b}_t + Q_t^{\frac{1}{2}}\tilde{x}_t, \tag{2.6}$$

$$\tilde{\mathbf{m}}_t = \mathbf{b}_t + \mathbf{K}_t(x_t - \mathbf{F}_t\mathbf{b}_t), \tag{2.7}$$

where $\mathbf{K}_t$ *and* Q_t *are respectively defined in (2.4) and (1.21). Then* $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$.

The algorithm in Lemma 2.1 is unconventional and nontrivial to derive. In contrast, Lemma 2.2 has a direct connection to the KF. Specifically, the one-step-ahead predictive distribution in step (ii) of the KF, given in Chapter 1, enables computing $\mathbf{L}^{-1}\mathbf{x}$ for any vector $\mathbf{x}$ (Lemma 1.3). Lemma 2.2 essentially reverses this operation, computing $\mathbf{L}\tilde{\mathbf{x}}$ for any vector $\tilde{\mathbf{x}}$, leading to the term *Inverse Kalman filter* (IKF).

The IKF algorithm, outlined in Algorithm 2.1, sequentially applies Lemmas 2.1 and 2.2 to reduce the computational cost and storage for $\boldsymbol{\Sigma}\mathbf{u}$ from $\mathcal{O}(N^2)$ to $\mathcal{O}(\tilde{q}^3N)$ without approximation. Note that the $\mathcal{O}(\tilde{q}^3N)$ is primarily due to the matrix-matrix multiplication required to obtain $\mathbf{B}_t$ (Equation (1.20)) in the initial KF. Once these quantities are pre-computed, the core iterations in Lemmas 2.1-2.2 only involve matrix-vector multiplications, which require $\mathcal{O}(\tilde{q}^2N)$. The IKF applies to all DLMs, including GPs that admit equivalent DLM representation, such as GPs with Matérn kernels where roughness parameters $1/2$ and $5/2$ correspond to $\tilde{q} = 1$ and $\tilde{q} = 3$, respectively.

While Algorithm 2.1 can be applied to any DLM in (1.18)-(1.19) regardless of the

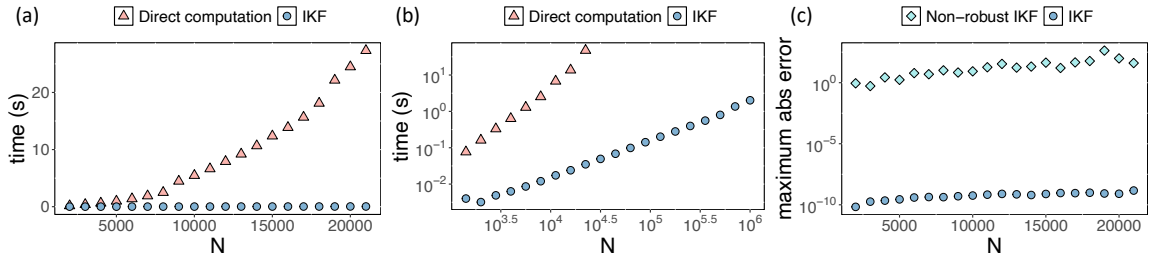

Figure 2.2: Panels (a) and (b) compare the IKF (blue dots) with direct computation (red triangles) for calculating $\mathbf{\Sigma u}$ in the original time scale and logarithmic scale (base 10), respectively. Panel (c) shows the predictive error comparison between the non-robust IKF (light blue squares) and the robust IKF (blue dots).

noise variance, we do not recommend directly computing $\mathbf{\Sigma u}$ in scenarios when the noise variance $\sigma_{0,t}^2$ is zero or close to zero in (1.18). This is because, when $\sigma_{0,t}^2 = 0$, Q_t in (1.21), the variance of the one-step-ahead predictive distribution, could be extremely small, which leads to large numerical errors in (1.22) due to the unstable inversion of Q_t. To ensure robustness, we suggest computing $\mathbf{x} = (\mathbf{\Sigma} + V\mathbf{I}_N)\mathbf{u}$ with a positive scalar V and outputting $\mathbf{x} - V\mathbf{u}$ as the result of $\mathbf{\Sigma u}$. Here, $\mathbf{\Sigma} + V\mathbf{I}_N$ can be interpreted as the covariance matrix of a DLM with noise variance V. Essentially, we introduce artificial noise to ensure that Q_t computed in the KF is at least V for numerical stability. The result remains exact and independent of the choice of V.

We compare the IKF approach with the direct matrix-vector multiplication of $\mathbf{\Sigma u}$ in noise-free scenarios in Figure 2.2. The inputs of $\mathbf{\Sigma}$ are uniformly sampled from $[0, 1]$ and the Matérn covariance with unit variance, roughness parameter $5/2$, and range parameter $\gamma = 0.1$ [23] is used. Panels (a) and (b) show that the IKF significantly reduces computational costs compared to direct computation. The IKF achieves covariance matrix-vector multiplication for an output vector of 10^6 dimensions in about 2 seconds on a desktop, making it highly scalable to be applied in iterative algorithms like the CG algorithm [59] and the randomized log-determinant approximation [66]. Figure 2.2(c) compares the maximum absolute error

between the robust IKF with $V = 0.1$ and the non-robust IKF with $V = 0$. The non-robust IKF exhibits large numerical errors due to instability when Q_t approaches zero, while the robust IKF remains stable by ensuring that Q_t is no smaller than V, even with near-singular covariance matrices $\mathbf{\Sigma}$.

Lemmas 2.1 and 2.2 facilitate the computation of each element in $\mathbf{L}$ using DLM parameters and enable the linear computation of $(\mathbf{L}^T)^{-1}\tilde{\mathbf{x}}$. These results are respectively summarized in Lemmas 2.3 and 2.4 below, with proofs provided in the Appendix B.1.

Lemma 2.3 (Cholesky factor from the inverse Kalman filter)**.** *Each entry (t', t) in the lower triangular matrix $\mathbf{L}$ with $t' \geq t$ has the form*

$$L_{t',t} = Q_t^{\frac{1}{2}} \ell_{t',t}, \tag{2.8}$$

where $\ell_{t',t'} = 1$, and, for $t' > t$, $\ell_{t',t}$ is defined as

$$\ell_{t',t} = \mathbf{F}_{t'} \left(\prod_{l=t+1}^{t'} \mathbf{G}_l \right) \mathbf{K}_t, \tag{2.9}$$

with $\prod_{l=t+1}^{t'} \mathbf{G}_l = \mathbf{G}_{t'} \mathbf{G}_{t'-1} \dots \mathbf{G}_{t+1}$, $\mathbf{K}_t$ and Q_t respectively defined in (2.4) and (1.21).

Lemma 2.4 (Compute $\mathbf{u} = (\mathbf{L}^T)^{-1}\tilde{\mathbf{x}}$ with linear computational cost)**.** *For any $\tilde{\mathbf{x}}$, let $u_N = Q_N^{-1/2} \tilde{x}_N$, and $u_{N-1} = Q_{N-1}^{-1/2} \tilde{x}_{N-1} - \ell_{N,N-1} u_N$. For $t = N-2, \dots, 1$, iteratively compute u_t using*

$$u_t = Q_t^{-\frac{1}{2}} \tilde{x}_t - \tilde{\ell}_{t+1,t} - \ell_{t+1,t} u_{t+1}, \tag{2.10}$$

with $\tilde{\ell}_{t+1,t}$ and $\ell_{t+1,t}$ defined in Lemma 2.1. Then $\mathbf{u} = (\mathbf{L}^T)^{-1}\tilde{\mathbf{x}}$.

2.3 Nonparametric estimation of particle interaction functions

The IKF algorithm is motivated by the problem of estimating interactions between particles, which is crucial to understanding complex behaviors of molecules and active matter, such as migrating cells and flocking birds. Minimal physical models, such as the Vicsek model [27] and its extensions [67, 68], provide a framework for explaining the collective motion of active matter.

Consider a physical model encompassing multiple types of latent interactions. Let $\mathbf{y}_i$ be a D_y-dimensional output vector corresponding to particle i' at time τ, which is influenced by J distinct interaction types. For the j-th type of interaction, the i-th observational vector interacts with a subset $p_{i,j}$ of particles rather than with all particles, typically those within a radius r_j. This relationship is expressed as a latent factor model:

$$\mathbf{y}_i = \sum_{j=1}^{J} \sum_{k=1}^{p_{i,j}} \mathbf{a}_{i,j,k} z_j(d_{i,j,k}) + \boldsymbol{\epsilon}_i, \tag{2.11}$$

with $\boldsymbol{\epsilon}_i \sim \mathcal{MN}(0, \sigma_0^2 I_{D_y})$ denoting a Gaussian noise vector. The term $\mathbf{a}_{i,j,k}$ is a D_y-dimensional *known* factor loading that links the i-th output to the k-th neighbor in the j-th interaction, with the *unknown* interaction function $z_j(\cdot)$ evaluated at a scalar-valued input $d_{i,j,k}$, such as the distance between particles i and k, for $k = 1, \ldots, p_{i,j}$, $i = 1, \ldots, n$, and $j = 1, \ldots, J$. For a dataset of n_p particles over n_τ time points, the total number of observations is $\tilde{N} = nD_y = n_p n_\tau D_y$.

An illustrative example of this framework is an unnormalized Vicsek model. Unlike the original Vicsek model [27], which enforces a constant speed for all particles, we allow velocity magnitudes to vary over time. Time-varying velocities are more sensible in many

real-world scenarios, including cellular movements studied in Section 2.4.4. When the system approaches a high degree of confluence, the average velocity of the cells typically decreases because of cell proliferation. Assume that the 2-dimensional velocity of the i'-th particle at time τ, $\mathbf{v}_{i'}(\tau) = (v_{i',1}(\tau), v_{i',2}(\tau))^T$, is updated by averaging the velocity of its neighbors from the previous time step. Specifically, for each direction $l = 1, 2$,

$$v_{i',l}(\tau) = \frac{1}{p_{i'}(\tau - 1)} \sum_{k \in ne_{i'}(\tau-1)} v_{k,l}(\tau - 1) + \epsilon_{i',l}(\tau), \tag{2.12}$$

where $\epsilon_{i'}(\tau)$ is a zero-mean Gaussian noise with variance σ_0^2. The set of neighbors $ne_{i'}(\tau - 1)$ includes particles within a radius of r from particle i' at time $\tau - 1$, i.e., $ne_{i'}(\tau - 1) = \{k : \|\mathbf{s}_{i'}(\tau - 1) - \mathbf{s}_k(\tau - 1)\| < r\}$, where $\mathbf{s}_{i'}(\tau - 1)$ and $\mathbf{s}_k(\tau - 1)$ are 2-dimensional position vectors of particle i' and its neighbor k, respectively, and $p_{i'}(\tau - 1)$ denotes the total number of neighbors of the i'-th particle, including itself, at time $\tau - 1$. The illustration of this model is shown in Fig. B.1(b) in the Appendix B.3.

The unnormalized Vicsek model in (2.12) is a special case of the general framework in (2.11) with scalar output $\mathbf{y}_i = v_{i',l}(\tau)$, i.e. $D_y = 1$, and the index of the i-th observation $i = 2(\tau - 1)n_p + 2(i' - 1) + l$ is a function of time τ and particle i'. This model contains a single interaction, i.e. $J = 1$, with a linear interaction function $z(d) = d$, where d is the velocity component of the neighboring particle, and the factor loading being $1/p_{i'}(\tau - 1)$.

Numerous other physical models of self-propelled particles can also be formulated as in (2.11) with a parametric form of a particle interaction function [67]. Nonparametric estimation of particle interactions is preferred when the physical mechanism is unknown [69, 70], but the high computational cost limits its applicability to systems with large numbers of particles over long time periods.

In this study, we model each latent interaction function $z_j(\cdot)$ nonparametrically

using a GP. By utilizing an equivalent DLM representation, we can leverage the proposed IKF algorithm to significantly expedite computation. This includes commonly used kernels such as the Matérn kernel with a half-integer roughness parameter [23], often used as the default setting for GP models to predict nonlinear functions, as the smoothness of the process can be controlled by its roughness parameters [45].

For each interaction j, we form the input vector $\mathbf{d}_j^{(u)} = [d_{1,j}^{(u)}, \ldots, d_{N_j,j}^{(u)}]^T$, where superscript (u) indicates that the input entries are unordered, $d_{t,j}^{(u)} = d_{i,j,k}$ with $t = \sum_{i'=1}^{i-1} p_{i',j} + k$ for any tuple (i, j, k), $k = 1, \ldots, p_{i,j}$, $i = 1, \ldots, n$, $j = 1, \ldots, J$, and $N_j = \sum_{i=1}^{n} p_{i,j}$. The marginal distribution of the j-th factor follows

$$\mathbf{z}_j^{(u)} = \left[z_j(d_{1,j}^{(u)}), \ldots, z_j(d_{N_j,j}^{(u)})\right]^T \sim \mathcal{MN}\left(\mathbf{0},\, \mathbf{\Sigma}_j^{(u)}\right),$$

where the (t, t')-th entry of the $N_j \times N_j$ covariance matrix $\mathbf{\Sigma}_j^{(u)}$ is $\mathrm{Cov}[z_j(d_{t,j}^{(u)}), z_j(d_{t',j}^{(u)})] = \sigma_j^2 c_j(d_{t,j}^{(u)}, d_{t',j}^{(u)}) = \sigma_j^2 c_j(d)$, with $d = |d_{t,j}^{(u)} - d_{t',j}^{(u)}|$, and σ_j^2 and $c_j(\cdot)$ the variance parameter and correlation function, respectively. We employ the Matérn correlation with roughness parameters $\nu_j = 1/2$ and $\nu_j = 5/2$ introduced in Section 1.1 for each interaction j.

By integrating out latent factor processes $\mathbf{z}_j$, the marginal distribution of the observational vector $\mathbf{y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_n^T)^T$, with dimension $\tilde{N} = nD_y$, follows a multivariate normal distribution:

$$\left(\mathbf{y} \mid \mathbf{\Sigma}_j^{(u)}, \sigma_0^2\right) \sim \mathcal{MN}\left(\mathbf{0}, \sum_{j=1}^{J} \mathbf{A}_j \mathbf{\Sigma}_j^{(u)} \mathbf{A}_j^T + \sigma_0^2 \mathbf{I}_{\tilde{N}}\right). \tag{2.13}$$

Here, $\mathbf{A}_j$ is a sparse $\tilde{N} \times N_j$ block diagonal matrix, such that the i-th diagonal block is a $D_y \times p_{i,j}$ matrix $\mathbf{A}_{i,j} = [\mathbf{a}_{i,j,1}, \ldots, \mathbf{a}_{i,j,p_{i,j}}]$ for $i = 1, \ldots, n$. The posterior predictive

distribution of the latent variable $z_j(d^*)$ for any given input d^* is also a normal distribution:

$$\big(z_j(d^*) \mid \mathbf{y}, \sigma_0^2, \boldsymbol{\sigma}^2, \boldsymbol{\gamma}, \mathbf{r}\big) \sim \mathcal{N}(\hat{z}_j(d^*), \sigma_j^2 c_j^*(d^*)), \tag{2.14}$$

with the predictive mean and variance given by

$$\hat{z}_j(d^*) = \boldsymbol{\Sigma}_j^{(u)}(d^*)^T \mathbf{A}_j^T \boldsymbol{\Sigma}_y^{-1} \mathbf{y}, \tag{2.15}$$

$$\sigma_j^2 c_j^*(d^*) = \sigma_j^2 c_j(d^*, d^*) - \boldsymbol{\Sigma}_j^{(u)}(d^*)^T \mathbf{A}_j^T \boldsymbol{\Sigma}_y^{-1} \mathbf{A}_j \boldsymbol{\Sigma}_j^{(u)}(d^*). \tag{2.16}$$

Here, $\boldsymbol{\Sigma}_y = \sum_{j=1}^J \mathbf{A}_j \boldsymbol{\Sigma}_j^{(u)} \mathbf{A}_j^T + \sigma_0^2 \mathbf{I}_{\tilde{N}}$, $\boldsymbol{\Sigma}_j^{(u)}(d^*) = \sigma_j^2 [c_j(d_{1,j}^{(u)}, d^*), \ldots, c_j(d_{N_j,j}^{(u)}, d^*)]^T$ and $\mathbf{r}$ represents additional system parameters.

The main computational challenge in calculating the predictive distribution in (2.14) lies in inverting the covariance matrix $\boldsymbol{\Sigma}_y$, where $\tilde{N}$ and N_j range between 10^5 and 10^6 in our applications. Though various GP approximation methods have been proposed [19, 17, 71, 72, 62, 22, 63, 20], they typically approximate the parametric covariance $\boldsymbol{\Sigma}_j^{(u)}$ rather than $\boldsymbol{\Sigma}_y$, and thus they are not directly applicable for computing the distribution in (2.14).

To address this, we employ the CG algorithm [59] to accelerate the computation of the predictive distribution. Each CG iteration needs to compute $\boldsymbol{\Sigma}_y \mathbf{u}$ for an $\tilde{N}$-dimensional vector $\mathbf{u}$. To employ the IKF, we first rearrange the input vector $\mathbf{d}_j^{(u)}$ into a non-decreasing input sequence $\mathbf{d}_j = [d_{1,j}, \ldots, d_{N_j,j}]^T$. Denote the covariance $\mathrm{Cov}[\mathbf{z}_j] = \boldsymbol{\Sigma}_j$ with $\mathbf{z}_j = [z_j(d_{1,j}), \ldots, z_j(d_{N_j,j})]^T$. The covariance of the observations can be written as a weighted sum of $\boldsymbol{\Sigma}_j$ by introducing a permutation matrix for each interaction: $\boldsymbol{\Sigma}_y = \sum_{j=1}^J \mathbf{A}_{\pi,j} \boldsymbol{\Sigma}_j \mathbf{A}_{\pi,j}^T + \sigma_0^2 \mathbf{I}_N$, where $\mathbf{A}_{\pi,j} = \mathbf{A}_j \mathbf{P}_j^T$ with $\mathbf{P}_j$ a permutation matrix such that $\boldsymbol{\Sigma}_j = \mathbf{P}_j \boldsymbol{\Sigma}_j^{(u)} \mathbf{P}_j^T$. After this reordering, the computation of $\boldsymbol{\Sigma}_y \mathbf{u}$ can be broken into four steps:

(1) $\mathbf{u}_j = \mathbf{A}_{\pi,j}^T \mathbf{u}$,

(2) $\mathbf{x}_j = \mathbf{\Sigma}_j \mathbf{u}_j$,

(3) $\hat{\mathbf{u}}_j = \mathbf{A}_{\pi,j} \mathbf{x}_j$,

(4) $\sum_{j=1}^{J} \hat{\mathbf{u}}_j + \sigma_0^2 \mathbf{u}$.

Here, $\mathbf{A}_j$ (and thus $\mathbf{A}_{\pi,j}$) is a sparse matrix with $N_j D_y$ non-zero entries. The IKF algorithm is used in step (2) to accelerate the most expensive computation, with all computations performed directly using terms in the KF algorithm without explicitly constructing the covariance matrix.

We refer to the resulting approach as the IKF-CG algorithm. This approach reduces the computational cost for computing the posterior distribution from $\mathcal{O}(\tilde{N}^3)$ operations to pseudolinear order with respect to N_j, as shown in Table B.1 in the Appendix B.6. Furthermore, the IKF-CG algorithm can accelerate the parameter estimation via both cross-validation and maximum likelihood estimation, with the latter requiring an additional approximation of the log-determinant [66]. Details of the CG algorithm, the computation of the predictive distribution, parameter estimation procedures, and computational complexity are discussed in Sections B.2, B.3, B.5, and B.6 of the Appendix B, respectively.

2.4 Numerical results for particle interaction function estimation

2.4.1 Evaluation criteria

We conduct simulation studies in Sections 2.4.2-2.4.3 and analyze real cell trajectory data in Section 2.4.4 to estimate particle interaction functions in physical models. The code and data to reproduce all numerical results are available at `https://github.com/UncertaintyQuantification/IKF`. We have implemented the main functions of the IKF

and IKF-CG algorithms for estimating particle interaction functions in the `FastGaSP` package on CRAN [73]. The observations in simulation are generated at equally spaced time frames $\tau = 1, \ldots, n_\tau$ with interval $\Delta t = 0.1$, though the proposed approach is applicable to both equally and unequally spaced time frames. For simplicity, the number of particles n_p is assumed constant across all time frames during simulations.

For each of the J latent factors, predictive performance is assessed using the normalized root mean squared error (NRMSE_j), the average length of the 95% posterior credible interval ($L_j(95\%)$), and the proportion of interaction function values covered within the 95% posterior credible interval ($P_j(95\%)$), based on n^* test inputs $\mathbf{d}_j^* = (d_{1,j}^*, \ldots, d_{n^*,j}^*)^T$:

$$\text{NRMSE}_j = \left(\frac{\sum_{i=1}^{n^*} (\hat{z}_j(d_{i,j}^*) - z_j(d_{i,j}^*))^2}{\sum_{i=1}^{n^*} (\bar{z}_j - z_j(d_{i,j}^*))^2} \right)^{1/2}, \tag{2.17}$$

$$L_j(95\%) = \frac{1}{n^*} \sum_{i=1}^{n^*} \text{length}\left\{ CI_{i,j}(95\%) \right\}, \tag{2.18}$$

$$P_j(95\%) = \frac{1}{n^*} \sum_{i=1}^{n^*} 1_{z_j(d_{i,j}^*) \in CI_{i,j}(95\%)}. \tag{2.19}$$

Here, $\hat{z}_j(d_{i,j}^*)$ represents the predicted mean at $d_{i,j}^*$, $\bar{z}_j$ is the average of the j-th interaction function, and $CI_{i,j}(95\%)$ denotes the 95% posterior credible interval of $z_j(d_{i,j}^*)$, for $j = 1, \ldots, J$. A desirable method should have a low NRMSE_j, small $L_j(95\%)$, and $P_j(95\%)$ close to 95%.

To account for variability due to initial particle positions, velocities, and noise, each simulation scenario is repeated $E = 20$ times to compute the average performance metrics. Unless otherwise specified, parameters are estimated by cross-validation, with 80% of the data used as a training set and the remaining 20% as the validation set. All computations are performed on a macOS Mojave system with an 8-core Intel i9 processor running at 3.60 GHz and 32 GB of RAM.

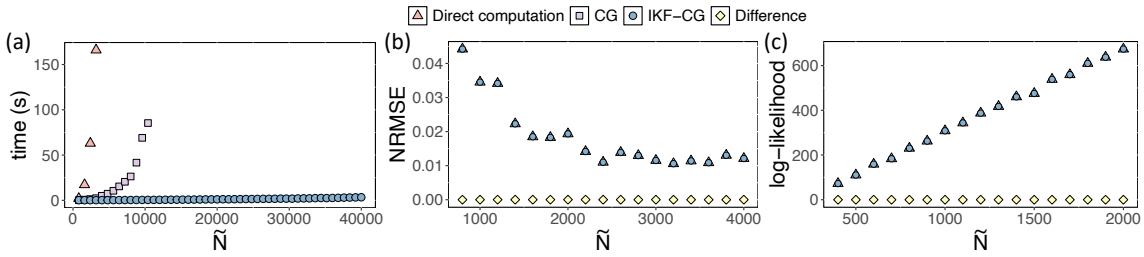

Figure 2.3: (a) Computational time of calculating predictive means using the IKF-CG algorithm (blue dots), CG method (purple squares), and direct computation (red triangles) for the unnormalized Vicsek model with different numbers of observations. (b), (c) NRMSE of predictive means for the interaction function and log-likelihood computed via direct computation (red triangles) and the IKF-CG algorithm (blue dots), with differences (yellow diamonds) of orders 10^{-7} to 10^{-5} and 10^{-5} to 10^{-4}, respectively.

2.4.2 Unnormalized Vicsek model

We first consider the unnormalized Vicsek model introduced in Section 2.3. For each particle i' at time τ, after obtaining its velocity using (2.12), its position is updated as

$$\mathbf{s}_{i'}(\tau) = \mathbf{s}_{i'}(\tau - 1) + \mathbf{v}_{i'}(\tau)\Delta t, \quad i' = 1, \ldots, n_p, \tag{2.20}$$

where Δt is the time step size. The initial velocity of each particle is set to $\mathbf{v}_{i'}(0) = [v\cos(\phi_{i'}(0)), v\sin(\phi_{i'}(0))]^T$, where $\phi_{i'}(0)$ is drawn from a uniform distribution on $[-\pi, \pi]$ and $v = 2^{1/2}/2 \approx 0.71$. Initial particle positions $\mathbf{s}_{i'}(0)$ in (2.20) are uniformly sampled from $[0, {n_p}^{1/2}] \times [0, {n_p}^{1/2}]$ to keep consistent particle density across experiments. The goal is to estimate the interaction function nonparametrically, without assuming linearity. The interaction radius $r = 0.5$ is estimated alongside other parameters, with the estimation results detailed in the Appendix B.7.

We first compare the computational cost and accuracy of the IKF-CG, CG, and direct computation for calculating the predictive mean in (2.15) and the marginal likelihood in (2.13) using the Matérn covariance in (1.2). Simulations are conducted with $n_p = 100$

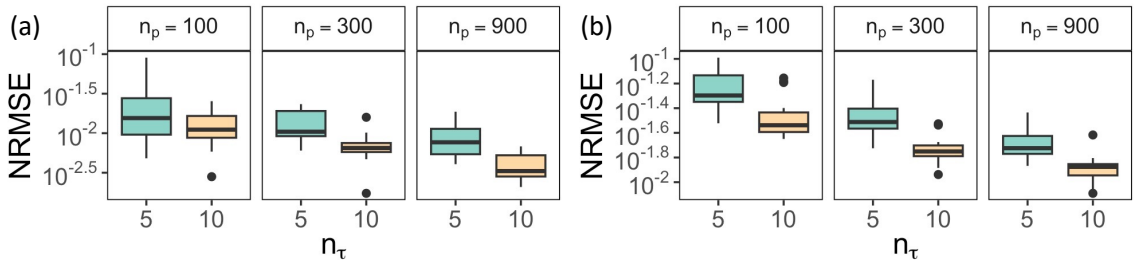

Figure 2.4: Boxplots of NRMSE in (2.17) for estimating the latent interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.1^2$ based on 20 experiments. Panel (a) uses the Matérn covariance with roughness parameter 5/2, and panel (b) uses the exponential covariance.

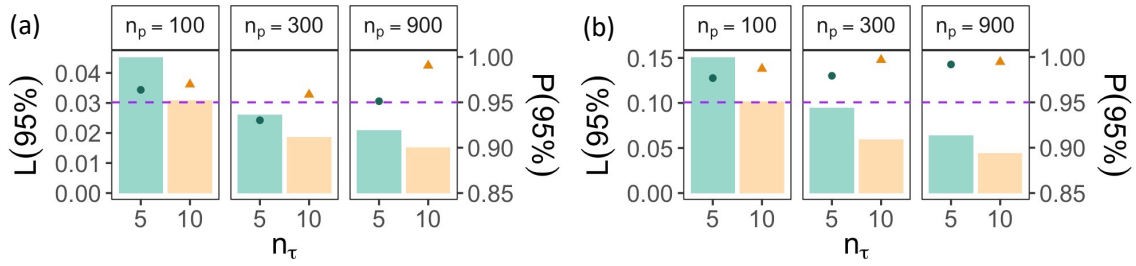

Figure 2.5: Uncertainty assessment of predicted interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.1^2$ using (a) Matérn covariance function in (1.2) and (b) exponential covariance function. Bars represent average lengths of 95% posterior credible intervals (2.18). Dots indicate the proportions covered in the 95% intervals (2.19), with the optimal coverage (0.95) shown as the purple dashed lines.

particles and noise variance $\sigma_0^2 = 0.2^2$. Figure 2.3(a) shows that the IKF-CG approach substantially outperforms both direct computation and the CG algorithm in computational time when predicting $n^* = 100$ test inputs with varying time lengths n_τ, ranging from 4 to 200. Figure 2.3(b) compares the NRMSE of the predictive mean between the direct computation and IKF-CG method when the number of observations ranges from 400 to 2,000, corresponding to n_τ from 4 to 20. The two methods yield nearly identical predictive errors, with negligible differences in NRMSE. Figure 2.3(c) shows a comparison between the log-likelihood values computed via the direct computation and the IKF-CG algorithm, where the latter employs the log-determinant approximation detailed in the Appendix B.5. Both methods produce similar log-likelihood values across different sample sizes.

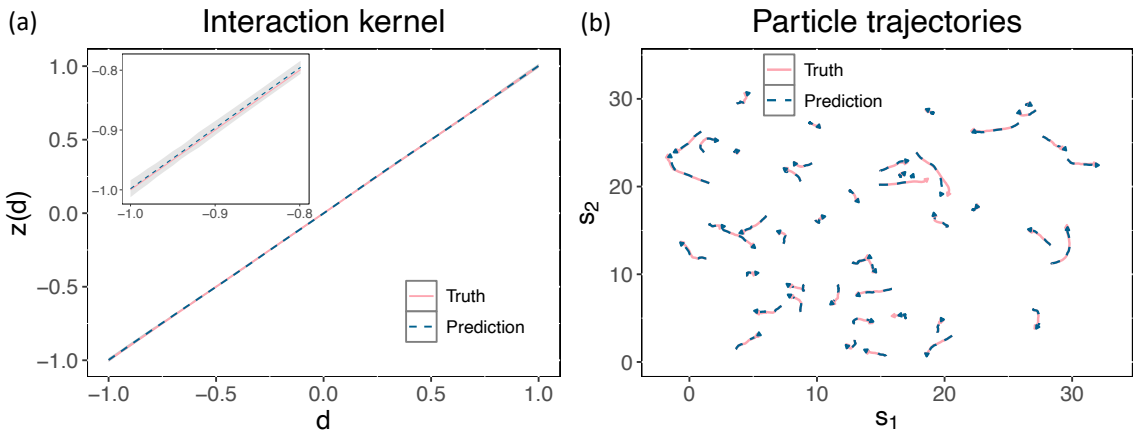

Figure 2.6: (a) Predicted (blue dashed lines) and true (pink solid lines) interaction functions with the 95% posterior credible interval (grey shaded area) for the unnormalized Vicsek model with $n_\tau = 900$, $T = 10$, and $\sigma_0^2 = 0.1^2$, using Matérn covariance function with $\nu = 5/2$. (b) Trajectories of 45 randomly sampled particles over 50 time frames using estimated (blue dashed lines) and true (pink solid lines) interaction functions with identical initial positions and noise samples.

Next, we evaluate the performance of the IKF-CG algorithm across 12 scenarios with varying particle numbers ($n_p = 100, 300, 900$), time frames ($n_\tau = 5, 10$), and noise variances ($\sigma_0^2 = 0.1^2, 0.2^2$). The predictive performance is evaluated using $n^* = 200$ test velocity inputs evenly spaced across a representative input domain of the interaction function in the data, $[-1, 1]$, averaged over $E = 20$ experiments.

Panels (a) and (b) of Figures 2.4-2.5 show the predictive performance of particle interactions using the Matérn covariance with roughness parameter $\nu = 5/2$ and the exponential covariance (Matérn covariance with $\nu = 1/2$), respectively, for noise variance $\sigma_0^2 = 0.1^2$. Results for $\sigma_0^2 = 0.2^2$ are provided in the Appendix B.7. Across all scenarios, NRMSEs remain low, with improved accuracy for larger datasets and smaller noise variances. The decrease in the average length of the 95% posterior credible interval with increasing sample size, along with the relatively small interval span compared to the output range, indicates improved prediction confidence with more observations. Moreover, the coverage proportion for the 95% posterior credible interval is close to the nominal 95% level in all cases,

validating the reliability of the uncertainty quantification. By comparing panels (a) and (b) in Figures 2.4-2.5, we find that the Matérn covariance in (1.2) yields lower NRMSE and narrower 95% posterior credible intervals than the exponential kernel. This improvement is due to the smoother latent process induced by (1.2), which is twice mean-squared differentiable, while the process with an exponential kernel is not differentiable.

Figure 2.6(a) shows close agreement between predicted and true interaction functions. The 95% posterior credible interval is narrow yet covers approximately 95% of the test samples of the interaction function. Figure 2.6(b) compares particle trajectories over 50 time steps by the predicted mean of one-step-ahead predictions and the true interaction function, both having the same noise vectors. The trajectories are visually similar.

2.4.3 A modified Vicsek model with multiple interactions

Various extensions of the Vicsek model have been studied to capture more complex particle dynamics [67, 68]. For illustration, we consider a modified Vicsek model with two interaction functions. The 2-dimensional velocity $\mathbf{v}_{i'}(\tau) = (v_{i',1}(\tau), v_{i',2}(\tau))^T$, corresponding to the output in (2.11), is updated according to:

$$\mathbf{v}_{i'}(\tau) = \frac{\sum_{k \in ne_{i'}(\tau-1)} \mathbf{v}_k(\tau-1)}{p_{i'}(\tau-1)} + \frac{\sum_{k \in ne'_{i'}(\tau-1)} f(d_{i',k}(\tau-1))\mathbf{e}_{i',k}(\tau-1)}{p'_{i'}(\tau-1)} + \boldsymbol{\epsilon}_{i'}(\tau), \quad (2.21)$$

where $\boldsymbol{\epsilon}_{i'}(\tau) = (\epsilon_{i',1}(\tau), \epsilon_{i',2}(\tau))^T$ is a Gaussian noise vector with variance σ_0^2 and $D_y = 2$. The first term in (2.21) models velocity alignment with neighboring particles, and the second term introduces a distance-dependent interaction function $f(\cdot)$. The definitions of neighbor sets, 2-dimensional vector $\mathbf{e}_{i',k}(\tau-1)$, and the second interaction function $f(\cdot)$ are provided in Appendix B.7.2.

We simulate 12 scenarios, each replicated $E = 20$ times, using the same number

Table 2.1: The predictive accuracy and uncertainty assessment by (2.17)-(2.19) for the modified Vicsek model with $\sigma_0^2 = 0.1^2$ using Matérn covariance function and roughness parameter $\nu_j = 5/2$.

		First Interaction			Second Interaction		
		NRMSE_1	$L_1(95\%)$	$P_1(95\%)$	NRMSE_2	$L_2(95\%)$	$P_2(95\%)$
$n_p = 100$	$n_\tau = 5$	6.8×10^{-3}	0.098	92%	8.7×10^{-2}	0.29	94%
	$n_\tau = 10$	6.0×10^{-3}	0.097	95%	3.8×10^{-2}	0.18	96%
$n_p = 300$	$n_\tau = 5$	1.2×10^{-2}	0.23	87%	3.2×10^{-2}	0.21	96%
	$n_\tau = 10$	4.2×10^{-3}	0.10	98%	2.0×10^{-2}	0.13	97%
$n_p = 900$	$n_\tau = 5$	5.6×10^{-3}	0.10	96%	1.4×10^{-2}	0.12	98%
	$n_\tau = 10$	4.4×10^{-3}	0.12	97%	1.4×10^{-2}	0.10	96%

of particles, time frame, and noise level as in the unnormalized Vicsek model in Section 2.4.2. The predictive performance of the IKF-CG algorithm is evaluated using $n^* = 200$ test inputs evenly spaced across the input domain of each interaction function. We focus on the model with the Matérn kernel in (1.2), with the results for the exponential kernel reported in the Appendix B.7. Consistent with the results of the unnormalized Vicsek model, we find that models with the Matérn kernel in (1.2) are more accurate due to the smooth prior imposed on the interaction function.

Table 2.1 summarizes the predictive performance for $\sigma_0^2 = 0.1^2$; results for $\sigma_0^2 = 0.2^2$ are reported in the Appendix B.7. While both interaction functions exhibit relatively low NRMSEs, the second interaction function has a higher NRMSE than the first, because the repulsion at short distances creates fewer training samples with close-proximity neighbors. This discrepancy can be mitigated by increasing the number of observations. The average length of the 95% posterior credible interval for the second interaction decreases as the sample size increases, and the coverage proportion remains close to the nominal 95% level across all scenarios.

In Figure 2.7, we show the predictions of the interaction functions, where the shaded regions represent the 95% posterior credible intervals. The input velocities are larger than the unnormalized Vicsek model due to the inclusion of the second interaction function,

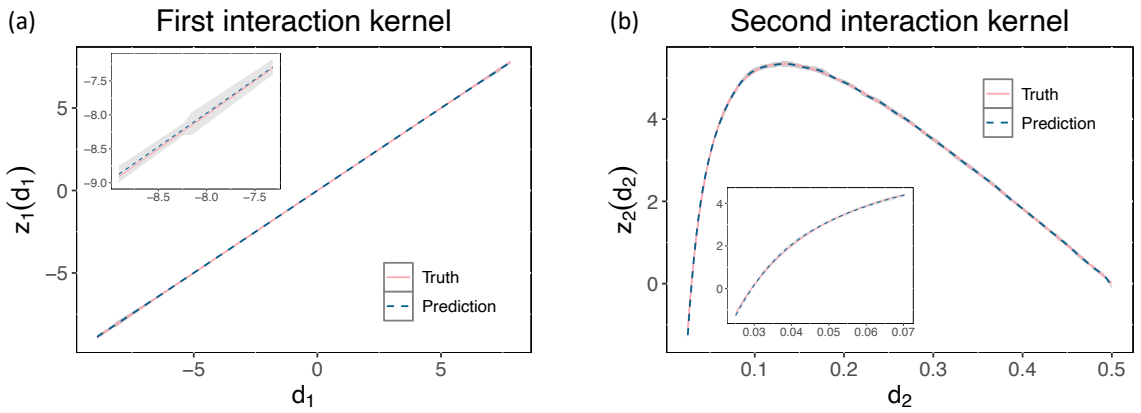

Figure 2.7: Predictions (blue dashed curves), truth (pink solid curves), and the 95% posterior credible interval (shaded regions) of particle interaction functions when $n_p = 900$, $n_\tau = 10$, and $\sigma_0^2 = 0.1^2$.

and thus the test input domain in Figure 2.7(a) is wider than that in Figure 2.6(a). The predictions closely match the true values, and the credible intervals, while narrow relative to the output range and nearly invisible in the plots, mostly cover the truth. These results suggest high confidence in the predictions.

2.4.4 Estimating cell-cell interactions on anisotropic substrates

We analyze video microscopy data from [74], which tracks the trajectories of over 2,000 human dermal fibroblasts moving on a nematically aligned, liquid-crystalline substrate. This experimental setup encourages cellular alignment along a horizontal axis, with alignment order increasing over time, though the underlying mechanism remains largely unknown. Cells were imaged every 20 minutes over a 36-hour period, during which the cell count grew from 2,615 to 2,953 due to proliferation. Our objective is to estimate the latent interaction function between cells. Given the vast number of velocity observations ($\tilde{N} \approx 300{,}000$), direct formation and inversion of the covariance matrix is impractical.

We apply the IKF-CG algorithm to estimate the latent interaction function. Because of the anisotropic substrate, cellular velocities differ between horizontal and vertical directions,

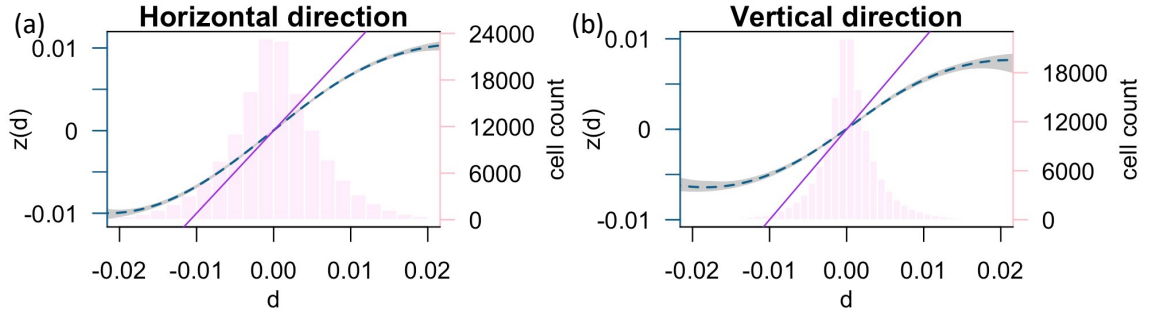

Figure 2.8: The blue dashed curves show the predicted interaction function for the horizontal and vertical directions. The grey shaded area is the 95% posterior credible interval. The purple line of slope 1 is the prediction using the unnormalized Vicsek model. The light pink histogram shows the velocity distribution of all training samples.

so we independently model the velocity of the i'-th cell in each direction l as

$$v_{i',l}(\tau) = \frac{1}{p_{ne_{i',l}(\tau-1)}} \sum_{k \in ne_{i',l}(\tau-1)} z_l(d_{k,l}) + \epsilon_{i',l}(\tau), \quad i' = 1, \ldots, n_p(\tau) \tag{2.22}$$

where $n_p(\tau)$ is the cell count at time τ, $l = 1, 2$ correspond to horizontal and vertical directions, respectively, and $\epsilon_{i',l}(\tau) \sim \mathcal{N}(0, \sigma^2_{0,l}(\tau))$ denotes the noise. Inspired by the modified Vicsek model, we set $d_{k,l} = v_{k,l}(\tau - 1)$. To account for velocity decay caused by increasing cell confluence, we model the noise variances as $\sigma^2_{0,l}(\tau) = \omega_l \sigma^2_{v,l}(\tau - 1)$, where $\sigma^2_{v,l}(\tau - 1)$ is the sample velocity variance at time $(\tau - 1)$, and ω_l is a parameter estimated from data for $l = 1, 2$. The neighboring set in (2.22), formed as $ne_{i',l}(\tau-1) = \{k : \|\mathbf{s}_{i'}(\tau-1) - \mathbf{s}_k(\tau-1)\| < r_l$ and $\mathbf{v}_{i'}(\tau - 1)^T \mathbf{v}_k(\tau - 1) > 0\}$, where r_l denotes the interaction radius of direction l for $l = 1, 2$, excludes particles moving in opposite directions, reflecting the observed gliding and intercalating behavior of cells [75].

We use observations from the first half of the time frames as the training set and the latter half as the testing set. The predicted interaction functions in Figure 2.8 show diminishing effects in both directions, likely due to cell-substrate interactions such as friction. The estimated uncertainty (obtained via residual bootstrap) increases when the absolute

Table 2.2: One-step-ahead prediction performance on the testing dataset. Here RMSE = $\{\sum_{\tau=1}^{n_\tau^*}\sum_{i'=1}^{n_p(\tau)}(\hat{v}_{i',l}(\tau)-v_{i',l}(\tau))^2/\sum_{\tau=1}^{n_\tau^*}n_p(\tau)\}^{1/2}$, with n_τ^* denoting the number of testing time frames. L(95%) and P(95%) are computed similarly to those in (2.18) - (2.19), but with the predictive interval of z replaced by that of y, where y represents the velocity in each direction.

	Horizontal direction			Vertical direction		
	RMSE	L(95%)	P(95%)	RMSE	L(95%)	P(95%)
Unnormalized Vicsek	3.9×10^{-3}	0.029	99%	2.6×10^{-3}	0.029	99%
Anisotropic Vicsek	3.9×10^{-3}	0.034	99%	2.6×10^{-3}	0.022	99%
Nonparametric estimation	3.5×10^{-3}	0.015	95%	2.3×10^{-3}	0.0092	95%

velocity of neighboring cells in the previous time frame is large, which is attributed to fewer observations at the boundaries of the input domain. Furthermore, the interaction in the vertical direction is weaker than in the horizontal direction, due to the confinement from the anisotropic substrate in the vertical direction.

Our nonparametric estimation of the interaction functions is compared with two models for one-step-ahead forecasts of directional velocities: the unnormalized Vicsek model introduced in Section 2.4.2, which uses a shared interaction radius for both velocity components, and the anisotropic Vicsek model, which uses distinct radii in each direction. As shown in Table 2.2, both Vicsek models have similar RMSE as the difference of interaction radii does not have a large impact for estimating interaction functions. Our model outperforms both Vicsek models in the RMSE for one-step-ahead forecasts of directional velocities, despite the large inherent stochasticity in cellular motion. Moreover, our model has notably shorter average interval lengths that cover approximately 95% of the held-out observations. These findings underscore the importance of the IFK-CG algorithm in enabling the use of large datasets to overcome high stochasticity and capture the underlying dynamics of cell alignment processes.

2.5 Implementation of Inverse Kalman Filter in `FastGaSP`

The methods introduced in this chapter have been implemented in the R package `FastGaSP` [73]. The package supports fast and exact computation of GPs with the Matérn kernel via the KF, and additionally provides implementations of the IKF algorithm (Section 2.2) and core functions for particle interaction estimation (Section 2.4). In this section, we describe the key functions relevant to our approach.

2.5.1 Core IKF functions

The `IKF` function implements Algorithm 2.1 to compute $\mathbf{\Sigma u}$, where $\mathbf{\Sigma}$ is the covariance matrix induced by a DLM and $\mathbf{u}$ is an arbitrary N-dimensional vector. This function supports the Matérn kernel with roughness parameters $1/2$ (`kernel_type = "exp"`) and $5/2$ (`kernel_type = "matern_5_2"`), with the latter being the default choice. The function requires four additional input arguments: the inverse range parameter $1/\gamma$, the noise variance $\sigma^2_{0,t}$ in Equation (1.18), a vector of consecutive differences between ordered inputs, and the output vector $\mathbf{u}$.

To demonstrate the usage of `IKF` function, we first generate $N = 2000$ sorted inputs, compute their consecutive differences, and create a random output vector:

```
> N <- 2000
> input <- sort(runif(N))
> delta_x <- input[-1] - input[-N]  # consecutive differences
> u <- rnorm(N)
```

We use the Matérn kernel with roughness parameter $5/2$ for demonstration purposes and set the inverse range parameter to 10. In the presence of observation noise with variance, we can directly use the `IKF` function by setting `tilde_nu` to the noise variance:

```
> beta <- 10  # inverse range parameter
> V <- 0.2  # noise variance
> x_IKF_noisy <- IKF(beta = beta, tilde_nu = V, delta_x = delta_x,
+                    output = u, kernel_type = "matern_5_2")
```

The `IKF` function substantially reduces computational time compared to direct matrix multiplication, as shown in Figure 2.2(a)-(b). To verify its accuracy, we compare the result with that of direct matrix computation and obtain a difference near zero.

```
> R0 <- abs(outer(input, input, "-"))
> R <- (1 + sqrt(5)*beta*R0 + 5*beta^2*R0^2/3) * exp(-sqrt(5)*beta*R0)
> x_direct_noisy <- (R + V * diag(N)) %*% u
> max(abs(x_direct_noisy - x_IKF_noisy))
[1] 1.669775e-13
```

We also demonstrate the use of the `IKF` function when there's no noise, where a value for `tilde_nu` is set as a stabilizing parameter for the robust IKF. The following example compares the robust IKF (with stabilization) to the non-robust IKF (without stabilization). The robust version maintains accuracy comparable to direct computation, while the non-robust version suffers from numerical instability:

```
> # Robust IKF (with stabilizing parameter)
> tilde_nu <- 0.1  # stabilizing parameter
> x_IKF <- IKF(beta = beta, tilde_nu = tilde_nu, delta_x = delta_x,
+              output = u, kernel_type = "matern_5_2") - tilde_nu * u
> # Non-robust IKF (no stabilization parameter)
> x_non_robust_IKF <- IKF(beta = beta, tilde_nu = 0, delta_x = delta_x,
+                         output = u, kernel_type = "matern_5_2")
> x_direct <- R %*% u
> max(abs(x_direct - x_IKF))
[1] 2.629008e-13
> max(abs(x_direct - x_non_robust_IKF))
[1] 1.462629
```

The error comparison between robust and non-robust IKF across varying sample sizes is demonstrated in Figure 2.2(c).

For users who wish to work directly with DLMs without going through the GP specification, the package offers additional flexibility. Instead of using the `IKF` function, users can manually specify the $\mathbf{G}_t$ and $\mathbf{W}_t$ matrices in the DLM representation and call the functions `Get_Q_K` and `Get_R_y`. The `Get_Q_K` function computes Q_t and Kalman gain $\mathbf{K}_t$ from the KF (Lemma 1.1), while `Get_R_y` computes $\boldsymbol{\Sigma}\mathbf{u}$ using these pre-computed quantities.

The `FastGaSP` package also includes fundamental operations based on the Cholesky decomposition factor $\mathbf{L}$ ($\mathbf{\Sigma} = \mathbf{L}\mathbf{L}^T$), which form the building blocks of the IKF algorithm. Specifically, the package provides four independent functions for efficiently computing matrix-vector products related to $\mathbf{L}$, applicable to any N-dimensional vector $\mathbf{u}$:

- `Get_L_y`: compute $\mathbf{L}\mathbf{u}$ (Lemma 2.2),
- `Get_L_t_y`: compute $\mathbf{L}^T\mathbf{u}$ (Lemma 2.1),
- `Get_L_inv_y`: compute $\mathbf{L}^{-1}\mathbf{u}$ (Lemma 1.3),
- `Get_L_t_inv_y`: compute $(\mathbf{L}^T)^{-1}\mathbf{u}$ (Lemma 2.4).

These functions take four arguments: the matrices $\mathbf{G}_t$ from DLM, Q_t, the Kalman gain $\mathbf{K}_t$ from KF, and the vector $\mathbf{u}$. Since these functions assume the KF quantities are pre-computed, each of them requires only $\mathcal{O}(\tilde{q}^2 N)$ operations, where $\tilde{q}$ is the dimension of the latent state.

2.5.2 Application to particle interaction estimation

The `FastGaSP` package implements several functions for the particle interaction estimation applications described in Section 2.4. These functions incorporate the IKF algorithm within the CG method to handle the general covariance structure in Equation (2.1), providing a complete workflow from data generation or preprocessing to parameter estimation and prediction.

For simulation studies, the `simulate_particle` function generates particle trajectory data according to the Vicsek-type models. It supports both the unnormalized Vicsek model (Section 2.4.2) and the modified Vicsek model with two interaction functions (Section 2.4.3). The only mandatory argument for the `simulate_particle` function is the initial velocity magnitude, for which there is no default value. By default, the funciton simulates 100 particles over 5 time steps with a time step size of 0.1, an interaction radius of 0.5, and Gaussian noise with a standard deviation of 0.1. We demonstrate the simulation of the

unnormalized Vicsek model with an initial velocity magnitude $v = \sqrt{2}/2$ using the default setting:

```
> v_abs <- sqrt(2)/2
> sim <- simulate_particle(v_abs=v_abs, model = "unnormalized_Vicsek")
```

This returns an object of class `particle.data` that contains the simulated particle positions, velocities, and other information.

For experimental data, the `trajectory_data` function processes raw particle tracking data into the standardized `particle.data` format. This function handles time series with a varying number of particles across time frames and maintains particle identity tracking between consecutive time steps. The input must be a data frame with columns for particle positions (`px`, `py`), velocities (`vx`, `vy`), time step (`time`), and particle identifier (`particleID`):

```
> # Load experimental data
> raw_data <- read.csv("cell_tracking.csv")
> traj <- trajectory_data(raw_data)
```

The `extract_time_window` function allows users to select specific time frames for train-test splitting or focused analysis.

The `fit` function performs parameter estimation and prediction for particle interaction systems using the IKF-CG algorithm. We illustrate its application using the simulated data generated previously. First, we define the testing inputs, an upper bound of the interaction radius, and initial parameter values:

```
> testing_inputs <- matrix(as.numeric(seq(-1,1,length.out=200)),nr=1)
> cut_r_max=1.5 # upper bound of the interaction radius
> # Initial parameter values
> param_ini <- log(c(0.3,1000,0.3/(cut_r_max-0.3)))
```

The parameter vector contains the log-transformed values of the inverse range parameter, the variance-noise ratio, and a transformed interaction radius ensured to be smaller than `cut_r_max`. These initial values serve as the starting point for the optimization process when

`est_param = TRUE` (the default). The function then estimates the parameters and computes predictions at the testing inputs:

```
> est <- fit(sim,param=param_ini,cut_r_max=cut_r_max,
+            testing_inputs = testing_inputs, compute_CI = TRUE)
```

The returned object contains the estimated parameters, predicted interaction function values at the testing inputs, and 95% credible intervals if `compute_CI = TRUE` (see Figure 2.6(a) for an example). Additional options include specifying the kernel type (`kernel_type = "matern_5_2"` or `"exp"`), restricting interactions to particles moving in the same direction (`apolar_vicsek = TRUE`), and setting the convergence tolerance for the CG algorithm (`tol`).

Chapter 3

Data-driven model construction for anisotropic dynamics of active matter

Uncovering the biophysical mechanisms behind collective cell movement is crucial for understanding embryonic development and tissue morphogenesis. Experimental observations show that under the influence of weak guidance from a molecularly aligned substrate, the orientational alignment of cells increases over time, accompanied by cell proliferation. This chapter, based on the classic Vicsek model framework (a model where particles update their velocity directions according to the average direction of nearby neighbors plus random fluctuations), constructs a parameterized model through data-driven statistical testing, thereby avoiding exhaustive testing of all feature combinations through simulation. This method, while maintaining model simplicity and interpretability, quantitatively reproduces the key system-level parameters: the temporal progression of the velocity orientational order parameter and the variability of velocity vectors. Compared to the non-parametric

estimation in Chapter 2, this chapter focuses more on parametric modeling that provides a better grasp of the underlying mechanisms. The material in this chapter is reproduced with permission from Mengyang Gu, Xinyi Fang, and Yimin Luo, "Data-Driven Model Construction for Anisotropic Dynamics of Active Matter," *PRX Life* **1**, 013009 (2023).

3.1 Introduction

Active matter refers to systems composed of many individual agents that interact with each other and consume energy from their surroundings or an internal source to generate complex, global behaviors [76, 77, 78]. It encompasses a wide range of biological systems, such as flocks of birds, schools of fish, groups of humans, herds of sheep, monolayers of cells, colonies of bacteria, and others, all exhibiting intriguing out-of-equilibrium phenomena. The dynamics of active matter, which we term "active dynamics", is the key to describing the collective, spatiotemporal self-organization that arises from interactions and decision-making at the individual agent level.

Disorder-to-order transitions in biological tissues, characterized by the emergence of patterns, are a common feature of cellular systems. These transitions have been compared to phases of matter, such as disordered gas and amorphous solids [79, 80]. They underlie many developmental processes [81], and are implicated in cancer invasion [82]. The resulting patterns impart tissue with form, function, and integrity, as seen in the basket-weave-like pattern of the dermis that serves as a shield for the deeper layers [83, 84], the generation of forces in blood vessels [85, 86], and the coordination of cell fates during embryonic development [87].

The mechanisms behind these cellular transitions are not well understood, but

traditionally, collective behaviors are thought to be regulated by cell chemosensing [2], and upstream biochemical signaling pathways [88]. While biochemical regulations are on-demand and short-lived, physical guidance ensures the generation of cell phenotypes and long-term maintenance of tissue structures [89]. More recently, simple two-dimensional transitions of such nature have been realized through *in vitro* experiments [90, 91, 92, 93], by subjecting a cell monolayer to an external guiding field, such as a well-aligned molecular field [74], which can lead cells to collectively orient along a predetermined axis in a density-dependent manner.

Modeling spontaneous alignment of particles in active matter systems has been the subject of numerous studies [27, 94, 95, 96, 68, 97]. The two main approaches in modeling are either by constructing mechanistic models through physical laws, or by applying data-driven methods to extract information from observations to construct models [98, 99]. Describing individual cell behaviors often starts from a Langevin equation of Brownian particles [100]: $m\dot{v} = -\zeta v + \epsilon_F$, with m, v and ζ being the mass, velocity and friction coefficient, respectively, and ϵ_F being a random force fluctuation, typically assumed to be a Gaussian white noise. Variants of it have been widely used for modeling persistent random motions of cells in experiments and simulation [101, 102, 103, 104, 105], whereas the complex cell-cell interaction at high density, often implicated in tissue formation and repair, is not explicitly considered in these models.

In data-driven methods, regression techniques, for instance, are widely applied to estimate linear coefficients of a set of additive basis [106, 107, 108], and to learn particle interaction kernel functions by piece-wise linear functions [70], neural networks [109], and Gaussian processes [110]. Learning the distance-based interaction kernel function has been successfully applied to model schools of golden shiner [69], flocks of surf scoters [111] and systems of interacting particles [70]. However, these approaches are rarely applied to

estimate cell-cell interaction from experiments with a large number of cellular trajectories, partly because the large computational cost prohibits accurate interaction kernel learning approaches [110]. Furthermore, as the underlying mechanism of these interactions remains largely unknown, a principled way to identify the relevant input of interaction kernel and delineating neighboring sets is needed as adding nonrelevant inputs can dramatically reduce estimation efficiency.

Despite the prevalence of the disorder-to-order transition observed in experiments [74, 112, 113], capturing the temporal progression of dynamic quantities in active matter systems, such as the orientational order parameter and variability of velocity, remains a challenging task due to a few reasons. First of all, active matter systems intrinsically contain large fluctuations. Gaussian distributions of fluctuation, for instance, are often assumed for modeling the velocity progression in constructing mechanistic models [114, 115]. In a data-driven approach, the model is usually optimized by minimizing a loss function. One common choice of the loss function is the sum of squared errors, and minimizing it is equivalent to finding the maximum likelihood estimator, under the assumption of having independent Gaussian noises with equal variances. However, we find that velocity distributions from our experiments significantly deviate from Gaussian. In fact, even though models assuming Gaussian distribution of the velocity fluctuation can fit the magnitude of temporally dependent velocity, they cannot capture the progression of orientational order parameters. Non-Gaussian distributions of displacements have been widely observed in biological systems, such as particles absorbed on lipid bilayers, immersed in entangled actin solutions [116], in a bath of swimming algal cells [117], liposomes in active actin solutions [118], and receptors on living cell membranes [119]. While models have been developed to explain the fluctuation of displacements independent of time [120, 121] and for how these distributions arise [122], the fluctuation distributions have rarely been incorporated

in simulation to reproduce the temporal progression of velocity and order parameters in nonequilibrium cell migrations.

Second, in principle, having a large number of observations containing thousands of particle trajectories over hundreds of time points can help offset the uncertainty of estimation due to large fluctuations of the active particles. Nonetheless, calibrating the simulation model with such a large number of observations is a nontrivial task. Placing a smooth Bayesian prior on the interaction kernel function, such as a Gaussian process prior, can partially filter the noise in estimation but also leads to large computational costs. Thus, most of the current studies of learning interaction kernels are restricted to a small number of particles and time frames [70, 123]. Hence, there is a need for robust and computationally scalable algorithms for feature selection and estimation from a large number of observations.

This chapter aims to address these issues by introducing an efficient workflow to automate model construction, promoting convergence between simulation and experiments. We take a hybrid approach that integrates physical models and statistical learning approaches: First, we start from a conventional form of the physical models (e.g. the Vicsek model [27, 28]). Second, we apply statistical tests for feature selections. Then, these features are utilized to construct a data-generative model, instead of minimizing a loss function as usually adopted in other data-driven discovery approaches. The data generative model provides a better physical interpretation of the estimated quantities, and more importantly, the uncertainty of the estimation can also be quantified. We develop an automated feature selection and estimation approach, which is applied to learning physical quantities from live microscopy of fibroblasts moving on liquid crystal elastomer substrates, to discover a variety of new features. Simulation models constructed with these new features can reproduce the progression of system properties, such as orientational order parameters and velocity distributions, whereas models missing any of these features do not match experimental

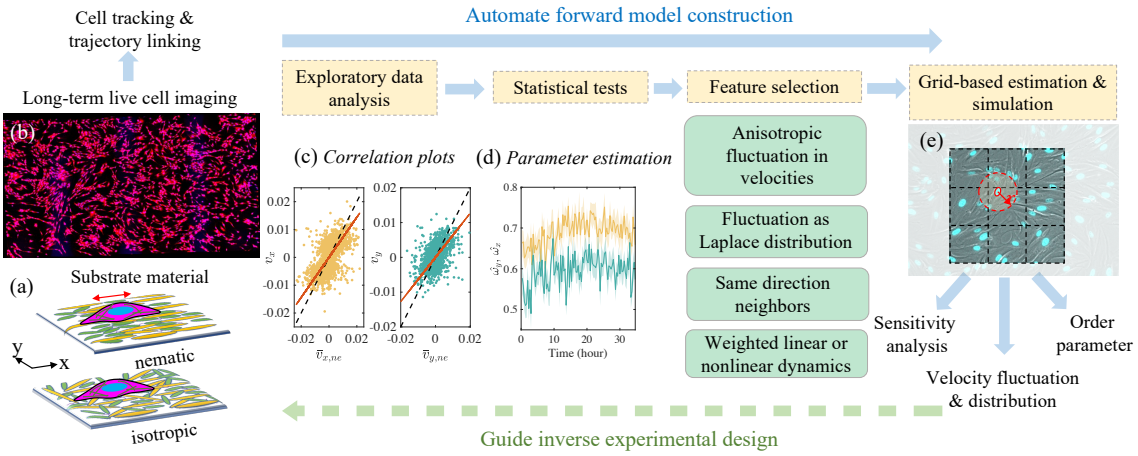

Figure 3.1: (a) Schematics of the two types of substrates (isotropic and nematic) tested in this study. On the nematic substrate, the molecular alignment (denoted by a red double-sided arrow) is parallel to x. (b) A stitched view of a snapshot of long-term cell imaging where the cell cytoplasm is labeled in red. (c) Correlation plots in exploratory data analysis, where x, y velocity components are plotted against those of the neighboring average in the previous step. The slope of the fit (red solid lines) is smaller than 1 (black dashed lines). (d) Parameter estimation of the slope parameters, ω, the slope value of the red solid line in (c), and their 95% confidence intervals (shaded area). The statistical tests yield four potential features (green blocks), representing new features added to the classical Vicsek model in fitting the current data set. (e) The grid-based system to search for the neighbors. In order to find the neighbor of the target cell i (circled in red), a radius of r will be searched. The computational procedure is simplified by only searching through neighboring squares connected to the square cell i is located in. This workflow generates various physical parameters via simulation to compare to quantities computed from the experiment.

findings.

The main contribution is to develop an efficient feature selection and estimation approach for cellular movement experiments on liquid crystal elastomer substrates from video microscopy. The four selected features can be classified into two groups. The first group concerns cell-substrate interactions. We found that the velocity of the cells is anisotropic and has heavier tails than a Gaussian distribution, motivating the use of a Laplace distribution or generalized Gaussian distribution of fluctuation for characterizing cell-substrate interaction. Though Laplace distribution was used to fit the probability distribution of the displacements [124, 116, 118] and the mechanism of exponentially distributed cellular motility was explored

in [122], its effect on capturing the progression of the orientational order parameter in simulations has not been demonstrated. The second group of discoveries concern cell-cell interactions. We observe from experiments that when cells interact, cell bodies become elongated, suggesting cells pulling each other along through a tensional network. Our findings indicate that cells are not perfectly aligned with the average of their neighbors in the prior time frame, as is the case in the classic Vicsek model [27]; Instead, effects from the previous step shrink towards zero, likely due to a frictional force term in the Langevin equation [100]. Additionally, we discovered that cells traveling in the opposite direction may be excluded from the neighboring set, leading to improved predictions of the velocity at the subsequent time point. This likely arises from the nematic nature of cell-cell interactions [112], where cells traveling in the opposite direction can simply glide past each other without influencing each other's velocity. Typical cell trajectories and interactions are illustrated in the Supplemental Material of [6] available in [125].

Furthermore, we develop scalable computational tools for analyzing and simulating dynamical processes of active matter. Figure 3.1 illustrates an instance of the particle-based module of our computational tools. In Figure 3.1(c), we present a plot showing the correlation between cellular velocity and the mean velocity of neighboring cells in the previous time point. The slopes of the best-fit lines (red solid curve) are smaller than 1 for both x and y directions. To determine if this result applies to all time points, we plot the fitted weight parameters (equivalent to the slope of the fit in Figure 3.1(c)) over time and calculate the 95% confidence interval, which is represented by the shaded area in Figure 3.1(d). The upper bounds of the 95% confidence intervals are substantially smaller than 1, indicating that the weight parameters must be included in the simulation model. All the identified features are included in the simulation model, and a maximum likelihood estimator is derived for efficient parameter estimation (Appendices C.1 and C.2). For both

parameter estimation and simulation, we employ a grid-based approach [126] by storing the particles in a coarse-grained grid (Figure 3.1(e)), which can improve the efficiency in searching for neighbors. With a typical microscopy video of $n \approx 2500$ cells and $T \approx 100$ time frames, the time it takes to perform feature selection, parameter estimation, and simulation of dynamics with particle interactions, is less than 30 seconds on a desktop computer. This provides almost immediate feedback that can be used to inversely guide the design of the experiment. The data sets and code used in the article have been made publicly available at `https://github.com/UncertaintyQuantification/data_driven_cell_model`.

3.2 Feature selection from experimental results

We begin by discussing experimental methods for live cell imaging and the process of extracting trajectories from cells cultured on aligned (nematic) and disordered (isotropic) liquid crystal elastomer substrates. We then conduct exploratory data analysis and statistical tests to identify significant features from the experimental data. These features are used as building blocks to extend a baseline model. Lastly, we perform simulations to validate our findings in Section 3.4.

3.2.1 Experimental setting: In vitro cell alignment experiment

The experimental data were derived from [74], consisting of cell trajectories. Liquid crystal elastomer (LCE) substrates were fabricated with a mixture of reactive monomers RM82 and RM23 (SYNTHON Chemicals GmbH & Co.) in a 1:1 molar ratio with the molecular field having either isotropic or nematic (uniform along the x-direction) configurations. The monomers were crosslinked to make a solid film, which was topographically flat but molecularly aligned, previously examined by wide angle X-ray scattering and atomic force microscopy [74]. Human dermal fibroblasts (HDFs, American Type Culture Collection)

were seeded onto the substrates at cell density $\rho \approx 50$ mm^{-2}, and grown to desired ρ. Prior to imaging, cell nuclei and cytoplasm were stained with Dyes Hoechst 33342 and CellTracker Deep Red (both from ThermoFisher), respectively. Cells are maintained at 37 ^{o}C, 5% CO_2 and imaged every 20 min for about 36 hours. Tens of images were taken at every time point and stitched together to construct an image on the order of square millimeters in area. LCE nematic substrates impose a molecular direction to guide HDF growth. During the experiment, cell density ρ grows from roughly 350 to 400 mm^{-2}. With an increase in cell density, the monolayer undergoes a disorder-to-order transition. Initially, there were approximately 2600 cells captured within the frame. When we finished imaging, the count had increased to about 2950 cells. To obtain the trajectories as inputs, we fit an ellipse to the nuclei and track particle trajectories using ImageJ [127], a standard cellular imaging tool. On the nematic substrate, cells develop a long-range order along the x-direction, the same direction as the molecular orientation in the aligned substrate, but not on isotropic substrates.

From the experiment, we extract the 2D position vector $\mathbf{s}_i(t) = (s_{i,x}(t), s_{i,y}(t))^T$ and velocity vector $\mathbf{v}_i(t) = (v_{i,x}(t), v_{i,y}(t))^T$ of cell i, at time frame t, for $i = 1, ..., n_t$, with n_t being the number of cells on time frame t, for $t = 1, ..., T$. The velocity is computed by dividing the displacement over time intervals as follows: $v_{i,x} = \frac{s_{i,x}(t+\Delta t) - s_{i,x}(t)}{\Delta t}$, and similarly for $v_{i,y}$. We then store the data matrix, where each row contains the 2D position, velocity and a unique cell identification number resulting from trajectory linking. The goal is to construct an interpretable model and constrain the model by data, for reproducing the temporally dependent variability of the velocity and orientational order parameter. We plot the standard deviation of the velocity at x and y directions at each time frame in Figure 3.2(a). Since the nucleus does not distinguish between the head and tail, its order is apolar, we cannot apply the polar order parameter to quantify the system alignment

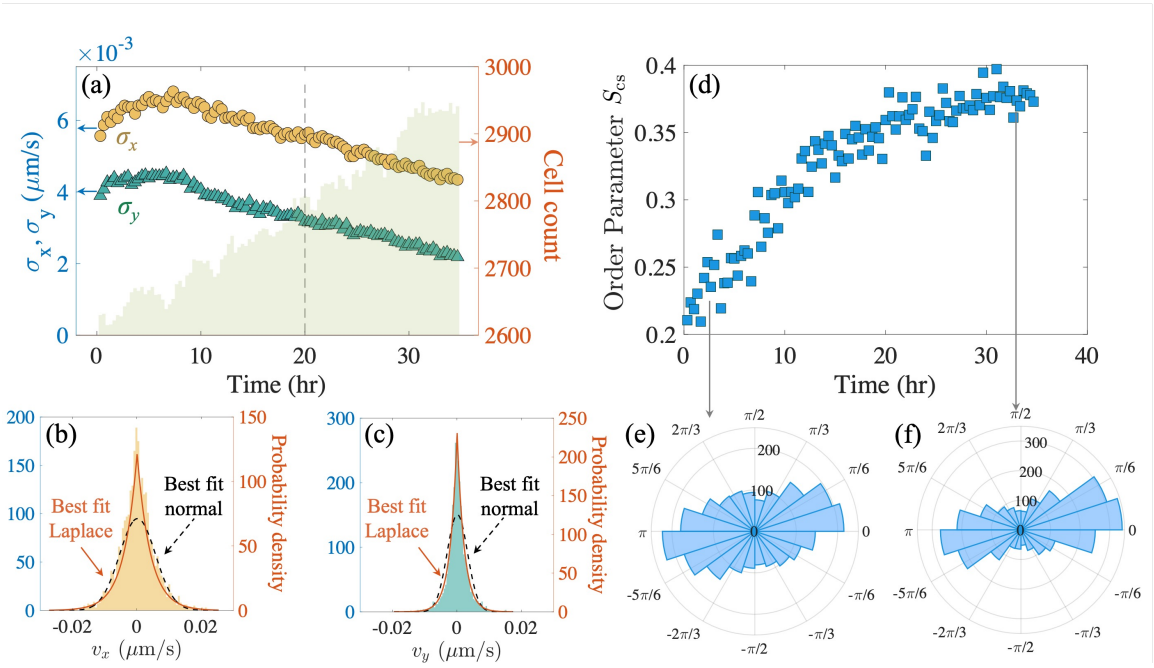

Figure 3.2: Parameters estimated from the data set. (a) The standard deviation of the cell velocities is shown in yellow circles and green triangles. The light-colored histogram denotes the cell number count (b-c). The velocity distribution at a representative time = 20 hours. The dashed black lines denote the best fit normal distribution. The solid orange lines denote the best fit Laplace distribution. (d) The blue squares denote the polar order parameter, and the polar histograms of the velocity angle at two different time points are plotted in (e-f).

[128]. Instead, we first transform velocity angle of the particle i at time frame t, denoted by $\phi_i(t) = \arctan(v_{i,y}(t)/v_{i,x}(t))$, to $[0, \pi]$ by letting:

$$\tilde{\phi}_i(t) = \begin{cases} \phi_i(t) + \pi & \text{if } \phi_i(t) < 0, \\ \phi_i(t) & \text{if } \phi_i(t) \geq 0. \end{cases} \tag{3.1}$$

The orientational order parameter at time t, a commonly used measure of collective alignment, is defined as $S_{\text{cs}}(t) = n_t^{-1} \sum_{i=1}^{n_t} \cos(2\tilde{\phi}_i(t))$, an ensemble of transformed velocity angles amongst all cells, where the subscripts "c" and "s" denote cells and substrate, respectively. This parameter takes values close to 1 when cell velocities are highly aligned and close to 0 when directions are random. This way, antiparallelly traveling cell pairs ($\phi_i = \phi_j + \pi$) have

the same contribution to orientational order parameters. In Figure 3.2(d), we show that the orientational order parameter $S_{\mathrm{cs}}(t)$ increases with time. The distributions of the velocity angles at two different time points are plotted in Figure 3.2(e) and (f), showing that over time the velocity distribution becomes more parallel along x-direction.

To account for observed migrational patterns, we start by identifying unexpected deviations of the magnitude and distributions from the classical Vicsek Model. Exploratory data analysis (EDA) [129] is a useful step to visualize patterns from complex experimental data before performing statistical tests and modeling. Here, we first perform EDA on various aspects of the data, followed by statistical tests and estimation to identify features to include in the modeling. The main text focuses on the analysis of an experiment where cells initially cover roughly 50% of the substrates. During the experiment, the order parameter increases rapidly with cell proliferation. We present results on the analysis of two additional experiments at different cell densities or on isotropic substrates in Appendix C.4.

3.2.2 Permutation F-test on variances of the velocities

To test whether the variances of the velocities along x and y directions are the same at all time frames, we first compute the sample standard deviation of the velocity vector at x and y coordinates: $\hat{\sigma}_x^2(t) = \sum_{i=1}^{n_t}(v_{i,x}(t) - \bar{v}_x(t))^2/(n_t - 1)$ and $\hat{\sigma}_y^2(t) = \sum_{i=1}^{n_t}(v_{i,y}(t) - \bar{v}_y(t))^2/(n_t - 1)$, where $\bar{v}_x(t)$ and $\bar{v}_y(t)$ are the mean of the velocity at time t computed from an ensemble of all cells in the system, and both are close to zero. As shown in Figure 3.2(a), we found that the standard deviation of the velocity along the x coordinate is always larger compared to that along y. The larger magnitude of cellular velocity along the x coordinate is induced by the liquid crystal elastomer substrate, which is molecularly aligned along the x direction in this experiment. As the velocity magnitudes along x and y directions become more unequal over time, the orientational order parameter increases (Figure 3.2(d)).

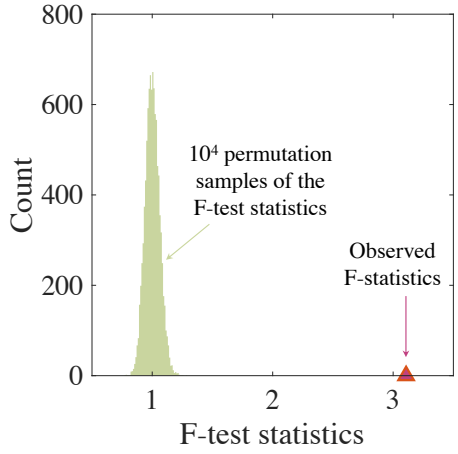

Figure 3.3: A permutation F-test shows that the variance of the fluctuation along x and y directions is indeed unequal at time =20 hrs.

Is the observed difference between $\sigma_x(t)$ and $\sigma_y(t)$ due to signal or noise? Typically, testing the equality of the variance is performed using the F-test assuming both populations are normally distributed. However, as we will see later, velocity distributions do not follow a normal (Gaussian) distribution, while the F-test is sensitive to the normality assumption [130].

Here we circumvent the normality assumption by running a permutation F-test. The permutation test is a general nonparametric test that is insensitive to the distributional assumption. At each time t, we begin by combining the velocities at x and y directions into a long vector. Subsequently, we randomly assign these combined velocity values into two groups of equal size, creating a permuted sample. We repeat this step B times to collect B permuted samples: $\mathbf{v}_x^{(b)}(t) = [v_{1,x}^{(b)}(t), ..., v_{n_t,x}^{(b)}(t)]$ and $\mathbf{v}_y^{(b)}(t) = [v_{1,y}^{(b)}(t), ..., v_{n_t,y}^{(b)}(t)]$ for $b = 1, ..., B$. Then we compute the permuted F-test statistics $F^{(b)}(t) = \hat{\sigma}_{b,x}^2(t)/\hat{\sigma}_{b,y}^2(t)$, where $\hat{\sigma}_{b,x}^2(t)$ and $\hat{\sigma}_{b,y}^2$ denote the sample variances of the velocity for x and y directions, respectively, from the permuted sample b. The p-value is twice the minimum of the probabilities $\Pr(F^{(b)}(t) > F(t))$ and $\Pr(F^{(b)}(t) < F(t))$, which can be computed empirically using the permuted samples.

Figure 3.3 shows the distribution of the F-test statistics of the permuted samples

and the observed F-test statistics for velocity observations at time equal to 20 hours. Since the observed F-test statistics is larger than any of the permutation sample, the p-value is smaller than 10^{-4}, indicating that the variance of the velocity along x and y directions is unequal at this time frame. We perform the permutation F-test for each of the time points and verify that the variances $\sigma_x^2(t) \neq \sigma_y^2(t)$ for all time points. Thus, cellular movements are anisotropic when cells are grown on a molecularly aligned substrate.

3.2.3 Shapiro-Wilk test for normality of velocity distribution

While at first glance cell motility bears much resemblance to passive, freely diffusing particles, our study reveals that velocity distributions in our system deviate from the Gaussian distribution typically observed for passive particles in a homogeneous environment. To illustrate that, we plot the probability density function of velocity distributions at a representative time (Figure 3.2(b) and (c)). Both velocity distributions along x and y have spiky modes and heavy tails, which cannot be captured by the best-fit Gaussian distribution (Figure 3.2, black dashed curves), for which the mean and standard deviation are specified as the sample mean and sample standard deviation, respectively. The shape of the probability density distribution suggests the use of the Laplace distribution (Figure 3.2(b) and (c), orange solid curves), which fits the velocity distributions reasonably well.

To test whether the velocity distribution follows the normal distribution at each time frame, we compute p-values of the Shapiro-Wilk test for normality [131], plotted in Figure 3.4. The p-values are all extremely small, indicating the velocity distribution at any time frame substantially deviates from Gaussian. Furthermore, we plot the sample quantile-theoretical quantile from a normal distribution (QQ-plot) of the velocities along both directions at 20 hours, both of which deviate from theoretical quantiles, shown as the black line in the inset. The p-values get smaller at later time frames when the system approaches

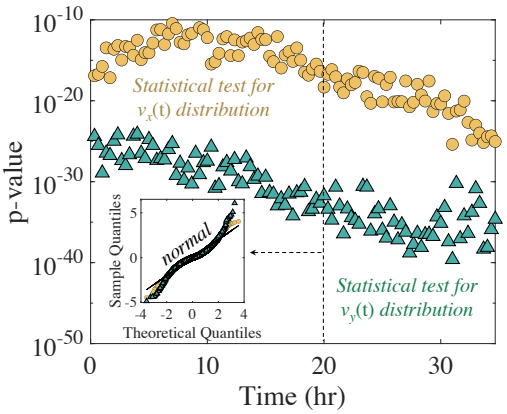

Figure 3.4: Shapiro-Wilk test for normality. The figure shows the p-value of individual frames, which is a statistical measurement applied to validate whether or not the observed data follows a normal distribution. A small p-value indicates statistical significance. The inset shows a representative quantile-quantile (QQ) plot at time = 20 hours to compare the sample quantiles of normalized velocity with the theoretical quantiles from the standard normal distribution. The black curve indicates the linear curve with slope 1 (normal).

jamming, as the network effects are larger. Furthermore, the p-values for velocities along y-direction are smaller than those from the x-direction, indicating that the deviation from normality is also bigger, signaling more confinement along y-direction.

3.2.4 Neighboring feature construction for cellular interaction

Next, we discuss the contribution of the neighbor ensemble to cell alignment. For any cell, the conventional choice of the neighboring set is to include all cells within a radial distance r (as shown in Figure 3.5(a)). Evidence of cell intercalation, where cells squeeze past their neighbors and they exchange positions, is increasingly found in recent studies [132, 133]. When antiparallelly aligned cells move past each other, they minimally impact each other's velocity. This motivates us to investigate a neighboring set that excludes the cells with opposite velocities, as shown in Figure 3.5(d). To compare these two approaches of accounting for the neighbors, in Figure 3.5(b), we plot the cellular velocity of each particle $v_{i,x}(t)$ at time = 20 hours versus the mean velocity of neighboring cells at the previous time frame at x coordinate, while in Figure 3.5(e) the same quantities are computed when the

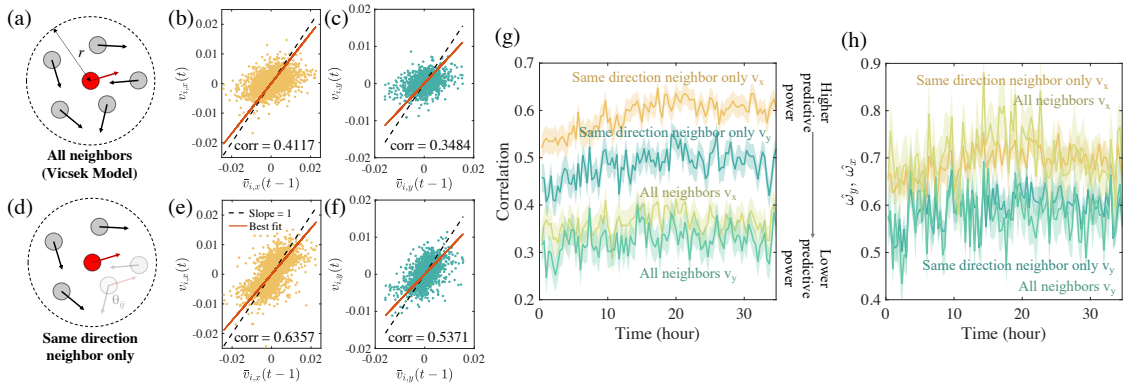

Figure 3.5: Effects of accounting for different neighbor ensembles. (a-c) Parameters and correlation plots using the neighbor ensemble of the classical Vicsek model for time = 20 hours. (d-f) The scenario where only cells traveling in the same direction are considered. [(b), (c), (e), and (f)] The correlation between the cell at t versus the mean of the velocity of its neighbors at $t-1$. (g) Correlations between velocity $v_{i,x}(t)$, $v_{i,y}(t)$ of any cell i and mean velocity of the neighboring cells from the previous step $\bar{v}_{i,x}(t-1)$, $\bar{v}_{i,y}(t-1)$, where the 95% confidence intervals, shown in fainted shades, are computed based on Fisher transformation of correlation coefficients. Two scenarios are shown, either by including all neighbors (bottom 2 curves), or only the same direction neighbors (top 2 curves). (h) The estimated weights, equivalent to the fitted slopes, $\hat{\omega}_x(t)$, $\hat{\omega}_y(t)$ from these two scenarios and their confidence intervals are plotted.

cells with opposite velocities are excluded from the neighboring set. As the neighboring set in the previous step includes the cell itself, it is not surprising that both approaches yield a relatively high 1-step forecast accuracy, since individual cells tend to migrate with a persistent velocity [134].

The correlation in Figure 3.5(e) is around 0.64, which is substantially higher than the correlation of 0.41 in Figure 3.5(b), indicating that excluding the particles from the neighbors substantially improves the 1-step predictive accuracy. The same conclusion holds for velocity updates along the y coordinate, shown in Figure 3.5(c) and (f). Furthermore, by evaluating the 1-step correlation over all time points (Figure 3.5(g)), we find that the method including only same-direction neighbors substantially outperforms the one including all of the neighbors, as in the Vicsek model. These results indicate that the HDF cells in our experiment appear to distinguish between head and tail polarities [135], despite extensive

analogy that has been drawn between weakly interacting fibroblasts and active nematics [112, 136],

3.2.5 Reduced magnitude of local alignment

Unlike [67], we do not normalize the velocity to keep the velocity magnitude to be the same, as the velocities can also change due to an increase in cell density. Figure 3.5(e) and (f) indicate that the velocity of the cell i at time t may be modeled by the mean of the velocity of the neighboring cells at time $t-1$, that is, $\mathbb{E}[v_{i,x}(t)] = w_x(t)\bar{v}_{i,x}(t-1)$ and $\mathbb{E}[v_{i,y}(t)] = w_y(t)\bar{v}_{i,y}(t-1)$, where $\bar{v}_{i,x}(t-1)$ and $\bar{v}_{i,y}(t-1)$ are the mean of the velocity of the neighboring cells at time $t-1$. To further test whether the statement is statistically significant, we compute the maximum likelihood estimator of the weights when the random fluctuation follows a Laplace distribution and generalized Gaussian distribution, with a different set of parameters at x and y coordinates.

As shown in Figure 3.5(h), the maximum likelihood estimators of weights $w_x(t)$ and $w_y(t)$ are both smaller than 1. Furthermore, the shaded area shows that the 95% confidence intervals of the estimation, which is calculated by the residual bootstrap estimate [137]. In all plots, we found the upper bounds of the 95% confidence intervals of $w_x(t)$ and $w_y(t)$ are smaller than 1 at any time frame, indicating that the velocity of a cell aligns with the velocity of neighboring cells, while the velocity alignment is offset by other forces, such as frictional force between substrates and cells [138, 2, 95, 139].

3.2.6 Summary of data-driven findings and model development

Here, we summarize the discovery from the EDA and statistical tests. First, variances of the random motion along the x and y coordinates are distinct, and both decrease over time due to cell proliferation. Second, the probability density of the velocity distribution

at both x and y directions has shown non-Gaussian behavior with a heavy tail and spiky mode near zero. Third, excluding cells with opposite velocities substantially improves the correlation between cellular velocity and mean velocity of the neighboring cells at the previous time step, as cells glide across each other in a manner reminiscent of intercalation [140]. Fourth, the slope of the coefficient of a linear fit between the particle density and mean of neighboring particles in the prior time frame is smaller than one, indicating velocity alignment with its neighboring particles is offset by cell-substrate forces. All of the statistical tests we have developed so far are not restricted to the context of externally guided alignment; they are generally applicable to video microscopy which contains rich spatiotemporal data. Next, we illustrate how these analyses can be utilized to develop simulation models and parameter estimation for these models.

3.3 Model constructed from selected features and maximum likelihood estimation

Our objective is to develop a minimum physical model that can account for temporally dependent orientational order parameters and velocities. We begin our model construction based on the seminal work by Vicsek [27], consisting of agents moving at a constant speed and updating their direction of movement at each step. The "new" velocity direction is determined by the summation of the average of the velocities of neighboring agents in the prior time frame, plus a random fluctuation. Despite its simplicity, this system exhibits a wide range of phenomena, including a transition from disordered to ordered behavior by adjusting the magnitude of the fluctuation. Later modification to this model includes incorporation of distance-dependent interactions [95], which effectively reproduces observed phenomena such as local cohesion [67] and jamming transitions [141]. These

phenomena are characteristic of a migrating epithelium [142], which has strong cell-cell junctions. Here we utilized a data-driven approach to make systematic improvements to the Vicsek model by incorporating the four selected features into the model. Though some of these aspects have been noted to some extent in literature, to our knowledge, there has been no work incorporating all of them and systematically testing their applicability for quantitatively reproducing the evolving orientational order observed in the experiment. The velocity vector of the i-th particle at time frame t, $\mathbf{v}_i(t) = (v_{i,x}(t), v_{i,y}(t))^T$, for $t = 1, ..., T$ and $i = 1, ..., n_t$, is modeled by two terms. The first and second terms represent the cell-cell, and cell-substrate interactions, respectively, as follows

$$v_{i,x}(t) = w_x(t)\bar{v}_{i,x}(t-1) + \epsilon_{i,x}(t), \tag{3.2}$$

$$v_{i,y}(t) = w_y(t)\bar{v}_{i,y}(t-1) + \epsilon_{i,y}(t), \tag{3.3}$$

where $\epsilon_{i,x}(t)$ and $\epsilon_{i,y}(t)$ are independent zero-mean random variables with respective variances $\mathrm{Var}[\epsilon_{i,x}(t)] = \tau_x^2(t)$ and $\mathrm{Var}[\epsilon_{i,y}(t)] = \tau_y^2(t)$, $\bar{v}_{i,x}(t-1)$ and $\bar{v}_{i,y}(t-1)$ are the mean of the x and y directional velocities of neighboring particles at time $t-1$, and $w_x(t)$ and $w_y(t)$ are real-valued scalars denoting the weights. The mean of the velocity at the x and y coordinates is modeled by

$$\bar{v}_{i,x}(t-1) = \frac{1}{p_{ne_i(t-1)}} \sum_{j \in ne_i(t-1)} v_{j,x}(t-1),$$

$$\bar{v}_{i,y}(t-1) = \frac{1}{p_{ne_i(t-1)}} \sum_{j \in ne_i(t-1)} v_{j,y}(t-1),$$

where $ne_i(t-1)$ is the neighboring set of cell i at time frame $t-1$ and $p_{ne_i(t-1)}$ is the number of cells in the neighboring set. Let $\mathbf{s}_i(t-1)$ be the 2D position of cell i at time frame $t-1$. The neighboring set is defined as $ne_i(t-1) = \{j : \|\mathbf{s}_j(t-1) - \mathbf{s}_i(t-1)\| <$

r and $\mathbf{v}_j(t-1) \cdot \mathbf{v}_i(t-1) > 0\}$, which contains particles within a radius distance r but excludes particles with opposite velocities.

Section 3.2.3 provides compelling evidence that the fluctuation distribution in the model significantly deviates from Gaussian distributions. Here, we introduce two distributions to model the random fluctuations in velocity: the Laplace distribution (or the double exponential distribution) and the generalized Gaussian distribution (or the stretched exponential distribution). Both distributions have been fit to the probability distribution of displacements that go beyond Gaussian assumptions, particularly when the system approaches jamming [143, 124, 118, 144]. Yet, finding the maximum likelihood estimator of the parameters with non-Gaussian fluctuations is not a computationally trivial task, given that the video contains 10^2 time frames and each frame has over 2000 cells. Hence, we introduce computationally-scalable approaches to compute the maximum likelihood estimator for models with these non-Gaussian fluctuations.

3.3.1 Maximum likelihood estimator with Laplace fluctuation

For the sake of simplicity, we will only demonstrate the model using velocity along the x direction as an example. The model for velocity along the y direction can be constructed in a similar manner. To model the fluctuation by the Laplace distribution $\epsilon_{i,x}(t) \sim \text{Laplace}(0, \tau_x(t))$, entails that the residual of velocity of particle i at time t $e_{i,x}(t) = v_{i,x}(t) - w_x(t)\bar{v}_{i,x}(t-1)$ follows

$$p(e_{i,x}(t) \mid \tau_x(t)) = \frac{1}{\sqrt{2}\tau_x(t)} \exp\left(-\frac{\sqrt{2}|e_{i,x}(t)|}{\tau_x(t)}\right).$$

The maximum likelihood estimator of $w_x(t)$ can be obtained by iterative algorithms such as the expectation-maximization algorithm and iterative reweighted least squared

algorithm [145, 146], while a faster alternative is through linear programming [147, 148]. The likelihood function and the maximum likelihood estimator of the weight parameter, $\hat{w}_x(t)$, are introduced in Appendix C.1. The maximum likelihood estimator of the standard deviation of the random fluctuation can be obtained by maximizing the profile likelihood after substituting in $\hat{w}_x(t)$:

$$\hat{\tau}_x(t) = \frac{\sqrt{2}}{n_t} \sum_{i=1}^{n_t} |v_{i,x}(t) - \hat{w}_x(t)\bar{v}_{i,x}(t-1)|. \tag{3.4}$$

3.3.2 Maximum likelihood estimator with generalized Gaussian distribution

In general, both the Laplace and the Gaussian distribution are special cases of the generalized Gaussian distribution (GGD), also referred to as a stretched exponential distribution [149, 150]. Here, we illustrate the formulation by applying the model to fit the velocity component along the x direction. We assume that the residual $e_{i,x}(t)$ follows a zero-mean GGD with parameters $\alpha_x(t)$ and $\beta_x(t)$ at time frame t:

$$p(e_{i,x}(t) \mid \alpha_x(t), \beta_x(t)) = \left(\frac{\beta_x(t)}{2\alpha_x(t)\Gamma\left(\frac{1}{\beta_x(t)}\right)} \right) \exp\left\{ - \left(\frac{|e_{i,x}(t)|}{\alpha_x(t)} \right)^{\beta_x(t)} \right\}. \tag{3.5}$$

The variance of the GGD follows $\tau_x^2(t) = \mathrm{Var}[e_{i,x}(t)] = \alpha_x^2(t)\Gamma(3/\beta_x(t))/\Gamma(1/\beta_x(t))$, with $\Gamma(\cdot)$ denoting the gamma function. When $\beta_x(t) = 2$ and $\alpha_x(t) = \sqrt{2}\tau_x(t)$, GGD reduces to a Gaussian distribution. When $\beta_x(t) = 1$ and $\alpha_x(t) = \tau_x(t)/\sqrt{2}$, GGD reduces to a Laplace distribution. Therefore, the shape parameter $\beta_x(t)$ controls how close the GGD resembles a Gaussian distribution. Similarly, the GGD of the residual along y can also be defined in terms of parameter $\alpha_y(t)$ and $\beta_y(t)$ at any given time frame t.

It should be noted that at each time frame, the GGD has three parameters, and

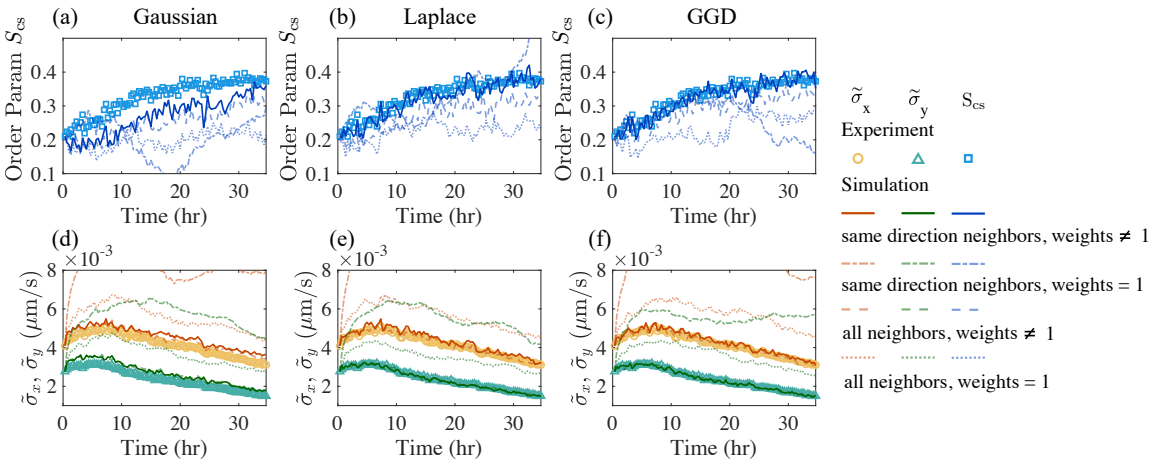

Figure 3.6: Reproducing orientational order parameters and mean absolute deviation of velocity with different simulation models at a representative radius $r = 75$ μm. The left, middle, right panels compare simulations using the Gaussian, Laplace, GGD fluctuation with experimental observations. For each type of fluctuation distributions, simulations of different interaction rules are presented. (a-c) show the orientational order parameter, (d-f) show the average absolute deviation of the velocities.

some of these parameters, such as the power parameter, are notoriously difficult to estimate [151]. Hence, we derive closed-form derivatives to numerically compute the maximum likelihood estimator of the parameters, introduced in Appendix C.2.

3.4 Results

Here we perform simulations to compare models with and without the selected features, and the simulation details are provided in Appendix C.3.

3.4.1 Model comparison by simulated orientational order parameter and velocity variability

We compare models using simulated order parameters and variability of velocity. The simulated order parameters is computed by $S_{\text{cs}}(t) = n_t^{-1} \sum_{i=1}^{n_t} \cos(2\tilde{\phi}_i(t))$ with $\tilde{\phi}_i(t)$ defined in Equation (3.1). As the velocity distribution more closely resembles a Laplace

distribution, we use the average absolute deviation or L_1 loss to measure the variability of velocity: $\tilde{\sigma}_x(t) = \frac{1}{n_t}\sum_{i=1}^{n_t} |v_{i,x}(t) - \bar{v}_x(t)|$ and $\tilde{\sigma}_y(t) = \frac{1}{n_t}\sum_{i=1}^{n_t} |v_{i,y}(t) - \bar{v}_y(t)|$ with $\bar{v}_x(t) \approx 0$ and $\bar{v}_y(t) \approx 0$, instead of the more commonly used variance or L_2 loss to quantify the variability of the velocity. This is because the velocity distribution is closer to a Laplace distribution than a Gaussian distribution (Figure 3.2(b) and (c)), and thus the L_1 loss is more appropriate to account for the variability of the distribution.

In Figure 3.6, we compare experimental observations and simulated results of the particle-based simulation, from which we computed the dynamics of the orientational order parameters and the average absolute deviation of the velocity. We apply the estimation and simulation procedure to scenarios where either all features are present or one or more of them are missing. Remarkably, we find simple models from Equations (3.2)-(3.3) with four features found in Section 3.2 can reproduce the dynamical progression of order parameters (solid curves in Figure 3.6(b) and (c)) and average absolute deviation of the velocity (Figure 3.6(e) and (f)), even though we do not fit a loss function for these ensemble properties directly. In Figure C.4 and Figure C.5 in Appendix C.4, we show the simulation using the estimated parameters from two other experiments and observe that our model is sufficiently general to capture the progression of system characteristics with varying initial densities and substrate materials.

In all simulation models, the magnitude and variability of the velocity along the x coordinate are both larger than those along the y coordinate, stemming from our first finding that the velocity distribution is anisotropic. The growing anisotropy of the velocity induced by the molecularly aligned substrate leads to the increase of orientational order parameters.

Second, the simulation with neighbor ensembles that include only cells traveling in the same direction (solid curves in Figure 3.6) more accurately reproduces both the

progression of the orientational order and velocity than the simulation with interactions that includes all cells within a radial distance. This finding, which is derived solely from tracking the nuclei, reflects what has been observed from the video microscopy: cells elongate, pulling on one another, as they migrate in the same direction. Conversely, when a pair of cells move pass each other in opposite direction, both cells maintain similar direction and velocities before the encounter. This effect likely occurs because of the complex interplay of cell-cell interactions, which are mediated through a cascade of signaling molecules. Consequently, the direction and polarity of contact are crucial factors in determining the strength of cell-cell interaction [135].

Third, the weight parameters $w_x(t)$ and $w_y(t)$ estimated by observations are always smaller than 1 (Figure 3.5(h)). In the simulation, we find that the model with estimated weights typically outperforms the model where the weights are constrained to be 1, particularly for reproducing the dynamical progression of the average absolute deviation of the velocity (Figure 3.6(a-c)). If the estimated slope parameters are equal to 1, the simulated velocity magnitude is frequently larger than what is observed. In comparison, the inclusion of the estimated weights appears to resolve this issue, as the velocity is constrained. This effect likely arises due to a loss of momentum to friction. Furthermore, a unique aspect of our formulation is that the variability of velocity can change over time, applying estimated weights enables us to capture a wider array of behaviors, such as slow-downs. For instance, with increasing cell density, the average magnitude of the velocity must change. Ultimately, crowding leads to arrested dynamics [80]. Nonetheless, simulation models often oversimplify this aspect by only modeling the velocity angle, overlooking the important role of slow-downs or heterogeneous dynamics [27, 68].

Lastly, we find that the simulation model validates the finding that velocities are not normally distributed (Figure 3.4)–the model with fluctuations following Laplace

distributions (Figure 3.6(b)) better reproduces the progression of the orientational order parameter than the model with Gaussian fluctuation (Figure 3.6(a)), even if they contain the same number of fitting parameters. Besides, the model with GGD fluctuation (Figure 3.6(c)) fits slightly better than the one with Laplace fluctuation, but it also contains one more parameter at each time point than both the Gaussian and Laplace distributions, and thus it has more flexibility in controlling the decay of the tail of the distribution. It is important to note that reproducing solely the variability of the velocity is inadequate to replicate the velocity orientational order parameter in our system, which is sensitive to the change in the velocity distribution.

To further explore the difference between the effect of different random fluctuation distributions, in Figure 3.7, we show the distribution of the simulated residuals in x direction in (a-c): $e_{i,x}^{sim} = v_{i,x}^{sim}(t) - \hat{w}_x(t)\bar{v}_{i,x}^{sim}(t-1)$ and in y direction in (d-f): $e_{i,y}^{sim} = v_{i,y}^{sim}(t) - \hat{w}_y(t)\bar{v}_{i,y}^{sim}(t-1)$ by the colored circles and triangles, and the solid curve denotes the distribution of the residuals in the simulations. The fitted weight is derived from the maximum likelihood estimator where the fluctuations follow Gaussian, Laplace, and GGD, from the left to the right. Simulated Gaussian fluctuation distributions along x and y directions are shown in Figure 3.7(a) and (d), which underestimates the number of cells with near-zero velocities in both directions. Instead, simulation fluctuation distributions with either Laplace distribution or GGD better reproduce the experiments, as shown in Figure 3.6(b), (c), (e) and (f).

Here we use a neighbor radius of $r = 75$ μm. Fit of orientational order parameters by models with different neighbor radii is plotted in Appendix C.5, and no significant difference amongst them has been observed. Thus, the radius seems to play a role in refining the fit, but to a much lesser degree than the choice of neighbor and fluctuation models. To further explore difference between models, we present a summary of the results of incorporating

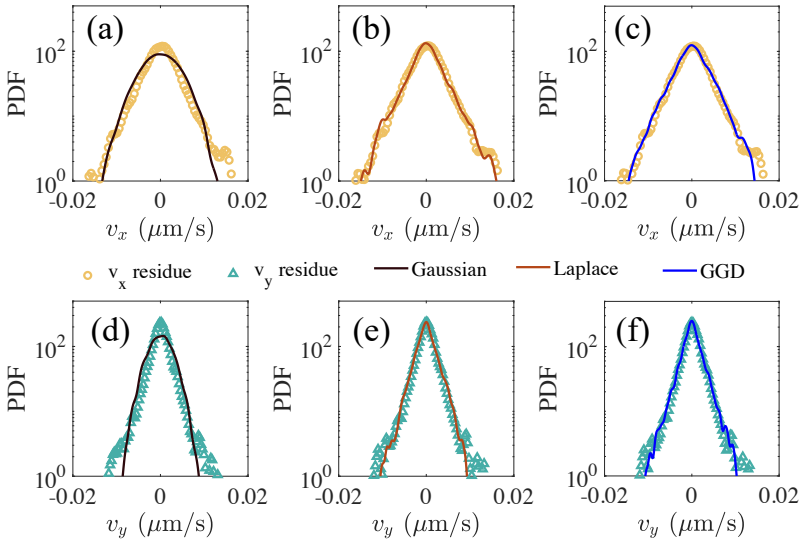

Figure 3.7: Distributions of random fluctuations between the observations and simulated models at a representative radius $r = 75$ μm and time = 20 hours with the distributions of the fluctuation in the simulation following Gaussian (a,d), Laplace (b,e) and GGD (c,f).

various modeling parameters and compare the simulation to the experimentally obtained orientational order parameter in Table 3.1. The root mean squared error (RMSE) of the velocity orientational order parameter is calculated as the following:

$$\text{RMSE}_S = \sqrt{\sum_{t=1}^{T} \frac{(S^*_{cs}(t) - S_{cs}(t))^2}{T}},$$

where $S^*_{cs}(t)$ and $S_{cs}(t)$ are the ensemble orientational order parameters from the simulation and experiments at time frame t, respectively. The RMSE for $\tilde{\sigma}_x$ and $\tilde{\sigma}_y$ can be defined similarly, and they are shown in Table 3.2. RMSE values that are better than the baseline average absolute deviation are bolded. Values from Table 3.1 and 3.2 indicate that, while there are several effective methods to describe the average absolute deviation of velocity, it is more difficult to capture the progression of the order parameter. This is not surprising as the order parameter is a complex function of the distribution, which is not explicitly controlled by any parameters in the likelihood function, given $v_{i,x}$ and $v_{i,y}$ are modeled independently. When modeling the velocity fluctuation, approaches incorporating Laplace

Table 3.1: RMSE_S for the orientational order parameter estimates. All assumed the weights parameters from the fluctuation from the x and y directions are separately estimated. The standard deviation of the velocity orientational order parameter is 0.050 and an effective method (highlighted in bold) has RMSE_S smaller than this value. AN (= all neighbors) and SDN (= same direction neighbor) stand for methods with all neighboring cells within the radius and the same direction neighboring cells, respectively. Gau, Lap, and GGD are Gaussian, Laplace, and generalized Gaussian distributions of the fluctuation.

Orientational order	$\omega = 1$			$\omega \neq 1$		
parameter RMSE_S	Gau	Lap	GGD	Gau	Lap	GGD
r = 25 μm AN	0.17	0.12	0.14	0.090	0.061	0.058
r = 25 μm SDN	0.12	0.16	0.14	0.077	**0.034**	**0.038**
r = 50 μm AN	0.11	0.13	0.12	0.079	**0.047**	**0.043**
r = 50 μm SDN	**0.041**	0.12	**0.033**	0.069	**0.022**	**0.025**
r = 75 μm AN	0.13	0.11	0.10	0.081	0.058	**0.042**
r = 75 μm SDN	0.15	0.060	0.099	0.063	**0.020**	**0.018**

and GGD fluctuations better capture the progression of the orientational order parameter, which is the main characteristic of the system, compared to the model with the Gaussian fluctuation.

3.4.2 Sensitivity analysis of the simulation

As captured by Equations (3.2) and (3.3), the anisotropy of substrate materials induces asymmetric velocities in cells, which are manifested through both interactions and fluctuations. These contributions are temporally dependent, as changes in cell density due to proliferation lead to fluctuations in the velocity magnitude. For a given set of model parameters and fluctuation distribution, it will be helpful to understand their impacts on the asymptotic order parameters, such as those that can be achieved after enough time has elapsed, for refining experimental designs and controls.

Here, we perform a sensitivity analysis for simulation with Laplace fluctuations and all other selected features. We vary the ratio of the variance of the fluctuation τ_x/τ_y and the ratio of their weights ω_x/ω_y in the model (Equations (3.2) and (3.3)). While the variation of the order parameter is controlled by four parameters (τ_x, τ_y, ω_x, and ω_y), we find that the

Table 3.2: RMSE for the average absolute deviation of the velocity estimates $\tilde{\sigma}_x$ and $\tilde{\sigma}_y$. All assumed the weights parameters from the fluctuation from the x and y directions are separately estimated. The benchmark RMSE of the average absolute deviation (calculated by the standard deviation of the average absolute deviation) for $\tilde{\sigma}_x$, $\tilde{\sigma}_y$ are 5.7×10^{-4} and 5.3×10^{-4}, respectively. An effective method (highlighted in bold) has RMSE smaller than this value. AN (= all neighbors) and SDN (= same direction neighbor) stand for methods with all neighboring cells within the radius and the same direction neighboring cells, respectively. RMSE = (table value) $\times10^{-4}$.

	$\omega = 1$					
Average absolute deviation of velocity estimates	Gaussian		Laplace		GGD	
	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$
r = 25 μm AN	38	35	38	30	39	32
r = 25 μm SDN	137	68	140	59	130	63
r = 50 μm AN	24	20	20	17	25	20
r = 50 μm SDN	74	40	88	36	77	37
r = 75 μm AN	14	14	14	12	16	13
r = 75 μm SDN	42	34	52	32	49	32
	$\omega \neq 1$					
Average absolute deviation of velocity estimates	Gaussian		Laplace		GGD	
	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$	$\tilde{\sigma}_x$	$\tilde{\sigma}_y$
r = 25 μm AN	**1.1**	**2.7**	**1.4**	**0.56**	**1.3**	**0.67**
r = 25 μm SDN	**3.0**	**3.5**	**2.9**	**1.1**	**2.6**	**1.4**
r = 50 μm AN	**2.4**	**3.1**	**0.83**	**0.52**	**0.89**	**0.54**
r = 50 μm SDN	**3.4**	**3.4**	**2.4**	**0.67**	**1.7**	**0.83**
r = 75 μm AN	**3.0**	**3.6**	**0.87**	**0.47**	**0.72**	**0.53**
r = 75 μm SDN	**3.9**	**3.4**	**2.1**	**0.56**	**1.5**	**0.51**

estimated ω_x (Figure 3.5(h)) does not change drastically over time. Furthermore, we show in Appendix C.2 that simulations with two different choices of τ_x do not have a large impact on the variation of the order parameter so long as the ratios of ω's and τ's remain the same. Hence, we are able to represent the asymptotic value of the order parameter on a 2D plot with $\tau_x = 0.02$ and $\omega_x = 0.7$ fixed to be the mean of estimation over all time frames. Each simulation is implemented in 70 time points, where the order parameter fully equilibrates after 20 time points (see the evolution of the order parameter for select simulation parameter in Appendix C.2). Due to this, the order parameter in Figure 3.8 was computed by averaging the last 50 time points.

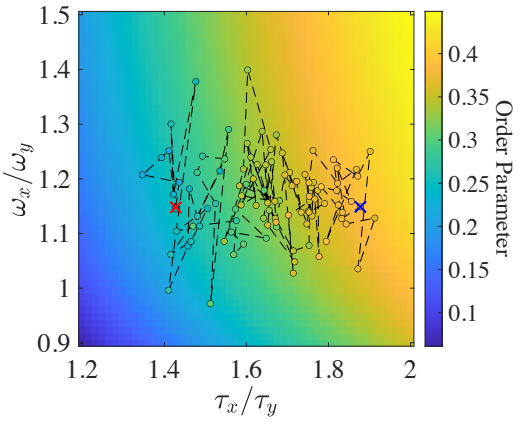

Figure 3.8: Change of order parameters due to the change of simulation parameters with $\tau_x = 0.02$ and $w_x = 0.7$. The color bar corresponds to the simulated order parameter at long times. The dashed lines and circles denote the actual path of the experimental results plotted against the phase diagram. The colors of the markers correspond to the order parameter as denoted by the color bar. Red and blue X's denote the beginning and the end of the trajectory, respectively.

We find that the asymptotic order parameter increases when either τ_x/τ_y or ω_x/ω_y increases (Figure 3.8). The order parameters computed from the experiment at different time points are overlaid on top of the diagram, where the warmer colors represent a larger magnitude of the order parameter, consistent with the color scheme of the phase diagram. The ratio ω_x/ω_y fluctuates around a fixed value ~ 1.2, potentially due to fixed properties of the external substrate guiding cells to migrate preferentially along x. That is to say, the substrate has a fixed modulus ratio along the x and y directions, as found in [74]. This ratio is reflected in the variance of the velocity along the corresponding directions, thereby restricting their ratio. As the system evolves over time, the caging effect becomes more prominent along y, which increases the ratio τ_x/τ_y. This analysis provides further evidence to support our conclusion that substrate anisotropy and cell crowding both influence alignment. However, at low cell densities, the alignment effect is likely drowned out by noise. Our findings also open up exciting design opportunities, where τ_x/τ_y can potentially be controlled by varying the initial cell density or the composition of the substrate [93], so the

cell alignment dynamics can be tuned as a result. Overall, good agreements between the experiment and simulation are obtained.

Chapter 4

Universal phase identification of block copolymers from physics-informed machine learning

This chapter addresses a classification problem in materials characterization: predicting the phase morphology of block copolymers from small-angle X-ray scattering (SAXS) intensity curves. Accurate phase identification is crucial for understanding structure-property relationships, but the high noise levels in experimental SAXS data challenge the direct application of standard classification methods. We develop a physics-informed machine learning approach that first denoises the raw SAXS intensity curves using the Kalman filter and then extracts features that are independent of chemical details but reflect the microstructure, which are then used to build the classification model. This physics-informed feature selection effectively reduces dimensionality, which enables the model to achieve high accuracy in predicting phases of both novel and existing materials in the training dataset. Furthermore, by leveraging the inherent uncertainty of the model, overall accuracy can be

further improved by manually reviewing only a small fraction of samples with high predictive uncertainty.

4.1 Introduction

Block copolymers, which consist of two or more chemically distinct segments (blocks), are an important class of materials known for undergoing self-assembly into well-defined nanostructures, underpinning their use in applications including drug delivery, high-performance materials, and advanced electronics [152, 153]. Self-assembly into a variety of nanostructures, including body-centered cubic spheres (BCC), hexagonally-packed cylinders (HEX), double gyroid networks (GYR), and lamellae (LAM) can be precisely tuned by various parameters including block chemistry, volume fraction (f), molecular weight (M_{n}), and architecture [154]. More recently, Frank–Kasper phases, hexagonally close-packed spheres, hierarchical X-in-Y structures, and complex network phases have been discovered in block copolymers, highlighting the ever-expanding palette of potential morphologies and properties achievable in this class of soft materials [155, 156, 157, 158, 159].

Traditionally, studying the phase behavior of block copolymers is laborious, involving iterative synthesis across many different compositions and molecular weights, coupled with characterizing each distinct material using tools such as small-angle X-ray scattering (SAXS), a technique that measures the scattering intensity of X-rays as a function of wave vector magnitude q to produce a 1D intensity profile. The complexity of this approach is underscored by the need for rigorous peak indexing of each SAXS pattern to accurately determine a given structure, which is complicated by issues including poor long-range order, missing reflections (i.e., form-factor minima that suppress allowed reflections), limited peak resolution, sample purity, coexisting phases, and extraneous atmospheric scattering

[160, 161, 162, 163, 164]. At best, analyzing SAXS datasets is time-consuming and slow; at worst, the aforementioned issues cause challenges that only an expert-level understanding of X-ray theory can help resolve. The considerable complexity, time, and effort required to accurately identify nanostructures formed by block copolymers—and other materials—is compounded by an expansive parameter space, highlighting the potential utility of a workflow that accelerates the study of new materials, ideally in a fashion that is accessible to researchers from many different backgrounds.

Thinking about solutions to such experimental bottlenecks can find inspiration from recent advances in computation and theory. In many ways, similar issues are encountered with the *de-facto* tool for simulating block copolymer phase behavior, such as self-consistent field theory (SCFT) and coarse-grained model [165, 166, 167, 168, 169, 170, 171]. Yet the SCFT and other simulation approaches rely on idealized parameters and assumptions that may not capture the distinct phase behaviors of block copolymers with different monomers. In addressing these challenges, recent advances in machine learning have created new opportunities to automate structural analysis by detecting patterns in datasets with high dimensionality [172, 173, 174, 175]. For example, Jayaraman and coworkers developed a high-throughput machine learning enhanced computational method which analyzes SAXS patterns to reconstruct real-space 3D structures of materials [176, 177, 178]. Pozzo and colleagues introduced an algorithm which analyzes the shape of SAXS profiles to automatically generate phase maps [179]. Furthermore, Olsen and colleagues leveraged a random forest model to analyze the phase behavior of diblock copolymers using experimental data mined from literature [1]. While Olsen's method is effective in predicting phases of existing polymers in the database, this method, which we refer to as a chemistry-dependent random forest (CD-RF), relies on chemical-specific features, such as monomer identities, volume fraction, molecular weight, and temperature, which limits the predictive accuracy of phases formed

by new monomers that are not in the database.

Here, we build on these advances and report a machine learning method that rapidly, automatically, and accurately determines the morphology of block copolymers SAXS data to enable real-time data analysis. Unlike conventional machine-learning methods for analyzing SAXS data that process the full intensity curve without explicit noise filtering, our approach is informed by X-ray scattering theory and automatically extracts pivotal physics-informed morphological features from the reduced 1D intensity scattering pattern to construct a universal classification model and enable real-time data analysis for researchers. Statistically, our method introduces two major innovations. First, by modeling each intensity curve with a Gaussian process [9, 45], we incorporate experimental uncertainties across different wave vector magnitudes, q. Second, by integrating the physical principles of X-ray diffraction, we enhanced the effectiveness and accuracy of our predictive models. X-ray diffraction measures the power spectrum of the Fourier transform of the sample's electron density. Thus, details of a sample's phase and nanostructure manifest most prominently as diffraction peaks in the X-ray data profile; where the q-location of the strongest peak, peak width, and peak locations of all peaks with respect to the primary peak are the most important features with direct physical significance. With this knowledge, full-intensity curves were instead transformed into a small set of informative features, dramatically reducing the required size of the training set.

The combination of advanced statistical modeling and informed feature reduction significantly enhances the robustness and reliability of our predictive models for accelerating materials discovery. Our machine learning process is structured into three stages: filtering the noise of intensity curves with a Gaussian process, detecting peaks and extracting crucial curve features, and applying a classification model, such as random forests [180], bagging [181], and gradient boosting [182] for material phase prediction. Importantly, we couple this

new physics-informed machine-learning algorithm with a powerful experimental technique recently developed by our groups—automated chromatography—that yields a large set of well-defined and purified block copolymers from a very small number of as-synthesized samples [183, 184, 162, 185, 157, 186, 158]. A notable advantage of this separation process over traditional iterative synthesis is its ability to remove homopolymer impurities and reduce the dispersity of each fraction compared to the as-synthesized parent block copolymer, enhancing the purity and reproducibility of fractionated samples' morphologies. Together, this combined workflow minimizes the synthetic burden of creating comprehensive and systematic sets of training data that were used to evaluate the accuracy and predictive capabilities of the machine-learning framework. Notably, this method achieves approximately 95% predictive accuracy across a variety of block copolymer chemistries, molecular weights, domain spacings, and nanostructures, including both new and existing materials in the training dataset, which is much higher than other alternative approaches. Furthermore, this method can further identify and correct data that was initially mislabeled as a result of human error based on quantified uncertainty of the prediction. After scrutinizing about 15% of the data with the highest uncertainty in predictive labels, the accuracy of our method approaches nearly 100%, marking a crucial advancement towards automated, high-throughput laboratories integrated with SAXS systems [187, 188]. The data and code used in this chapter are publicly available (`https://github.com/UncertaintyQuantification/automated_polymer_phase_identification`).

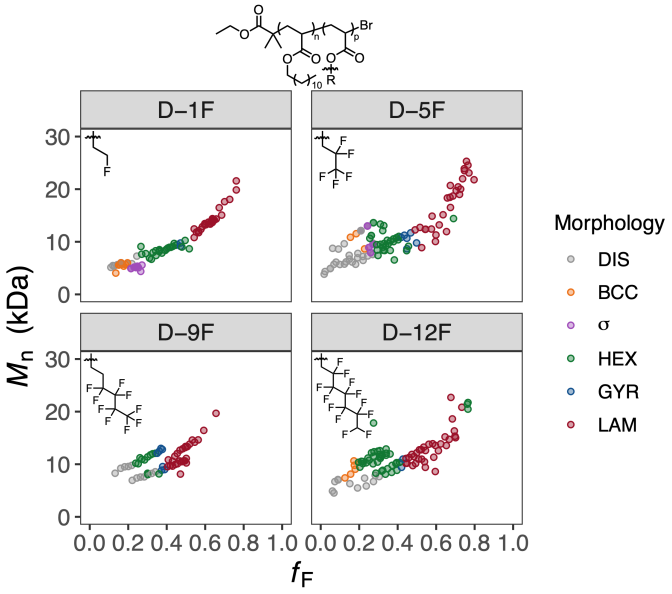

Figure 4.1: A library of 364 well-ordered diblock copolymers derived from the synthesis and separation of only 16 parent copolymer samples was used to evaluate the machine-learning algorithms developed herein. Overall block copolymer molecular weight is denoted by M_n. Volume fraction of the semi-fluorinated block is denoted by f_F. Color indicates the morphology as determined by manual analysis of SAXS data.

4.2 Results and discussion

4.2.1 Generating block copolymer libraries

To evaluate the efficacy of our new machine-learning algorithm as applied to analyzing small-angle X-ray scattering (SAXS) data, we leveraged a large experimental dataset (364 SAXS patterns) spanning four different block copolymer chemistries with systematically varying molecular weights and volume fractions as derived from automated chromatography. Note that we recently reported the phase behavior of these materials, which was manually determined by painstaking analysis of each individual SAXS pattern; these experimental phase portraits are reproduced in Figure 4.1 [162]. Each material is a diblock copolymer with a poly(dodecyl acrylate) block connected to one of four semi-fluorinated acrylates we denote the volume fraction of the semi-fluorinated block in each case by f_F. Four classes of AB diblock copolymers with increasing degrees of fluorination were synthesized via sequential photo-

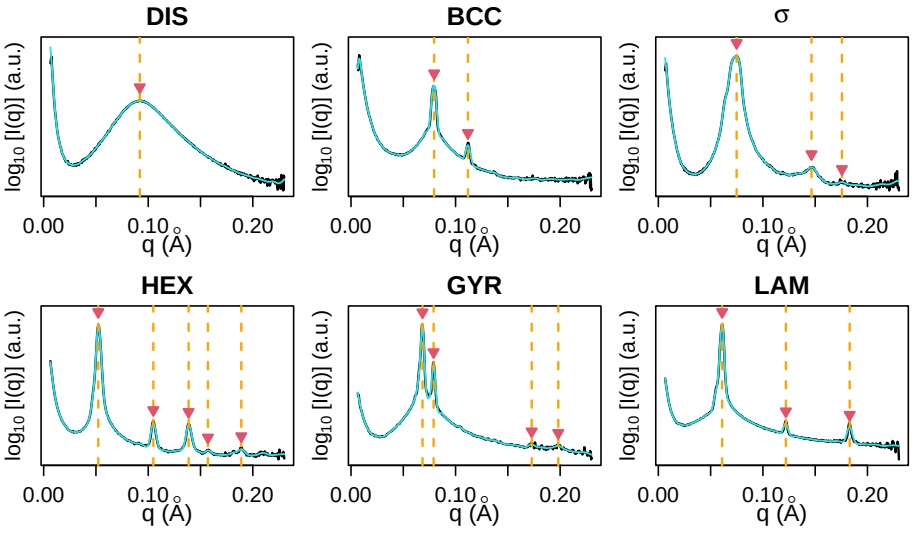

Figure 4.2: Panels display a representative example of each morphology. Black and blue curves represent the original log intensity and the smoothed intensity curves, respectively. Detected peak locations are highlighted by red triangles and marked with orange dashed lines.

initiated atom transfer radical polymerization: poly(dodecyl acrylate)-*b*-poly(2-fluoroethyl acrylate) (D-1F), poly(dodecyl acrylate)-*b*-poly(2,2,3,3,3-pentafluoropropyl acrylate) (D-5F), poly(dodecyl acrylate)-*b*-poly(1H,1H,2H,2H-nonafluorohexyl acrylate) (D-9F), poly(dodecyl acrylate)-*b*-poly(2,2,3,3,4,4,5,5,6,6,7,7-dodecafluoroheptyl acrylate) (D-12F). Readers interested in the experimental details of the synthesis and chromatographic separation are referred to our previous publication and the supporting information [162].

4.2.2 Filtering and feature extraction

Automated chromatography yielded an extensive library of materials with high-quality self-assembly across a wide range of volume fractions ($f_{\mathrm{F}} = 0.02$ – 0.80). Representative scattering patterns from six distinct morphologies (disordered (DIS), BCC, σ, HEX, GYR, and LAM) are plotted in Figure 4.2, where the blue curve represents the scattering intensities after noise filtration. The filtering is performed via a twice differentiable Gaussian process with a Matérn kernel [110], implemented through the Kalman filter framework

described in Chapter 1, which provides computational efficiency for the smoothing along the one-dimensional q coordinate. We have innovated an approach that integrates experimental uncertainties in filtering and denoising with minimal computational demand, which is crucial for processing experimental data. In X-ray diffraction, certain peaks may be absent or suppressed due to factors like poor resolution or limited long-range order. For example, the σ pattern typically requires synchrotron-level capabilities to resolve its intricate diffraction pattern often comprised of circa 48 distinct peaks, whereas using traditional bench-top SAXS instruments would impart significant peak broadening despite analyzing the same nanostructure [155]. Excessive smoothing can obscure these critical small or overlapping peaks, whereas insufficient smoothing might misinterpret noise as peaks. The smoothing level in Gaussian process regression is controlled by the range and nugget parameter of the covariance [43], which are determined using the maximum likelihood estimate from all DIS samples in our dataset. Comprehensive details on the formulae for smoothing and peak detection are provided in the Appendix D.1.

Figure 4.3(a) shows a t-distributed stochastic neighbor embedding (t-SNE) plot of logarithmic SAXS intensities curves [189], where each point represents an individual curve, color-coded by morphology. Utilizing the entire 1D intensity scattering curves leads to poor separation in t-SNE visualization, likely due to subtle differences in overall scattering patterns across various morphologies as well as baseline noise interference. To address these challenges, we employ X-ray scattering theory to construct physics-informed morphological features (PIMF), which are automatically extracted and input into the machine learning model rather than using the full scattering curve. Specifically, the analysis of the location of the first three peaks proved highly diagnostic of distinct morphologies. Figures 4.3(b) and 4.3(c) respectively show the first primary scattering peak location against the ratio between the first and second peaks and the ratio between the second and third peaks. These plots

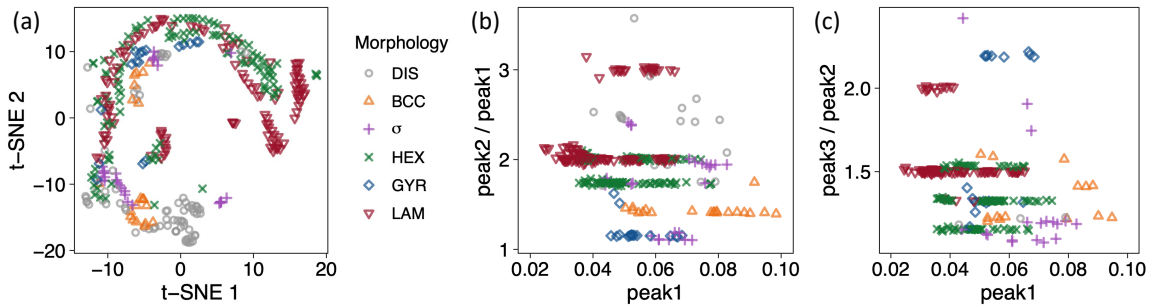

Figure 4.3: (a) T-distributed stochastic neighbor embedding (t-SNE) for visualization of original log intensity. (b) First peak location vs. the ratio between second and first peak. (c) First peak location vs. the ratio between third and second peak.

illustrate that physics-informed features, such as ratios between peak locations, facilitate a more effective structural determination than traditional methods which analyze the entire scattering curve. These refined features—the location of the first primary scattering peak and pairwise ratios of the first three peak locations—hold particular significance for researchers seeking to analyze novel materials that may lack extensive long-range order. We limit our analysis to the first three peaks here, as using fewer yields insufficient information and extending beyond often proves impractical, given that over 30% of samples lack a fourth peak, leading to less accurate classification.

In addition to peak locations, the width and sharpness of the peaks are also critical for identifying morphologies in block copolymers. Typically, disordered samples exhibit a single broad and low-intensity peak, whereas ordered block copolymers often display multiple well-resolved and sharp reflections. However, there are instances, such as in HEX, BCC, and σ phases, where a single peak might emerge due to overlapping peaks or being close to the order-disorder boundary. To address these challenges, we include in our model the width of the first primary scattering peak and its sharpness, measured by the second derivative of the peak.

Our approach effectively reduced the original intensity curves to 6 salient features:

the pairwise ratio of the first three peak locations, and the location, width, and sharpness of the first primary scattering peak. These physics-informed features substantially simplify the high-dimensional scattering intensity patterns to enable facile structure determination. Importantly, these features are independent of the specific chemical composition of the material, underscoring the versatility of this approach as a universal algorithm for phase identification in a wide range of block copolymers. Including more morphologically relevant features can improve predictive accuracy in similar tasks, whereas adding a large number of less informative features may also degrade the predictive accuracy of the model. Significantly, this method's ability to extract essential morphological information from scattering patterns enables its broad applicability across a diverse chemical landscape extending to various block chemistries, molecular weights, dispersities, number of blocks, and polymer architectures. Notably, well-ordered materials of a specific morphology exhibit identical series of scattering reflections, regardless of polymer identity. The q values of these reflections may shift based on polymer molecular weight, but the overall pattern remains consistent. By removing the impact of specific chemical characteristics, our model offers a powerful tool for analyzing diverse block copolymer systems efficiently.

4.2.3 Phase identification of new block copolymer chemistries

To demonstrate the effectiveness of our method with PIMF in rapidly analyzing novel block copolymers, we first trained our model using manually identified SAXS patterns from three block copolymer chemistries to predict the morphologies of a fourth material group. Due to the increased conformational asymmetry of D-1F and D-5F block copolymers, these materials exhibit a window of σ stability that vanishes in more conformationally symmetric D-9F and D-12F materials [162]. Holding out these two groups as the training data could lead to inaccurate predictions due to the limited number of σ samples. Thus, we

Table 4.1: Prediction results of random forest models on predicting D-9F block copolymer morphologies using PIMF-RF, CD-RF, and Curve-RF.

PIMF-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	10		1				1
	HEX				12			0
	GYR					7		0
	LAM				1		26	1
CD-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	4	2		5			7
	HEX				12			0
	GYR				7			7
	LAM				4	4	19	8
Curve-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	10	1					1
	HEX	8		2	2			10
	GYR			2	3		2	7
	LAM	11			9		7	20

focused on two testing scenarios: using D-1F, D-5F, and D-12F to predict the morphology of D-9F, and using the other three block copolymer libraries to predict D-12F.

We present the results of out-of-sample prediction by the physics-informed morphological features in the random forest (PIMF-RF) model [180] and compare the accuracy with the CD-RF model that relies on volume fraction, total molar mass, temperature, and monomer identity of the diblock copolymers [1] and the Curve-RF model that directly uses the original SAXS curve. We also employ bagging and the gradient boosting model with our feature set, which provides similar results detailed in Appendix D.2.

Table 4.1 presents the classification performance of different approaches when predicting the phase behavior of group D-9F. Notably, our approach only has 2 misclassifications out of 57 samples, significantly outperforming the 22 and 38 misclassifications observed in the CD-RF and Curve-RF methods, respectively. Compared with using the entire SAXS curve as the input in the Curve-RF method, the six physically-formed features in PIMF-RF

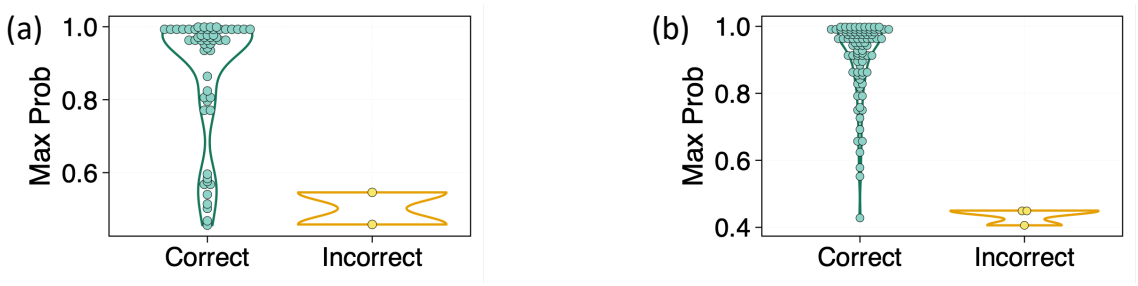

Figure 4.4: Maximum predicted probability for correctly and incorrectly predicted morphologies for (a) the D-9F block copolymer library; (b) the D-12F block copolymer library using PIMF-RF.

are informative in reducing the high dimensionality of the SAXS curve and they lead to much higher accuracy in predictions. Furthermore, the CD-RF approach does not rely on the use of SAXS data and it is shown to be accurate for predicting phases with monomers that appear in the training data set [1]. The results indicate that the automated analysis of SAXS data in PIMF-RF effectively enhances the accuracy of predicting the phase of the novel monomers not in the training data set.

The RF model classifies phases based on the predicted probability for each phase, where the phase with the highest probability is selected as the predicted outcome. The low maximum predicted probability indicates high uncertainty of the method, which can be used to control the predictive error [10]. Figure 4.4(a) represents the maximum predicted probabilities for all test samples, overlaid on violin plots that illustrate the distribution of these probabilities. [190] The plot reveals that the 2 misclassified samples in group D-9F have probabilities below 0.55, indicating low prediction confidence. Conversely, correctly classified samples generally display maximum probabilities exceeding 0.7, demonstrating a higher confidence level. This pattern suggests that inspecting a small subset of test samples with low maximum predicted probabilities could enhance the accuracy close to 100%, which will substantially reduce the manual review workload compared to examining all samples.

Table 4.2: Results of random forest models on predicting D-12F block copolymer morphologies using PIMF-RF, CD-RF, and Curve-RF.

PIMF-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	12						0
	BCC		5					0
	HEX		3		36			3
	GYR					2		0
	LAM						33	0
CD-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	8	1		3			4
	BCC	4	1					4
	HEX	4	1		29	1	4	10
	GYR				1		1	2
	LAM					1	32	1
Curve-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	9			2		1	3
	BCC				3	2		5
	HEX	8		1	6	5	19	33
	GYR	1		1				2
	LAM	6			22		5	28

Subsequently, we applied the same PIMF-RF approach to predict the morphologies of the D-12F block copolymer library with the classification results detailed in Table 4.2. Of the 91 D-12F block copolymers, only three are inaccurately predicted, an improvement over the CD-RF [1] and Curve-RF approaches, which misclassify 21 and 71 morphologies, respectively. Figure 4.4(b) shows that the maximum predicted probabilities of 3 misclassified samples in the D-12F library are all below 0.5, reinforcing the earlier observation that reviewing samples with low predicted probabilities can reduce prediction errors of our approach. Significantly, our PIMF-RF model accurately identifies the morphology of 142 out of 147 block copolymers (96.6% accuracy) consisting of novel monomers not present in the training dataset, highlighting the power of this synergistic approach combining machine learning with automated chromatographic separation towards accelerated materials discovery.

4.2.4 Phase identification of mixed block copolymer chemistries

To further evaluate the robustness of our high-throughput analysis method under diverse training conditions, we combined all four classes of diblock copolymers and implemented a 5-fold cross-validation strategy. The dataset was randomly divided into five folds, with each iteration using four-folds for training and one-fold for testing. Employing the PIMF-RF approach results in 25 misclassified morphologies out of 364. Upon careful review, we found three block copolymers initially mislabeled during manual morphology assignment were identified and correctly predicted by our approach. These three SAXS patterns are shown in Appendix D.2. Figure 4.5(a) displays the maximum predicted probabilities for all morphologies, with mislabeled materials highlighted in red. We retrained our model of the revised dataset after correcting these labels, and the number of misclassified morphologies using our PIMF-RF method reduced to 20, which is much smaller than the 62 misclassifications observed with the CD-RF approach [1] and 85 misclassified samples in Curve-RF approach using the same training data. Detailed results are provided in Appendix D.2.

Figure 4.5(b) presents the violin plot of the maximum predicted probabilities of all held-out test samples by the PIMF-RF method. Analysis shows that when using a threshold of 0.6 for the predicted probabilities, we need to examine only 32 SAXS patterns (9% of the data) to achieve a predictive accuracy of 98.4%. Increasing this threshold to 0.8 requires inspecting 69 samples (19% of the data), which results in remarkably accurate predictions with 100% accuracy. This pattern suggests that strategically examining a small subset of SAXS patterns with lower predicted probabilities enables the ML approach to attain high predictive accuracy.

Figure 4.5: Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using the physics-informed features for (a) the original samples; (b) revised samples after correcting three mislabeled detected by our method.

Chapter 5

Reliable emulation of complex functionals by active learning with error control

In many scientific problems, physical simulations are computationally expensive, and the input space is often high-dimensional or even infinite-dimensional. Traditional "space-filling" designs, including random sampling and Latin hypercube sampling, become inefficient as the dimensionality of the input variables increases, and the predictive accuracy of the emulator can degrade substantially for test inputs far from the training input set. This chapter develops an active learning with error control framework, which guides adaptive sampling by quantifying prediction reliability based on the predictive distribution of Gaussian processes. This algorithm is applicable to infinite-dimensional mappings and achieves high-fidelity predictions with controlled prediction error. By applying it to classical density functional theory, a statistical-mechanical framework for modeling the equilibrium properties of molecular systems, we demonstrate the algorithm's high computational efficiency and

improved performance compared to conventional "space-filling" designs and alternative active learning methods. The material in this chapter is reproduced from Xinyi Fang, Mengyang Gu, and Jianzhong Wu, "Reliable emulation of complex functionals by active learning with error control," *J. Chem. Phys.* **157**, 214109 (2022), with the permission of AIP Publishing.

5.1 Introduction

Theoretical research in chemistry and materials science is often hampered by computational bottlenecks in applying complex physical models to predict the microscopic structure and physicochemical properties of many-body systems. Statistical and machine learning (ML) methods, such as Gaussian process (GP) regression, basis expansion methods, and neural network models, have been widely used as a surrogate of complex physical models to emulate the atomic force fields, density functionals, chemical reactions, and diverse properties of materials and chemical systems [191, 192, 193, 194, 195, 196, 197, 198]. Here a typical aim of a ML approach is to learn the map with a high dimensional input or a functional input from observations. Trained on a set of pre-specified inputs, a statistical emulator can obtain accurate predictions when a test point is close to the training inputs [199, 200, 201].

Constructing a statistical emulator often starts with selecting a set of "space-filling" samples in the input space, such as uniform samples or Latin hypercube samples (LHS) [32]. The statistical emulator will be trained based on the outputs of the simulation at these pre-selected inputs. The idea is to ensure that for each test point in the input space, there are a sufficiently large number of inputs near this test point, where the outcomes (such as the potential energy or atomic force in emulating the first-principles calculations) were observed. Using the space-filling samples and assuming a set of regularity conditions, the

predictive error of some ML methods, such as the GP regression, is guaranteed to converge to zero, when the sample size goes to infinity [202]. The space-filling samples have two main drawbacks in emulating physics-based calculations such as molecular dynamics simulation (MD). First, the input values, such as atomic positions or pairwise inter-atomic distances in MD, may not be directly controlled. Second, physics-based modeling of molecules and materials often involves a high or infinite dimensional input space, such as the external potential function and atomic configurations. If the statistical emulator is trained in a small parameter subspace, the predictive accuracy of the emulator can be dramatically degraded outside the subspace where no training input is available. This scenario often occurs when the outcome is required in a slightly different system or at another experimental condition, such as different temperatures or pressures, whereas the statistical emulator becomes inaccurate if it is only trained on one set of conditions. Obtaining accurate predictions of any test input with a controlled predictive error is important to bypass expensive computations based on complex physical models.

To address the fundamental difficulty in predicting functions and functionals with a high-dimensional input space, active learning has become one of the most promising techniques to sequentially reinforce the emulation of physics-based modeling [39]. The active learning approach can be implemented in two broad scenarios. The first one is an online scenario or "on-the-fly" prediction, where the test inputs sequentially come and one does not know what future input needs to be predicted [40, 203, 204]. The online prediction scenario has many applications, such as Bayesian optimization [15, 16] and model calibration [8, 205], where the outcome of the simulation needs to be sequentially predicted for every sampled input. The second scenario is to sequentially select the samples to run a computer simulation for predicting the entire input space, where the uncertainty of the emulator is often used for selecting the next design run [206]. In this chapter, we focus on the online

prediction scenario and test our approach using the classical density functional theory (cDFT) calculations for one-dimensional (1D) hard-rod systems, with different external potentials. The availability of exact densities and free-energy functional for various 1D systems of hard rods allows us to validate the efficiency of the surrogate model with the ground truth. Here the fundamental difficulty is on a user-specified external potential–an arbitrary functional input with infinite dimensions. For constructing statistical emulators, we assume that no parametric form of this functional input is available. We demonstrate that the new integrated approach provides a high-fidelity prediction of particle density profiles and grand potentials with a controlled error in the probabilistic sense, more accurate than a GP emulator with conventional "space-filling" designs, and it is more computationally scalable than direct cDFT calculations.

We highlight the key novelty of this work in developing the surrogate model and error control criteria for functional mapping. First, while some popular criteria, such as the D-optimality, were introduced in the active learning framework [40, 207], the threshold of determining whether a test input can be reliably predicted may be hard to choose, as the threshold may not be directly related to the error in prediction. In practice, one may need to tune the threshold for different systems for a given error bound. Here, we use the internal uncertainty assessment of the GP emulator to decide whether a test input can be predicted precisely. For any given threshold of predictive error and probability tolerance bound, our approach automatically defines the criterion to control the predictive error below the threshold with a large probability satisfying the probability tolerance bound (see Section 5.3.2). We demonstrate that our approach can identify whether an upcoming input can be reliably predicted, and thus it has much better performance than the conventional statistical emulation by training the emulator with a space-filling design, such as random sampling. Furthermore, we derive an iterative formula that reduces the computational complexity for

GP models with augmented samples. In each iteration, updating our algorithm only requires $\mathcal{O}(n^2)$ operations, while a direct approach requires $\mathcal{O}(n^3)$ operations. Other emulation approaches, such as those enforcing the invariance symmetries, can be easily included in our integrated framework for reliable predictions with a controlled error.

5.2 Classical density functional theory

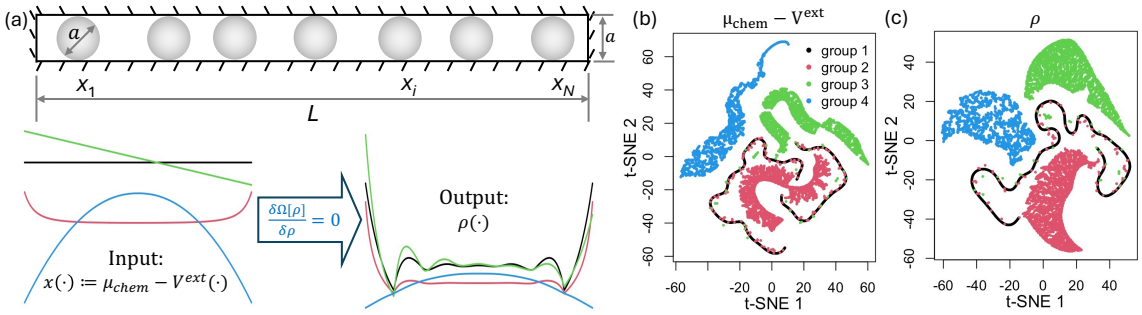

Figure 5.1: (a) Functional learning in the context of cDFT for one-dimensional hard rods of uniform size a in an external field $V^{ext}(.)$ defined in a domain of size L. (b)-(c) t-distributed stochastic neighbor embedding (t-SNE) for visualization of input $\mu_{chem} - V^{ext}(.)$ (middle panel) and output $\rho(.)$ (right panel). For each group, 2000 density profiles are generated based on different combinations of μ_{chem} and parameters in $V^{ext}(.)$.

Classical density functional theory (cDFT) is a statistical-mechanical method that describes the thermodynamic properties of equilibrium systems in terms of the density profiles of individual particles [208, 209]. In a nutshell, cDFT calculations are based on the minimization of a grand potential functional that, for systems consisting of only one type of particle, can be expressed as

$$\Omega[\rho] = F_{id}[\rho] + F_{ex}[\rho] + \int d\mathbf{s}\rho(\mathbf{s})[V^{ext}(\mathbf{s}) - \mu_{chem}], \tag{5.1}$$

where $\rho(\mathbf{s})$ denotes the particle density at position $\mathbf{s}$, $F_{id}[\rho]$ is the free-energy functional of an ideal gas, and $F_{ex}[\rho]$ is called the excess free-energy functional, $V^{ext}(\mathbf{s})$ denotes a

one-body external potential, and μ_{chem} is the chemical potential of the particles. The ideal-gas functional has an exact form

$$\beta_{en}F_{id}[\rho] = \int d\mathbf{s}\rho(\mathbf{s})\{\log[\rho(\mathbf{s})\Lambda_{\text{th}}^3] - 1\},$$

where Λ_{th} is the thermal wavelength and $\beta_{en} = 1/(k_B T_{emp})$ with k_B being the Boltzmann constant and T_{emp} the absolute temperature. While T_{emp} and μ_{chem} are thermodynamic variables that define the equilibrium condition, Λ_{th} and $F_{ex}[\rho]$ depend on the intrinsic properties of the system such as the particle mass and inter-particle potential. In this chapter, we set $\beta_{en} = \Lambda_{\text{th}} = 1$ as the units of energy and length.

One key premise of cDFT calculations is that $V^{ext}(\mathbf{s})$ is uniquely determined by $\rho(\mathbf{s})$ such that, given an analytical expression for $F_{ex}[\rho]$, $\rho(\mathbf{s})$ can be solved by minimizing $\Omega[\rho]$ and, subsequently, all thermodynamic properties of the system can be derived. The equilibrium density profile minimizes Ω and thus satisfies the Euler-Lagrange equation:

$$\rho(\mathbf{s}) = \exp\left\{\mu_{chem} - \delta F_{ex}/\delta\rho - V^{ext}(\mathbf{s})\right\}. \tag{5.2}$$

Equation (5.2) provides a functional map between the one-body density $\rho(\mathbf{s})$ and the one-body external potential $V^{ext}(\mathbf{s})$. Because in general F_{ex} is a complicated functional of $\rho(\mathbf{s})$, solving $\rho(\mathbf{s})$ from the Euler-Lagrange equation is often computationally demanding. Besides, the formulation of an accurate expression for F_{ex} is a formidable task for complex molecular systems. In this chapter, we demonstrate that statistical emulation will be helpful to solve both problems.

For proof of concept, we consider a statistical emulator of cDFT for one-dimensional (1D) systems consisting of identical hard rods (HR). For such systems, the excess free-energy

functional, thus the grand potential, is exactly known [210]. The analytical results were derived decades ago and are reproduced in Appendix E.1. Given any external potential $V^{ext}(\mathbf{s})$ and chemical potential μ_{chem}, the density profile of hard rods can be numerically solved (e.g., by Picard iteration) from Equation (5.2), and it is plugged into Equations (5.2), (E.2), and (E.3) for computing the grand potential and other thermodynamic quantities.

For a given system defined by μ_{chem} and $V^{ext}(\mathbf{s})$, statistical emulation aims to predict the particle density and free-energy functionals as in direct cDFT calculations (illustrated in Figure 5.1(a)). Previous work in cDFT [211, 195, 212] and Kohn-Sham density functional theory (KS-DFT) [213, 214] demonstrate the performance of ML models with a single parametric form for the external potential with different choices of parameters. In this chapter, we show that the active learning procedure is applicable to multiple classes of external potential similar to cDFT calculations with the same intrinsic Helmholtz energy functional. We demonstrate that, when presented with a new class of external potentials, the model can discern whether this new input can be accurately predicted. As mentioned above, functional mapping represents one of the biggest obstacles in statistical emulation as well as machine learning. Filling the entire functional input space directly with a limited number of simulation runs, such as random sampling or the LHS, is not generally possible. When the external potential function is distant from the training input set, both statistical emulators and machine learning methods become inaccurate. We provide a solution to this fundamental problem by integrating direct cDFT calculations with a GP emulator through an active-learning framework. We derive a probabilistic bound to control the overall prediction error of the particle density based on the internal assessment of the uncertainty from the statistical emulator, making our approach applicable to functional learning in infinite dimensional spaces, as well as reducing the computational costs for inputs that can be accurately predicted. Here our algorithm can reliably learn multiple classes of external

potentials, while earlier applications of ML methods in both KS-DFT (e.g. [214]) and cDFT (e.g. [195, 212]) only demonstrate the performance of their models on one particular class of the external potential. Besides, the active learning scheme developed in this chapter enables the ML approaches to discern whether outcomes of a test input can be reliably predicted. This ability allows us to control predictive errors as well as substantially reduce the computational cost.

In Figure 5.1(b) and (c), the input $\mu_{chem} - V^{ext}(.)$ and output $\rho(.)$ of the statistical emulator are projected onto two t-distributed Stochastic Neighborhood Embedding (t-SNE) factors as commonly used for visualizing high-dimensional data [189]. It is evident that the inputs generated from different groups of input functions form different clusters. Similar patterns can be observed for the corresponding particle densities. In addition, group 1 forms a curve in Figure 5.1(b) due to the fact that the only change of the input in group 1 is μ_{chem}, which only results in a shift in the input space, making group 1 mapping the simplest to learn for the statistical emulator, as to be demonstrated later in the section on numerical results. We noticed several points in groups 2 and 3 are close to those in group 1. These points correspond to samples whose input parameters are close to 0 and are thus nearly equivalent to group 1's inputs. These patterns demonstrate the input-output relationship can be captured by distance metrics, as the smaller distance between chemical potentials leads to the higher similarity between the particle densities. This validates the use of kernels in the GP to map the distance between inputs (chemical and external potentials) to the correlation between output (particle densities), as the smaller distance in the input space indicates a larger correlation or similarity between outputs. Unlike the dimension reduction approach shown in Figure 5.1, we will model the entire density profile jointly in a statistical emulator to encode all information.

5.3 A reliable Gaussian process emulator by active learning with error control

5.3.1 Parallel partial Gaussian process emulation for vectorized outputs

For demonstration purposes, our statistical emulator is built upon the parallel partial Gaussian Process (PP-GP) emulation approach [47], introduced in Section 1.1.2, for representing the map between the particle density, a vectorized output defined on massive grid points, and the input function composed of the chemical and external potentials. Other emulators that produce uncertainty quantification for predictions can be used as well. The GP emulator is one of the most commonly used surrogate models for expensive physics-based calculations, and it has been widely used for emulating molecular simulation, the potential energy surface, and atomic force fields [191, 215, 216, 217, 197]. In combination with the dimension reduction technique, the GP emulator has also been used as a surrogate model for approximating the KS-DFT calculation of electron density [214].

Here, we let $\mathbf{x} = \mu_{chem} - V^{ext}(.)$ denote the functional input discretized at p spatial grid points. For any input $\mathbf{x}$, the particle density is defined at k grid points, written as $\boldsymbol{\rho}(\mathbf{x}) = [\rho_1(\mathbf{x}), \rho_2(\mathbf{x}), \ldots, \rho_k(\mathbf{x})]$. In this example, we set the number of spatial grid points used for discretizing external potential and density to be the same, i.e., $p = k$. When applying the PP-GP emulator to model the particle density in cDFT, we employ a constant mean and an isotropic Matérn kernel function with roughness parameter 5/2 (Equation (1.2)) for the spatial correlation. Thus, given n cDFT calculations with inputs $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the density vector at any spatial grid point j is assumed to follow a multivariate normal distribution $\boldsymbol{\rho}_j = [\rho_j(\mathbf{x}_1), \ldots, \rho_j(\mathbf{x}_n)]^T \sim \mathcal{MN}(\beta_j \mathbf{1}_n, \sigma_j^2 \mathbf{R})$. The predictive distribution then follows (1.14) with all terms related to z replaced by ρ.

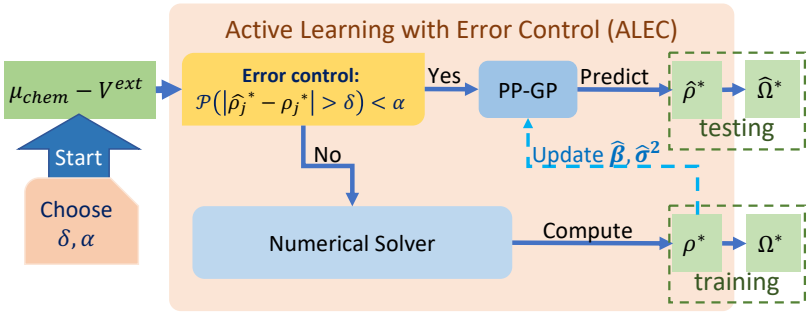

Figure 5.2: An overview of the active learning with error control framework.

5.3.2 ALEC: active learning with error control approach

Active learning is a sequential design approach in statistics. In an "on-the-fly" online prediction scenario, there is no information regarding the future input to be predicted. If a configuration is new to the current training set, i.e., it is located in a sparsely sampled region, we may not be able to accurately predict this sample by the emulator. For a large functional input space, it is almost inevitable to extrapolate the sampled input space, as the number of simulated runs that populate this input space is sparse. Therefore, it is necessary to establish a criterion for determining whether an upcoming configuration can be reliably predicted.

In this chapter, we develop the active learning with error control (ALEC) emulator that can automatically control predictive error for simulations with high or infinite dimensional input space. A few criteria were studied before to detect the test case hard to be predicted, such as the D-optimality criterion [40] and the predictive variance criterion [204]. Yet, the threshold of these strategies often requires to be chosen empirically, as they may not be directly related to the predictive error. In our automatic approach, a new criterion has been established such that the threshold can be chosen automatically for any given predictive error bound and probability tolerance bound, based on the internal uncertainty assessment of the emulator. Figure 5.2 shows an overview of our strategy. For a given testing

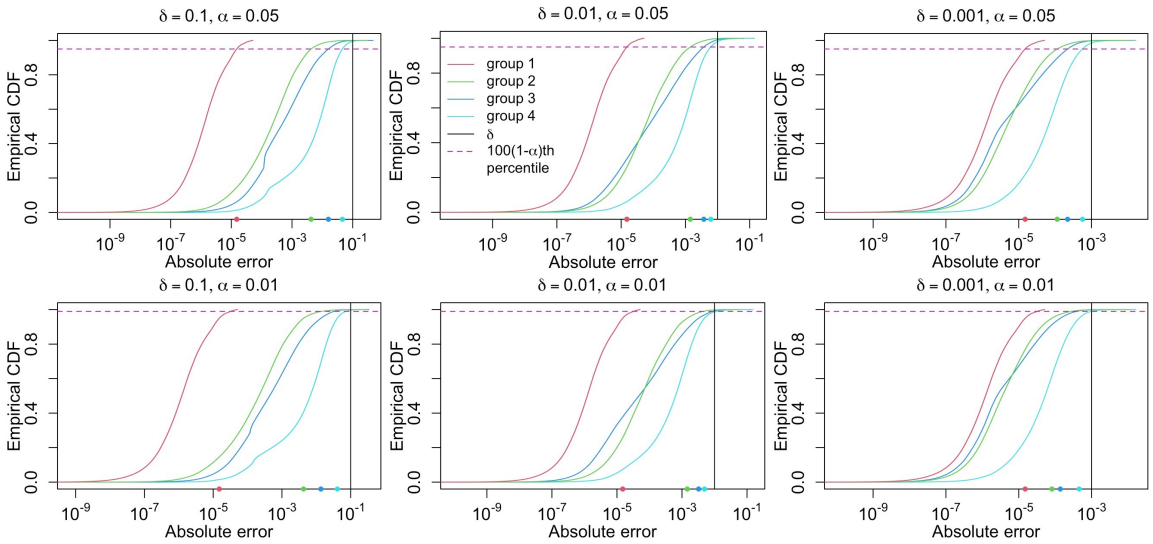

Figure 5.3: Empirical CDF of absolute error of particle densities $|\rho_j(\mathbf{x}_i^*) - \hat{\rho}_j(\mathbf{x}_i^*)|$ for 4 classes of input functions at all grid points using the criterion (5.8) with different error bounds δ and probability tolerance thresholds α. The horizontal line and the vertical line are $1-\alpha$ and δ, respectively. On the x-axis, the $1-\alpha$ percentiles of the absolute errors are recorded, all of which are less than δ.

input $\mathbf{x}^*$, if the criterion that controls the predictive error is satisfied as shown in the yellow box, the particle density profile will be predicted; otherwise, a numerical solver will be called for computing the particle density corresponding to $\mathbf{x}^*$, and this new information will be added to the training set and used to update the PP-GP emulator.

Let us first consider controlling the predictive error for particle density at a fixed coordinate j based on the PP-GP algorithm. Given a prespecified error bound $\delta_j > 0$, we aim to have the predictive error of a test input $\mathbf{x}^*$ smaller than $\delta_j > 0$ for more than $1-\alpha$ of the predictions, where α is a small value (e.g. $\alpha = 0.05$). This means

$$\mathbb{P}\big(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \delta_j\big) \leq \alpha, \tag{5.3}$$

i.e., the probability that the absolute error between the predicted and true densities at the j-th coordinate exceeds δ_j is less than α. Since small predictive variance from the emulator

indicates high-fidelity in prediction, we predict the test input $\mathbf{x}^*$ if

$$\sqrt{\hat{\sigma}_j^2 c^{**}} \leq \frac{\delta_j}{t_{\alpha/2}(n - p_\mu)}, \tag{5.4}$$

where $t_{\alpha/2}(n-p_\mu)$ is the upper $\alpha/2$ quantile of a Student's t-distribution with $n-p_\mu$ degrees of freedom, c^{**} and $\hat{\sigma}_j^2$ are given in Equations (1.12) and (1.17), respectively. Otherwise, we will compute this particle density directly from cDFT using a numerical solver, as the assessed uncertainty for predicting this test input is large. This strategy satisfies the probabilistic error control outlined in Equation (5.3), as shown in Appendix E.3.

We have a few remarks regarding this strategy. First, the error bound δ_j and probability tolerance α have interpretations. Suppose, for instance, one expects to see the predictive error in more than 95% of predictions smaller than 0.01, one can then let $\delta = 0.01$ and $\alpha = 0.05$. Second, in the active learning strategy, one reduces the sample space from the original space $\mathcal{X}$ to $\mathcal{X}_{F,j} = \{\mathbf{x}^* : \sqrt{\hat{\sigma}_j^2 c^{**}} \leq \delta_j / t_{\alpha/2}(n - p_\mu)\}$, where the subscript F denotes the input set that can be reliably predicted. A numerical solver for cDFT needs to be called for any input $\mathbf{x}^*$ that does not satisfy Equation (5.4). Though selecting a smaller δ or α leads to a smaller predictive error, more evaluations from the numerical solver would be required, which increases the computational cost. Thus, a balance between the size of the cut-off value and computational time is needed in practice. Third, the degree of freedom of the Student's t-distribution increases with the sample size. When the sample size is sufficiently large, the t-distribution can be accurately approximated by a standard normal distribution. For instance, when $n - p_\mu = 300$ and $\alpha = 0.05$, $t_{\alpha/2}(n - p_\mu) \approx 1.968$ and the upper $\alpha/2$ quantile of a normal distribution $Z_{\alpha/2} \approx 1.960$, which only differs from that of a t-distribution by around 0.5%. Thus one can use the normal approximation if the sample size is large. Finally, the error control from Equation (5.4) means that if the

statistical emulator is a correct representation of the reality, and the number of test samples is infinitely large, we have at least $1-\alpha$ probability such that the absolute predictive error is less than δ. This does not preclude the cases where in slightly more than $1-\alpha$ of the predictions, the predictive error is larger than δ, due to model misspecification or the limited number of test samples. Fortunately, the predictive error at these inputs is typically close to δ, as the assessed uncertainty of these inputs is smaller than the threshold.

We can extend the strategy for controlling the error of the density at a fixed coordinate j to the overall error of density prediction. Consider that we use a uniform cutoff value δ, and that we make predictions for a test input $\mathbf{x}^*$ if each coordinate satisfies Equation (5.4), i.e.,

$$\sqrt{\max_j\{\hat{\sigma}_j^2 c^{**}\}} \leq \frac{\delta}{t_{\alpha/2}(n-p_\mu)}. \tag{5.5}$$

We call the criterion in Equation (5.5) the maximum variance selection criterion. Under this criterion, we are able to derive the probabilistic bound to control the predictive error of particle density at each coordinate,

$$\mathbb{P}\big(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \delta\big) \leq \alpha, \quad \text{for } j = 1, ..., k, \tag{5.6}$$

and the root mean squared error (RMSE), defined as $\text{RMSE} = \sqrt{\sum_{j=1}^k (\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2/k}$, follows

$$\mathbb{P}\big(\text{RMSE} > \delta\big) \leq \alpha. \tag{5.7}$$

The detailed derivation of Equation (5.6) and (5.7) is given in Appendix E.3. Equations (5.6) and (5.7) imply that the overall predictive error can be controlled in a probabilistic way under the maximum variance selection criterion in Equation (5.5).

In practice, the cut-off in (5.5) is too conservative, and we rarely see more than

α samples larger than the claimed threshold. A less restrictive criterion to control the predictive error is by predicting any input set $\mathbf{x}^*$ such that

$$\delta^* := \sqrt{\frac{\sum_{j=1}^{k} \hat{\sigma}_j^2 c^{**}}{k}} \leq \frac{\delta}{t_{\alpha/2}(n - p_\mu)}. \tag{5.8}$$

We call the criterion in (5.8) the average variance selection criterion. Equation (5.8) controls the predictive error of vectorized outputs (particle densities) in the average sense, while it does not guarantee the error control as in Equation (5.5). In practice, one may select a large number of points for numerical solvers to run using criterion (5.5), thereby requiring a larger computational cost, while the predictive error control can be achieved using criterion (5.8) with much less training samples.

To quantify whether the error is indeed controlled based on our threshold, we plot the empirical cumulative distribution function (CDF) of the absolute errors in Figure 5.3 using the criterion (5.8) with different error bounds and probability tolerance thresholds α for all 4 classes of input functions considered herein. In the upper middle panel, for instance, the vertical solid line marks the chosen threshold $\delta = 0.01$ and the purple dashed line labels the percentile corresponding to $1 - \alpha = 0.95$. On the x-axis, we mark the values of the 95th percentile of the absolute errors for each group of input, all of which are less than 0.01. In the remaining panels of Figure 5.3, the empirical CDFs of the absolute errors for various α and δ values are illustrated. In general, the smaller error bound or probability tolerance threshold, the smaller the predictive error is, while more samples are required for achieving this accuracy level. Thus, a balance between the computational cost and accuracy level is generally required.

5.3.3 Sequential update of the ALEC emulator

When calculating the predictive mean and predictive correlation in Equations (1.15) and (1.12), one needs to solve $\mathbf{R}^{-1}\mathbf{u}$, for an n-dimensional real-valued vector $\mathbf{u} \in \mathbb{R}^n$. In the PP-GP emulator [47], we implement the Cholesky decomposition of the correlation matrix $\mathbf{R} = \mathbf{L}\mathbf{L}^T$, and use the forward and backward substitution algorithms to compute $\mathbf{R}^{-1}\mathbf{u}$. In the ALEC emulator, whenever a sample $\mathbf{x}^* \in \mathbb{R}^p$ is added to the previous training set with n samples, the PP-GP model needs to be updated accordingly. Directly applying this approach will require $\mathcal{O}(n^3)$ operations for computing the Cholesky decomposition each time when the sample size is n. To reduce the computational cost, we introduce a new approach that takes only $\mathcal{O}(n^2)$ operations in performing the Cholesky decomposition.

To update $\mathbf{L}$ with the addition of $\mathbf{x}^*$, we denote the correlation matrix of previous n design elements as $\mathbf{R}_n = \mathbf{L}_n\mathbf{L}_n^T$, with $\mathbf{L}_n$ being the Cholesky decomposition of $\mathbf{R}_n$. Next, we write the updated correlation matrix $\mathbf{R}_{n+1}$:

$$\mathbf{R}_{n+1} = \begin{bmatrix} \mathbf{R}_n & \mathbf{r}_n(\mathbf{x}^*) \\ \mathbf{r}_n^T(\mathbf{x}^*) & c(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix},$$

where $\mathbf{r}_n(\mathbf{x}^*) = [c(\mathbf{x}^*, \mathbf{x}_1), ..., c(\mathbf{x}^*, \mathbf{x}_n)]^T$. In evaluating $\mathbf{R}_{n+1} = \mathbf{L}_{n+1}\mathbf{L}_{n+1}^T$, it is clear that the first n rows of the lower triangular part of $\mathbf{L}_{n+1}$ is identical to the lower triangular part of $\mathbf{L}_n$

$$\mathbf{L}_{n+1} = \begin{bmatrix} \mathbf{L}_n & \mathbf{0}_n \\ \mathbf{L}_{n+1}(n+1, 1:n) & L_{n+1}(n+1, n+1) \end{bmatrix}.$$

Therefore, only the last row of $\mathbf{L}_{n+1}$ needs to be computed:

$$L_{n+1}(n+1,j) = \frac{1}{L_n(j,j)}\Big(r_n(j) - \sum_{m=1}^{j-1} L_{n+1}(n+1,m)L_n(j,m)\Big) \quad j = 1,\ldots,n, \quad (5.9)$$

$$L_{n+1}(n+1,n+1) = \sqrt{c(\mathbf{x}^*,\mathbf{x}^*) - \sum_{m=1}^{n} L_{n+1}(n+1,m)^2}, \quad (5.10)$$

with $L_n(i,j)$ and $L_{n+1}(i,j)$ denoting the (i,j) entry of $\mathbf{L}_n$ and $\mathbf{L}_{n+1}$, respectively, and $r_n(j)$ denoting the j-th element of $\mathbf{r}_n(\mathbf{x}^*)$. Equations (5.9) and (5.10) indicate that updating $\mathbf{L}_{n+1}$ requires $\mathcal{O}(n^2)$ computational operations, for $i,j = 1,...,n$. After obtaining the Cholesky decomposition $\mathbf{L}_{n+1}$ we can update other terms required in computing the predictive mean and variance.

5.3.4 Algorithm and the computational cost

We summarize the ALEC emulator in Algorithm 5.1. Given the initial PP-GP emulator, predictive error bound δ, and probability tolerance threshold α, the ALEC emulator determines whether the particle densities can be reliably predicted for a new test input. The output contains the predictive particle densities for all test inputs and they are plugged into the energy functionals to compute the energy. Note that the range parameter in the kernel needs to be numerically estimated, which has the largest computational cost in the algorithm. When more training samples are added, the previously estimated range parameter $\hat{\gamma}$ may no longer be accurate if our initial model has a small training sample size. To compromise the computational cost and accuracy in predictions, the range parameter in the ALEC emulator is re-estimated for every 50 training samples, if the training size is less than 350. When the training size is large, the estimated range parameter does not change much. In general, the frequency of re-estimating the kernel parameter depends on the computational budget.

Denote the initial training size as n_{ini} and the number of augmented samples as

n_{aug}. The computational cost of constructing the Cholesky decomposition of the initial ALEC emulator is $\mathcal{O}(n_{ini}^3)$, and the total computational order for updating the Cholesky decomposition is then $\mathcal{O}(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^2)$. For computing the predictive mean and predictive variance, it takes $\mathcal{O}(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} nk)$ and $\mathcal{O}(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^2k)$, respectively.

The computational complexity of the online updating approach reduces the cost compared to directing computing the Cholesky decomposition at each iteration, which costs $\mathcal{O}(\sum_{n=n_{ini}+1}^{n_{ini}+n_{aug}} n^3)$. The computational complexity is significantly faster than the time required to calculate the density using a numerical solver for computing the particle density at each test input. Indeed, the largest computational cost of our approach comes from the numerical solver for the training input set, as will be seen in Figure 5.7. Since only a small fraction of samples are used as the initial training and augmented sample, the computational cost of the ALEC emulator is much smaller than running numerical solvers for each density.

Algorithm 5.1 ALEC emulator

Input: The number of test samples n_t, testing inputs $\mathcal{X}$, Cholesky factor $\mathbf{L}$ of $\mathbf{R}$ in the initial PP-GP model, estimated variance parameters $\hat{\boldsymbol{\sigma}}^2 = [\hat{\sigma}_1^2, ..., \hat{\sigma}_k^2]$ and mean parameters $\hat{\boldsymbol{\beta}}^2 = [\hat{\beta}_1^2, ..., \hat{\beta}_k^2]$ in the initial PP-GP model, probability level α and threshold δ chosen by the user.

$n_{aug} \leftarrow 0$;
for $i = 1$ to n_t **do**
    Set i-th test sample in $\mathcal{X}$ as $\mathbf{x}^*$ and calculate $\hat{\sigma}_j^2 c^{**}$ for $j = 1, ..., k$ and δ^* via (1.12) and (5.8), respectively;
    if $\delta^* < \delta / t_{\alpha/2}(n - p_\mu)$ **then**
        Predict for the predictive distribution in (1.14) for this $\mathbf{x}^*$, and record predictive mean $\hat{\boldsymbol{\rho}}(\mathbf{x}^*)$ and scale $\hat{\boldsymbol{\sigma}}^2 c^{**}$;
    else
        $n_{aug} \leftarrow n_{aug} + 1$;
        Using the numerical solver to compute the corresponding output $\boldsymbol{\rho}(\mathbf{x}^*)$;
        if the training size is a multiple of 50 **and** the training size ≤ 350 **then**
            Rebuild the PP-GP emulator with retrained the range parameter γ in Matérn kernel;
        else
            Update PP-GP emulator with the added input $\mathbf{x}^*$;
            Update parameters $\hat{\boldsymbol{\sigma}}^2$ and $\hat{\boldsymbol{\beta}}^2$;
        end if
    end if
end for

Output: The number of augmented size n_{aug}, predicted mean, and variance of particle densities for remaining $n_t - n_{aug}$ samples.

5.4 Numerical Results

Here we evaluate the predictive performance of our surrogate model based on particle densities generated from four classes of external potentials (closed-forms shown in Appendix E.2) with the length of hard rods $a = 1$ and the domain size $L = 9$. The values of chemical external input and parameters in each external potential are changed when generating the data. We assume the external potential functions and chemical potentials are used as inputs, while the parametric forms of the external potential functions are not known. We will test the predictive performance separately for each class of external potential functions, and jointly by combining all classes of external potential functions, in Section 5.4.1 and Section 5.4.2, respectively. Section 5.4.3 gives the predictive performance of our

approach for predicting a new class of densities. The code and data used in this chapter are publicly available: `https://github.com/UncertaintyQuantification/ALEC`.

We record the out-of-sample root mean squared error (RMSE) of particle density and the grand potential of the test samples from

$$RMSE_\rho = \sqrt{\frac{\sum_{i=1}^{n^*}\sum_{j=1}^{k}(\rho_j(\mathbf{x}_i^*) - \hat{\rho}_j(\mathbf{x}_i^*))^2}{n^*k}},$$
$$RMSE_\Omega = \sqrt{\frac{\sum_{i=1}^{n^*}(\Omega(\mathbf{x}_i^*) - \hat{\Omega}(\mathbf{x}_i^*))^2}{n^*}},$$

where $\hat{\rho}_j(\mathbf{x}_i^*)$ is the predicted density of i-th hold out sample at j-th coordinate, and $\Omega(\mathbf{x}_i^*)$ and $\hat{\Omega}(\mathbf{x}_i^*)$ are the predicted and true grand potential for i-th hold out sample, respectively. Other criteria, such as the proportion of the test samples covered in the 95% predictive intervals and the average length of the predictive intervals, are provided in supplementary materials. Note that we only compute the predictive error on those inputs we predict, not including those initial samples or the augmented samples computed by the numerical solver.

We generate 2000 samples for each group with LHS, and compare our surrogate model with three other commonly used approaches. The first two approaches use conventional random sample (RS) designs, where we randomly draw n_{train} samples from the LHS samples as the training dataset and use the remaining samples as testing data. We have two distinct training sample sizes for RS samples. The sample size in the first RS sample (RS1) is specified as the initial training size in ALEC, while the sample size in the second RS sample (RS2) is specified as the final training size in ALEC. For predicting any test input, the training sample size in active learning is neither smaller than the training sample size in RS1 nor larger than the training sample size in RS1. We have also implemented the active learning algorithm with D-optimality [40], where the details are given in Appendix E.4.

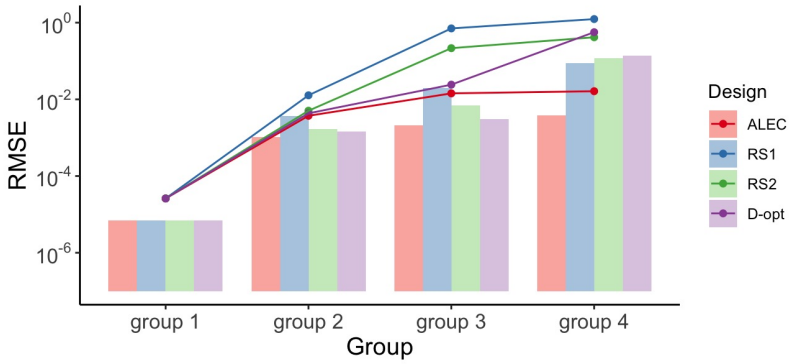

Figure 5.4: The bar plots and line plots show the out-of-sample RMSE of density ρ, and RMSE of grand potential Ω, respectively. The ALEC emulator is compared with other methods with three other designs: RS1, RS2, and D-opt (Active learning with D-optimality). For the ALEC emulator, we use $\delta = 0.01$ and $\alpha = 0.05$ as the threshold, and the augmented sizes for groups 1-4 are 0, 7, 21, and 504, respectively. In D-opt, the augmented sizes for groups 1-4 are 0, 7, 21, and 582, respectively. The number of training sample sizes from RS1 and RS2 are the same as the initial sample size and terminal sample size in ALEC.

5.4.1 Particle density and grand potential prediction for each group of external potential field separately

We first test different approaches for four families of functional inputs separately. The density profiles are initially trained using PP-GP with $n_{ini} = 20$ inputs per group. Then following our ALEC algorithm, the surrogate model is sequentially updated by augmenting the train data with samples selected by the average variance selection criterion in Equation (5.8). After obtaining the predicted density $\hat{\rho}$ for the remaining test samples, we predict the grand potential Ω by plugging $\hat{\rho}$ into the energy functional in Equation (E.2) in Appendix E.1.

Figure 5.4 displays the overall performance of different surrogate models (more detailed results can be found in the table in the supplementary materials). Here the threshold in the ALEC algorithm is set as $\delta = 0.01$ and $\alpha = 0.05$, representing a strategy for having less than 5% of the coordinate to have an absolute error of density prediction larger than 0.01. The first group of external potentials has the least flexibility, as the only change in

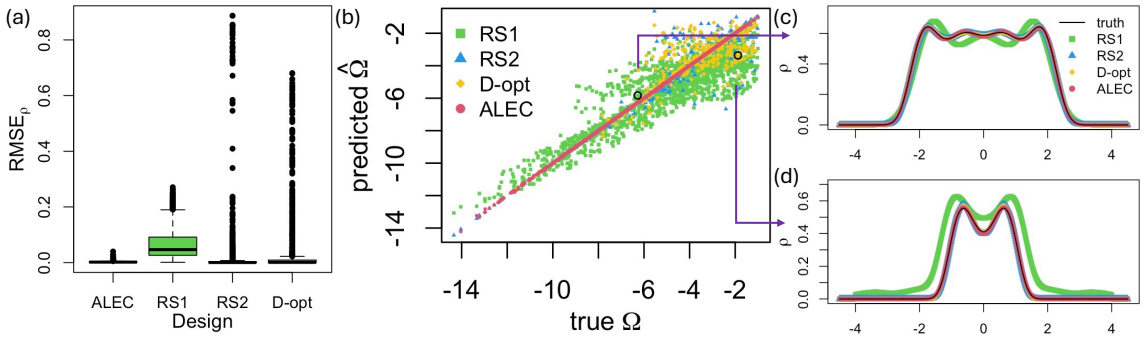

Figure 5.5: A comparison of four surrogate models for predicting the group 4 functions: Active learning with error control (ALEC), random sampling with the number of training points as the initial number of training points in ALEC (RS1) and with the number of training points as the final number of training points in ALEC (RS2), and active learning with D-optimality (D-opt). (a) Boxplots of predictive RMSE of each predicted density in the testing dataset. (b) Predicted grand potential $\hat{\Omega}$ vs. the truth Ω. (c)-(d) True and predicted densities for two selected density profiles.

the input is μ_{chem}. Thus, predicting the density profile for the first group of functions is the easiest. Models with only 20 inputs in the conventional RS design can predict particle densities relatively well, and no augmented sample is needed in the ALEC emulator. For the other three functional groups, the active learning algorithm has the smallest predictive RMSE on density ρ and grand potential Ω among all four designs. It is worth noting that the number of training inputs from active learning is always not larger than that in the RS2 design, but the predictive error by active learning is much smaller than the one by RS2. This is because the active learning strategy accurately identifies test samples hard to be predicted, and a numerical solver is called for computing these samples. Thus the computing budget is more efficiently spent with respect to the accuracy of predictions.

Figure 5.5 presents more detailed prediction results of the particle densities and grand potentials for the fourth class of external potentials. The boxplot in Figure 5.5(a) gives the out-of-sample RMSE for predicting the particle densities. Many densities are inaccurately predicted based on the samples from RS1, RS2, or D-opt, resulting in large errors in predicting the grand potential (Figure 5.5(b)). In comparison, the ALEC emulator

identifies the samples that are difficult to be predicted and it uses them to enhance the emulator, thus providing accurate predictions for the remaining test samples. Although the total computational cost of the ALEC is similar to the methods with RS2 and D-opt schemes, the ALEC emulator results in better predictive accuracy.

5.4.2 Predicting particle densities and grand potentials for all groups of external potentials

In this subsection, we describe a more challenging scenario, where we aim to predict four groups of samples together. We assume that only the chemical potential and the external potential are used as input, while the parametric forms of the external potential are not known. The density profile and grand potential are harder to be predicted accurately in this case, as the input contains more degrees of freedom, leading to larger variability of the density and energy functionals, compared to the ones from any single class of input function alone.

Since the inputs are a random sample from four classes of external potentials, we use the first $n_{ini} = 80$ samples as initial inputs to build the ALEC emulator. When a test sample comes, we use the criterion from Equation (5.8) to determine whether the test sample can be accurately predicted by the ALEC emulator or a numerical solver will be needed for solving the system directly.

We show the emulator's overall performance in the leftmost group (named as all groups) in Figure 5.6 (detailed results are displayed in the supplementary materials). On average, the ALEC emulator performs much better, where the predictive RMSE of both particle densities and grand potentials is at least one magnitude smaller than all other approaches. A total of 873 training samples are added through the active learning algorithm, which is more than the total number of training samples added for building the emulator

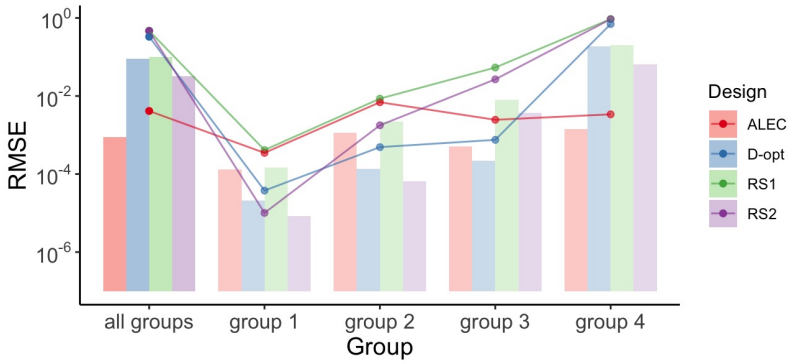

Figure 5.6: Out-of-sample RMSE of particle density ρ (bar plots) and grand potential Ω (line plots) for active learning with error control, RS1 design, RS2 design and active learning with D-optimality. The overall RMSE for combined all groups (solid bars and dots) and the RMSE of each individual group (shadow bars and dots) are both recorded. ALEC contains 80 inputs as initial training size, and the total number of augmented samples in the ALEC emulator ($\delta = 0.01$, $\alpha = 0.05$) is 873, with 0, 1, 58, and 814 augmented samples in groups 1 to 4, respectively. In D-opt, the total number of augmented samples is 935, and the augmented sizes for groups 1-4 are 59, 73, 393, and 410, respectively. RS1 contains a total of the first 80 training samples after uniformly mixing all four groups of density together. RS2 contains the first 953 training inputs, with 245, 221, 238, and 249 for groups 1-4, respectively. Since the ALEC emulator identifies the hardest region (group 4) to be predicted, thereby augmenting most samples from group 4, it has the smallest predictive error overall and predictive errors of all groups are controlled.

separately. This is because having a larger input space containing all 4 classes of external potentials increases the difficulty of learning densities, requiring more samples given the same threshold of the error. Note that the good predictive performance by ALEC is achieved using a similar or smaller sample size in training the emulator compared to the D-opt and RS2 approaches, by efficiently allocating the computing budget on computing particle densities hard to be predicted, and accurately predicting the rest of the densities with a negligible computational cost.

When examining the performance of different approaches in each function group, we found that the RMSE of particle densities from ALEC is homogenous across groups, while RS2 and D-opt approaches have smaller predictive errors for the first two groups, but have much worse performance for the fourth group. This is because RS2 and D-opt select more points from the first two groups to train the emulator, thereby yielding a similar

or better result than ALEC in these two groups. However, the error from the first two groups can be controlled below the threshold of 0.01 for more than 95% of the test samples with no augmented samples or only a small number of augmented samples. In the ALEC emulator, for instance, the augmented samples from groups 1 and 2 are 0 and 1, respectively. On the contrary, the particle densities and grand potential energy from group 4 are the hardest to be predicted. This is automatically identified by ALEC, and the most computing budget (814 augmented samples) is spent on the fourth group to ensure the predictive errors of most samples are controlled below the threshold. This is exactly what we intend to accomplish. The ALEC can identify input regions that are difficult to predict, and then put more computing resources on those regions to control the predictive error. Consequently, the predictive error is controlled below the threshold for all subclasses, even if we do not know the labels of subclasses from the test samples, as all input classes are combined for predictions.

Another interesting finding is that the additional training samples from other input functional classes do not substantially improve the predictive performance for a specific input class. This is because the inputs from one class are distant from another class, as shown in Figure 5.1(b), and thus the correlation between different input classes is small. It may be of interest to automatically identify the cluster in the training and split the training sample when constructing the emulator. This is useful as splitting the samples can reduce the computational complexity of the emulator. To explore this idea, we utilized a simple input decomposition technique. When the size of a cluster reaches 400, we divide it into two sub-clusters using the K-nearest neighbors (KNN) algorithm [218]. In this approach, the number of augmented samples and predictive RMSE of particle densities are 779 and 9.73×10^{-4}, respectively. It achieves similar accuracy with 94 fewer training samples than the ALEC algorithm without input decomposition. Developing a more comprehensive way

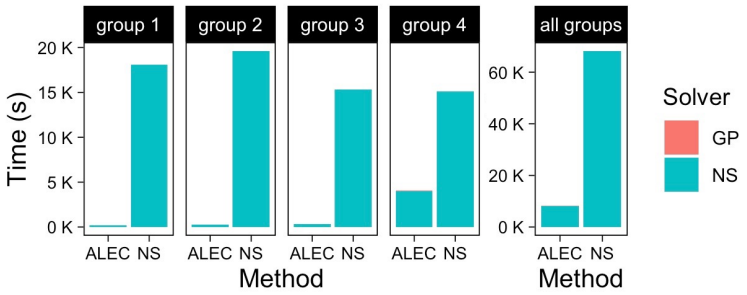

Figure 5.7: The running time for each function group using the ALEC algorithm and using numerical solver (NS). As the running time for building the GP model in ALEC is very small, the bars for GP are nearly invisible. Other approaches, such as RS2 and D-opt, have similar computational costs as ALEC as the training and augmented sample sizes are similar.

for input decomposition approach can further enhance the efficiency and accuracy of the surrogate model.

Figure 5.7 compares the computational time required in the ALEC approach and in calling a numerical solver (NS) each time. The largest computational cost in the ALEC algorithm comes from calling the NS for computing the particle densities of the initial and augmented training samples, while the time in constructing the ALEC emulator and making predictions is negligible, as the training sample size is not large. Since the ALEC emulator only requires a fraction of the samples to be numerically solved, it is much faster than running a numerical solver for each test sample. Note that RS2 and D-opt compared herein have a similar computational cost as the ALEC emulator, as the number of density profiles to be solved by NS is similar, yet the predictive error was not controlled in those approaches.

5.4.3 Predicting a new class of external potential

In this section, we examine the performance of the ALEC approach to predict inputs from a new functional form that is not in the training data set. To test this case, in addition to the existing 8000 samples, we generated 2000 new samples based on the weighted

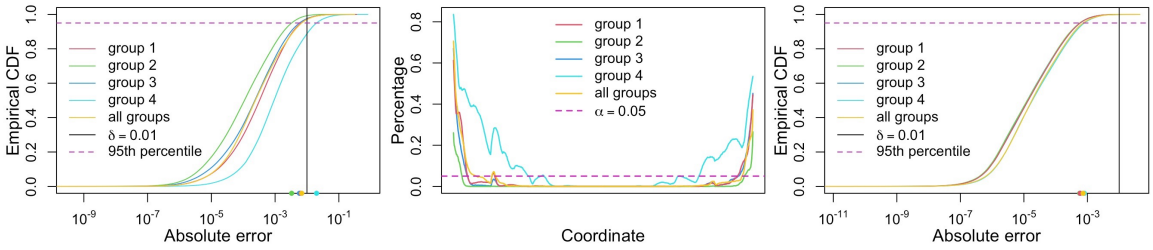

Figure 5.8: Prediction results from the ALEC emulator on a new class of external potentials where ALEC models are trained by the previous four functional groups separately and by all functional groups together. $\delta = 0.01$ and $\alpha = 0.05$ are used. Left panel: the empirical CDF of absolute error using the average variance selection criterion in Equation (5.8). The number of augmented samples using the model for groups 1-4 and all groups combined are 92, 122, 67, 48, and 15, respectively. Middle panel: the percentage that the absolute difference is greater than δ for each coordinate, i.e. $\sum_{i=1}^{n*} \mathbb{1}\{\rho_j(\mathbf{x}_i^*) - \hat{\rho}_j(\mathbf{x}_i^*) > \delta\}/n^*, j = 1, ..., k$. Right panel: the empirical CDF of absolute error using the maximum variance selection criterion in Equation (5.5). The number of augmented samples using the model for groups 1-4 and all groups combined are 408, 395, 384, 362, and 263, respectively.

inputs between groups 2 and 3 as explained in Appendix E.2. We test the performance of the ALEC emulator for predicting the new input class, starting with the final GP emulator developed for groups 1-4 and all groups combined.

The left panel of Figure 5.8 displays empirical CDF plots of the absolute errors of the ALEC emulator using the average variance selection criterion in Equation (5.8). First, the predictive error of most of the test samples in the new group is small, for any initial emulator built upon other groups. Less than $\alpha = 5\%$ of the absolute errors are greater than the chosen error bound $\delta = 0.01$ for models using groups 1-3 and all groups combined. When we start from the initial emulator built from group 4 input, 11.2% of predictive errors are greater than $\delta = 0.01$, which is larger than the probability tolerance threshold $\alpha = 5\%$. A close examination reveals that the percentage of the absolute error greater than δ is typically large for coordinates at the tail of the particle densities (Figure 5.8 middle panel), because of a rapid change in density at both ends, making it distinct from existing densities in group 4. The difference in the particle densities in the new class and group 4 increases the model discrepancy and makes the uncertainty in predictions harder to be quantified. Among the

11% of the test samples with RMSE larger than the threshold, however, most RMSEs are still very close to the threshold value, making this criterion ideal if the computational budget is not large.

On the other hand, the right panel of Figure 5.8 shows the CDF of the RMSEs based on the maximum variance selection criterion in Equation (5.5). Much less than α probability of the predictive error exceeds the threshold δ for all the cases shown here, as the maximum variance selection criterion is conservative, yielding more augmented samples and higher predictive accuracy. When the model discrepancy is large, one may use the maximum variance selection criterion to achieve a higher level of accuracy in predictions.

Chapter 6

Conclusion and future directions

This thesis introduces novel statistical approaches and algorithms for challenging inverse estimation and forward prediction problems motivated by physics, chemistry, and materials science. These methods revolve around a central theme: how to achieve scalable statistical computation and efficient estimation from a limited number of experiments, where each experiment can contain large amounts of data, potentially with substantial noise and high-dimensional inputs.

Chapter 2 introduces the inverse Kalman filter (IKF) method to address the computational bottlenecks of Gaussian processes (GPs) in large-scale inference. The core of IKF is a fast algorithm for multiplying the covariance matrix induced by the dynamic linear model with any vector in linear time. Combined with the conjugate gradient (CG) algorithm, this method enables scalable GP computation via the state-space representation. It effectively handles complex dependency structures that are challenging for traditional Kalman filter (KF) and existing GP approximations. In problems such as particle interactions, where a single observation depends on multiple latent states, the proposed method significantly reduces computational costs compared to full GP while maintaining predictive accuracy. This method is implemented in the `FastGaSP` R package.

To gain a mechanistic understanding from the same dynamic data, Chapter 3 introduces data-driven statistical tests to identify key features and expand the baseline Vicsek model. The maximum likelihood estimator of this model is derived and implemented for model calibration and simulation. By incorporating features such as the non-Gaussian and anisotropic fluctuations, reduced alignment strength, and restricting interactions to neighboring cells with same velocity direction, the model quantitatively reproduces key system-level parameters, whereas models missing any of the features fail to capture these temporally dependent parameters. Compared to traditional methods that require exhaustive simulations to evaluate all possible combinations of potential features, this method relies on individual statistical tests, substantially improving modeling efficiency.

Chapter 4 proposes a physics-informed machine learning method for the automatic identification of phase behaviors in diblock copolymers. By integrating experimental uncertainties into a KF, this method efficiently denoises raw small-angle X-ray scattering (SAXS) curves. Building on this, we extract physics-informed morphological features (PIMF), such as peak spacing ratio, peak width, and peak curvature, which significantly reduces data dimensionality compared to the full intensity curve. Since these PIMFs are independent of specific chemical identities, our classification model successfully generalizes to novel materials, which outperforms chemistry-dependent models that rely on features like monomer type and molecular weight and approaches that input the entire noisy SAXS curves. This method provides a practical and generalizable computational tool for high-throughput material characterization.

Chapter 5 develops active learning with error control (ALEC), a framework that leverages the uncertainty assessment inherent in a partial parallel GP (PP-GP) to achieve adaptive sampling for approximating computationally expensive functionals. The proposed method is fully automatic and provides probabilistic interpretability. In classical density

functional theory calculations, ALEC demonstrates superior predictive accuracy compared to both random sampling and active learning strategies based on an established criterion. Regarding computational cost, ALEC requires only a small number of calls to the numerical solver while controlling the predictive error. Therefore, the ALEC emulator can serve as a reliable building block for solving challenging simulation tasks.

6.1 Extensions of the inverse Kalman filter

The IKF method developed in Chapter 2 can be extended to a broader range of scenarios. First, the fast computation of large matrix-vector products serves as a general computational building block with potential extensions to various large-scale models, including certain deep learning architectures. Second, while IKF achieves linear-time computation for covariance matrix-vector products, full maximum likelihood estimation of parameters also requires computing the log-determinant of the covariance matrix. Developing scalable methods for computing log-determinant without approximation would enable IKF to be used for estimating the parmaters from the likelihood without approximation.

Furthermore, the current IKF method is applicable to one-dimensional inputs, such as the distance between particles. For higher-dimensional inputs, the framework could be extended to separable kernels, where the corresponding covariance matrix is expressed as the Hadamard product of covariance matrices from each dimension. Achieving efficient computation with this structure is a direction worth exploring. Finally, while the integration of IKF with CG performs well in practice, further research could focus on improving efficiency for ill-conditioned problems through tailored preconditioning strategies. Beyond iterative methods, a significant future direction is to investigate direct matrix inversion approaches. Specifically, exploring methods to compute the predictive distribution directly without

relying on iterative solvers like CG could further enhance the robustness and computational speed of the framework.

6.2 Future work on active learning

The ALEC framework in Chapter 5 focuses primarily on online active learning, where the next evaluation point is selected during the prediction process. This error control criterion can be naturally extended to offline design, where the design points are selected prior to prediction, or to batch selection problems, where a set of input points is chosen simultaneously.

In addition, while the ALEC emulator handles functional inputs, prespecifying a suitable kernel for high-dimensional inputs after discretization is challenging. Chapter 4 demonstrated the value of dimension reduction through physics-informed feature extraction. However, since many problems lack clear physical guidance, data-driven dimension reduction remains a crucial research direction. In many applications, the effective dimension of the input space is often much lower than the nominal dimension p. Techniques such as PCA or active subspace methods [219], which identify important input directions by analyzing the gradients of outputs with respect to inputs, could be integrated into the ALEC framework to reduce the problem's dimensionality.

Moreover, the current ALEC framework is based on standard GPs, which encounter computational bottlenecks as the sample size grows. Combining ALEC with scalable GP approximation methods is a vital extension, and the Vecchia approximation is particularly promising, as discussed in Section 6.2.1.

6.2.1 Extensions of the Vecchia approximations in high dimensions

A common approach to scaling GPs to large datasets is through approximation methods, such as the Vecchia approximation [20]. This framework decomposes the joint density of a GP into a product of conditional densities and approximates it by restricting each conditioning set to a small subset $\mathcal{C}(i) \subset \{1, \ldots, i-1\}$ of size at most m_v:

$$f(\mathbf{y}) = \prod_{i=1}^{n} f(y_i \mid \mathbf{y}_{1:i-1}) \approx \hat{f}(\mathbf{y}) = \prod_{i=1}^{n} f(y_i \mid \mathbf{y}_{\mathcal{C}(i)}), \tag{6.1}$$

where the index i follows a pre-specified ordering of all samples, and $\mathcal{C}(i)$ typically consists of the m_v nearest neighbors of y_i among the previously ordered points. A popular ordering choice is maximin ordering, which sequentially places each point as far as possible from all preceding ones, and has been shown to perform well compared to alternative orderings in low-dimensional settings [220].

For prediction, let $\mathbf{z}^*$ denote the latent function values at n^* test locations. The Vecchia approximation is applied to the joint vector $[\mathbf{y}, \mathbf{z}^*]$ by ordering training and test points together. Under a response-first ordering, where all training observations are placed before test points, the approximated joint distribution $\hat{f}(\mathbf{y}, \mathbf{z}^*)$ is multivariate normal with sparse precision matrix $\mathbf{Q} = \mathbf{U}\mathbf{U}^T$. Here, $\mathbf{U}$ is a sparse upper triangular matrix with at most m_v nonzero entries per column. This ordering, adopted by the `GpGp` package [221], enables the predictive mean and variance of $\mathbf{z}^*$ to be computed using only the last n^* columns of $\mathbf{U}$. The resulting computational cost is $\mathcal{O}(n^* m_v^2 (p + m_v))$, where p is the input dimension.

The scaled Vecchia approximation [21] extends this framework to high-dimensional inputs by treating the range parameter in each input dimension as a separate scaling factor. In this approach, ordering and neighbor selection are performed in the scaled input space. Dimensions with weak influence on the response receive large scaling factors, effectively

minimizing their contribution to neighbor selection. As a scaling factor diverges, the corresponding dimension is effectively eliminated, providing an implicit form of dimension reduction. More recently, the Vecchia parallel partial emulator (VPPE) [222] combines this scaled Vecchia with the PP-GP framework [47]. By sharing the covariance parameter and conditioning set across output dimensions, VPPE handles simulators with both large sample sizes and high-dimensional outputs.

Several directions can be explored to extend the Vecchia framework to higher input dimensions. A major bottleneck in high-dimensional settings is nearest neighbor search: as the input dimension p increases, the computational cost rises substantially, and the traditional concept of "closeness" becomes less meaningful in extremely high dimensions. Developing fast nearest neighbor search methods for large p is therefore crucial, and approximate nearest neighbor (ANN) search offers a promising path. Traditional tools like k-d trees [223], used in the `GpGp` package, become inefficient in high dimensions, but recent advances in ANN provide more scalable alternatives. For instance, graph-based methods such as hierarchical navigable small world (HNSW) graphs [224] achieve logarithmic search complexity through a hierarchical multi-layer structure, while quantization-based methods such as RaBitQ [225, 226] approximate high-dimensional vectors with compact discrete codes to enable fast distance estimation. These methods are being actively developed in large-scale retrieval systems and large language model inference [227, 228], but their application within the Vecchia approximation framework remains an open area for research.

The choice of ordering strategy is also an important factor in high-dimensional settings. While maximin ordering is effective in low dimensions, its advantages tend to diminish as p increases. In such cases, middle-out ordering, which sorts points based on their distance to a reference location, has been shown to be more suitable [220]. Determining how to select a suitable reference location for middle-out ordering is an open research question.

Combining the Vecchia approximation with active learning introduces an additional challenge: incorporating new training points requires updating both the ordering and nearest neighbor structure. To maintain efficiency, it is essential to develop incremental update methods rather than recomputing the entire structure from scratch with each new observation. Exploring data structures that facilitate efficient updates while maintaining low overall cost is central to achieving scalable error-controlled active learning with GP approximation methods.

Appendix A

Appendix of Chapter 1

A.1 Derivation of the predictive t-distribution

Here we present the derivation of the predictive t-distribution of $z(\mathbf{x}^*)$ for a new input $\mathbf{x}^*$ in Equation (1.11). In this derivation, we treat the range parameter $\boldsymbol{\gamma}$ as fixed at some estimate $\hat{\boldsymbol{\gamma}}$ (e.g., for the mode of the maximum marginal likelihood estimator, $\hat{\boldsymbol{\gamma}} = \hat{\boldsymbol{\gamma}}_{MMPE}$). Conditional on $(\mathbf{z}, \boldsymbol{\beta}, \sigma^2, \hat{\boldsymbol{\gamma}})$, we have

$$z(\mathbf{x}^*) \mid \mathbf{z}, \boldsymbol{\beta}, \sigma^2, \hat{\boldsymbol{\gamma}} \sim \mathcal{N}(\hat{\mu}(\mathbf{x}^*), \sigma^2 c^*), \tag{A.1}$$

where $\hat{\mu}(\mathbf{x}^*) = \mathbf{h}(\mathbf{x}^*)\boldsymbol{\beta} + \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\boldsymbol{\beta})$ and $c^* = c(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{r}(\mathbf{x}^*)$.

Denote the precision parameter $\phi = 1/\sigma^2$ and the reference prior becomes $\pi(\boldsymbol{\beta}, \phi) \propto$

$1/\phi$. The posterior distribution of mean and variance parameters $(\boldsymbol{\beta}, \phi)$ follows

$$
\begin{aligned}
p(\boldsymbol{\beta}, \phi \mid \boldsymbol{\rho}, \hat{\boldsymbol{\gamma}}) \propto & p(\boldsymbol{\beta}, \phi) p(\mathbf{z}, \hat{\boldsymbol{\gamma}} \mid \boldsymbol{\beta}, \phi) \\
& \propto \frac{1}{\phi} \phi^{n/2} \exp\left(-\frac{\phi}{2}(\mathbf{z}-\mathbf{H}\boldsymbol{\beta})^T \mathbf{R}^{-1}(\mathbf{z}-\mathbf{H}\boldsymbol{\beta})\right) \\
& \propto \phi^{n/2-1} \exp\left(-\frac{\phi}{2}(\mathbf{z}^T\mathbf{R}^{-1}\mathbf{z} - 2\mathbf{z}^T\mathbf{R}^{-1}\mathbf{H}\boldsymbol{\beta} + \boldsymbol{\beta}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\boldsymbol{\beta})\right) \\
& \propto \underbrace{\phi^{p_\mu/2} \exp\left(-\frac{\phi}{2}(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})^T(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})(\boldsymbol{\beta}-\hat{\boldsymbol{\beta}})\right)}_{p(\boldsymbol{\beta}|\mathbf{z},\phi,\hat{\boldsymbol{\gamma}})} \times \\
& \qquad \underbrace{\phi^{n/2-p_\mu/2-1} \exp\left(-\frac{\phi}{2}(\mathbf{z}-\mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{H}\hat{\boldsymbol{\beta}})\right)}_{p(\phi|\mathbf{z},\hat{\boldsymbol{\gamma}})},
\end{aligned}
$$

where $\hat{\boldsymbol{\beta}} = (\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z}$ and the last step follows from adding and subtracting $\hat{\boldsymbol{\beta}}^T\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\hat{\boldsymbol{\beta}}$ in the exponent. Thus, $(\boldsymbol{\beta} \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}})$ and $(\phi \mid \mathbf{z}, \hat{\boldsymbol{\gamma}})$ are distributed as a multivariate normal distribution and a gamma distribution, respectively,

$$
\boldsymbol{\beta} \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}} \sim \mathcal{MN}\left(\hat{\boldsymbol{\beta}}, (\phi\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\right), \tag{A.2}
$$

$$
\phi \mid \mathbf{z}, \hat{\boldsymbol{\gamma}} \sim \text{Gamma}\left(\frac{n-p_\mu}{2}, \frac{(\mathbf{z}-\mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z}-\mathbf{H}\hat{\boldsymbol{\beta}})}{2}\right). \tag{A.3}
$$

Marginalizing out $\boldsymbol{\beta}$. Since both Equations (A.1) and (A.2) are normal distributions, we have

$$
z(\mathbf{x}^*) \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}} \sim \mathcal{N}(\hat{z}(\mathbf{x}^*), \phi^{-1}c^{**}(\mathbf{x}^*, \mathbf{x}^*)), \tag{A.4}
$$

where the mean and variance can be obtained by the law of total expectation and variance:

$$
\begin{aligned}
\mathbb{E}[z(\mathbf{x}^*) \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}] &= \mathbb{E}[\mathbb{E}[z(\mathbf{x}^*) \mid \mathbf{z}, \boldsymbol{\beta}, \phi, \hat{\boldsymbol{\gamma}}]] \\
&= \mathbb{E}\Big[\mathbf{h}(\mathbf{x}^*)\boldsymbol{\beta} + \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\boldsymbol{\beta}) \,\Big|\, \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}\Big] \\
&= \mathbf{h}(\mathbf{x}^*)\hat{\boldsymbol{\beta}} + \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}}) \coloneqq \hat{z}(\mathbf{x}^*), \\
\mathrm{Var}[z(\mathbf{x}^*) \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}] &= \mathbb{E}[\mathrm{Var}[z(\mathbf{x}^*) \mid \mathbf{z}, \boldsymbol{\beta}, \phi, \hat{\boldsymbol{\gamma}}]] + \mathrm{Var}[\mathbb{E}[z(\mathbf{x}^*) \mid \mathbf{z}, \boldsymbol{\beta}, \phi, \hat{\boldsymbol{\gamma}}]] \\
&= \mathbb{E}[\phi^{-1}c^* \mid \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}] + \mathrm{Var}\Big[\mathbf{h}(\mathbf{x}^*)\boldsymbol{\beta} + \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\boldsymbol{\beta}) \,\Big|\, \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}\Big] \\
&= \phi^{-1}c^* + \mathrm{Var}\Big[\mathbf{h}(\mathbf{x}^*)\boldsymbol{\beta} - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\boldsymbol{\beta} \,\Big|\, \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}\Big] \\
&= \phi^{-1}c^* + \mathrm{Var}\Big[\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)\boldsymbol{\beta} \,\Big|\, \mathbf{z}, \phi, \hat{\boldsymbol{\gamma}}\Big] \\
&= \phi^{-1}c^* \\
&\quad + \left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)(\phi\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)^T \\
&= \phi^{-1}\Big[c^* \\
&\quad + \left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})^{-1}\left(\mathbf{h}(\mathbf{x}^*) - \mathbf{r}(\mathbf{x}^*)^T\mathbf{R}^{-1}\mathbf{H}\right)^T\Big] \\
&\coloneqq \phi^{-1}c^{**}.
\end{aligned}
$$

Marginalizing out ϕ. Now ϕ follows the Gamma distribution in (A.3). Define $\hat{\sigma}^2 = (n - p_\mu)^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})^T\mathbf{R}^{-1}(\mathbf{z} - \mathbf{H}\hat{\boldsymbol{\beta}})$, we obtain Equation (1.11) by applying lemma A.1 with $D = 1, m = \hat{z}(\mathbf{x}^*), \Sigma = c^{**}(\mathbf{x}^*, \mathbf{x}^*)$, and $\nu = n - p_\mu$.

Lemma A.1. *Let* $\mathbf{g} \mid \phi \sim \mathcal{MN}(\mathbf{m}, \boldsymbol{\Sigma}/\phi)$ *(D-variate normal) and* $\phi \sim$ *Gamma*$((n - p_\mu)/2, (n - p_\mu)\hat{\sigma}^2/2)$. *Then* $\mathbf{g}$ *has a D dimensional multivariate t distribution*

$$\mathbf{g} \sim \mathbf{t}(\mathbf{m}, \hat{\sigma}^2\boldsymbol{\Sigma}, n - p_\mu),$$

with density

$$p(\mathbf{g}) \propto \left[1 + \frac{1}{n-p_\mu}\frac{(\mathbf{g}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{g}-\mathbf{m})}{\hat{\sigma}^2}\right]^{-\frac{D+n-p_\mu}{2}}.$$

Proof. The joint density of $\mathbf{g}$ and ϕ is

$$\begin{aligned} p(\mathbf{g},\phi) &\propto \phi^{D/2}\exp\left(-\frac{\phi}{2}(\mathbf{g}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{g}-\mathbf{m})\right) \times \phi^{(n-p_\mu)/2-1}\exp\left(-(n-p_\mu)\hat{\sigma}^2\phi/2\right) \\ &\propto \phi^{(D+n-p_\mu)/2-1}\exp\left(-\frac{\phi}{2}\left((\mathbf{g}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{g}-\mathbf{m}) + (n-p_\mu)\hat{\sigma}^2\right)\right), \end{aligned}$$

which is a Gamma$((D+n-p_\mu)/2, ((\mathbf{g}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{g}-\mathbf{m}) + (n-p_\mu)\hat{\sigma}^2)/2$. Then the distribution of $\mathbf{g}$ follows from marginalizing out ϕ:

$$\begin{aligned} p(\mathbf{g}) &\propto \int p(\mathbf{g},\phi)d\phi \\ &\propto ((\mathbf{g}-\mathbf{m})^T\mathbf{\Sigma}^{-1}(\mathbf{g}-\mathbf{m}) + (n-p_\mu)\hat{\sigma}^2)^{-(n-p_\mu+D)/2} \\ &\propto (1 + (\mathbf{g}-\mathbf{m})^T(\hat{\sigma}^2\mathbf{\Sigma})^{-1}(\mathbf{g}-\mathbf{m})/(n-p_\mu))^{-(n-p_\mu+D)/2}, \end{aligned}$$

which gives the D dimensional multivariate t distribution $\mathbf{t}(\mathbf{m}, \hat{\sigma}^2\Sigma, n-p_\mu)$. □

A.2 Proof

Proof of Lemma 1.3. From (1.21) in step (ii) of the Kalman filter in Lemma 1.1, we have

$$p(\mathbf{y}_{1:n}) = \prod_{t=1}^{n}\left\{(2\pi Q_t)^{-\frac{1}{2}}\exp\left(-\frac{(y_t-f_t)^2}{2Q_t}\right)\right\}. \tag{A.5}$$

Additionally, since $\mathbf{y}_{1:n}$ follows a zero-mean multivariate normal distribution, we have

$$p(\mathbf{y}_{1:n}) = (2\pi)^{-\frac{n}{2}}|\mathbf{\Sigma}|^{-\frac{1}{2}}\exp\left(-\frac{1}{2}(\mathbf{L}^{-1}\mathbf{y})^T(\mathbf{L}^{-1}\mathbf{y})\right), \tag{A.6}$$

where $\mathbf{L}$ is the Cholesky factor of $\mathbf{\Sigma}$, a lower triangular matrix with positive diagonal entries. By equating (A.5) and (A.6), we have

$$\sum_{t=1}^{n} \frac{(y_t - f_t)^2}{Q_t} = (\mathbf{L}^{-1}\mathbf{y})^T(\mathbf{L}^{-1}\mathbf{y}) = \sum_{t=1}^{n} \tilde{y}_t^2, \tag{A.7}$$

$$\prod_{t=1}^{n} Q_t^{\frac{1}{2}} = |\mathbf{\Sigma}|^{\frac{1}{2}} = |\mathbf{L}| = \prod_{t=1}^{n} L_{t,t}. \tag{A.8}$$

Since the Kalman filter operates incrementally and applies to any number of observations, these equalities hold not only for the entire set of observations up to n but also for any subset up to $t' \leq n$. This means the expressions in (A.7) and (A.8) are valid for sums and products up to any $t' = 1, ..., n$. We will use this property to prove both statements of the lemma by mathematical induction.

First, we prove $\tilde{y}_t = (y_t - f_t)/Q_t^{1/2}$ for $t = 1, ..., n$. When we consider only the first observation, i.e., $t' = 1$, the matrix $\mathbf{\Sigma}$ reduces to the scalar Q_1, and the statement directly follows from the definition of the Cholesky decomposition which has positive diagonals and $f_1 = 0$. For the first $t' - 1$ observations, we have $\sum_{t=1}^{t'-1}(y_t - f_t)^2/Q_t = \sum_{t=1}^{t'-1} \tilde{y}_t^2$. Including the t'-th observation, we get $\sum_{t=1}^{t'}(y_t - f_t)^2/Q_t = \sum_{t=1}^{t'} \tilde{y}_t^2$. Subtracting the sum of the first $t' - 1$ observations, we have $(y_{t'} - f_{t'})^2/Q_{t'} = \tilde{y}_{t'}^2$, which gives

$$\tilde{y}_{t'} = \frac{y_{t'} - f_{t'}}{Q_{t'}^{1/2}} \qquad \text{or} \qquad \tilde{y}_{t'} = -\frac{(y_{t'} - f_{t'})}{Q_{t'}^{1/2}}, \tag{A.9}$$

where $f_{t'}$ is not related to the observation $y_{t'}$ based on the Kalman filter. Let $\tilde{L}_{t',t}$ denote the (t', t)-th element of $\mathbf{L}^{-1}$. We can write

$$\tilde{y}_{t'} = \sum_{t=1}^{t'} \tilde{L}_{t',t} y_t. \tag{A.10}$$

By equating the coefficient of $y_{t'}$ in (A.9) and (A.10), we obtain $\tilde{L}_{t',t'} = Q_{t'}^{-1/2}$ or $\tilde{L}_{t',t'} = -Q_{t'}^{-1/2}$. As $\mathbf{L}^{-1}\mathbf{L} = \mathbf{I}_n$ and $\mathbf{L}$ is a lower triangular matrix with the diagonal value $L_{t',t'} > 0$ for $t' = 1, ..., n$, then $\tilde{L}_{t',t'} = 1/L_{t',t'} > 0$. The only solution is $\tilde{y}_{t'} = (y_{t'} - f_{t'})/Q_{t'}^{1/2}$.

The second statement, $L_{t,t} = Q_t^{1/2}$, follows the similar logic with (A.8) by mathematical induction. The detailed proof is omitted here for brevity. □

A.3 Gaussian processes with half-integer Matérn covariances as dynamic linear models

In this section, we provide the connection between Gaussian processes (GPs) having the Matérn covariance with roughness parameters $\nu = 1/2$ and $\nu = 5/2$ and dynamic linear models (DLMs). Let $y_t = y(x_t)$ for $t = 1, ..., n$, modeled as

$$y(x_t) = z(x_t) + \epsilon_t,$$

with $z(\cdot)$ follows a zero-mean GP with covariance function $c(\cdot, \cdot)$ and range parameter γ, and ϵ_t represents independent Gaussian noise with variance $\sigma_{0,t}^2$.

For a GP having Matérn covariance function with roughness parameter $\nu = 5/2$ in (1.2), the latent stochastic process $z(\cdot)$ can be expressed via stochastic differential equations [25]:

$$\dot{\boldsymbol{\theta}}(x) = \mathbf{J}\boldsymbol{\theta}(x) + \tilde{\mathbf{F}}u(x), \tag{A.11}$$

or equivalently in matrix form

$$\frac{\partial}{\partial x}\begin{pmatrix} z(x) \\ z^{(1)}(x) \\ z^{(2)}(x) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\lambda^3 & -3\lambda^2 & -3\lambda \end{pmatrix}\begin{pmatrix} z(x) \\ z^{(1)}(x) \\ z^{(2)}(x) \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} u(x),$$

where ∂ denotes the differentiation, $z^{(p)}(\cdot)$ denotes the p-th derivative of the process $z(\cdot)$, and $u(x) \sim \mathcal{N}(0, \sigma^2)$ follows Gaussian white noise with variance σ^2, $\lambda = \sqrt{2\nu}/\gamma$.

For any x_t with $t = 1, ..., n$, the solution of (A.11) can be represented as a DLM:

$$y(x_t) = \mathbf{F}\boldsymbol{\theta}(x_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2_{0,t}) \tag{A.12}$$

$$\boldsymbol{\theta}(x_t) = \mathbf{G}(x_t)\boldsymbol{\theta}(x_{t-1}) + \mathbf{w}(x_t),\ \mathbf{w}(x_t) \sim \mathcal{MN}(\mathbf{0}, \mathbf{W}(x_t)), \tag{A.13}$$

where $\mathbf{G}(x_t) = e^{\mathbf{J}(x_t - x_{t-1})}$, $\mathbf{W}(x_t) = \int_0^{x_t - x_{t-1}} e^{\mathbf{J}s}\tilde{\mathbf{F}}c\tilde{\mathbf{F}}^T e^{\mathbf{J}^T s} ds$ for $t = 2, \ldots, n$, $\mathbf{F} = [1, 0, 0]$, and the stationary distribution is $\boldsymbol{\theta}(x_t) \sim \mathcal{MN}(\mathbf{0}, \mathbf{W}(x_1))$ with $\mathbf{W}(x_1) = \int_0^\infty e^{\mathbf{J}s}\tilde{\mathbf{F}}c\tilde{\mathbf{F}}^T e^{\mathbf{J}^T s} ds$ and $c = 16\sigma^2\lambda^5/3$.

Write $\mathbf{G}(x_t) = \mathbf{G}_t$, $\mathbf{W}(x_t) = \mathbf{W}_t$ and $\boldsymbol{\theta}(x_t) = \boldsymbol{\theta}_t$, the joint distribution of $(\boldsymbol{\theta}_1^T, \ldots, \boldsymbol{\theta}_n^T)^T$ follows a multivariate normal distribution with the covariance being the inverse of a block tri-diagonal matrix

$$\begin{pmatrix} \boldsymbol{\theta}_1 \\ \boldsymbol{\theta}_2 \\ \boldsymbol{\theta}_3 \\ \vdots \\ \boldsymbol{\theta}_n \end{pmatrix} \sim \mathcal{MN}\left(\mathbf{0}, \begin{pmatrix} \boldsymbol{W}_1^{-1} + \boldsymbol{G}_2^T\boldsymbol{W}_2^{-1}\boldsymbol{G}_2 & -\boldsymbol{G}_2^T\boldsymbol{W}_2^{-1} & & & \\ -\boldsymbol{W}_2^{-1}\boldsymbol{G}_2 & \boldsymbol{W}_2^{-1} + \boldsymbol{G}_3^T\boldsymbol{W}_3^{-1}\boldsymbol{G}_3 & & & \\ & \ddots & \ddots & \ddots & \\ & & & \boldsymbol{W}_{n-1}^{-1} + \boldsymbol{G}_n^T\boldsymbol{W}_n^{-1}\boldsymbol{G}_n & -\boldsymbol{G}_{n-1}^T\boldsymbol{W}_{n-1}^{-1} \\ & & & -\boldsymbol{W}_{n-1}^{-1}\boldsymbol{G}_{n-1} & \boldsymbol{W}_n^{-1} \end{pmatrix}^{-1}\right), \tag{A.14}$$

where all $\mathbf{G}_t$ and $\mathbf{W}_t$ have closed-form expressions as a function of $\Delta_{x_t} = |x_t - x_{t-1}|$ and

$\lambda = \sqrt{2\nu}/\gamma$:

$$\mathbf{G}_t = e^{\mathbf{J}\Delta_{x_t}} = \frac{e^{-\lambda\Delta_{x_t}}}{2}\begin{pmatrix} \lambda^2\Delta_{x_t}^2 + 2\lambda + 2 & 2(\lambda\Delta_{x_t}^2 + \Delta_{x_t}) & \Delta_{x_t}^2 \\ -\lambda^3\Delta_{x_t}^2 & -2(\lambda^2\Delta_{x_t}^2 - \lambda\Delta_{x_t} - 1) & 2\Delta_{x_t} - \lambda\Delta_{x_t}^2 \\ \lambda^4\Delta_{x_t}^2 - 2\lambda^3\Delta_{x_t} & 2(\lambda^3\Delta_{x_t}^2 - 3\lambda^2\Delta_{x_t}) & \lambda^2\Delta_{x_t}^2 - 4\lambda\Delta_{x_t} + 2 \end{pmatrix}$$

$$\mathbf{W}(x_t) = \frac{4\sigma^2\lambda^5}{3}\begin{pmatrix} W_{1,1}(x_t) & W_{1,2}(x_t) & W_{1,3}(x_t) \\ W_{2,1}(x_t) & W_{2,2}(x_t) & W_{2,3}(x_t) \\ W_{3,1}(x_t) & W_{3,2}(x_t) & W_{3,3}(x_t) \end{pmatrix},$$

$$\begin{aligned}
W_{1,1}(x_t) &= \frac{e^{-2\lambda\Delta_{x_t}}(3 + 6\lambda\Delta_{x_t} + 6\lambda^2\Delta_{x_t}^2 + 4\lambda^3\Delta_{x_t}^3 + 2\lambda^4\Delta_{x_t}^4) - 3}{-4\lambda^5}, \\
W_{1,2}(x_t) &= W_{2,1}(x_t) = \frac{e^{-2\lambda\Delta_{x_t}}\Delta_{x_t}^4}{2}, \\
W_{1,3}(x_t) &= W_{3,1}(x_t) = \frac{e^{-2\lambda\Delta_{x_t}}(1 + 2\lambda\Delta_{x_t} + 2\lambda^2\Delta_{x_t}^2 + 4\lambda^3\Delta_{x_t}^3 - 2\lambda^4\Delta_{x_t}^4) - 1}{4\lambda^3}, \\
W_{2,2}(x_t) &= \frac{e^{-2\lambda\Delta_{x_t}}(1 + 2\lambda\Delta_{x_t} + 2\lambda^2\Delta_{x_t}^2 - 4\lambda^3\Delta_{x_t}^3 + 2\lambda^4\Delta_{x_t}^4) - 1}{-4\lambda^3}, \\
W_{2,3}(x_t) &= W_{3,2}(x_t) = \frac{e^{-2\lambda\Delta_{x_t}}\Delta_{x_t}^2(4 - 4\lambda\Delta_{x_t} + \lambda^2\Delta_{x_t}^2)}{2}, \\
W_{3,3}(x_t) &= \frac{e^{-2\lambda\Delta_{x_t}}(-3 + 10\lambda\Delta_{x_t} - 22\lambda^2\Delta_{x_t}^2 + 12\lambda^3\Delta_{x_t}^3 - 2\lambda^4\Delta_{x_t}^4) + 3}{4\lambda},
\end{aligned}$$

and

$$\mathbf{W}(x_1) = \begin{pmatrix} \sigma^2 & 0 & -\sigma^2\lambda^2/3 \\ 0 & \sigma^2\lambda^2/3 & 0 \\ -\sigma^2\lambda^2/3 & 0 & \sigma^2\lambda^4 \end{pmatrix}.$$

For GPs with exponential covariance functions or, equivalently, the Matérn covariance functions with the roughness parameter being $\nu = 1/2$, the corresponding DLM representation follows (A.12)-(A.13) with $F_t = 1$, $G_t = \exp(-|x_t - x_{t-1}|/\gamma)$ and $W_t = \sigma^2(1 - G_t)$ for

$t = 2, ..., n$. The stationary distribution $z(x_t) \sim \mathcal{N}(0, \sigma^2)$. The joint distribution $(\theta_1, ..., \theta_n)^T$ follows (A.14), where the inverse covariance matrix is a tri-diagonal matrix. All results shown above are coded in the `FastGaSP` package on CRAN [73].

Appendix B

Appendix of Chapter 2

B.1 Proofs

In the following proofs, we first prove Lemma 2.2, which, along with Lemma 1.3, is essential for proving Lemma 2.3. The outcome of Lemma 2.3 is then used to prove Lemma 2.1. The proof of Lemma 2.4 is presented at the end.

Proof of Lemma 2.2. For any $t = 1, ..., N$, we invert (1.24) in Kalman filter (KF) to compute $x_t = f_t + Q_t^{1/2}\tilde{x}_t$ with $f_t = \mathbf{F}_t\mathbf{b}_t$, which leads to (2.6). The other two steps in (2.5)-(2.7) are from the KF with a known observation x_t. By Lemma 1.3, for any N-vector $\mathbf{x}$, the second step of KF produces $\tilde{\mathbf{x}} = \mathbf{L}^{-1}\mathbf{x}$. As the steps of KF are reversed in Lemma 2.2, we have $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$ for any N-vector $\tilde{\mathbf{x}}$. □

Proof of Lemma 2.3. Denote $\mathbf{x} = \mathbf{L}\tilde{\mathbf{x}}$. The t'-th entry $x_{t'}$ is

$$x_{t'} = \sum_{t=1}^{t'} L_{t',t}\tilde{x}_t. \tag{B.1}$$

We will prove (2.8) with $\ell_{t,t'}$ defined in (2.9) using the mathematical induction. By Lemma 1.3, we have $L_{t',t} = Q_t^{1/2}$ for $t' = t$, so $L_{1,1} = Q_1^{1/2}$. First, we show that $L_{t',t'-1} = Q_{t'-1}^{1/2}\ell_{t',t'-1}$.

For $x_{t'}$, we iterate through equations in Lemma 2.2 to get

$$x_{t'} = \mathbf{F}_{t'}\mathbf{G}_{t'}(\mathbf{b}_{t'-1} + \mathbf{K}_{t'-1}Q_{t'-1}^{\frac{1}{2}}\tilde{x}_{t'-1}) + Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'}. \tag{B.2}$$

Note that $\mathbf{b}_{t'-1}$ does not depend on $\tilde{x}_{t'-1}$ or $\tilde{x}_{t'}$. From (B.1) and Lemma 1.3, we also have

$$x_{t'} = \sum_{t=1}^{t'-2} L_{t',t}\tilde{x}_t + L_{t',t'-1}\tilde{x}_{t'-1} + Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'}. \tag{B.3}$$

By equating (B.2) and (B.3), we have that for $t' = t+1$, $L_{t',t} = L_{t',t'-1} = \mathbf{F}_{t'}\mathbf{G}_{t'}\mathbf{K}_{t'-1}Q_{t'-1}^{1/2} = Q_t^{1/2}\ell_{t',t}$, where $\ell_{t',t} = \mathbf{F}_{t'}\mathbf{G}_{t'}\mathbf{K}_{t'-1}$.

Now, assume that for any $t' \geq t+1$, $L_{t',t} = Q_t^{1/2}\ell_{t',t} = Q_t^{1/2}\mathbf{F}_{t'}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t$. We will show that this result holds for $L_{t'+1,t}$. From (B.1), Lemma 1.3 and plugging $L_{t',t} = Q_t^{1/2}\ell_{t',t} = Q_t^{1/2}\mathbf{F}_{t'}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t$ for $t' \geq t+1$, we have

$$x_{t'} = \mathbf{F}_{t'}\sum_{t=1}^{t'-1} Q_t^{\frac{1}{2}}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t + Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'}. \tag{B.4}$$

Compare (B.4) with (2.6) in Lemma 2.2, we obtain $\mathbf{b}_{t'} = \sum_{t=1}^{t'-1} Q_t^{1/2}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t$. Then iterate through Lemma 2.2, we derive

$$\begin{aligned}
\mathbf{m}_{t'} &= \sum_{t=1}^{t'-1} Q_t^{\frac{1}{2}}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t + \mathbf{K}_{t'}Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'},\\
\mathbf{b}_{t'+1} &= \sum_{t=1}^{t'-1} Q_t^{\frac{1}{2}}\left(\prod_{l=t+1}^{t'+1}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t + \mathbf{G}_{t'+1}\mathbf{K}_{t'}Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'},\\
x_{t'+1} &= \mathbf{F}_{t'+1}\left(\sum_{t=1}^{t'-1} Q_t^{\frac{1}{2}}\left(\prod_{l=t+1}^{t'+1}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t + \mathbf{G}_{t'+1}\mathbf{K}_{t'}Q_{t'}^{\frac{1}{2}}\tilde{x}_{t'}\right) + Q_{t'+1}^{\frac{1}{2}}\tilde{x}_{t'+1},\\
&= \mathbf{F}_{t'+1}\left(\sum_{t=1}^{t'} Q_t^{\frac{1}{2}}\left(\prod_{l=t+1}^{t'+1}\mathbf{G}_l\right)\mathbf{K}_t\tilde{x}_t\right) + Q_{t'+1}^{\frac{1}{2}}\tilde{x}_{t'+1}.
\end{aligned} \tag{B.5}$$

By equating (B.5) with (B.1) for $x_{t'+1}$: $x_{t'+1} = \sum_{t=1}^{t'} L_{t'+1,t}\tilde{x}_t + L_{t'+1,t'+1}\tilde{x}_{t'+1}$, we get

$L_{t'+1,t} = Q_t^{1/2}\mathbf{F}_{t'+1}\left(\prod_{l=t+1}^{t'+1}\mathbf{G}_l\right)\mathbf{K}_t = Q_t^{1/2}\ell_{t'+1,t}$, thus completing the proof. □

Proof of Lemma 2.1. Denote $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$. The N-th and $(N-1)$-th entries of $\tilde{\mathbf{x}}$ directly follow from Lemma 2.3. For $t = N-2, \ldots, 1$, we will prove by induction. For the t-th entry x_t, by Lemma 2.3, we have

$$\tilde{x}_t = \sum_{t'=t}^{N} L_{t',t}u_{t'} = Q_t^{\frac{1}{2}}\left(\sum_{t'=t+2}^{N}\ell_{t',t}u_{t'} + \ell_{t+1,t}u_{t+1} + u_t\right).$$

Thus, it suffices to prove $\tilde{\ell}_{t+1,t} = \mathbf{g}_{t+1}\mathbf{G}_{t+1}\mathbf{K}_t = \sum_{t'=t+2}^{N}\ell_{t',t}u_{t'}$. When $t = N-2$,

$$\ell_{N,N-2}u_N = \mathbf{F}_N\mathbf{G}_N\mathbf{G}_{N-1}\mathbf{K}_{N-2}u_N = \mathbf{g}_{N-1}\mathbf{G}_{N-1}\mathbf{K}_{N-2} = \tilde{\ell}_{N-1,N-2}.$$

Assume for any given t with $t \le N-2$, $\mathbf{g}_{t+1}\mathbf{G}_{t+1}\mathbf{K}_t = \sum_{t'=t+2}^{N}\ell_{t',t}u_{t'}$. We will prove the formula holds for $t-1$. From Lemma 2.3, we have

$$\sum_{t'=t+2}^{N}\ell_{t',t}u_{t'} = \sum_{t'=t+2}^{N}\mathbf{F}_{t'}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)\mathbf{K}_t u_{t'} = \left\{\sum_{t'=t+2}^{N}\mathbf{F}_{t'}\left(\prod_{l=t+2}^{t'}\mathbf{G}_l\right)u_{t'}\right\}\mathbf{G}_{t+1}\mathbf{K}_t,$$

where the last equation hols as $u_{t'}$ is a scalar. Thus

$$\mathbf{g}_{t+1} = \sum_{t'=t+2}^{N}\mathbf{F}_{t'}\left(\prod_{l=t+2}^{t'}\mathbf{G}_l\right)u_{t'}.$$

Then for $(t-1)$-th entry, similarly we have

$$\begin{aligned}\sum_{t'=t+1}^{N}\ell_{t',t-1}u_{t'} &= \left\{\sum_{t'=t+1}^{N}\mathbf{F}_{t'}\left(\prod_{l=t+1}^{t'}\mathbf{G}_l\right)u_{t'}\right\}\mathbf{G}_t\mathbf{K}_{t-1}\\ &= \left\{\mathbf{g}_{t+1}\mathbf{G}_{t+1} + \mathbf{F}_{t+1}\mathbf{G}_{t+1}u_{t+1}\right\}\mathbf{G}_t\mathbf{K}_{t-1}\\ &= \mathbf{g}_t\mathbf{G}_t\mathbf{K}_{t-1},\end{aligned}$$

where the last equation follows from (2.3). Therefore, $\mathbf{g}_t\mathbf{G}_t\mathbf{K}_{t-1} = \sum_{t'=t+1}^{N} \ell_{t',t-1}u_{t'}$ and we have proved the results hold for $t-1$ from any given $t \leq N-1$ which concludes the proof. □

Proof of Lemma 2.4. In Lemma 2.1, the iterative algorithm gives $\tilde{\mathbf{x}} = \mathbf{L}^T\mathbf{u}$ for any N-vector $\mathbf{u}$. We invert (2.2) to get $u_t = Q_t^{-1/2}\tilde{x}_t - \tilde{\ell}_{t+1,t} - \ell_{t+1,t}u_{t+1}$ for any t, which leads to (2.10). Since we reverse the steps of the iterative algorithm in Lemma 2.1, we have $\mathbf{u} = (\mathbf{L}^T)^{-1}\tilde{\mathbf{x}}$ for any N-vector $\tilde{\mathbf{x}}$. □

B.2 The conjugate gradient method

The conjugate gradient (CG) algorithm [59] is an iterative method to solve a linear system $\mathbf{\Sigma}_y\hat{\mathbf{u}} = \mathbf{y}$, particularly useful for a sparse matrix $\mathbf{\Sigma}_y$. The key advantage of the CG algorithm is that it avoids the computationally intensive matrix inversion and only requires matrix-vector multiplication, making it a potentially scalable alternative for large-scale applications. The detailed algorithm is summarized in Algorithm B.1.

B.3 Scalable computation of the predictive distribution in particle dynamics

In this section, we detail the scalable computation of the predictive mean and variance for the particle interaction function in (2.15) and (2.16), respectively, applicable to both single and multiple test inputs. We use an example of particles on 2-dimensional spatial coordinates to illustrate the data pre-processing step. The space is divided into $n_1 \times n_2$ grids, where n_1 and n_2 represent the number of grids along the two orthogonal coordinates. Each particle is assigned to a grid, with neighboring particles located within the adjacent nine grids, as depicted in Figure B.1(a). For each grid, we record the positions and

Algorithm B.1 Conjugate gradient algorithm

Input: $\boldsymbol{\Sigma}_y$, $\mathbf{y}$ from the linear system $\boldsymbol{\Sigma}_y\hat{\mathbf{u}} = \mathbf{y}$, maximum iteration i_{max}, and the error tolerance tol.

$i \leftarrow 0$
$\hat{\mathbf{u}} \leftarrow \mathbf{0}$
$\boldsymbol{\varepsilon} \leftarrow \mathbf{y} - \boldsymbol{\Sigma}_y\hat{\mathbf{u}}$
$\mathbf{u} \leftarrow \boldsymbol{\varepsilon}$
$\delta_{old} \leftarrow \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$
$\delta_{new} \leftarrow 1$
while $i < i_{max}$ and $\delta_{new} > tol$ **do**
    $\tilde{\mathbf{u}} \leftarrow \boldsymbol{\Sigma}_y\mathbf{u}$
    $\alpha \leftarrow \delta_{old}/(\mathbf{u}^T\tilde{\mathbf{u}})$
    $\hat{\mathbf{u}} \leftarrow \hat{\mathbf{u}} + \alpha\mathbf{u}$
    $\boldsymbol{\varepsilon} \leftarrow \boldsymbol{\varepsilon} - \alpha\tilde{\mathbf{u}}$
    $\delta_{new} \leftarrow \boldsymbol{\varepsilon}^T\boldsymbol{\varepsilon}$
    $\delta_{ratio} \leftarrow \delta_{new}/\delta_{old}$
    $\mathbf{u} \leftarrow \boldsymbol{\varepsilon} + \delta_{ratio}\mathbf{u}$
    $\delta_{old} \leftarrow \delta_{new}$
    $i \leftarrow i + 1$
end while

Output: $\hat{\mathbf{u}}$.

velocities of particles within the adjacent nine grids. This step is critical as it allows efficient neighbor searching within the adjacent grids during parameter estimation, avoiding the need to loop over all particles. The computational cost of this step is $\mathcal{O}(\sum_{\tau=1}^{n_\tau} n_p(\tau))$, where $n_p(\tau)$ is the number of particles at time τ, and it needs to be performed only once. Following the pre-processing step, constructing neighboring distances for each type of interaction incurs $\mathcal{O}(p_j \sum_{\tau=1}^{n_\tau} n_p(\tau))$ operations, with p_j being the average number of particles in the neighboring nine grids for j-th interaction, typically no larger than 10 in our applications, which is much smaller than $n_p(\tau)$, thereby reducing the cost substantially compared to $\mathcal{O}(\sum_{\tau=1}^{n_\tau}(n_p(\tau))^2)$ without the pre-processing step.

For prediction on either single or multiple test inputs, the primary computational task is solving for $\hat{\mathbf{u}} = \boldsymbol{\Sigma}_y^{-1}\mathbf{y}$, where $\mathbf{y}$ is a vector of $\tilde{N}$ dimensions. This is equivalent to solving a linear system $\boldsymbol{\Sigma}_y\hat{\mathbf{u}} = \mathbf{y}$, which can be computed iteratively by the CG algorithm. Starting with an initial guess $\mathbf{u}_{(0)} = \mathbf{0}$, the CG method iteratively refines the solution

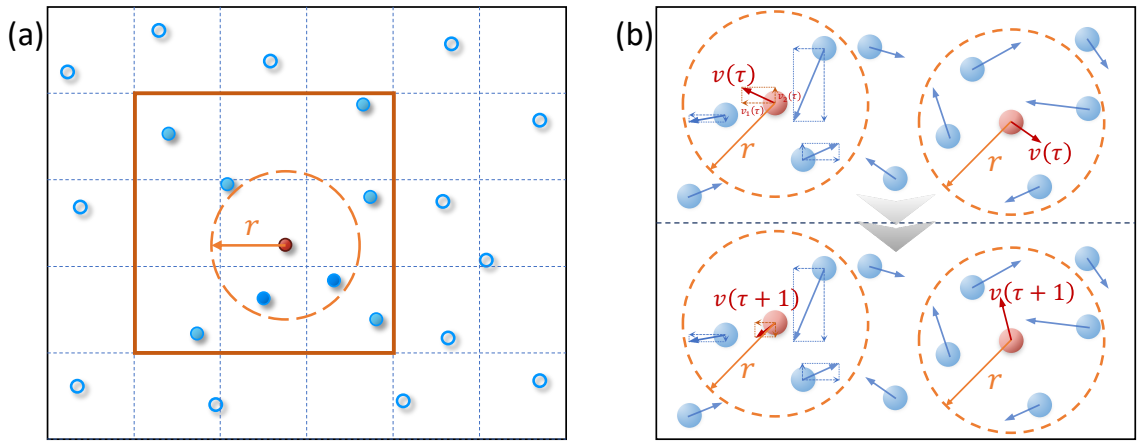

Figure B.1: (a) Illustration for preprocessing the particles into coarse-grained grids. The domain is divided into square grid cells with side lengths equal to or greater than the maximum interaction radius. For a given particle (red dot), as only particles in the current and eight adjacent grid cells (orange square region) need to be considered, it substantially accelerates the computation. Particles (blue dots) within the interaction radius (the dashed circle) and itself (red dot) can be identified as neighbors. (b) Visualization of velocity alignment in the unnormalized Vicsek model. Each red particle updates its velocity $v(\tau) = [v_1(\tau), v_2(\tau)]^T$ by aligning with all neighboring particles, including itself, within interaction radius r. While all particles simultaneously update their directions, this illustration focuses on the movement of the red particle.

$\mathbf{u}_{(k)}$ within the Krylov subspace span$[\mathbf{y}, \mathbf{\Sigma}_y\mathbf{y}, \ldots, \mathbf{\Sigma}_y^{k-1}\mathbf{y}]$ at each step k, minimizing the norm $\|\hat{\mathbf{u}} - \mathbf{u}_{(k)}\|^2_{\mathbf{\Sigma}_y} = (\hat{\mathbf{u}} - \mathbf{u}_{(k)})^T\mathbf{\Sigma}_y(\mathbf{u} - \mathbf{u}_{(k)})$. In each iteration step, we need to compute $\mathbf{\Sigma}_y\mathbf{u} = (\sum_{j=1}^J \mathbf{A}_j\mathbf{\Sigma}_j^{(u)}\mathbf{A}_j^T + \sigma_0^2\mathbf{I}_{\tilde{N}})\mathbf{u}$ for any $\tilde{N}$-dimensional vector $\mathbf{u}$, as detailed in Algorithm B.1 in Section B.2. For each interaction $j, j = 1, \ldots, J$, we decompose the computation of $\mathbf{A}_j\mathbf{\Sigma}_j^{(u)}\mathbf{A}_j^T\mathbf{u}$ into four steps.

In Step 1, we compute $\mathbf{u}_j^{(u)} = \mathbf{A}_j^T\mathbf{u}$, where $\mathbf{A}_j$ is a sparse block diagonal matrix of dimensions $\tilde{N} \times N_j$, with the i-th digonal block being $\mathbf{A}_{i,j} = [\mathbf{a}_{i,j,1}, \ldots, \mathbf{a}_{i,j,p_{i,j}}]$ for $i = 1, \ldots, n$. We stack all non-zero terms in $\mathbf{A}_j$ as a vector $\mathbf{a}_j = [\text{vec}(\mathbf{A}_{1,j})^T, \ldots, \text{vec}(\mathbf{A}_{n,j})^T]^T = [\text{vec}(\mathbf{a}_{1,j})^T, \ldots, \text{vec}(\mathbf{a}_{N_j,j})^T]^T$ of dimension D_yN_j, where vec$(\cdot)$ denotes the vectorization operation and $\mathbf{a}_{t,j} = \mathbf{a}_{i,j,k}$ with $t = \sum_{i'=1}^{i} p_{i',j} + k$ for $i = 1, \ldots, n$ and $k = 1, \ldots, p_{i,j}$, with

$p_{0,j} = 0$. The t-th entry of $\mathbf{u}_j^{(u)}$, denoted as $u_{t,j}^{(u)}$, can be computed by

$$u_{t,j}^{(u)} = \mathbf{a}_{t,j}^T \mathbf{u}_{(t-1)D_y+(1:D_y)},$$

where $\mathbf{u}_{(t-1)D_y+(1:D_y)}$ is a vector containing the $((t-1)D_y+1)$-th to tD_y-th entries of $\mathbf{u}$.

In Step 2, we rearrange $\mathbf{u}_j^{(u)}$ into $\mathbf{u}_j$ based on the sorted order from $\mathbf{d}_j^{(u)} = [d_{1,j}^{(u)}, \ldots, d_{N_j,j}^{(u)}]^T$ to $\mathbf{d}_j = [d_{1,j}, \ldots, d_{N_j,j}]^T$. We obtain the index set $\{g_t\}_{t=1}^{N_j}$, where $d_{g_t,j}^{(u)} = d_{t,j}$. Using this index set, the t-th entry of $\mathbf{u}_j$ is $u_{t,j} = u_{g_t,j}^{(u)}$.

In Step 3, we use the IKF algorithm presented in Algorithm 2.1 to compute $\mathbf{x}_j = \mathbf{\Sigma}_j \mathbf{u}_j$. We then rearrange the entries of $\mathbf{x}_j$ from $\mathbf{x}_j = [x_{1,j}, \ldots, x_{N_j,j}]^T$ to $\mathbf{x}_j^{(u)} = [x_{1,j}^{(u)}, \ldots, x_{N_j,j}^{(u)}]^T$, where $x_{g_t,j}^{(u)} = x_{t,j}$, to preserve the original order of $\mathbf{u}_j$ for subsequent calculations.

In Step 4, we compute $\hat{\mathbf{u}}_j = \mathbf{A}_j \mathbf{x}_j^{(u)}$. We partition the vector $\hat{\mathbf{u}}_j$ into $\hat{\mathbf{u}}_j = [\hat{\mathbf{u}}_{1,j}^T, \ldots, \hat{\mathbf{u}}_{n,j}^T]$, where each sub-vector $\hat{\mathbf{u}}_{i,j}$ has D_y dimensions and can be computed by

$$\hat{\mathbf{u}}_{i,j} = \mathbf{A}_{i,j} \mathbf{x}_{i,j}^{(u)},$$

with $\mathbf{x}_{i,j}^{(u)}$ being the $(\sum_{i'=1}^{i-1} p_{i',j} + (1 : p_{i,j}))$-th entry of the vector $\mathbf{x}_j^{(u)}$ for $i = 1, \ldots, n$.

After completing these four steps for $j = 1, .., J$, we obtain $\mathbf{\Sigma}_y \mathbf{u} = \sum_{j=1}^{J} \hat{\mathbf{u}}_j + \sigma_0^2 \mathbf{u}$. These steps are repeated iteratively in a CG algorithm to compute $\hat{\mathbf{u}} = \mathbf{\Sigma}_y^{-1} \mathbf{y}$.

The output of the CG algorithm, $\hat{\mathbf{u}}$, is used to calculate the predictive mean. We will first explain how to compute the predictive mean and variance for a single test input, and then discuss a scalable approach to compute the predictive mean of a latent interaction function on a large number of inputs. In both scenarios, we first compute the matrix-vector multiplication $\hat{\mathbf{u}}_j = \mathbf{A}_j^T \hat{\mathbf{u}}$ using Step 1 above.

The process of calculating the predictive distribution for a single test input distance d^* is straightforward. Once N_j-vector $\mathbf{\Sigma}_j^{(u)}(d^*)$ is constructed, the predictive mean can be obtained by direct vector-matrix multiplication: $\hat{z}_j(d^*) = \mathbf{\Sigma}_j^{(u)}(d^*)^T \hat{\mathbf{u}}_j$. To compute the predictive variance in (2.16), we first calculate the N_j-dimensional vector $\tilde{\mathbf{\Sigma}}_j^{(u)}(d^*) = \mathbf{A}_j \mathbf{\Sigma}_j^{(u)}(d^*)$ using Step 1. Then Steps 1-4 in the CG algorithm are applied to compute $\hat{\mathbf{\Sigma}}_j^{(u)}(d^*) = \mathbf{\Sigma}_y^{-1} \tilde{\mathbf{\Sigma}}_j^{(u)}(d^*)$, and the predictive variance is thus computed as $c_j^*(d^*) = c_j(d^*, d^*) - \hat{\mathbf{\Sigma}}_j^{(u)}(d^*)^T \tilde{\mathbf{\Sigma}}_j^{(u)}(d^*)$.

For computing the predictive distribution of j-th interaction functions on multiple test inputs $\mathbf{d}_j^* = [d_{1,j}^*, \ldots, d_{N_j^*,j}^*]^T$, once $\hat{\mathbf{u}}_j = \mathbf{A}_j^T \mathbf{\Sigma}_y^{-1} \mathbf{y}$ is obtained, the direct calculation of the posterior mean involves matrix-vector multiplication $\mathbf{\Sigma}_j^{(u)}(\mathbf{d}_j^*)^T \hat{\mathbf{u}}_j$, which costs $\mathcal{O}(N_j^* N_j)$ if computed directly. This remains computationally intensive for large N_j^*, especially during parameter estimation that requires multiple iterations. To reduce computational complexity, we generate an augmented distance vector $\mathbf{d}_j^{aug,(u)} = [(\mathbf{d}_j^*)^T, (\mathbf{d}_j^{(u)})^T]^T$. We then sort this $(N_j^* + N_j)$-dimensional vector into a non-decreasing sequence: $\mathbf{d}_j^{aug} = [d_{1,j}^{aug}, \ldots, d_{N_j^*+N_j,j}^{aug}]^T$, where $d_{t,j}^{aug} \leq d_{t',j}^{aug}$ if $t < t'$. This sorting generates the index set $\{\tilde{g}_t\}_{t=1}^{N_j^*+N_j}$ such that $d_{t,j}^{aug} = d_{\tilde{g}_t,j}^{aug,(u)}$. Next, we define the augmented vector $\hat{\mathbf{u}}_j^{aug,(u)} = [\mathbf{0}_{N_j^*}^T, \hat{\mathbf{u}}_j^T]^T$ of $N_j^* + N_j$ dimensions and rearrange it as $\hat{\mathbf{u}}_j^{aug}$ with $\hat{u}_{t,j}^{aug} = \hat{u}_{\tilde{g}_t,j}^{aug,(u)}$ for $t = 1, \ldots, N_j^* + N_j$. The IKF from Algorithm 2.1 is then used to compute $\hat{\mathbf{z}}_j^{aug} = \mathbf{\Sigma}_j^{aug} \hat{\mathbf{u}}_j^{aug}$, where $\mathbf{\Sigma}_j^{aug}$ is the covariance matrix of $\mathbf{d}_j^{aug}$. Finally, $\hat{\mathbf{z}}_j^{aug}$ is reordered to the initial order $\hat{\mathbf{z}}_j^{aug,(u)}$ with $\hat{z}_{\tilde{g}_t,j}^{aug,(u)} = \hat{z}_{t,j}^{aug}$, and the predictive mean $\hat{\mathbf{z}}_j(\mathbf{d}_j^*)$ is the first N_j^* entries from $\hat{\mathbf{z}}_{\tilde{g}_t,j}^{aug,(u)}$. This approach reduces computational operations and storage cost to $\mathcal{O}(\tilde{q}_j^3(N_j^* + N_j))$ without approximation.

To compute the predictive variance on multiple test inputs, we employ the IKF to compute the most demanding terms $\mathbf{\Sigma}_y^{-1} \mathbf{A}_j \mathbf{\Sigma}_j^{(u)}(d^*)$ as for predicting on a single test input. Unlike the posterior mean computation, which is required multiple times in parameter estimation, the calculation of the posterior variance only needs to be performed once, making

the computation typically feasible after being accelerated by the IKF algorithm. In addition, one may approximate the predictive variance of a test input by the predictive variance on the neighboring grid.

B.4 Application of predicting incomplete lattice for correlated data

Missing values in lattice data are ubiquitous in scientific datasets, such as satellite radar interferograms [229] and temperature measurements [230]. Here, we demonstrate the use of the IKF-CG algorithm to efficiently compute predictions of incomplete lattice data. Let $\mathbf{Z} \in \mathbb{R}^{n_1 \times n_2}$ represent the unobserved complete lattice data, where the lattice is defined on input variables $\mathbf{s}_1 = [s_{1,1}, \ldots, s_{n_1,1}]$ and $\mathbf{s}_2 = [s_{1,2}, \ldots, s_{n_2,2}]$. Each entry $Z_{i,j}$ corresponds to the value of the function $Z(s_{i,1}, s_{j,2})$ at the input location $(s_{i,1}, s_{j,2})$ on the 2D grid. The lattice is vectorized into an N-dimensional vector $\mathbf{z}$ where $\text{vec}(\mathbf{Z}) = \mathbf{z}$ with $\text{vec}(\cdot)$ being the vectorization operator and $N = n_1 \times n_2$. Denote the observed incomplete data as $\mathbf{y}$, consisting of $\tilde{N}$ observations with $\tilde{N} < N$. We model $\mathbf{y}$ as $\mathbf{y} = \mathbf{A}\mathbf{z} + \boldsymbol{\epsilon}$, where $\mathbf{A}$ is a sparse $\tilde{N} \times N$ matrix that maps the latent vector $\mathbf{z}$ to the observed data $\mathbf{y}$. Specifically, $\mathbf{A}$ is formed by removing rows corresponding to missing regions from an $N \times N$ identity matrix, and contains $\tilde{N}$ 1s and $N\tilde{N} - \tilde{N}$ 0s. The Gaussian noise vector follows $\boldsymbol{\epsilon} \sim \mathcal{MN}(\mathbf{0}, \sigma_0^2 \mathbf{I}_{\tilde{N}})$.

For demonstration purposes, we consider $\mathbf{z} \sim \mathcal{MN}(\mathbf{0}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma} = \sigma^2 \mathbf{R}_1 \otimes \mathbf{R}_2$, with $\mathbf{R}_1$ and $\mathbf{R}_2$ being the $n_1 \times n_1$ and $n_2 \times n_2$ correlation matrices of input variables $\mathbf{s}_1$ and $\mathbf{s}_2$ of the lattice, respectively, and $\otimes$ denotes the Kronecker product. Each of $\mathbf{R}_1$ and $\mathbf{R}_2$ is parameterized by a correlation function, such as the Matérn correlation, which enables the use of the IKF for fast matrix-vector multiplication. This framework can be extended to models with semi-separable or non-separable covariances, and distinct mean patterns [52].

After integrating out the latent factors $\mathbf{z}$, the marginal distribution of $\mathbf{y}$ follows

$$\left(\mathbf{y} \mid \boldsymbol{\Sigma}, \sigma_0^2\right) \sim \mathcal{MN}\left(\mathbf{0}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T + \sigma_0^2 \mathbf{I}_{\tilde{N}}\right).$$

Given the observed data, the posterior distribution of $\mathbf{z}$ is a multivariate normal distribution with mean $\hat{\mathbf{z}}$ given by

$$\hat{\mathbf{z}} = \boldsymbol{\Sigma}\mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{y}, \tag{B.6}$$

where $\boldsymbol{\Sigma}_y = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T + \sigma_0^2\mathbf{I}_{\tilde{N}}$. To calculate the predictive mean, we employ the IKF-CG algorithm. Each iteration of the CG algorithm requires computing $\boldsymbol{\Sigma}_y\mathbf{u} = (\mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T + \sigma_0^2\mathbf{I}_{\tilde{N}})\mathbf{u}$ for a $\tilde{N}$-dimensional vector $\mathbf{u}$. The most computationally expensive part is computing $\boldsymbol{\Sigma}\mathbf{u} = (\mathbf{R}_1 \otimes \mathbf{R}_2)\mathbf{u} = \text{vec}(\mathbf{R}_2\mathbf{U}\mathbf{R}_1)$, where $\text{vec}(\mathbf{U}) = \mathbf{u}$. This computation can be accelerated using the IKF algorithm when the correlation matrix is induced by DLM. We first compute $\mathbf{X} = \mathbf{R}_2\mathbf{U}$ by applying the IKF algorithm column-wise to $\mathbf{U}$, and then compute $\hat{\mathbf{U}} = (\mathbf{R}_1\mathbf{X}^T)^T$ by again applying the IKF algorithm column-wise. This reduces the computational cost of each matrix-matrix multiplication from $\mathcal{O}(N^2)$ to $\mathcal{O}(\tilde{q}^3 N)$ if we use $\tilde{q}$ latent states for both inputs. Details of parameter estimation are provided in Section B.5.

B.5 Parameter estimation

This section discusses the parameter estimation methods for both applications: estimating particle interaction functions (Section 2.3) and predicting incomplete lattice data (Section B.4). In the application described in Section 2.3, the model parameters include the range parameters $\boldsymbol{\gamma} = (\gamma_1, \ldots, \gamma_J)$, the variance of each latent factor process $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_J^2)$, the variance of the noise σ_0^2, and other physical parameters $\mathbf{r}$, such

as the radius of interactions between particles. For computational efficiency, we define $\tilde{\boldsymbol{\eta}} = (\tilde{\eta}_1, \ldots, \tilde{\eta}_J)$ with $\tilde{\eta}_j^2 = \sigma_j^2/\sigma_0^2$ for $j = 1, \ldots, J$ as the ratio of the variance of the J interaction functions to the variance of the noise, and σ_0^2 can be marginalized out explicitly. This parameterization is preferred over σ_0^2/σ_j^2 (the nugget defined in Chapter 1), because σ_j^2 could be close to zero if the effect of the j-th particle interaction is small, thus causing numerical instability. In contrast, the variance σ_0^2, accounting for measurement noise and model inadequacy, is typically larger than zero, making the parameterization by $\tilde{\boldsymbol{\eta}}$ a preferred choice for our applications. Similarly, for the application in Section B.4, we define $\tilde{\eta} = \sigma^2/\sigma_0^2$.

In [110], the authors developed a fast algorithm, which is only applicable for a Gaussian process (GP) with an exponential kernel, fixed hyperparameters of covariance functions, and physical parameters. In this study, we extend the methodology by employing two parameter estimation methods applicable to GPs with any DLM-induced covariance structure and utilizing the residual bootstrap method for quantifying the uncertainty of parameter estimation. All these approaches can be accelerated using the proposed IKF algorithm.

First, we focus on the cross-validation approach for parameter estimation, where we split the observations into training and hold-out validation sets, denoted as $\mathbf{y}_{train}$ and $\mathbf{y}_{val}$ for N_{train} training and N_{val} validation vectors, respectively. Parameters are estimated by minimizing a loss function, chosen as the mean squared error, during cross-validation:

$$(\hat{\tilde{\boldsymbol{\eta}}}, \hat{\boldsymbol{\gamma}}, \hat{\mathbf{r}}) = \text{argmin}_{(\tilde{\boldsymbol{\eta}}, \boldsymbol{\gamma}, \mathbf{r})} \left\{ \frac{1}{N_{val}} (\hat{\mathbf{y}}_{val} - \mathbf{y}_{val})^T (\hat{\mathbf{y}}_{val} - \mathbf{y}_{val}) \right\},$$

where for application in Section 2.3, $\hat{\mathbf{y}}_{val} = \sum_{j=1}^{J} \mathbf{A}_{j,val} \hat{\mathbf{z}}_{j,val}$, with $\hat{\mathbf{z}}_{j,val}$ being the posterior mean of the latent interaction function for cross-validation inputs $\mathbf{d}_j^*$ of size N_j^* and $\mathbf{A}_{j,val}$ being a sparse latent factor loading matrix of dimensions $N_{val} \times N_j^*$, for $j = 1, \ldots, J$. For

application in Section B.4, $\hat{\mathbf{y}}_{val} = \mathbf{A}_{val}\hat{\mathbf{z}}_v$, where $\mathbf{A}_{val}$ is an $N_{val} \times N$ matrix formed by removing rows corresponding to the entries of the training data from an $N \times N$ identity matrix.

Conditioning on other estimated parameters, the maximum likelihood estimate of σ_0^2 is obtained via $\hat{\sigma}_0^2 = \mathbf{y}^T\mathbf{R}_y^{-1}\mathbf{y}/\tilde{N}$, where $\mathbf{R}_y = \mathbf{\Sigma}_y/\sigma_0^2$. The computation of $\mathbf{R}_y^{-1}\mathbf{y}$ can be efficiently handled using the IKF-CG algorithm. The parameters are then transformed to obtain $\hat{\sigma}_j^2 = \hat{\tilde{\eta}}_j^2\hat{\sigma}_0^2$ and $\hat{\sigma}^2 = \hat{\tilde{\eta}}^2\hat{\sigma}_0^2$ in applications of Sections 2.3 and B.4, respectively.

Next, we discuss the maximum likelihood estimation, which requires computing the determinant of a large $\tilde{N} \times \tilde{N}$ matrix $\mathbf{\Sigma}_y$. This computationally expensive step, requiring $\mathcal{O}(\tilde{N}^3)$ operations in direct computation, can be addressed with a low-rank approximation $\mathbf{\Sigma}_a$ of rank N_a for the matrix $\mathbf{\Sigma}_0$ [66], with $\mathbf{\Sigma}_0 = \sum_{j=1}^{J} \mathbf{A}_j\mathbf{\Sigma}_j^{(u)}\mathbf{A}_j^T/\sigma_0^2$ and $\mathbf{\Sigma}_0 = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T/\sigma_0^2$ for applications in Sections 2.3 and B.4, respectively. This approach approximates the log-determinant of the covariance matrix by $\log\left|\mathbf{\Sigma}_0 + \mathbf{I}_{\tilde{N}}\right| \approx \log|\mathbf{\Sigma}_a + \mathbf{I}_{N_a}|$. The key assumption is that the matrix $\mathbf{\Sigma}_0$ has N_a dominant eigenvalues, a sensible assumption for both applications.

To construct $\mathbf{\Sigma}_a$, we first set the dimension N_a of the matrix $\mathbf{\Sigma}_a$ with $N_a \ll \tilde{N}$ and initialize a $\tilde{N} \times N_a$ matrix $\mathbf{\Omega}$ with i.i.d. standard normal entries. Next, we compute $\mathbf{\Sigma}_0^{M_0}\mathbf{\Omega}$, where M_0 is a positive integer. The QR decomposition of $\mathbf{\Sigma}_0^{M_0}\mathbf{\Omega}$ yields an orthogonal matrix $\mathbf{U} \in \mathbb{R}^{\tilde{N}\times N_a}$, and the low-rank matrix $\mathbf{\Sigma}_a$ is defined as $\mathbf{\Sigma}_a = \mathbf{U}^T\mathbf{\Sigma}_0\mathbf{U}$. All computations can be accelerated using the IKF method, resulting in a cost of $\mathcal{O}(M_0\sum_{j=1}^{J} N_j(\log(N_j)+D_y+\tilde{q}_j^3))$ and $O(\tilde{q}^3M_0N)$ for applications in Sections 2.3 and B.4, respectively. When the gap between dominant and sub-dominant eigenvalues is large, a small M_0 suffices. Figure 2.3(c) presents the results using $M_0 = 1$, which shows an accurate approximation. Then with the maximum likelihood estimate of $\hat{\sigma}_0^2 = \mathbf{y}^T\mathbf{R}_y^{-1}\mathbf{y}/\tilde{N}$, parameters $(\tilde{\boldsymbol{\eta}}, \boldsymbol{\gamma}, \mathbf{r})$ are estimated by maximizing

the logarithm of the marginal likelihood after integrating out the latent factors:

$$(\hat{\tilde{\boldsymbol{\eta}}}, \hat{\boldsymbol{\gamma}}, \hat{\mathbf{r}}) = \underset{(\tilde{\boldsymbol{\eta}}, \boldsymbol{\gamma}, \mathbf{r})}{\operatorname{argmax}} \left\{ -\frac{1}{2} \log |\boldsymbol{\Sigma}_a + \mathbf{I}_{N_a}| - \frac{\tilde{N}}{2} \log \left(\mathbf{y}^T \mathbf{R}_y^{-1} \mathbf{y} \right) \right\}.$$

This approximation is as fast as cross-validation, with small approximation errors in our applications, as demonstrated in Figure 2.3(c) for a simulated study.

Lastly, we employ residual bootstrap to quantify uncertainty in parameter estimation [137, 231]. This involves computing the posterior mean of the observations $\hat{\mathbf{y}}$ using the predictive interaction function with estimated parameters $(\hat{\tilde{\boldsymbol{\eta}}}, \hat{\boldsymbol{\gamma}}, \hat{\mathbf{r}})$, and calculating residuals $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}} = [e_1, \ldots, e_{\tilde{N}}]^T$. We then generate a new set of $\tilde{N}$ residuals $\mathbf{e}^* = [e_1^*, \ldots, e_{\tilde{N}}^*]^T$, such that $\text{pr}(e_i^* = e_{i'}) = 1/\tilde{N}$ for $i' = 1, \ldots, \tilde{N}$. This creates a new set of bootstrap observations via $\mathbf{y}^* = \hat{\mathbf{y}} + \mathbf{e}^*$. Using the same inputs and bootstrap observations, we refit the model to obtain the estimates $\hat{\boldsymbol{\gamma}}^*$, $\hat{\tilde{\boldsymbol{\eta}}}^*$, and $\hat{\mathbf{r}}^*$. Repeating this procedure B times, we obtain confidence intervals by taking the percentiles of these B estimates for each parameter.

B.6 Computational complexity

The primary computational challenge lies in estimating the hyperparameters in the model. Here, we mainly focus on parameter estimation with cross-validation. The computational cost of maximum likelihood estimation is similar. First, we analyze the computational complexity of learning particle interactions in Section 2.3. Table B.1 lists the computational complexity of three different parameter estimation algorithms, namely our IKF-CG method, the conventional CG algorithm, and the direct computation via Cholesky decomposition and forward-backward solver (see Appendix A.4 of [9]). We assume that all algorithms require M iterations for parameter estimation, and in each iteration, the CG and IKF-CG methods require S iterations to compute $\boldsymbol{\Sigma}_y^{-1}\mathbf{y}$. In our applications, both M

Table B.1: The computational complexity of parameter estimation with M iterations of numerical optimization of the parameters in learning particle interactions with $N_j^{all} = N_j + N_j^*$. The CG and direct computation algorithms involve forming an $N_j \times N_j$ covariance matrix, while the IKF-CG only requires computing the ordered inputs. We assume computing weights $\hat{\mathbf{u}}_j = \mathbf{A}_j^T \boldsymbol{\Sigma}_y^{-1} \mathbf{y}$ is a separate step with complexity shown in the third row of the table.

Steps	IKF-CG	CG	Direct computation
Form covariance	/	$O(M \sum_{j=1}^J N_j^2)$	$\mathcal{O}(M \sum_{j=1}^J N_j^2)$
Sort inputs	$\mathcal{O}(\sum_{j=1}^J (N_j^{all}) \log(N_j^{all}))$	/	/
Compute weights	$\mathcal{O}(MS \sum_{j=1}^J (D_y + \tilde{q}_j^3) N_j)$	$O(MS \sum_{j=1}^J N_j^2)$	$\mathcal{O}(M \tilde{N}^3)$
Compute loss	$\mathcal{O}(M \sum_{j=1}^J (N_j^{all}))$	$\mathcal{O}(M \sum_{j=1}^J N_j^* N_j)$	$\mathcal{O}(M \sum_{j=1}^J N_j^* N_j)$

and S are around a hundred to ensure the approximation error is a few orders of magnitude smaller than the predictive error (Figure 2.3). We will pre-process the particle information and save the results in coarse-grained grids, shown in Figure B.1(a), which only takes $\mathcal{O}(D_y \sum_{\tau=1}^{n_\tau} n_p(\tau)) = \mathcal{O}(\tilde{N})$ operations.

After constructing the neighbors, the cross-validation estimation of the parameters is split into three parts: $\hat{\mathbf{u}} = \boldsymbol{\Sigma}_y^{-1} \mathbf{y}$, $\hat{\boldsymbol{z}}_j(\mathbf{d}_j^*) = \boldsymbol{\Sigma}_j^{(u)}(\mathbf{d}_j^*) \mathbf{A}_j^T \hat{\mathbf{u}}$ and $\hat{\mathbf{y}}_{val} = \sum_{j=1}^J \mathbf{A}_{val,j}^T \hat{\boldsymbol{z}}_j(\mathbf{d}_j^*)$. To compute $\boldsymbol{\Sigma}_y^{-1} \mathbf{y}$, we first sort $\mathbf{d}_j^{(u)}$ in a nondecreasing order to obtain the sorted inputs $\mathbf{d}_j$ for each interaction kernel, which requires $\mathcal{O}(\sum_{j=1}^J N_j \log(N_j))$ operations. As described in Section B.3, for each iteration in the CG algorithm, we compute $\mathbf{u}_j^{(u)} = \boldsymbol{A}_j^T \mathbf{u}$ with $\mathbf{u}$ being the current vector for optimization, which requires $\mathcal{O}(D_y N_j)$ operations for $j = 1, \ldots, J$. We then rearrange the vector $\mathbf{u}_j^{(u)}$ to obtain $\mathbf{u}_j$ and use the IKF algorithm to compute $\mathbf{x}_j = \boldsymbol{\Sigma}_j \mathbf{u}_j$ with a computational cost of $\mathcal{O}(\tilde{q}_j^3 N_j)$ for all j. After reordering $\mathbf{x}_j$ to get $\mathbf{x}_j^{(u)}$, we compute $\hat{\mathbf{u}}_j = \mathbf{A}_j \mathbf{x}_j^{(u)}$, which again requires $\mathcal{O}(D_y N_j)$ operations. Finally, we calculate $\boldsymbol{\Sigma}_y \mathbf{u} = \sum_{j=1}^J \hat{\mathbf{u}}_j + \sigma_0^2 \mathbf{u}$ with $\mathcal{O}(J\tilde{N})$ operations. Typically $N_j \geq N$ as the particles contain no less than one neighbor on average. Thus, the total computational operations for $\boldsymbol{\Sigma}_y^{-1} \mathbf{y}$ amount to $\mathcal{O}(\sum_{j=1}^J N_j \log(N_j)) + \mathcal{O}(S D_y \sum_{j=1}^J N_j) + \mathcal{O}(S \sum_{j=1}^J \tilde{q}_j^3 N_j)$, where S is the total number of CG iterations. Typically, around 100 iterations are sufficient to provide an accurate estimation in our applications unless the system has extremely small noise

and large correlation, which may make the covariance near singular. Second, computing $\sum_{j=1}^{J} \mathbf{\Sigma}_j^{(u)}(\mathbf{d}_j^*)\mathbf{A}_j^T\hat{\mathbf{u}}$ costs $\mathcal{O}(\sum_{j=1}^{J}(N_j^*+N_j)\log(N_j^*+N_j))+\mathcal{O}(\sum_{j=1}^{J}\tilde{q}_j^3(N_j^*+N_j))$ operations using the IKF algorithm with the augmented covariance matrix and output vector, and the details are discussed in Section B.3. Lastly, computing $\sum_{j=1}^{J}\mathbf{A}_{val,j}^T\hat{\mathbf{z}}_j(\mathbf{d}_j^*)$ requires $\mathcal{O}(\sum_{j=1}^{J} D_y N_j^*)$ operations.

In particle interaction applications, $\tilde{N}$ and N_j stand for the number of observations and the number of inputs in the j-th interaction, respectively, which are typically large in real-world applications, ranging from 10^3 to 10^6. The number of validation inputs N_j^* for the j-th interaction has the same order as N_j, and all other quantities listed are relatively small. The number of interactions, J, is typically 1 or 2. For modeling interaction functions, Matérn covariances with roughness parameters $1/2$ and $5/2$ are used, corresponding to $\tilde{q}_j = 1$ and $\tilde{q}_j = 3$, respectively. The dimension of the output vector, D_y, is generally no more than 3 in practical scenarios.

The direct computation method requires $\mathcal{O}(M\sum_{j=1}^{J} N_j^2)$ and $\mathcal{O}(M\tilde{N}^3)$ operations for forming the covariance matrix and computing its inversion, respectively, which is prohibitively slow for large dataset. The CG algorithm is typically faster than the direct computation method, but still prohibitively expensive to compute $\hat{\mathbf{u}}_j = \mathbf{A}_j^T\mathbf{\Sigma}_y^{-1}\mathbf{y}$ with MS iterations for parameter estimation. The IKF-CG algorithm significantly improves the computational order by removing the cost of forming the $N_j \times N_j$ covariance matrix and computing the weights and loss. It is approximately N_j times faster than the CG algorithm for matrix-vector multiplications involving a dense covariance $\mathbf{\Sigma}_j$. The scalability of the IKF-CG algorithm makes it a suitable alternative for fast parameter estimation and predictions.

The computational cost associated with modeling incomplete lattice data using the IKF-CG algorithm is more straightforward. Specifically, parameter estimation for the

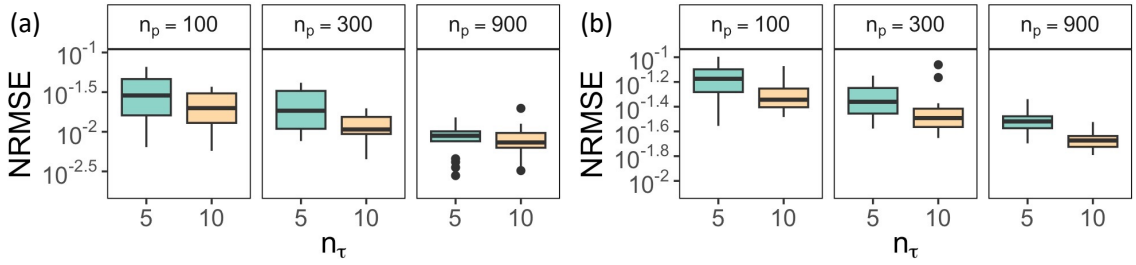

Figure B.2: Boxplots of NRMSE (2.17) for estimating the latent interaction function in the unnormalized Vicsek model with $\sigma_0^2 = 0.2^2$. Results are based on 20 experiments for each scenario, using the Matérn covariance with roughness parameter 5/2 (Panel (a)) and exponential covariance (Panel (b)).

IKF-CG by cross-validation primarily involves computing the predictive mean in (B.6), which requires $\mathcal{O}(MS\tilde{q}^3N)$ operations and $\mathcal{O}(\tilde{q}^3N)$ storage cost, both linear to the size of the covariance matrix. The cost of the IKF-CG algorithm is significantly lower than that of direct computational methods, which require $\mathcal{O}(M\tilde{N}^3)$ for matrix inversion and $\mathcal{O}(\tilde{N}^2)$ for storing the covariance matrix.

B.7 Additional numerical results for estimating interaction functions

B.7.1 Unnormalized Vicsek model

Here we provide additional numerical results for the unnormalized Vicsek model under various scenarios. Figures B.2 and B.3 present the predictive performance with $\sigma_0^2 = 0.2^2$. Panels (a) and (b) show results for the Matérn kernel with a roughness parameter of 5/2 and the exponential kernel, respectively. As in scenarios with $\sigma_0^2 = 0.1^2$, we observe consistently low NRMSE, with increasing prediction accuracy and decreasing length of the posterior credible intervals, as the number of observations increases.

Figure B.4 shows boxplots of the estimated radius parameters for each of the

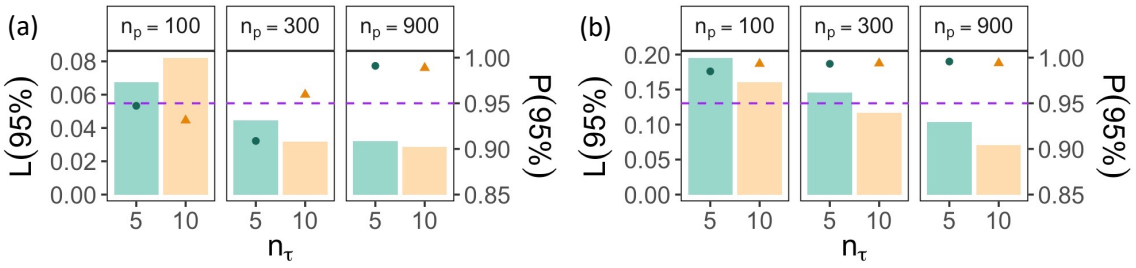

Figure B.3: Uncertainty assessment of the predicted interaction function of the unnormalized Vicsek model using the Matérn covariance function with roughness parameter 5/2 (Panel (a)) and the exponential covariance function (Panel (b)), both with $\sigma_0^2 = 0.2^2$. The bars represent the mean length of the 95% posterior credible interval (2.18) over 20 experiments, and the dots represent the average percentage of test data covered by 95% posterior credible interval (2.19) across 20 experiments. The purple dashed line at 0.95 indicates the optimal coverage level for the dots.

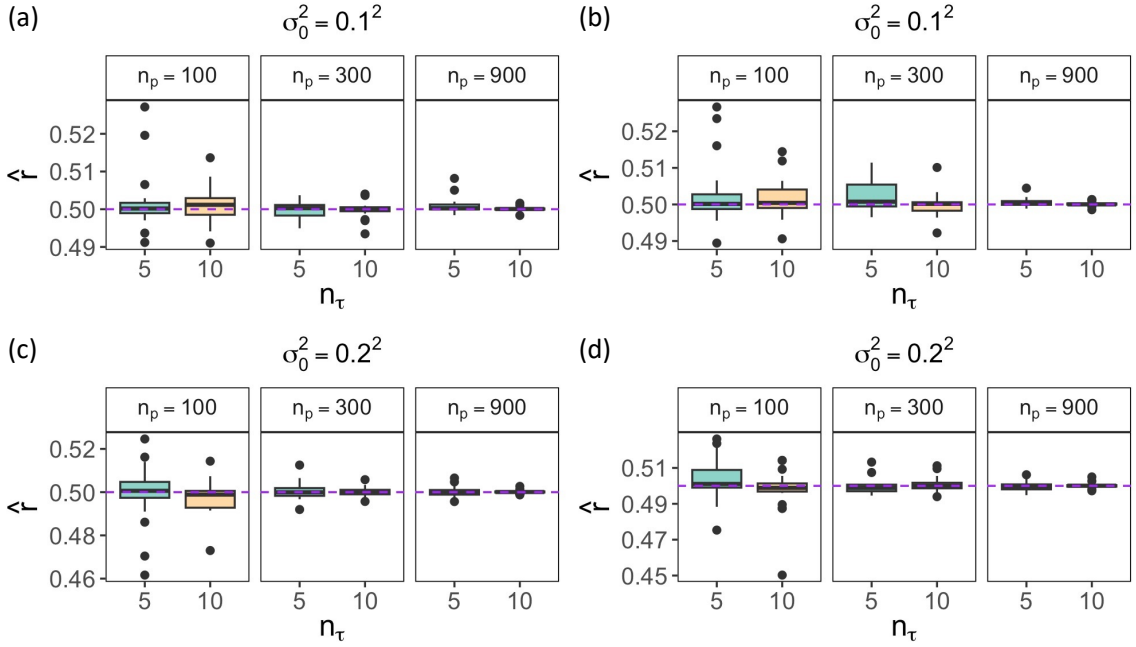

Figure B.4: Boxplots of the estimated radius distance $\hat{r}$ in the unnormalized Vicsek model. Panels (a) and (c) use the Matérn covariance with a roughness parameter 5/2 for $\sigma_0^2 = 0.1^2$ and $\sigma_0^2 = 0.2^2$, respectively, while panels (b) and (d) use the exponential covariance for the same noise levels. The purple dashed lines mark the true radius distance $r = 0.5$.

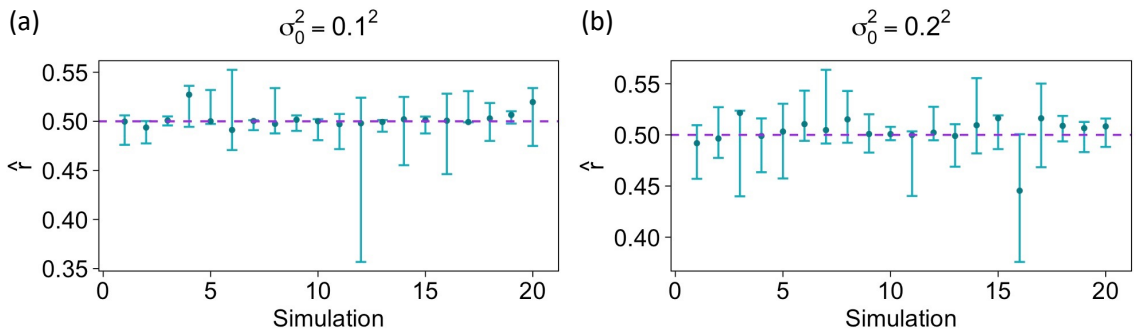

Figure B.5: The 95% credible intervals for the estimated interaction radius in the unnormalized Vicsek model, obtained using the Matérn 5/2 covariance and $B = 100$ residual bootstrap samples. Results are based on 20 simulations for $\sigma_0^2 = 0.1^2$ and $\sigma_0^2 = 0.2^2$, with $n_p = 100$ and $n_\tau = 5$. The dots represent the estimated radius $\hat{r}$ from the original data, and the purple dashed lines represent the true radius $r = 0.5$.

12 simulated scenarios using the Matérn covariance function with a roughness of 5/2 and exponential covariance. Across all simulations, the estimated radii closely align with the true value $r = 0.5$, and the accuracy of radius estimation improves with larger sample sizes.

To quantify the uncertainty of the estimated radius $\hat{r}$, as well as other parameters, we employ the residual bootstrap method introduced in Section B.5. Figure B.5 shows the 95% intervals obtained from $B = 100$ bootstrap samples, using the Matérn kernel of roughness parameter 5/2 for two scenarios with $\sigma_0^2 = 0.1^2$ and $\sigma_0^2 = 0.2^2$, where $n_\tau = 5$ and $n_p = 100$ are used for both scenarios. All bootstrap intervals cover the true radius r, which indicates a reliable assessment of parameter uncertainty in these applications.

B.7.2 Modified Vicsek model

The neighbor set in the first term of the modified Vicsek model in (2.21) is defined as $ne_{i'}(\tau - 1) = \{k : \|\mathbf{s}_k(\tau - 1) - \mathbf{s}_{i'}(\tau - 1)\| < r\}$, with $p_{i'}(\tau - 1)$ being the total neighbors in this set. The second term introduces a distance-dependent interaction governed by $f(d_{i',k}(\tau - 1)) = -2(5(d_{i',k}(\tau - 1) + 0.01))^{-1} - 20d_{i',k}(\tau - 1) + 10.8$, where $d_{i',k}(\tau - 1) = \|\mathbf{s}_{i'}(\tau - 1) - \mathbf{s}_k(\tau - 1)\|$. This interaction captures repulsion at close distances and attraction

Table B.2: The predictive accuracy and uncertainty assessment by (2.17)-(2.19) for the modified Vicsek model with $\sigma_0^2 = 0.2^2$ using Matérn covariance function and roughness parameter 5/2.

		First Interaction			Second Interaction		
		NRMSE_1	$L_1(95\%)$	$P_1(95\%)$	NRMSE_2	$L_2(95\%)$	$P_2(95\%)$
$n_p = 100$	$n_\tau = 5$	9.9×10^{-3}	0.16	98%	1.6×10^{-1}	0.45	93%
	$n_\tau = 10$	8.9×10^{-3}	0.19	94%	1.0×10^{-1}	0.37	94%
$n_p = 300$	$n_\tau = 5$	9.7×10^{-3}	0.23	98%	4.8×10^{-2}	0.31	97%
	$n_\tau = 10$	9.2×10^{-3}	0.22	96%	3.5×10^{-2}	0.22	96%
$n_p = 900$	$n_\tau = 5$	6.2×10^{-3}	0.19	100%	2.5×10^{-2}	0.21	99%
	$n_\tau = 10$	7.3×10^{-3}	0.23	99%	2.0×10^{-2}	0.18	99%

Table B.3: The root mean squared error (RMSE) of the estimated radius for modified Vicsek model using Matérn covariance function with roughness parameter 5/2, computed as $(\sum_{i=1}^{20}(\hat{r}_i - r)^2/20)^{1/2}$ with $\hat{r}_i$ being the estimated radius of the i-th experiment for each scenario.

	$n_p = 100$		$n_p = 300$		$n_p = 900$	
	$n_\tau = 5$	$n_\tau = 10$	$n_\tau = 5$	$n_\tau = 10$	$n_\tau = 5$	$n_\tau = 10$
$\sigma_0^2 = 0.1^2$	3.5×10^{-3}	2.7×10^{-3}	1.7×10^{-3}	1.1×10^{-3}	4.2×10^{-4}	1.9×10^{-4}
$\sigma_0^2 = 0.2^2$	3.2×10^{-3}	4.0×10^{-3}	1.6×10^{-3}	1.0×10^{-3}	7.9×10^{-4}	3.1×10^{-4}

at longer distances with the unit vector $\mathbf{e}_{i',k}(\tau-1) = (\mathbf{s}_k(\tau-1) - \mathbf{s}_{i'}(\tau-1))/d_{i',k}(\tau-1)$ specifying the interaction direction. The corresponding neighboring set excludes particle i' itself and is defined as $ne'_{i'}(\tau-1) = \{k : \|\mathbf{s}_k(\tau-1) - \mathbf{s}_{i'}(\tau-1)\| < r \text{ and } i' \neq k\}$, with $p'_{i'}(\tau-1)$ denoting the number of neighbors.

The modified Vicsek model in (2.21) can be represented as a latent factor model in (2.11) with two latent interactions. The first interaction is linear, $z_1(d_1) = d_1$, where $d_1 = v_{i',1}(\tau)$ or $d_1 = v_{i',2}(\tau)$, the same as in the unnormalized Vicsek model. The second interaction is nonlinear, $z_2(d_2) = f(d_2)$, where $d_2 = d_{i',k}(\tau-1)$. A common interaction radius $r = 0.5$ is assumed for both interactions.

We first present additional results for Matérn covariance function with a roughness of 5/2 in the modified Vicsek model discussed in Section 2.4.3, which consists of two types of interaction. Table B.2 summarizes the predictive performance with $\sigma_0^2 = 0.2^2$. Consistent with results for $\sigma_0^2 = 0.1^2$, we observe improved performance with an increasing number of observations, especially for the second interaction. Table B.3 presents the root mean squared

error (RMSE) of the estimated radius. All RMSE values are small, with estimation accuracy improving as the training size increases.

Table B.4: The prediction results of modified Vicsek model with exponential covariance function.

		$\sigma_0^2 = 0.1^2$					
		First Interaction			Second Interaction		
		NRMSE_1	$L_1(95\%)$	$P_1(95\%)$	NRMSE_2	$L_2(95\%)$	$P_2(95\%)$
$n_p = 100$	$n_\tau = 5$	1.1×10^{-1}	0.70	88%	3.6×10^{-1}	1.07	92%
	$n_\tau = 10$	4.9×10^{-2}	0.35	94%	6.7×10^{-2}	0.47	98%
$n_p = 300$	$n_\tau = 5$	1.2×10^{-1}	0.45	84%	5.3×10^{-2}	0.68	98%
	$n_\tau = 10$	3.1×10^{-2}	0.29	92%	3.3×10^{-2}	0.31	99%
$n_p = 900$	$n_\tau = 5$	6.6×10^{-2}	0.52	91%	3.0×10^{-2}	0.25	98%
	$n_\tau = 10$	5.4×10^{-2}	0.50	91%	2.5×10^{-2}	0.23	99%
		$\sigma_0^2 = 0.2^2$					
		First Interaction			Second Interaction		
		NRMSE_1	$L_1(95\%)$	$P_1(95\%)$	NRMSE_2	$L_2(95\%)$	$P_2(95\%)$
$n_p = 100$	$n_\tau = 5$	6.6×10^{-2}	0.65	95%	1.9×10^{-1}	1.24	96%
	$n_\tau = 10$	4.7×10^{-2}	0.82	94%	1.6×10^{-1}	1.01	95%
$n_p = 300$	$n_\tau = 5$	8.5×10^{-2}	0.64	90%	8.1×10^{-2}	0.63	98%
	$n_\tau = 10$	8.3×10^{-2}	0.59	90%	5.5×10^{-2}	0.43	98%
$n_p = 900$	$n_\tau = 5$	7.7×10^{-2}	0.54	89%	4.5×10^{-2}	0.41	99%
	$n_\tau = 10$	4.1×10^{-2}	0.55	93%	3.7×10^{-2}	0.34	99%

Table B.5: The RMSE of the estimated radius for modified Vicsek model using exponential covariance function.

	$n_p = 100$		$n_p = 300$		$n_p = 900$	
	$n_\tau = 5$	$n_\tau = 10$	$n_\tau = 5$	$n_\tau = 10$	$n_\tau = 5$	$n_\tau = 10$
$\sigma_0^2 = 0.1^2$	3.1×10^{-3}	1.7×10^{-3}	1.3×10^{-3}	1.3×10^{-3}	3.3×10^{-4}	2.7×10^{-4}
$\sigma_0^2 = 0.2^2$	5.3×10^{-3}	1.9×10^{-3}	2.1×10^{-3}	9.9×10^{-4}	5.1×10^{-4}	2.9×10^{-4}

Next, we present the numerical results of latent factor model using an exponential covariance function. The results, presented in Table B.4, exhibit similar trends to those obtained by using the Matérn covariance with a roughness parameter of 5/2. Specifically, the NRMSE for the second interaction is larger than that of the first interaction due to fewer observations with small inputs of the second interaction, which can be resolved by increasing the number of observations. In addition, the length of the 95% posterior credible interval decreases as the number of data points grows, while the coverage proportion remains close to the 95% nominal level. Table B.5 displays the RMSE for radius estimation, which exhibits

comparable performance to that achieved by using the Matérn covariance with a roughness parameter of 5/2. The accuracy of the estimation improves as we have more observations.

B.8 Numerical results of predicting incomplete lattice data

B.8.1 Evaluation criteria

In this section, we present the results of a simulated study in Section B.8.2 and a real data analysis in Section B.8.3 to predict incomplete lattices of correlated data. The NRMSE is used to evaluate the predictive performance of the missing regions. Similar to the definition of observation $\mathbf{y}$ in Section B.4, the $N - \tilde{N}$ dimensional vector of missing entries is modeled by $\mathbf{y}^{(m)} = \mathbf{A}^{(m)}\mathbf{z} + \boldsymbol{\epsilon}^{(m)}$, where $\mathbf{A}^{(m)} \in \mathbb{R}^{(N-\tilde{N})\times N}$ has $(N - \tilde{N})$ 1s, with all other entries being 0s to map the latent vector to the missing values, and $\boldsymbol{\epsilon}^{(m)}$ is a Gaussian noise vector with variance σ_0^2. The simulated mean of the missing region is $\mathbf{z}^{(m)} = \mathbf{A}^{(m)}\mathbf{z}$ and the predicted values of the missing region are denoted as $\hat{\mathbf{z}}^{(m)} = \mathbf{A}^{(m)}\hat{\mathbf{z}}$. The NRMSE in this application is defined as

$$\text{NRMSE} = \left(\frac{\frac{1}{N-\tilde{N}} \sum_{i=1}^{N-\tilde{N}} (\hat{z}_i^{(m)} - z_i^{(m)})^2}{\frac{1}{N} \sum_{i=1}^{N} (\bar{z} - z_i)^2} \right)^{1/2}, \tag{B.7}$$

where $\hat{z}_i^{(m)}$, $z_i^{(m)}$, and z_i represent the i-th entry of $\hat{\mathbf{z}}^{(m)}$, $\mathbf{z}^{(m)}$, and $\mathbf{z}$, respectively, and $\bar{z}$ is the mean of the vector $\mathbf{z}$. Since for real data, the underlying true $\mathbf{z}$ is unknown, the NRMSE in real data analysis in Section B.8.3 is computed as in (B.7) with $\hat{z}_i^{(m)}$, $z_i^{(m)}$, z_i, and $\bar{z}$ replaced by $\hat{y}_i^{(m)}$, $y_i^{(m)}$, y_i, and $\bar{y}$, representing the i-th entry of $\hat{\mathbf{y}}^{(m)}$, $\mathbf{y}^{(m)}$, $\mathbf{y}$ and the average of $[\mathbf{y}^{(m)}, \mathbf{y}]$, respectively.

Our IKF-CG approach is compared with four popular GP approximation methods, including the Vecchia approximation [19] and Scaled Vecchia approximation (SVecchia) [21],

both configured with a model condition size of 90 and a prediction condition size of 150, nearest neighbor Gaussian process (NNGP) [63, 64] with 30 neighbors, and local approximate Gaussian process (laGP) [22, 232] with 70 neighbors. We choose a larger conditioning size of the Vecchia and SVecchia approaches, and more neighbors in NNGP and laGP than the default setting, to improve the precision of the approximation methods, despite the increased computational cost, as shown in Table B.6 and Figure B.10. All methods utilize the Matérn kernel with a roughness parameter of 5/2, except for laGP, which only allows the Gaussian kernel in the package.

B.8.2 Branin function

We first employ the Branin function to evaluate the performance of our IKF-CG algorithm in modeling incomplete lattice data [233]. The data are generated using

$$y(s_1, s_2) = \left(s_2 - \frac{5.1}{4\pi^2}s_1^2 + \frac{5}{\pi}s_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(s_1) + 10 + \epsilon, \tag{B.8}$$

where ϵ is a Gaussian noise with the variance of $\sigma_0^2 = 10^2$. We consider a lattice with dimensions $n_1 = 100$ and $n_2 = 100$ over the domain $(s_{1,i}, s_{2,j}) \in [-5, 10] \times [0, 15]$ for $1 \leq i \leq n_1, 1 \leq j \leq n_2$ in (B.8). Three scenarios are analyzed, two having 20% missing locations at disks with different centers and one having 20% randomly selected missing locations. In each scenario, we repeat the simulation $E = 20$ times to account for variability from the noise. The observations from the first simulated case of the 20% disk missing scenario are shown in Figure B.6(a), with the underlying mean shown in panel (b). The prediction of the IKF-CG algorithm is plotted in panel (c), which demonstrates a high degree of accuracy.

Figure B.7 presents violin plots of the NRMSE distribution for the missing regions

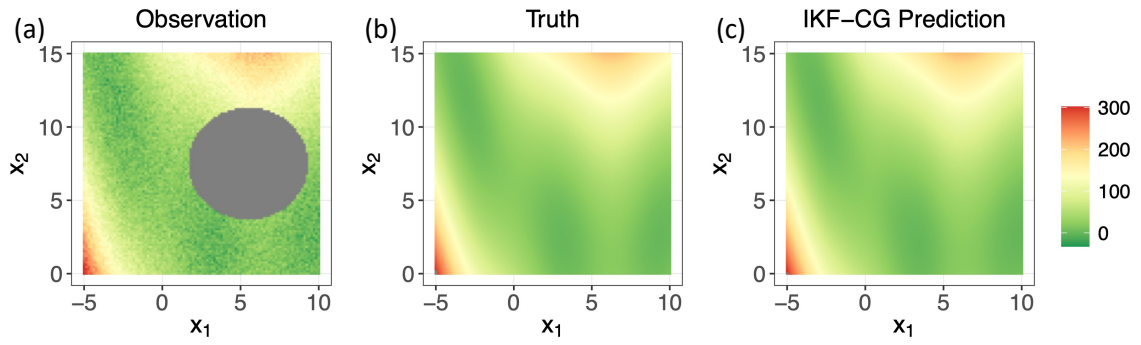

Figure B.6: (a) Observed data from a simulation of the Branin function with the first disk missing. (b) Latent mean of the Branin function. (c) Predictions obtained using the IKF-CG algorithm.

Table B.6: The average computational time in seconds over 20 experiments for estimating the parameters and predicting Branin function.

	IKF-CG	Vecchia	SVecchia	NNGP	laGP
Disk1	107.1	53.2	130.6	105.2	120.4
Disk2	108.9	52.6	142.3	105.1	120.1
Random	103.2	63.4	134.3	105.7	118.4

in each scenario, comparing IKF-CG with the other four GP approximation methods, with dots indicating the average NRMSE across all experiments. The results reveal that all methods perform worse with disk-missing data compared to random-missing data, which is expected. Across all scenarios, the IKF-CG consistently achieves the lowest NRMSE values. Specifically, the average NRMSE for the IKF-CG is 0.022, 0.022, and 0.014 for scenarios shown in Figure B.7 (a)-(c), respectively, while the average NRMSE for other methods is at least twice as large. The average computational times for each scenario are provided in Table B.6. The Vecchia method is the fastest, while the other four methods have a similar computational time. Overall, the IKF-CG method is more accurate than the alternatives for this example with a similar computational cost.

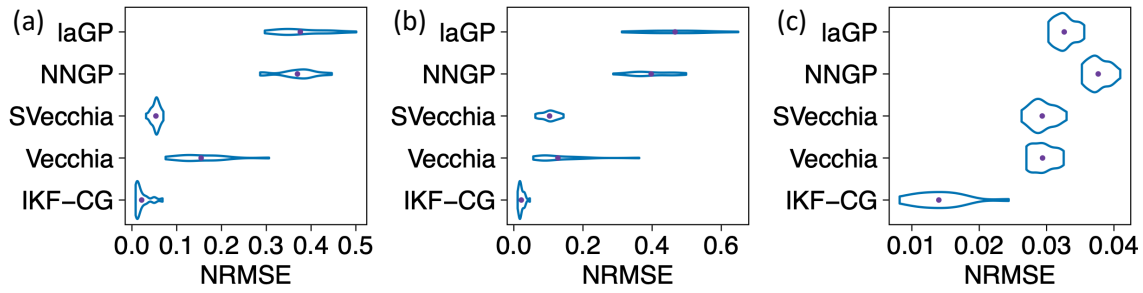

Figure B.7: Violin plots illustrating the NRMSE of missing regions for different methods under 20% disk missing with the first center (panel (a)), 20% disk missing with the second center (panel (b)), and 20% random missing (panel (c)). The point on each violin represents the mean NRMSE for each method.

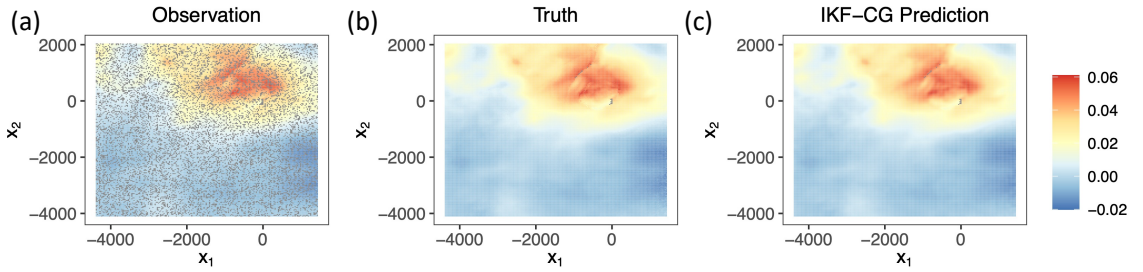

Figure B.8: (a) Observed data from interferograms with the 25% random missing. (b) True values of the interferogram data. (c) Predictions obtained using the IKF-CG algorithm.

B.8.3 Interferometric synthetic aperture radar interferogram

Interferometric synthetic aperture radar (InSAR) interferograms can measure ground deformation with centimeter-level precision, widely used for understanding geophysical processes and hazard quantification [234, 229]. We test our IKF-CG algorithm for predicting missing values in a COSMO-SkyMed satellite interferogram spanning from October 17, 2011, to May 4, 2012, for Kīlauea Volcano studied in [235]. For demonstration purposes, we investigate two types of missing data on a 200×200 lattice (disk missing and random missing) across five missing proportions (5%, 10%, 15%, 20%, and 25%).

Figure B.8(a) plots the InSAR interferograms with 25% random missing data, while panel (b) shows the complete data without any missing values, and panel (c) presents the

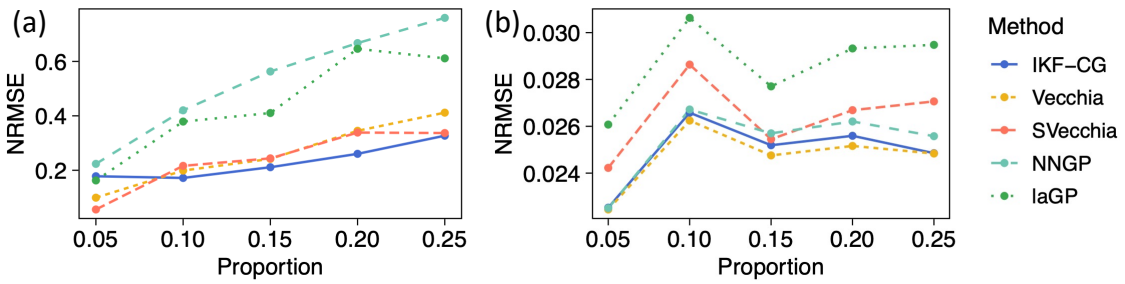

Figure B.9: The NRMSE of the missing regions in disk missing (panel (a)) and random missing (panel (b)) scenarios for interferograms.

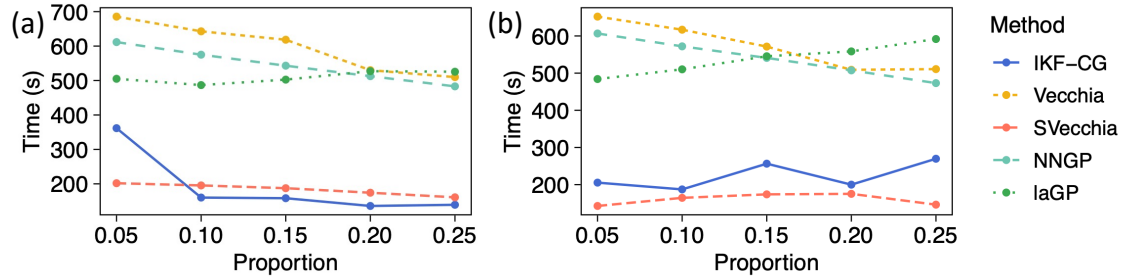

Figure B.10: Total computational time for the missing regions in disk missing (panel (a)) and random missing (panel (b)) scenarios for interferograms.

predictions obtained using the IKF-CG algorithm. The prediction of IKG-CG shows a high degree of accuracy.

Figures B.9(a) and B.9(b) illustrate the NRMSE for disk missing and random missing scenarios, respectively. The corresponding run times are provided in Figure B.10, which shows that the computational order of all approaches is similar. Generally, the predictive error is around an order of magnitude smaller for random missing data compared to disk missing data. In Figure B.9(a), the IKF-CG outperforms other methods across most scenarios, especially at higher proportions of missing data (10%, 15%, 20%, and 25%), and it is also the fastest method for these higher proportions shown in Figure B.10. Figure B.9(b) reveals that while Vecchia performs slightly better than the IKF-CG for the first four proportions in random missing scenarios, the IKF-CG achieves the lowest NRMSE for

the 25% missing scenario. The difference in NRMSE between these two methods is small. Though predicting missing values in lattice data is not the main application in our study, the results underscore the accuracy and scalability of the IKF-CG algorithm in different applications.

Appendix C

Appendix of Chapter 3

C.1 Maximum likelihood estimator with Laplace fluctuation

Here, we discuss the maximum likelihood estimator when the noise fluctuation follows the Laplace distribution. We use the observations for the x coordinate as an example, and the derivation for the y coordinate follows similarly. We denote the observation vector $\mathbf{v}_x = (\mathbf{v}_x^T(1), \mathbf{v}_x^T(2), ..., \mathbf{v}_x^T(T))^T$ with $\mathbf{v}_x(t) = (v_{1,x}(t), ..., v_{n_t,x}(t))^T$ for time frame $t = 1, ..., T$, and assume that the initial velocity $\mathbf{v}_x(0)$ is given. The likelihood function of the parameters $\mathbf{w}_x = (w_x(1), ..., w_x(T))^T$ and $\boldsymbol{\tau}_x = (\tau_x(1), ..., \tau_x(T))^T$ can be written as

$$\begin{aligned} p(\mathbf{v}_x \mid \mathbf{w}_x, \boldsymbol{\tau}_x, \mathbf{v}_x(0)) =& \prod_{t=1}^{T} p(\mathbf{v}_x(t) \mid \mathbf{v}_x(t-1), w_x(t), \tau_x(t)) \\ =& \prod_{t=1}^{T} \prod_{i=1}^{n_t} p(v_{i,x}(t) \mid \bar{v}_{i,x}(t-1), w_x(t), \tau_x(t)) \\ =& (\sqrt{2}\tau_x(t))^{-\sum_{t=1}^{T} n_t} \exp\left(- \sum_{t=1}^{T} \sum_{i=1}^{n_t} \frac{\sqrt{2}\,|e_{i,x}(t)|}{\tau_x(t)} \right), \end{aligned}$$

where the residual of the i-th particle at time frame t is defined as $e_{i,x}(t) = v_{i,x}(t) - w_x(t)\bar{v}_{i,x}(t-1)$.

For any time frame t, the maximum likelihood estimator of $w_x(t)$ is equivalent to the least absolute deviation (LAD) regression below:

$$\hat{w}_x(t) = \underset{\tau_x(t), w_x(t)}{\operatorname{argmin}} \sum_{i=1}^{n_t} |v_{i,x}(t) - w_x(t)\bar{v}_{i,x}(t-1)|.$$

Although there is no closed-form solution for the estimator in the LAD regression, fast algorithms, such as the Barrodale and Roberts (BR) algorithm [148], that can transform the LAD regression into a linear programming problem, are available. The BR algorithm is available in standard software platforms. For instance, the package "L1pack" in the Comprehensive R Archive Network (CRAN) implements the BR algorithm to solve a LAD problem. After obtaining the estimator $\hat{w}_x(t)$, we substitute it into the likelihood function and maximize the profile likelihood to obtain the maximum likelihood estimator of $\tau_x(t)$, which is given in Equation (3.4).

C.2 Maximum likelihood estimator with generalized Gaussian fluctuation

Next, we discuss the maximum likelihood estimator for models with generalized Gaussian fluctuation. To do so, we denote three vectors of parameters $\mathbf{w}_x$, $\boldsymbol{\alpha}_x$, and $\boldsymbol{\beta}_x$, each having T dimensions. The logarithm of the likelihood function follows

$$\log(p(\mathbf{v}_x \mid \mathbf{w}_x, \boldsymbol{\beta}_x, \boldsymbol{\alpha}_x)) = \sum_{t=1}^{T} \left\{ n_t \log\left(\frac{\beta_x(t)}{2\alpha_x(t)\Gamma\left(\frac{1}{\beta_x(t)}\right)} \right) - \sum_{i=1}^{n_t} \left(\frac{|e_{i,x}(t)|}{\alpha_x(t)} \right)^{\beta_x(t)} \right\},$$

where $e_{i,x}(t) = v_{i,x}(t) - w_x(t)\bar{v}_{i,x}(t-1)$ and $\Gamma(\cdot)$ denotes the Gamma function, with $w_x(t) \in \mathbb{R}$, $\alpha_x(t) \in \mathbb{R}^+$ and $\beta_x(t) \in \mathbb{R}^+$.

We first differentiate the likelihood function with respect to $\alpha_x(t)$ and set it to zero. For any given $\beta_x(t)$ and $w_x(t)$, the likelihood is maximized when

$$\hat{\alpha}_x(t) = \left(\frac{\beta_x(t)}{n_t} \sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)} \right)^{\frac{1}{\beta_x(t)}}. \tag{C.1}$$

After substituting the $\hat{\alpha}_x(t)$ from Equation (C.1) into the log-likelihood function, we obtain the logarithm of the profile likelihood of $(\boldsymbol{w}_x, \boldsymbol{\beta}_x)$:

$$\begin{aligned} \log(p(\mathbf{v}_x \mid \mathbf{w}_x, \boldsymbol{\beta}_x, \hat{\boldsymbol{\alpha}}_x)) = \sum_{t=1}^{T} \Bigg\{ & n_t \log \left(\frac{\beta_x(t)}{2\Gamma(\frac{1}{\beta_x(t)})} \right) \\ & - \frac{n_t}{\beta_x(t)} \log \left(\frac{\beta_x(t)}{n_t} \sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)} \right) - \frac{n_t}{\beta_x(t)} \Bigg\}. \end{aligned} \tag{C.2}$$

Denoting the logarithm of the profile likelihood by $\ell(\mathbf{w}_x, \boldsymbol{\beta}_x) = \log(p(\mathbf{v}_x \mid \mathbf{w}_x, \boldsymbol{\beta}_x, \hat{\boldsymbol{\alpha}}_x))$ and differentiating Equation (C.2) with respect to $w_x(t)$ and $\alpha_x(t)$, we then find

$$\frac{\partial \ell(\mathbf{w}_x, \boldsymbol{\beta}_x)}{\partial w_x(t)} = \frac{n_t \sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)-1} \bar{v}_{i,x}(t-1) \operatorname{sgn}(e_{i,x}(t))}{\sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)}}, \tag{C.3}$$

$$\begin{aligned} \frac{\partial \ell(\mathbf{w}_x, \boldsymbol{\beta}_x)}{\partial \beta_x(t)} = & \frac{n_t}{\beta_x(t)} + \frac{n_t}{\beta_x^2(t)} \Psi \left(\frac{1}{\beta_x(t)} \right) + \frac{n_t}{\beta_x^2(t)} \log \left(\frac{\beta_x(t)}{n_t} \sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)} \right) \\ & - \frac{n_t}{\beta_x(t)} \frac{\sum_{i=1}^{n_t} \left(|e_{i,x}(t)|^{\beta_x(t)} \log |e_{i,x}(t)| \right)}{\sum_{i=1}^{n_t} |e_{i,x}(t)|^{\beta_x(t)}}, \end{aligned} \tag{C.4}$$

where $\Psi(z) = \Gamma'(z)/\Gamma(z)$ stands for the ratio between the derivative of a Gamma function and a Gamma function for any z, and $\operatorname{sgn}(e_{i,x}(t))$ denotes the sign of $e_{i,x}(t) = v_{i,x}(t) - w_x(t)\bar{v}_{i,x}(t-1)$. Thereafter, we iteratively maximize the likelihood function with respect to $w_x(t)$ and $\beta_x(t)$ using the profile likelihood in Equation (C.2) and closed-form derivative in Equations (C.3) and (C.4) by the low-storage quasi-Newton optimization method (L-BFGS) [236]. Note that, for each t, we only need to iteratively maximize the log profile likelihood

with respect to two parameters, making the computational procedure both fast and robust.

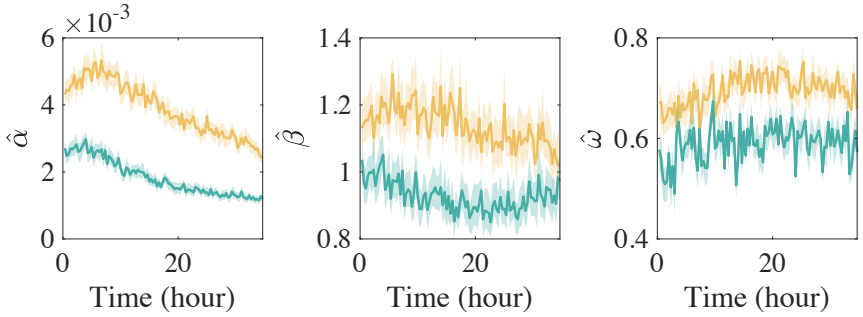

Figure C.1: Parameter estimation with interval (shaded) for a Generalized Gaussian distribution following Equation (3.5) along x-direction (yellow), and along y-direction (green).

The maximum likelihood estimators of parameters $\alpha_x(t)$, $\alpha_y(t)$, $\beta_x(t)$, $\beta_y(t)$, $w_x(t)$ and $w_y(t)$ assuming the fluctuation follows the generalized Gaussian distribution, are shown in Figure C.1. The estimated $\beta_x(t)$ and $\beta_y(t)$ are both close to 1, which means that the fluctuation is closer to the Laplace distribution. In addition, $\beta_y(t)$ is typically smaller than $\beta_x(t)$, as there is more confinement in the y direction because of the substrate-imposed directionality. Furthermore, both $w_x(t)$ and $w_y(t)$ are smaller than 1 for any t, consistent with the findings when assuming the fluctuation follows the Laplace distribution.

To demonstrate the impact of parameter selection on the order parameter, we investigate the 2D parameter space defined by the ratios ω_x/ω_y and τ_x/τ_y. Figure 4.1 illustrates one possible path of order development, using a selected set of parameters τ_x=0.01 and ω_x=0.7. Next, we show that when τ_x=0.01 and τ_x=0.04, encompassing the experimentally observed values range, the order parameter exhibits consistent behavior along the temporal course of the experiment for different τ_x values (= 0.01, 0.02, 0.04) [Figure C.2(a) and Figure C.2(b)]. To determine the order parameter, we average the last 50 steps in the simulation, as it reaches a plateau after the first 20 steps (Figure C.3).

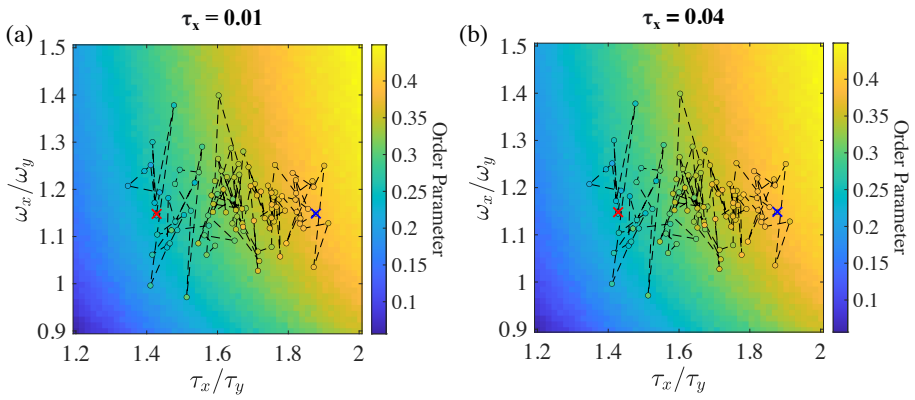

Figure C.2: (a-b) Change of order parameters due to the change of simulation parameters with $\tau_x = 0.01$ in (a) and $\tau_x = 0.04$ in (b), which cover the range of the observed τ's. $w_x = 0.7$ are assumed for both cases. Red and blue X's denote the beginning and the end of the trajectory.

C.3 Details on simulation

We first briefly introduce the setup of the simulation: Cells are modeled as particles, and their initial velocities and positions are imported from the data. For simplicity, we assume that the cell number remains constant. We start by partitioning the space into grids, and the spacing between gridlines is no smaller than the cell-cell interaction radius r. In our simulation, grids are populated with cells based on cells' positions $\mathbf{s}_i(t) = (s_{i,x}(t), s_{i,y}(t))^T$ (Figure 3.1(e)). The grid-based approach [126] improves computational efficiency for searching neighbors in simulation since as such, identifying a cell's neighbors only requires searching the nine grids surrounding it. Using this method, computing the likelihood function for parameter estimation and simulation only requires $\mathcal{O}(\sum_{t=1}^{T} n_t)$ operations, which is much faster than the conventional approach requiring $\mathcal{O}(\sum_{t=1}^{T} n_t^2)$ operations, where n_t is the number of cells at time frame t. We note that this coarse-graining strategy to search for neighbors only improves computational efficiency, but does not change the simulation results.

The velocities $v_{i,x}(t)$ and $v_{i,y}(t)$ are updated according to Equations (3.2)-(3.3), with three types of distribution of fluctuations due to the cell-substrate interaction: Gaussian,

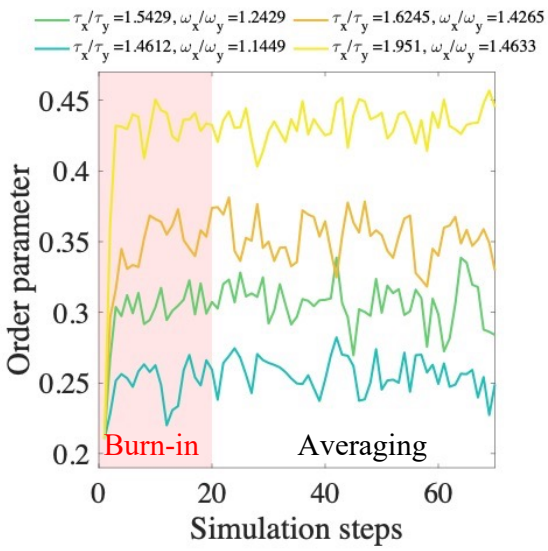

Figure C.3: Several typical simulation progressions, where the first 20 steps are regarded as burn-ins and left out in computing the averages in the phase diagram.

Laplace, or GGD; two types of neighbors: interactions with all neighboring cells or only neighboring cells with the same direction velocity, and weights (ω_x and ω_y) of the interaction fixed to be 1 or estimated from the data, which are due to cell-cell interactions. All parameters are estimated based on the maximum likelihood estimators introduced in Section 3.3.

C.4 Additional results on experimental data

The four new features are used to construct a minimum physics model that can quantitatively capture the progression of the orientational order parameter and velocity distribution, for various experiments with different initial cell densities and liquid crystal elastomer substrates. The results of several additional experiments are also analyzed to validate the generality of our algorithm (Figure C.4-C.5). Similar to the main text, we evaluate models by velocity orientational order parameter and the average absolute deviation of the velocity at each time point.

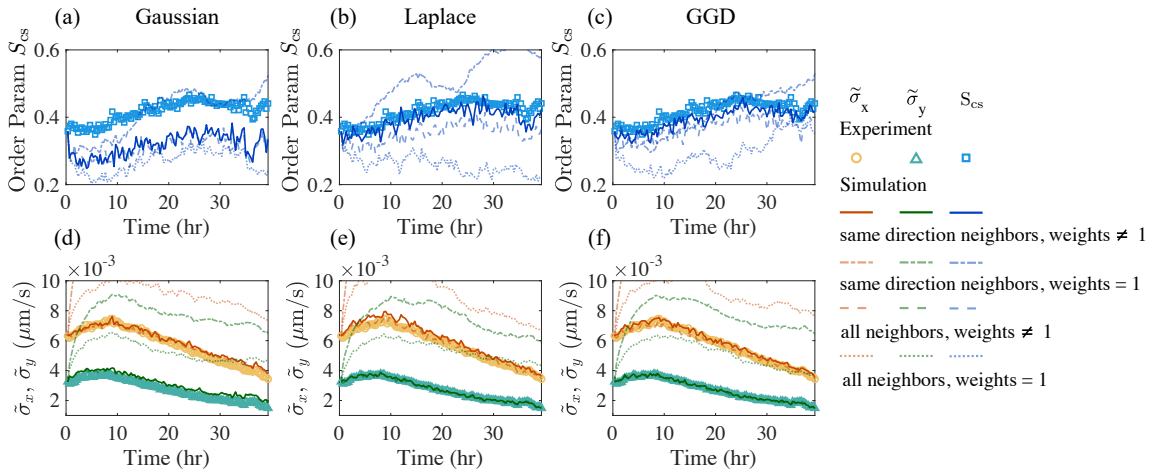

Figure C.4: Reproducing orientational order parameter (a-c) and average absolute deviation of velocities (d-f) from different simulation models at a representative radius $r = 75$ μm for experiments of cell migrating on a nematic substrate when starting at a higher initial density.

The experimental setup in Figure C.4 has the same substrate material as the one presented in Figure 3.6. Except that at the start of imaging, cell density is higher. During the course of the experiment, the cell density ρ grows from $\rho = 500$ to 650 mm^{-2}. The simulation model with selected features can accurately capture the progression of orientational order parameter and velocity magnitude, shown in upper and lower panels in Figure C.4. Models without all selected features cannot reproduce some of the properties. In particular, if Gaussian fluctuation is used with the other three features (blue curve in panel (a)), the model substantially underestimates the orientational order parameter, while the model with Laplace or GGD fluctuations, shown by blue curves in panel (b) and (c), respectively, reproduces the progression of these parameters reasonably well. Furthermore, all approaches incorporating estimated weights and considering neighbor ensembles of only cells traveling in the same direction capture the average absolute deviation of velocities $\tilde{\sigma}_x$ and $\tilde{\sigma}_y$ reasonably well.

On the other hand, results presented in Figure C.5 are derived from cell movements

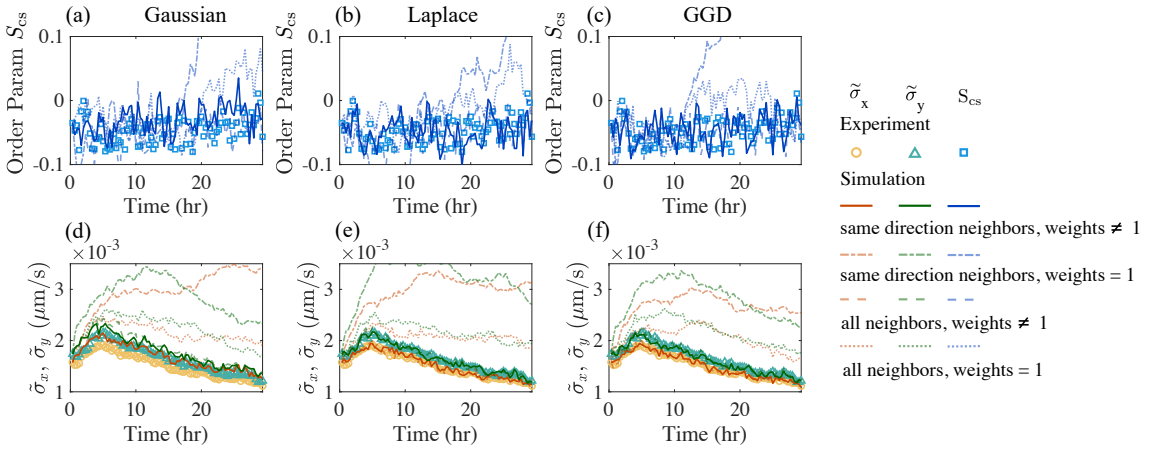

Figure C.5: Reproducing orientational order parameter (a-c) and average absolute deviation of velocities (d-f) with different simulation models at a representative radius $r = 75$ μm for experiments of cells moving on an isotropic substrate, which do not align, and have isotropic velocity.

on an isotropic substrate, during this time, cell density changes from $\rho = 420$ to 470 mm^{-2}. We have observed that, on the isotropic substrate, we reproduce order parameters that are around zero with any type of fluctuations (Figure C.5(a-c)), signaling the lack of order, and the velocity variance is also isotropic (Figure C.5(d-f)). It can be noted that when the cell moves on an isotropic substrate, choices of neighbor, fluctuation distributions, and weights become less important in fitting the order parameter (Figure C.5(a-c)). Nonetheless, the absolute deviation$\tilde{\sigma}_x$ and $\tilde{\sigma}_y$ change over time due to proliferation. Approaches that utilize estimated weights and consider ensembles of neighboring cells traveling in the same direction effectively capture and accommodate these changes.

C.5 Additional results on simulation using different neighbor radii

Here, we explore the effect of radii in reproducing the velocity order parameter for the entire monolayer. Given the cell width ≈ 20 μm and length ≈ 100 μm in projection, we

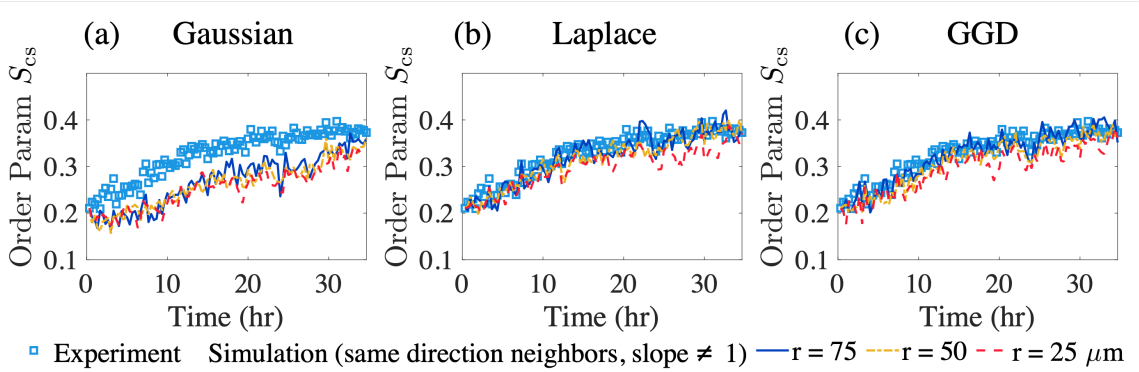

Figure C.6: Reproducing orientational order parameters with different radii. The left, middle, and right panels compare simulations using the Gaussian, Laplace, and GGD fluctuation with experimental observations. Open square symbols show the order parameter calculated from experimental data, dark blue solid line, yellow dash-dotted line, and red dashed line represent simulation results of different radii 75, 50, and 25 μm.

explore length scales comparable to these dimensions. Below, we plot the order parameters reproduced by using different pair-wise distances: 25, 50, and 75 μm, in simulation (Figure C.6). All simulations are reproduced by including only the same direction neighbors and applying weights less than 1, as discussed before. Our results show that the variation in r plays a much smaller role than changing the model for fitting the velocity fluctuations.

Appendix D

Appendix of Chapter 4

D.1 Methods

D.1.1 Materials

All reagents were used as received except where noted. Ethyl α-bromoisobutyrate (EBiB), copper (II) bromide (CuBr2), 2,2,2-trifluoroethanol (TFE), and 2-trifluoromethyl-2-propanol were (TFMP) were purchased from Sigma Aldrich. Tris[2-(dimethylamino) ethyl]amine (Me6Tren) was obtained from Alfa Aesar. Dodecyl acrylate (DDA, $>$ 98%), 2,2,3,3,3-pentafluoropropyl acrylate, 1H,1H,2H,2H-nonafluorohexyl acrylate, and 2,2,3,3,4,4,5,5,6,6,7,7-dodecafluoroheptyl acrylate were obtained from TCI chemicals. Monomers were passed through a column of basic alumina to remove inhibitor prior to use. Deuterated solvents were acquired from Cambridge Isotope Laboratories. 2-fluoroethyl acrylate was synthesized and stored in a solution of hexanes. All solvents were ACS grade or better and used without further purification.

D.1.2 Molecular Characterization

The photo-mediated ATRP polymerization light source (UV: $\lambda_{max} \approx 360nm$) was a commercial nail curing lamp (Thermal Spa, obtained online from Amazon) equipped with 3 × 16 W bulbs. Solution state 1H nuclear magnetic resonance (NMR) spectra were recorded on a Varian 600 MHz spectrometer. Chemical shifts (δ) are reported in ppm relative to residual protio-solvent in CDCl3 (7.26 ppm). NMR measurements were performed with samples concentrations of 10–20 mg/mL. Size-exclusion chromatography (SEC) was conducted on a Waters Alliance HPLC System, 2690 Separation Module using chloroform with 0.25% triethylamine as the eluent with a flow rate of 0.35 mL/min. Refractive index traces from a Waters 2410 Differential Refractometer detector were used for estimates of the molar mass and dispersity relative to linear polystyrene standards with a chloroform mobile phase.

Automated flash chromatography was performed on a Biotage Selekt unit installed with an external evaporative light scattering detector (ELSD), using Biotage Sfär cartridge series (50 g/100 g) eluted with suitable hexanes/ethyl acetate solvent gradients. All chromatographic solvents were ACS grade or better and used without further purification. In fractionation experiments, the mass recovery was taken as the ratio of the collected mass to the injected mass.

SAXS measurements were collected on a custom-built high-brilliance laboratory beamline for small and wide-angle X-ray scattering (SAXS/WAXS) at the BioPACIFIC Materials Innovation Platform at UCSB. The instrument is constructed using a high-brightness liquid metal jet X-ray source (D2+ 70 kV from Excillum), a low background scatterless slit beam collimation system developed in-house, and a 4-megapixel hybrid photon counting area detector (Eiger2 R 4M from Dectris) housed inside a 3 meter-long vacuum

vessel. The combination of the high brightness source and a large area detector results in more than 10X reduction in data collection times compared to standard lab-based SAXS/WAXS instruments, enabling high throughput nano-structural characterization of a wide range of samples. 2D-data were reduced to a one-dimensional form of intensity (arbitrary units) as a function of the magnitude of the wave vector $q = |q| = 4\pi \sin(\phi/2)/\lambda_{\text{xray}}$, where λ_{xray} is the X-ray wavelength and ϕ is the scattering angle. Samples consisted of 5–10 mg of bulk polymer pressed within a metal washer backed with Kapton tape annealed at 115 ºC for 10 minutes, then 70 ºC for 12 hours under vacuum (200 mTorr).

D.1.3 Procedure for the synthesis of 2-fluoroethyl acrylate

In a round bottom flask, 2-fluoroethanol (8 g, 125 mmol) was dissolved in dry dichloromethane (210 mL) containing triethylamine (19 g, 187 mmol). This solution was cooled in an ice bath. Acryloyl chloride (17 g, 187 mmol) was then added slowly to the cooled alcohol solution, which gradually turned a light-yellow color. The mixture was stirred in an ice bath for an additional 4 h and then reacted overnight. The solution was transferred to a separatory funnel, washed with 1 M KOH (100 mL), 1 M HCl (100 mL), and brine solution (100 mL), dried over MgSO4, gravity filtered, and concentrated in vacuo. The resulting residue was purified by flash column chromatography using a hexanes/ethyl acetate gradient to yield the monomer as a colorless liquid in a hexane solution.

D.1.4 General procedure for the synthesis of poly(dodecyl acrylate) homopolymer

A solution of Cu(II)Br2 (5 mg, 0.022 mmol) and Me6Tren (36 μL, 0.13 mmol) was prepared in 1 mL of TFE. In a scintillation vial containing EBiB (167 mg, 0.86 mmol), dodecyl acrylate (7.3 g, 30 mmol) was dissolved in a mixture of 6 mL of toluene, 0.7 mL

catalyst stock solution, and 0.8 mL of additional TFE. The vial was capped with a septum and the solution was degassed with nitrogen for 15 minutes. With stirring, the polymerization mixture was then irradiated (λ_{xray} = 360 nm) in a commercial UV nail lamp system until 70% conversion was achieved as monitored by 1H NMR. The resulting polymer precursor was purified via dissolution in a minimal amount of toluene and precipitated into cold methanol five times.

D.1.5 General procedure for the synthesis of fluorinated diblock copolymers

A solution of Cu(II)Br2 (5 mg, 0.022 mmol) and Me6Tren (36 μL, 0.13 mmol) was prepared in 1 mL of fluorinated alcohol (TFMP or TFE). In a scintillation vial containing poly(dodecyl acrylate) homopolymer (1 g, 0.12 mmol) dissolved in 2 mL of toluene, fluorinated monomer (5.3 mmol) was dissolved in a mixture of 0.3 mL of fluorinated alcohol (TFMP or TFE) and 0.4 mL catalyst stock solution. The vial was capped with a septum and the solution was degassed with nitrogen for 15 minutes. With stirring, the polymerization mixture was then irradiated (λ_{xray} = 360 nm) in a commercial UV nail lamp system until 70% conversion was achieved as monitored by 1H NMR. The resulting polymer precursor was purified via dissolution in a minimal amount of toluene and precipitated into cold methanol five times or column chromatography using a hexanes/ethyl acetate gradient. Note that TFE was used as a solvent for polymerization of 2-fluoroethyl acrylate. TFMP was used to polymerize 2,2,3,3,3-pentafluoropropyl acrylate, 1H,1H,2H,2H-nonafluorohexyl acrylate, and 2,2,3,3,4,4,5,5,6,6,7,7-dodecafluoroheptyl acrylate.

D.1.6 General procedure for fractionation of diblock copolymers via automated flash chromatography

Automated flash chromatography was performed using a Biotage Selekt purification system equipped with an external evaporative light scattering detector (ELSD). Biotage Sfär 50/100 g cartridges were used with a flow rate of 40 mL/min. For separations of less than 1.5 g of polymer, a 50 g column was used, whereas separations greater than 1.5 g of polymer required a 100 g column. The column was equilibrated with two column volumes of hexanes. After equilibration was complete, the parent diblock copolymers were dissolved in dichloromethane and loaded directly onto the chromatography column. The parent diblock copolymers were eluted with a programmed hexanes/ethyl acetate gradient. All chromatographic solvents were ACS grade or better and used without further purification. Fractions were monitored by an evaporative light-scattering detector and collected in 22 mL increments. The overall mass recovery was determined as the ratio between the collected and injected mass. Volume fractions and molecular weights of the fractionated materials were calculated by 1H NMR.

D.1.7 Logarithmic space transformation

The original intensity from SAXS data is denoted as a n-dimensional vector $\tilde{\mathbf{y}} = [\tilde{y}(q_1), ..., \tilde{y}(q_n)]^T$, where each component corresponds to the intensity at different magnitudes of wave vectors q_j. The associated uncertainty, which represents the standard deviation of the scattering intensity, is expressed as $\tilde{\boldsymbol{\sigma}}_0 = [\tilde{\sigma}_0(q_1), ..., \tilde{\sigma}_0(q_n)]^T$.

Given that our analysis operates within the logarithmic intensity space, we define the logarithm with base 10 of the intensity as $\mathbf{y} = [y(q_1), ..., y(q_n)]^T$, where $y(q_j) = \log_{10}(\tilde{y}(q_j))$. The next step involves approximating the variance of noise in this logarithmic space using the variance of the original measurements, where we utilize the Taylor expansion to approximate

the logarithm of the random variable $\tilde{Y}(q_j)$, representing the intensity at q_j, which has a mean $\tilde{\mu}(q_j)$ and variance $\tilde{\sigma}_0^2(q_j)$:

$$\begin{aligned}
\log_{10}(\tilde{Y}(q_j)) &= \log_{10}(\tilde{\mu}(q_j) + \tilde{Y}(q_j) - \tilde{\mu}(q_j)) \\
&\approx \log_{10}(\tilde{\mu}(q_j)) + \frac{\tilde{Y}(q_j) - \tilde{\mu}(q_j)}{\tilde{\mu}(q_j)\log_e(10)} - \frac{(\tilde{Y}(q_j) - \tilde{\mu}(q_j))^2}{2\tilde{\mu}(q_j)^2\log_e^2(10)} + \dots \\
\mathbb{E}[\log_{10}(\tilde{Y}(q_j))] &\approx \log_{10}(\tilde{\mu}(q_j)) - \frac{\tilde{\sigma}_0^2(q_j)}{2\tilde{\mu}^2(q_j)\log_e^2(10)} \\
\mathrm{Var}(\log_{10}(\tilde{Y}(q_j))) &\approx \frac{\tilde{\sigma}_0^2(q_j)}{\tilde{\mu}^2(q_j)\log_e^2(10)}
\end{aligned}$$

To employ these approximations, an estimate of the mean of the random variable $\tilde{Y}(q_j)$ is required. In our dataset, given the relatively small standard deviation, we approximate the mean $\tilde{\mu}(q_j)$ using $\tilde{y}(q_j)$, a sample from $\tilde{Y}(q_j)$. Consequently, the approximation of the noise variance in the log space is denoted as $\mathbf{e}_0^2 = [e_0^2(q_1), ..., e_0^2(q_n)]^T$, where $e_0^2(q_j) = \tilde{\sigma}_0^2(q_j)/(\tilde{y}^2(q_j)\log_e^2(10))$.

D.1.8 Noise filtering with Kalman filter

In diblock copolymer analysis, discerning the true signal from noise is crucial, especially for precise peak detection. Here, we model the observed logarithmic intensities $\mathbf{y}$ with a Gaussian process with Matérn kernel [9, 45] and roughness parameter 5/2 (Equation (1.2)) to filter out the noise given the provided noise variance in the data. Since the input (the wave vector magnitude q_j) is one-dimensional, we can apply the Kalman filter and Rauch-Tung-Striebel (RTS) smoother, summarized in Lemmas 1.1 and 1.2 in Section 1.2, to efficiently perform this filtering. The noise variance provided in the data is heterogeneous,

so instead of an identity noise covariance matrix, the model takes the form:

$$\mathbf{y} \sim \mathcal{MN}\left(\mathbf{0}, \sigma^2\mathbf{R} + \mathbf{\Sigma}_0\right) = \mathcal{MN}\left(\mathbf{0}, \sigma^2(\mathbf{R} + \eta\,\mathbf{\Lambda})\right), \tag{D.1}$$

where σ^2 is the variance, $\mathbf{\Sigma}_0 = \text{diag}(\sigma_0^2(q_1), \ldots, \sigma_0^2(q_n))$, $\mathbf{\Lambda} = \text{diag}(e_0^2(q_1), \ldots, e_0^2(q_n))$, with the entries $e_0^2(q_j)$ computed in Section D.1.7. When apply the Kalman filter for smoothing, we set $\sigma_{0,j}^2 = \sigma_0^2(q_j)$ in Equation (1.18), with index t replaced by j. After obtaining the predictive mean $\mathbf{s}_j$ using the RTS smoother in Equation (1.23), the filtered intensity at each q_j is recovered by computing $\mathbf{F}_j\mathbf{s}_j$.

Two parameters of interest are η in Equation (D.1) and γ in the kernel function (1.2). To estimate these parameters, we adopt a strategy of maximizing the sum of log likelihoods across all DIS samples. We choose DIS samples for their inherent smoothness compared to other morphologies, enabling them to more effectively filter out noise. Alternatively, one could opt to maximize the sum of likelihoods across all samples or focus on maximizing the likelihood of a specific intensity curve. Once other parameters have been estimated, the variance parameter σ^2 can be determined through maximum likelihood estimation, given by $\hat{\sigma}^2 = \mathbf{y}^T(\mathbf{R} + \eta\mathbf{\Lambda})^{-1}\mathbf{y}/n$.

D.1.9 Peak detection and feature selection

Reducing the original intensity curves to a handful of informative features is important for later classification models. A primary feature set we consider is the pairwise ratios of peak locations. This choice is grounded in the traditional manual approach to morphology determination, where these ratios play a pivotal role. To facilitate this, our first task is to reliably detect all peaks across each intensity curve.

Our peak detection method begins by identifying potential peaks, $\tilde{P} = \{\tilde{p}_1, ..., \tilde{p}_{\tilde{f}}\}$,

where the smoothed logarithmic intensity changes from an increasing to a decreasing trend and recording the peak heights as $\tilde{h}_1, ..., \tilde{h}_{\tilde{f}}$. To ensure accuracy, we exclude peaks that appear at the initial or final segments of the curve, as these are often subject to varying conditions and may not represent true peaks. Next, we discard those peaks where the ratio of their height to the height of the subsequent peak is less than a certain threshold r_c, recognizing these as noise. We then retain only those peaks exceeding a predetermined height threshold h_c, which ensures that we focus on the most significant features. Write it formally as:

$$P = \{\tilde{p}_i \in \tilde{P} : \tilde{h}_i/\tilde{h}_{i+1} > r_c \text{ and } \tilde{h}_i > h_c\}.$$

In scenarios where the first two detected peaks are close, we keep only the peak with the greater height. The locations of the remaining detected peaks are denoted as $p_1, p_2, ..., p_f$ for f detected peaks (potential features), with each $p_l = q_j$ for some j. In this chapter, we use the pairwise ratio between the first three peaks, $p_2/p_1, p_3/p_1$, and p_3/p_2, as they capture essential characteristics without including too much noise.

Additionally, three features were incorporated into the analysis: the position of the first peak, its width, and its sharpness. The sharpness, defined by the second derivative, reflects the peak's curvature. Assuming the first peak occurs at $p_1 = q_s$ for some s, we define it as:

$$y''(p_1) = y''(q_s) \approx \frac{y(q_{s+1}) - 2y(q_s) + y(q_{s-1})}{(\Delta q)^2}, \tag{D.2}$$

where $\Delta q = |q_s - q_{s-1}|$ is the spacing between successive points. The width of the first peak, w_1, is determined by the range over which the second derivative stays negative around the

peak:

$$
\begin{aligned}
j_{\text{left}} &:= \max_j \{j < s : y''(q_j) \geq 0, y''(q_{j+1}) < 0\}, \\
j_{\text{right}} &:= \min_j \{j > s : y''(q_j) \geq 0, y''(q_{j-1}) < 0\}, \\
w_1 &= j_{\text{right}} - j_{\text{left}}.
\end{aligned} \tag{D.3}
$$

These steps allow us to refine the intensity curves into a set of six physics-informed morphological features (PIMF), independent of the initial chemical identities of the diblock copolymer, making it a universal feature set. Thus, it enables reliable morphology prediction for copolymers, even those composed of new chemical identities.

D.1.10 Machine learning models for classification

The final step involves the construction of classification models. We primarily utilize bagging, random forest (RF), and extreme gradient boosting (XGBoost) techniques, though other models can also be applied. Our approach first categorizes all test samples with no detected peaks as DIS. We then train the model using samples that exhibit at least one peak and apply the model to predict the morphologies of the remaining test samples.

We evaluate our methodology under two different scenarios. In the first scenario, models trained on data from three chemical groups are employed to predict the phases of a fourth, previously unseen chemical group. The second scenario involves combining data from all four chemical groups and implementing 5-fold cross-validation for evaluating model performance. Across these scenarios, we observe that all three methods with PIMF demonstrate comparable performance.

D.2 More numerical results of machine learning models for classification

D.2.1 Phase identification of new chemical groups

Here we discuss the importance of different predictors in the RF model, where the importance of each variable is quantified by the mean decrease in model accuracy that results from permuting the values of that variable. This analysis, applied to the phase prediction of diblock copolymers from the third and fourth groups, is depicted in Figure D.1. The plot demonstrates that the ratio between the locations of the third and first peaks stands out as the paramount factor for precise phase behavior prediction in both groups.

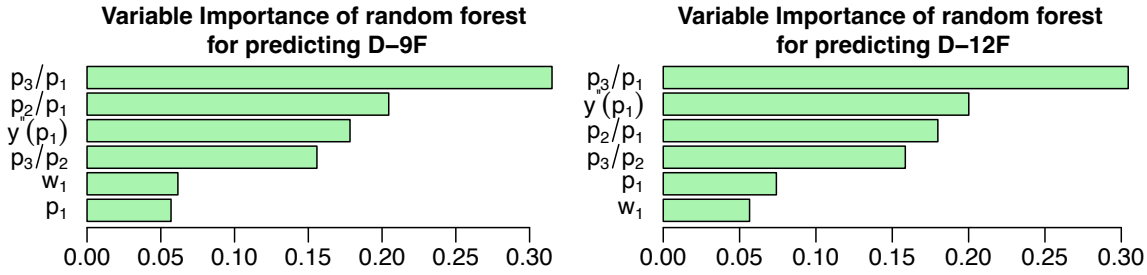

Figure D.1: Variable importance of the PIMF-RF model in predicting the third group (left) and the fourth group (right).

Additionally, we show the performance of the bagging model, with numerical results provided in Table D.1 and Figures D.2 - D.3 all using the constructed PIMF as inputs. Similarly, the performance of the XGBoost is shown in Table D.2 and Figures D.4 - D.5. The analysis reveals that the bagging model achieves accuracy comparable to that of the random forest model, whereas the XGBoost model exhibits slightly better performance in predicting the phase behavior of the new copolymer. Furthermore, for both the bagging and XGBoost models, the ratio p_3/p_1 is identified as the most significant feature for classification. Note

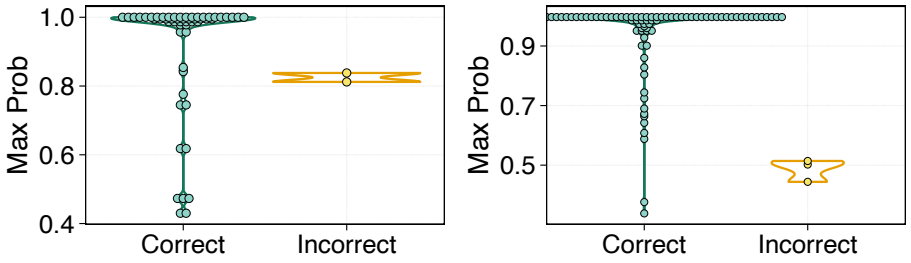

Figure D.2: Maximum predicted probability for correctly and incorrectly predicted samples of D-9F group (left) and D-12F group (right) using PIMF-bagging model.

that the feature importance in XGBoost is defined differently than the other two models, which quantifies the fractional contribution of each feature to the model, calculated from the total gain of the splits where the feature is used. A higher percentage indicates a more significant role in improving the model's predictive accuracy.

Table D.1: The confusion matrix of PIMF-bagging in predicting the third group (D-9F) (top) and the fourth group (bottom).

		pred DIS	σ	HEX	GYR	LAM	# Misclassified
true	DIS	10	1				1
	HEX			12			0
	GYR				7		0
	LAM			1		26	1

		pred DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	12						0
	BCC		5					0
	HEX		1		38			1
	GYR					2		0
	LAM			1	1		31	2

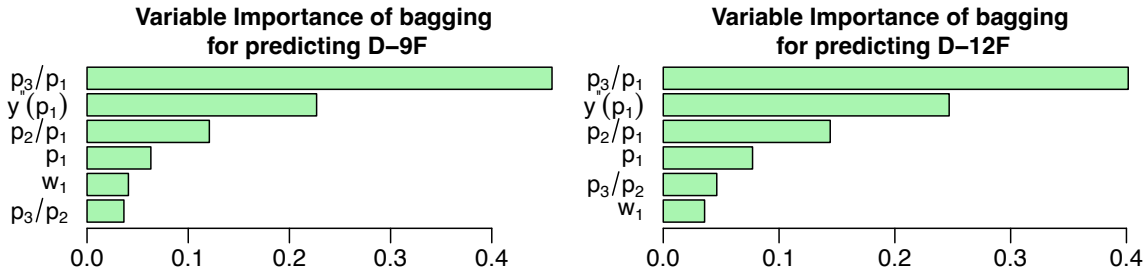

Figure D.3: Variable importance of PIMF-bagging in predicting the third group (left) and the fourth group (right).

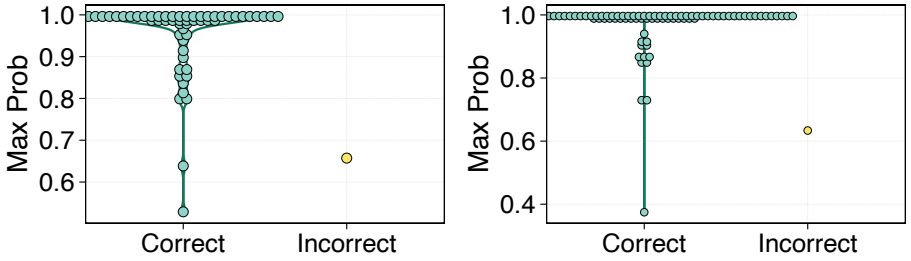

Figure D.4: Maximum predicted probability for correctly and incorrectly predicted samples of D-9F group (left) and D-12F group (right) using PIMF-XGBoost model.

Table D.2: The confusion matrix of PIMF-XGBoost in predicting the fourth group (D-12F) (top) and the fourth group (bottom).

		pred DIS	σ	HEX	GYR	LAM	# Misclassified
true	DIS	10	1				1
	HEX			12			0
	GYR				7		0
	LAM					27	0

		pred DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	12						0
	BCC		5					0
	HEX				39			0
	GYR					2		0
	LAM			1			32	1

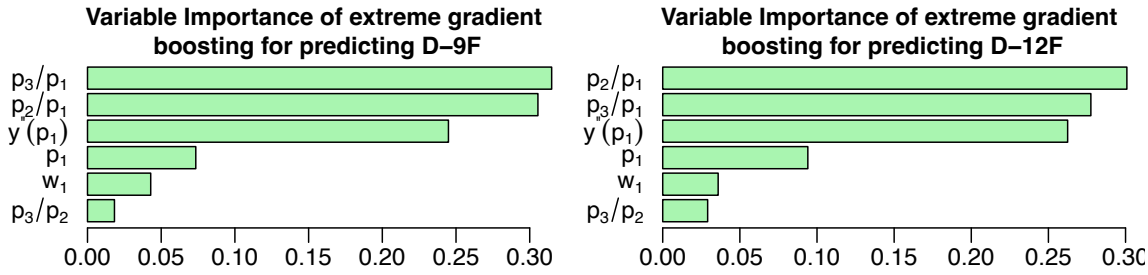

Figure D.5: Variable importance of PIMF-XGBoost in predicting the third group (left) and the fourth group (right).

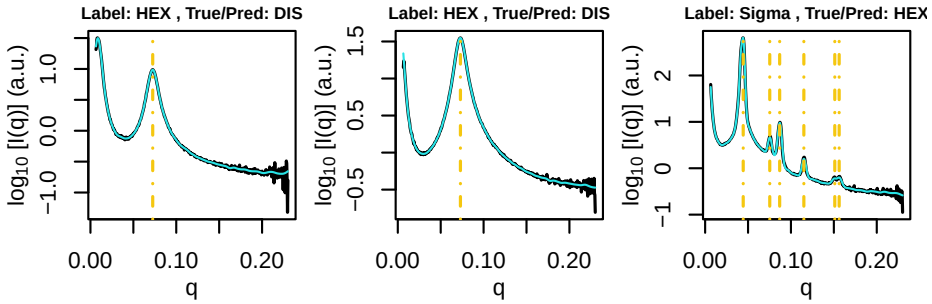

Figure D.6: Three mislabeled samples during manual assignment in the dataset.

D.2.2 Phase identification of mixed chemical groups

The three mislabeled samples during initial manual assignment are shown in Figure D.6. The classification results of RF using PIMF and chemistry-dependent (CD) features in predicting the phase behavior across all four groups, utilizing 5-fold cross-validation are reported in Table D.3.

Table D.3: The prediction results of the PIMF-RF, CD-RF [1], and Curve-RF on predicting mixed chemical groups with out-of-sample cross-validation.

PIMF-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	69		3	1			4
	BCC	1	19	1				2
	σ	4		24	1		1	6
	HEX		1	2	103		1	4
	GYR		1	1		15		2
	LAM				2		114	2
CD-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	60	1	7	5			13
	BCC	13	7	1				14
	σ	6		24				6
	HEX	2		1	94	3	7	13
	GYR				8	5	4	12
	LAM				2	2	112	4
Curve-RF		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	53	2	4	12			18
	BCC	4	10	1	6			11
	σ	1	2	23	4			7
	HEX	5			91		11	16
	GYR				11	4	2	13
	LAM				18		98	18

The performance of bagging and XGBoost models is shown in Figures D.7 - D.8 and summarized in Tables D.4 - D.5. Analysis of the data which includes manual identification errors reveals that both models identify all three incorrectly labeled samples. Upon correction of these mislabeled samples and retraining the models, the number of misclassified samples by the PIMF-bagging is reduced from 26 to 21 instances, while the PIMF-XGBoost has a decrease from 24 to 19 instances.

Figure D.7: Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using PIMF-bagging model.

Figure D.8: Maximum predicted probability for correctly and incorrectly predicted samples of all diblock copolymers using PIMF-XGBoost model.

Table D.4: The classification results of the bagging model with out-of-sample cross-validation using PIMF.

		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	68	1	4	1			5
	BCC	1	19	1				2
	σ	3		25	1		1	5
	HEX			4	101	1	1	6
	GYR				1	16		1
	LAM				2		114	2

Table D.5: The prediction results of the XGBoost model with out-of-sample cross-validation using PIMF.

		pred						
		DIS	BCC	σ	HEX	GYR	LAM	# Misclassified
true	DIS	67	1	4	1			6
	BCC	1	19	1				2
	σ	3		26	1			4
	HEX	1		2	103		1	4
	GYR	1	1			15		2
	LAM			1			115	1

D.3 Further inspection of misclassified samples by quantified uncertainty

We discussed using the maximum predicted probability as an indication of quantified uncertainty for further inspecting the misclassified samples when predicting the diblock with new chemical identities. In general, the uncertainty of the classification of a test sample is large when the maximum predicted probability is small. Panels (a)-(b) in Figure D.9 illustrate the SAXS curves incorrectly predicted for the third and fourth groups, respectively, while panel (c) presents representative SAXS curves for the phases BCC, HEX, LAM, σ, and DIS. Note the samples of phase GYR are absent in panel (c) as there were no misclassifications associated with this phase. Further, Figures D.10 and D.11 detail the feature values used in predicting the third and fourth groups, respectively, using the jittered plot, which added random noise to the x-coordinate of the scatter plot for better visualization of overlapping data points. Herein, training values are represented in blue, values for correctly predicted test samples in green, and those for inaccurately predicted samples in red. The presence of zero values in these plots accounts for missing values, as not all curves have 3 peaks in our dataset.

In predicting the third group, one sample identified as DIS is incorrectly classified as σ (Figure D.9(a)). While most DIS samples exhibit no more than one peak, a subset, due to phase mixing, may display two peaks. Notably, the erroneously classified DIS sample is distinguished by the detection of three peaks. As illustrated in Figure D.10, this test sample is the only one that has three detected peaks (observed from panels of p_3/p_1 and p_3/p_2), with its other feature values closely aligning with those of σ samples, leading to its prediction as σ. Another misclassification within the third group involved a sample identified as LAM being predicted as HEX. The SAXS curves for HEX and LAM phases often exhibit

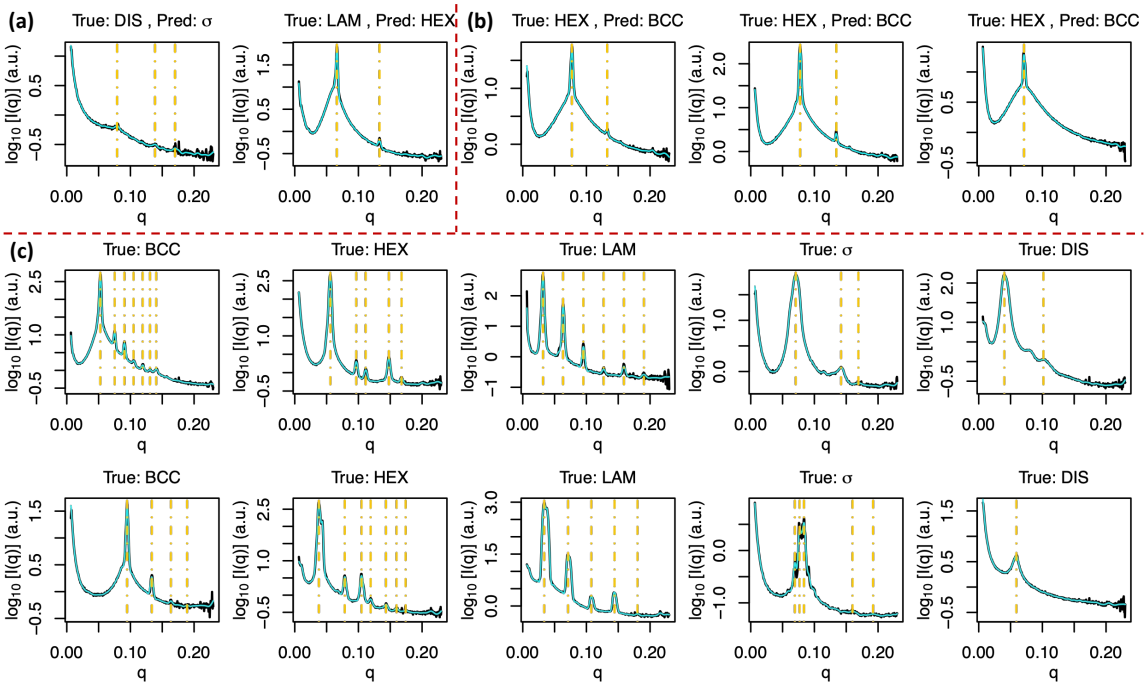

Figure D.9: (a) Two incorrectly predicted samples using PIMF-RF for the third group, D-9F. (b) Three incorrectly predicted samples for the fourth group, D-12F. (c) Two representative SAXS curves for the phases BCC, HEX, LAM, σ, and DIS. The black dots and blue curves represent the original log intensity and the smoothed intensity curves, respectively. Detected peak locations are marked with yellow dotted lines.

similar patterns, generally featuring at least three peaks. However, the misclassified sample presented only two detected peaks, making it harder to decide. Furthermore, while HEX phase curves theoretically adhere to a peak ratio of $[1, \sqrt{3}, 2, ...]$, the suppression of the second peak can result in a peak ratio of $[1, 2, ...]$ (Figure D.10), mirroring the initial peak ratios observed in LAM samples. A detailed examination of the predicted probabilities for this sample revealed that LAM possessed the second-highest probability, which aligns with our expectations.

For the prediction of the fourth group, all three misclassified samples are HEX samples predicted as BCC, as shown in Figure D.9(b). Among these, two samples exhibit two peaks, and one sample presents only a single peak, deviating from the typical pattern of HEX samples, which usually display more than three peaks. As highlighted in Figure D.11,

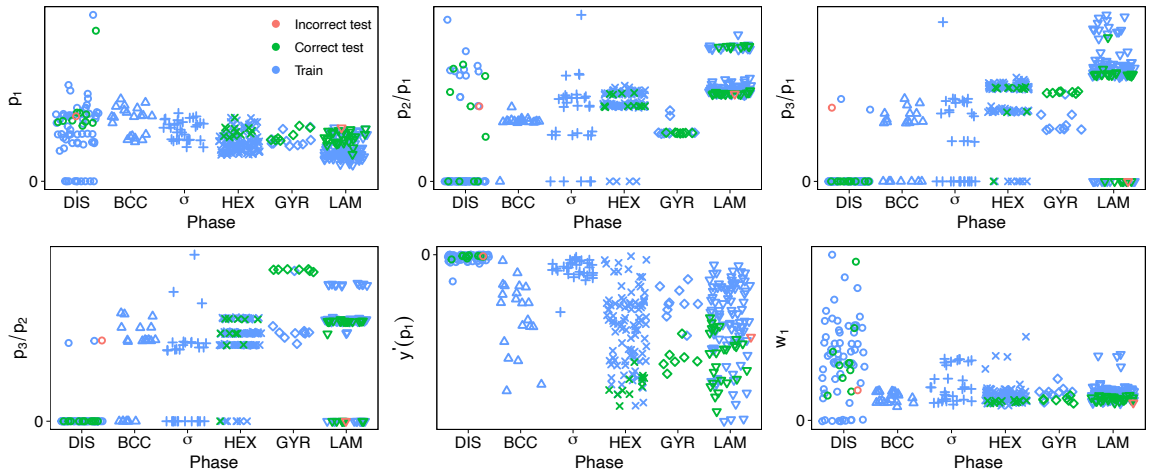

Figure D.10: The value of each PIMF in the training and testing set when predicting the third group, D-9F.

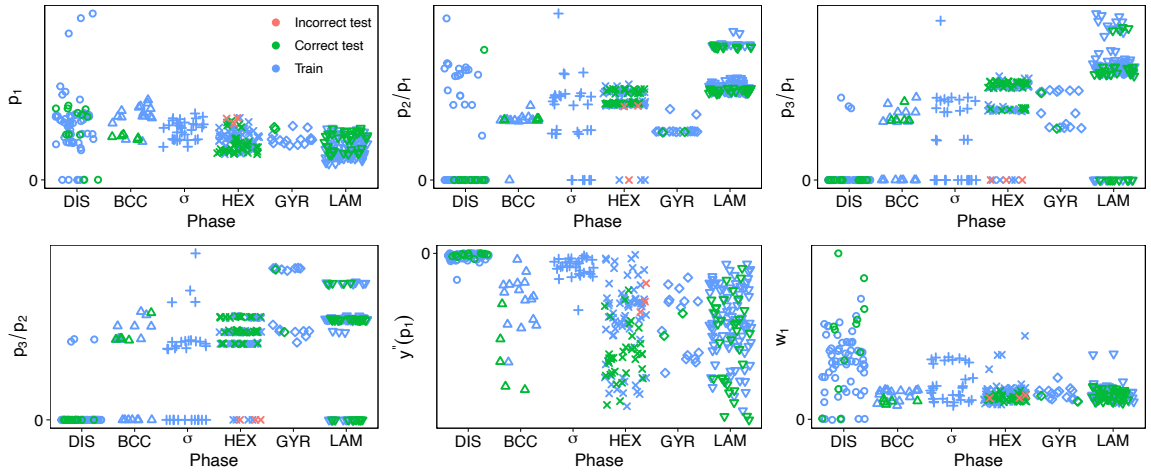

Figure D.11: The value of each PIMF in the training and testing set when predicting the last group, D-12F.

these incorrectly predicted samples are the only three samples that have at most two peaks detected among the test set, whereas a larger fraction of BCC samples have no more than two peaks. Thus, accurately predicting the phase of these samples proves challenging. When closely examining the predicted probability of these three samples, for the two samples with two peaks, the HEX phase emerged as the second most likely prediction. For the sample with a single detected peak, the σ phase is assigned the second-highest probability. This is because σ samples, similar to the misclassified sample, can also feature just one peak, alongside comparable values for the sharpness and width of the first peak.

Appendix E

Appendix of Chapter 5

E.1 Analytical forms in cDFT

The closed-form expressions of $\delta F_{ex}/\delta\rho$, Ω, and F_{ex} for 1D HR [210, 237] are summarized here:

$$\frac{\delta\beta_{en}F_{ex}[\rho]}{\delta\rho(s)} = \int_{s}^{s+a} \frac{\rho(s'')}{1-\int_{s''-a}^{s''}\rho(s')\,ds'}\,ds'' - \log\left(1-\int_{s-a}^{s}\rho(s')\,ds'\right) \tag{E.1}$$

$$\beta_{en}\Omega[\rho] = -\int_{-\infty}^{\infty} \frac{\rho(s-a)}{1-\int_{s-a}^{s}\rho(s')\,ds'}\,ds \tag{E.2}$$

$$\beta_{en}F_{ex}[\rho] = -\int \rho(s)\log\left(1-\int_{s-a}^{s}\rho(s')\,ds'\right)ds \tag{E.3}$$

where a is the length of each hard rod.

E.2 External potentials and chemical potentials

Let L be the distance between two hard walls, and a is the length of the hard rods (HR). We consider the following four parametric forms of the external potential.

The first functional form applies to uniform hard rods confined between hard walls

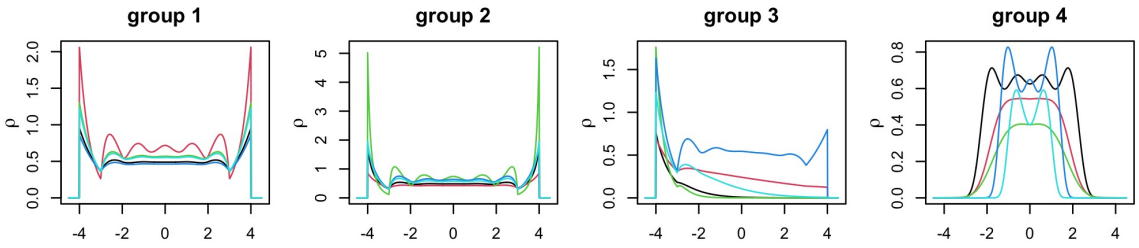

Figure E.1: Each plot gives 5 examples of generated densities from external potential in E.4-E.7.

[238]:

$$V^{ext}(s) = \begin{cases} \infty, & s < a/2, s > L - a/2, \\ 0, & a/2 < s < L - a/2. \end{cases} \tag{E.4}$$

In the second functional form, the hard rods experience a van der Waals-like attraction from the hard walls

$$V^{ext}(s) = \begin{cases} \infty, & s < a/2, s > L - a/2, \\ -\epsilon\{(\frac{a}{s+a/2})^3 + (\frac{a}{L+a/2-s})^3\}, & a/2 < s < L - a/2. \end{cases} \tag{E.5}$$

Equation(E.5) is adopted from [238]. With fixed a and L, ϵ is a parameter in the range of $\epsilon \in (0.1, 2.2)$. Note that when $\epsilon = 0$, we have the same potential as the first one.

The third functional form consists of hard-wall potentials and a repulsion linearly proportional to the distance

$$V^{ext}(s) = \begin{cases} \infty, & s < a/2, s > L - a/2, \\ m_g s, & a/2 < s < L - a/2. \end{cases} \tag{E.6}$$

Intuitively, the linear form mimics the gravitational potential.[239] This is the only asymmetric potential considered in this chapter. In training the machine-learning models, the range of m_g is chosen to be $(0.1, 3)$.

The power-law external potential [239] is the fourth functional class considered in

this chapter. This external potential has more flexibility as it has three parameters, u_0, x_0, and a_0 that can be used to modify the range of interactions:

$$V^{ext}(s) = \begin{cases} \infty, & s < a/2, s > L - a/2, \\ u_0 \left| \frac{s - L/2}{x_0} \right|^{a_0}, & a/2 < s < L - a/2, \end{cases} \tag{E.7}$$

where $a_0 > 0$. In this chapter, we use $u_0 \in (1, 3), x_0 \in (1, 3)$ and $a_0 \in (2, 5)$. Figure E.1 shows five examples of generated densities for each external potential.

In section 5.4.2, the performance of our strategy is evaluated on a new set of densities. The external potential of these densities is generated from the weighted average of (E.5) and (E.6):

$$V^{ext}(s) = \begin{cases} \infty, & s < a/2, s > L - a/2, \\ -w\epsilon\{(\frac{a}{s+a/2})^3 + (\frac{a}{L+a/2-s})^3\} + (1-w)m_g s, & a/2 < s < L - a/2, \end{cases}$$

where $w \in (0, 1)$.

E.3 Derivation of probabilistic upper bounds

First, we derive the probabilistic upper bound for the absolute difference of densities. For a fixed coordinate j, based on the predictive distribution in (1.14), we have $\mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \sqrt{\hat{\sigma}_j^2 c^{**}} t_{\alpha/2}(n - p_\mu)\right) = \alpha$. Since $\rho_j(\mathbf{x}^*)$ is predicted for sample $\mathbf{x}^*$ only if $\sqrt{\hat{\sigma}_j^2 c^{**}} \leq \delta_j / t_{\alpha/2}(n - p_\mu)$, the probability of predictive error larger than error bound

δ_j follows

$$\begin{aligned}
\mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \delta_j\right) &= \mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \frac{\delta_j}{t_{\alpha/2}(n-p_\mu)} t_{\alpha/2}(n-p_\mu)\right) \\
&\leq \mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \sqrt{\hat{\sigma}_j^2 c^{**}}\, t_{\alpha/2}(n-p_\mu)\right) \\
&= \alpha.
\end{aligned}$$

Next, let us consider controlling the grid-level predictive error using the maximum variance selection criterion in (5.5), where $\boldsymbol{\rho}(\mathbf{x}^*)$ is predicted for $\mathbf{x}^*$ only if $\sqrt{\max_j\{\hat{\sigma}_j^2 c^{**}\}} \leq \delta/t_{\alpha/2}(n-p_\mu)$. Then for any j, $j = 1, ..., k$,

$$\begin{aligned}
\mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \delta\right) &\leq \mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \sqrt{\max_j \hat{\sigma}_j^2 c^{**}}\, t_{\alpha/2}(n-p_\mu)\right) \\
&\leq \mathbb{P}\left(|\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*)| > \sqrt{\hat{\sigma}_j^2 c^{**}}\, t_{\alpha/2}(n-p_\mu)\right) \\
&= \alpha,
\end{aligned}$$

from which equation (5.6) is proved.

Third, let us derive the probabilistic bound for root mean squared error ($\text{RMSE} = \sqrt{\sum_{j=1}^k (\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2/k}$) under the maximum variance selection criterion in Equation (5.5). For $j = 1, ..., k$, we have

$$\begin{aligned}
\mathbb{P}(\text{RMSE} > \delta) &= \mathbb{P}\left(\sum_{j=1}^k \frac{(\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2}{k} > \delta^2\right) \\
&\leq \mathbb{P}\left(\max_j (\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2 > \delta^2\right) \\
&\leq \mathbb{P}\left(\max_j (\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2 > \max_j \hat{\sigma}_j^2 c^{**} t_{\alpha/2}^2(n-p_\mu)\right),
\end{aligned}$$

where the third inequality follows from the condition in Equation (5.5). Supposing the left

hand side inside the probability is maximized when $j = j^*$, then we have

$$\begin{aligned}\mathbb{P}(\text{RMSE} > \delta) &\leq \mathbb{P}\left((\rho_{j^*}(\mathbf{x}^*) - \hat{\rho}_{j^*}(\mathbf{x}^*))^2 > \max_j \hat{\sigma}_j^2 c^{**} t^2_{\alpha/2}(n - p_\mu)\right), \\ &\leq \mathbb{P}\left((\rho_{j^*}(\mathbf{x}^*) - \hat{\rho}_{j^*}(\mathbf{x}^*))^2 > \hat{\sigma}_{j^*}^2 c^{**} t^2_{\alpha/2}(n - p_\mu)\right) = \alpha,\end{aligned}$$

from which equation (5.7) is proved.

Lastly, we give the upper bound for the expectation of mean squared error, $\text{MSE} = \sum_{j=1}^k (\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2/k$, under the average variance selection criterion (Equation (5.8)),

$$\begin{aligned}\mathbb{E}_{\boldsymbol{\rho}(\mathbf{x}^*)}(\text{MSE}) &= \mathbb{E}_{\boldsymbol{\rho}(\mathbf{x}^*)}\left[\sum_{j=1}^k \frac{(\rho_j(\mathbf{x}^*) - \hat{\rho}_j(\mathbf{x}^*))^2}{k}\right] \\ &= \sum_{j=1}^k \frac{\hat{\sigma}_j^2 c^{**}}{k} \frac{n - p_\mu}{n - p_\mu - 2} \\ &\leq \frac{\delta^2}{t^2_{\alpha/2}(n - p_\mu)} \frac{n - p_\mu}{n - p_\mu - 2}.\end{aligned}$$

For moderately large n, and small α, we have $\mathbb{E}_{\boldsymbol{\rho}(\mathbf{x}^*)}(\text{MSE}) \leq \delta^2$. For instance, for any $n \geq 20$ and constant mean basis ($p_\mu = 1$), $\mathbb{E}_{\boldsymbol{\rho}(\mathbf{x}^*)}(\text{MSE}) \leq \delta^2$ for any $\alpha \leq 0.2$.

E.4 Details of the D-optimality criterion

Here we discuss the details of the D-optimality criterion. Consider the correlation function $c(\mathbf{x}, \mathbf{x}_i), i = 1, ..., n$, with each training set sample serving as a set of basis functions. Then, by definition, these basis functions of the training dataset form the correlation matrix $\mathbf{R}$, and the basis functions with any testing sample $\mathbf{x}^*$ gives $\mathbf{r}^T(\mathbf{x}^*)$. Define the row vector $\mathbf{c} = \mathbf{r}^T(\mathbf{x}^*)\mathbf{R}^{-1}$ for each sample $\mathbf{x}^*$. Set the threshold $c_{th} \geq 1$ for the maximum absolute value in $\mathbf{c}$. If the uncertainty estimate $c_{max} := \max |c_i| > c_{th}$, a numerical solver will be called to solve the system and the outcomes at $\mathbf{x}^*$ will be added to the training set. Otherwise,

the emulator will be used to predict the test input $\mathbf{x}^*$. Since the D-optimality criterion is not directly related to the predictive error, selecting threshold c_{th} may be performed in a case-by-case manner. In order to make a comparison with our method, the threshold is adjusted to make the number of augmented samples selected from D-optimality at least as large as the one in our strategy. For more discussion about the D-optimality criterion in emulating molecular simulations, we refer the readers to [40] and section 15.2.3 in [39].

E.5 Out of sample prediction results of particle density and grand potential

The performance of the ALEC emulator is compared to the performance with three other designs: random sampling using the same number of training size as the initial training size in ALEC (RS1), random sampling using the same number of training size as the final training size in ALEC (RS2), and active learning algorithm with D-optimality criterion. The performance is compared based on root mean squared error (RMSE) of density and energy, and 95% interval coverage and length. Suppose we have a total of n^* testing samples remaining after running the model, then the criteria we adopt are:

$$
\begin{aligned}
RMSE_\rho &= \sqrt{\frac{\sum_{i=1}^{n^*}\sum_{j=1}^{k}(\rho_j(\mathbf{x}_i^*)-\hat{\rho}_j(\mathbf{x}_i^*))^2}{n^*k}},\\
coverage_\rho &= \frac{1}{n^*k}\sum_{i=1}^{n^*}\sum_{j=1}^{k}\mathbb{1}\{\rho_j(\mathbf{x}_i^*)\in CI_{i,j}(95\%)\},\\
length_\rho &= \frac{1}{n^*k}\sum_{i=1}^{n^*}\sum_{j=1}^{k} length\{CI_{i,j}(95\%)\},\\
RMSE_\Omega &= \sqrt{\frac{\sum_{i=1}^{n^*}(\Omega(\mathbf{x}_i^*)-\hat{\Omega}(\mathbf{x}_i^*))^2}{n^*}},
\end{aligned}
$$

where $\hat{\rho}_j(\mathbf{x}_i^*)$ is the predicted density of i-th hold out sample at j-th coordinate, and $CI_{i,j}(95\%)$ is the 95% confidence interval of $\hat{\rho}_j(\mathbf{x}_i^*)$ based on the predictive distribution.

Table E.1: The prediction results for building models on four groups separately. In ALEC emulator, we ues $\delta = 0.01$ and $\alpha = 0.05$ as threshold. In D-opt, we adjust the threshold c_{th} such that it has the similar number of augmented samples in AL, and c_{th} for group 1-4 are, $30, 3.4, 2.4$ and 1.03, respectively.

		training n	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
	Group 1	20	6.97×10^{-6}	87.7%	2.47×10^{-5}	2.60×10^{-5}
RS1	Group 2	20	3.62×10^{-3}	93.9%	6.79×10^{-3}	1.27×10^{-2}
	Group 3	20	2.00×10^{-2}	92.0%	8.81×10^{-2}	7.07×10^{-1}
	Group 4	20	8.81×10^{-2}	91.2%	3.69×10^{-1}	1.24
		$n_{ini} + n_{aug}$	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
	Group 1	20+0	6.97×10^{-6}	87.7%	2.47×10^{-5}	2.60×10^{-5}
ALEC	Group 2	20+7	$\mathbf{1.02 \times 10^{-3}}$	**95.6**%	$\mathbf{1.97 \times 10^{-3}}$	$\mathbf{3.73 \times 10^{-3}}$
	Group 3	20+21	$\mathbf{2.12 \times 10^{-3}}$	94.2%	$\mathbf{4.25 \times 10^{-3}}$	$\mathbf{1.43 \times 10^{-2}}$
	Group 4	20+504	$\mathbf{3.79 \times 10^{-3}}$	85.1%	$\mathbf{6.61 \times 10^{-3}}$	$\mathbf{1.62 \times 10^{-2}}$
		training n	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
	Group 1	20	6.97×10^{-6}	87.7%	2.47×10^{-5}	2.60×10^{-5}
RS2	Group 2	27	1.71×10^{-3}	96.3%	2.53×10^{-3}	5.05×10^{-3}
	Group 3	41	6.89×10^{-3}	**94.9**%	8.97×10^{-3}	2.16×10^{-1}
	Group 4	524	1.14×10^{-1}	**96.6**%	3.94×10^{-1}	4.20×10^{-1}
		$n_{ini} + n_{aug}$	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
	Group 1	20+0	6.97×10^{-6}	87.7%	2.47×10^{-5}	2.60×10^{-5}
D-opt	Group 2	20+7	1.44×10^{-3}	94.9%	2.27×10^{-3}	4.32×10^{-3}
	Group 3	20+21	3.00×10^{-3}	93.6%	4.53×10^{-3}	1.43×10^{-2}
	Group 4	20+582	1.37×10^{-1}	98.1%	3.73×10^{-1}	5.64×10^{-1}

Table E.2: The prediction results for building models on four groups together. We use $\delta = 0.01$ and $\alpha = 0.05$ in the ALEC emulator. In D-opt, the threshold $c_{th} = 1.12$. For each strategy, we evaluate the overall performance, as well as the performance for each subgroup.

			training n	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
RS1	All		80	1.02×10^{-1}	**94.7%**	3.44×10^{-1}	4.70×10^{-1}
		Group 1	24	1.45×10^{-4}	94.9%	2.66×10^{-4}	4.13×10^{-4}
		Group 2	15	2.18×10^{-3}	93.0%	1.94×10^{-3}	8.57×10^{-3}
		Group 3	23	7.92×10^{-3}	99.5%	7.07×10^{-2}	5.37×10^{-2}
		Group 4	18	2.04×10^{-1}	91.4%	1.30	9.38×10^{-1}
			$n_{ini} + n_{aug}$	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
ALEC	All		80+873	$\mathbf{8.76 \times 10^{-4}}$	91.1%	$\mathbf{1.70 \times 10^{-3}}$	$\mathbf{4.14 \times 10^{-3}}$
		Group 1	24+0	1.30×10^{-4}	90.6%	1.25×10^{-4}	3.49×10^{-4}
		Group 2	15+1	1.13×10^{-3}	80.3%	7.11×10^{-4}	6.93×10^{-3}
		Group 3	23+58	5.12×10^{-4}	99.1%	2.84×10^{-3}	2.46×10^{-3}
		Group 4	18+814	1.42×10^{-3}	97.0%	4.17×10^{-3}	3.39×10^{-3}
			training n	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
RS2	All		953	3.25×10^{-2}	93.3%	8.42×10^{-2}	4.64×10^{-1}
		Group 1	245	8.29×10^{-6}	74.3%	1.16×10^{-5}	1.01×10^{-5}
		Group 2	221	6.50×10^{-5}	100%	7.25×10^{-4}	1.78×10^{-3}
		Group 3	238	3.76×10^{-3}	100%	2.92×10^{-2}	2.67×10^{-2}
		Group 4	249	6.51×10^{-2}	98.7%	3.09×10^{-1}	9.31×10^{-1}
			$n_{ini} + n_{aug}$	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
D-opt	All		80+935	8.91×10^{-2}	98.3%	7.60×10^{-2}	3.31×10^{-1}
		Group 1	24+59	2.11×10^{-5}	92.8%	1.42×10^{-4}	3.77×10^{-5}
		Group 2	15+73	1.37×10^{-4}	98.0%	2.29×10^{-4}	4.90×10^{-4}
		Group 3	23+393	2.23×10^{-4}	99.9%	1.05×10^{-3}	7.53×10^{-4}
		Group 4	18+410	1.88×10^{-1}	97.1%	3.36×10^{-1}	6.98×10^{-1}

Table E.3: The prediction results of the ALEC approach, using average-sense criterion and maximum-sense criterion, on a new functional form that is not in the training data set, starting with the final GP emulator developed for groups 1-4 and all groups combined. $\delta = 0.01$ and $\alpha = 0.05$ are used.

ALEC		$n_{ini} + n_{aug}$	$RMSE_\rho$	$coverage_\rho$	$length_\rho$	$RMSE_\Omega$
average	Group 1	20+90	5.04×10^{-3}	83.9%	4.87×10^{-3}	1.96×10^{-2}
	Group 2	27+120	2.05×10^{-3}	92.2%	4.02×10^{-3}	6.08×10^{-3}
	Group 3	41+66	4.47×10^{-3}	87.7%	4.91×10^{-3}	1.47×10^{-2}
	Group 4	524+47	1.48×10^{-2}	69.7%	6.11×10^{-3}	6.46×10^{-2}
	All groups	953+15	4.68×10^{-3}	77.7%	3.42×10^{-3}	1.83×10^{-2}
maximum	Group 1	20+408	4.34×10^{-4}	94.9%	4.87×10^{-3}	7.89×10^{-4}
	Group 2	27+395	4.69×10^{-4}	95.0%	4.02×10^{-3}	7.85×10^{-4}
	Group 3	41+384	4.39×10^{-4}	95.1%	4.91×10^{-3}	7.70×10^{-4}
	Group 4	524+361	5.73×10^{-4}	90.5%	6.11×10^{-3}	7.15×10^{-4}
	All groups	953+268	5.18×10^{-4}	92.5%	3.42×10^{-3}	7.24×10^{-4}

Bibliography

[1] A. Arora, T.-S. Lin, N. J. Rebello, S. H. Av-Ron, H. Mochigase, and B. D. Olsen, "Random forest predictor for diblock copolymer phase behavior," *ACS Macro Letters*, vol. 10, no. 11, pp. 1339–1345, 2021.

[2] B. A. Camley, J. Zimmermann, H. Levine, and W.-J. Rappel, "Emergent collective chemotaxis without single-cell gradient sensing," *Physical Review Letters*, vol. 116, no. 9, p. 098101, 2016.

[3] K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev, and A. Walsh, "Machine learning for molecular and materials science," *Nature*, vol. 559, no. 7715, pp. 547–555, 2018.

[4] K. J. Bergen, P. A. Johnson, M. V. de Hoop, and G. C. Beroza, "Machine learning for data-driven discovery in solid earth geoscience," *Science*, vol. 363, no. 6433, p. eaau0323, 2019.

[5] G. Schiebinger, J. Shu, M. Tabaka, B. Cleary, V. Subramanian, A. Solomon, J. Gould, S. Liu, S. Lin, P. Berube, *et al.*, "Optimal-transport analysis of single-cell gene expression identifies developmental trajectories in reprogramming," *Cell*, vol. 176, no. 4, pp. 928–943, 2019.

[6] M. Gu, X. Fang, and Y. Luo, "Data-driven model construction for anisotropic dynamics of active matter," *PRX Life*, vol. 1, no. 1, p. 013009, 2023.

[7] X. Fang and M. Gu, "The inverse Kalman filter," *Biometrika*, vol. 112, no. 4, p. asaf054, 2025.

[8] M. C. Kennedy and A. O'Hagan, "Bayesian calibration of computer models," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 63, no. 3, pp. 425–464, 2001.

[9] C. E. Rasmussen, *Gaussian processes for machine learning*. MIT Press, 2006.

[10] X. Fang, M. Gu, and J. Wu, "Reliable emulation of complex functionals by active learning with error control," *The Journal of Chemical Physics*, vol. 157, no. 21, p. 214109, 2022.

[11] M. L. Stein, *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 1999.

[12] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. Chapman and Hall/CRC, 2020.

[13] D. J. C. MacKay, "Information-based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 590–604, 1992.

[14] D. A. Cohn, Z. Ghahramani, and M. I. Jordan, "Active learning with statistical models," *Journal of artificial intelligence research*, vol. 4, pp. 129–145, 1996.

[15] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas, "Taking the human out of the loop: A review of Bayesian optimization," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 148–175, 2015.

[16] B. J. Shields, J. Stevens, J. Li, M. Parasram, F. Damani, J. I. M. Alvarado, J. M.

Janey, R. P. Adams, and A. G. Doyle, "Bayesian reaction optimization as a tool for chemical synthesis," *Nature*, vol. 590, no. 7844, pp. 89–96, 2021.

[17] E. Snelson and Z. Ghahramani, "Sparse Gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, p. 1257, 2006.

[18] M. Titsias, "Variational learning of inducing variables in sparse Gaussian processes," in *Artificial Intelligence and Statistics*, pp. 567–574, PMLR, 2009.

[19] A. V. Vecchia, "Estimation and model identification for continuous spatial processes," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 50, no. 2, pp. 297–312, 1988.

[20] M. Katzfuss and J. Guinness, "A general framework for Vecchia approximations of Gaussian processes," *Statistical Science*, vol. 36, no. 1, pp. 124–141, 2021.

[21] M. Katzfuss, J. Guinness, and E. Lawrence, "Scaled Vecchia approximation for fast computer-model emulation," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 10, no. 2, pp. 537–554, 2022.

[22] R. B. Gramacy and D. W. Apley, "Local Gaussian process approximation for large computer experiments," *Journal of Computational and Graphical Statistics*, vol. 24, no. 2, pp. 561–578, 2015.

[23] M. S. Handcock and M. L. Stein, "A Bayesian analysis of kriging," *Technometrics*, vol. 35, no. 4, pp. 403–410, 1993.

[24] P. Whittle, "On stationary processes in the plane," *Biometrika*, pp. 434–449, 1954.

[25] J. Hartikainen and S. Särkkä, "Kalman filtering and smoothing solutions to tempo-

ral Gaussian process regression models," in *2010 IEEE International Workshop on Machine Learning for Signal Processing*, pp. 379–384, 2010.

[26] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[27] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, and O. Shochet, "Novel type of phase transition in a system of self-driven particles," *Physical Review Letters*, vol. 75, no. 6, p. 1226, 1995.

[28] J. Toner and Y. Tu, "Long-range order in a two-dimensional dynamical XY model: how birds fly together," *Physical Review Letters*, vol. 75, no. 23, p. 4326, 1995.

[29] L. Ward, A. Agrawal, A. Choudhary, and C. Wolverton, "A general-purpose machine learning framework for predicting properties of inorganic materials," *npj Computational Materials*, vol. 2, no. 1, pp. 1–7, 2016.

[30] X. Fang, E. A. Murphy, P. A. Kohl, Y. Li, C. J. Hawker, C. M. Bates, and M. Gu, "Universal phase identification of block copolymers from physics-informed machine learning," *Journal of Polymer Science*, vol. 63, no. 6, pp. 1433–1440, 2025.

[31] S. Oh, X. Fang, I.-H. Lin, P. Dee, C. S. Dunham, S. M. Copp, A. G. Doyle, J. Read de Alaniz, and M. Gu, "Synergizing chemical and AI communities for advancing laboratories of the future," *ACS Central Science*, vol. 12, no. 2, pp. 157–173, 2026.

[32] T. J. Santner, B. J. Williams, and W. I. Notz, *The design and analysis of computer experiments*. Springer Science & Business Media, 2003.

[33] C. D. Lin, B. Tang, *et al.*, "Latin hypercubes and space-filling designs," *Handbook of design and analysis of experiments*, pp. 593–625, 2015.

[34] M. E. Johnson, L. M. Moore, and D. Ylvisaker, "Minimax and maximin distance designs," *Journal of statistical planning and inference*, vol. 26, no. 2, pp. 131–148, 1990.

[35] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments," *Statistical Science*, vol. 4, no. 4, pp. 409–423, 1989.

[36] J.-L. Wang, J.-M. Chiou, and H.-G. Müller, "Functional data analysis," *Annual Review of Statistics and Its Application*, vol. 3, no. 1, pp. 257–295, 2016.

[37] P. Hohenberg and W. Kohn, "Inhomogeneous electron gas," *Physical Review*, vol. 136, no. 3B, p. B864, 1964.

[38] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," *arXiv preprint arXiv:2010.08895*, 2020.

[39] K. T. Schütt, S. Chmiela, O. A. von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, *Machine learning meets quantum physics*. Springer Cham, 2020.

[40] E. V. Podryabinkin and A. V. Shapeev, "Active learning of linearly parametrized interatomic potentials," *Computational Materials Science*, vol. 140, pp. 171–180, 2017.

[41] R. B. Gramacy and H. K. Lee, "Adaptive design and analysis of supercomputer experiments," *Technometrics*, vol. 51, no. 2, pp. 130–145, 2009.

[42] O. Roustant, D. Ginsbourger, and Y. Deville, "DiceKriging, DiceOptim: Two R packages for the analysis of computer experiments by Kriging-based metamodeling and optimization," *Journal of Statistical Software*, vol. 51, no. 1, pp. 1–55, 2012.

[43] M. Gu, J. Palomo, and J. O. Berger, "RobustGaSP: Robust Gaussian stochastic process emulation in R," *The R Journal*, vol. 11, no. 1, pp. 112–136, 2019.

[44] J. O. Berger, V. De Oliveira, and B. Sansó, "Objective Bayesian analysis of spatially correlated data," *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1361–1374, 2001.

[45] M. Gu, X. Wang, and J. O. Berger, "Robust Gaussian stochastic process emulation," *The Annals of Statistics*, vol. 46, no. 6A, pp. 3038–3066, 2018.

[46] M. Gu, "Jointly robust prior for Gaussian stochastic process in emulation, calibration and variable selection," *Bayesian Analysis*, vol. 14, no. 1, 2018.

[47] M. Gu and J. O. Berger, "Parallel partial Gaussian process emulation for computer models with massive output," *The Annals of Applied Statistics*, vol. 10, no. 3, pp. 1317–1347, 2016.

[48] S. Conti and A. O'Hagan, "Bayesian emulation of complex multi-output and dynamic computer models," *Journal of Statistical Planning and Inference*, vol. 140, no. 3, pp. 640–651, 2010.

[49] M. A. Alvarez, L. Rosasco, and N. D. Lawrence, "Kernels for vector-valued functions: A review," *Foundations and Trends® in Machine Learning*, vol. 4, no. 3, pp. 195–266, 2012.

[50] D. Higdon, J. Gattiker, B. Williams, and M. Rightley, "Computer model calibration using high-dimensional output," *Journal of the American Statistical Association*, vol. 103, no. 482, pp. 570–583, 2008.

[51] M. Gu and W. Shen, "Generalized probabilistic principal component analysis of

correlated data," *Journal of Machine Learning Research*, vol. 21, no. 13, pp. 1–41, 2020.

[52] M. Gu and H. Li, "Gaussian orthogonal latent factor processes for large incomplete matrices of correlated data," *Bayesian Analysis*, vol. 17, no. 4, pp. 1219 – 1244, 2022.

[53] Y. Lin, X. Liu, P. Segall, and M. Gu, "Fast data inversion for high-dimensional dynamical systems from noisy measurements," *arXiv preprint arXiv:2501.01324*, 2025.

[54] M. West and J. Harrison, *Bayesian forecasting and dynamic models*. Springer, 1997.

[55] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*, vol. 38. OUP Oxford, 2012.

[56] G. Petris, S. Petrone, and P. Campagnoli, "Dynamic linear models," in *Dynamic linear models with R*, Springer, 2009.

[57] R. Prado and M. West, *Time series: modeling, computation, and inference*. Chapman and Hall/CRC, 2010.

[58] D. C. Rapaport, *The art of molecular dynamics simulation*. Cambridge university press, 2004.

[59] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, p. 409, 1952.

[60] M. L. Stein, J. Chen, and M. Anitescu, "Stochastic approximation of score functions for Gaussian processes," *The Annals of Applied Statistics*, vol. 7, no. 2, pp. 1162–1191, 2013.

[61] S. Majumder, Y. Guan, B. J. Reich, and A. K. Saibaba, "Kryging: geostatistical analysis of large-scale datasets using Krylov subspace methods," *Statistics and Computing*, vol. 32, no. 5, p. 74, 2022.

[62] F. Lindgren, H. Rue, and J. Lindström, "An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 73, no. 4, pp. 423–498, 2011.

[63] A. Datta, S. Banerjee, A. O. Finley, and A. E. Gelfand, "Hierarchical nearest-neighbor Gaussian process models for large geostatistical datasets," *Journal of the American Statistical Association*, vol. 111, no. 514, pp. 800–812, 2016.

[64] A. O. Finley, A. Datta, and S. Banerjee, "spNNGP R package for nearest neighbor Gaussian process models," *Journal of Statistical Software*, vol. 103, no. 5, pp. 1–40, 2022.

[65] T. Hastie and R. Tibshirani, "Varying-coefficient models," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 55, no. 4, pp. 757–779, 1993.

[66] A. K. Saibaba, A. Alexanderian, and I. C. Ipsen, "Randomized matrix-free trace and log-determinant estimators," *Numerische Mathematik*, vol. 137, no. 2, pp. 353–395, 2017.

[67] H. Chaté, F. Ginelli, G. Grégoire, F. Peruani, and F. Raynaud, "Modeling collective motion: variations on the Vicsek model," *The European Physical Journal B*, vol. 64, no. 3, pp. 451–456, 2008.

[68] F. Ginelli, F. Peruani, M. Bär, and H. Chaté, "Large-scale collective properties of self-propelled rods," *Physical Review Letters*, vol. 104, no. 18, p. 184502, 2010.

[69] Y. Katz, K. Tunstrøm, C. C. Ioannou, C. Huepe, and I. D. Couzin, "Inferring the structure and dynamics of interactions in schooling fish," *Proceedings of the National Academy of Sciences*, vol. 108, no. 46, pp. 18720–18725, 2011.

[70] F. Lu, M. Zhong, S. Tang, and M. Maggioni, "Nonparametric inference of interaction laws in systems of agents from trajectory data," *Proceedings of the National Academy of Sciences*, vol. 116, no. 29, pp. 14424–14433, 2019.

[71] R. Furrer, M. G. Genton, and D. Nychka, "Covariance tapering for interpolation of large spatial datasets," *Journal of Computational and Graphical Statistics*, vol. 15, no. 3, pp. 502–523, 2006.

[72] N. Cressie and G. Johannesson, "Fixed rank kriging for very large spatial data sets," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 70, no. 1, pp. 209–226, 2008.

[73] M. Gu, X. Fang, and Y. Lin, *FastGaSP: Fast and Exact Computation of Gaussian Stochastic Process*, 2025. R package version 0.6.2.

[74] Y. Luo, M. Gu, M. Park, X. Fang, Y. Kwon, J. M. Urueña, J. Read de Alaniz, M. E. Helgeson, C. M. Marchetti, and M. T. Valentine, "Molecular-scale substrate anisotropy, crowding and division drive collective behaviours in cell monolayers," *Journal of the Royal Society Interface*, vol. 20, no. 204, p. 20230160, 2023.

[75] E. Walck-Shannon and J. Hardin, "Cell intercalation from top to bottom," *Nature Reviews Molecular Cell Biology*, vol. 15, no. 1, pp. 34–48, 2014.

[76] M. C. Marchetti, J.-F. Joanny, S. Ramaswamy, T. B. Liverpool, J. Prost, M. Rao, and R. A. Simha, "Hydrodynamics of soft active matter," *Reviews of Modern Physics*, vol. 85, no. 3, p. 1143, 2013.

[77] T. Sanchez, D. T. Chen, S. J. DeCamp, M. Heymann, and Z. Dogic, "Spontaneous motion in hierarchically assembled active matter," *Nature*, vol. 491, no. 7424, pp. 431–434, 2012.

[78] S. Ramaswamy, "Active matter," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2017, no. 5, p. 054002, 2017.

[79] H. Yang, A. F. Pegoraro, Y. Han, W. Tang, R. Abeyaratne, D. Bi, and M. Guo, "Configurational fingerprints of multicellular living systems," *Proceedings of the National Academy of Sciences*, vol. 118, no. 44, p. e2109168118, 2021.

[80] T. E. Angelini, E. Hannezo, X. Trepat, M. Marquez, J. J. Fredberg, and D. A. Weitz, "Glass-like dynamics of collective cell migration," *Proceedings of the National Academy of Sciences*, vol. 108, no. 12, pp. 4714–4719, 2011.

[81] P.-F. Lenne and V. Trivedi, "Sculpting tissues by phase transitions," *Nature Communications*, vol. 13, no. 1, pp. 1–14, 2022.

[82] A. Ray, R. K. Morford, N. Ghaderi, D. J. Odde, and P. P. Provenzano, "Dynamics of 3D carcinoma cell invasion into aligned collagen," *Integrative biology*, vol. 10, no. 2, pp. 100–112, 2018.

[83] L. C. Biggs, C. S. Kim, Y. A. Miroshnikova, and S. A. Wickström, "Mechanical forces in the skin: roles in tissue architecture, stability, and function," *Journal of Investigative Dermatology*, vol. 140, no. 2, pp. 284–290, 2020.

[84] M. W. Ferguson and S. O'Kane, "Scar-free healing: from embryonic mechanisms to adult therapeutic intervention," *Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences*, vol. 359, no. 1445, pp. 839–850, 2004.

[85] S. Sarkar, G. Y. Lee, J. Y. Wong, and T. A. Desai, "Development and characterization of a porous micro-patterned scaffold for vascular tissue engineering applications," *Biomaterials*, vol. 27, no. 27, pp. 4775–4782, 2006.

[86] J. S. Choi, Y. Piao, and T. S. Seo, "Circumferential alignment of vascular smooth muscle cells in a circular microfluidic channel," *Biomaterials*, vol. 35, no. 1, pp. 63–70, 2014.

[87] T. D. Hinnant, J. A. Merkle, and E. T. Ables, "Coordinating proliferation, polarity, and cell fate in the Drosophila female germline," *Frontiers in cell and developmental biology*, vol. 8, p. 19, 2020.

[88] A. J. Ridley, M. A. Schwartz, K. Burridge, R. A. Firtel, M. H. Ginsberg, G. Borisy, J. T. Parsons, and A. R. Horwitz, "Cell migration: integrating signals from front to back," *Science*, vol. 302, no. 5651, pp. 1704–1709, 2003.

[89] R. Tonndorf, D. Aibibu, and C. Cherif, "Isotropic and anisotropic scaffolds for tissue engineering: Collagen, conventional, and textile fabrication technologies and properties," *International Journal of Molecular Sciences*, vol. 22, no. 17, p. 9561, 2021.

[90] G. Grégoire, H. Chaté, and Y. Tu, "Moving and staying together without a leader," *Physica D: Nonlinear Phenomena*, vol. 181, no. 3-4, pp. 157–170, 2003.

[91] T. Turiv, J. Krieger, G. Babakhanova, H. Yu, S. V. Shiyanovskii, Q.-H. Wei, M.-H. Kim, and O. D. Lavrentovich, "Topology control of human fibroblast cells monolayer by liquid crystal elastomer," *Science Advances*, vol. 6, no. 20, p. eaaz6485, 2020.

[92] G. Babakhanova, J. Krieger, B.-X. Li, T. Turiv, M.-H. Kim, and O. D. Lavrentovich, "Cell alignment by smectic liquid crystal elastomer coatings with nanogrooves," *Journal of Biomedical Materials Research Part A*, vol. 108, no. 5, pp. 1223–1230, 2020.

[93] D. Martella, L. Pattelli, C. Matassini, F. Ridi, M. Bonini, P. Paoli, P. Baglioni, D. S. Wiersma, and C. Parmeggiani, "Liquid crystal-induced myoblast alignment," *Advanced healthcare materials*, vol. 8, no. 3, p. 1801489, 2019.

[94] J. Toner and Y. Tu, "Flocks, herds, and schools: A quantitative theory of flocking," *Physical Review E*, vol. 58, no. 4, p. 4828, 1998.

[95] B. Szabo, G. Szöllösi, B. Gönci, Z. Jurányi, D. Selmeczi, and T. Vicsek, "Phase transition in the collective migration of tissue cells: experiment and model," *Physical Review E*, vol. 74, no. 6, p. 061908, 2006.

[96] H. Chaté, F. Ginelli, G. Grégoire, and F. Raynaud, "Collective motion of self-propelled particles interacting without cohesion," *Physical Review E*, vol. 77, no. 4, p. 046113, 2008.

[97] X. Li, R. Balagam, T.-F. He, P. P. Lee, O. A. Igoshin, and H. Levine, "On the mechanism of long-range orientational order of fibroblasts," *Proceedings of the National Academy of Sciences*, vol. 114, no. 34, pp. 8974–8979, 2017.

[98] J. Bongard and H. Lipson, "Automated reverse engineering of nonlinear dynamical systems," *Proceedings of the National Academy of Sciences*, vol. 104, no. 24, pp. 9943–9948, 2007.

[99] D. B. Brückner, N. Arlt, A. Fink, P. Ronceray, J. O. Rädler, and C. P. Broedersz, "Learning the dynamics of cell–cell interactions in confined cell migration," *Proceedings of the National Academy of Sciences*, vol. 118, no. 7, p. e2016602118, 2021.

[100] R. Zwanzig, *Nonequilibrium statistical mechanics*. Oxford university press, 2001.

[101] D. Selmeczi, S. Mosler, P. H. Hagedorn, N. B. Larsen, and H. Flyvbjerg, "Cell motility

as persistent random motion: theories from experiments," *Biophysical journal*, vol. 89, no. 2, pp. 912–931, 2005.

[102] P.-H. Wu, A. Giri, S. X. Sun, and D. Wirtz, "Three-dimensional cell migration does not follow a random walk," *Proceedings of the National Academy of Sciences*, vol. 111, no. 11, pp. 3949–3954, 2014.

[103] D. B. Brückner, A. Fink, C. Schreiber, P. J. Röttgermann, J. O. Rädler, and C. P. Broedersz, "Stochastic nonlinear dynamics of confined cell migration in two-state systems," *Nature Physics*, vol. 15, no. 6, pp. 595–601, 2019.

[104] D. B. Brückner, P. Ronceray, and C. P. Broedersz, "Inferring the dynamics of underdamped stochastic systems," *Physical Review Letters*, vol. 125, no. 5, p. 058103, 2020.

[105] F. Ferretti, V. Chardes, T. Mora, A. M. Walczak, and I. Giardina, "Building general Langevin models from discrete datasets," *Physical Review X*, vol. 10, no. 3, p. 031018, 2020.

[106] I.-C. Chou and E. O. Voit, "Recent developments in parameter estimation and structure identification of biochemical and genomic systems," *Mathematical biosciences*, vol. 219, no. 2, pp. 57–83, 2009.

[107] S. H. Rudy, S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Data-driven discovery of partial differential equations," *Science advances*, vol. 3, no. 4, p. e1602614, 2017.

[108] P. A. Reinbold, D. R. Gurevich, and R. O. Grigoriev, "Using noisy or incomplete data to discover models of spatiotemporal dynamics," *Physical Review E*, vol. 101, no. 1, p. 010203, 2020.

[109] J. Colen, M. Han, R. Zhang, S. A. Redford, L. M. Lemma, L. Morgan, P. V. Ruijgrok, R. Adkins, Z. Bryant, Z. Dogic, *et al.*, "Machine learning active-nematic hydrodynamics," *Proceedings of the National Academy of Sciences*, vol. 118, no. 10, p. e2016708118, 2021.

[110] M. Gu, X. Liu, X. Fang, and S. Tang, "Scalable marginalization of correlated latent variables with applications to learning particle interaction kernels," *The New England Journal of Statistics in Data Science*, vol. 1, no. 2, pp. 172–186, 2023.

[111] R. Lukeman, Y.-X. Li, and L. Edelstein-Keshet, "Inferring individual rules from collective behavior," *Proceedings of the National Academy of Sciences*, vol. 107, no. 28, pp. 12576–12580, 2010.

[112] G. Duclos, S. Garcia, H. Yevick, and P. Silberzan, "Perfect nematic order in confined monolayers of spindle-shaped cells," *Soft matter*, vol. 10, no. 14, pp. 2346–2353, 2014.

[113] S. Garcia, E. Hannezo, J. Elgeti, J.-F. Joanny, P. Silberzan, and N. S. Gov, "Physics of active jamming during collective cellular motion in a monolayer," *Proceedings of the National Academy of Sciences*, vol. 112, no. 50, pp. 15314–15319, 2015.

[114] Y. Fily and M. C. Marchetti, "Athermal phase separation of self-propelled particles with no alignment," *Physical Review Letters*, vol. 108, no. 23, p. 235702, 2012.

[115] C. Bechinger, R. Di Leonardo, H. Löwen, C. Reichhardt, G. Volpe, and G. Volpe, "Active particles in complex and crowded environments," *Reviews of Modern Physics*, vol. 88, no. 4, p. 045006, 2016.

[116] B. Wang, S. M. Anthony, S. C. Bae, and S. Granick, "Anomalous yet Brownian," *Proceedings of the National Academy of Sciences*, vol. 106, no. 36, pp. 15160–15164, 2009.

[117] K. C. Leptos, J. S. Guasto, J. P. Gollub, A. I. Pesci, and R. E. Goldstein, "Dynamics of enhanced tracer diffusion in suspensions of swimming eukaryotic microorganisms," *Physical Review Letters*, vol. 103, no. 19, p. 198103, 2009.

[118] B. Wang, J. Kuo, S. C. Bae, and S. Granick, "When Brownian diffusion is not Gaussian," *Nature materials*, vol. 11, no. 6, pp. 481–485, 2012.

[119] W. He, H. Song, Y. Su, L. Geng, B. J. Ackerson, H. Peng, and P. Tong, "Dynamic heterogeneity and non-Gaussian statistics for acetylcholine receptors on live cell membrane," *Nature communications*, vol. 7, no. 1, p. 11701, 2016.

[120] P. Dieterich, R. Klages, R. Preuss, and A. Schwab, "Anomalous dynamics of cell migration," *Proceedings of the National Academy of Sciences*, vol. 105, no. 2, pp. 459–463, 2008.

[121] A. G. Cherstvy, O. Nagel, C. Beta, and R. Metzler, "Non-Gaussianity, population heterogeneity, and transient superdiffusion in the spreading dynamics of amoeboid cells," *Physical Chemistry Chemical Physics*, vol. 20, no. 35, pp. 23034–23054, 2018.

[122] A. Czirók, K. Schlett, E. Madarász, and T. Vicsek, "Exponential distribution of locomotion activity in cell cultures," *Physical Review Letters*, vol. 81, no. 14, p. 3038, 1998.

[123] J. Miller, S. Tang, M. Zhong, and M. Maggioni, "Learning theory for inferring interaction kernels in second-order interacting agent systems," *Sampl Theory Signal Process. Data Anal.*, vol. 21, p. 21, 2020.

[124] P. Chaudhuri, L. Berthier, and W. Kob, "Universal nature of particle displacements close to glass and jamming transitions," *Physical Review Letters*, vol. 99, no. 6, p. 060604, 2007.

[125] "See Supplemental Material at https://journals.aps.org/prxlife/abstract/10.1103/PRXLife.1.013009 for a detailed description of the supplemental video and instructions on how to access the code."

[126] F. Ginelli, "The physics of the Vicsek model," *The European Physical Journal Special Topics*, vol. 225, pp. 2099–2117, 2016.

[127] C. A. Schneider, W. S. Rasband, and K. W. Eliceiri, "NIH Image to ImageJ: 25 years of image analysis," *Nature methods*, vol. 9, no. 7, pp. 671–675, 2012.

[128] J. Löber, F. Ziebert, and I. S. Aranson, "Collisions of deformable cells lead to collective migration," *Scientific reports*, vol. 5, no. 1, p. 9172, 2015.

[129] J. W. Tukey, *Exploratory data analysis*, vol. 2. Addision-Wesley, 1977.

[130] G. E. Box, "Non-normality and tests on variances," *Biometrika*, vol. 40, no. 3/4, pp. 318–335, 1953.

[131] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

[132] C. Guillot and T. Lecuit, "Mechanics of epithelial tissue homeostasis and morphogenesis," *Science*, vol. 340, no. 6137, pp. 1185–1189, 2013.

[133] D. Bi, J. H. Lopez, J. M. Schwarz, and M. L. Manning, "Energy barriers and cell migration in densely packed tissues," *Soft matter*, vol. 10, no. 12, pp. 1885–1890, 2014.

[134] W.-J. Rappel and L. Edelstein-Keshet, "Mechanisms of cell polarization," *Current opinion in systems biology*, vol. 3, pp. 43–53, 2017.

[135] D. Li and Y.-l. Wang, "Coordination of cell migration mediated by site-dependent

cell–cell contact," *Proceedings of the National Academy of Sciences*, vol. 115, no. 42, pp. 10678–10683, 2018.

[136] R. Mueller, J. M. Yeomans, and A. Doostmohammadi, "Emergence of active nematic behavior in monolayers of isotropic cells," *Physical Review Letters*, vol. 122, no. 4, p. 048004, 2019.

[137] A. C. Davison and D. V. Hinkley, *Bootstrap methods and their application*. No. 1, Cambridge university press, 1997.

[138] N. Sepúlveda, L. Petitjean, O. Cochet, E. Grasland-Mongrain, P. Silberzan, and V. Hakim, "Collective cell motion in an epithelial sheet can be quantitatively described by a stochastic interacting particle model," *PLoS computational biology*, vol. 9, no. 3, p. e1002944, 2013.

[139] K. H. Nagai, Y. Sumino, R. Montagne, I. S. Aranson, and H. Chaté, "Collective motion of self-propelled particles with memory," *Physical Review Letters*, vol. 114, no. 16, p. 168001, 2015.

[140] R. J. Huebner and J. B. Wallingford, "Coming to consensus: a unifying model emerges for convergent extension," *Developmental cell*, vol. 46, no. 4, pp. 389–396, 2018.

[141] S. Henkes, Y. Fily, and M. C. Marchetti, "Active jamming: Self-propelled soft particles at high density," *Physical Review E*, vol. 84, no. 4, p. 040301, 2011.

[142] E. Méhes and T. Vicsek, "Collective motion of cells: from experiments to models," *Integrative biology*, vol. 6, no. 9, pp. 831–854, 2014.

[143] E. R. Weeks, J. C. Crocker, A. C. Levitt, A. Schofield, and D. A. Weitz, "Three-dimensional direct imaging of structural relaxation near the colloidal glass transition," *Science*, vol. 287, no. 5453, pp. 627–631, 2000.

[144] H. E. Castillo and A. Parsaeian, "Local fluctuations in the ageing of a simple structural glass," *Nature physics*, vol. 3, no. 1, pp. 26–28, 2007.

[145] E. Schlossmacher, "An iterative technique for absolute deviations curve fitting," *Journal of the American Statistical Association*, vol. 68, no. 344, pp. 857–859, 1973.

[146] R. F. Phillips, "Least absolute deviations estimation via the EM algorithm," *Statistics and Computing*, vol. 12, no. 3, pp. 281–285, 2002.

[147] I. Barrodale and F. D. Roberts, "An improved algorithm for discrete l_1 linear approximation," *SIAM Journal on Numerical Analysis*, vol. 10, no. 5, pp. 839–848, 1973.

[148] I. Barrodale and F. Roberts, "Algorithm 478: Solution of an overdetermined system of equations in the l1 norm [f4]," *Communications of the ACM*, vol. 17, no. 6, pp. 319–320, 1974.

[149] J. Mattsson, H. M. Wyss, A. Fernandez-Nieves, K. Miyazaki, Z. Hu, D. R. Reichman, and D. A. Weitz, "Soft colloids make strong glasses," *Nature*, vol. 462, no. 7269, pp. 83–86, 2009.

[150] J. Song, Q. Zhang, F. de Quesada, M. H. Rizvi, J. B. Tracy, J. Ilavsky, S. Narayanan, E. Del Gado, R. L. Leheny, N. Holten-Andersen, *et al.*, "Microscopic dynamics underlying the stress relaxation of arrested soft materials," *Proceedings of the National Academy of Sciences*, vol. 119, no. 30, p. e2201566119, 2022.

[151] P. J. Green, "Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 46, no. 2, pp. 149–170, 1984.

[152] R. A. Segalman, "Patterning with block copolymer thin films," *Materials Science and Engineering: R: Reports*, vol. 48, no. 6, pp. 191–226, 2005.

[153] R. Liggins and H. Burt, "Polyether–polyester diblock copolymers for the preparation of paclitaxel loaded polymeric micelle formulations," *Advanced Drug Delivery Reviews*, vol. 54, no. 2, pp. 191–202, 2002.

[154] C. M. Bates and F. S. Bates, "50th anniversary perspective: Block polymers pure potential," *Macromolecules*, vol. 50, no. 1, pp. 3–22, 2017.

[155] S. Lee, M. J. Bluemle, and F. S. Bates, "Discovery of a Frank-Kasper σ phase in sphere-forming block copolymer melts," *Science*, vol. 330, no. 6002, pp. 349–353, 2010.

[156] M. W. Bates, J. Lequieu, S. M. Barbon, R. M. Lewis III, K. T. Delaney, A. Anastasaki, C. J. Hawker, G. H. Fredrickson, and C. M. Bates, "Stability of the A15 phase in diblock copolymer melts," *Proceedings of the National Academy of Sciences*, vol. 116, no. 27, pp. 13194–13199, 2019.

[157] C. Zhang, D. L. Vigil, D. Sun, M. W. Bates, T. Loman, E. A. Murphy, S. M. Barbon, J.-A. Song, B. Yu, G. H. Fredrickson, A. K. Whittaker, C. J. Hawker, and C. M. Bates, "Emergence of hexagonally close-packed spheres in linear block copolymer melts," *Journal of the American Chemical Society*, vol. 143, no. 35, pp. 14106–14114, 2021.

[158] S. Cui, E. A. Murphy, W. Zhang, A. Zografos, L. Shen, F. S. Bates, and T. P. Lodge, "Cylinders-in-undulating-lamellae morphology from abc bottlebrush block terpolymers," *Journal of the American Chemical Society*, vol. 146, no. 10, pp. 6796–6805, 2024.

[159] H. Lee, S. Kwon, J. Min, S.-M. Jin, J. H. Hwang, E. Lee, W. B. Lee, and M. J.

Park, "Thermodynamically stable plumber's nightmare structures in block copolymers," *Science*, vol. 383, no. 6678, pp. 70–76, 2024.

[160] C. Sinturel, F. S. Bates, and M. A. Hillmyer, "High χ–low N block polymers: How far can we go?," *ACS Macro Letters*, vol. 4, no. 9, pp. 1044–1050, 2015.

[161] R. Gupta, M. Misra, W. Zhang, A. Mukhtyar, S. P. Gido, A. Ribbe, F. A. Escobedo, and E. B. Coughlin, "Topological frustration as a new parameter to tune morphology revealed through exploring the continuum between A-B-C 3-arm star and linear triblock polymers," *Macromolecules*, vol. 54, no. 9, pp. 4401–4411, 2021.

[162] E. A. Murphy, S. J. Skala, D. Kottage, P. A. Kohl, Y. Li, C. Zhang, C. J. Hawker, and C. M. Bates, "Accelerated discovery and mapping of block copolymer phase diagrams," *Physical Review Materials*, vol. 8, no. 1, p. 015602, 2024.

[163] R. V. Garcia, E. A. Murphy, N. J. Sinha, Y. Okayama, J. M. Urueña, M. E. Helgeson, C. M. Bates, C. J. Hawker, R. D. Murphy, and J. Read de Alaniz, "Tailoring writability and performance of star block copolypeptides hydrogels through side-chain design," *Small*, vol. 19, no. 50, p. 2302794, 2023.

[164] S. Bae and K. G. Yager, "Chain redistribution stabilizes coexistence phases in block copolymer blends," *ACS Nano*, vol. 16, no. 10, pp. 17107–17115, 2022.

[165] E. Helfand, "Block copolymer theory. III. statistical mechanics of the microdomain structure," *Macromolecules*, vol. 8, no. 4, pp. 552–556, 1975.

[166] L. Leibler, "Theory of microphase separation in block copolymers," *Macromolecules*, vol. 13, no. 6, pp. 1602–1617, 1980.

[167] M. W. Matsen and M. Schick, "Stable and unstable phases of a diblock copolymer melt," *Physical Review Letters*, vol. 72, no. 16, p. 2660, 1994.

[168] T. Shefelbine, M. E. Vigild, M. Matsen, D. Hajduk, M. Hillmyer, E. Cussler, and F. Bates, "Core-shell gyroid morphology in a poly(isoprene-*block*-styrene-*block*-dimethylsiloxane) triblock copolymer," *Journal of the American Chemical Society*, vol. 121, no. 37, pp. 8457–8465, 1999.

[169] C. A. Tyler and D. C. Morse, "Orthorhombic *Fddd* network in triblock and diblock copolymer melts," *Physical Review Letters*, vol. 94, no. 20, p. 208302, 2005.

[170] B. Vorselaars, J. U. Kim, T. L. Chantawansri, G. H. Fredrickson, and M. W. Matsen, "Self-consistent field theory for diblock copolymers grafted to a sphere," *Soft Matter*, vol. 7, no. 11, pp. 5128–5137, 2011.

[171] G. Khaira, M. Doxastakis, A. Bowen, J. Ren, H. S. Suh, T. Segal-Peretz, X. Chen, C. Zhou, A. F. Hannon, N. J. Ferrier, *et al.*, "Derivation of multiple covarying material and process parameters using physics-based modeling of X-ray data," *Macromolecules*, vol. 50, no. 19, pp. 7783–7793, 2017.

[172] T. Aoyagi, "Deep learning model for predicting phase diagrams of block copolymers," *Computational Materials Science*, vol. 188, p. 110224, 2021.

[173] S. Zhao, T. Cai, L. Zhang, W. Li, and J. Lin, "Autonomous construction of phase diagrams of block copolymers by theory-assisted active machine learning," *ACS Macro Letters*, vol. 10, no. 5, pp. 598–602, 2021.

[174] M. E. Deagen, B. Dalle-Cort, N. J. Rebello, T.-S. Lin, D. J. Walsh, and B. D. Olsen, "Machine translation between BigSMILES line notation and chemical structure diagrams," *Macromolecules*, vol. 57, no. 1, pp. 42–53, 2023.

[175] J. A. Mysona, P. F. Nealey, and J. J. de Pablo, "Machine learning models and

dimensionality reduction for prediction of polymer properties," *Macromolecules*, vol. 57, no. 5, pp. 1988–1997, 2024.

[176] M. G. Wessels and A. Jayaraman, "Computational reverse-engineering analysis of scattering experiments (CREASE) on amphiphilic block polymer solutions: cylindrical and fibrillar assembly," *Macromolecules*, vol. 54, no. 2, pp. 783–796, 2021.

[177] C. M. Heil, A. Patil, A. Dhinojwala, and A. Jayaraman, "Computational reverse-engineering analysis for scattering experiments (CREASE) with machine learning enhancement to determine structure of nanoparticle mixtures and solutions," *ACS Central Science*, vol. 8, no. 7, pp. 996–1007, 2022.

[178] C. M. Heil, Y. Ma, B. Bharti, and A. Jayaraman, "Computational reverse-engineering analysis for scattering experiments for form factor and structure factor determination ("p(q) and s(q) CREASE")," *JACS Au*, vol. 3, no. 3, pp. 889–904, 2023.

[179] K. Vaddi, K. Li, and L. D. Pozzo, "Metric geometry tools for automatic structure phase map generation," *Digital Discovery*, vol. 2, no. 5, pp. 1471–1483, 2023.

[180] L. Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[181] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, pp. 123–140, 1996.

[182] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of statistics*, vol. 29, no. 5, pp. 1189–1232, 2001.

[183] E. A. Murphy, C. Zhang, C. M. Bates, and C. J. Hawker, "Chromatographic separation: A versatile strategy to prepare discrete and well-defined polymer libraries," *Accounts of Chemical Research*, vol. 57, no. 8, pp. 6306–6310, 2024.

[184] C. Zhang, M. W. Bates, Z. Geng, A. E. Levi, D. Vigil, S. M. Barbon, T. Loman,

K. T. Delaney, G. H. Fredrickson, C. M. Bates, *et al.*, "Rapid generation of block copolymer libraries using automated chromatographic separation," *Journal of the American Chemical Society*, vol. 142, no. 21, pp. 9843–9849, 2020.

[185] E. A. Murphy, Y.-Q. Chen, K. Albanese, J. R. Blankenship, A. Abdilla, M. W. Bates, C. Zhang, C. M. Bates, and C. J. Hawker, "Efficient creation and morphological analysis of abc triblock terpolymer libraries," *Macromolecules*, vol. 55, no. 19, pp. 8875–8882, 2022.

[186] B. Oschmann, J. Lawrence, M. W. Schulze, J. M. Ren, A. Anastasaki, Y. Luo, M. D. Nothling, C. W. Pester, K. T. Delaney, L. A. Connal, A. J. McGrath, P. G. Clark, C. M. Bates, and C. J. Hawker, "Effects of tailored dispersity on the self-assembly of dimethylsiloxane–methyl methacrylate block co-oligomers," *ACS Macro Letters*, vol. 6, no. 7, pp. 668–673, 2017.

[187] N. J. Szymanski, C. J. Bartel, Y. Zeng, M. Diallo, H. Kim, and G. Ceder, "Adaptively driven X-ray diffraction guided by machine learning for autonomous phase identification," *npj Computational Materials*, vol. 9, no. 1, p. 31, 2023.

[188] J.-W. Lee, W. B. Park, J. H. Lee, S. P. Singh, and K.-S. Sohn, "A deep-learning technique for phase identification in multiphase inorganic compounds using synthetic XRD powder patterns," *Nature Communications*, vol. 11, no. 1, p. 86, 2020.

[189] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE.," *Journal of Machine Learning Research*, vol. 9, no. 11, pp. 2579–2605, 2008.

[190] J. L. Hintze and R. D. Nelson, "Violin plots: A box plot-density trace synergism," *The American Statistician*, vol. 52, no. 2, pp. 181–184, 1998.

[191] A. P. Bartók, M. C. Payne, R. Kondor, and G. Csányi, "Gaussian approximation

potentials: The accuracy of quantum mechanics, without the electrons," *Physical Review Letters*, vol. 104, no. 13, p. 136403, 2010.

[192] S. Chmiela, H. E. Sauceda, K.-R. Müller, and A. Tkatchenko, "Towards exact molecular dynamics simulations with machine-learned force fields," *Nature Communications*, vol. 9, no. 1, pp. 1–10, 2018.

[193] A. V. Shapeev, "Moment tensor potentials: A class of systematically improvable interatomic potentials," *Multiscale Modeling & Simulation*, vol. 14, no. 3, pp. 1153–1173, 2016.

[194] K. T. Schütt, H. E. Sauceda, P.-J. Kindermans, A. Tkatchenko, and K.-R. Müller, "SchNet–a deep learning architecture for molecules and materials," *The Journal of Chemical Physics*, vol. 148, no. 24, p. 241722, 2018.

[195] S.-C. Lin, G. Martius, and M. Oettel, "Analytical classical density functionals from an equation learning network," *The Journal of Chemical Physics*, vol. 152, no. 2, p. 021102, 2020.

[196] L. Li, S. Hoyer, R. Pederson, R. Sun, E. D. Cubuk, P. Riley, K. Burke, *et al.*, "Kohn-Sham equations as regularizer: Building prior knowledge into machine-learned physics," *Physical Review Letters*, vol. 126, no. 3, p. 036401, 2021.

[197] V. L. Deringer, A. P. Bartók, N. Bernstein, D. M. Wilkins, M. Ceriotti, and G. Csányi, "Gaussian process regression for materials and molecules," *Chemical Reviews*, vol. 121, no. 16, pp. 10073–10141, 2021.

[198] P. Yatsyshin, S. Kalliadasis, and A. B. Duncan, "Physics-constrained Bayesian inference of state functions in classical density-functional theory," *The Journal of Chemical Physics*, vol. 156, no. 7, p. 074105, 2022.

[199] J. Westermayr, M. Gastegger, K. T. Schütt, and R. J. Maurer, "Perspective on integrating machine learning into computational chemistry and materials science," *The Journal of Chemical Physics*, vol. 154, no. 23, p. 230903, 2021.

[200] O. T. Unke, S. Chmiela, H. E. Sauceda, M. Gastegger, I. Poltavsky, K. T. Schütt, A. Tkatchenko, and K.-R. Müller, "Machine learning force fields," *Chemical Reviews*, vol. 121, no. 16, pp. 10142–10186, 2021. PMID: 33705118.

[201] M. Sajjan, J. Li, R. Selvarajan, S. H. Sureshbabu, S. S. Kale, R. Gupta, V. Singh, and S. Kais, "Quantum machine learning for chemistry and physics," *Chemical Society Reviews*, 2022.

[202] A. W. van der Vaart and J. H. van Zanten, "Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth," *The Annals of Statistics*, pp. 2655–2675, 2009.

[203] E. Uteva, R. S. Graham, R. D. Wilkinson, and R. J. Wheatley, "Active learning in Gaussian process interpolation of potential energy surfaces," *The Journal of Chemical Physics*, vol. 149, no. 17, p. 174114, 2018.

[204] J. Vandermause, S. B. Torrisi, S. Batzner, Y. Xie, L. Sun, A. M. Kolpak, and B. Kozinsky, "On-the-fly active learning of interpretable Bayesian force fields for atomistic rare events," *npj Computational Materials*, vol. 6, no. 1, pp. 1–11, 2020.

[205] M. Gu and L. Wang, "Scaled Gaussian stochastic process for computer model calibration and prediction," *SIAM/ASA Journal on Uncertainty Quantification*, vol. 6, no. 4, pp. 1555–1583, 2018.

[206] A. Sauer, R. B. Gramacy, and D. Higdon, "Active learning for deep Gaussian process surrogates," *Technometrics*, pp. 1–15, 2022.

[207] X. Yue, Y. Wen, J. H. Hunt, and J. Shi, "Active learning for Gaussian process considering uncertainties with application to shape control of composite fuselage," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 1, pp. 36–46, 2020.

[208] R. Evans, *Density functionals in the theory of nonuniform fluids*, pp. 85–175. New York: Marcel Dekker, 1992.

[209] J. Wu and Z. Li, "Density-functional theory for complex fluids," *Annual Review of Physical Chemistry*, vol. 58, pp. 85–112, 2007.

[210] J. Percus, "Equilibrium state of a classical fluid of hard rods in an external field," *Journal of Statistical Physics*, vol. 15, no. 6, pp. 505–511, 1976.

[211] L. Shang-Chun and M. Oettel, "A classical density functional from machine learning and a convolutional neural network," *SciPost Physics*, vol. 6, no. 2, p. 025, 2019.

[212] P. Cats, S. Kuipers, S. De Wind, R. Van Damme, G. M. Coli, M. Dijkstra, and R. Van Roij, "Machine-learning free-energy functionals using density profiles from simulations," *APL Materials*, vol. 9, no. 3, p. 031109, 2021.

[213] J. C. Snyder, M. Rupp, K. Hansen, K.-R. Müller, and K. Burke, "Finding density functionals with machine learning," *Physical Review Letters*, vol. 108, no. 25, p. 253002, 2012.

[214] F. Brockherde, L. Vogt, L. Li, M. E. Tuckerman, K. Burke, and K.-R. Müller, "By-passing the Kohn-Sham equations with machine learning," *Nature Communications*, vol. 8, no. 1, pp. 1–10, 2017.

[215] M. Rupp, A. Tkatchenko, K.-R. Müller, and O. A. Von Lilienfeld, "Fast and accurate

modeling of molecular atomization energies with machine learning," *Physical Review Letters*, vol. 108, no. 5, p. 058301, 2012.

[216] S. Chmiela, A. Tkatchenko, H. E. Sauceda, I. Poltavsky, K. T. Schütt, and K.-R. Müller, "Machine learning of accurate energy-conserving molecular force fields," *Science Advances*, vol. 3, no. 5, p. e1603015, 2017.

[217] H. Li, M. Zhou, J. Sebastian, J. Wu, and M. Gu, "Efficient force field and energy emulation through partition of permutationally equivalent atoms," *The Journal of Chemical Physics*, vol. 156, no. 18, p. 184304, 2022.

[218] E. Fix and J. L. Hodges, "Discriminatory analysis. nonparametric discrimination: Consistency properties," *International Statistical Review/Revue Internationale de Statistique*, vol. 57, no. 3, pp. 238–247, 1989.

[219] P. G. Constantine, E. Dow, and Q. Wang, "Active subspace methods in theory and practice: applications to kriging surfaces," *SIAM Journal on Scientific Computing*, vol. 36, no. 4, pp. A1500–A1524, 2014.

[220] J. Guinness, "Permutation and grouping methods for sharpening Gaussian process approximations," *Technometrics*, vol. 60, no. 4, pp. 415–429, 2018.

[221] J. Guinness, M. Katzfuss, Y. J. Hou, and Z. Lang, *GpGp: Fast Gaussian Process Computation Using Vecchia's Approximation*, 2024. R package version 0.5.0.

[222] J. Seidman and E. T. Spiller, "VPPE: Application of scaled Vecchia approximations to parallel partial emulation," *arXiv preprint arXiv:2508.19144*, 2025.

[223] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[224] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[225] J. Gao and C. Long, "RaBitQ: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search," *Proceedings of the ACM on Management of Data*, vol. 2, no. 3, pp. 1–27, 2024.

[226] J. Gao, Y. Gou, Y. Xu, Y. Yang, C. Long, and R. C.-W. Wong, "Practical and asymptotically optimal quantization of high-dimensional vectors in Euclidean space for approximate nearest neighbor search," *Proceedings of the ACM on Management of Data*, vol. 3, no. 3, pp. 1–26, 2025.

[227] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, H. Wang, H. Wang, *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, vol. 2, no. 1, p. 32, 2023.

[228] A. Zandieh, M. Daliri, M. Hadian, and V. Mirrokni, "TurboQuant: Online vector quantization with near-optimal distortion rate," *arXiv preprint arXiv:2504.19874*, 2025.

[229] K. R. Anderson, I. A. Johanson, M. R. Patrick, M. Gu, P. Segall, M. P. Poland, E. K. Montgomery-Brown, and A. Miklius, "Magma reservoir failure and the onset of caldera collapse at Kīlauea volcano in 2018," *Science*, vol. 366, no. 6470, 2019.

[230] J. R. Stroud, M. L. Stein, and S. Lysen, "Bayesian and maximum likelihood estimation for Gaussian processes on an incomplete lattice," *Journal of Computational and Graphical Statistics*, vol. 26, no. 1, pp. 108–120, 2017.

[231] J. Shao and D. Tu, *The jackknife and bootstrap*. Springer Science & Business Media, 2012.

[232] R. B. Gramacy, "laGP: large-scale spatial modeling via local approximate Gaussian processes in R," *Journal of Statistical Software*, vol. 72, no. 1, pp. 1–46, 2016.

[233] V. Picheny, T. Wagner, and D. Ginsbourger, "A benchmark of kriging-based infill criteria for noisy optimization," *Structural and Multidisciplinary Optimization*, vol. 48, no. 3, pp. 607–626, 2013.

[234] R. Bürgmann, P. A. Rosen, and E. J. Fielding, "Synthetic aperture radar interferometry to measure Earth's surface topography and its deformation," *Annual Review of Earth and Planetary Sciences*, vol. 28, pp. 169–209, may 2000.

[235] M. Gu, K. Anderson, and E. McPhillips, "Calibration of imperfect geophysical models by multiple satellite interferograms with measurement bias," *Technometrics*, vol. 65, no. 4, pp. 453–464, 2023.

[236] J. Nocedal, "Updating quasi-Newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.

[237] P. Tarazona, J. A. Cuesta, and Y. Martínez-Ratón, "Density functional theories of hard particle systems," in *Theory and simulation of hard-sphere fluids and related systems*, pp. 247–341, Springer, 2008.

[238] T. Vanderlick, H. Davis, and J. Percus, "The statistical mechanics of inhomogeneous hard rod mixtures," *The Journal of Chemical Physics*, vol. 91, no. 11, pp. 7136–7145, 1989.

[239] B. Bakhti, M. Karbach, P. Maass, and G. Müller, "Monodisperse hard rods in external potentials," *Physical Review E*, vol. 92, no. 4, p. 042112, 2015.