Development of Small Angle Neutron Scattering Tools for Probing the Phase-Behavior and Self-Assembly of Sequence-Defined Polymers

by

Kexin (Charlotte) Dai

B. S. Chemical Engineering, University of California, Santa Barbara, 2020

M.S. Chemical Engineering Practice, Massachusetts Institute of Technology, 2024

Submitted to the Department of Chemical Engineering

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

February 2026

Signature of Author: _____
Department of Chemical Engineering
September 19, 2025

Certified by: _____
Bradley D. Olsen
Alexander and I. Michael (1960) Kasser Professor of Chemical Engineering
Thesis Supervisor

Accepted by: _____
Hadley Sikes
Willard Henry Dow Professor in Chemical Engineering

Chairman, Committee for Graduate Students

Development of Small Angle Neutron Scattering Tools for Probing the Phase-Behavior and Self-Assembly of Sequence-Defined Polymers

Kexin (Charlotte) Dai

Submitted to the Department of Chemical Engineering on September 23, 2025, in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Chemical Engineering

Abstract

Sequence control in polymers offer a powerful platform for programming self-assembly and phase behavior with monomer-level precision, bridging the structural control of biological macromolecules and the functional tunability of synthetic polymers. However, understanding how primary sequence dictates structure and material properties remains a fundamental challenge due to the vast design space and the lack of high-throughput characterization tools and predictive models. This thesis addresses these challenges by combining recombinant polymer synthesis, coarse-grained simulations, and information-guided scattering techniques to interrogate how molecular-level sequence features drive mesoscale structure, phase behavior, and self-assembly in dilute and concentrated aqueous environments.

To enable efficient structural screening, a high-throughput small-angle neutron scattering (SANS) workflow was developed to rapidly evaluate large libraries of sequence-defined polymers under diverse solvent and temperature conditions. By analyzing three classes of model polymer systems and simulating reduced-count datasets, it is shown that accurate parameter estimates (within 5–10% of full-count values) can be obtained using only 1–50% of the original counts, depending on the sample and parameter. Both AIC and BIC can successfully identify the correct model from a candidate set, even with limited data. This approach provides a practical framework to optimize SANS beamtime and supports efficient structural characterization of material libraries.

On the experimental end, elastin like polypeptides (ELPs) were used as a model system to understand sequence-property relationships. Thermoresponsive behavior of ABA triblock ELPs was systematically investigated. Two variants differing only in midblock hydrophobicity revealed distinct phase behaviors: one formed temperature-triggered micellar aggregates with syneresis, while the other remained viscoelastic and unstructured across conditions. SANS, rheology, and depolarized light scattering collectively demonstrated how sequence-encoded hydrophobicity and block architecture govern micelle formation, water partitioning, and gel properties near the LCST.

The sequence-property landscape was mapped using a series of ELPs with varied guest residues. Turbidimetry in water/ethanol/salt mixtures uncovered how hydrophobicity and charge combine define cononsolvency boundaries and critical transition temperatures, highlighting the role of solvation competition in shaping UCST- and LCST-type transitions.

Finally, a highly coarse-grained dumbbell model was developed for globular protein–polymer bioconjugates. Molecular dynamics simulations demonstrated that treating the protein as a hard sphere tethered to a soft polymer tail as a soft sphere captures the essential physics of self-assembly in those systems. Simulated phase diagrams aligned with experimental trends and

provided a predictive model for designing protein-polymer bioconjugates and understanding the thermodynamic driving force of self-assembly.

By integrating high-throughput structural characterization, insightful coarse-grained models, and large amount of experimental data, this thesis provides quantitative design rules for encoding self-assembly and transitions across molecular, mesoscale, and macroscopic regimes—paving the way for next-generation programmable polymer materials.

Thesis Supervisor: Bradley D. Olsen, Alexander and I. Michael (1960) Kasser Professor of

Chemical Engineering

Acknowledgements

My Ph.D. journey would have not been possible without the continuous support of many people. I have grown so much both academically and personally during this journey and found my true passion during my time at MIT.

First, I would like to thank Professor Bradley Olsen for the rigorous scientific training and support for the last five and half years. His guidance has shaped me to become a better scientist, engineer, and problem solver. I am also grateful for Brad to give me the opportunity to explore projects that provide me training in both experimental and simulation work to gain many valuable research skills and to explore my interests. I am always inspired by his scientific insight, rigor, and contribution to polymer science.

Next, I would like to thank my committee. Professor Ariel Furst and Professor Patrick Doyle for their support, advice, and questions during my committee meeting and their encouragement when research is not going as well as expected. Professor Furst had been kindly giving me access to her lab equipment when our equipment is broken down, which has saved me so much during scattering preparation. She also offered me to go with her group on SAXS scattering trip and to get valuable preliminary data at Brookhaven National Lab.

The Olsen lab has taught me many things both academically and personally; given the interdisciplinary research environment it fosters and the expertise from people with different backgrounds. It would not have been possible to complete my Ph.D. without the generous help, support, and encouragement from past and present Olsen lab members. I have been inspired by amazing female scientists in the group. In particular, I want to thank my deskmate and scattering buddy Professor Yu Zheng. She has provided me so much helpful advice and support when I am stuck and feel unmotivated to move forward. She was not only an amazing, talented scientist, but also a role model for me personally and for people in the group. I would like to thank Dr. Helen Yao for giving me the chance to work on Chapter 6 of my thesis, which was one of my favorite projects. Even though she has graduated, she is still very passionate about her work and science. One of my favorite quotes from her was, “Do what is the best for science.” She would make time to meet with me after busy day of work and give me suggestions on any questions I have. Thanks to her and Brad, I was able to pick up the simulations in a short period of time. I would also like to thank the protein subgroup, Dr. Oliver Xie, Zhi Kai Tio, Dr. Zixian Cui, and Hannah Ulh for their helpful scientific discussion and generous support. Especially Zhi Kai and Zixian, who helped me so much with experimental work after I already moved. I would like to thank everyone in the Olsen group for the great scientific discussions, generous help with experiments, and their kind support during my time in the Olsen group: Dr. Ameya Rao, Dr. Haley Beech, Dr. Weizhong Zou, Dr. Celestine Hong, Dr. Smita Mankar, Dr. Katharina Fransen, Clara Troyano-Valls, Devosmita Sen, Alexis Hocken, Natalie Mamrol, Gabrielle Godbille-Cardona, Brian Carrick, Alex Zappi. Special thanks go to Devosmita and Zixian for proofreading Chapter 2 of my thesis and gave helpful feedback!

The scattering work in the thesis would not have been possible without the help of beamline scientists at Oak Ridge National Lab, Dr. Lilin He and Dr. Changwoo Doo. During my last scattering trip, Dr Lilin He helped me to run SANS experiment from 4PM to 11PM to ensure everything was successful.

My family has been incredibly supportive during my Ph.D. journey. My dad was diagnosed with cancer during the fourth year of my Ph.D. Even though it has been tough and makes me extremely worry, he was able to fight it quickly thanks Keytruda and Chemotherapy. My mom has been supporting me through taking care of my daughter Ai and me during my Ph.D. so I can focus 200% on research in the last few months. They have taught me not only to focus on success academically, but also how to become a caring, honest, and resilient person.

I would also like to thank my husband Dr. Andrew Salmon and his family. It has been a privilege to be able to solve all the math and coding issues for my research projects elegantly and creatively. He has made me think quantitatively and realize the beauty of mathematical modeling, encouraging me to explore areas that I thought I am not good. More importantly, he also taught me many other lessons outside of research. I want to thank my mother-in-law Julie and father-in-law Chris for their generous help and support to help with taking care of Ai when I am still in Ph.D.

I have made a lot of friends outside of lab during my time. I would like to thank them for making me feeling welcomed and aspired. Dr. Oscar Wu, Zijie Liu, Clara Troyano-Valls, Kariana Sader, Dr. Herry Jin, Dr. Rhoda Zhang, Dr. Wenhao Gao, Maysa Ilamanova, Nathan Ewell, Dr. Bhavish Dinakar. Dr. Oscar Wu has been my good friend since UCSB and a role model who convinced me to do my Ph.D. at MIT during my visit weekend. I loved one of his quotes that I am translating from Chinese to English here, “We should not be like candles that burn to light up others, because their light does not last long. Instead, we should be like the sun, able to shine endlessly and give infinite light.” Indeed, he has been like the sun for everyone around him, including myself. He has taught me different ways of thinking about life, and I really aspire to be like him. Clara and Kariana have been so sweet to me and provided me so much emotional support when I really needed it. Herry, Rhoda, Wenhao, thank you for helping a lot during first year core classes and the fun get together during the pandemic. Nathan and Bhavish’s humor had made practice school experience so fun.

Professor Katie Galloway and Professor Hadley Sikes have been an incredible role models and provided great support throughput my Ph.D. They have been so friendly and generous with her time. As a Ph.D. mom, I really appreciate her advice with balancing Ph.D. and having a child.

I would like to thank my academic role models from UCSB. Professor Mike Gordon. I would not have been pursuing a Ph.D. degree without his positive influence and encouragement during my undergraduate time at UCSB. Professor Matt Helgeson was my undergraduate research advisor, and he trusted me with independence in research, who showed me the interesting aspects of polymer and soft matter research.

Table of Contents

List of Figures

Chapter 1 Introduction

1.1 Motivation

The primary sequence of a polymer determines its chain conformation[1, 2], inter- and intramolecular interactions[3, 4], and ultimately its material properties[5]. In sequence-defined biopolymers, such as DNA and proteins, the sequence–structure–property relationship is fundamental. The specific order of nucleotides in DNA not only encodes genetic information, but also governs its ability to adopt double helical structures, form higher-order packing, and participate in recognition events such as transcription or hybridization. Similarly, the primary amino acid sequence of a protein dictates its chain conformation and folding pathway, resulting in distinct secondary and tertiary structures stabilized by non-covalent interactions. These folded structures, in turn, determine the protein's mechanical stability, chemical reactivity, and binding specificity. In both cases, a well-defined sequence leads to predictable structures, which ultimately give rise to functional properties essential for biological activity.

Extending this principle to synthetic polymers opens avenues for designing materials that can achieve diverse functionality, such as information storage[6], cellular signaling[7], catalysis[8], and programmable self-assembly[9]. Sequence-defined polymers can bridge the gap between conventional synthetic polymers, which are chemically diverse but functionally limited, and biological polymers, which are highly functional but compositionally constrained[10] as shown in Figure 1-1. New fundamental understanding and tools are required to guide the rational and efficient design of the vast sequence space in sequence-defined polymers.

Some of the grand challenges in experimentally designing sequence-defined polymers lie in both material synthesis and characterization, including the synthesis of polymers that can match the precision and functionality of those in nature[11], design of the sequence on the monomer level

to achieve functionality[5], and high throughput screening and characterization of sequence-defined polymer libraries[10] to identify sequence-property relationships.

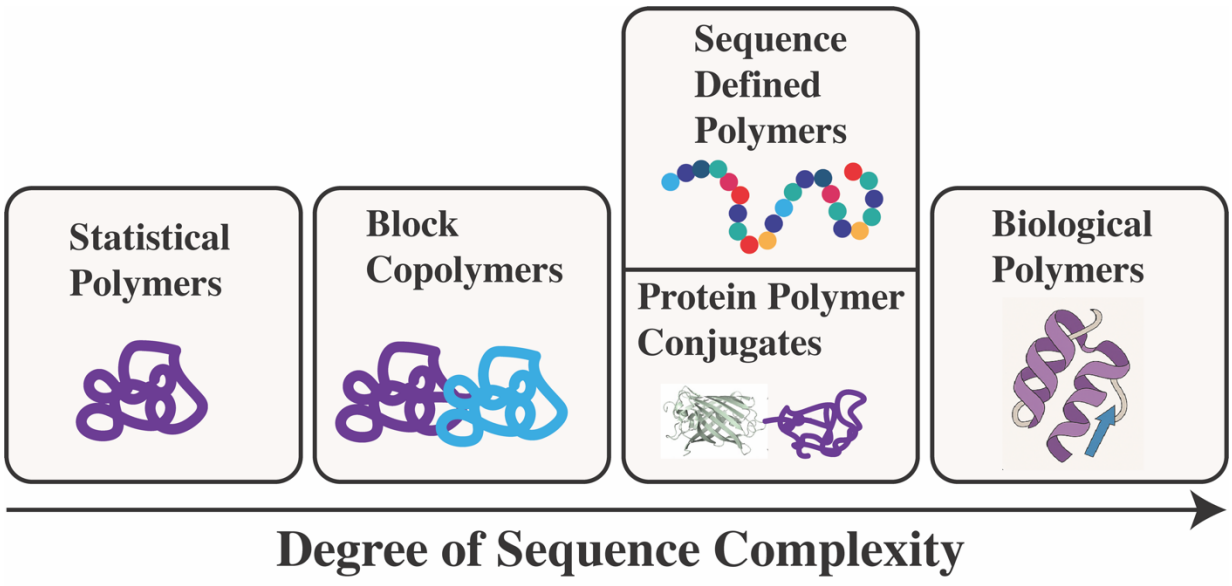


Figure 1-1. Different classes of linear polymers with varying degrees of sequence complexity. Sequence-defined polymers bridge the gap between biological and synthetic polymers. Adapted from Rosales et al[12].

This thesis tackles those challenges by combining experimental design and simulation tools with high-throughput structural characterization to investigate sequence-property relationships. To understand the sequence-property relationships from molecular interactions, elastin like polypeptides (ELPs) and protein–polymer bioconjugates provide biologically inspired, polymers with high degree of sequence complexity, whose self-assembly and phase behavior can be systematically tuned through variations in sequence composition, patterning, and length. On the other hand, studying these systems one at a time is impractical given the vast design space. To meet this challenge, a high-throughput small-angle neutron scattering (SANS) algorithm was developed that enables rapid, quantitative screening of large polymer libraries under diverse

solution conditions. Together, these efforts aim to provide insight and useful tools for rational design and effectively characterization of sequence-defined polymers.

1.2 Block Copolymers Self Assembly

1.2.1 Synthetic Block Copolymers

As illustrated in Figure 1-1, one of the earlier examples that demonstrates the importance of sequence control is block copolymers. These polymers consist of two or more covalently linked, chemically distinct polymer blocks that are immiscible yet forced into proximity, resulting in microphase separation on the nanometer scale[13]. Microphase separation can form various nanostructures, such as lamellar, hexagonally packed cylinder, gyroid, body-centered cubic, etc, depending on the segregation length (χN) or the volume fraction of the block[14]. χ stands for the Flory-Huggins interaction parameter and N stands for the degree of polymerization of the block copolymer.

The classical phase diagram of AB diblock copolymers, derived through mean-field theory and confirmed by experiments as shown in Figure 1-2, was first systematically mapped by Bates and Fredrickson using small-angle X-ray and neutron scattering techniques[15]. As block architectures grow more complex—such as ABA triblocks, star-shaped polymers, or ABC triblock terpolymers—the range of accessible morphologies expands significantly. Recently, linear ABC triblock terpolymers when mixed with linear AB diblock copolymers can form more complex nanostructures, including core-shell gyroids hexagonal-packed alternating cylinders, sphere-in-perforated lamellae, by tuning the χ parameters between each block pair and the volume fractions of each polymer blend[16]. Using directed self-assembly, a 3-miktoarm star terpolymer (polyisoprene–polystyrene–polyferrocenylethylmethylsilane) can form a highly ordered (4.8^2) Archimedean tiling pattern (p4mm symmetry), with phase transitions dependent on annealing

conditions and film thickness, enabling tunable square-symmetry nanostructures in both bulk and thin-film morphologies[17].

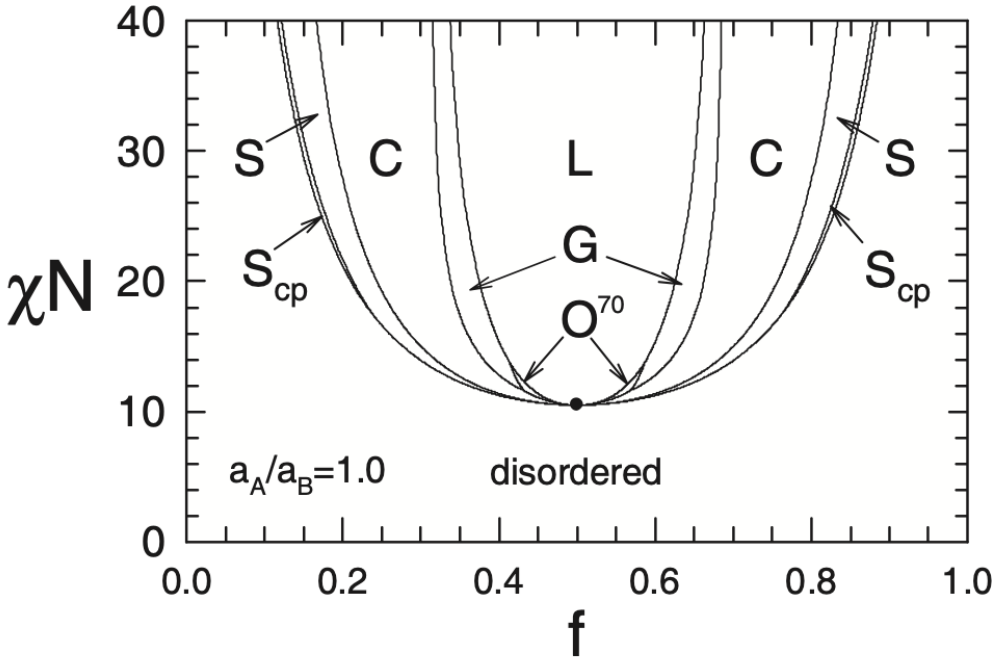


Figure 1-2. AB diblock copolymer phase diagram derived from self-consistent mean field theory (SCFT) calculations. S, S': spheres, body-center cubic (BCC) spheres; Scp: close-packed spheres; C, C': hexagonally packed cylinders; G, G': gyroid; O70: orthorhombic. The (') indicates an inverse phase. The ratio a_A:a*B* indicates the ratio of statistical segment lengths. Figure reproduced from Matson *et al*[18] with copyright permission from Springer Nature.

In addition to melt-state self-assembly, block copolymers exhibit rich behavior in solution as shown in Figure 1-3. The presence of a selective solvent introduces an additional thermodynamic lever by altering the effective χ between blocks and the solvent, thereby influencing the solubility of each block and their spatial organization. In dilute or semi-dilute solutions above the critical micelle concentration (CMC), amphiphilic block copolymers can self-

assemble into micelles, with one block forming a solvophobic core and the other forming a solvophilic corona. The morphology of these aggregates—spherical micelles, wormlike micelles, vesicles—depends on the packing parameter, chain length, and solvent conditions. As concentration increases, these micelles can interact and eventually order into crystalline arrays such as body-centered cubic or face-centered cubic lattices, forming "hard" physical gels[19]. With further concentration or temperature tuning, these ordered lattices may melt into disordered but percolating micelle networks, resulting in "soft" gels with viscoelastic behavior[20]. This gelation behavior is not only of fundamental interest but also forms the basis for stimuli-responsive materials, injectable hydrogels, and nanocarrier systems.

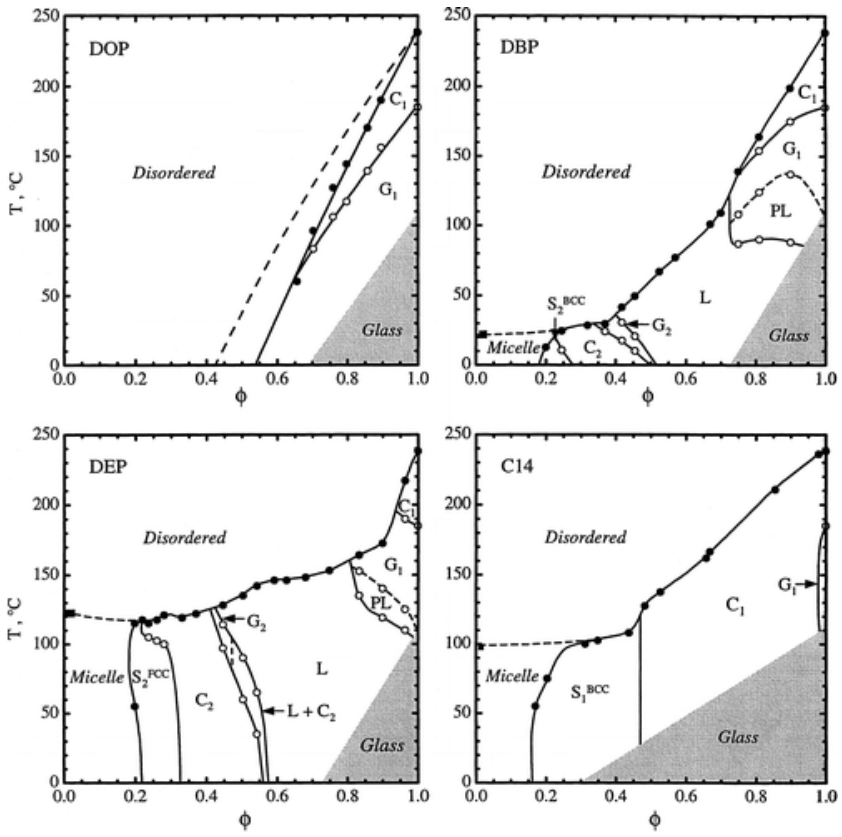


Figure 1-3. Phase diagram for a polystyrene-b-polyisoprene (PS-b-PI) diblock with Mn = 3.2 x 10^4 g/mol and f_{PS} = 0.31 in the solvents indicated. The volume fraction of polymer is denoted f.

The critical micelle temperature in dilute solution is indicated by a filled square. The ordered phases are denoted: L, lamellae; C, hexagonal-packed cylinders; G, gyroid; PL, perforated lamellae; S, cubic-packed spheres. The subscript 1 indicates a normal phase (minority PS component in minority domains) and 2 indicates an inverted phase (PS in majority domains). The smooth curves are guides to the eyes, except for DOP in which the order-order transition and order-disorder transitions phase boundaries (solid lines) show the previously determined scaling of the PS-PI interaction parameter. The dashed line corresponds to the dilution approximation. Reproduced from Hanley et al[20] with copyright permission from American Chemical Society.

Moreover, external stimuli—temperature, pH, ionic strength, or cosolvents—can modulate block interactions and phase behavior. For instance, temperature-responsive block copolymers like poly(N-isopropylacrylamide)-b-poly(ethylene glycol) exhibit lower critical solution temperature (LCST) transitions that allow reversible sol–gel transitions[21]. pH-sensitive systems have also been extensively studied in drug delivery applications, enabling enhanced tumor targeting[22].

Collectively, these studies demonstrate that even coarse sequence control at the block level enables remarkable control over nanoscale structure and macroscopic material properties. However, as synthetic methods advance, the field is now pushing toward monomer-level sequence control, aiming to recapitulate the precision and functional complexity of biological polymers.

1.2.2 Elastin-like Polypeptides Based Block Copolymer

Unlike synthetic polymers, polypeptides offer monodispersity and sequence precision via recombinant expression, enabling detailed control over interactions such as hydrophobicity and charge. This makes them ideal for studying how primary sequence influences self-assembly in solution. Elastin-like polypeptides (ELPs) are a family of recombinant biopolymers inspired by

the mammalian protein tropoelastin, composed of repeating pentapeptide units with the consensus motif $(VPGXG)_n$, where the guest residue X can be systematically varied to scalably produce a form of sequence-controlled macromolecules. ELPs display lower critical solution temperature (LCST) phase behavior in aqueous solution, transitioning from a soluble to an insoluble phase upon heating above a sequence- and environment-dependent transition temperature (Tt). This reversible phase transition has been extensively studied and is known to depend on parameters such as hydrophobicity, charge, concentration, molecular weight, and solvent conditions. Urry et al found that the identity and hydrophobicity of the guest residue exert a strong influence, with more hydrophobic substitutions (e.g., Ile, Leu) lowering the Tt and promoting aggregation[23].

Studies in ELPs have been primarily focused on dilute or semi-dilute phase behaviors driven by their relevant applications. For example, Urry et al. established the hydrophobicity scale based on the ELPs' transition temperatures at semi-dilute conditions[23]. They found that substituting more hydrophobic guest residues in the ELP sequence can decrease the transition temperature, while substituting charged or hydrophilic guest residues can increase the transition temperature. Conticello and coworkers pioneered the work of ELP block copolymer self-assembly in solution[24]. With $[VPGEG(IPGAG)_4]_{14}$ as the hydrophilic and $[VPGFG(IPGVG)_4]_{16}$ as the hydrophobic domain, the ELPs can self-assemble into nanoparticles above the critical micelle temperature and aggregate above the transition temperature of ELPs. They demonstrated that ELPs can have a high degree of morphology control of the self-assembled nanostructure, and temperature can be adjusted for the formation of nanoparticles or aggregates in dilute solutions. This opened up possibilities using ELPs for drug delivery applications. Dreher et al reported a diblock AB ELPs that can self-assemble into spherical micelles in clinically relevant hyperthermic temperatures[25]. They envisioned potential applications in targeted drug delivery by using the ELPs as multivalent

carriers of drug. Glassman et al designed tough ELP hydrogels for tissue scaffold with good mechanical properties[26]. Wirtz and coworkers demonstrated novel protease responsive diblock ELPs that phase separates upon cleavage in dilute solution[27].

Recent studies by Navarro et al. revealed that diblock copolypeptides consists of resilin-like and elastin-like polypeptides (RLP_n-b-ELP_{80}) can form microphase-separated structures in concentrated aqueous solutions. The length of the ELP is kept constant at 80 while varying the RLP length n from 20 to 100 in 20 increments. Using small-angle X-ray scattering (SAXS), Navarro and colleagues demonstrated the formation of ordered nanostructures, such as hexagonally packed cylinders and lamellae, at concentrations of 30–50 wt%. These structures were highly sensitive to the RLP fraction, block length, concentrations, and temperature, emphasizing the role of polymer-solvent interactions in driving phase transitions.

1.2.3 Protein–Polymer Bioconjugates

Globular protein–polymer bioconjugates (i.e. protein–polymer block copolymers) are hybrid macromolecules in which a folded globular protein is covalently linked to a synthetic polymer chain. They contain higher sequence complexity than synthetic block copolymers due to the sequence control in globular protein, leading to diverse functionality, and different self-assembly phase diagrams from synthetic block copolymers. A well-studied experimental model system is the red fluorescent protein mCherry conjugated with poly(*N*-isopropylacrylamide) (PNIPAM). This globular protein–polymer conjugate has been used to demonstrate how such hybrids can form ordered nanostructures. For example, when solutions of mCherry–PNIPAM are concentrated (or cast as films), the conjugates self-assemble into bulk "solid-state" nanostructures containing roughly 30–35% protein by weight[28, 29]. Notably, the specific structure can depend on processing conditions: solvent evaporation under different selectivities yielded

either disordered lamellae or perforated lamellae, and subsequent solvent annealing induced a transition to well-ordered lamellar layers with the globular proteins arranged in bilayer sheets. This mirrors the behavior of traditional block copolymers, which form lamellar, cylindrical, or network phases depending on the volume fractions and processing. Crucially, in the mCherry–PNIPAM system the protein's tertiary structure remained intact within the nanophase-separated domains – SAXS and circular dichroism showed no protein crystallization or secondary structure loss upon self-assembly[30]. Thus, proteins can be nanopatterned into solid materials without denaturing, opening opportunities to create biofunctional solid films and coatings.

Even though the self-assembly of globular protein–polymer bioconjugates is well-characterized experimentally as illustrated in Figure 1-4, models that can capture the phase behavior in those systems are currently lacking. There are a few technical challenges associated with modeling protein-polymer bioconjugates. First, proteins have a high degree of shape anisotropy, with difference in size and tertiary or quaternary structures. In recent years, data-driven models have transformed protein structure prediction – most prominently, DeepMind's AlphaFold2 demonstrated that a neural network can learn the mapping from amino acid sequence to folded 3D structure with remarkable accuracy[31]. However, extending those types of models to protein-polymer conjugates self-assembly is nontrivial. There lacks a good encoding format for representing the protein-polymer conjugates. Although experimental data are available in literature, there lacks a large, standardized dataset. Second, the self-assembly in protein-polymer conjugates spans length scale from atomic-level interactions to mesoscale structures, which can be difficult to capture in a single physics-based model. Finally, at higher concentrations, it becomes computationally infeasible to simulate those systems due to the number of macromolecules. It is

necessary to simplify the system through coarse graining while still capturing the important interactions for self-assembly in protein-polymer bioconjugates.

Empirical models have been developed to capture and predict the self-assembly behavior of those protein-polymer bioconjugates. Huang *et al.* analyzed a library of 15 protein–PNIPAM conjugates (comprising 11 distinct globular proteins tethered to PNIPAM) and looked for correlations between protein properties and self-assembly behavior[32]. Interestingly, despite the diverse nature of the proteins, many attributes of phase behavior were *similar across all conjugates* – for example, all showed disordered micellar phases at low concentration/high temperature and lamellar ordering at intermediate conditions. This suggested that a few key variables might govern the behavior. By performing a regression analysis, the authors found that the protein's molecular weight and its secondary structure content (particularly the fraction of residues in β-sheets) were strong predictors of the quality of ordering in the conjugate. In fact, they demonstrated a simple model where just those two protein features could quantitatively predict the SAXS peak sharpness (a measure of ordering) for the different conjugates. Proteins that were larger and had higher β-sheet fraction tended to promote better ordering, which makes intuitive sense: larger proteins provide a bigger immiscible domain, and β-rich proteins may be more rigid or uniformly structured, aiding periodic packing. This kind of statistical model provides a foundation for more advanced machine learning models – it shows that there *are* measurable protein features that correlate with self-assembly outcomes. Despite the wealth of knowledge on the self-assembly of these systems, a unifying theory describing the phase behavior of globular protein–polymer bioconjugates does not exist.

Another promising simulation strategy involves field-theoretic approaches for modeling mesoscale self-assembly. For protein–polymer conjugates, but progress from polymer nanocomposite modeling offers a potential solution. Fredrickson et al. introduced a hybrid particle–field (HPF) simulation method in which during the particle-to-field transformation, the positions of the nanoparticles are left as explicit degrees of freedom in the partition function.[16] *Riggleman et al.* have developed a unified field-theoretic framework to handle polymers containing spherical or anisotropic nanoparticles, going beyond simple mean-field treatments.[17] Their method incorporates the nanoparticles into the field theory, which allows correlations between particle positions and polymer structuring around them to be captured. This kind of fully fluctuating field simulation could, in principle, be adapted to a *single-tether* nanoparticle (the protein–polymer conjugate) to predict self-assembled phases. However, *Riggleman et al.* pointed out that their approach works better for soft nanoparticles and tends to have numerical instabilities for hard nanoparticles.[17] Recent advances in modeling block copolymer/superparamagnetic nanoparticle composites further demonstrate the versatility of field-theoretic approach. Sharma et al. showed that dipolar interactions under magnetic fields can induce morphological transitions, with nanoparticle chaining or lattice formation driving polymer domain alignment.[18] Complementarily, they introduced a dynamic hybrid of external potential dynamics and Brownian dynamics to capture orientational ordering kinetics, explicitly accounting for excluded volume interactions between nanoparticles and polymers.[19] However, key challenges arise at the particle–polymer interface and junction, which reduces their computational efficiency. The disparity between particle and protein length scales, the hardness of the protein, and modeling the protein-polymer junction[18] continue to be challenges that limit the applicability of these methods to protein-polymer conjugates.

A logical starting point is to coarse-grain the protein–polymer conjugate into a simpler particle-based representation to understand the molecular basis and driving force for self-assembly. In coarse-grained simulations, groups of atoms are lumped into abstract "beads" or effective interaction sites, sacrificing chemical detail for computational efficiency. For globular protein–polymer bioconjugates, a common approach is to represent the protein as a single (or a few) large bead(s) and the polymer as a string of smaller beads (a bead-spring chain). This essentially treats each bioconjugate as a colloidal particle with a polymer tail**,** reminiscent of a polymer-grafted nanoparticle.

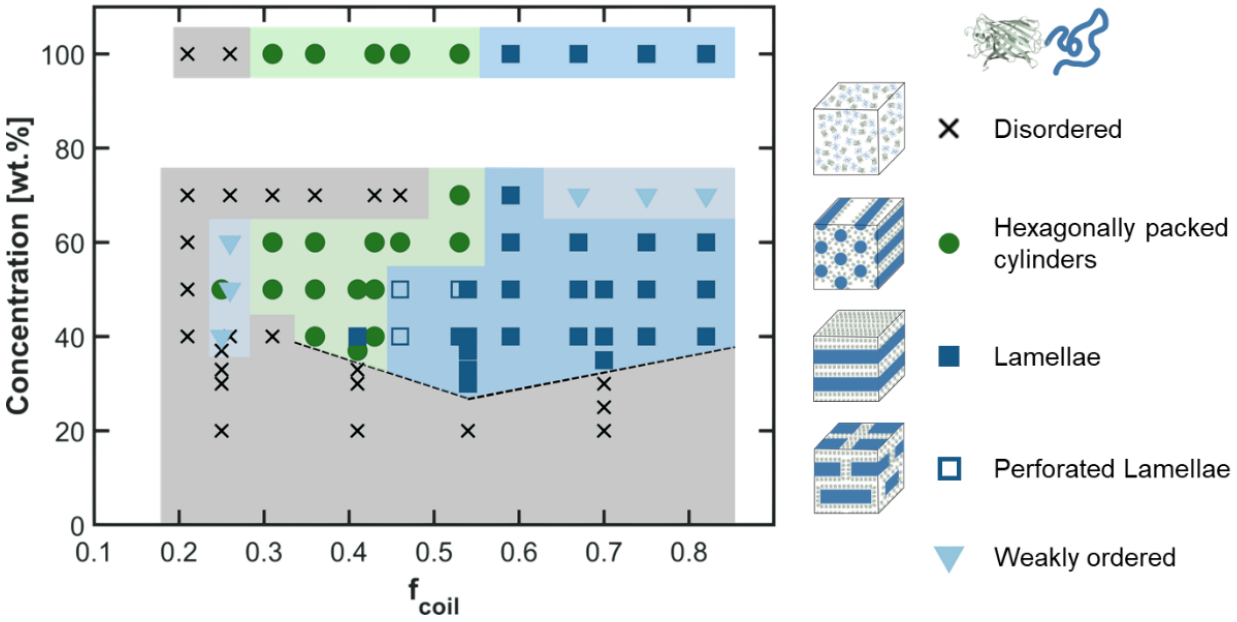


Figure 1-4. Experimental phase diagram for solution-state mCherry-*b*-PNIPAM bioconjugates (shown schematically in inset) at 25 °C reproduced from Yao[37].

1.3 Sequence-Defined Polymers

Given the vast sequence space, computational tools are extremely helpful for screening the sequence of interest and simulating their properties. A few approaches have been developed for sequence defined polymers. Xie et al extended the classical self-consistent field theory to

sequence-defined polymers[38]. Its computational cost is a factor N_s, where N_s is number of points used to discretize the polymer contour length) more expensive than the classical SCFT formalism. This approach focuses specifically on predicting the self-assembled morphologies. Webb et al developed a machine learning model from a coarse-grained polymer genome consists of four types of beads and three topologies for targeting radius of gyration[39, 40]. They were able to find polymer sequences that exhibit globular, swollen, or rod-like behaviors, which are verified by explicit simulations. Reinhart and Statts developed a large language model (LLMs)[41], Anthropic's Claude 3.5 that can be used as effective evolutionary optimizers for designing sequence-defined macromolecules. Without any fine-tuning or training data, the LLM iteratively proposes polymer sequences that assemble into target morphologies, guided only by a surrogate model trained on molecular dynamics (MD) data. Compared to traditional methods like active learning and genetic algorithms, the LLM consistently converges faster and finds better solutions across a range of morphologies. As evidenced by those literature results, computational tools are available for studying sequence-defined polymers; however, the challenge to synthesize truly sequence defined polymers and the lack of high-quality experimental dataset that are applicable for those models are slowing down the development for new polymer materials experimentally.

To create sequence-defined polymers that match the sequence complexity of biological polymers, polymer chemists have tried different polymerization techniques to increase the control over primary sequence, such as single monomer addition in reversible addition–fragmentation chain transfer (RAFT), atom-transfer radical polymerization, step growth polymerization, iterative synthesis, etc [11]. However, most methods cannot achieve sequence-defined polymers with chain monodispersity. Iterative synthesis can achieve sequence-defined polymers with ~50 monomers

but is expensive, time consuming, and not scalable[11], which makes it unsuitable for most challenges in materials development.

Another approach is to use sequence-defined biological polymerization processes: DNA replication, transcription, and translation. There are many biological polymerization methods at different levels of complexity. The simplest method uses non-enzymatic template directed synthesis of nucleic acid *in vitro*[42]. This non-enzymatic polymerization method opens up potential for incorporating nonnatural nucleic acids, but is limited by lower polymerization yield and lower numbers of consecutive coupled nucleic acids[42]. The higher complexity methods involve either polymerases to synthesize nucleic acid *in vitro*[43], or bacteria to synthesize amino acids *in vivo*[44]. The enzymatic polymerization *in vitro* uses polymerase to copy and amplify DNA strands. An example of this technique is polymerase chain reaction (PCR), which uses Taq polymerase derived from thermophiles to catalyze the extension of DNA. The *in vivo* method uses the full cellular machinery of bacteria to produce recombinant protein. An artificial gene which encodes the desired protein is transformed into the cell and overexpressed inside the host bacterium. The cells are lysed, and the proteins are purified. The advantages of this approach are robustness, reasonable yield, and the ability to scale up, which makes it a good approach for studying and engineering sequence-defined polymers[44].

For protein-based sequence-defined polymers, consensus-sequence protein engineering clarifies sequence-property relationships that evolution often obscures[44]**.** By aligning large families of homologous proteins and tallying residue frequencies at every position, researchers identify *conserved* residues that dominate a given site and lock them into a “consensus” scaffold, while treating semi-conserved or non-conserved sites as programmable handles. This statistics-driven process—routinely executed with UniProt or OWL libraries—yields modular

"consensus repeat" units whose sequence variation is stripped to a well-defined core, enabling systematic perturbation. Because the resulting polymers draw on billions of years of evolutionary sampling yet are simplified to monomer-level precision, consensus design has become a powerful bridge between protein engineering and polymer science. Chilkoti et al. have designed consensus sequences using intrinsically disordered protein polymers and found that by tuning the consensus sequence in the primary polypeptide chain, those protein polymers can exhibit either a lower critical solution temperature (LCST) or an upper critical solution temperature (UCST)[45].

Many recombinant protein materials are now designed by minimal "consensus" repeat that nature already encodes for a given structural motif, then varying the few mutable positions in that repeat to dial in function[44, 46]. For example, collagen-mimetic triple helices rely on the tripeptide consensus repeats of GXO, where X is a guest residue[47]**.** Swapping residues at X tunes helix stability, melting point, and mineral affinity, allowing bespoke fibrous scaffolds. Resilin like polypeptides, originated from the insects' wing hinge, contain 16-residue repeats of GGRPSDSYGAPGGGN or GYSGGRPGGQDLG, whose β-turn-rich sequences yield highly hydrated networks[48]. Elastin like polypeptides contains XGGXG or XPGXG consensus repeats and can form β-turns upon increasing temperature above the transition temperature[49]. The guest residues allow for tuning their phase behavior using hydrophobicity and charge. Reflectin contains motif M/FD$(X)_5$MD$(X)_5$MD$(X)_{3/4}$, where X can be M, R, G, D, S, or Q, which allows light interacting properties[50]. The charged-and-aromatic distribution lets the chain expand or contract with phosphorylation or salt, forming tunable Bragg stacks that modulate visible and near-IR light. Nucleoporin-like proteins contain FG (Phe–Gly) repeats and their variants (FxFG, GLFG, PSFG) form intrinsically disordered, low-complexity domains that generate the selective permeability

barrier of the nuclear pore complex[51-53]. The engineered hydrogel can selectively enhance transport of cargo–carrier complexes similar to the natural nuclear pore system[51].

Consensus-engineered protein repeats offer an unparalleled platform for studying sequence-defined polymers because they pair monomer-precision sequencing with the hierarchical folding and self-assembly absent in most synthetic analogues. By locking conserved residues into minimal consensus motifs and varying only the mutable positions, researchers can build libraries that change a single residue, repeat number, or block architecture while keeping all other variables fixed, directly correlating sequence edits with macroscopic properties such as LCST/UCST transitions, hydrogel stiffness, and selective transport. The quantitative design rules extracted from these systems—e.g., linear hydrophobicity control of ELP phase behavior or silk/ELP block-ratio tuning of mechanical strength—translate seamlessly to coarse-grained simulations and machine-learning models used for fully synthetic chains. As a result, consensus repeats function as living laboratories whose experimentally validated sequence–property maps inform the theory, computation, and synthesis of the next generation of precisely coded materials.

The role of sequence in governing the phase behavior of polymers remains underexplored. Recent studies by Mills *et al.* have revealed that a hydrophobic periodic ELP sequence, similar to many thermoresponsive synthetic polymers[54], also exhibits cononsolvency behavior—a decrease in solubility when mixed in solvent blends. ELP undergoes LCST behavior at low ethanol concentrations but transitions to upper critical solution temperature (UCST) behavior at higher ethanol content[55]. This cononsolvency effect arises from subtle competition between water, cosolvent, and polymer interactions. Mills *et al.* argued that the primary driving forces resemble those in poly(N-isopropylacrylamide) (PNIPAM), yet the dominant hydrogen-bonding sites differ: ELPs form hydrogen bonds along the peptide backbone while alcohol preferentially solvates their

hydrophobic side chains, whereas PNIPAM forms hydrogen bonds via its amide side chains and alcohol instead solvates the backbone.

1.4 Small Angle Neutron Scattering

Neutron scattering is a useful technique to study the static and dynamic structures in a variety of materials such as polymers[56], colloids[57], biomacromolecules[58], metals[59], glasses[60], and ceramics[61]. Among many neutron scattering techniques, small angle neutron scattering (SANS) is one of the most widely used in probing the microstructure of polymer and soft materials[62]. SANS has been successfully used to characterize the structural formation in polymer solutions, melts, and networks and has advanced the understanding of fundamental polymer physics[63, 64]. The use of neutrons provide several advantages over other radiation sources, such as X-ray or light, because neutrons are nondestructive to soft matter[65] and allow contrast variation using isotopic labeling[66]. However, neutron count rates are inherently lower because neutrons do not interact very strongly with many materials, and neutron sources generally have lower flux than other radiation sources. These effects make SANS experiments slower. Typically, SANS data is acquired for a pre-determined number of counts above the background. After data acquisition and reduction, models are fit to experimental data to extract relevant structural parameters or identification of structures based on the goodness of fit as illustrated in Figure 1-5. The measurement time to obtain SANS data for each sample can range from ten minutes to tens of hours depending on the facility, the scattering power of the sample, and instrument configuration[67]. Because neutron sources are expensive to construct and maintain, neutron scattering is only available at a handful of facilities worldwide, limiting access to this characterization technique.

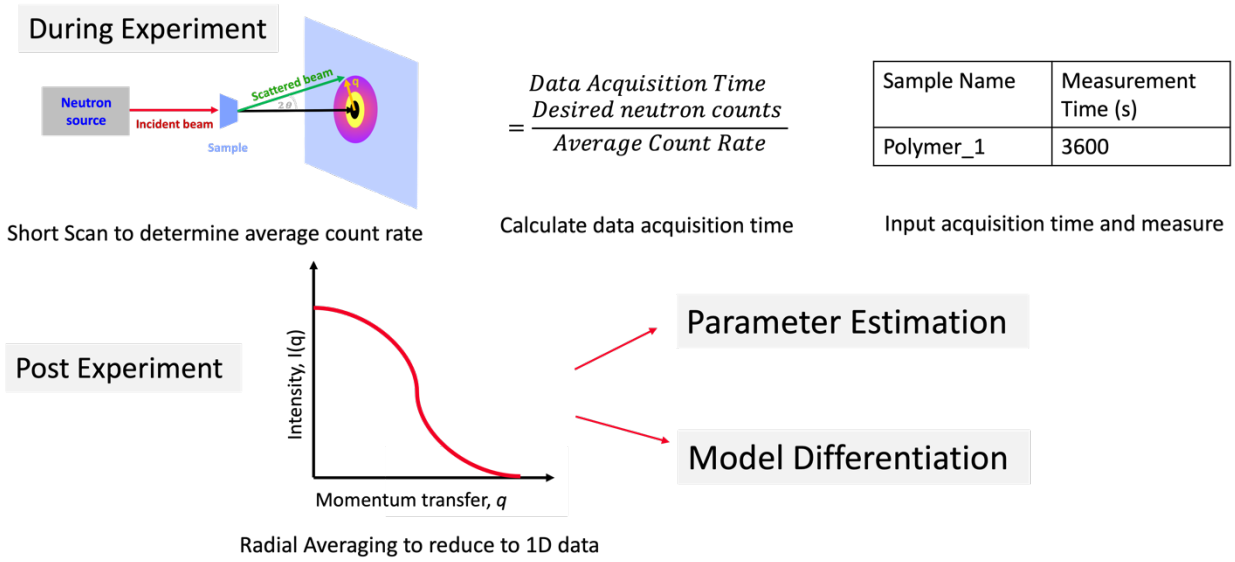


Figure 1-5. Illustrative figure of classic SANS workflow.

With the surge of machine learning[68, 69] and data driven research in material discovery[70], the need to generate databases of materials with experimental data has led to many automated high throughput material syntheses in daily experimental workflow[71-73]. With the importance of neutron scattering in studying the microstructure and dynamics of many materials, many large facilities have been improving the hardware of the instruments, upgrading the software at the beamline[74], making the data reduction process automated, and building higher flux neutron facilities[75] to increase the accessibility of neutrons to users in the world.

There have been increasing efforts in literature to optimize the use of beam time through experimental design, statistical analysis, and machine learning approaches. Steinhart and Pleštil proposed an algorithm to balance the tradeoff between measurement time and resolution by using the scattering intensity data from background measurement to optimize the measurement time of the sample at each angle for small angle X-ray experiments[76]. Saito and colleagues applied kernel density estimation in reducing measurement time for anisotropic samples, achieving a significant reduction without fully addressing the smearing effect[77]. Kanazawa et al. proposed accelerated

small-angle scattering experiments with simulation-based machine learning[78]. They created a database of virtual experiments by simulation and use those data adaptively during real experiments to aid the selection of optimal wavevector (q) ranges sequentially to achieve the best sampling at each measurement. Another attempt to accelerate the data acquisition by Chang and coworkers involved increasing the size of binning of the detector pixels at the sacrifice of resolution and reconstructing the high-resolution image using a deep-learning based super resolution technique trained by data on EQ-SANS at Oakridge National Lab[67]. Their work shows that the reconstructed 2D scattering image is comparable in resolution to the true experimental data and effective at saving beamtime if implemented online during SANS acquisition. However, there remains an opportunity to critically analyze the question of how many counts are required for data acquisition based on the type of information being obtained from the scattering experiment.

1.5 Overview of Thesis

This thesis explores how polymer primary sequence governs phase behavior and self-assembly across the spectrum of sequence-defined polymers—from recombinant polypeptides to protein–polymer bioconjugates while developing the experimental and computational tools needed to interrogate vast design spaces. Chapter 3 introduces an information-driven workflow for small-angle neutron scattering (SANS) to screen through large number of sequence-defined polymers. By terminating data collection once user-defined uncertainty bounds are met, the method enables high-throughput mapping of sequence libraries. Chapter 4 designed a pair of ABA triblock elastin-like polypeptides (ELPs) that differ only in the hydrophobicity of the mid-block. The study combines SANS, depolarized light scattering, and rheology to reveal how a single patterning change drives micelle formation, water partitioning, and rheological transitions near the LCST—linking monomer-scale hydrophobicity to mesoscale architecture. Chapter 5 widens the

scope to a ELP library with conserved length but systematically varied guest residues. Turbidimetry measurements in water/ethanol/NaCl blends show how hydrophobicity, charge, and compete to set cononsolvency boundaries. Chapter 6 develops a highly coarse-grained dumbbell model for globular protein–polymer bioconjugates. Molecular-dynamics simulations—validated against experimental phase diagrams—show that treating the protein as a rigid "hard-sphere block" and the polymer as a soft tail captures the self-assembly pathways inaccessible to mean-field theories, providing a scalable framework for understanding the driving force of self-assembly in high degree of sequence complexity. Together, the thesis advances both the fundamental understanding and the practical tools needed to engineer the next generation of precisely coded soft materials.

1.6 References

1. DeStefano, A. J.; Mengel, S. D.; Bates, M. W.; Jiao, S.; Shell, M. S.; Han, S.; Segalman, R. A., Control over Conformational Landscapes of Polypeptoids by Monomer Sequence Patterning. *Macromolecules* **2024,** *57* (4), 1469-1477.
2. Jin, T.; Coley, C. W.; Alexander-Katz, A., Sequence-Sensitivity in Functional Synthetic Polymer Properties. *Angewandte Chemie International Edition* **2025,** *64* (2), e202415047.
3. O'Toole, E. M.; Panagiotopoulos, A. Z., Effect of sequence and intermolecular interactions on the number and nature of low-energy states for simple model proteins. *The Journal of Chemical Physics* **1993,** *98* (4), 3185-3190.
4. Hartmann, L.; Börner, H. G., Precision Polymers: Monodisperse, Monomer-Sequence-Defined Segments to Target Future Demands of Polymers in Medicine. *Advanced Materials* **2009,** *21* (32-33), 3425-3431.
5. Perry, S. L.; Sing, C. E., 100th Anniversary of Macromolecular Science Viewpoint: Opportunities in the Physics of Sequence-Defined Polymers. *ACS Macro Letters* **2020,** *9* (2), 216-225.
6. Pääbo, S.; Gifford, J. A.; Wilson, A. C., Mitochondrial DNA sequences from a 7000-year old brain. *Nucleic Acids Research* **1988,** *16* (20), 9775-9787.
7. Bhattacharyya, R. P.; Reményi, A.; Yeh, B. J.; Lim, W. A., Domains, Motifs, and Scaffolds: The Role of Modular Interactions in the Evolution and Wiring of Cell Signaling Circuits. *Annual Review of Biochemistry* **2006,** *75* (Volume 75, 2006), 655-680.
8. Meier, M. A. R.; Barner-Kowollik, C., A New Class of Materials: Sequence-Defined Macromolecules and Their Emerging Applications. *Advanced Materials* **2019,** *31* (26), 1806027.
9. Buehler, M. J.; and Ackbarow, T., Nanomechanical strength mechanisms of hierarchical biological materials and tissues. *Computer Methods in Biomechanics and Biomedical Engineering* **2008,** *11* (6), 595-607.
10. DeStefano, A. J.; Segalman, R. A.; Davidson, E. C., Where Biology and Traditional Polymers Meet: The Potential of Associating Sequence-Defined Polymers for Materials Science. *JACS Au* **2021**.
11. Lutz, J.-F.; Ouchi, M.; Liu, D. R.; Sawamoto, M., Sequence-Controlled Polymers. *Science* **2013,** *341* (6146), 1238149.
12. Rosales, A. M.; Segalman, R. A.; Zuckermann, R. N., Polypeptoids: a model system to study the effect of monomer sequence on polymer properties and self-assembly. *Soft Matter* **2013,** *9* (35), 8400-8414.
13. Leibler, L., Theory of microphase separation in block copolymers. *Macromolecules* **1980,** *13* (6), 1602-1617.
14. Hamley, I. W., *The physics of block copolymers*. Oxford University Press: 1998.
15. Bates, F. S.; Fredrickson, G. H., Block Copolymers—Designer Soft Materials. *Physics Today* **1999,** *52* (2), 32-38.
16. Bates, C. M.; Bates, F. S., 50th Anniversary Perspective: Block Polymers—Pure Potential. *Macromolecules* **2017,** *50* (1), 3-22.
17. Aissou, K.; Nunns, A.; Manners, I.; Ross, C. A., Square and Rectangular Symmetry Tiles from Bulk and Thin Film 3-Miktoarm Star Terpolymers. *Small* **2013,** *9* (23), 4077-4084.
18. Matsen, M. W., Fast and accurate SCFT calculations for periodic block-copolymer morphologies using the spectral method with Anderson mixing. *The European Physical Journal E* **2009,** *30* (4), 361.

19. Hamley, I. W., *Block copolymers in solution: fundamentals and applications*. John Wiley & Sons: 2005.
20. Hanley, K. J.; Lodge, T. P.; Huang, C.-I., Phase Behavior of a Block Copolymer in Solvents of Varying Selectivity. *Macromolecules* **2000,** *33* (16), 5918-5931.
21. Zhang, W.; Shi, L.; Wu, K.; An, Y., Thermoresponsive micellization of poly (ethylene glycol)-b-poly (N-isopropylacrylamide) in water. *Macromolecules* **2005,** *38* (13), 5743-5747.
22. Liu, Y.; Wang, W.; Yang, J.; Zhou, C.; Sun, J., pH-sensitive polymeric micelles triggered drug release for extracellular and intracellular drug targeting delivery. *Asian Journal of Pharmaceutical Sciences* **2013,** *8* (3), 159-167.
23. Urry, D. W.; Gowda, D. C.; Parker, T. M.; Luan, C.-H.; Reid, M. C.; Harris, C. M.; Pattanaik, A.; Harris, R. D., Hydrophobicity scale for proteins based on inverse temperature transitions. *Biopolymers* **1992,** *32* (9), 1243-1250.
24. Wright, E. R.; Conticello, V. P., Self-assembly of block copolymers derived from elastin-mimetic polypeptide sequences. *Advanced Drug Delivery Reviews* **2002,** *54* (8), 1057-1073.
25. Dreher, M. R.; Simnick, A. J.; Fischer, K.; Smith, R. J.; Patel, A.; Schmidt, M.; Chilkoti, A., Temperature Triggered Self-Assembly of Polypeptides into Multivalent Spherical Micelles. *Journal of the American Chemical Society* **2008,** *130* (2), 687-694.
26. Glassman, M. J.; Avery, R. K.; Khademhosseini, A.; Olsen, B. D., Toughening of Thermoresponsive Arrested Networks of Elastin-Like Polypeptides To Engineer Cytocompatible Tissue Scaffolds. *Biomacromolecules* **2016,** *17* (2), 415-426.
27. Wirtz, B. M.; Yun, A. G.; Wick, C.; Gao, X. J.; Mai, D. J., Protease-Driven Phase Separation of Elastin-Like Polypeptides. *Biomacromolecules* **2024,** *25* (8), 4898-4904.
28. Thomas, C. S.; Olsen, B. D., Coil fraction-dependent phase behaviour of a model globular protein–polymer diblock copolymer. *Soft Matter* **2014,** *10* (17), 3093-3102.
29. Lam, C. N.; Olsen, B. D., Phase transitions in concentrated solution self-assembly of globular protein–polymer block copolymers. *Soft Matter* **2013,** *9* (8), 2393-2402.
30. Lam, C. N.; Kim, M.; Thomas, C. S.; Chang, D.; Sanoja, G. E.; Okwara, C. U.; Olsen, B. D., The Nature of Protein Interactions Governing Globular Protein–Polymer Block Copolymer Self-Assembly. *Biomacromolecules* **2014,** *15* (4), 1248-1258.
31. Jumper, J.; Evans, R.; Pritzel, A.; Green, T.; Figurnov, M.; Ronneberger, O.; Tunyasuvunakool, K.; Bates, R.; Žídek, A.; Potapenko, A., Highly accurate protein structure prediction with AlphaFold. *nature* **2021,** *596* (7873), 583-589.
32. Huang, A.; Paloni, J. M.; Wang, A.; Obermeyer, A. C.; Sureka, H. V.; Yao, H.; Olsen, B. D., Predicting Protein–Polymer Block Copolymer Self-Assembly from Protein Properties. *Biomacromolecules* **2019,** *20* (10), 3713-3723.
33. Koski, J.; Chao, H.; Riggleman, R. A., Field theoretic simulations of polymer nanocomposites. *The Journal of chemical physics* **2013,** *139* (24).
34. Sides, S. W.; Kim, B. J.; Kramer, E. J.; Fredrickson, G. H., Hybrid Particle-Field Simulations of Polymer Nanocomposites. *Physical Review Letters* **2006,** *96* (25), 250601.
35. Raman, V.; Sharma, R.; Hatton, T. A.; Olsen, B. D., Magnetic field induced morphological transitions in block copolymer/superparamagnetic nanoparticle composites. *ACS Macro Letters* **2013,** *2* (8), 655-659.
36. Raman, V.; Hatton, T. A.; Olsen, B. D., Kinetics of Magnetic Field-Induced Orientational Ordering in Block Copolymer/Superparamagnetic Nanoparticle Composites. *Macromolecular rapid communications* **2014,** *35* (23), 2005-2011.

37. Yao, H. Driving forces of self-assembly in protein-polymer bioconjugates. Massachusetts Institute of Technology, 2020.
38. Xie, O.; Olsen, B. D., A Self-Consistent Field Theory Formalism for Sequence-Defined Polymers. *Macromolecules* **2022,** *55* (15), 6516-6524.
39. Patel, R. A.; Borca, C. H.; Webb, M. A., Featurization strategies for polymer sequence or composition design by machine learning. *Molecular Systems Design & Engineering* **2022,** *7* (6), 661-676.
40. Webb, M. A.; Jackson, N. E.; Gil, P. S.; de Pablo, J. J., Targeted sequence design within the coarse-grained polymer genome. *Science advances* **2020,** *6* (43), eabc6216.
41. Reinhart, W. F.; Statt, A., Large language models design sequence-defined macromolecules via evolutionary optimization. *npj Computational Materials* **2024,** *10* (1), 262.
42. Orgel, L. E., Molecular replication. *Nature* **1992,** *358* (6383), 203-209.
43. Niu, J.; Hili, R.; Liu, D. R., Enzyme-free translation of DNA into sequence-defined synthetic polymers structurally unrelated to nucleic acids. *Nature chemistry* **2013,** *5* (4), 282-292.
44. Chang, M. P.; Huang, W.; Mai, D. J., Monomer-scale design of functional protein polymers using consensus repeat sequences. *Journal of Polymer Science* **2021**.
45. Quiroz, F. G.; Chilkoti, A., Sequence heuristics to encode phase behaviour in intrinsically disordered protein polymers. *Nature Materials* **2015,** *14* (11), 1164-1171.
46. Yang, Y. J.; Holmberg, A. L.; Olsen, B. D., Artificially Engineered Protein Polymers. *Annu Rev Chem Biomol Eng* **2017,** *8*, 549-575.
47. Brodsky, B.; Ramshaw, J. A., The collagen triple-helix structure. *Matrix Biol* **1997,** *15* (8-9), 545-54.
48. Ardell, D. H.; Andersen, S. O., Tentative identification of a resilin gene in Drosophila melanogaster. *Insect Biochem Mol Biol* **2001,** *31* (10), 965-70.
49. Debelle, L.; Tamburro, A. M., Elastin: molecular description and function. *Int J Biochem Cell Biol* **1999,** *31* (2), 261-72.
50. Crookes, W. J.; Ding, L.-L.; Huang, Q. L.; Kimbell, J. R.; Horwitz, J.; McFall-Ngai, M. J., Reflectins: The Unusual Proteins of Squid Reflective Tissues. *Science* **2004,** *303* (5655), 235-238.
51. Kim, M.; Chen, W. G.; Kang, J. W.; Glassman, M. J.; Ribbeck, K.; Olsen, B. D., Artificially engineered protein hydrogels adapted from the nucleoporin Nsp1 for selective biomolecular transport. *Advanced Materials* **2015,** *27* (28), 4207-4212.
52. Yang, Y. J.; Mai, D. J.; Dursch, T. J.; Olsen, B. D., Nucleopore-inspired polymer hydrogels for selective biomolecular transport. *Biomacromolecules* **2018,** *19* (10), 3905-3916.
53. Yang, Y. J.; Mai, D. J.; Li, S.; Morris, M. A.; Olsen, B. D., Tuning selective transport of biomolecules through site-mutated nucleoporin-like protein (nlp) hydrogels. *Biomacromolecules* **2021,** *22* (2), 289-298.
54. Schild, H. G.; Muthukumar, M.; Tirrell, D. A., Cononsolvency in mixed aqueous solutions of poly(N-isopropylacrylamide). *Macromolecules* **1991,** *24* (4), 948-952.
55. Mills, C. E.; Ding, E.; Olsen, B. D., Cononsolvency of elastin-like polypeptides in water/alcohol solutions. *Biomacromolecules* **2019,** *20* (6), 2167-2173.
56. Roe, R.-J., *Methods of X-ray and neutron scattering in polymer science*. Oxford University Press on Demand: 2000.
57. Helgeson, M. E.; Moran, S. E.; An, H. Z.; Doyle, P. S., Mesoporous organohydrogels from thermogelling photocrosslinkable nanoemulsions. *Nature materials* **2012,** *11* (4), 344-352.

58. Yearley, Eric J.; Zarraga, Isidro E.; Shire, Steven J.; Scherer, Thomas M.; Gokarn, Y.; Wagner, Norman J.; Liu, Y., Small-Angle Neutron Scattering Characterization of Monoclonal Antibody Conformations and Interactions at High Concentrations. *Biophysical Journal* **2013,** *105* (3), 720-731.
59. Baym, G., Direct Calculation of Electronic Properties of Metals from Neutron Scattering Data. *Physical Review* **1964,** *135* (6A), A1691-A1692.
60. Wright, A. C.; Clare, A. G.; Grimley, D. I.; Sinclair, R. N., Neutron scattering studies of network glasses. *Journal of Non-Crystalline Solids* **1989,** *112* (1-3), 33-47.
61. Allen, A. J., Characterization of ceramics by x-ray and neutron small-angle scattering. *Journal of the American Ceramic Society* **2005,** *88* (6), 1367-1381.
62. Melnichenko, Y. B.; Wignall, G. D., Small-angle neutron scattering in materials science: Recent practical applications. *Journal of Applied Physics* **2007,** *102* (2), 3.
63. De Gennes, P.-G.; Gennes, P.-G., *Scaling concepts in polymer physics*. Cornell university press: 1979.
64. Doi, M.; Edwards, S. F.; Edwards, S. F., *The theory of polymer dynamics*. oxford university press: 1988; Vol. 73.
65. Jeffries, C. M.; Ilavsky, J.; Martel, A.; Hinrichs, S.; Meyer, A.; Pedersen, J. S.; Sokolova, A. V.; Svergun, D. I., Small-angle X-ray and neutron scattering. *Nature Reviews Methods Primers* **2021,** *1* (1), 1-39.
66. Li, L.; Jakowski, J.; Do, C.; Hong, K., Deuteration and Polymers: Rich History with Great Potential. *Macromolecules* **2021,** *54* (8), 3555-3584.
67. Chang, M.-C.; Wei, Y.; Chen, W.-R.; Do, C., Deep learning-based super-resolution for small-angle neutron scattering data: attempt to accelerate experimental workflow. *MRS Communications* **2020,** *10* (1), 11-17.
68. Raccuglia, P.; Elbert, K. C.; Adler, P. D.; Falk, C.; Wenny, M. B.; Mollo, A.; Zeller, M.; Friedler, S. A.; Schrier, J.; Norquist, A. J., Machine-learning-assisted materials discovery using failed experiments. *Nature* **2016,** *533* (7601), 73-76.
69. Chen, Z.; Andrejevic, N.; Drucker, N. C.; Nguyen, T.; Xian, R. P.; Smidt, T.; Wang, Y.; Ernstorfer, R.; Tennant, D. A.; Chan, M.; Li, M., Machine learning on neutron and x-ray scattering and spectroscopies. *Chemical Physics Reviews* **2021,** *2* (3), 031301.
70. Wang, Z.; Sun, Z.; Yin, H.; Liu, X.; Wang, J.; Zhao, H.; Pang, C. H.; Wu, T.; Li, S.; Yin, Z.; Yu, X.-F., Data-Driven Materials Innovation and Applications. *Advanced Materials* **2022,** *34* (36), 2104113.
71. Hoogenboom, R.; Meier, M. A.; Schubert, U. S., Combinatorial methods, automated synthesis and high-throughput screening in polymer research: past and present. *Macromolecular rapid communications* **2003,** *24* (1), 15-32.
72. Trobe, M.; Burke, M. D., The molecular industrial revolution: automated synthesis of small molecules. *Angewandte Chemie International Edition* **2018,** *57* (16), 4192-4214.
73. Zhang, C.; Bates, M. W.; Geng, Z.; Levi, A. E.; Vigil, D.; Barbon, S. M.; Loman, T.; Delaney, K. T.; Fredrickson, G. H.; Bates, C. M.; Whittaker, A. K.; Hawker, C. J., Rapid Generation of Block Copolymer Libraries Using Automated Chromatographic Separation. *Journal of the American Chemical Society* **2020,** *142* (21), 9843-9849.
74. Heller, W. T.; Hetrick, J.; Bilheux, J.; Calvo, J. M. B.; Chen, W.-R.; DeBeer-Schmitt, L.; Do, C.; Doucet, M.; Fitzsimmons, M. R.; Godoy, W. F.; Granroth, G. E.; Hahn, S.; He, L.; Islam, F.; Lin, J.; Littrell, K. C.; McDonnell, M.; McGaha, J.; Peterson, P. F.; Pingali, S. V.; Qian, S.; Savici, A. T.; Shang, Y.; Stanley, C. B.; Urban, V. S.; Whitfield, R. E.; Zhang, C.;

Zhou, W.; Billings, J. J.; Cuneo, M. J.; Leal, R. M. F.; Wang, T.; Wu, B., drtsans: The data reduction toolkit for small-angle neutron scattering at Oak Ridge National Laboratory. *SoftwareX* **2022,** *19*, 101101.
75. ShuoQian; Heller, W.; Chen, W.-R.; Christianson, A.; Do, C.; Wang, Y.; Lin, J. Y. Y.; Huegle, T.; Jiang, C.; Boone, C.; Hart, C.; Graves, V., CENTAUR—The small- and wide-angle neutron scattering diffractometer/spectrometer for the Second Target Station of the Spallation Neutron Source. *Review of Scientific Instruments* **2022,** *93* (7), 075104.
76. Steinhart, M.; Pleštic, J., Possible improvements in the precision and accuracy of small-angle X-ray scattering measurements. *Journal of applied crystallography* **1993,** *26* (4), 591-601.
77. Saito, K.; Yano, M.; Hino, H.; Shoji, T.; Asahara, A.; Morita, H.; Mitsumata, C.; Kohlbrecher, J.; Ono, K., Accelerating small-angle scattering experiments on anisotropic samples using kernel density estimation. *Sci Rep* **2019,** *9* (1), 1526.
78. Kanazawa, T.; Asahara, A.; Morita, H., Accelerating small-angle scattering experiments with simulation-based machine learning. *Journal of Physics: Materials* **2020,** *3* (1), 015001.

Chapter 2 Materials and Methods

2.1 Design and Synthesis of Elastin Like Polypeptides (ELPs)

Two ELPs were designed for Chapter 4: SI75 with primary sequence $(VPGSG)_{30}(VPGIG)_{25}(VPGSG)_{45}$ and SN75 with primary sequence $(VPGSG)_{30}(VPGNG)_{25}(VPGSG)_{45.}$ The genes were ordered in vector pET28a(+) and cloned into vector pQE-9.

SI75

Gene:

ATGAGAGGATCGCATCACCATCACCATCACGGATCCGAAAACTTGTACTTTCAAGGC
GTACCAGGAAGTGGAGTTCCCGGGTCAGGTGTGCCGGGTAGCGGTGTTCCAGGTTCT
GGTGTGCCGGGTAGCGGTGTGCCGGGTAGCGGTGTGCCGGGTTCAGGCGTTCCTGGT
AGCGGCGTACCGGGTAGCGGTGTGCCGGGCAGCGGTGTGCCGGGTTCCGGTGTCCC
GGGCTCGGGCGTGCCGGGATCTGGGGTTCCGGGTTCCGGTGTTCCCGGCAGCGGCGT
GCCGGGCTCCGGAGTCCCGGGTTCTGGCGTACCGGGCTCCGGTGTCCCGGGCAGCG
GTGTCCCAGGCAGCGGTGTTCCAGGTTCCGGTGTTCCTGGCTCCGGTGTTCCGGGTT
CCGGCGTCCCAGGCAGCGGTGTTCCAGGTAGCGGCGTGCCGGGTAGCGGCGTCCCG
GGCAGCGGTGTTCCGGGCAGCGGCGTTCCGGGTAGCGGTGTTCCGGGTTCGGGCGT
ACCGGGTATTGGCGTTCCGGGTATCGGCGTGCCGGGTATCGGCGTTCCGGGGATCGG
TGTTCCGGGTATTGGTGTTCCGGGTATTGGCGTGCCGGGTATTGGTGTCCCAGGCAT
CGGTGTGCCGGGTATTGGCGTGCCGGGGATCGGCGTCCCGGGCATTGGCGTGCCGG
GCATCGGCGTGCCGGGTATTGGTGTTCCGGGGATCGGCGTGCCGGGCATCGGCGTGC
CGGGGATCGGTGTTCCGGGCATTGGCGTGCCCGGCATCGGCGTGCCGGGCATCGGT

GTCCCGGGGATCGGCGTGCCGGGCATTGGCGTTCCGGGCATCGGCGTTCCGGGTATT

GGCGTTCCAGGCATTGGCGTGCCGGGCATCGGCGTTCCGGGTTCTGGTGTTCCGGGG

AGTGGTGTGCCGGGCAGCGGTGTTCCGGGTTCCGGTGTACCGGGCAGCGGTGTCCCG

GGCAGCGGTGTTCCAGGCTCTGGCGTGCCAGGTAGCGGTGTTCCGGGTTCTGGCGTT

CCGGGGAGCGGTGTACCGGGTAGCGGCGTGCCGGGCTCGGGTGTTCCGGGTAGCGG

TGTTCCGGGCTCAGGCGTGCCGGGCTCCGGCGTTCCGGGTAGCGGGGTTCCGGGTTC

TGGCGTGCCGGGCTCTGGTGTGCCAGGTAGCGGCGTGCCGGGTTCAGGCGTGCCGG

GTTCCGGCGTCCCTGGCTCGGGCGTGCCGGGTTCCGGCGTGCCGGGCAGCGGTGTGC

CGGGTAGCGGTGTTCCGGGCAGCGGTGTGCCGGGTTCTGGTGTGCCAGGTTCCGGTG

TGCCGGGTTCTGGTGTGCCGGGTAGCGGTGT

Amino Acid:

MRGSHHHHHHGSENLYFQGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGS

GVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPG

SGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGIGVP

GIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPG

VPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGIGVPGSGVPG

SGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVP

GSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGV

PGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSG

VPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSG*

SN75

Gene:

ATGAGAGGATCGCATCACCATCACCATCACGGATCCGAAAACTTGTACTTTCAAGGC

GTACCCGGAAGTGGAGTTCCAGGTTCAGGGGTTCCGGGTTCCGGCGTGCCTGGTTCC

GGCGTTCCGGGTAGCGGTGTCCCGGGTTCTGGCGTGCCGGGTAGCGGTGTTCCGGGT

TCCGGTGTGCCAGGCTCCGGTGTTCCGGGTAGTGGTGTGCCGGGTAGCGGTGTCCCG

GGTAGCGGCGTTCCGGGCAGCGGCGTCCCGGGTAGCGGTGTGCCAGGCTCTGGCGT

GCCGGGTAGCGGCGTGCCTGGCAGCGGTGTTCCGGGTAGCGGTGTGCCAGGCTCGG

GTGTGCCGGGCAGCGGTGTGCCGGGTTCAGGCGTTCCGGGTAGCGGTGTTCCGGGTA

GCGGTGTGCCTGGTAGCGGTGTGCCGGGTAGCGGCGTGCCGGGCAGCGGTGTTCCG

GGTAGCGGCGTGCCGGGTTCTGGGGTTCCGGGTTCCGGCGTGCCGGGTAGCGGCGTC

CCGGGTAATGGCGTGCCAGGCAACGGCGTTCCGGGCAACGGTGTTCCGGGAAACGG

CGTTCCGGGTAATGGTGTGCCGGGTAATGGTGTTCCGGGCAACGGCGTGCCGGGCA

ACGGCGTGCCTGGCAACGGTGTACCGGGTAATGGCGTTCCGGGAAACGGCGTGCCG

GGCAACGGTGTCCCAGGTAATGGCGTGCCTGGCAACGGTGTGCCGGGCAACGGTGT

CCCGGGCAACGGTGTGCCGGGTAATGGTGTTCCAGGCAATGGTGTTCCGGGCAATG

GCGTCCCGGGTAATGGTGTCCCGGGTAACGGCGTTCCTGGCAACGGCGTTCCGGGA

AACGGCGTACCGGGAAACGGTGTCCCGGGAAACGGTGTGCCAGGTTCAGGTGTCCC

GGGTTCTGGCGTTCCAGGCAGCGGCGTGCCGGGTTCTGGCGTACCGGGCAGCGGTGT

TCCGGGTTCGGGTGTCCCGGGTAGCGGTGTTCCGGGTAGCGGTGTTCCGGGCTCTGG

TGTTCCGGGTTCCGGTGTGCCGGGCAGCGGTGTGCCGGGTAGCGGCGTGCCGGGCA

GCGGTGTCCCGGGTAGTGGCGTACCGGGCTCCGGCGTTCCGGGTAGCGGCGTTCCG

GGTTCTGGTGTTCCGGGTAGCGGTGTTCCGGGCTCGGGCGTTCCAGGCTCGGGCGTG

CCAGGCAGCGGTGTTCCGGGTAGCGGGGTGCCGGGTAGCGGTGTTCCGGGTTCAGG

CGTGCCGGGCTCCGGCGTGCCGGGTTCCGGCGTGCCGGGTAGCGGCGTTCCCGGTTC
CGGTGTTCCGGGTAGCGGCGTGCCGGGCAGCGGTGTGCCGGGTTCGGGCGTTCCGG
GATCGGGGGTGCCAGGTAGCGGTGTACCGGGTTCCGGCGTTCCGGGTAGCGGTGTTC
CGGGGTCTGGCGTTCCGGGTTCCGGCGTTCCGGGCTCCGGTGTTCCGGGTAGCGGTG
TGCCGGGCAGCGGCGTGCCGGGCTCTGGCGTCCCGGGTTCTGGCGTGCCGGGCTCTG
GCGTCCCAGGCTCCGGCGTGCCGGGTAGCGGTTGA

Amino Acid:

MRGSHHHHHHGSENLYFQGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGS
GVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPG
SGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGNGVP
GNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPG
NGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGNGVPGN
GVPGNGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPG
SGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVP
GSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGV
PGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSGVPGSG*

Seven ELP sequences used in Chapter 5 were ordered to reflect hydrophobicity and charge. All genes except for ELP-N were ordered in pET28b(+) from Genscript. ELP-N was ordered in pET28a(+). All genes except for ELP-N were cloned into pET15b with restriction sites NdeI/XhoI using standard restriction cloning protocol. All sequences were confirmed via Sanger sequencing (Genewiz, USA).

ELP-I

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC

CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTACCTACGCCAGGAACACCCACTCC

GGGAACACCGATCCCAGGTACTCCAACTCCCGGTACGCCGACTCCTGGGACACCGA

CTCCAGGCACACCAACTCCGGGCACTCCGATCCCGGGTACGCCCACTCCCGGAACTC

CGACACCAGGGACACCAACCCCTGGAACTCCCACCCCTGGCACCCCAATCCCTGGT

ACCCCCACACCTGGAACCCCTACCCCCGGAACCCCGACACCTGGGACTCCCACGCC

AGGGACCCCTATGCCCGGGACTCCGACTCCGGGTACCCCTACTCCAGGTACACCCAC

ACCGGGTACTCCTACTCCTGGTACACCTATGCCAGGCACTCCTACACCCGGAACACC

TACTCCGGGGACACCTACACCGGGAACTCCAACCCCAGGAACGCCAATCCCCGGTA

CACCAACACCTGGTACGCCAACTCCAGGGACTCCAACGCCCGGTACCCCAACACCA

GGTACCCCGATGCCTGGAACGCCTACTCCCGGCACTCCCACACCAGGCACCCCTACA

CCAGGAACCCCAACTCCTGGAACACCAATGCCAGGTACGCCTACACCTGGCACTCC

AACACCCGGTACTCCGACGCCGGGGACTCCTACGCCTGGGACCCCAATGCCGGGTA

CACCGACCCCAGGGACGCCTACCCCGGGAACCCCCACTCCAGGAACTCCTACCCCA

GGCACGCCTATGCCTGGTACTCCCACTCCTGGCACACCTACCCCTGGGACTGGATCC

CGACCCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCAGG

CCGCTGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGIGVPGVGVPG

VGVPGVGVPGVGVPGIGVPGVGVPGVGVPGVGVPGVGVPGIGVPGVGVPGVGVPGVG

VPGVGVPGIGVPGVGVPGVGVPGVGVPGVGVPGIGVPGVGVPGVGVPGVGVPGVGVP

GIGVPGVGVPGVGVPGVGVPGVGVPGIGVPGVGVPGVGVPGVGVPGVGVPGIGVPGVG

VPGVGVPGVGVPGVGVPGIGVPGVGVPGVGVPGVGVPGVGVPGIGVPGVGVPGVGRS

KLAAALEDPAANKARKEAELAAATAEQ*

ELP-V

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC

CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTGCCAACACCGGGTACACCCACAC

CAGGCACACCAACCCCAGGCACTCCCACTCCGGGTACCCCAACACCCGGAACTCCC

ACGCCAGGTACGCCGACTCCCGGGACTCCAACGCCTGGTACACCTACGCCCGGTAC

GCCCACTCCCGGCACTCCAACACCTGGTACGCCTACACCCGGGACACCGACTCCGG

GAACTCCTACCCCGGGTACTCCTACTCCCGGTACTCCAACCCCTGGGACCCCAACTC

CCGGAACCCCTACACCGGGGACACCAACTCCAGGAACACCCACCCCAGGTACTCCG

ACACCAGGGACTCCGACGCCAGGGACGCCTACCCCTGGTACTCCCACCCCTGGCACT

CCTACGCCGGGTACGCCAACTCCTGGAACCCCGACTCCAGGCACGCCTACTCCTGGC

ACACCTACACCAGGTACCCCGACACCCGGTACCCCTACGCCTGGAACTCCGACTCCT

GGTACCCCCACACCTGGAACACCGACACCTGGGACTCCCACACCGGGAACCCCCAC

TCCTGGGACACCCACTCCAGGGACCCCTACCCCCGGTACACCAACGCCAGGCACCC

CTACTCCAGGTACACCGACCCCTGGAACGCCTACGCCAGGAACGCCAACCCCGGGA

ACACCTACCCCAGGGACACCTACTCCGGGCACTCCGACCCCAGGAACCCCAACCCC

CGGAACACCAACACCAGGAACTCCAACTCCGGGGACTCCTACACCTGGCACTGGAT

CCCGACCCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCA

GGCCGCTGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGVGVPGVGVP

GVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPG

VGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGV

GVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVG

VPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGVPGVGV

PGVGRSKLAAALEDPAANKARKEAELAAATAEQ*

ELP-N

Gene:

ATGGGCAGCAGCCATCATCATCATCACAGCAGCGGCCTGGTGCCGCGCGGCAG

CCATATGGCTAGCATGACTGGTGGACAGCAAATGGGTCGCGGATCCGTCCCAGGAG

TTGGGGTTCCCGGGGTAGGCGTACCGGGCAATGGAGTTCCGGGCGTTGGTGTTCCAG

GTGTGGGAGTGCCCGGAGTAGGTGTCCCTGGAGTCGGGGTACCTGGCAATGGGGTG

CCTGGAGTAGGAGTGCCAGGAGTGGGGGTTCCAGGGGTAGGGGTACCGGGTGTAGG

AGTCCCGGGTAACGGAGTACCCGGAGTCGGAGTCCCAGGGGTTGGAGTCCCTGGTG

TCGGCGTTCCAGGAGTCGGTGTCCCCGGCAATGGCGTTCCGGGGGTAGGTGTGCCAG

GTGTAGGGGTCCCAGGCGTGGGAGTTCCAGGCGTAGGTGTACCGGGGAACGGGGTT

CCTGGTGTAGGTGTTCCCGGTGTAGGCGTTCCTGGCGTAGGAGTTCCCGGAGTGGGA

GTACCTGGGAATGGCGTACCTGGAGTGGGTGTACCCGGTGTTGGGGTCCCTGGGGTA

GGAGTACCAGGTGTCGGAGTGCCTGGCAACGGTGTACCTGGTGTTGGAGTACCGGG
AGTAGGGGTGCCAGGGGTGGGAGTCCCCGGAGTTGGAGTGCCGGGAAACGGCGTAC
CCGGCGTAGGGGTTCCGGGTGTTGGCGTGCCTGGTGTGGGTGTTCCGGGAGTGGGCG
TACCAGGGAATGGGGTACCAGGAGTAGGCGTCCCTGGCGTCGGTGTTCCTGGAGTT
GGTGTCCCAGGTGTTGGTGTACCAGGCAATGGTGTGCCTGGGGTCGGAGTTCCTGGG
GTTGGTTAA

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRGSVPGVGVPGVGVPGNGVPGVGVP
GVGVPGVGVPGVGVPGNGVPGVGVPGVGVPGVGVPGVGVPGNGVPGVGVPGVGVPG
VGVPGVGVPGNGVPGVGVPGVGVPGVGVPGVGVPGNGVPGVGVPGVGVPGVGVPGV
GVPGNGVPGVGVPGVGVPGVGVPGVGVPGNGVPGVGVPGVGVPGVGVPGVGVPGNG
VPGVGVPGVGVPGVGVPGVGVPGNGVPGVGVPGVGVPGVGVPGVGVPGNGVPGVGV
PGVG*

ELP-S

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC
CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTGCCAACTCCGGGCACTCCTACACC
AGGTACCCCGCTCCCGGGTACTCCCACCCCAGGAACACCCACTCCGGGTACACCAA
CACCAGGCACACCTACCCCCGGTACTCCAGACCCCGGAACTCCCACTCCAGGCACTC
CAACCCCAGGTACACCGACTCCTGGAACTCCTACTCCAGGAACCCCCGAACCGGGT
ACGCCAACACCCGGAACCCCTACCCCGGGAACCCCGACTCCGGGAACTCCAACTCC

TGGCACCCCTGACCCTGGTACTCCGACACCTGGTACGCCGACTCCCGGTACACCCAC

ACCTGGAACCCCAACTCCCGGCACTCCCGAGCCGGGTACCCCTACACCCGGTACCCC

AACACCTGGCACACCAACGCCCGGTACGCCCACACCAGGAACGCCGCTGCCAGGCA

CGCCTACCCCTGGCACTCCGACGCCAGGTACGCCTACTCCCGGAACACCGACACCA

GGGACGCCTGAGCCCGGGACTCCGACCCCTGGAACACCAACCCCTGGGACACCTAC

GCCAGGGACCCCTACTCCTGGGACCCCAGAGCCAGGAACTCCGACTCCAGGGACAC

CAACTCCAGGTACTCCTACCCCAGGGACTCCCACGCCGGGAACGCCACTGCCTGGG

ACTCCAACACCGGGGACTCCTACGCCTGGTACCCCCACTCCTGGTACACCTACTCCG

GGGACACCGGAGCCTGGAACGCCTACACCGGGAACACCTACACCTGGGACTGGATC

CCGACCCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCAG

GCCGCTGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGSGVPGVGVP

GVGVPGVGVPGVGVPGSGVPGVGVPGVGVPGVGVPGVGVPGSGVPGVGVPGVGVPG

VGVPGVGVPGSGVPGVGVPGVGVPGVGVPGVGVPGSGVPGVGVPGVGVPGVGVPGV

GVPGSGVPGVGVPGVGVPGVGVPGVGVPGSGVPGVGVPGVGVPGVGVPGVGVPGSGV

PGVGVPGVGVPGVGVPGVGVPGSGVPGVGVPGVGVPGVGVPGVGVPGSGVPGVGVPG

VGRSKLAAALEDPAANKARKEAELAAATAEQ*

ELP-G

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC

CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTTCCTACGCCCGGTACACCGACCCC

AGGGACTCCCCCTCCGGGAACACCTACTCCGGGTACGCCCACTCCAGGAACGCCAA

CACCAGGGACGCCTACTCCTGGCACCCCACCTCCCGGCACTCCCACACCAGGCACTC

CTACTCCAGGTACTCCTACACCCGGGACACCTACACCGGGAACTCCTCCCCCGGGAA

CCCCTACACCAGGAACACCAACACCCGGAACTCCCACTCCTGGTACCCCAACGCCTG

GAACGCCTCCGCCGGGTACTCCCACGCCAGGTACCCCGACTCCGGGGACACCAACC

CCAGGCACACCTACCCCAGGTACGCCACCACCCGGTACTCCGACGCCAGGAACTCC

GACTCCCGGAACACCGACTCCTGGGACACCGACACCTGGGACTCCGCCCCCAGGAA

CCCCAACACCGGGTACCCCCACTCCCGGTACCCCTACGCCAGGCACCCCTACCCCTG

GCACACCACCGCCTGGTACACCCACCCCTGGAACACCCACTCCGGGCACTCCGACA

CCAGGTACACCAACTCCCGGGACTCCACCCCCTGGGACCCCAACTCCAGGCACGCCT

ACCCCCGGTACGCCTACACCTGGTACTCCAACTCCTGGAACCCCGCCGCCAGGGACA

CCCACACCTGGAACTCCAACACCTGGCACTCCAACCCCTGGTACGCCGACTCCAGGG

ACCCCTCCACCGGGGACTCCTACCCCGGGTACACCTACGCCTGGGACTGGATCCCGA

CCCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCAGGCCG

CTGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGGGVPGVGVP

GVGVPGVGVPGVGVPGGGVPGVGVPGVGVPGVGVPGVGVPGGGVPGVGVPGVGVPG

VGVPGVGVPGGGVPGVGVPGVGVPGVGVPGVGVPGGGVPGVGVPGVGVPGVGVPGV

GVPGGGVPGVGVPGVGVPGVGVPGVGVPGGGVPGVGVPGVGVPGVGVPGVGVPGGG

VPGVGVPGVGVPGVGVPGVGVPGGGVPGVGVPGVGVPGVGVPGVGVPGGGVPGVGV

PGVGRSKLAAALEDPAANKARKEAELAAATAEQ*

ELP-K

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC

CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTACCTACCCCAGGCACGCCTACACC

CGGAACACCTTTGCCCGGTACGCCTACTCCGGGAACTCCTACTCCAGGTACACCGAC

TCCGGGCACACCTACTCCCGGAACGCCCTTCCCCGGTACACCAACTCCCGGTACCCC

TACTCCTGGAACACCAACACCCGGTACTCCAACGCCTGGGACACCCTTACCCGGGAC

ACCGACACCTGGTACCCCCACTCCGGGTACGCCAACACCTGGGACCCCAACACCGG

GTACTCCTTTGCCGGGTACACCTACACCTGGCACACCCACACCAGGCACACCAACCC

CAGGAACACCGACCCCTGGAACGCCTTTGCCAGGCACTCCCACGCCCGGAACTCCA

ACACCAGGAACGCCAACTCCTGGTACTCCCACTCCTGGCACCCCTTTGCCTGGTACA

CCCACTCCAGGGACTCCGACTCCTGGGACTCCCACACCTGGAACCCCAACCCCCGGA

ACCCCCTTGCCAGGTACCCCAACTCCGGGGACACCAACGCCGGGGACTCCAACTCC

AGGAACCCCGACTCCCGGGACTCCTTTCCCTGGCACTCCTACACCAGGGACCCCTAC

ACCGGGAACCCCTACCCCTGGGACGCCTACCCCGGGAACACCCTTCCCAGGTACTCC

GACACCAGGTACGCCCACCCCAGGGACACCTACGCCTGGAACTCCGACGCCAGGAA

CTCCCTTTCCCGGCACTCCAACCCCTGGTACGCCGACTCCAGGCACTGGATCCCGAC

CCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCAGGCCGC

TGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGKGVPGVGVP

GVGVPGVGVPGVGVPGKGVPGVGVPGVGVPGVGVPGVGVPGKGVPGVGVPGVGVPG

VGVPGVGVPGKGVPGVGVPGVGVPGVGVPGVGVPGKGVPGVGVPGVGVPGVGVPGV

GVPGKGVPGVGVPGVGVPGVGVPGVGVPGKGVPGVGVPGVGVPGVGVPGVGVPGKG

VPGVGVPGVGVPGVGVPGVGVPGKGVPGVGVPGVGVPGVGVPGVGVPGKGVPGVGV

PGVGRSKLAAALEDPAANKARKEAELAAATAEQ*

ELP-D

Gene:

TTATTGCTCAGCGGTGGCAGCAGCCAACTCAGCTTCCTTTCGGGCTTTGTTAGCAGC

CGGATCCTCGAGTGCGGCCGCAAGCTTAGATCTACCAACACCTGGGACACCAACGC

CTGGAACGCCATCCCCTGGGACTCCAACCCCAGGTACACCTACGCCTGGGACGCCTA

CCCCTGGTACGCCAACACCGGGGACACCGTCTCCCGGCACTCCCACGCCAGGAACA

CCCACACCTGGCACACCTACCCCGGGTACACCCACTCCGGGTACCCCGTCACCGGGT

ACTCCAACTCCAGGCACGCCTACTCCTGGAACCCCAACACCCGGTACGCCTACACCA

GGCACACCATCGCCAGGCACTCCTACTCCCGGTACTCCTACACCTGGTACTCCGACC

CCAGGAACGCCGACTCCAGGTACCCCATCACCCGGAACTCCAACACCAGGGACCCC

TACTCCAGGGACTCCGACACCTGGAACACCGACTCCGGGCACTCCATCCCCAGGGA

CACCTACTCCGGGAACTCCTACGCCCGGTACACCGACACCAGGTACTCCCACCCCTG

GAACTCCGTCGCCTGGTACACCAACTCCTGGTACCCCTACACCGGGAACCCCTACCC

CAGGCACCCCTACGCCGGGTACGCCGTCCCCCGGTACCCCCACTCCCGGAACGCCTA

CGCCAGGTACGCCCACTCCAGGAACTCCCACTCCTGGCACCCCATCCCCGGGAACAC

CTACACCCGGGACTCCTACCCCCGGAACACCAACCCCTGGCACTCCGACTCCTGGGA

CCCCATCTCCGGGGACTCCCACACCAGGAACCCCGACGCCAGGGACCGGATCCCGA
CCCATTTGCTGTCCACCAGTCATGCTAGCCATATGGCTGCCGCGCGGCACCAGGCCG
CTGCTGTGATGATGATGATGATGGCTGCTGCCCAT

Amino Acids:

MGSSHHHHHHSSGLVPRGSHMASMTGGQQMGRDPVPGVGVPGVGVPGDGVPGVGVP
GVGVPGVGVPGVGVPGDGVPGVGVPGVGVPGVGVPGVGVPGDGVPGVGVPGVGVPG
VGVPGVGVPGDGVPGVGVPGVGVPGVGVPGVGVPGDGVPGVGVPGVGVPGVGVPGV
GVPGDGVPGVGVPGVGVPGVGVPGVGVPGDGVPGVGVPGVGVPGVGVPGVGVPGDG
VPGVGVPGVGVPGVGVPGVGVPGDGVPGVGVPGVGVPGVGVPGVGVPGDGVPGVGV
PGVGRSKLAAALEDPAANKARKEAELAAATAEQ*

2.1.2 Molecular Cloning

This section outlines the standard procedures used throughout the molecular cloning workflow.

Reagents and Materials. Restriction enzymes and reaction buffers were purchased from New England Biolabs (NEB). Enzymes were supplied as 20,000 units/mL glycerol stock solutions; high-fidelity (HF) versions were used whenever possible. Reaction buffers were obtained as 10× concentrates. E. coli strain NEB-5α (NEB) was used for all cloning procedures.

Preparation of Chemically Competent Cells. NEB-5α cells were made chemically competent using the Mix & Go! kit (Zymo Research #T3002) without modifications. Cells were aliquoted into 50 µL portions using sterile technique and stored at −80 °C until use.

Restriction Digests. Standard digests were performed in 50 µL total volume containing 1 µg of mini-prepped DNA, 5 µL of CutSmart buffer (NEB #B7204), and 1 µL of each restriction enzyme.

The reaction was brought to volume with Milli-Q water, mixed by pipetting, briefly centrifuged (~2 s), and incubated at 37 °C for an hour in a BioRad thermal cycler. Products were purified via agarose gel electrophoresis as described below.

Dephosphorylation. Dephosphorylation was performed following the restriction digest and prior to purification. To the reaction, 5.5 µL of Antarctic Phosphatase Buffer and 0.5 µL of Antarctic Phosphatase (NEB) were added. The mixture was pipetted to mix, centrifuged briefly, and incubated at 37 °C for 30 minutes.

Agarose Gel Electrophoresis and DNA Extraction. Digests were combined with 10 µL of 6× Orange DNA Loading Dye (ThermoFisher R0631) and run on 1% agarose gels prepared in 1× TAE (40 mM Tris, 20 mM acetic acid, 1 mM EDTA, pH 8.0) containing SYBR Safe DNA stain (ThermoFisher S33102). Gels were run at 100 V for 30 min, and DNA bands were visualized under UV light and excised with a clean razor. DNA was extracted using the EZNA Gel Extraction Kit (Omega Bio-Tek) and stored in Milli-Q water at −20 °C.

Ligation. Ligation reactions were carried out in 10 µL total volume with DNA at 7 ng/µL. Gel-purified vector and insert were combined at a 1:3 molar ratio (vector:insert), along with 1 µL of 10× T4 DNA Ligase buffer and 0.5 µL of T4 DNA Ligase (NEB). If the DNA concentration is below the detection limit of nanodrop, 2 ng/µL is assumed to be the concentration. Reactions were pipetted to mix, centrifuged briefly, and incubated at 16 °C overnight in a thermal cycler.

Transformation. Chemically competent NEB-5α cells were transformed with ligation products or freshly mini-prepped DNA following the NEB high efficiency transformation protocol (https://www.neb.com/en-us/protocols/0001/01/01/high-efficiency-transformation-protocol-

c2987). Briefly, to transform the ligated genes into NEB 5-alpha Competent *E. coli* cells, begin by thawing a tube of C2987H cells on ice for 10 minutes, or for C2987I cells, thaw on ice until the last ice crystals disappear. Gently mix and pipette 50 µL of cells into a pre-chilled transformation tube. Add 1–5 µL of plasmid DNA (containing 1 pg to 100 ng) to the cell suspension and gently flick the tube 4–5 times to mix—do not vortex. Incubate the mixture on ice for 30 minutes without mixing. Heat shock the cells at exactly 42 °C for 30 seconds, then immediately return the tube to ice for 5 minutes. Add 950 µL of room temperature SOC medium to the cells and incubate at 37 °C for 1 hour with vigorous shaking (250 rpm) or rotation. Meanwhile, pre-warm selection plates to 37 °C. After incubation, mix the cells by flicking and inverting the tube, then briefly centrifuge and remove 900 µL of the supernatant to concentrate the cells. Resuspend the pellet in the remaining volume. Plate 50–100 µL of the resuspended cells onto the selection plates and incubate overnight at 37 °C.

Mini-Prep. Overnight cultures (5–10 mL LB + antibiotic) of transformed colonies were mini-prepped using the Omega Bio-Tek D6942 Mini-Prep kit, following manufacturer instructions. DNA was eluted in 30 µL of Milli-Q water at concentrations of 100–200 ng/µL.

Sanger Sequencing. Forward and reverse Sanger sequencing was performed by Genewiz using 10 µL samples at 50 ng/µL in Milli-Q water. The T7 forward and reverse primers were supplied by Genewiz. pQE-9 primers were ordered from Genewiz:

pQE forward primer: 5′-CCCGAAAAGTGCCACCTG-3′

pQE reverse primer: 3′-GGTCATTACTGGAGTCTTG-5′

2.1.3 Protein Expression

After the genes were transformed into competent cells for expression and sequence confirmation, single colonies were then used to seed starter cultures, which were grown in 50 mL of LB media with corresponding antibiotics in baffled shake-flasks at 37 °C for 8–16 h. The starter culture (50 mL) was used to inoculate 5 L of TB or LB expression media supplemented with the corresponding antibiotics. Cells were grown at 37 °C and induced with 1 mM isopropyl-β-D-thiogalactoside (IPTG) once the expression media reached an OD600 of 0.6–1.0. After induction, cells were allowed to express protein at 37 °C for an additional 3 hours for ELP-K, 6–9 h for ELP-S, ELP-N, ELP-V, ELP-D, SI75, SN75, and 20 h for ELP-I. Cells were harvested by centrifugation and frozen in -20C until lysis.

2.1.4 Protein Purification

Cells were resuspended in 100 mL of lysis buffer (100mM NaH_2PO_4, 10mM Tris Cl, 8M urea, 0.6M NaCl, 20mM beta-mercaptoethanol, pH = 8) for every 30 g of wet cell mass. The resuspended cells were sonicated using a tip sonicator. This lysate was clarified by high-speed centrifugation at 4 °C for 30 mins.

Ni-NTA affinity resin was added to the clarified lysate for purification. 20mL Ni-NTA resins were added to the clarified lysate and let the mixture bind overnight at 4°C shaker. The Ni-NTA affinity column was washed with 4 column volume buffer B twice at pH = 8, 7.3, 6.3. The protein was eluted from the column with buffer B at pH = 5.9 twice and 4.3 five times to ensure complete elution of protein. The purification was confirmed with SDS-PAGE. The pure eluted proteins were dialyzed three times in EDTA buffer and five times in MillQ water, at least 3 h between each solvent change. For some proteins, small amounts of impurities become insoluble

after solvent exchange. The dialyzed products were centrifuged at 4°C and the supernatant were lyophilized for 3 days to yield a cotton-like white powder.

If the eluted protein is not pure, it is subjected to two rounds of inverse transition cycling (ITC). The lyophilized protein was dissolved at 2.5mg/mL in 2M NaCl where precipitation was induced by the addition of 2 M sodium chloride and incubation at 37 °C, and resolubilization was achieved by resuspension in water at 4 °C.

2.2 Turbidimetry

2.2.1 Sample Preparation. Lyophilized proteins were dissolved at 5 mg/mL in water overnight at 4 °C. Samples were then diluted with a combination of water, alcohol, and 0.5 M sodium chloride or 5 M sodium chloride to bring the solution to the correct solvent content and salt concentration and to bring the total protein concentration in solution to 1 mg/mL.

2.2.2 UV-Vis Setup. Samples were loaded into a quartz cuvette and capped. Dry air is flown into the holder to prevent condensation. Turbidimetry was measured using UV-Vis absorbance in a Varian Cary 50 spectrometer with a 1 cm pathlength quartz cuvette. Absorbance data were corrected using Milli-Q water as background. Temperature control was achieved via a water bath circulator using the T-App program, while absorbance was recorded using the Kinetics program. Ice was added to the water bath to achieve low temperatures. To synchronize the two, both programs were run concurrently. The cuvette temperature was heated at a rate of 1°C/min, and transmittance at 600 nm was monitored.

2.2.3 Data Analysis. A MATLAB script was used to convert absorbance to transmittance and merge the transmittance and temperature data. Transmission was normalized across its full range, and the cloud point temperature was defined as the temperature at which 50% of the normalized

transmittance was lost. Error bars on transition temperatures correspond to standard deviations over three measurements.

2.3 Small Angle Neutron Scattering

2.3.1 Sample Preparation and Loading. All deuterated solvents at 99.9% purity were purchased from Cambridge Isotopic Laboratories, Inc. End-functional poly(ethylene glycol) (PEG) polymer was synthesized following procedures in literature[1, 2] and dissolved in deuterated DMF at a concentration of 16mM. The associative protein hydrogel is formed by an artificial protein denoted as P4, which contains four associative coiled-coil domains (P) connected by flexible polyelectrolyte linkers (C10). The gene sequence has been reported in literature,[3] and the protein expression protocols have been developed previously[4]. The P4 hydrogel was prepared at 6.5% (w/v) by dissolving P4 in 100mM deuterated phosphate buffer at pD = 7.6. The blends were equilibrated overnight at 5 °C before stirring and centrifugation at 13, 100 X g at 10 °C to ensure homogeneous gel formation. Pluronic F-127 was purchased from Millipore Sigma. The polymer was dissolved in D_2O at 1.5 wt%. The P4 gel was loaded into a 1 mm demountable copper cell by spreading the gel on one side of the quartz window and compressing the second quartz window onto the sample carefully to prevent bubble formation. The Pluronic F-127 and the 16mM PEG solutions were loaded into 1 mm quartz banjo cells.

The SI75 and SN75 were dissolved in 100mM deuterated phosphate buffer with pH = 7.1 at 40w/v%. They were loaded into 0.5mm demountable copper cell by pipetting the samples on one side of the quartz window and compressing the second quartz window onto the sample carefully to prevent bubble formation. The 100mM phosphate buffer was loaded into 1mm banjo cell.

2.3.2 SANS Configuration. The small-angle neutron scattering experiments were performed on GP-SANS at Oak Ridge National Lab (ORNL)[5]. The lower q configuration has a sample-to-detector distance of 15 m, which corresponds to q range of 0.003698 to 0.04988 Å. The higher q configuration has a sample to detector distance of 2 m, which corresponds to q range of 0.03 to 0.43 Å. The wavelength for both configurations is 4.75 Å. All measurements were performed at room temperature. P4 gel and 16mM PEG solutions had approximately 500,000 counts above the background, and Pluronic F-127 had approximately 1,000,000 counts above the background.

The small-angle neutron scattering experiments were performed on GP-SANS and EQ-SANS at Oak Ridge National Lab (ORNL)[5]. For GP-SANS, the lower q configuration has a sample-to-detector distance of 4 m, which corresponds to low q range of $6.58 * 10^{-3}$ to $1.07 * 10^{-1}$Å. The higher q configuration has a sample to detector distance of 1.5 m, which corresponds to high q range of $3.76 * 10^{-2}$ to $4.64 * 10^{-1}$ Å. The wavelength for both configurations is 6 Å. Neutron counts are between 1 to 6 million at the low q and approximately 20 million at high q. SANS data were background subtracted. For EQ-SANS, two configurations were used to cover q range of 0.05 to 0.73 Å. The lower q configuration has a sample-to-detector distance of 7m with a wavelength of 10 Å. The higher q configuration has a sample to detector distance of 2.5m with wavelength of 2.5 Å. The aperture of the beam is 10mm. All observed transitions were reversible with temperature, indicating that minimal beam damage is observed.

2.3.2 Contrast Variation. 40 w/v% SI75 solutions were prepared at six different D_2O/H_2O blend compositions at 80/20, 60/40, 40/60, 30/70, 15/85, and 5/95 to provide variable contrast with both hydrophilic endblocks and hydrophobic midblock.

The data analysis for water partitioning was adapted from literature[6]. The intensities were subtracted for background. The primary peak intensities were selected at each D_2O/H_2O blend

compositions. The volume fraction of protein and water blend are calculated for 40w/v%. Assume 1.35g/mL as the protein density.

$$\frac{0.40g}{mL\ solvent} * \frac{mL\ solute}{1.35g} = 0.296 \frac{mL\ of\ solute}{mL\ of\ solvent} \tag{2.1}$$

$$\theta_{protein} = \frac{mL\ of\ solute}{mL\ of\ solute + mL\ of\ solvent} \tag{2.2}$$

By assuming a basis of 1mL of solvent here, the volume fraction of protein is

$$\theta_{protein} = \frac{0.296\ mL\ of\ solute}{0.296\ mL\ of\ solute + 1\ mL\ of\ solvent} = 0.2286 \tag{2.3}$$

$$\theta_{water\ blend} = 1 - \theta_{protein} \tag{2.4}$$

Multiply $\theta_{protein}$ by f = 0.75 to get volume fraction of S block.

$$\theta_S = \theta_{protein} * f \tag{2.5}$$

Multiply $\theta_{protein}$ by 1-f to get volume fraction of I block.

$$\theta_I = \theta_{protein} * (1 - f) \tag{2.6}$$

The scattering length densities (SLDs) of S block and I block were calculated using the NIST SLD calculator (https://www.ncnr.nist.gov/resources/activation/). The hydrated SLDs of each block were calculated as

$$SLD\ of\ hydrated\ S\ block = \frac{\theta_S * SLDs + \epsilon * SLD_{water\ blend}}{\theta_S + \epsilon} \tag{2.7}$$

where ϵ is the overall volume fraction of water blend hydrating S block with respect to total volume. The SLD of hydrated I block were calculated similarly.

$$SLD\ of\ hydrated\ I\ block = \frac{\theta_I * SLDs + \epsilon * SLD_{water\ blend}}{\theta_I + \epsilon} \quad (2.8)$$

Perform fitting to minimize the residual

$$Residual = \sum (I_{data} - C\Delta SLD^2) \quad (2.9)$$

Calculate the volume fraction with respect to the total amount of total water hydrating S block $\theta_{w,S}$ as

$$\theta_{w,S} = \frac{\epsilon}{\theta_{water\ blend}} \quad (2.10)$$

The rest of the water goes to the I block the volume fraction with respect to the total amount of total water hydrating S block $\theta_{w,I}$

$$\theta_{w,I} = 1 - \theta_{wS} \quad (2.11)$$

2.3.3 Model Functions

Debye Function

When uncharged polymers are dissolved in theta solvents, they form Gaussian coils in solution which can be described by the Debye function[29]. The Debye function is commonly used to obtain the radius of gyration of the polymer. The form factor of the Debye function to describe monodisperse polymer chains is

$$P(q) = \frac{2\left(e^{-q^2Rg^2} + q^2Rg^2 - 1\right)}{(q^2Rg^2)^2} \quad (2.12)$$

where q is the scattering wavevector, and R_g is the radius of gyration of the polymer. The bounds for R_g are set between 10 and 80 Å. For dilute solution, form factor fitting was used to perform parameter estimation. The scattering intensity can be expressed as

$$I\,(q) = A * P(q) + b \tag{2.13}$$

where A is the scale and b is the background scattering intensity in cm^{-1}. The bounds for A are set between 0 and 100 and the bounds for b are set between 0 to 10 for the multiple initializations.

Broad Peak Model

For physical gel systems, the correlation length and the peak components are important to determine the material properties. The characteristic structural information for many polymer gels can be obtained by fitting to the empirical broad peak model[30-32] based on the correlation length model[32-39] to demonstrate the effectiveness of this algorithm on empirical models. The fitting function for scattering intensity is

$$I(q) = \frac{A'}{q^n} + \frac{C'}{1 + (|q - q_0|\xi)^m} + b \tag{2.14}$$

where q_0 is the peak position of the primary peak in q, ξ is a local correlation length, n and m are Porod and Lorentzian exponents, respectively, A' and C' are empirical parameters, and b is the incoherent background scattering intensity in cm^{-1}. The range of q_0 is set to 0.01 to 0.07. The range of ξ is set to 0 to 100. The range of m and n is set to 0 to 5. The range of A' is set to 0 to 0.1. The range of C' is set to 0 to 10 times the maximum value of intensity. Following fitting methods of the same system and model functions[27], the fitting is first performed on the low q region using $\frac{A\prime}{q^n}$. After obtaining values of A' and n, those two values are fixed, and the fitting is performed over

the entire q range with Equation 2.14 to obtain the rest of the fitting parameters. The peak position q_0 can be used to calculate the domain spacing d_0.

$$d_0 = \frac{2\pi}{q_0} \tag{2.15}$$

Spherical Micelle Model with Polydispersity

Pluronic F-127 is a triblock copolymer poly(ethylene oxide)$_{99}$-poly(propylene oxide)$_{69}$-poly(ethylene oxide)$_{99}$[40] (PEO$_{99}$-PPO$_{69}$-PEO$_{99}$). Above the critical micelle concentration, this polymer self-assembles into spherical micelles. Spherical micelle model with polydispersity has been chosen based on literature[41]. Here, the parameters of interest are the radius of gyration R_g of the PEO and the radius of the PPO core R. The bounds for R_g are set between 10 to 80 Å and the bounds for R are set between 20 to 80 Å based on literature[41]. The analytical expression of the form factor $P(q)$ in the spherical micelle model[42, 43] contains four terms that contribute to the scattering intensity: the self-correlation term of the core, the self-correlation term of the chains, the cross-term between the core and the chains, and the cross term between different chains[44]:

$$P(q,R) = N^2\beta_s^2 F_s(q) + N\beta_c^2 F_c(q) + 2N^2\beta_s\beta_c S_{sc}(q) + N(N-1)\beta_c^2 S_{cc}(q) \tag{2.16}$$

Here, β_s and β_c are the total scattering length of the core and the corona, respectively, calculated as

$$\beta_s = V_s(\rho_s - \rho_{solvent}) \tag{2.17}$$

$$\beta_c = V_c(\rho_c - \rho_{solvent}) \tag{2.18}$$

where V_s and V_c are the volumes of a block in the core and in the corona of a single copolymer molecules. Based on literature reported values[45], V_s is set to 6283 Å^3 and V_c is set to 14667 Å^3.

N is the aggregation number of the micelle and can be calculated as follows

$$N = \frac{\frac{4}{3}\pi R^3}{V_s} \tag{2.19}$$

ρ_s and ρ_c are the corresponding scattering length densities and $\rho_{solvent}$ is the scattering length density of the solvent, which is deuterated water in this case. ρ_s and ρ_c are fitting parameters, since they can vary depending on the solvent distribution. Based on the scattering length density calculations of PEO and PPO, the bounds for ρ_s are set between 4.0*10^{-6} Å^{-2} to 5.0 *10^{-6} Å^{-2} and the bounds for ρ_c are set between 4.5 *10^{-6} Å^{-2} to 5.5*10^{-6} Å^{-2}. $\rho_{solvent}$ is set to 6.3*10^{-6} Å^{-2}. The self-correlation term of the core is given as

$$F_s(q) = \Phi^2(qR) \tag{2.20}$$

where

$$\Phi(qR) = \frac{3[\sin(qR) - qR\cos(qR)]}{(qR)^3} \tag{2.21}$$

The self-correlation term $F_c(q)$ of the Gaussian chains is given by the Debye function[29] in Equation 2.12.

The interference cross term between the core and the chains is then

$$S_{sc}(q) = \Phi(qR)\psi\left(qR_g\right)\frac{\sin\left(q\left[R + dR_g\right]\right)}{q\left[R + dR_g\right]}, \tag{2.22}$$

where $\psi(x) = [1 - e^{-x}]/x$. The interference term between the chains in the corona is

$$S_{cc}(q) = \psi^2(qR_g)\left[\frac{\sin(q[R + dR_g])}{q[R + dR_g]}\right]^2. \quad (2.23)$$

where *d* is the factor to mimic non-penetration of Gaussian chains. The bounds for *d* are set between 0.5 to 2.0.

The scattering intensity with polydispersity can be expressed as

$$I(q) = n_{density} * P_{poly}(q) + b, \quad (2.24)$$

$$P_{poly}(q) = \int f_{SZ}(R, z)P(q, R)dR \quad (2.25)$$

where $n_{density}$ is the number density of micelle, which is defined as the number of micelles per unit volume in cm^3 and calculated using mass concentration $C_{wt\%}$, aggregation number *N,* solution density ρ_{sol}, and Avogadro's number N_A.

$$n_{density} = \frac{C_{wt\%}}{100} * \frac{\rho_{sol}}{N * Munimer} * N_A \quad (2.26)$$

$P_{poly}(q)$ is the form factor with polydispersity, *b* is the background from incoherent scattering. The bounds for *b* are set between 0 to 1, given the low background noise in the data. The distribution $f_{SZ}(R, z)$ is chosen as a Schulz-Zimm distribution of the radius of the PPO core *R* following literature procedures[46] defined as follows.

$$f_{SZ}(R, z) = \frac{R^z}{\Gamma(z + 1)}\left(\frac{z + 1}{\langle R\rangle}\right)^{z+1} e^{\left(-(z+1)\frac{R}{\langle R\rangle}\right)} \quad (2.27)$$

$$z = \frac{1}{\sigma^2} - 1 \tag{2.28}$$

Where $\langle R \rangle$ is the expectation value of the micelle core radius. z is the parameter related to the width of the distribution and σ is the polydispersity factor. Here σ has bounds between 0.1 to 1. The lower bound is set to 0.1 because small values of σ yields very large z, which can cause numerical instability (e.g., overflow or underflow) when evaluating terms such as $(z+1)^{z+1}$. These instabilities can lead to NaN values in numerical implementations, especially in floating-point arithmetic.

Percus Yevick Model. The Percus Yevick model[25] with disordered hard spheres was used to estimate the structural parameters in SI75 at temperatures above the transition temperature.

The structure factor *S(q)* is

$$S(q) = \frac{1}{1 + 24\eta\, G(2qR)} \tag{2.29}$$

Where η is the micelle packing fraction. *G(x)* is expressed as

$$G(x) = \frac{\alpha}{x^3}(sinx - xcosx) + \frac{\beta}{x^4}(2xsinx + (2 - x^2)cosx - 2) + \frac{\gamma}{x^6}(-x^4 cosx + 4[(3x^2 - 6)cosx + (x^3 - 6x)sinx + 6]) \tag{2.30}$$

With constants

$$\alpha = \frac{(1 + 2\eta)^2}{(1 - \eta)^4} \tag{2.31}$$

$$\beta = -6\frac{\left(1+\frac{\eta}{2}\right)^2}{(1-\eta)^4} \tag{2.32}$$

$$\gamma = \frac{\eta}{2}\frac{(1+2\eta)^2}{(1-\eta)^2} \tag{2.33}$$

The form factor *F(q)* was computed for polydisperse spheres,

$$F(q) = \frac{\int P(r)f^2(q,r)dr}{\int P(r)dr} \tag{2.34}$$

Where variable *r* represents the size of the spheres. The function $f(q,r)$ is expressed as

$$f(q,r) = \frac{4}{3}\pi r^3 \frac{3}{(qr)^3}(sin\ qr - qr\cos qr) \tag{2.35}$$

The lognormal probability distribution *P(r)* for the size of the spheres is given by

$$P(r) = \sqrt{2\overline{\sigma_p}^2\pi}\exp\left(\frac{(-\ln r - \overline{r_0})^2}{2\overline{\sigma_p}^2}\right) \tag{2.36}$$

The expected value (r_0) and standard deviation (σ_p) of the size distribution could be computed according to

$$r_0 = \exp\left(\overline{r_0} + \frac{1}{2}\overline{\sigma_p}^2\right) \tag{2.37}$$

$$\sigma_p = r_0\sqrt{\exp\left(\overline{\sigma_p}^2\right) - 1} \tag{2.38}$$

The intensity of the scattering is

$$I(q) = KF(q)S(q) + background \tag{2.39}$$

Where *K* is a multiplicative constant and *background* is a constant term accounting for any uniform background scattering in the experimental data.

Correlation Length Model. The correlation length model[10] is an empirical function for SANS, particularly when the scattering profile exhibits distinct behaviors at low and high q. This model effectively captures the combined effects of Porod scattering from clusters and Lorentzian scattering from polymer chains, providing insights into the structural characteristics of complex systems. This model was used to fit temperature dependent scattering data of SN75.

The scattering intensity *I(q)* is expressed as

$$I(q) = \frac{A}{q^n} + \frac{C}{1 + (q\xi)^m} + background \tag{2.40}$$

where A, the Porod Scale Factor, determines the amplitude of the Porod scattering term, reflecting the contribution from large-scale structures or clusters. n, the Porod Exponent, influences the power-law decay of the Porod term. C, the Lorentzian Scale Factor, sets the amplitude of the Lorentzian term, associated with the scattering from polymer chains or similar structures. ξ, the correlation Length, represents the characteristic length scale over which density correlations persist in the system. m, the Lorentzian Exponent, determines the sharpness of the Lorentzian peak. *Background* is a constant term accounting for any uniform background scattering in the experimental data.

2.4 Depolarized Light Scattering

2.4.1 Sample Preparation. Depolarized light scattering (DPLS) was performed on samples loaded into a 1 mm thick Teflon mold with an inner diameter of 3 mm and sealed between two quartz disks in a brass sample holder.

2.4.2 Measurement. A Coherent OBIS LX660 laser with wavelength λ = 662 nm and continuous wave output power 20 mW was used. Samples were equilibrated at 4°C for 30 minutes and then heated at 1°C/min to 60°C for SI75 and 70°C for SN75, allowed to equilibrate for 5 minutes. A NESLAB recirculating water chiller was programmed in MATLAB (See Appendix) to control the temperature of the sample stage. The static birefringence was corrected for transmission and dark field background; necessary background detector readings and the intensity of the incident laser beam needed for data analysis were measured at least once at the beginning of each day of DPLS experiments. For turbidimetry measurements, the same apparatus was used without the rear polarizer to enable measurement of sample transmission.

1. I_{dark} is the detector voltage when the laser is turned off or the shutter is closed.

2. I_{cross} is the detector voltage when the front and rear polarizers are at perpendicular orientation to one another.

3. I_{open} is the detector voltage reading measured with the laser operating at 20 mW with the neutral density (ND) filter set to 6 (to prevent saturation and damage of the photodiode detector) and without any sample loaded.

2.4.3 Data Analysis. Transmission *Ts* is calculated as

$$Ts = \frac{I_{transmission}}{I_{open}} \tag{2.41}$$

Where $I_{transmission}$ is measured from turbidimetry.

Power fractions I_{PF} are calculated as the ratio of the birefringent intensity I to the laser intensity I_{open}

$$I = \frac{1}{Ts}(I_B - I_{dark}) - (I_{cross} - I_{dark}) \tag{2.42}$$

$$I_{PF} = \frac{I}{I_{open}} \tag{2.43}$$

2.5 Rheology

2.5.1 Sample Preparation and Loading. SI75 and SN75 were dissolved in 100mM phosphate buffer at pH = 7.1 and placed in fridge. Rheological measurements were performed on an Anton Paar MCR 702 rheometer. Rheological measurements were performed on an Anton Paar MCR 702 rheometer. Thermal control was achieved using a Peltier plate as the bottom geometry. A cone (diameter: 25 mm, cone angle: 1°) was mounted onto a shaft as the upper geometry. Inertial calibration and motor adjustment were performed before each measurement. Temperature was reduced to 4°C prior to sample loading for SN75 and 0°C for SI75. Samples were placed on ice and loaded with chilled pipette tips. After reaching the measurement position, mineral oil was added to the sample edge to minimize solvent evaporation.

2.5.2 Measurement. Prior to measurement, the sample was equilibrated for 30 minutes to an hour to ensure that the moduli are stable. For SN75, strain sweep from γ 0.01 to 5% was first performed at 4, 30, and 60°C to determine the optimal strain value for linear regime. Temperature sweep was performed from 4°C to 70°C at strain γ 0.1% at 1°C/min at ω = 1 rad/s. Frequency sweep was performed at 4°C, 30°C, 40°C, 50°C, 60°C, 70°C from ω 100 to 0.01 rad/s. For SI75, strain sweep from 0.01 to 10% was first performed at 2, 25, 30, 60°C to determine the optimal strain value for linear regime. Temperature sweep was performed from 2°C to 65°C at strain γ 0.5% and 1% at 1°C/min at ω = 1 rad/s. Frequency sweep was performed at 2°C, 14°C, 30°C, 40°C, 60°C from ω 100 to 0.01 rad/s.

2.6 Molecular Dynamics Simulations

2.6.1 Simulation Setup

The simulation setup is reproduced and modified from the thesis by Yao et al[26]. In the hard sphere–soft sphere (HS) dumbbell model for protein-polymer conjugates, globular protein–polymer bioconjugates were represented as dumbbells of hard spheres bonded to soft spheres *via* a stiff harmonic spring in implicit solvent (Figure 2.1). The hard sphere represents the globular protein, and the soft sphere represents the flexible polymer. In the simulation, the base unit of length (σ) was set to be the diameter of the hard sphere. This was taken to be twice the radius of gyration of mCherry (σ = 3.12 nm),[27] which is the model globular protein most frequently used in experimental studies of bioconjugate self-assembly in concentrated solution (Figure 2-1(a)). The diameter of the hard sphere was set to σ for all bioconjugates. The radius of the soft sphere in a dumbbell was varied relative to the hard sphere based on the desired coil fraction of the bioconjugate (between 0.21 – 0.82, corresponding to those studied in experiments), as shown in Figure 2-1(e). The volume fraction of the soft sphere was mapped to a physical coil fraction through Equation 2.44:

$$f_2 = \frac{N_2 v_2}{\frac{\pi\sigma^3}{6} + N_2\, v_2} = \frac{6R_{g2}^2 v_2/b^2}{\frac{\pi\sigma^3}{6} + 6R_{g2}^2 v_2/b^2} \tag{2.44}$$

Here, N_2 is the degree of polymerization, v_2 is the molecular volume of a monomer unit, R_{g2} is the radius of gyration of the polymer, b is the Kuhn length, and σ is the diameter of a hard sphere. Species 1 denotes the hard sphere (protein), and species 2 denotes the soft sphere (polymer). The degree of polymerization was substituted with the ideal chain expression for radius of gyration, $N = 6R_g^2/b^2$. The desired coil fractions were obtained from experiments on the self-

assembly of mCherry-*b*-PNIPAM bioconjugates;[28, 29] thus, PNIPAM was selected as the reference polymer for Equation 2.44, with a Kuhn length of 0.733 nm, calculated from R_g measured for PNIPAM with a molar mass at the symmetric coil fraction (26.3 kDa).[27] Using this statistical segment length, the radius of gyration was calculated for selected coil fractions from experiment and used to set the size of the soft spheres (Figure 2-1e). Five coil fractions were selected from the mCherry-*b*-PNIPAM phase diagram[28, 29] to span the range of coil fractions characterized in SAXS experiments (Figure 2-1e).

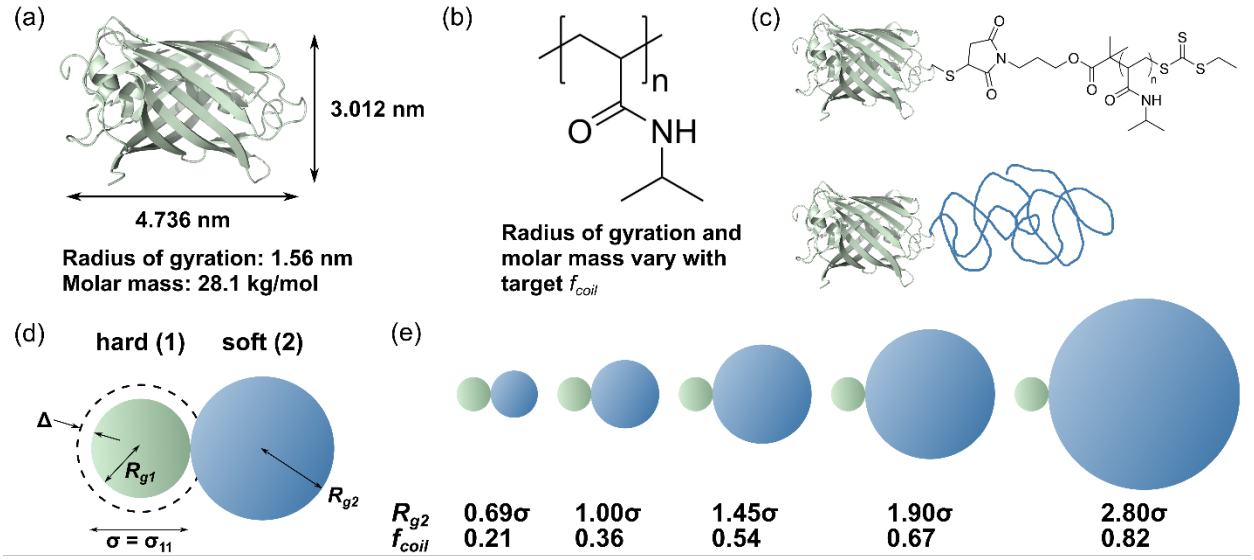


Figure 2-1. Schematic of coarse-grained HS dumbbell model and physical protein and polymer on which it is based. (a) mCherry structure and size (b) Chemical structure of poly(*N*-isopropylacrylamide) (PNIPAM) (c) Structure of mCherry-*b*-PNIPAM bioconjugate (d) Coarse-grained HS dumbbell model of bioconjugate consisting of a hard sphere (green, representing protein) and a soft sphere (blue, representing polymer) with relevant length scales noted (e) Bioconjugates with selected coil fractions probed by simulations with equivalent polymer sizes. Reproduced from manuscript.

Dumbbell concentration was calculated as mass per unit volume (i.e. g/mL) based on the number of molecules in a simulation box and the volume of the simulation box. Given a box of volume V in units of σ^3 and n molecules (i.e. bioconjugates or dumbbells) with a molar mass of M in g/mol, the concentration c in equivalent g/mL can be calculated as Equation 2.45.

$$c = \frac{10^{21} nM}{8 N_A V R_{g1}^3} \tag{2.45}$$

The numerical prefactors stem from unit conversion from simulation units of σ to cm. N_A is Avogadro's number.

There are three types of intermolecular interactions in this system: hard–hard (1–1), soft–soft (2–2), and hard–soft (1–2). The model uses a shifted Weeks-Chandler-Andersen (WCA) potential (Equation 2.46) for the hard–hard interaction and Gaussian potentials[30] for the soft–soft (Equation 2.47) and hard–soft (Equation 2.48) interactions. The shifted WCA potential was selected to capture the purely repulsive interactions between hard spheres with an added exclusion layer, Δ, to enforce hardness and represent the hydration shell of the protein block. The Gaussian potential was chosen for the interaction between soft spheres, as it was adapted from a potential developed for modeling polymers as chains of fluctuating soft spheres.[30] The base simulation length and energy units are σ and ε, which are referenced to the diameter of the hard sphere protein and 1 k_BT, respectively. A temperature T of 1.00 corresponded to 25 °C.

$$U_{11}(r) = \begin{cases} 4\epsilon_{11}\left[\left(\frac{\sigma_{11}}{r-\Delta}\right)^{12} - \left(\frac{\sigma_{11}}{r-\Delta}\right)^{6}\right] + \epsilon_{11}, r < r_{c,11} + \Delta \\ 0, r \geq r_{c,11} + \Delta \end{cases} \tag{2.46}$$

$$U_{22}(r) = \begin{cases} \epsilon_{22}\left[\frac{3}{2\pi\left(R_{g2}^2 + R_{g2}^2\right)}\right]^{3/2} \exp\left[\frac{-3r^2}{2\left(R_{g2}^2 + R_{g2}^2\right)}\right], r < r_{c,22} \\ 0, r \geq r_{c,22} \end{cases} \quad (2.47)$$

$$U_{12}(r) = \begin{cases} \epsilon_{12}\left[\frac{3}{2\pi\left(R_{g1}^2 + R_{g2}^2\right)}\right]^{3/2} \exp\left[\frac{-3r^2}{2\left(R_{g1}^2 + R_{g2}^2\right)}\right], r < r_{c,12} \\ 0, r \geq r_{c,12} \end{cases} \quad (2.48)$$

The parameters used in each of the potentials in Equations 2.46-2.48 are summarized in Table 2-1. For the hard–hard interaction, the parameters are derived from experiments characterizing mCherry block copolymers. The parameter Δ was set to be the size of the hydration shell; this was approximated as 0.28σ (0.88 nm), which is the size of the hydration shell for green fluorescent protein (GFP),[31] which has a similar folded structure and second virial coefficient as mCherry.[32] The radius of gyration of mCherry, which sets $\sigma_{11} = \sigma = 3.12$ nm, was determined from a dilute solution SANS experiment using Guinier analysis by Lam et al.[27] The Lennard-Jones interaction parameter ε was derived from the experimentally obtained second virial coefficient of mCherry *via* Equation 2.49.

$$B_2(T) = \frac{A_2 M^2}{N_A} = -2\pi \int_0^\infty r^2\left(e^{-\beta U(r)} - 1\right)dr \quad (2.49)$$

$B_2(T)$ is the thermodynamically defined second virial coefficient, which is a function of temperature (T), A_2 is the second virial coefficient measured for mCherry from a static light scattering experiment, M is the molar mass of mCherry, N_A is Avogadro's number, β is $1/k_BT$, and $U(r)$ is the intermolecular potential (here, the shifted WCA potential shown in Equation 2.46). This formulation of the second virial coefficient uses A_2 for mCherry, which is $1.1 \pm 0.1 \times 10^{-4}$ mol mL g^{-2}, measured at 25 °C. [28] This is equivalent to $B_2 = 144.2$ nm^3. The difference between the second

and third terms in Equation 2.49 was solved numerically for ε_{11} using *fsolve* in MATLAB. The integration was also performed in MATLAB using the trapezoidal rule (*trapz*). This yielded a value of $\varepsilon_{11} = 1.54\varepsilon$. As a physical comparison, Equation 2.49 was also solved analytically for the true hard sphere potential ($U(r) = \infty, r < \sigma$), which shows that a hard sphere with B_2 = 144.2 nm^3 would have a theoretical diameter of 4.1 nm; using twice the mCherry radius of gyration and a hydration layer of 0.88 nm yields a "hard sphere" with a diameter of approximately 4.0 nm.

The soft–soft interaction was modeled by the Gaussian potential developed by Vettorel et al.[30] and was parametrized from work done by Louis et al. in modeling polymers as "soft colloids,"[33] which suggests that $U_{22}(0) \sim 2k_BT$. The radius of gyration of the polymer depends on the coil fraction of interest.

The hard–soft interaction length parameters are set by the sizes of protein and polymer corresponding to different coil fractions of block copolymer studied in experiment.[28, 29] The interaction between hard colloids and polymers has been studied in the semidilute regime. In the colloid limit, Bolhuis and Louis have parametrized an effective pair potential between 500 polymers obeying a self-avoiding random walk around a single colloid sphere, going up to a concentration of 2.18 times the overlap concentration.[34] Extrapolating this to a higher concentration, for example, 5 times the overlap concentration, suggests that the interaction potential at r = 0 lies around $9k_BT$. In the protein limit, Odijk has proposed cross second virial coefficients between a single protein surrounded by polymers in semidilute solution that asymptotically approach $\left(\sqrt{6}R_{g2}\right)^{\frac{5}{3}}R_{g1}^{\frac{4}{3}}$ and $R_{g2}^2R_{g1}$ in good and theta solvent, respectively.[35] Using Equation 2.49 with $U_{12}(r)$ instead of $U_{11}(r)$ suggests that the overlap energy would lie between ~0.3 – 1.2k_BT in good solvent and between ~0.8 – 2.1k_BT in theta solvent, depending on

the coil fraction. SANS partial structure factor analysis for mCherry and PNIPAM, where both species are in the semidilute regime, also suggests a small interaction (~0.012k_BT) between protein and polymer.[36] This estimate was obtained by fitting Equation 2.50, which uses the Gaussian potential shown in Equation 2.48 for $U(r)$, to the extrapolated experimental cross structure factor[36] for mCherry and PNIPAM with molar mass 26.3 kDa.

$$S_{12}(T) = \frac{4\pi}{q}\int_0^\infty r\sin(qr)\left(e^{-\beta U(r)} - 1\right)dr \tag{2.50}$$

While hard–soft interaction is a good approximation for systems where the polymer and protein are similar in size, our system involves polymers whose radii of gyration exceed that of mCherry by a factor of 2–5. In this case, the Gaussian potential becomes an effective, empirical description. We acknowledge that for highly asymmetric size ratios or high concentrations, more sophisticated treatments may be required.

For all of these estimates, the theory or experiment was developed near overlap or in the semidilute regime; therefore, the assumptions that underlie these estimates may not apply in the concentrated regime of interest in this study. This pairwise approximation is most accurate in the dilute to semidilute regime, Vettorel et al. demonstrate that, when parameterized carefully, it can reliably describe even polymer melts. Nevertheless, at high colloid volume fractions, many-body interactions may become relevant, and we note this as a limitation of the current model. Thus, the value of the hard–soft interaction parameter was varied between 0.012–200k_BT.

Table 2-1. Intermolecular Potential Parameters.

Parameter	**Value**	**Physical Interpretation**	**Source**
σ_{11}	1σ	$2R_g$ of mCherry	SANS experiment[27]

ε_{11}	1.54ε	Protein interaction energy	Second virial coefficient from experiment[32]
Δ	0.28σ	Hydration shell	QENS[31] experiment
$r_{c,11}$	$2^{1/6}\sigma$	Cutoff for WCA-type potential	
R_{g1}	0.5σ	Radius of gyration of mCherry (1.56 nm)	SANS experiment[27]
R_{g2}	Depends on f_{coil}	Radius of gyration of polymer	Desired coil fractions chosen from experimental phase diagrams[28, 29] with R_{g2} calculated *via* Equation 2.44.
ε_{22}	Selected such that $U_{22}(r = 0) = 2k_BT$		Theory[30, 33]
ε_{12}	Explored values in range of $U_{12}(r = 0) = 0.012k_BT - 200k_BT$		
$r_{c,12}$ or $r_{c,22}$	Cutoff for Gaussian potentials; selected such that $U_{ij}(r = r_{c,ij}) = 0.001k_BT$		

The hard sphere was bonded to the soft sphere via a harmonic spring bond with a spring constant of $1000\varepsilon\sigma^{-2}$ and an equilibrium bond length equal to $R_{g1} + R_{g2}$. There were no intramolecular interactions between spheres. The dimensionless mass of the hard sphere was set to 1, and the mass of the soft sphere was adjusted to reflect the coil fraction and molar mass of the experimental polymer block relative to the molar mass of the protein block.

2.6.2 Initialization, Equilibration, and Production

All MD simulations were run using LAMMPS in NPT ensemble. Initial structures were built in Python script. Simulations were moved into the isothermal–isobaric ensemble by applying a Nose/Hoover temperature thermostat and Nose/Hoover pressure barostat to maintain the temperature at $T = 1.00$ and pressure to achieve desired average concentrations within 1%; the damping factor was two orders of magnitude larger than the time step, as suggested by LAMMPS documentation (this condition was found to be sufficient for reliable temperature control).

At the end of the simulation, the system configuration was saved for the equilibration and production runs. Equilibration generally required 500,000–2,000,000 time steps, depending on the size of the system, after which morphologies and potential energies remained stable for the production run. Trajectories were saved into XYZ-style dump files and visualized using Ovito. Structure factor was calculated from simulation trajectories using Fast Fourier Transform (FFT) from the production runs using Fourier transforms in MATLAB. Block copolymer morphology was determined using both structure factor peak indexing and visual confirmation in Ovito.

2.6.2 Addressing commensurability

2.6.2.1 Building Axially Aligned Initial Structures

Incommensurability arises when the ordered structure domain spacing and the periodicity of the simulation box are mismatched during MD simulations, resulting in frustration, distortions, and unphysical states in the final morphology. Even though the box length chosen here is an integer number of the domain spacing, the resulted lamellar structure was slightly rotated from random initialization[26].

To demonstrate that the rotated lamellar structure is not due to incommensurability, the lamellar bilayers need to be aligned with the box axis, and the commensurate box size needs to be

determined systematically, and the domain spacing should be the same with the rotated lamellar from random initialization. A hexagonally packed lamellar structure was constructed using a Python-based coordinate generation script for molecular dynamics simulations. The system consisted of paired atoms arranged in bilayers formed by two parallel hexagonal planes. Alternating orientations between layers created an A–B–A–B stacking pattern to approximate face-centered cubic (FCC) packing. Interlayer spacing and in-plane offsets were based on tetrahedral geometry to maintain close-packing. Hexagonal planes were tiled in two dimensions, and bilayers were stacked along the third dimension to fill the simulation volume. The generated structure was rescaled and centered within a periodic simulation box. A subset of molecules was randomly selected to downsize to the target concentration, and coordinates were wrapped to enforce periodic boundary conditions. The final atomic positions and bond connectivity were exported in a format compatible with molecular dynamics engines such as LAMMPS. The representative resulting structure is shown in Figure 2-2(a). The detailed code can be found in Appendix named Build_Lamellar_Stack.py.

For generating hexagonally packed cylinders, a hexagonally packed array of cylindrical structures was generated using a coordinate-building algorithm that places atom pairs along helically rotating triangular motifs. Each cylinder was constructed as a vertical stack of equilateral triangles, where the orientation of each triangle rotated incrementally along the cylinder axis, resulting in a chiral or twisted structure. Cylinders were arranged in a hexagonal lattice within the simulation box using a shifted grid, with alternating rows offset to achieve close-packed symmetry. Each atom pair within a cylinder was defined by a directional vector normal to the triangle edge, with one atom positioned along the vector and the other at the base vertex. The resulting structure was trimmed to the desired system size by randomly selecting a subset of molecules. Atomic

positions and bonding information were written in a format compatible with LAMMPS for molecular dynamics simulations. The representative resulting structure is shown in Figure 2-2(b). The detailed code can be found in Appendix D.2 Build_HEX.py.

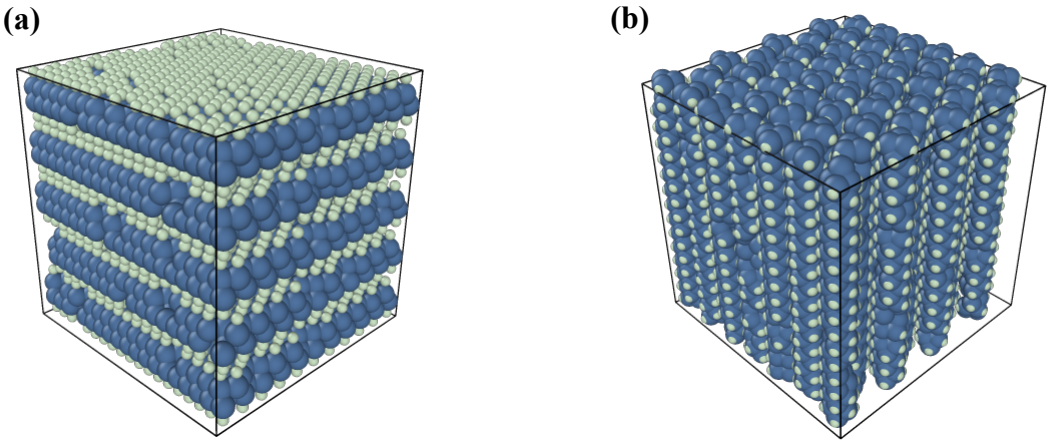


Figure 2-2. Representative Initial Structures from Python Script. (a) Lamellae. (b) Hexagonally packed cylinders.

2.6.2.2 Determining Pressure in NPT Ensemble

To determine the pressure corresponding to a target density under isothermal–isobaric (NPT) conditions, a numerical bisection method was implemented. The procedure involved running a series of short NPT ensemble molecular dynamics simulations at varying pressure values, starting from an initial bracket of upper and lower bounds. At each iteration, the simulation was run at the midpoint pressure, and the resulting equilibrium density was extracted from the time-averaged trajectory. Depending on whether the observed density was above or below the target, the pressure bounds were updated accordingly to narrow the search interval. The process continued until the computed density was within a specified tolerance of the desired value. This iterative

refinement enabled accurate calibration of the pressure required to reproduce a given density. The converged pressure value was then used for subsequent production simulations.

2.6.3 Equilibrium Assessment

Mesoscale self-assembly, such as that of protein–polymer bioconjugates at high concentrations, can become kinetically trapped in both experiments and simulations. Therefore, rigorous equilibrium testing is essential to distinguish true equilibrium structures from metastable states. In this study, two levers are employed to assess equilibration convergence: increasing the temperature and using different crystal assembly configurations as initial structure.

2.6.3.1 Mean square displacement (MSD) Analysis

Mean square displacement (MSD) analysis was performed to quantify the dynamics particles over time to assess whether the particles were kinetically trapped. Unwrapped particle trajectories were extracted from NVT simulations, and the MSD was computed as a function of lag time τ using Equation 2.51.

$$MSD(\tau) = \langle \|r_i(t+\tau) - r_i(t)\|^2 \rangle_{i,t} = \frac{1}{N(F-m)} \sum_{j=0}^{F-m-1} \sum_{i=1}^{N} \|r_i(t+\tau) - r_i(t)\|^2 \qquad (2.51)$$

$r_i(t)$ is the position vector of particle i at time step t. τ is the lag time over which displacement is measured. N is the total number of particles being tracked. F is the total number of time steps in the trajectory. m is the number of steps corresponding to lag time τ. j is the time origin index, ranging from 0 to $F-m-1$.

This time- and ensemble-averaged quantity captures the extent of particle displacement due to diffusive motion. For systems with periodic boundary conditions, unwrapped coordinates were used to ensure accurate displacement calculations. The resulting MSD curves and the maximum

value of MSD value were used to assess the diffusion of particles and the regions of kinetic arrest.

If the MSD computed from Equation 2.51 as a function of lag time τ is linear with a slope of 1 with error based on the baseline at low concentrations and R^2 value of 1 with respect to τ, the lag time, the system exhibits is ergodic. This shows that it can freely explore its configurational landscape. If the slope and R^2 value starts to deviate from the baseline at low concentrations, it indicates that the system is in high probability of kinetically arrest.

2.6.3.2 Time-Resolved Clustering Analysis

To identify and track dynamic clustering behavior in molecular dynamics simulations to ensure equilibration, a time-resolved density-based clustering analysis was performed using the DBSCAN algorithm in Python. Trajectories were saved every 10^5 time steps as one time frame. For each time frame, the center-of-mass (COM) of the soft sphere was computed from atomic coordinates, and molecules were clustered in three-dimensional space using DBSCAN with a distance cutoff (*eps*) and a minimum of 5 neighboring points based on the position of the soft sphere. The first time frame corresponds to the time frame at which the structure forms, and each DBSCAN-identified cluster is assigned a unique persistent cluster ID. In the subsequent time frames, clustering was re-applied independently, and the newly identified clusters were matched to previously tracked ones by comparing the spatial proximity of their centers of mass. Matching was based on the minimum image convention to account for periodic boundary conditions. Each persistent cluster's center was updated for the subsequent time steps, and its lifetime was tracked for all the subsequent time frames. Molecules were considered part of a cluster if they were assigned to any non-noise DBSCAN label, while those not satisfying the density condition were labeled as noise. Cluster transitions for each molecule were recorded across frames to determine

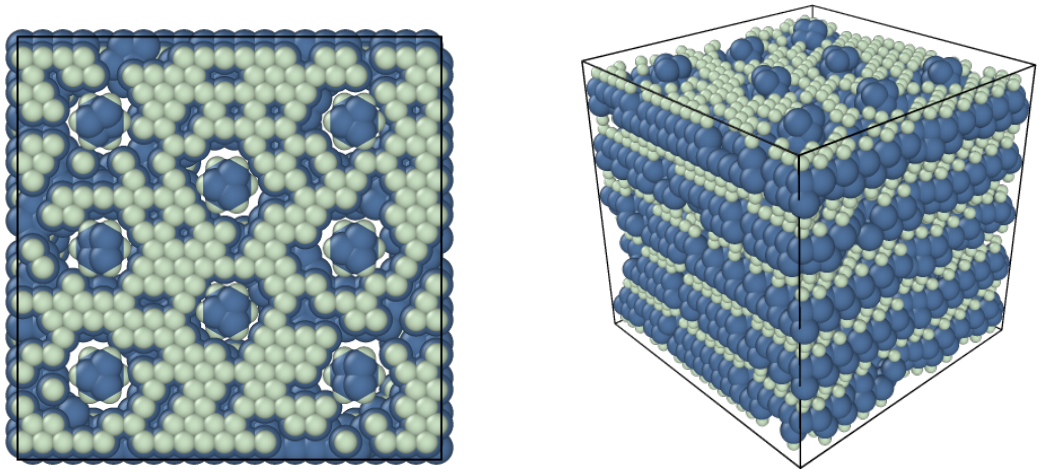
whether molecules remained trapped in a single cluster, transitioned between multiple clusters, or remained unclustered. The number of molecules transferring between clusters at each tiem frame was also tracked and recorded. The fraction of molecules that transitioned between clusters was calculated by dividing the number of molecules transferring between clusters at each saved time frame (corresponding to every 10^5 time steps) by the total number of molecules, followed by averaging across all saved frames. Cluster composition and evolution were visualized using time frame snapshots and statistical distributions of cluster lifetimes and molecular transitions. This approach enabled quantitative characterization of the spatial and temporal organization of molecular assemblies. The clustering analysis code can be found in Appendix D.

2.6.3.3 Initialization from Other Crystal Assembly

For lamellar and hexagonally packed cylinders, the initial structures are built the same as described in section 2.6.2.1.

A perforated lamellar structure was generated by modifying a hexagonally packed bilayer system constructed with alternating layers of atom pairs in face-centered cubic (FCC)-like arrangements. The base lamellar stack was built by assembling paired atoms into bilayers, with each layer consisting of two parallel hexagonal planes. Alternating orientation between layers created an A–B–A–B stacking geometry. To introduce perforations, cylindrical regions were defined at specified in-plane coordinates that allows even distribution of the cylinders, and atoms located within a given radial distance from each perforation center were removed from the lamellae. These cylindrical voids were subsequently filled with a rigid framework consisting of atoms positioned along helical or triangular paths, producing a columnar reinforcement that spans the vertical extent of the simulation box. The resulting system was rescaled, centered, and wrapped into a periodic simulation domain. A fixed number of molecules was randomly sampled from the

structure to downsize to the target system concentration. Final atomic coordinates and bond connectivity were exported in a format suitable for molecular dynamics simulations as shown in Figure 2.3. The detailed code can be found in Appendix D named Build_Perforated_Lamellar.py.



(a) **(b)**

Figure 2-3. Representative Initial Perforated Lamellae Structure from Python Script. (a) Top View. (b) Side view.

2.6.3.4 Replica Exchange

Replica exchange was performed using the *temper* command in LAMMPS (available from the REPLICA package). Temperatures were distributed exponentially with the ratio between adjacent temperatures $T_1/T_0 = 1.01$. Additional temperature replicas are inserted depending on the acceptance rate. Replica exchange was performed for the phase boundaries of lamellar phases. Replicas were distributed across a temperature range of 1.00 to 1.85 or 2.0, which was designed to give an acceptance probability of ≥ 20% for all swapping pairs[37]. The acceptance probability was monitored after running three sets of replica exchange simulations in NPT for 500,000 steps, followed by two stages each with 2,000,000 steps and checked for acceptance rate. Potential energy histograms were built after each stage and monitored for change in energy. Once the energy reached a minimum (i.e. the histograms stopped drifting downward), an isothermal-isobaric

simulation with WCA and Gaussian potentials was run at $T = 1.00$ for 2,000,000 steps using the trajectories recorded in the restart file for the replica corresponding to this temperature.

2.7 References

1. Beech, H. K.; Johnson, J. A.; Olsen, B. D., Conformation of Network Strands in Polymer Gels. *ACS Macro Letters* **2023,** *12* (3), 325-330.

2. Zhong, M.; Wang, R.; Kawamoto, K.; Olsen, B. D.; Johnson, J. A., Quantifying the impact of molecular defects on polymer network elasticity. *Science* **2016,** *353* (6305), 1264-1268.

3. Glassman, M. J.; Chan, J.; Olsen, B. D., Reinforcement of Shear Thinning Protein Hydrogels by Responsive Block Copolymer Self-Assembly. *Adv Funct Mater* **2013,** *23* (9), 1182-1193.

4. Rao, A.; Yao, H.; Olsen, B. D., Bridging dynamic regimes of segmental relaxation and center-of-mass diffusion in associative protein hydrogels. *Physical Review Research* **2020,** *2* (4), 043369.

5. Heller, W. T.; Cuneo, M.; Debeer-Schmitt, L.; Do, C.; He, L.; Heroux, L.; Littrell, K.; Pingali, S. V.; Qian, S.; Stanley, C.; Urban, V. S.; Wu, B.; Bras, W., The suite of small-angle neutron scattering instruments at Oak Ridge National Laboratory. *Journal of Applied Crystallography* **2018,** *51* (2), 242-248.

6. Lam, C. N.; Olsen, B. D., Phase transitions in concentrated solution self-assembly of globular protein–polymer block copolymers. *Soft Matter* **2013,** *9* (8), 2393-2402.

7. Debye, P., Molecular-weight determination by light scattering. *The Journal of Physical Chemistry* **1947,** *51* (1), 18-32.

8. Horkay, F.; Hammouda, B., Small-angle neutron scattering from typical synthetic and biopolymer solutions. *Colloid and Polymer Science* **2008,** *286* (6), 611-620.

9. Edwards, C. E. R.; Mai, D. J.; Tang, S.; Olsen, B. D., Molecular anisotropy and rearrangement as mechanisms of toughness and extensibility in entangled physical gels. *Physical Review Materials* **2020,** *4* (1), 015602.

10. Hammouda, B.; Ho, D. L.; Kline, S., Insight into clustering in poly (ethylene oxide) solutions. *Macromolecules* **2004,** *37* (18), 6932-6937.

11. Hore, M. J. A.; Hammouda, B.; Li, Y.; Cheng, H., Co-Nonsolvency of Poly(n-isopropylacrylamide) in Deuterated Water/Ethanol Mixtures. *Macromolecules* **2013,** *46* (19), 7894-7901.

12. Hule, R. A.; Nagarkar, R. P.; Hammouda, B.; Schneider, J. P.; Pochan, D. J., Dependence of Self-Assembled Peptide Hydrogel Network Structure on Local Fibril Nanostructure. *Macromolecules* **2009,** *42* (18), 7137-7145.

13. Saffer, E. M.; Lackey, M. A.; Griffin, D. M.; Kishore, S.; Tew, G. N.; Bhatia, S. R., SANS study of highly resilient poly(ethylene glycol) hydrogels. *Soft Matter* **2014,** *10* (12), 1905-1916.

14. Matsunaga, T.; Sakai, T.; Akagi, Y.; Chung, U.-i.; Shibayama, M., SANS and SLS Studies on Tetra-Arm PEG Gels in As-Prepared and Swollen States. *Macromolecules* **2009,** *42* (16), 6245-6252.

15. Kanaya, T.; Ohkura, M.; Kaji, K.; Furusaka, M.; Misawa, M., Structure of Poly(vinyl alcohol) Gels Studied by Wide- and Small-Angle Neutron Scattering. *Macromolecules* **1994,** *27* (20), 5609-5615.

16. Mallam, S.; Horkay, F.; Hecht, A. M.; Rennie, A. R.; Geissler, E., Microscopic and macroscopic thermodynamic observations in swollen poly(dimethylsiloxane) networks. *Macromolecules* **1991,** *24* (2), 543-548.

17. Shibayama, M.; Isono, K.; Okabe, S.; Karino, T.; Nagao, M., SANS study on pressure-induced phase separation of poly (N-isopropylacrylamide) aqueous solutions and gels. *Macromolecules* **2004,** *37* (8), 2909-2918.

18. Hamley, I. W., *Block copolymers in solution: fundamentals and applications*. John Wiley & Sons: 2005.

19. Kim, T.-H.; Han, Y.-S.; Jang, J.-D.; Seong, B.-S., SANS study on self-assembled structures of Pluronic F127 triblock copolymer induced by additives and temperatureThis article will form part of a virtual special issue of the journal, presenting some highlights of the 15th International Small-Angle Scattering Conference (SAS2012). This special issue will be available in early 2014. *Journal of Applied Crystallography* **2014,** *47* (1), 53-59.

20. Pedersen, J. S., Form factors of block copolymer micelles with spherical, ellipsoidal and cylindrical cores. *Journal of Applied Crystallography* **2000,** *33* (3-1), 637-640.

21. Pedersen, J. S.; Gerstenberg, M. C., Scattering Form Factor of Block Copolymer Micelles. *Macromolecules* **1996,** *29* (4), 1363-1365.

22. Pedersen, J. S., Analysis of small-angle scattering data from colloids and polymer solutions: modeling and least-squares fitting. *Advances in Colloid and Interface Science* **1997,** *70*, 171-210.

23. Lin, Y.; Alexandridis, P., Temperature-Dependent Adsorption of Pluronic F127 Block Copolymers onto Carbon Black Particles Dispersed in Aqueous Media. *The Journal of Physical Chemistry B* **2002,** *106* (42), 10834-10844.

24. Manet, S.; Lecchi, A.; Impéror-Clerc, M.; Zholobenko, V.; Durand, D.; Oliveira, C. L. P.; Pedersen, J. S.; Grillo, I.; Meneau, F.; Rochas, C., Structure of Micelles of a Nonionic Block Copolymer Determined by SANS and SAXS. *The Journal of Physical Chemistry B* **2011,** *115* (39), 11318-11329.

25. Percus, J. K.; Yevick, G. J., Analysis of Classical Statistical Mechanics by Means of Collective Coordinates. *Physical Review* **1958,** *110* (1), 1-13.

26. Yao, H. Driving forces of self-assembly in protein-polymer bioconjugates. Massachusetts Institute of Technology, 2020.

27. Lam, C. N.; Chang, D.; Wang, M.; Chen, W.-R.; Olsen, B. D., The shape of protein–polymer conjugates in dilute solution. *J Polym Sci A Polym Chem* **2016,** *54* (2), 292-302.

28. Lam, C. N.; Olsen, B. D., Phase transitions in concentrated solution self-assembly of globular protein-polymer block copolymers. *Soft Matter* **2013,** *9* (8), 2393-2402.

29. Thomas, C. S.; Olsen, B. D., Coil fraction-dependent phase behaviour of a model globular protein-polymer diblock copolymer. *Soft Matter* **2014,** *10* (17), 3093-3102.

30. Vettorel, T.; Besold, G.; Kremer, K., Fluctuating soft-sphere approach to coarse-graining of polymer models. *Soft Matter* **2010,** *6* (10), 2282-2292.

31. Perticaroli, S.; Ehlers, G.; Stanley, C. B.; Mamontov, E.; O'Neill, H.; Zhang, Q.; Cheng, X.; Myles, D. A. A.; Katsaras, J.; Nickels, J. D., Description of Hydration Water in Protein (Green Fluorescent Protein) Solution. *J Am Chem Soc* **2017,** *139* (3), 1098-1105.

32. Lam, C. N.; Kim, M.; Thomas, C. S.; Chang, D.; Sanoja, G. E.; Okwara, C. U.; Olsen, B. D., The Nature of Protein Interactions Governing Globular Protein–Polymer Block Copolymer Self-Assembly. *Biomacromolecules* **2014,** *15* (4), 1248-1258.

33. Louis, A. A.; Bolhuis, P. G.; Hansen, J. P.; Meijer, E. J., Can Polymer Coils Be Modeled as ``Soft Colloids"? *Phys Rev Lett* **2000,** *85* (12), 2522-2525.

34. Bolhuis, P. G.; Louis, A. A., How To Derive and Parameterize Effective Potentials in Colloid−Polymer Mixtures. *Macromolecules* **2002,** *35* (5), 1860-1869.

35. Odijk, T., Protein−Macromolecule Interactions. *Macromolecules* **1996,** *29* (5), 1842-1843.

36. Huang, A.; Yao, H.; Olsen, B. D., SANS partial structure factor analysis for determining protein–polymer interactions in semidilute solution. *Soft Matter* **2019,** *15* (37), 7350-7359.

37. Sugita, Y.; Okamoto, Y., Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters* **1999,** *314*, 141-151.

Chapter 3 Accelerated Small Angle Neutron Scattering Algorithms for Polymeric Materials

3.1 Abstract

Small-angle neutron scattering (SANS) is an extremely powerful technique for characterizing a wide variety of soft, biological, magnetic, and quantum materials, but it is often throughput-limited. This work proposes an algorithm to accelerate small angle neutron scattering (SANS) experiments by estimating the minimum number of counts to perform parameter estimation and model differentiation tasks to a specified level of certainty. Three classes of model polymer materials were examined and analyzed, and time slices of SANS data were used to model a reduced number of counts. The scattering data with reduced numbers of counts were fitted to SANS model functions to perform parameter estimation and model differentiation tasks. For parameter estimation, estimators accurate to within 5-10% of the full count estimator can be produced with only 1-50% of the full counts depending upon the sample and parameter of interest. In order to project parameter uncertainties at lower number of counts prior to the completion of experiments, it is crucial to have a robust error quantification method that reflects the true uncertainty associated with each parameter. Uncertainties from Monte Carlo (MC) bootstrapping are shown to in general overestimate the error from fitting many experimental replicates. For most parameter estimation techniques, the weighted least squares estimator is unbiased; however, certain models yield biased estimators. To differentiate between models, both the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) can be used, and with either criterion, reduced numbers of counts can still identify the best model for our samples from a group of related candidate models for each material. The proposed algorithm can help SANS users optimize valuable beamtime and accelerate the use of SANS for structural characterization of libraries of materials while obtaining reasonable parameter estimation and model differentiation.

3.2 Introduction

Neutron scattering is a useful technique to study the static and dynamic structures in a variety of materials such as polymers[1], colloids[2], biomacromolecules[3], metals[4], glasses[5], and ceramics[6]. Among many neutron scattering techniques, small angle neutron scattering (SANS) is one of the most widely used in probing the microstructure of polymer and soft materials[7]. SANS has been successfully used to characterize the structural formation in polymer solutions, melts, and networks and has advanced the understanding of fundamental polymer physics[8, 9]. The use of neutrons provide several advantages over other radiation sources, such as X-ray or light, because neutrons are nondestructive to soft matter[10] and allow contrast variation using isotopic labeling[11]. However, neutron count rates are inherently lower because neutrons do not interact very strongly with many materials, and neutron sources generally have lower flux than other radiation sources. These effects make SANS experiments slower. Typically, SANS data is acquired for a pre-determined number of counts above the background. After data acquisition and reduction, models are fit to experimental data to extract relevant structural parameters or identification of structures based on the goodness of fit. The measurement time to obtain SANS data for each sample can range from ten minutes to tens of hours depending on the facility, the scattering power of the sample, and instrument configuration[12]. Because neutron sources are expensive to construct and maintain, neutron scattering is only available at a handful of facilities worldwide, limiting access to this characterization technique.

With the surge of machine learning[13, 14] and data driven research in material discovery[15], the need to generate databases of materials with experimental data has led to many automated high throughput material syntheses in daily experimental workflow[16-18]. With the importance of neutron scattering in studying the microstructure and dynamics of many materials, many large facilities

have been improving the hardware of the instruments, upgrading the software at the beamline[19], making the data reduction process automated, and building higher flux neutron facilities[20] to increase the accessibility of neutrons to users in the world.

Herein, a novel workflow is proposed that focuses on experimental information content rather than total counts, allowing counts to be reduced with minimal loss of knowledge gained by an experiment. To do this, two common tasks are considered: parameter estimation and structural model differentiation. The algorithm for counts optimization uses uncertainty estimated in real time for both tasks, and decisions can be made regarding the optimal number of counts to achieve a targeted uncertainty level. Three different model polymer samples for SANS experiments examined in this paper include a polymer in solution, associative protein hydrogel, and micellar solution; however, the proposed algorithm can be generally applied to many other systems.

3.3 Methods

3.3.1 SANS Model Fitting

The selected models were fit to reduced data as a part of parameter estimation experiments by minimizing the least squared residual weighted by the error in scattering intensity using MATLAB command *lsqnonlin*. The parameter constraints were set based on the physics of the parameter and literature values. Within the bounds, 100 different initial conditions were randomly generated and supplied to the optimization algorithm to assess the global convergence. The confidence interval was output by *nlparci*, which calculates 95% confidence intervals from asymptotic normal distributions[47]. Briefly, orthogonal-triangular decomposition is performed on the Jacobian matrix J computed by *lsqnonlin*,

$$J = QU \tag{3.1}$$

where Q is the orthogonal matrix and U is the upper triangular matrix.

U is inverted to M.

$$M = U^{-1} \tag{3.2}$$

The diagonal of the Fisher information matrix F can be computed as

$$F_{i,i} = \sum_j \left(M_{i,j}\right)^2 \tag{3.3}$$

The root mean square error (*RMSE*) is computed as

$$RMSE = \frac{\|resid\|_2}{\sqrt{v}} \tag{3.4}$$

where *resid* is the residual, and v is the degrees of freedom.

$$v = n - p \tag{3.5}$$

where n is the number of data points and p is the number of parameters.

The sample variance se_i for parameter i is then calculated as

$$se_i = \sqrt{F_{i,i}} * RMSE \tag{3.6}$$

The margin of error delta can be computed as

$$delta_i = se_i * t_{0.025,v} \tag{3.7}$$

where t is the value for a confidence interval calculated from the student t-distribution at 95% confidence with v degrees of freedom. The confidence interval can then be calculated from

$$CI = \theta \pm delta \tag{3.8}$$

where θ is the vector of parameter values obtained from weighted nonlinear least square fitting.

Other model functions used for model differentiation can be found in the Appendix A.5.

3.3.2 Analysis on SANS Intensity Errors

From time slicing into 0.01 fraction of counts, each full count SANS data set generates 100 time-sliced SANS experimental replicates. The variance of the 100 intensity values at a given q was calculated and compared against the square of the 100 errors of intensity (ΔI^2), which were output by the SANS reduction file provided by ORNL. The percent difference is calculated as

$$\frac{Var[I(q)] - \Delta I^2}{Var[I(q)]} \tag{3.9}$$

where $Var[I(q)]$ denotes the calculated variance from intensities and ΔI^2 denotes the variance outputted by the SANS reduction file.

The scaling analysis on ΔI was selected for low q (q = 0.0151), mid q (q = 0.03999), and high q (q = 0.250). Each ΔI was plotted as a function of the fraction of counts at 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1. The ΔI values for low q, mid q, and high q were fitted to

$$\Delta I = \frac{scale}{\sqrt{a}} \tag{3.10}$$

where a is the fraction of counts, and $scale$ is a fitting parameter. The objective function f for fitting $scale$ to minimize the sum of least squares is

$$f(scale) = \log(\Delta I) - \log\left(\frac{scale}{\sqrt{a}}\right) \tag{3.11}$$

3.3.3 Checking the Normality of Weighted Residuals

Experimentally measured SANS data, y, can be described as

$$y = f(X) + \varepsilon \tag{3.12}$$

where *f(X)* is the analytical model function with parameters *X*, and ε is the residual. For a perfect fit, the residual is random error that can be modeled using a Gaussian distribution with a mean of 0 and standard deviation of ΔI. The weighted residuals, which are residuals normalized by ΔI, should follow a normal distribution with mean of 0 and standard deviation of 1. The weighted residual from fitting the P4 experimental and MC bootstrapping SANS data to the broad peak model were computed and checked for normality. Histograms were generated to examine the distribution of weighted residuals from fitting both the experimental data and the MC bootstrapping data.

3.3.4 Estimating the Bias of Weighed Least Squares

After performing parameter estimation with weighted nonlinear least square fitting using full count experimental SANS data, the best fit parameters were used to generate an analytical fitted SANS intensity curve. To simulate measurement noise, Gaussian noise with a mean of 0 and standard deviation of ΔI was added to the analytical fitted intensity curve. Following this workflow, 100 replicates with Gaussian noise were generated to simulate SANS data and fitted to the same model. If the estimator is unbiased, parameters' expectation values from the 100 fitting results should be equal to the parameter values used to generate the simulated SANS curve. The expectation value was computed by taking the mean of each parameter distributions. The bias was computed as

$$Bias\ [\%] = \frac{E[X] - X_{true}}{X_{true}} * 100 \tag{3.13}$$

where X_{true} is the parameter value used to generate the simulated SANS data and $E[X]$ is the expectation value of parameter distribution from fitting the simulated SANS data.

3.3.5 Calculating the Fisher Information Matrix and its Eigenvalue Decomposition

The variance is estimated by

$$\sigma^2 = \frac{RSS}{N - k} \tag{3.14}$$

where N is the number of data points for the sample. RSS is the residual sum of squares. k is the number of free fitting parameters.

The Fisher Information Matrix F can be computed from the Jacobian matrix J^{50} computed by *lsqnonlin*

$$F = \frac{1}{\sigma^2} * J' * J \tag{3.15}$$

Eigenvalue decomposition is performed on F to obtain the corresponding eigenvalues and eigenvectors.

3.3.6 Calculating Spatial Correlation of Intensity Values

From time slicing into 0.01 fraction of counts, full count SANS data generates 100 time-sliced SANS experiment replicates. The residual function $RES(q)$ of intensity is computed as

$$RES(q) = I(q) - E[I(q)] \tag{3.16}$$

Where *I(q)* is the intensity at q, and $E[I(q)]$ is the expectation value or mean of the 100 replicates of intensity at each q. The correlation coefficient matrix of *RES(q)* as each pairwise q is computed based on the Pearson coefficient using the *corr* function in MATLAB.

The correlation function $G(\Delta q)$ is calculated as

$$G(\Delta q) = \frac{\sum_{100\ replicates} \sum_{all\ q} RES(q)RES(q+\Delta q)}{\sum_{100\ replicates} \sum_{all\ q} RES(q)^2} \tag{3.17}$$

3.3.7 Information Criteria Calculations

The Akaike Information Criteria (AIC), derived from the asymptotic approximation of the Kullback-Leibler divergence, is often useful for model selection by the maximum likelihood method. AIC considers not only the goodness of fit, but also penalizes the number of parameters. The expression for calculating AIC is

$$AIC = -2\ln(\mathcal{L}) + 2k \tag{3.18}$$

where $\mathcal{L}$ is the maximum likelihood estimate and k is the number of free parameters. By assuming independently and identically distributed Gaussian distribution of the data points, the maximum log-likelihood can be approximated as[51]

$$\ln(\mathcal{L}) = -\frac{1}{2} N log\left(\frac{RSS}{N}\right) \tag{3.19}$$

where N is the number of data points for the sample and *RSS* is the residual sum of squares.

Bayesian Information Criterion (BIC), on the other hand, is derived from Bayesian probability theory and assumes a Bayesian approach to model selection[52]. It introduces a stronger penalty for model complexity than AIC, aiming to select the model that is most likely, given the data, but also has the fewest parameters.

$$BIC = -2\ln(\mathcal{L}) + \mathrm{k}\log(N) \tag{3.20}$$

3.4 Results and Discussion

3.4.1 Parameter Estimation Task

Fitting the SANS curves of dilute polymer solution (16mM PEG solution) and spherical micelle (Pluronic F-127) shows that a reduced number of counts can achieve good parameter estimation. Literature has shown that d-DMF is well-approximated as a theta solvent for PEG[24]. The 16mM PEG solution demonstrated good parameter estimation when fit to the Debye model to extract R_g at a reduced number of counts. The R_g value with 95% confidence interval from fitting the full number of counts is 29.17 ± 0.248 Å. Figure 3-1 (a-d) shows the 1D SANS scattering pattern and nonlinear least square fitting curve as the fraction of counts increases. One hundred random initializations are used for assessing the global convergence of the parameter estimation. The same values of fitted parameters and the weighted least square values were obtained at each fraction of counts as shown in Supplementary Information. Even when the counts are reduced to 0.01 fraction of the full counts, the value of the R_g is within 5% of R_g from full number of counts as shown in Figure 3-2(a). At low fraction of counts, the mean parameter value R_g from fitting bootstrapping replicates deviates from the ground truth obtained from fitting experimental replicates. As the number of counts starts to increase, the parameter values obtained from fitting experimental replicates converge with those obtained from MC bootstrapping due to decreases in ΔI. The mean R_g value from MC bootstrapping is consistent with the experimental data it is simulated from.

The uncertainty of R_g from fitting experimental replicates decreases faster than that of MC bootstrapping and 95% confidence interval as shown in Figure 3-2. Three distinct notions of uncertainty are readily accessible: margins of error from 95% confidence intervals of nonlinear

least squares fitting, standard deviations of parameters from fitting MC bootstrapping replicates, and standard deviations of parameters from fitting experimental time sliced replicates. As shown in Figure 3-2(b), MC bootstrapping overestimates the uncertainty of R_g at any fraction of counts compared to experimental time-sliced replicates. Margin of error from 95% confidence interval underestimates the uncertainty obtained from experimental replicates at 0.01, 0.025, 0.1 fraction of counts, and overestimates at 0.05, 0.25, 0.5 fraction of counts. The uncertainties of R_g in 16mM PEG solution indicate that the 95% confidence interval is closer to the experimental uncertainty of R_g between 0.01 and 0.1 fraction of counts than MC bootstrapping. Above 0.1 fraction of counts, MC bootstrapping and margin of error from 95% confidence intervals give similar uncertainty values, which both overestimate the true experimental uncertainty.

The uncertainties on the parameter R_g from MC bootstrapping in general follows a $\frac{1}{\sqrt{a}}$ scaling law, where a is the fraction of counts as shown in Figure 3-2(b). The exception occurs at 0.01 fraction of counts, where the standard deviation of R_g obtained from MC bootstrapping is significantly higher than the scaling prediction and that from the experimental replicates. This can be attributed to the fact that the variances output from the SANS reduction file exceeds the empirical variances from time slicing. Variances output by the SANS reduction file (ΔI^2) are 485% higher at q = 0.015 and 5.63% higher at q = 0.25 compared to variances from the empirical intensity distributions ($Var[I(q)]$) at corresponding q values. The bootstrapping parameter error scaling allows users to decide the optimal number of counts to reach a desired parameter uncertainty level from a short SANS scan. Since the true uncertainty from fitting experimental replicates is always lower than those from MC bootstrapping, using the MC bootstrapping error can provide a reliable and stopping criterion.

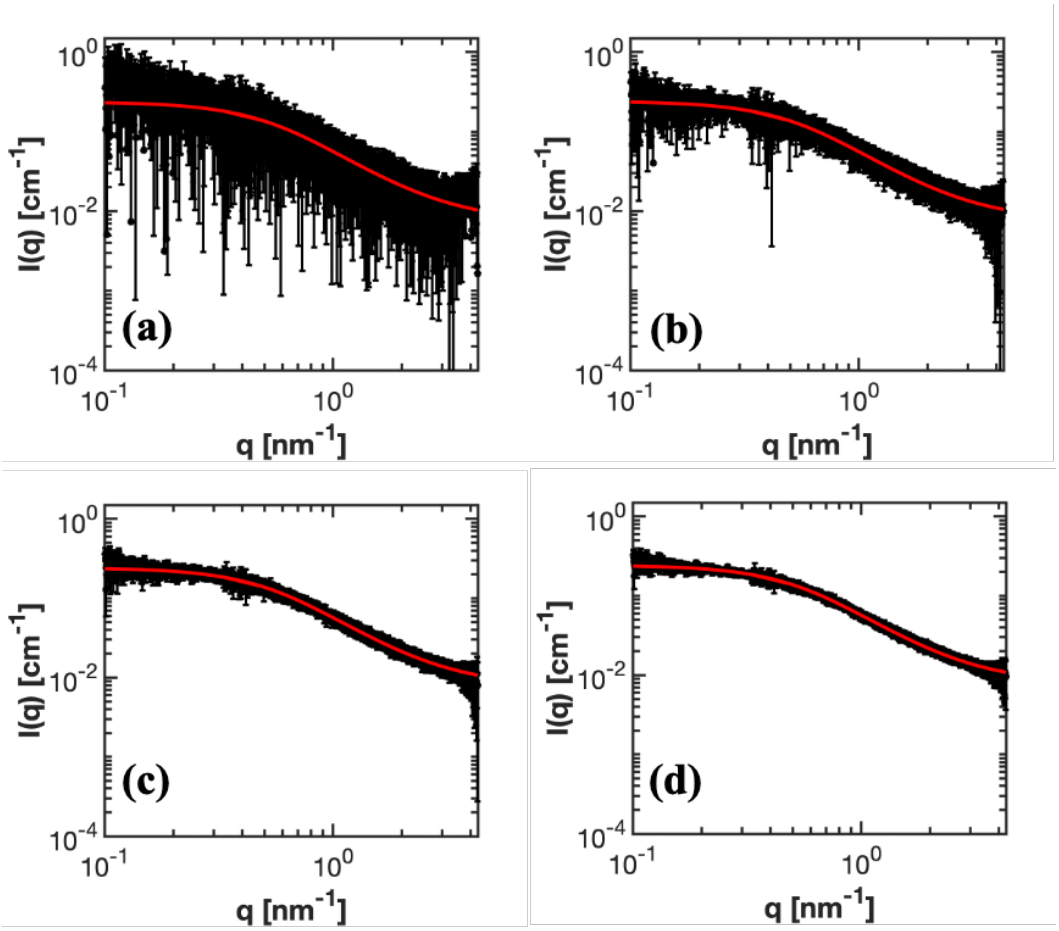


Figure 3-1. SANS intensity curves for 16mM PEG solution in deuterated DMF. (a-d) 0.01, 0.05, 0.25, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the Debye model.

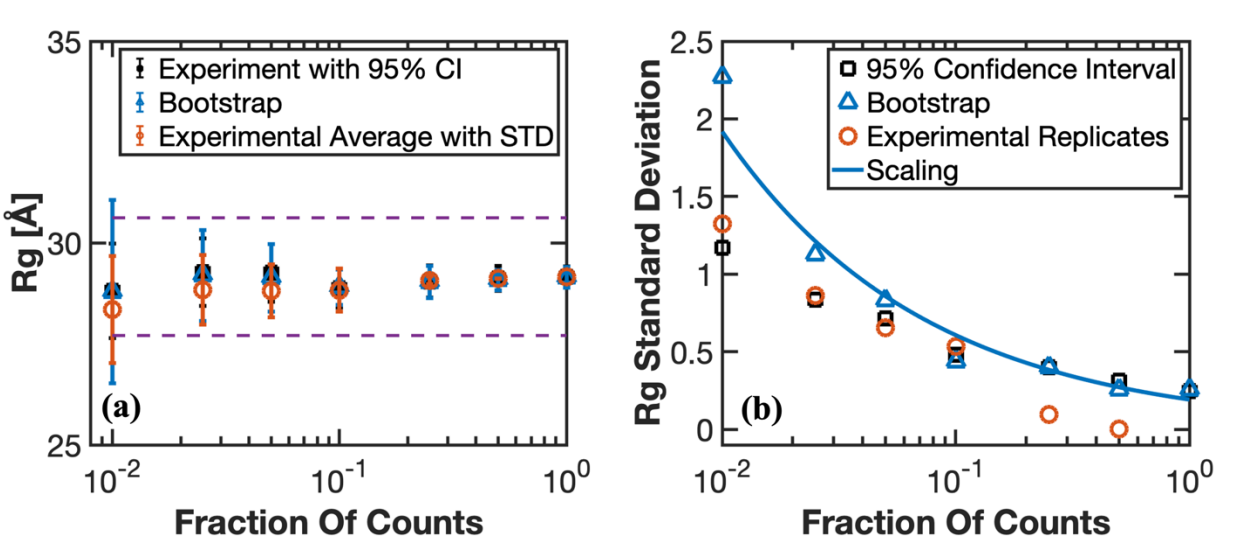


Figure 3-2. (a) Radius of gyration, R_g as a function of the fraction of total counts. The black square data points show the R_g values from one experimental dataset and the error bars denote 95% confidence interval. The blue hollow circle shows the mean R_g value fitted from 100 bootstrapping replicates. The error bars denote one standard deviation of the R_g distributions. The orange hollow circles are mean R_g values fitted from averaging the parameters obtained from experimental replicates from time slicing. The dashed blue line represents 5% of the R_g value extracted from the total counts' dataset. (b) The standard deviation of R_g from fitting MC bootstrapping replicates (blue circles), experimental replicates (orange circles), and scaling fitting with respect to bootstrap standard deviation (blue line) as a function of the fraction of total counts.

Parameter estimation analysis at reduced number of counts is further investigated on a block copolymer micelle system which has found many important applications in drug delivery[53], imaging[54], biosensing[55], removing hydrophobic pollutants[56]. The spherical micelle model with polydispersity provides a model for the SANS data from a Pluronic F-127 micellar solution[42]. Figure 3-3 (a-d) shows the 1D scattering curves for Pluronic F-127 with increasing number of

counts fitted to the spherical micelle model to estimate R, the micelle core radius, and R_g, the radius of gyration of the polymer chain in the corona. One hundred random initializations are used for assessing the global convergence of the parameter estimation. The same values of fitted parameters and the weighted least square values were obtained at each fraction of counts as shown in Supplementary Information. Among the scattering models tested, the spherical micelle model with polydispersity contains the largest number of fitting parameters (7 fitting parameters) with numerical integrations and is the most challenging to converge. The fact that all optimizations still converged to the global minimum demonstrates that the fitting algorithm is capable of robust parameter estimation given that the parameters are identifiable from eigenvalue and eigenvector analysis, even in the presence of high data uncertainty.

For Pluronic F-127, the micelle core radius (R) shows close agreement at all fractions of counts from 0.025 to 1 from fitting experimental replicates and MC bootstrapping replicates, and the parameter value R_g, radius of gyration of the polymer chain in the corona, shows agreement as the fraction of counts approaches 0.25. The value of R is 42.26 ± 1.20 Å, and the R_g value is 28.92 ± 2.78 Å from fitting the SANS data at the full number of counts, where the error is the margin of error from a 95% confidence interval. The value of R for all fraction of counts except for 0.01 remains within 5% of the full counts as shown in Figure 3-4(a). As shown in Figure 3-4(b), R_g remains 10% of the full counts starting at 0.1 fraction of counts. R captures the position of the peak at 1.32 nm^{-1}, making the fitting parameter more robust against noise than R_g. R_g displays larger error even with fitting experimental replicates. Therefore, for the Pluronic F-127 block copolymer micelle, the number of counts can be reduced by a factor of four while still achieving good parameter estimation for both R and R_g.

For Pluronic F-127 fitted to the spherical micelle model, MC bootstrapping and margin of error from 95% confidence intervals can either underestimate or overestimate the experimental uncertainty depending on the parameters of interest and the fraction of counts. MC bootstrapping estimates the experimental uncertainty with greater similarity for parameter R to experimental time-slicing replicates compared to the margin of error from 95% confidence intervals at lower count values, while at higher count values all estimators become comparable. On the contrary, for parameter R_g, the margin of error from 95% confidence intervals is closer to experimental uncertainty values at lower fraction of counts.

Similar to the uncertainty scaling of R_g in 16mM PEG solution, R and R_g in the spherical micelle also follow a $\frac{1}{\sqrt{a}}$ scaling law when fitted to Pluronic F-127 data as shown in Figure 3-4(c-d), where a is the fraction of counts. The scaling law can reasonably capture the trend of decrease in parameter uncertainties associated with experimental replicates and MC bootstrapping replicates, allowing users to quickly estimate the optimal counts at the beamline.

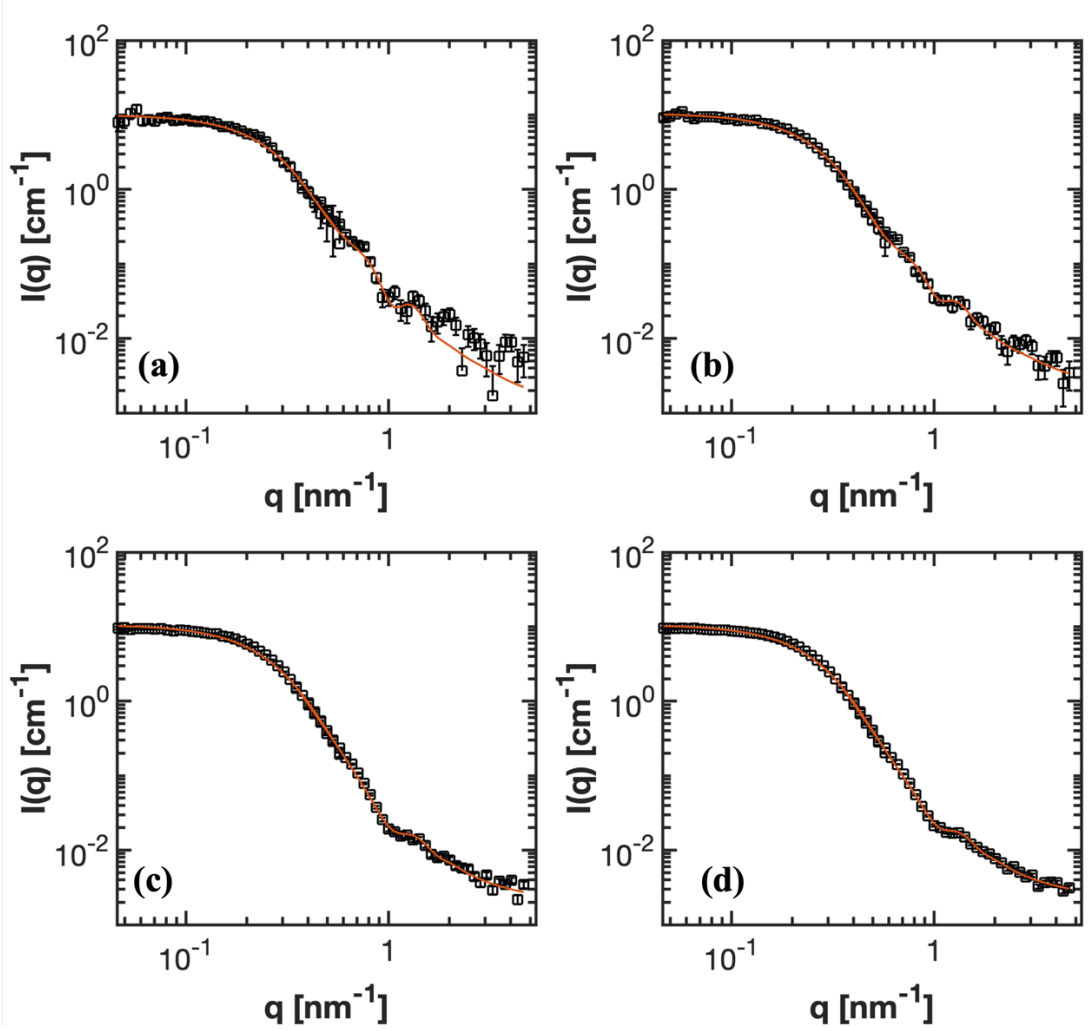


Figure 3-3. SANS intensity curves for Pluronic F-127 in deuterated water. (a-d) 0.01, 0.05, 0.25, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the spherical micelle model[42].

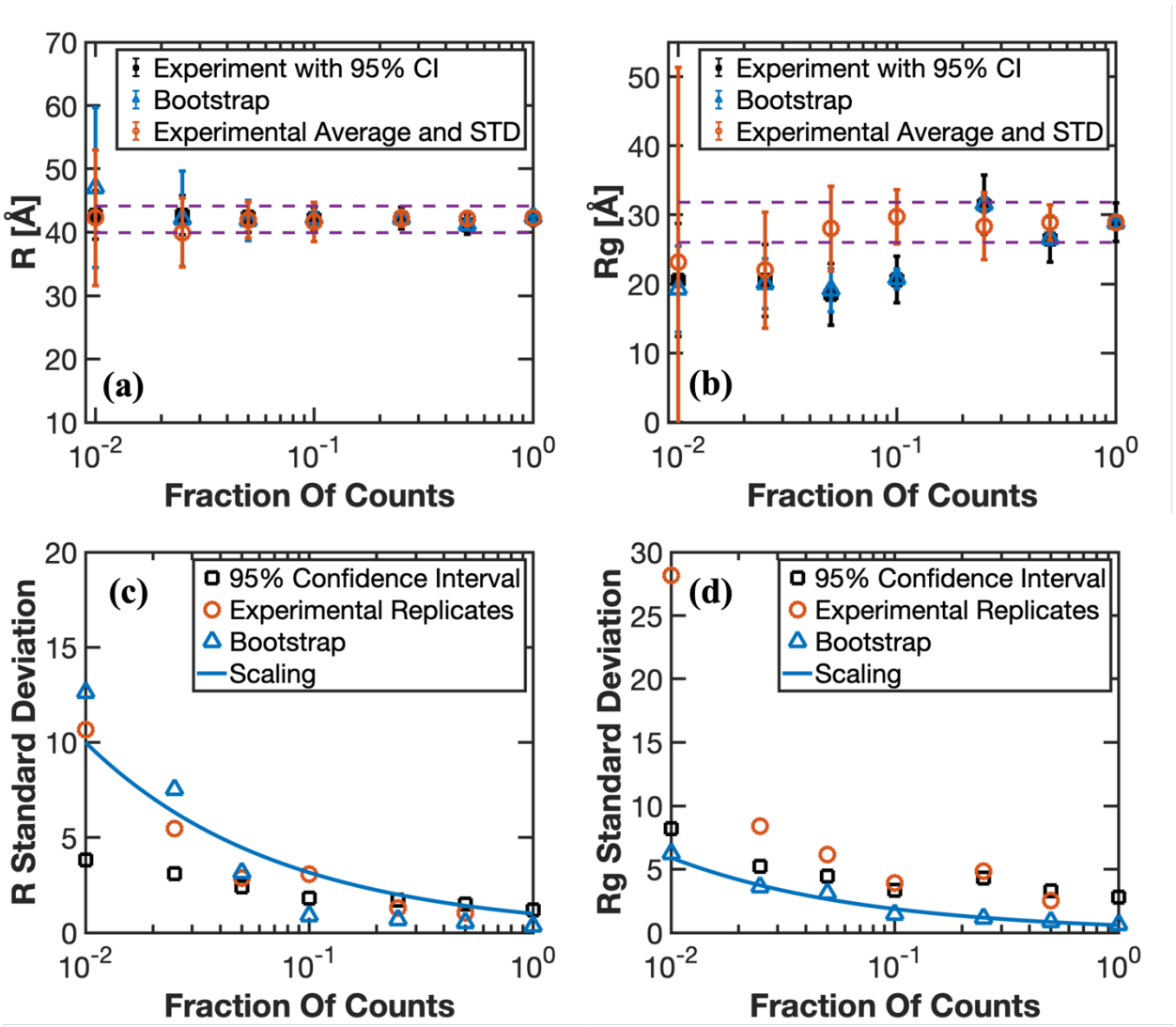


Figure 3-4. (a) Micelle radius (R) as a function of the fraction of total counts. The error bars are 95% confidence intervals from the curve fitting. The dashed line represents 5% of the R value extracted from the total counts' dataset. (b) Radius of gyration (R_g) as a function of the fraction of total counts. The error bars are 95% confidence intervals from the curve fitting. The dashed yellow line represents 10% of the R_g value extracted from the total counts' dataset. (c) The standard deviation of R from fitting MC bootstrapping replicates (blue circles), experimental replicates (orange circles), and scaling fitting (blue line) as a function of the fraction of total counts. (d) The standard deviation of R_g from fitting MC bootstrapping replicates (blue circles), experimental

replicates (orange circles), and scaling fit with respect to bootstrap standard deviation (blue line) as a function of the fraction of total counts.

This parameter estimation analysis enables systematic optimization of neutron counts in SANS measurements based on a user-defined certainty level for the parameters of interests. By performing a brief initial acquisition (~5000 counts) and fitting the data to relevant model functions, parameter values can be estimated, and parameter uncertainties assessed via MC bootstrapping. Leveraging the scaling relationship of parameter uncertainty with neutron count number ($uncertainty \sim \frac{1}{\sqrt{neutron\ count\ number}}$), the algorithm quickly predicts the optimal count required to a user-defined certainty level, such as 5% of the parameter values, providing systematic optimization of valuable SANS beamtime while maintaining desired signal to noise ratio.

This algorithm for parameter estimation has two limitations: (1) computational and (2) requires prior knowledge of the structural model. Computationally, the algorithm requires online reduction of the SANS data, applying MC bootstrapping and model fitting for the short SANS scans to predict additional measurement time/counts needed for the experiment. MC bootstrapping and model fitting take less than 20 seconds to compute on average, which is smaller than the time scale of measurements. Therefore, online implementation should be possible. Second, if the structural model is unknown, this algorithm cannot be applied for parameter estimation. In this case, the user is faced instead with the model differentiation task described below and would use that approach instead.

3.4.2 Parameter Estimation With Biased Estimators

Protein hydrogels "P4", made of molecules that consist of four rodlike associating coiled-coil domains ("P") linked by flexible strands ("C_{10}"), present a more complex parameter estimation challenge but still demonstrates good parameter estimation with a reduced number of counts. Above the overlap concentration (~5% w/v), the protein forms an unentangled physical gel held by coiled-coil association[57]. Figure 3-5(a-d) shows the 1D scattering pattern as the number of counts increases for P4 protein hydrogel fitted to the broad peak model to capture the structure of the protein network. From fitting the total counts scattering curves of P4, the correlation lengths (ξ), which captures the broadness of the peak, is 2.8 nm with 0.14 nm margin of error from 95% confidence interval and 0.04 nm error from MC bootstrapping. The peak component wave vector (q_0) is 0.25 nm^{-1} with 0.026 nm^{-1} margin of error from 95% confidence interval and 0.00298 nm^{-1} error from MC bootstrapping. As shown in Figure 3-6(a-b), the parameter estimation of P4 requires more counts than the Debye model to achieve parameter estimation within 5% error tolerance. q_0 achieved parameter estimation within 5% of the error tolerance for all fractions of counts. However, ξ requires fraction of counts to 0.25 to produce correlation length within 5% of the error tolerance for ξ. Because the nonlinear model is more complex and has more fitting parameters compared to Debye model, the margin of error of 95% confidence interval associated with ξ is 0.14 nm, which is the same magnitude as the 5% of ξ fitted from the full counts SANS curve. Therefore, for fitting of more complicated models the number of counts or measurement time can be still reduced while achieving good parameter estimation, but the level of reduction is smaller than for simpler models.

The q_0 values obtained from experimental data and bootstrapping replicates shows close agreement starting at 0.1 fraction of counts; however, the mean of ξ from MC bootstrapping constantly exceeds the value obtained from the experimental replicate even at full number of counts as shown in Figure 3-6(a-b). Two factors may contribute to this discrepancy: (1) the broad

peak model cannot capture the physics of the protein hydrogel, and (2) MC bootstrapping provides a biased estimator for ξ. The diagonal entries of the Fisher Information Matrix represent how precise the estimates are. The smaller the entry is, the harder it is to identify for the corresponding parameter. In this model, all the eigenvalues of the fisher information matrix are nonzero, where the smallest eigenvalue of the fisher information matrix is 3.489. The eigenvalues for the remaining parameters are much greater than the smallest eigenvalue on the order ranging from 10^3 to 10^{13}, indicating good identifiability. This is also supported by the full convergence of multiple initializations of the parameter estimation. One hundred random initializations are used for assessing the global convergence of the parameter estimation in the broad peak model. The same values of fitted parameters and the weighted least square values were obtained at each fraction of counts as shown in Supplementary Information. The eigenvector corresponding to the smallest eigenvalue is $[0.0028 \quad 0.0394 \quad 0.00064 \quad 0.9992]^T$. This indicates that the hardest identifiable direction is mostly in the ξ parameter that corresponds to the largest entry 0.992.

Literature finds that the broad peak model often fails to generate a good fit to experimental data, which they attribute to the broad peak model's inability to model the network heterogeneity in protein hydrogel systems[27, 58]. Our results for the P4 hydrogel systems are broadly consistent with these findings. The distribution of the weighed residuals from fitting the experimental data and bootstrapping data are compared as outlined in the methods section. As shown in Figure 3-7, the histogram of weighed residuals from fitting the experimental data has its largest bin to the left of 0. The weighted residual from fitting bootstrapping replicates is less skewed than that of experimental data. Quantitatively, the skewness of the weighted residual distribution, which are 0.47 and 0.27 for experimental data and bootstrapping data, respectively. MC bootstrapping cannot capture the higher degree of skewness that was present in the weighted residual in fitting

experimental data. When the model cannot adequately capture the data, the parameter mean from MC bootstrapping can fail to agree even at high fraction of counts.

The weighted least square estimator is a biased estimator for the broad peak model, which can also cause disagreement of parameter values from fitting experimental data and bootstrapping data at full number of counts. The weighted least square estimator is an unbiased estimator for linear models when the residuals follow a Gaussian distribution[59]. However, as shown in Table 3-1, from the bias testing, the expectation value $E[\xi]$ is 17% higher than the ξ value used for generating the simulated SANS curve, which is consistent with the bias observed in Figure 3-6(b). For all the fitting parameters in P4, there are various degrees of bias associated with the weighted nonlinear least square estimator, and even Gaussian noise based on standard deviation ΔI from SANS reduction file can deviate the fitting parameters from the ground truth. Bias from parameter estimation is intrinsic to the estimator and the model. Under the covariance analysis of broad peak model in Figure A-14 and Table A-1, all parameters from the model will be biased to different degrees with unsuited estimator. When the most biased parameter A is fixed to the value obtained at full counts, the absolute values of bias in the other parameters are reduced to various degrees. The largest bias reduction occurs in the parameter q_0, which the bias decreases from 36.42% to 17.71%. Table A-2 summarizes the degrees of bias for all parameters. When certain parameters can be known from either literature, complementary experimental characterization or simulations, the effect of bias in the weighted nonlinear least squares estimator can be greatly reduced. When the estimator is biased for the best model for certain SANS data, error quantification should include the degree of bias and the standard deviation of parameter values from fitting simulated SANS data with Gaussian noise.

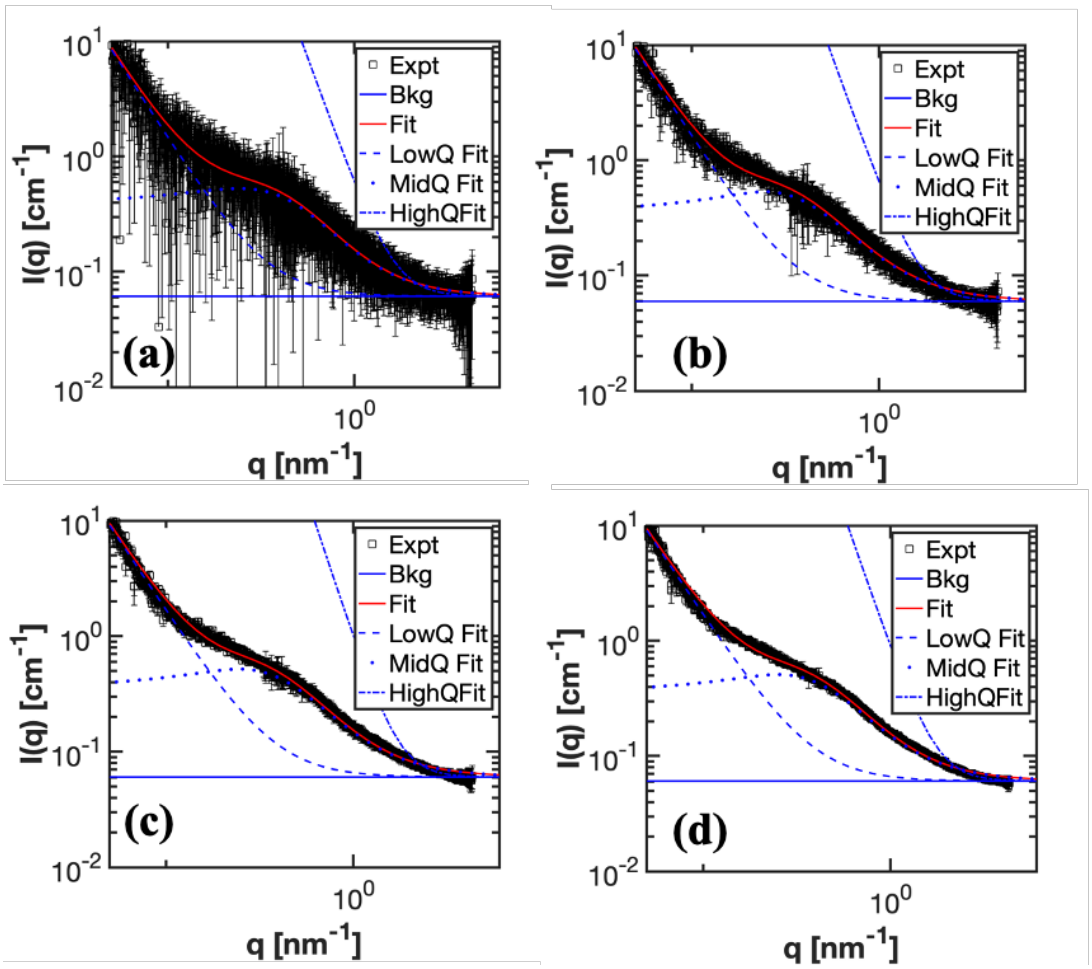


Figure 3-5. SANS intensity curves for P4 hydrogel in deuterated 100mM phosphate buffer, pD = 7.6. (a-d) 0.01, 0.05, 0.25, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the broad peak model.

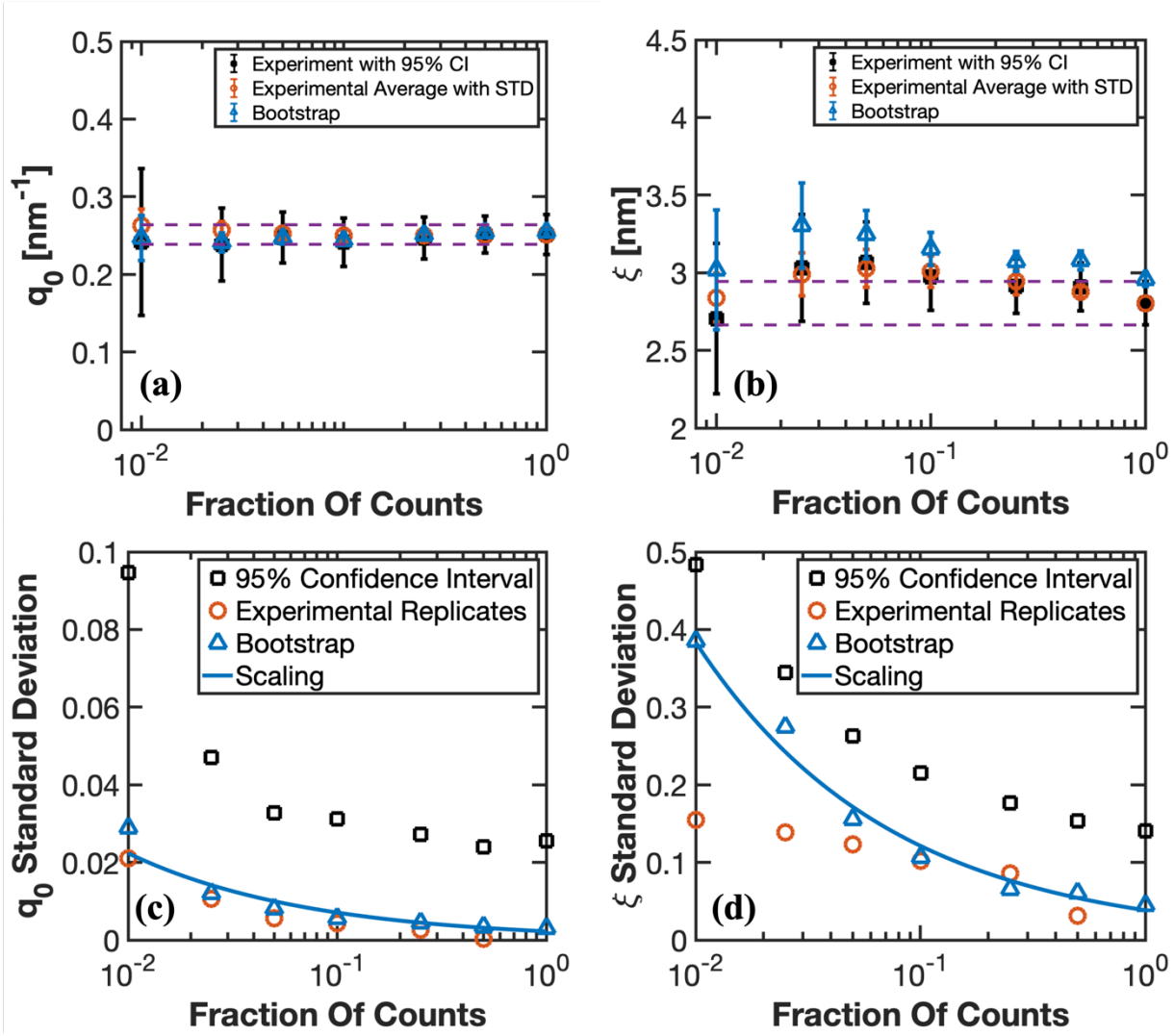


Figure 3-6. (a) Peak component wavevector (q_0) as a function of the fraction of total counts. The error bars are 95% confidence intervals from the curve fitting. The dashed line represents 5% of the value extracted from the total counts' dataset. (b) Correlation length (ξ) as a function of the fraction of total counts. The error bars are 95% confidence intervals from the curve fitting. The dashed line represents 5% of the ξ value extracted from the total counts' dataset. (c) Standard deviation of q_0 as a function of the fraction of total counts. (d) Standard deviation of ξ as a function of the fraction of counts. (c-d) Black squares represent uncertainty values from margin of error. Blue triangles represent uncertainty values from MC bootstrapping replicates. Orange circles

represent uncertainty values from experimental replicates (orange circles). The blue line represents scaling of uncertainty values from MC bootstrapping replicates.

For both q_0 and ξ, MC bootstrapping matches better with experimental uncertainties than margin of error from 95% confidence interval despite the biased estimator as shown in Figure 3-6(c-d). Similar to 16mM PEG solution and Pluronic F-127, both MC bootstrapping and margin of error in general predict parameter uncertainties for q_0 and ξ that are higher than experimental uncertainties. The P4 protein hydrogel's parameters show similar scaling of uncertainties when fitted to the broad peak model even though the estimators are biased. The standard deviation of q_0 is higher than the scaling prediction at 0.01 fraction of counts, as indicated by the 238% and 0.35% higher variances at low q and high q, respectively, in intensity for the 0.01 fraction of counts. For other fractions of counts, the scaling $\frac{1}{\sqrt{a}}$ matches well with the standard deviation of q_0 obtained from MC bootstrapping. The parameter scaling of ξ matches well with bootstrapping uncertainties at all fractions of counts. Therefore, MC bootstrapping remains an effective means for estimating minimum experimental times to determine a parameter within a given error tolerance even for models for which the estimator is significantly biased.

This accelerated SANS algorithm can present significant time savings at the beamline when the structural model is known *a priori*. MC bootstrapping takes less than 20 seconds on average on a laptop to compute. For the three samples used for the study, 16mM PEG solution was measured for 52 mins, P4 hydrogel was measured 29 mins, and Pluronic F-127 was measured for 10 mins to reach the desired neutron counts. Since measurement time scales linearly with neutron counts, measurement time can be reduced to 0.52 mins for 16mM PEG solution, 15 mins for P4 hydrogel, and 2.5 mins for Pluronic F-127 while obtaining good parameter estimation. In the case

of contrast matching, the algorithm can still be applied for optimizing beamtime if the structural model is known, but longer counting times may be required to account for incoherent background.

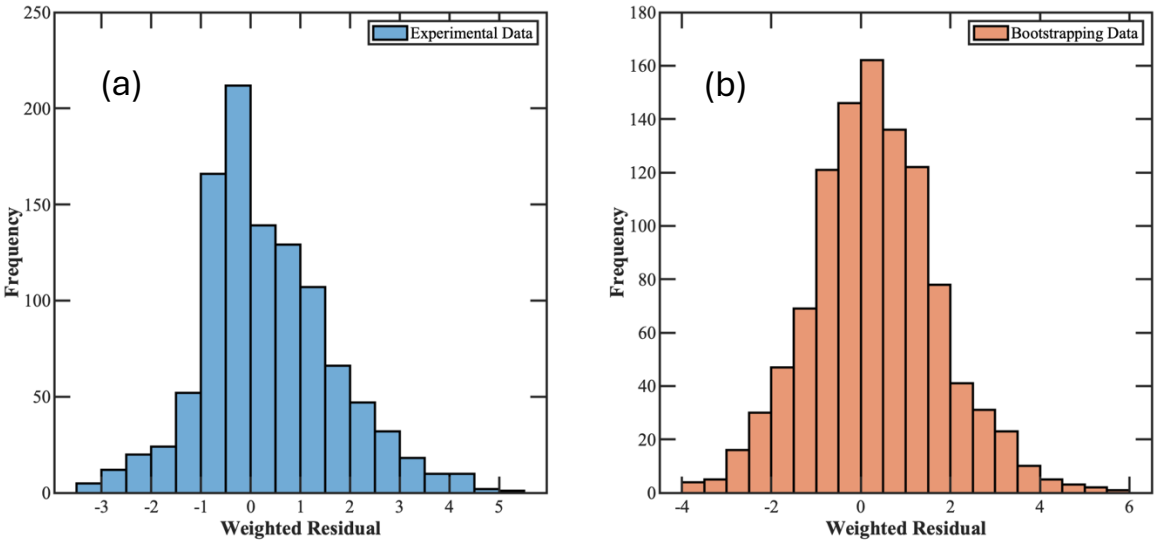


Figure 3-7. Histogram comparison from plotting the weighted residual from fitting the broad peak model to (a) P4 experimental data from full counts and (b) one MC bootstrapping replicate simulated from the same P4 experimental data.

Table 3-1. Comparison between true parameter X_{true} from fitting the P4 experimental data and the expectation value from fitting simulated data $E[X]$ to the broad peak model. The bias is the percent difference between X_{true} and $E[X]$.

Parameter X	X_{true}	$E[X]$	Bias (%)
A	$2.00 * 10^{-5}$	$9.04 * 10^{-5}$	350

n	2.46	2.17	-11.5
C	0.45	0.36	-19.62
m_0	1.87	1.66	-11.17
q_0	0.025	0.0343	36.42
ξ	28.04	32.79	16.94

3.4.3 Model Differentiation

In addition to parameter estimation, SANS can be used to test different structural hypotheses about a sample by comparing the goodness of fit for candidate structural models when the model is unavailable a priori to the experiments. To test whether a reduced number of counts can reliably perform model differentiation, competing SANS models are fitted to the SANS scattering data, and the AIC and BIC for each model are computed at different fractions of counts. The best model is selected based on the lowest AIC or BIC value at each fraction of counts. The protein gel P4 and the Pluronic F-127 micelles are used for model differentiation because those classes of materials can be fit to various SANS models.

AIC and BIC can differentiate the best model even at reduced fractions of counts from the competing models. The candidate models for the P4 protein gel are the broad peak model[30], the fine scale polymer gel[38, 60], the Gauss-Lorentz Gel model[61], the Debye-Bueche model[62], and the Ornstein-Zernike and squared Lorentz model[39]. All models can be used to describe polymer

network systems depending on the different nanostructures. As shown in Figure 3-8, the magnitude of AIC and BIC for the same model are the same across different numbers of counts because the maximum likelihood estimates dominate the value of AIC and BIC in comparison to the number of parameters in the models. The Debye-Bueche and the Ornstein-Zernike and squared Lorentz models have consistently higher AIC values regardless of the fraction of counts, which indicates that the models are unsuited to describe the scattering intensities of P4 protein hydrogel as shown in Figure 3-9 (d-e) and Appendix A. The constant AIC originates from the large residual sum of squares from the lower q region where the scattering intensities of the models plateau but the P4 scattering intensity is decreasing.

The broad peak model, the fine scale polymer gel model, and the Gauss Lorentz Gel model all fitted similarly well for the lower number of counts from 0.01 to 0.025 fractions of counts. The broad peak model has the lowest AIC and BIC values among all the models. As the fraction of counts increases to 0.05 and above, the broad peak model yields significantly lower AIC and BIC values. At 0.25 fraction of counts, the AIC and BIC fine scale polymer gel model and the Gauss Lorentz Gel model starts to reach relatively constant values. However, the AIC or BIC values for the broad peak model continue to decrease. The fitting results at full number of counts for each model are shown in Figure 3-9. From inspecting the fitting curves, the broad peak model can capture the low q and mid q region better than both the fine scale polymer gel model and the Gauss Lorentz model, which is consistent with the intuitive notion of goodness of fit. The best model from model differentiation agrees with the choice of model from previous published SANS fitting model for P4 protein hydrogel[27]. From comparing the AIC and BIC values, the broad peak model is the most suited model for P4 even at very low number of counts.

MC bootstrapping for model differentiation suggests that identifying the best model requires sufficient neutron counts, as shown in Figure A-7(b) in Appendix A, which differs from experimental data for model differentiation. This discrepancy arises from the larger ΔI used in MC bootstrapping at lower fraction of counts, evidenced by the 238% and 0.35% higher variances of ΔI than $Var[I]$ at low q and high q, respectively, in intensity for the 0.01 fraction of counts. High ΔI values used for simulating MC bootstrapping replicates introduces large noise to simulated data, increasing the values of AIC and BIC in the broad peak model slightly above the fine-scale polymer gel model at lower fraction of counts. As the fraction of counts increases and ΔI decreases, the SANS curve from MC bootstrapping becomes smoother and ΔI value approaches the true standard deviation in intensity, the SANS curve from bootstrapping smooths, and the broad peak model becomes the best fit.

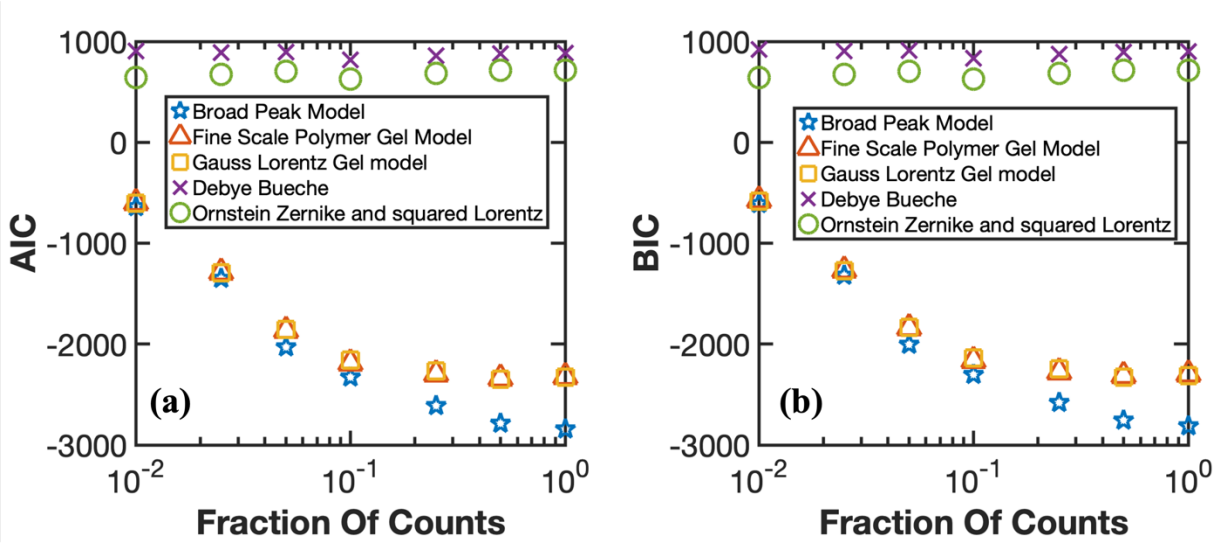


Figure 3-8. P4 protein hydrogel model differentiation. (a) AIC as a function of fraction of counts for the candidate models. (b) BIC as a function of fraction of counts for the candidate models.

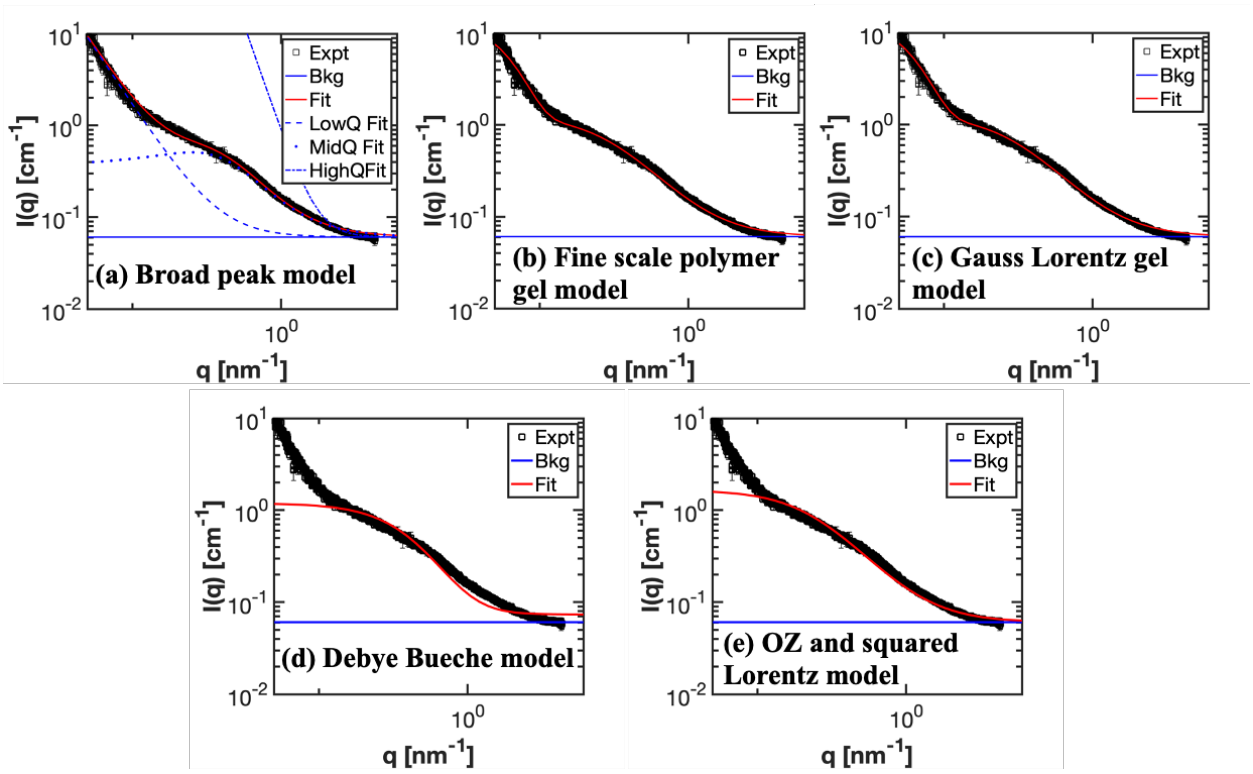


Figure 3-9. SANS fitting results of P4 protein hydrogel for full number of counts. (a) Broad peak model. (b) Fine scale polymer gel model. (c) Gauss Lorentz gel model. (d) Debye Bueche model. (e) Ornstein Zernike and squared Lorentz model.

The analysis of model differentiation on Pluronic F-127 has similar result as P4 protein hydrogel, which shows that AIC or BIC can differentiate models even at a low fraction of counts as shown in Figure 3-10. The candidate models for the Pluronic F-127 are the spherical micelle model with polydispersity[42, 46], the sphere model[63], and the fuzzy sphere model[64]. Since the sample is dilute, the SANS data were fitted to only the form factor in this model differentiation. All models have spherical shapes, and the SANS data should identify micelle formation between spherical shape models with the SANS scattering pattern. The various scattering models for fitting experimental data are shown in Figure A-12 and A-13 in the supporting information. The spherical micelle model with polydispersity has the lowest AIC or BIC across all fractions of counts. Starting at 0.025 fraction of counts, the slope of the AIC or BIC of the spherical micelle model is much steeper than those of the competing models. In this case, the AIC or BIC of the competing models remains relatively constant regardless of the fraction of counts. The AIC or BIC of the spherical micelle model with polydispersity also reaches relatively constant at 0.1 fraction of counts. In this example, AIC and BIC successfully identified micelle formation at 0.1 fraction of counts from spherical model candidates at reduced number of counts, indicating that relatively few counts are required in order to identify the most suitable model.

Model differentiation complements, rather than replaces, model development—it provides a systematic method to assess newly proposed scattering functions alongside established models using optimized neutron counts. This approach enables broader validation of novel models across diverse experimental systems, ultimately accelerating the refinement and adoption of physical models in the SANS community.

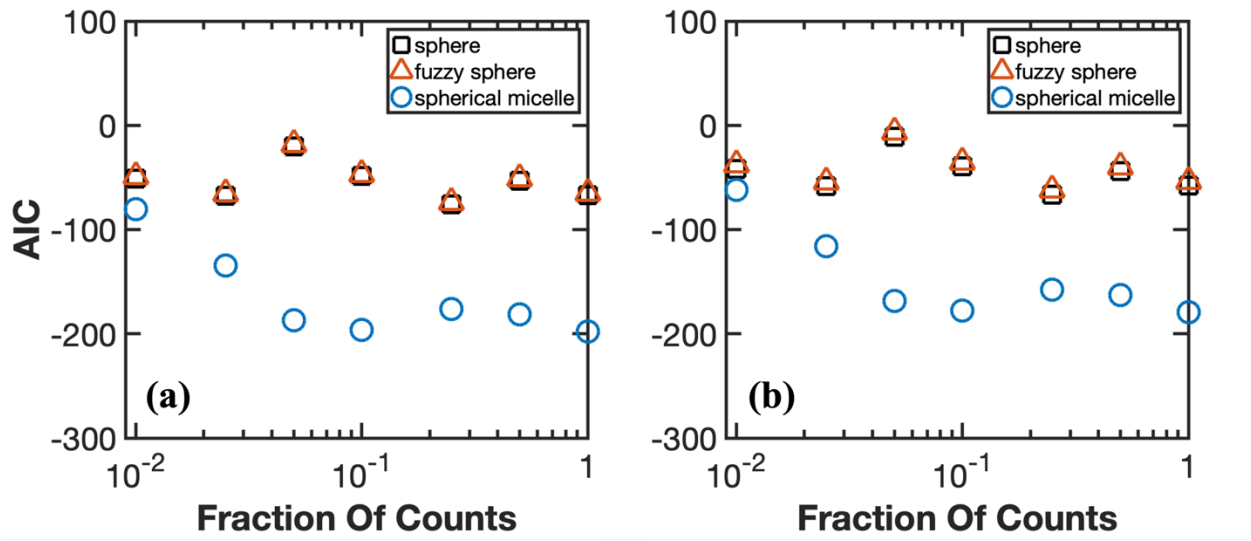


Figure 3-10. Pluronic F-127 model differentiation. (a) AIC as a function of fraction of counts for the candidate models. (b) BIC as a function of fraction of counts for the candidate models.

From the above analysis, SANS users can reliably identify best structural models even at relatively low fraction of counts. An accelerated model differentiation algorithm is proposed to determine optimal neutron counts for data acquisition. Users first select suitable scattering model functions that can capture the targeted nanostructure. SANS measurements with relatively few counts (for example, 5k, 10k, 20k counts) are then acquired, and weighed nonlinear least squares is used to fit each of the candidate model functions on-line, with corresponding AIC and BIC values. The computation of AIC or BIC values for each model allows users to compare both the values and their rate of decrease with increasing neutron counts. The best model is chosen, and data collection for the purpose of model differentiation is stopped, based on the following heuristic: the best model should both minimize AIC or BIC among candidate models for a fixed counts and should have the steepest decrease rate, so that the absolute difference between the best model's AIC and other candidate AIC values are increasing. If the decrease rate is insufficient, additional

counts should be acquired until the best model shows a significant AIC or BIC reduction and competing models plateau in AIC or BIC values.

3.5 Conclusion

This work proposes an accelerated SANS workflow that uses short SANS acquisitions to predict the optimal number of counts for parameter estimation and model differentiation. For parameter estimation, a short initial acquisition may be used to provide estimates on parameters and uncertainties using MC bootstrapping. Based on this estimate and the scaling of parameter uncertainties $\frac{1}{\sqrt{a}}$, where a is the fraction of counts, the minimum number of counts required to achieve a given error in the parameter may be determined at the beamline. Three representative polymer materials were used to demonstrate that parameter estimation can be achieved with reduced number of counts to within a targeted accuracy. Low error for structural parameter estimation can be obtained using 0.01, 0.25, and 0.5 fraction of the full total counts for the polymer in solution, the spherical micelle, and the associative protein hydrogel, respectively. By leveraging a method that estimates the minimum number of counts necessary to achieve a desired level of uncertainty in parameter estimation, this work addresses the critical challenge of balancing experimental throughput with data quality in SANS experiments.

A key finding of the work was the examination of the bias associated with estimator used in SANS model fitting. The weighted least squares estimator was found to introduce bias when applied to the broad peak model used for the P4 protein hydrogel. Fixing the most biased parameters can substantially reduce the fitting bias in the other parameters. The findings highlight the importance of carefully selecting appropriate estimators depending on the complexity of the model and reducing biases by determining values of certain parameters from either literature or complimentary characterizations or simulations.

For model differentiation, AIC and BIC can reliably differentiate between competing structural models even with reduced number of counts. P4 protein hydrogel and Pluronic F-127 micelles were used for model differentiation task because they can be described by multiple competing models. For both the P4 protein hydrogel and Pluronic F-127 micelles, the broad peak model and spherical micelle model, respectively, were identified as the best-fit models across various fractions of counts. The AIC and BIC values for the best models are consistently the lowest and decrease more rapidly as the count fractions increased, making them distinguishable even with less neutron counts. The ability to differentiate between competing models under such conditions is particularly valuable for high-throughput screening of material libraries, where rapid and accurate structural characterization is essential.

3.6 Acknowledgements

We thank the department of energy (Grant: DE-SC0007106) for supporting this research and the support of Oakridge National Laboratory (ORNL) for the neutron scattering facilities in this work. This research used resources at the High Flux Isotope Reactor, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory. We acknowledge Dr. Lilin He for the experimental assistance and data analysis at ORNL. We thank Dr. Ameya Rao and Dr. Haley Beech in assistance of the material preparation, Dr. Helen Yao for the curve fitting MATLAB code to selected models, and Dr. Andrew Salmon for the advice on statistical analysis.

3.7 References

1. Roe, R.-J., *Methods of X-ray and neutron scattering in polymer science*. Oxford University Press on Demand: 2000.

2. Helgeson, M. E.; Moran, S. E.; An, H. Z.; Doyle, P. S., Mesoporous organohydrogels from thermogelling photocrosslinkable nanoemulsions. *Nature materials* **2012,** *11* (4), 344-352.

3. Yearley, Eric J.; Zarraga, Isidro E.; Shire, Steven J.; Scherer, Thomas M.; Gokarn, Y.; Wagner, Norman J.; Liu, Y., Small-Angle Neutron Scattering Characterization of Monoclonal

Antibody Conformations and Interactions at High Concentrations. *Biophysical Journal* **2013,** *105* (3), 720-731.

4. Baym, G., Direct Calculation of Electronic Properties of Metals from Neutron Scattering Data. *Physical Review* **1964,** *135* (6A), A1691-A1692.

5. Wright, A. C.; Clare, A. G.; Grimley, D. I.; Sinclair, R. N., Neutron scattering studies of network glasses. *Journal of Non-Crystalline Solids* **1989,** *112* (1-3), 33-47.

6. Allen, A. J., Characterization of ceramics by x-ray and neutron small-angle scattering. *Journal of the American Ceramic Society* **2005,** *88* (6), 1367-1381.

7. Melnichenko, Y. B.; Wignall, G. D., Small-angle neutron scattering in materials science: Recent practical applications. *Journal of Applied Physics* **2007,** *102* (2), 3.

8. De Gennes, P.-G.; Gennes, P.-G., *Scaling concepts in polymer physics*. Cornell university press: 1979.

9. Doi, M.; Edwards, S. F.; Edwards, S. F., *The theory of polymer dynamics*. oxford university press: 1988; Vol. 73.

10. Jeffries, C. M.; Ilavsky, J.; Martel, A.; Hinrichs, S.; Meyer, A.; Pedersen, J. S.; Sokolova, A. V.; Svergun, D. I., Small-angle X-ray and neutron scattering. *Nature Reviews Methods Primers* **2021,** *1* (1), 1-39.

11. Li, L.; Jakowski, J.; Do, C.; Hong, K., Deuteration and Polymers: Rich History with Great Potential. *Macromolecules* **2021,** *54* (8), 3555-3584.

12. Chang, M.-C.; Wei, Y.; Chen, W.-R.; Do, C., Deep learning-based super-resolution for small-angle neutron scattering data: attempt to accelerate experimental workflow. *MRS Communications* **2020,** *10* (1), 11-17.

13. Raccuglia, P.; Elbert, K. C.; Adler, P. D.; Falk, C.; Wenny, M. B.; Mollo, A.; Zeller, M.; Friedler, S. A.; Schrier, J.; Norquist, A. J., Machine-learning-assisted materials discovery using failed experiments. *Nature* **2016,** *533* (7601), 73-76.

14. Chen, Z.; Andrejevic, N.; Drucker, N. C.; Nguyen, T.; Xian, R. P.; Smidt, T.; Wang, Y.; Ernstorfer, R.; Tennant, D. A.; Chan, M.; Li, M., Machine learning on neutron and x-ray scattering and spectroscopies. *Chemical Physics Reviews* **2021,** *2* (3), 031301.

15. Wang, Z.; Sun, Z.; Yin, H.; Liu, X.; Wang, J.; Zhao, H.; Pang, C. H.; Wu, T.; Li, S.; Yin, Z.; Yu, X.-F., Data-Driven Materials Innovation and Applications. *Advanced Materials* **2022,** *34* (36), 2104113.

16. Hoogenboom, R.; Meier, M. A.; Schubert, U. S., Combinatorial methods, automated synthesis and high-throughput screening in polymer research: past and present. *Macromolecular rapid communications* **2003,** *24* (1), 15-32.

17. Trobe, M.; Burke, M. D., The molecular industrial revolution: automated synthesis of small molecules. *Angewandte Chemie International Edition* **2018,** *57* (16), 4192-4214.

18. Zhang, C.; Bates, M. W.; Geng, Z.; Levi, A. E.; Vigil, D.; Barbon, S. M.; Loman, T.; Delaney, K. T.; Fredrickson, G. H.; Bates, C. M.; Whittaker, A. K.; Hawker, C. J., Rapid

Generation of Block Copolymer Libraries Using Automated Chromatographic Separation. *Journal of the American Chemical Society* **2020,** *142* (21), 9843-9849.

19. Heller, W. T.; Hetrick, J.; Bilheux, J.; Calvo, J. M. B.; Chen, W.-R.; DeBeer-Schmitt, L.; Do, C.; Doucet, M.; Fitzsimmons, M. R.; Godoy, W. F.; Granroth, G. E.; Hahn, S.; He, L.; Islam, F.; Lin, J.; Littrell, K. C.; McDonnell, M.; McGaha, J.; Peterson, P. F.; Pingali, S. V.; Qian, S.; Savici, A. T.; Shang, Y.; Stanley, C. B.; Urban, V. S.; Whitfield, R. E.; Zhang, C.; Zhou, W.; Billings, J. J.; Cuneo, M. J.; Leal, R. M. F.; Wang, T.; Wu, B., drtsans: The data reduction toolkit for small-angle neutron scattering at Oak Ridge National Laboratory. *SoftwareX* **2022,** *19*, 101101.

20. ShuoQian; Heller, W.; Chen, W.-R.; Christianson, A.; Do, C.; Wang, Y.; Lin, J. Y. Y.; Huegle, T.; Jiang, C.; Boone, C.; Hart, C.; Graves, V., CENTAUR—The small- and wide-angle neutron scattering diffractometer/spectrometer for the Second Target Station of the Spallation Neutron Source. *Review of Scientific Instruments* **2022,** *93* (7), 075104.

21. Steinhart, M.; Pleštic, J., Possible improvements in the precision and accuracy of small-angle X-ray scattering measurements. *Journal of applied crystallography* **1993,** *26* (4), 591-601.

22. Saito, K.; Yano, M.; Hino, H.; Shoji, T.; Asahara, A.; Morita, H.; Mitsumata, C.; Kohlbrecher, J.; Ono, K., Accelerating small-angle scattering experiments on anisotropic samples using kernel density estimation. *Sci Rep* **2019,** *9* (1), 1526.

23. Kanazawa, T.; Asahara, A.; Morita, H., Accelerating small-angle scattering experiments with simulation-based machine learning. *Journal of Physics: Materials* **2020,** *3* (1), 015001.

24. Beech, H. K.; Johnson, J. A.; Olsen, B. D., Conformation of Network Strands in Polymer Gels. *ACS Macro Letters* **2023,** *12* (3), 325-330.

25. Zhong, M.; Wang, R.; Kawamoto, K.; Olsen, B. D.; Johnson, J. A., Quantifying the impact of molecular defects on polymer network elasticity. *Science* **2016,** *353* (6305), 1264-1268.

26. Glassman, M. J.; Chan, J.; Olsen, B. D., Reinforcement of Shear Thinning Protein Hydrogels by Responsive Block Copolymer Self-Assembly. *Adv Funct Mater* **2013,** *23* (9), 1182-1193.

27. Rao, A.; Yao, H.; Olsen, B. D., Bridging dynamic regimes of segmental relaxation and center-of-mass diffusion in associative protein hydrogels. *Physical Review Research* **2020,** *2* (4), 043369.

28. Heller, W. T.; Cuneo, M.; Debeer-Schmitt, L.; Do, C.; He, L.; Heroux, L.; Littrell, K.; Pingali, S. V.; Qian, S.; Stanley, C.; Urban, V. S.; Wu, B.; Bras, W., The suite of small-angle neutron scattering instruments at Oak Ridge National Laboratory. *Journal of Applied Crystallography* **2018,** *51* (2), 242-248.

29. Debye, P., Molecular-weight determination by light scattering. *The Journal of Physical Chemistry* **1947,** *51* (1), 18-32.

30. Horkay, F.; Hammouda, B., Small-angle neutron scattering from typical synthetic and biopolymer solutions. *Colloid and Polymer Science* **2008,** *286* (6), 611-620.

31. Edwards, C. E. R.; Mai, D. J.; Tang, S.; Olsen, B. D., Molecular anisotropy and rearrangement as mechanisms of toughness and extensibility in entangled physical gels. *Physical Review Materials* **2020,** *4* (1), 015602.

32. Hammouda, B.; Ho, D. L.; Kline, S., Insight into clustering in poly (ethylene oxide) solutions. *Macromolecules* **2004,** *37* (18), 6932-6937.

33. Hore, M. J. A.; Hammouda, B.; Li, Y.; Cheng, H., Co-Nonsolvency of Poly(n-isopropylacrylamide) in Deuterated Water/Ethanol Mixtures. *Macromolecules* **2013,** *46* (19), 7894-7901.

34. Hule, R. A.; Nagarkar, R. P.; Hammouda, B.; Schneider, J. P.; Pochan, D. J., Dependence of Self-Assembled Peptide Hydrogel Network Structure on Local Fibril Nanostructure. *Macromolecules* **2009,** *42* (18), 7137-7145.

35. Saffer, E. M.; Lackey, M. A.; Griffin, D. M.; Kishore, S.; Tew, G. N.; Bhatia, S. R., SANS study of highly resilient poly(ethylene glycol) hydrogels. *Soft Matter* **2014,** *10* (12), 1905-1916.

36. Matsunaga, T.; Sakai, T.; Akagi, Y.; Chung, U.-i.; Shibayama, M., SANS and SLS Studies on Tetra-Arm PEG Gels in As-Prepared and Swollen States. *Macromolecules* **2009,** *42* (16), 6245-6252.

37. Kanaya, T.; Ohkura, M.; Kaji, K.; Furusaka, M.; Misawa, M., Structure of Poly(vinyl alcohol) Gels Studied by Wide- and Small-Angle Neutron Scattering. *Macromolecules* **1994,** *27* (20), 5609-5615.

38. Mallam, S.; Horkay, F.; Hecht, A. M.; Rennie, A. R.; Geissler, E., Microscopic and macroscopic thermodynamic observations in swollen poly(dimethylsiloxane) networks. *Macromolecules* **1991,** *24* (2), 543-548.

39. Shibayama, M.; Isono, K.; Okabe, S.; Karino, T.; Nagao, M., SANS study on pressure-induced phase separation of poly (N-isopropylacrylamide) aqueous solutions and gels. *Macromolecules* **2004,** *37* (8), 2909-2918.

40. Hamley, I. W., *Block copolymers in solution: fundamentals and applications*. John Wiley & Sons: 2005.

41. Kim, T.-H.; Han, Y.-S.; Jang, J.-D.; Seong, B.-S., SANS study on self-assembled structures of Pluronic F127 triblock copolymer induced by additives and temperatureThis article will form part of a virtual special issue of the journal, presenting some highlights of the 15th International Small-Angle Scattering Conference (SAS2012). This special issue will be available in early 2014. *Journal of Applied Crystallography* **2014,** *47* (1), 53-59.

42. Pedersen, J. S., Form factors of block copolymer micelles with spherical, ellipsoidal and cylindrical cores. *Journal of Applied Crystallography* **2000,** *33* (3-1), 637-640.

43. Pedersen, J. S.; Gerstenberg, M. C., Scattering Form Factor of Block Copolymer Micelles. *Macromolecules* **1996,** *29* (4), 1363-1365.

44. Pedersen, J. S., Analysis of small-angle scattering data from colloids and polymer solutions: modeling and least-squares fitting. *Advances in Colloid and Interface Science* **1997,** *70*, 171-210.

45. Lin, Y.; Alexandridis, P., Temperature-Dependent Adsorption of Pluronic F127 Block Copolymers onto Carbon Black Particles Dispersed in Aqueous Media. *The Journal of Physical Chemistry B* **2002,** *106* (42), 10834-10844.

46. Manet, S.; Lecchi, A.; Impéror-Clerc, M.; Zholobenko, V.; Durand, D.; Oliveira, C. L. P.; Pedersen, J. S.; Grillo, I.; Meneau, F.; Rochas, C., Structure of Micelles of a Nonionic Block Copolymer Determined by SANS and SAXS. *The Journal of Physical Chemistry B* **2011,** *115* (39), 11318-11329.

47. Seber, G. A. F.; Wild, C. J., *Nonlinear Regression*. John Wiley & Sons: New York, 1989.

48. Pérez, R. N.; Amaro, J.; Arriola, E. R., Bootstrapping the statistical uncertainties of NN scattering data. *Physics Letters B* **2014,** *738*, 155-159.

49. Yao, H.; Olsen, B. D., SANS quantification of bound water in water-soluble polymers across multiple concentration regimes. *Soft Matter* **2021,** *17* (21), 5303-5318.

50. Nielsen, F., A simple approximation method for the Fisher–Rao distance between multivariate normal distributions. *Entropy* **2023,** *25* (4), 654.

51. Burnham, K. P.; Anderson, D. R., Multimodel Inference. *Sociological Methods & Research* **2016,** *33* (2), 261-304.

52. Hastie, T.; Tibshirani, R.; Friedman, J. H.; Friedman, J. H., *The elements of statistical learning: data mining, inference, and prediction*. Springer: 2009; Vol. 2.

53. Movassaghian, S.; Merkel, O. M.; Torchilin, V. P., Applications of polymer micelles for imaging and drug delivery. *Wiley Interdisciplinary Reviews: Nanomedicine and Nanobiotechnology* **2015,** *7* (5), 691-707.

54. Kim, K. S.; Park, W.; Hu, J.; Bae, Y. H.; Na, K., A cancer-recognizable MRI contrast agents using pH-responsive polymeric micelle. *Biomaterials* **2014,** *35* (1), 337-343.

55. Sureka, H. V.; Obermeyer, A. C.; Flores, R. J.; Olsen, B. D., Catalytic biosensors from complex coacervate core micelle (C3M) thin films. *ACS applied materials & interfaces* **2019,** *11* (35), 32354-32365.

56. Gokhale, D.; Chen, I.; Doyle, P. S., Micelle-laden hydrogel microparticles for the removal of hydrophobic micropollutants from water. *ACS Applied Polymer Materials* **2022,** *4* (1), 746-754.

57. Malashkevich, V. N.; Kammerer, R. A.; Efimov, V. P.; Schulthess, T.; Engel, J., The Crystal Structure of a Five-Stranded Coiled Coil in COMP: A Prototype Ion Channel? *Science* **1996,** *274* (5288), 761-765.

58. Rao, A.; Olsen, B. D., Structural and dynamic heterogeneity in associative networks formed by artificially engineered protein polymers. *Soft Matter* **2023,** *19* (33), 6314-6328.

59. Bingham, N. H.; Fry, J. M., *Regression: Linear models in statistics*. Springer Science & Business Media: 2010.

60. Shibayama, M.; Tanaka, T.; Han, C. C., Small-angle neutron scattering study on weakly charged temperature sensitive polymer gels. *The Journal of Chemical Physics* **1992,** *97* (9), 6842-6854.

61. Evmenenko, G.; Theunissen, E.; Mortensen, K.; Reynaers, H., SANS study of surfactant ordering in κ-carrageenan/cetylpyridinium chloride complexes. *Polymer* **2001,** *42* (7), 2907-2913.

62. Debye, P.; Bueche, A., Scattering by an inhomogeneous solid. *Journal of Applied Physics* **1949,** *20* (6), 518-525.

63. Guinier, A.; Fournet, G.; Yudowitch, K. L., Small-angle scattering of X-rays. **1955**.

64. Stieger, M.; Pedersen, J. S.; Lindner, P.; Richtering, W., Are Thermoresponsive Microgels Model Systems for Concentrated Colloidal Suspensions? A Rheology and Small-Angle Neutron Scattering Study. *Langmuir* **2004,** *20* (17), 7283-7292.

Chapter 4 Temperature Induced Concentrated Solution Self Assembly in Asymmetric Elastin Like Polypeptides Triblock Copolymers

4.1 Abstract

Precise sequence design in thermoresponsive elastin-like polypeptides (ELPs) enables molecular-level control over hydrophobicity, charge, and chain architecture compared to synthetic polymers, offering a powerful route to tune self-assembly in water. Here, two asymmetric ABA triblock ELPs, SI75 and SN75, were engineered with identical length but contrasting midblock hydrophobicity to probe how sequence-encoded interactions dictate phase behavior and nanostructure formation in concentrated solutions. SI75, containing a hydrophobic (VPGIG) midblock, underwent a sharp lower critical solution temperature (LCST) transition, formed disordered micellar aggregates above 13.6 °C, and exhibited pronounced syneresis upon further heating. In contrast, SN75, with a more hydrophilic (VPGNG) midblock, remained fully soluble and unstructured over 8–70 °C. Small-angle neutron scattering (SANS) revealed temperature-driven micellization in SI75 without long-range ordering, consistent with moderate segregation strength ($\chi N \approx 100$), while rheology showed soft-solid behavior that weakened upon heating. SN75 displayed viscoelastic liquid behavior across most of the temperature range, transitioning to a soft solid only above 60°C. Contrast variation SANS further demonstrated preferential water partitioning into the hydrophilic corona blocks, with micellar hydration increasing upon heating to enhance packing efficiency. These findings directly link midblock hydrophobicity to thermoresponsive self-assembly and syneresis in ELP triblocks, establishing sequence-based design rules for controlling nanostructure, mechanics, and solvent organization in protein-based materials.

4.2 Introduction

Synthetic block copolymers can self-assemble into various nanostructures, such as spheres, cylinders, lamellae, gyroids, and vesicles, in bulk or solution that are thermodynamically driven by enthalpy of demixing and entropy of polymer chains[1]. Their tunable properties, biocompatibility, and cost-effectiveness have led to broad applications in lithography[2-4], sensing[5, 6], drug delivery[7, 8], and tissue engineering[9-11]. In particular, the nanostructures formed by block copolymer in highly concentrated solution are often termed gels given that they possess a finite yield stress and do not flow on their own. However, the gel properties resulted from the ordered structure rather than crosslinking between polymer chains. One of the well-studied triblock copolymer systems is the Pluronic triblock copolymers in pure aqueous or a mixture of solutions, given their commercial applications. They consist of hydrophilic polyethylene oxide (PEO) and hydrophobic polypropylene oxide (PPO) monomers with a chemical composition of $EO_mPO_nEO_m$. It has been found that the molar ratio of the blocks (m/n) and its total molecular weight has a significant influence on the phases formed on the phase diagram[12]. In general, the larger the PPO block and the greater the PEO content, the greater is the gelling ability of the Pluronic copolymer. Gelation in those systems occurs through a percolation transition[13].

Unlike synthetic polymers, block copolypeptides offer monodispersity and sequence precision via recombinant expression, enabling detailed control over interactions such as hydrophobicity and charge. This makes them ideal for studying how primary sequence influences self-assembly in solution. Specifically, elastin-like polypeptides (ELPs), comprising repeating units of $(VPGXG)_n$, where X is any residue except proline, are thermoresponsive and phase-separate in response to stimuli like temperature, pH, and ionic strength. Many ELP systems have been designed and characterized in dilute or semi-dilute solutions. Urry et al. established the

hydrophobicity scale based on the ELPs' transition temperatures at semi-dilute conditions[14]. They found that substituting more hydrophobic guest residues in the ELP sequence can decrease the transition temperature, while substituting charged or hydrophilic guest residues can increase the transition temperature. Their finding established important guidance for using sequence to achieve target transition temperatures. Conticello and coworkers pioneered the work of designing ELP block copolymer self-assembly in solution. With $[VPGEG(IPGAG)_4]_{14}$ as the hydrophilic and $[VPGFG(IPGVG)_4]_{16}$ as the hydrophobic domain, the ELPs can self-assemble into nanoparticles above the critical micelle temperature and aggregate above the transition temperature of ELPs[15]. They demonstrated that ELPs can have a high degree of morphology control of the self-assembled nanostructure and temperature can be adjusted for the formation of nanoparticles or aggregates in dilute solutions. This opened up possibilities using ELPs for drug delivery applications.

With higher concentrations in solution, ELPs can undergo arrested phase separation to form thermoresponsive hydrogels with remarkable mechanical properties. Glassman reported that ELPs with alanine in the third position of the pentapeptide repeat, such as $(XPAVG)_n$, form stiff hydrogels through a mechanism of arrested spinodal decomposition at concentrations above 15 wt%. These hydrogels exhibit shear moduli up to 1 MPa and display long stress relaxation times ($>10^3$ s), with a disordered bicontinuous nanostructure that can be tuned via sequence, molecular weight, and buffer conditions[16]. Recent studies by Navarro et al. revealed that diblock copolypeptides consists of resilin-like and elastin-like polypeptides (RLP_n-b-ELP_{80}) can form microphase-separated structures in concentrated aqueous solutions[17]. The length of the ELP is kept constant at 80 while varying the RLP length n from 20 to 100 in 20 increments. Using small-angle X-ray scattering (SAXS), Navarro and colleagues demonstrated the formation of ordered nanostructures, such as hexagonally packed cylinders and lamellae, at concentrations of 30–50

wt%. These structures were highly sensitive to the RLP fraction, block length, concentrations, and temperature, emphasizing the role of polymer-solvent interactions in driving phase transitions.

In this work, inspired by synthetic block copolymer and recombinant block copolypeptides literatures, two novel asymmetric triblock ELPs $A_{30}B_{25}A_{45}$ were designed to investigate how midblock hydrophobic contrast influences phase formation, self-assembly, and mechanical properties in concentrated solutions. The sequence with greater hydrophobic contrast formed micelles above its turbidity transition temperature and exhibited syneresis upon further heating. In contrast, the sequence with lower hydrophobic contrast showed no turbidity changes or long-range ordering with temperature increase but underwent a sol–gel transition at high temperature. These distinct behaviors, arising from systematic sequence design, provide new insights into sequence–property relationships in ELPs and inform the development of thermoresponsive materials.

4.3 Methods

4.3.1 Protein Expression and Purification. Two ELPs were designed for this study: SI75 with primary sequence $(VPGSG)_{30}(VPGIG)_{25}(VPGSG)_{45}$ and SN75 with primary sequence $(VPGSG)_{30}(VPGNG)_{25}(VPGSG)_{45.}$ The genes are cloned in expression vector pQE-9. Proteins were expressed in Escherichia coli strain C41(DE3). Expression was performed using 5 L of Luria Broth (LB) with 100 mg/L ampicillin in a fermenter with 7 L working volume at 37 °C for 6 h with 1mM IPTG induction. Cells were harvested by centrifugation, resuspended in 100 mL of lysis buffer B (100mM NaH_2PO_4, 10mM Tris Cl, 8M urea, pH = 8) per 30 g of wet cell mass. The lysate was then sonicated with a sonicator and clarified by high-speed centrifugation at 4 °C. 30mL Ni-NTA resins were added to the clarified lysate and let the mixture bind overnight at 4°C shaker. The Ni-NTA affinity column was washed with 4 column volume buffer B twice at pH = 8, 7.3, 6.3. The protein was eluted from the column with buffer B at pH = 5.9 twice and 4.3 five times to ensure

complete elution of protein. The eluted proteins were dialyzed three times in EDTA buffer and five times in MillQ water. The resulted protein solutions were further purified with thermal cycling by incubating in 70 °C for 5 hours and cooling at 4 °C overnight. The resulting solutions were filtered with PES filters with 0.22um pore size and lyophilized for characterization. The purity was confirmed with SDS-PAGE as shown in Figure B-1.

4.3.2 Small Angle Neutron Scattering. The SI75 and SN75 were loaded into 0.5mm demountable copper cell by pipetting the samples on one side of the quartz window and compressing the second quartz window onto the sample carefully to prevent bubble formation. The 100mM phosphate buffer was loaded into 1mm banjo cell. The small-angle neutron scattering experiments were performed on GP-SANS and EQ-SANS at Oak Ridge National Lab (ORNL)[18]. For GP-SANS, the lower q configuration has a sample-to-detector distance of 4 m, which corresponds to low q range of $6.58 * 10^{-3}$ to $1.07 * 10^{-1}$Å. The higher q configuration has a sample to detector distance of 1.5 m, which corresponds to high q range of $3.76 * 10^{-2}$ to $4.64 * 10^{-1}$ Å. The wavelength for both configurations is 6 Å. Neutron counts are between 1 to 6 million at the low q and approximately 20 million at high q. SANS data were background subtracted. For EQ-SANS, two configurations were used to cover q range of 0.05 to 0.73 Å. The lower q configuration has a sample-to-detector distance of 7m with a wavelength of 10 Å. The higher q configuration has a sample to detector distance of 2.5m with wavelength of 2.5 Å. The aperture of the beam is 10mm. All observed transitions were reversible with temperature, indicating that minimal beam damage is observed.

4.4 Results and Discussion

4.4.1 Sequence Design and Thermodynamic Considerations

Two asymmetric *ABA* triblock copolymer ELP sequences were systematically designed to investigate the effect of hydrophobicity on the self-assembly behavior in concentrated solutions as shown in Table 4-1. These sequences, named SN75 and SI75, share identical architecture and molecular weight, differing only in the guest residue of the hydrophobic B block.

Both sequences feature hydrophilic A blocks composed of $(VPGSG)_n$ repeats, flanking a central B block made of either $(VPGNG)_{25}$ in SN75 or $(VPGIG)_{25}$ in SI75. The isoleucine-containing midblock (SI75) is significantly more hydrophobic than its asparagine counterpart (SN75), enabling a controlled comparison of protein-protein and protein-solvent interactions. To ensure good solubility at higher concentrations, the hydrophobic block length was kept short, while the hydrophilic block fraction (f_A) was tuned to be either 0.3 or 0.45, depending on total sequence length. The total number of pentapeptide repeats (N) was fixed at 100 for both sequences, and the Flory-Huggins interaction parameter (χ) was used to estimate the thermodynamic incompatibility between A and B blocks. Values of χ were calculated using Hansen solubility parameters derived from group contribution methods[22], which can be found in Supplementary Information. The segregation strength, χN, was used as a dimensionless parameter to predict self-assembly behavior, drawing on analogies to synthetic block copolymer systems, where increasing χN leads to stronger microphase separation and more ordered nanostructures.

Table 4-1. ELPs sequences, molecular weight, composition, and interaction parameters.

ELP Name	Protein Sequence	Molecular Weight (kDa)	f_B	Predicted χN
SN75	$(VPGSG)_{30}(VPGNG)_{25}(VPGSG)_{45}$	42.9	0.25	30
SI75	$(VPGSG)_{30}(VPGIG)_{25}(VPGSG)_{45}$	42.9	0.25	100

4.4.2 Sequence-Dependent Phase Behavior and Morphology in Concentrated Solutions

The two sequences exhibited distinctively different behaviors consistent with their hydrophobic midblock compositions. SN75, containing the more hydrophilic $(VPGNG)_{25}$ midblock, remains optically clear over the full temperature range tested (8–70 °C) at 40 w/v%. The transition temperature of the homopolymer of VPGSG and VPGNG are estimated to be 50°C. No birefringence was observed across this range, and SANS measurements confirmed the absence of micellar or ordered nanostructures (Figure 4-2(a)). These observations align with a soluble, low-segregation-strength block copolymer in a good solvent.

In contrast, SI75, with the more hydrophobic $(VPGIG)_{25}$ midblock, displays a sharp turbidity transition at 13.6 °C, indicating thermally induced aggregation. Literature estimated the transition temperature of a homopolymer consists of VPGIG to be 10°C[14]. This transition is consistent with the aggregation of the hydrophobic $(VPGIG)_{25}$ midblock. Below this transition, the solution remains transparent and exhibited no birefringence, as indicated by a power fraction near zero. Above the transition temperature, the solution becomes opaque, and a slight increase in power fraction is observed. However, this increase can be result from instrument limitations: although the voltage signal remains unchanged (Figure B-2), the decrease in transmission artificially

amplifies the power fraction. This artifact, previously reported in the literature for highly scattering samples[23], limits the reliability of DPLS in quantifying structure with high turbidity under such conditions.

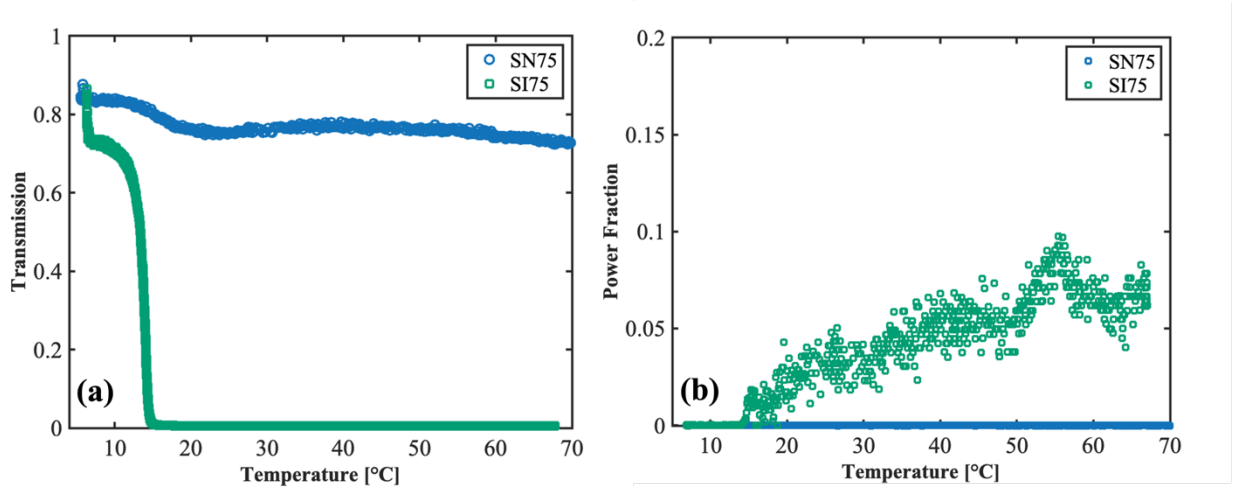


Figure 4-1. (a) Transmission as a function of temperature at 1°C/min. (b) Power fraction temperature ramp as a function of temperature at 1°C/min. Blue circles represent 40w/v% SN75 and green circles represent 40w/v% SI75.

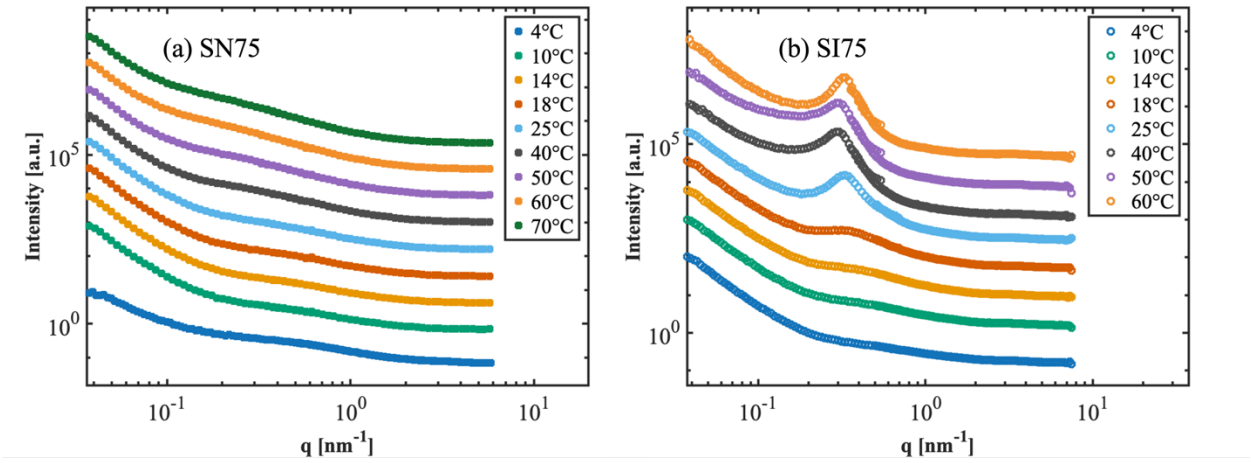


Figure 4-2. SANS temperature profile for (a) SN75 at 40w/v% and (b) SI75 at 40w/v%. The data were offset by 10 with increasing temperature for clarity.

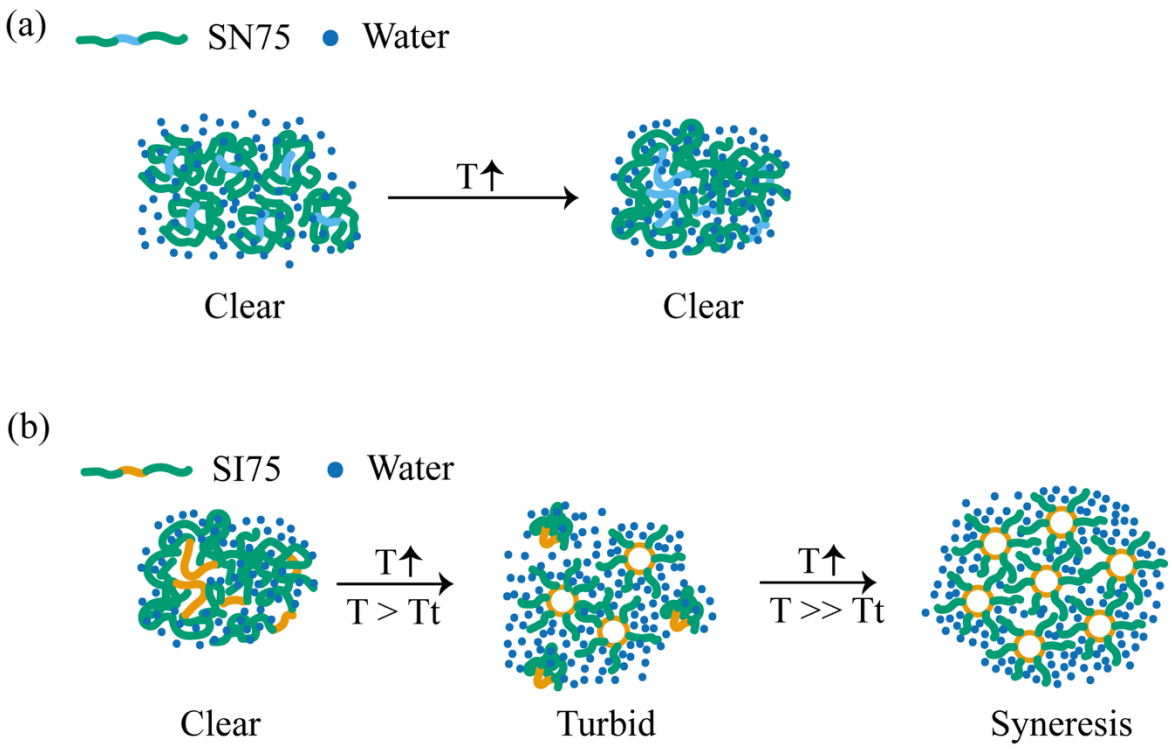


Figure 4-3. Illustration of molecular mechanisms as a function of increasing temperature in (a) SI75, and (b) SN75.

SANS analysis using the correlation length model shows that SN75 undergoes a temperature-driven structural transition from disordered concentrated polymer solution to a mesoscopically ordered dense fluid. The SANS curves show no sharp peaks from 4°C to 60°C. This observation is consistent with the absence of birefringence in DPLS measurements, which indicates lack of long-range periodicity in SN75. Because SN75 exhibits peak-free SANS profiles characteristic of a homogeneous, one-phase gel similar to the SANS profile of poly(ethylene glycol) hydrogel[24], data is fitted to the correlation length model. The Porod exponent n characterizes the fractal structure of the gel, while the Lorentzian exponent m characterizes the polymer–solvent interactions and therefore describes the thermodynamics of the system. In semi-dilute solutions, ξ is the average distance between polymer chains set by entanglements or transient

associations[24, 25]. In gels or network systems, the correlation length indicates the mesh size of the gel. The parameters were extracted across a range of temperatures (4–70 °C) as shown in Table 4-2 to understand how thermal stimuli affect the structural organization of SN75.

At 4 °C, SN75 exhibits a relatively short correlation length (ξ = 18.7 Å), consistent with a well-solvated, disordered polypeptide solution. As temperature increases, the correlation length increases monotonically up to 60°C with a maximum value of 57.1 Å and decreases slightly at 70°C. This results from system reaching percolation and transitioning from a liquid state to a gel state. A Porod exponent, *n*, of 2 or greater, suggests that SN75 remains a one-phase system even at elevated temperatures, which is consistent with the turbidimetry measurement that the system remains clear. The consistently high n ~ 4.15 throughout 40–70 °C reinforces the formation of a gel, which is also supported by rheological measurements in the later section. The Lorentzian exponents are close to 2, indicating good solvent quality as expected from hydrophilic ELPs.

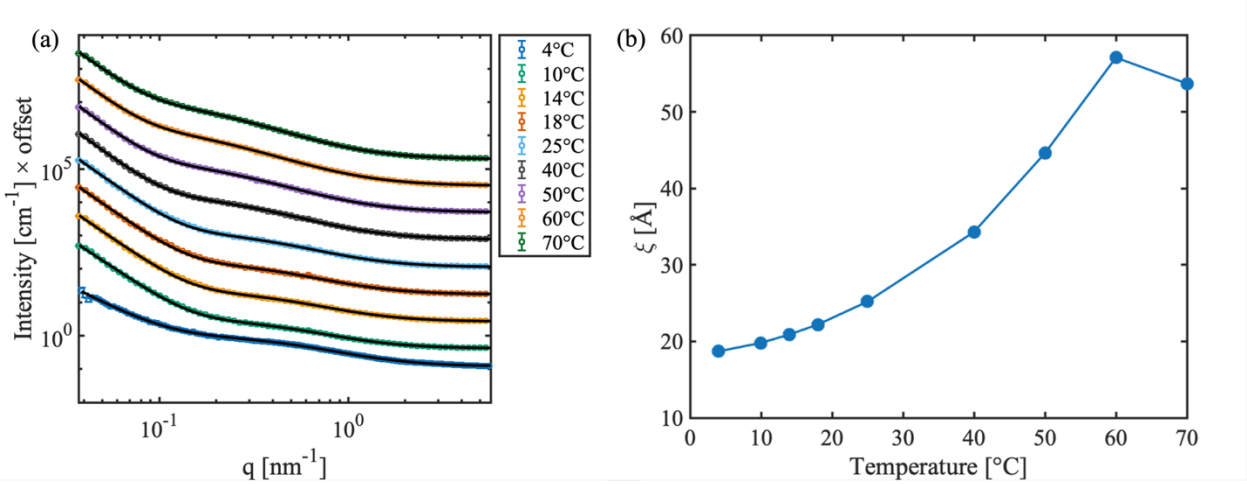


Figure 4-4. (a) SANS model fits of 40w/v% SN75 to the correlation length model as a function of temperature. The solid lines are fitting curves. The SANS data and fitting are offset for clarity. (b) Correlation length ξ from fitting correlation length model fits as a function of temperature.

Table 4-2. SN75 40w/v% SANS fitting parameters to the correlation length model.

Temperature [°C]	A	n	C	ξ [Å]	m	Background [cm^{-1}]
4	2.00×10^{-6}	2.904	0.78	18.7	2.04	0.1198
10	5.34×10^{-8}	3.79	0.37	19.8	2.16	0.0657
14	3.47×10^{-8}	3.90	0.41	20.9	2.14	0.0664
18	2.76×10^{-8}	3.96	0.48	22.2	2.13	0.0682
25	2.11×10^{-8}	4.02	0.64	25.2	2.11	0.0712
40	8.90×10^{-9}	4.17	1.26	34.3	2.07	0.0763
50	8.32×10^{-9}	4.18	2.11	44.6	2.04	0.0783
60	8.35×10^{-9}	4.19	3.47	57.1	2.03	0.0783
70	1.15×10^{-8}	4.13	3.29	53.7	2.09	0.078

At low temperatures (4–14 °C), SI75 exhibits liquid-like scattering behavior, similar to SN75. Above the phase transition temperature, a broad primary peak appears in the SANS profiles at 18 °C and sharpens progressively with increasing temperature (25–60 °C), indicating the formation of disordered micellar structures.

Unlike concentrated block copolymer melts or diblock ELP-RLP systems, the ABA triblock ELPs do not exhibit long-range ordering, as evidenced by the absence of higher-order peaks in the SANS data. This lack of ordering is consistent with a low segregation strength (χN), which is influenced by the Flory-Huggins interaction parameter (χ), polymer chain length (N), and concentration. Literature on block copolymer self-assembly suggests that higher χN values are typically required to drive the formation of well-ordered phases such as lamellae, hexagonally packed cylinders, body-centered cubic, or gyroid structures.

In the sequences studied here, only a single amino acid (X) is varied in the repeating VPGXG motif, potentially resulting in a smaller χ between blocks than ELP-RLP diblock copolypeptides. Additionally, the relatively short overall chain length ($N = 100$) may further limit χN, preventing the formation of long-range nanostructures. Increasing concentration could potentially enhance packing efficiency, but solubility constraints limited the analysis to 40 w/v%.

SANS profiles above the transition temperature suggest a macrophase-separated system, composed of a water-rich phase and a dense, disordered micellar phase, as evidenced by the appearance of primary peak and lack of secondary peaks. Similar structures have been reported in mCherry–PNIPAM conjugates. To quantitatively analyze the micellar phase, the Percus-Yevick hard sphere model was applied to SI75 SANS data from 18–60 °C. Fitting results are shown in Figure 4-5 and the parameter fits are summarized in **Table 4-3**.

As temperature increases, the primary peak becomes more defined, corresponding to a rise in micelle packing fraction (η), reaching 0.374 at 60 °C. This increase reflects a greater number of micelles forming at elevated temperatures. However, the packing fraction remains significantly lower than that of typical ordered phases in block copolymer melts—for example, a body-centered cubic (BCC) lattice has a packing fraction of ~0.68. The lower value observed for SI75 supports the conclusion that the system remains in a disordered micellar state.

The primary peak position shifts to lower q-values from 18 °C to 50 °C, indicating an increase in micelle size. The intermicellar half-distance (R_{HS}) grows from 7.2 nm at 18 °C to 10.4 nm at 50 °C. At 60 °C, a slight shift to higher q is observed, and R_{HS} decreases to 9.8 nm, suggesting mild compaction or rearrangement at higher temperatures. The fitted micelle radius (R_o) and polydispersity (σ_p) show similar temperature-dependent trends.

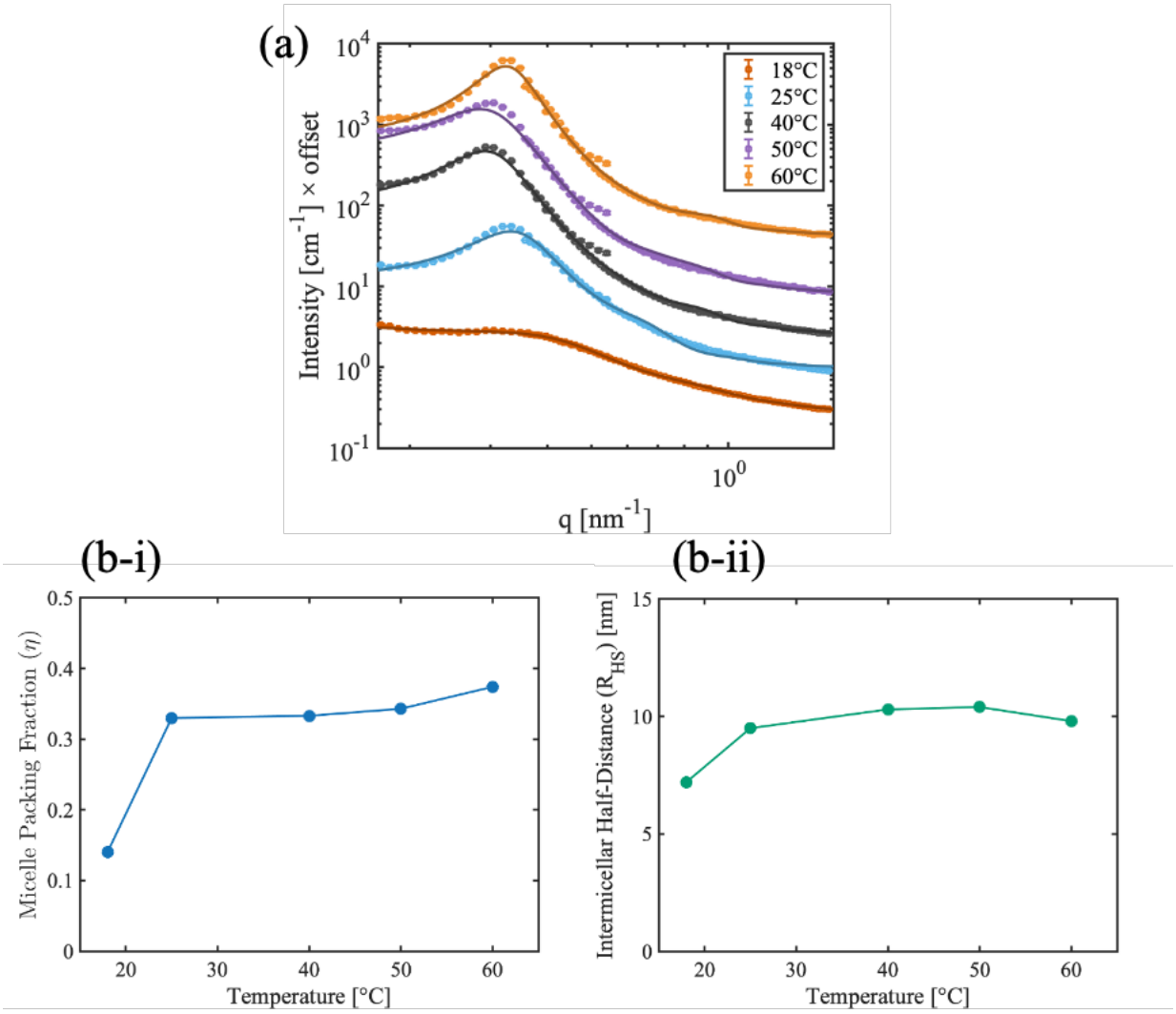


Figure 4-5. (a) SANS model fits of the Percus Yevick model to 40 w/v% SI75. The solid lines are fitting curves. (b) Fitting Parameters to the Percus Yevick Model as a function of temperature. (b-i) Micelle packing fraction. (b-ii) Intermicellar half-distance.

Table 4-3. SI75 40w/v% SANS fitting parameters to the Percus Yevick model.

Temperature [°C]	K [$\times 10^{-12}$]	η	R_{HS} [nm]	R_0 [nm]	σ_p [nm]	C [cm^{-1}]
18	3.12×10^5	0.140	7.2	0.4	0.328	0.22
25	321.7	0.330	9.5	5	0.843	0.32
40	223.7	0.333	10.3	7.1	1.4	0.13
50	183.0	0.343	10.4	7.3	1.2	0.10
60	211.4	0.374	9.8	6.6	1.1	0.14

The two ELP sequences also exhibits distinctive rheological behavior with respects to temperature consistent with SANS measurements. Temperature sweep of SN75 shows low storage and loss modulus from 4°C to 60°C, which shows viscoelastic liquid like behavior consistent with SANS. As the temperature increases to 63°C, the modulus sharply increases to 10^5 Pa, corresponding to SN75 block copolymer forming a gel upon heating without observing change in turbidity. The transition temperatures of the homopolymer VPGNG and VPGSG are estimated to be 50°C, which is much greater than the hydrophobic isoleucine ELP homopolymer. Since SN75 is very hydrophilic, this increase in mechanical property with only 8% change in turbidity might result from increasing interaction of the polymer with itself, and slight decrease in the polymer-solvent interaction upon heating as illustrated in Figure 4-3(a). Frequency sweeps as shown in Figure B-4 at various temperatures indicate that SN75 is a viscoelastic liquid at lower temperatures and transitions to a solid at 60°C after an hour of equilibration. For the other sequence SI75, during the initial equilibration, the sample transitions from liquid to solid. Temperature sweep shows a

gradual decrease of both storage and loss modulus from as temperature increases, where the transition temperature of SI75 from turbidimetry measurement lies in the region of decreasing modulus. This suggests a slower structural change upon increasing temperature, which agrees with the sharpening of the primary peak in SANS. The decrease in modulus is due to syneresis. As more micelles form upon increasing temperature, the micelles packs more tightly, and expels liquid from the gel. Frequency sweeps at various temperatures shows no crossover of storage and loss modulus with greater value of storage modulus, indicating that SI75 behaves like a gel similar to the Pluronics gel at high concentrations.

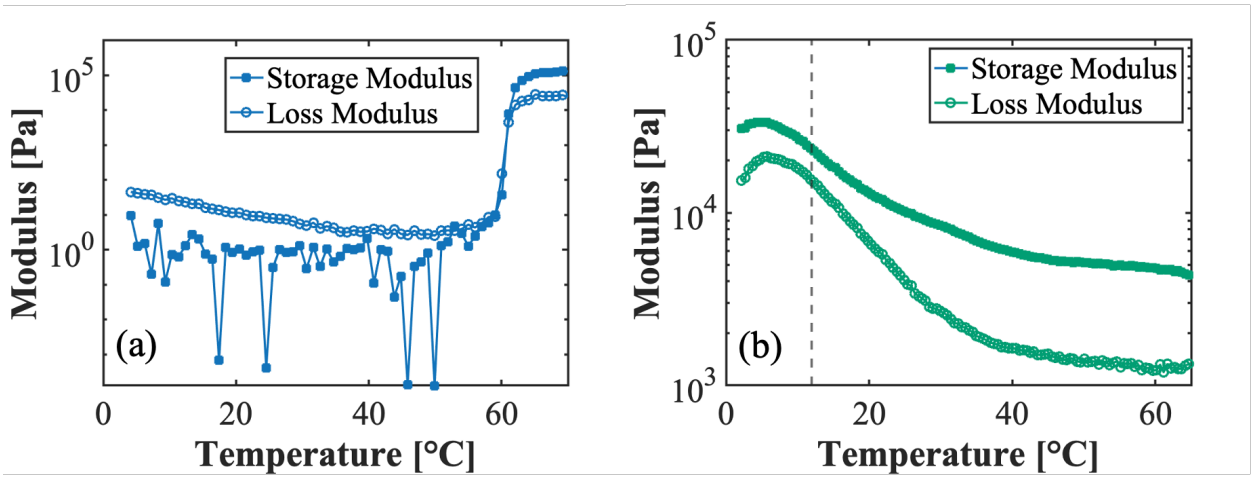


Figure 4-6. Temperature sweep of (a) SN75 at 0.1% strain from 4°C to 70°C. (b) SI75 at 1% strain from 2°C to 65°C. The dashed line indicates the transition temperature measured from turbidimetry.

4.4.3 Solvent Distribution in SI75

The solvent distribution is important in self-assembly, as its partition between the different blocks can shift the phase behavior in many block copolymer systems as a result of changes in segregation length. SANS water partition analysis shows that water always prefer to partition in the serine endblocks in SI75. Residual analysis from the fitting (Figure 4-7) shows two local minima—one at 0 and another at $\theta_{water\ blend}$—corresponding to hypothetical scenarios where all

the water partitions into either the hydrophobic midblock (VPGIG) or the hydrophilic endblocks (VPGSG), respectively. Across all temperatures, the minimum residual was consistently found at $\theta_{water\ blend}$, indicating that water preferentially solvates the hydrophilic endblocks.

At temperatures near the transition (12 °C and 14 °C), all detectable water remains associated with the (VPGSG) segments, consistent with a well-solvated pre-micellar or early micellar state. As temperature increases beyond the transition (e.g., 25 °C and 60 °C, Figure 4-8), this trend persists, confirming that micelle formation does not disrupt the preferential hydration of the hydrophilic blocks as shown in Figure 4-8(b). This is consistent with syneresis and micelle architecture, where the hydrophilic segments form the corona and remain solvated, while the hydrophobic midblock forms the collapsed core.

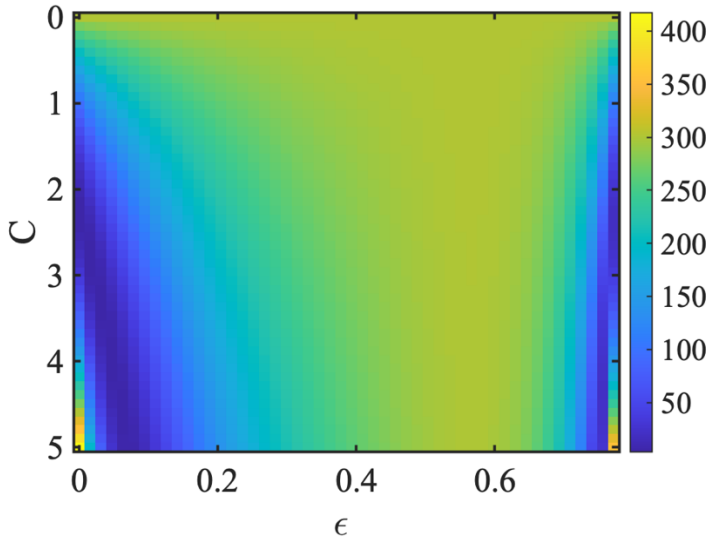


Figure 4-7. SI75 40w/v% at 25°C water distribution fitting. Heatmap of the residuals with respect to C and ϵ. C is the fitting constant and ε is the amount of water that partitions into the I block. The sidebar was from 0 to the maximum.

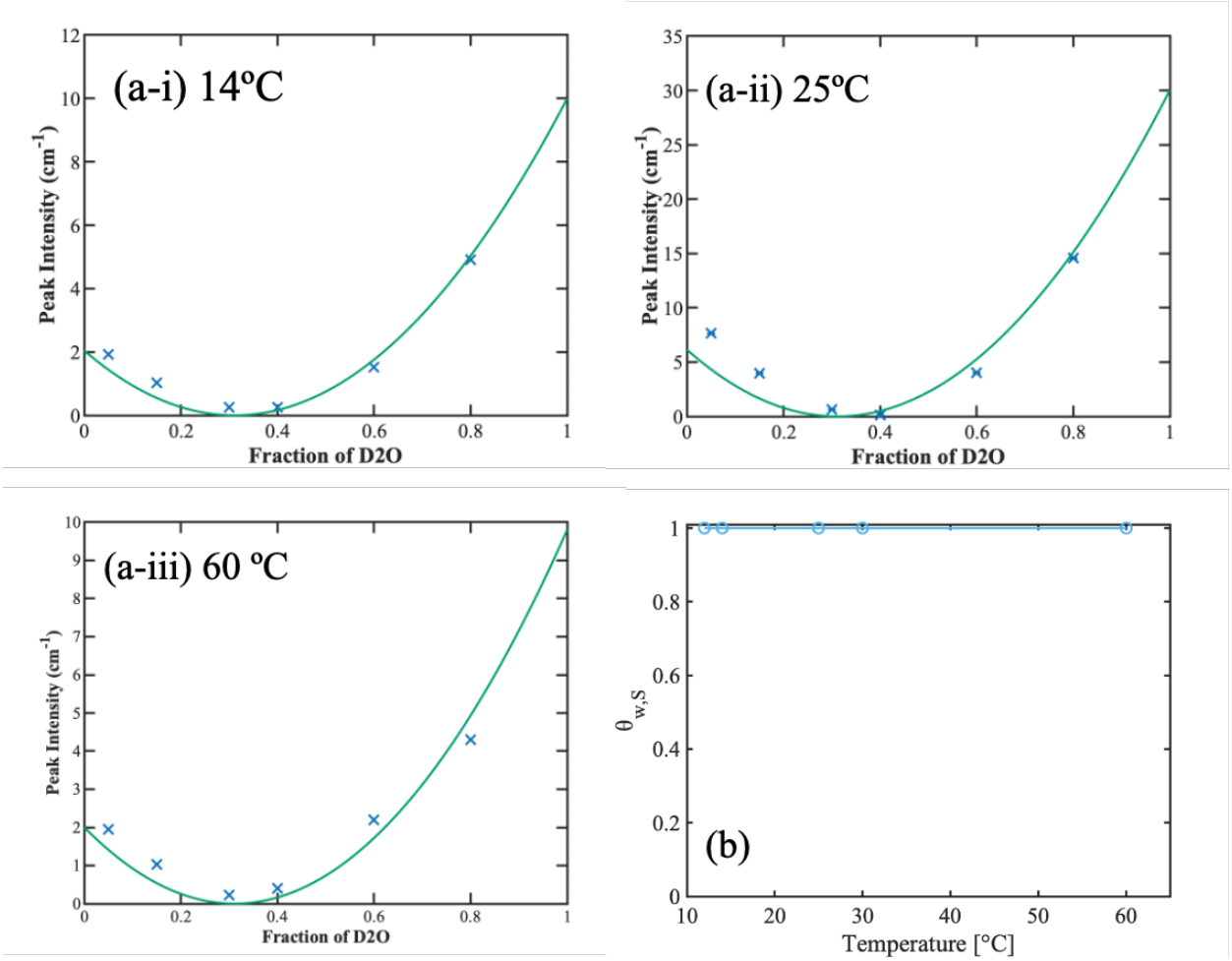


Figure 4-8. (a) Small-angle neutron scattering intensities (data points) and curve fits (solid line) for SI75 40w/v% in different H_2O/D_2O blend compositions at representative temperatures (a-i) 14 °C, (a-ii) 25 °C, and (a-iii) 60 °C. (b) Water distribution of SI75 as a function of temperature from the water partitioning analysis.

4.5 Conclusion

This study demonstrates how midblock variations in elastin-like polypeptide (ELP) triblock copolymers dictate their thermoresponsive self-assembly, mechanical properties, and solvent organization in concentrated solutions. Despite identical architecture and molecular weight, SI75 and SN75 exhibited fundamentally different behaviors due to the hydrophobicity contrast of

their midblocks. The isoleucine-containing SI75 underwent a sharp LCST transition, forming disordered micelles that progressively packed and expelled solvent, leading to syneresis and soft-solid behavior with decreasing modulus upon heating. In contrast, the asparagine-containing SN75 remained optically clear and disordered over a wide temperature range, with scattering profiles captured by a correlation length model that revealed an increase in correlation length with temperature, consistent with a transition from a concentrated polymer solution to a percolated gel at elevated temperatures. Rheological measurements corroborated these findings: SN75 displayed viscoelastic liquid behavior that evolved into a weak gel only above 60 °C, whereas SI75 maintained gel-like properties across the temperature range but with a progressive reduction in modulus due to solvent expulsion. Contrast-variation SANS further showed that water consistently partitions into the hydrophilic corona blocks of SI75, confirming a micelle architecture that supports temperature-driven syneresis.

Taken together, these results directly link midblock hydrophobicity to thermoresponsive self-assembly pathways in ELP triblocks. SI75 highlights how increased hydrophobic contrast can drive micellization and syneresis, while SN75 underscores the stabilizing influence of hydrophilic blocks in suppressing aggregation and promoting gelation only at elevated temperatures. Beyond elucidating sequence–property relationships, this work establishes molecular-level design rules for tuning nanostructure, mechanical properties, and solvent distribution in protein-based materials, offering new strategies for engineering thermoresponsive biomaterials and hydrogels with programmable behavior.

4.6 Acknowledgement

We thank the department of energy (Grant: DE-SC0007106) for supporting this research and the support of Oakridge National Laboratory (ORNL) for the neutron scattering facilities in

this work. This research used resources at the High Flux Isotope Reactor and the Spallation Neutron Source, a DOE Office of Science User Facility operated by the Oak Ridge National Laboratory. We acknowledge Dr. Lilin He for the experimental assistance and data analysis at ORNL. We thank Brian Carrick, Oliver Xie, and Zhi Kai Tio in assistance of the SANS measurement.

4.7 Reference

1. Karayianni, M.; Pispas, S., Block copolymer solution self-assembly: Recent advances, emerging trends, and applications. *Journal of Polymer Science* **2021,** *59* (17), 1874-1898.
2. Ross, C. A.; Berggren, K. K.; Cheng, J. Y.; Jung, Y. S.; Chang, J.-B., Three-Dimensional Nanofabrication by Block Copolymer Self-Assembly. *Advanced Materials* **2014,** *26* (25), 4386-4396.
3. Bates, C. M.; Maher, M. J.; Janes, D. W.; Ellison, C. J.; Willson, C. G., Block Copolymer Lithography. *Macromolecules* **2014,** *47* (1), 2-12.
4. Maekawa, S.; Seshimo, T.; Dazai, T.; Sato, K.; Hatakeyama-Sato, K.; Nabae, Y.; Hayakawa, T., Chemically tailored block copolymers for highly reliable sub-10-nm patterns by directed self-assembly. *Nature Communications* **2024,** *15* (1), 5671.
5. Chung, Y.-H.; Oh, J. K., Research Trends in the Development of Block Copolymer-Based Biosensing Platforms. *Biosensors* **2024,** *14* (11), 542.
6. Putranto, A. F.; Fleury, G.; Wulandari, C.; Muslihati, A.; Amrillah, Y. T.; Yuliarto, B.; Kogelschatz, M.; Nugroho, F. A. A.; Wasisto, H. S.; Zelsmann, M., Block Copolymer Self-Assembly for Biological and Chemical Sensing. *ACS Applied Polymer Materials* **2024,** *6* (24), 14970-15001.
7. Gaucher, G.; Dufresne, M.-H.; Sant, V. P.; Kang, N.; Maysinger, D.; Leroux, J.-C., Block copolymer micelles: preparation, characterization and application in drug delivery. *Journal of controlled release* **2005,** *109* (1-3), 169-188.
8. Kazunori, K.; Masayuki, Y.; Teruo, O.; Yasuhisa, S., Block copolymer micelles as vehicles for drug delivery. *Journal of controlled release* **1993,** *24* (1-3), 119-132.
9. Chung, H. J.; Park, T. G., Self-assembled and nanostructured hydrogels for drug delivery and tissue engineering. *Nano Today* **2009,** *4* (5), 429-437.
10. Kim, H. K.; Shim, W. S.; Kim, S. E.; Lee, K.-H.; Kang, E.; Kim, J.-H.; Kim, K.; Kwon, I. C.; Lee, D. S., Injectable in situ–forming pH/thermo-sensitive hydrogel for bone tissue engineering. *Tissue Engineering Part A* **2009,** *15* (4), 923-933.
11. Kutikov, A. B.; Song, J., Biodegradable PEG-based amphiphilic block copolymers for tissue engineering applications. *ACS biomaterials science & engineering* **2015,** *1* (7), 463-480.
12. Wanka, G.; Hoffmann, H.; Ulbricht, W., Phase diagrams and aggregation behavior of poly (oxyethylene)-poly (oxypropylene)-poly (oxyethylene) triblock copolymers in aqueous solutions. *Macromolecules* **1994,** *27* (15), 4145-4159.
13. Kelarakis, A.; Mingvanish, W.; Daniel, C.; Li, H.; Havredaki, V.; Booth, C.; Hamley, I. W.; Ryan, A. J., Rheology and structures of aqueous gels of diblock (oxyethylene–oxybutylene) copolymers with lengthy oxyethylene blocks. *Physical Chemistry Chemical Physics* **2000,** *2* (12), 2755-2763.
14. Urry, D. W.; Gowda, D. C.; Parker, T. M.; Luan, C.-H.; Reid, M. C.; Harris, C. M.; Pattanaik, A.; Harris, R. D., Hydrophobicity scale for proteins based on inverse temperature transitions. *Biopolymers* **1992,** *32* (9), 1243-1250.
15. Wright, E. R.; Conticello, V. P., Self-assembly of block copolymers derived from elastin-mimetic polypeptide sequences. *Advanced Drug Delivery Reviews* **2002,** *54* (8), 1057-1073.
16. Glassman, M. J.; Olsen, B. D., Arrested Phase Separation of Elastin-like Polypeptide Solutions Yields Stiff, Thermoresponsive Gels. *Biomacromolecules* **2015,** *16* (12), 3762-3773.
17. Navarro, L. A.; Ryan, J. J.; Dzuricky, M.; Gradzielski, M.; Chilkoti, A.; Zauscher, S., Microphase Separation of Resilin-like and Elastin-like Diblock Copolypeptides in Concentrated Solutions. *Biomacromolecules* **2021,** *22* (9), 3827-3838.

18. Heller, W. T.; Cuneo, M.; Debeer-Schmitt, L.; Do, C.; He, L.; Heroux, L.; Littrell, K.; Pingali, S. V.; Qian, S.; Stanley, C.; Urban, V. S.; Wu, B.; Bras, W., The suite of small-angle neutron scattering instruments at Oak Ridge National Laboratory. *Journal of Applied Crystallography* **2018,** *51* (2), 242-248.
19. Percus, J. K.; Yevick, G. J., Analysis of Classical Statistical Mechanics by Means of Collective Coordinates. *Physical Review* **1958,** *110* (1), 1-13.
20. Hammouda, B.; Ho, D. L.; Kline, S., Insight into clustering in poly (ethylene oxide) solutions. *Macromolecules* **2004,** *37* (18), 6932-6937.
21. Lam, C. N.; Olsen, B. D., Phase transitions in concentrated solution self-assembly of globular protein–polymer block copolymers. *Soft Matter* **2013,** *9* (8), 2393-2402.
22. Stefanis, E.; Panayiotou, C., Prediction of Hansen Solubility Parameters with a New Group-Contribution Method. *International Journal of Thermophysics* **2008,** *29* (2), 568-585.
23. Mills, C. E.; Michaud, Z.; Olsen, B. D., Elastin-like Polypeptide (ELP) Charge Influences Self-Assembly of ELP–mCherry Fusion Proteins. *Biomacromolecules* **2018,** *19* (7), 2517-2525.
24. Saffer, E. M.; Lackey, M. A.; Griffin, D. M.; Kishore, S.; Tew, G. N.; Bhatia, S. R., SANS study of highly resilient poly(ethylene glycol) hydrogels. *Soft Matter* **2014,** *10* (12), 1905-1916.
25. Hammouda, B.; Ho, D. L.; Kline, S., Insight into Clustering in Poly(ethylene oxide) Solutions. *Macromolecules* **2004,** *37* (18), 6932-6937.

Chapter 5 Sequence Effect in the Cononsolvency of Elastin Like Polypeptides in Water, Ethanol, and Sodium Chloride Solutions

5.1 Abstract

Elastin-like polypeptides (ELPs) are a family of recombinant biopolymers that offer precise sequence control and exhibit both lower critical solution temperature (LCST)-like behavior in aqueous solutions and upper critical solution temperature (UCST)-like behavior in water/alcohol mixtures. Seven ELP sequences with systematically varied hydrophobicity and charge were designed to investigate how hydrophobicity and charge influence dilute solution phase behavior in 0–40 mol% ethanol and 0–200 mM sodium chloride. Both hydrophobic and hydrophilic ELPs in this study display four characteristic regimes in their phase diagrams: LCST-like transitions at low ethanol concentrations, a one-phase region at low-to-moderate ethanol concentrations, UCST-like transitions at intermediate ethanol concentrations, and full miscibility at high ethanol concentrations. The observed LCST behavior is consistent with literature, showing an inverse relationship with hydrophobicity. The LCST-like transition temperature increases with increasing ethanol content. In contrast, UCST-like transitions are influenced by hydrogen bonding capacity, side chain polarity, and conformational flexibility. Despite identical overall composition with a previously studied ELP sequences, differences in sequence significantly impact phase behavior in ethanol/water mixtures. The presence of the 6xHis purification tag has a small impact on the observed results due to the comparatively high molar mass of the ELP sequence. These results reveal both sequence-dependence in the phase behavior of ELPs and a universal cononsolvency behavior in uncharged hydrophobic and hydrophilic ELPs.

5.2 Introduction

The primary sequence of a polymer determines its chain conformation[1, 2], inter- and intramolecular interactions[3, 4], and ultimately its material properties[5]. In biological

macromolecules, such as proteins, sequence encodes function through well-defined folding and interaction motifs[6]. Extending this principle to synthetic and recombinant polymers opens avenues for designing materials can achieve diverse functionality, such as informational storage[7], cellular signaling[8], catalysis[9], and programmable self-assembly[10]. Although many simulation methods have been developed to study and predict the properties of those novel polymer sequences[11-14], the underlying sequence–property relationships remain insufficiently understood experimentally due to the challenges with synthetic chemistry needed for precise sequence control[15].

Elastin-like polypeptides (ELPs) are a family of recombinant biopolymers inspired by the mammalian protein tropoelastin, composed of repeating pentapeptide units with the consensus motif $(VPGXG)_n$, where the guest residue X can be systematically varied to scalably produce a form of sequence-controlled macromolecules[16]. ELPs display lower critical solution temperature (LCST) phase behavior in aqueous solution, transitioning from a soluble to an insoluble phase upon heating above a sequence- and environment-dependent transition temperature (Tt). This reversible phase transition has been extensively studied and is known to depend on parameters such as hydrophobicity[17, 18], charge[19, 20], sequence directionality[21], concentration[22], molecular weight[23], the existence of purification tags[24], and solvent conditions[25, 26]. Urry et al found that the identity and hydrophobicity of the guest residue exert a strong influence, with more hydrophobic substitutions (e.g., Ile, Leu) lowering the Tt and promoting aggregation[17].

Cononsolvency has been observed in a variety of synthetic polymers[27] due to the competition between polymer–water, polymer–cosolvent, and cosolvent–water interactions. Several experimental and theoretical studies established that this effect arises from preferential cosolvent adsorption that perturbs the hydration shell, competition between water–cosolvent and

polymer–solvent interactions, and the stronger binding of cosolvent to one polymer state (coil or globule)[28]. In synthetic polymers, charge has been shown to weaken cononsolvency effects. Alenichev et al. demonstrated that introducing ionic comonomers into PNIPAM and poly(N-isopropylmethacrylamide) networks in water/ethanol mixtures shifts or even eliminates the cononsolvency window: negative charges move the transition to higher ethanol content, while positive charges can suppress it entirely[29]. They attributed this to charges stabilizing water–polymer hydrogen bonds, making disruption by alcohol more difficult. Recent studies by Mills *et al.* have revealed that a hydrophobic ELP sequence, similar to many thermoresponsive synthetic polymers[30], also exhibits cononsolvency behavior—a decrease in solubility when mixed in solvent blends. ELP undergoes LCST behavior at low ethanol concentrations but transitions to upper critical solution temperature (UCST) behavior at higher ethanol content[26]. This cononsolvency effect arises from subtle competition between water, cosolvent, and polymer interactions. Mills *et al.* argued that the primary driving forces resemble those in poly(N-isopropylacrylamide) (PNIPAM), yet the dominant hydrogen-bonding sites differ: ELPs form hydrogen bonds along the peptide backbone while alcohol preferentially solvates their hydrophobic side chains, whereas PNIPAM forms hydrogen bonds via its amide side chains and alcohol instead solvates the backbone. Despite extensive work on synthetic polymers, the influence of sequence on cononsolvency behavior has not yet been systematically studied.

In this work, systematic variations in ELP sequence were designed, and the phase behavior of these protein polymers was mapped out using turbidimetry in water, ethanol, and sodium chloride solutions. By designing a series of ELPs with identical number of pentapeptide repeats but varying guest residues, the roles of hydrophobicity and charge in modulating phase behavior are investigated. The resulting phase behavior is compared to that of literature-reported ELP[26] to

understand the effect of sequence in tuning solubility. The effect of 6xHis Tag and flanking peptide sequence on the phase behavior of ELPs is also investigated and compared using thrombin cleavage. Those findings expand the sequence–property map of ELPs and provide fundamental design basis for protein-based materials.

5.3 Materials and Methods

5.3.1 Materials. All solvents and salts used were reagent grade. Water used in all experiments was purified using a Millipore Milli-Q system and passed through a 0.22 μm filter.

5.3.2 Gene Design and Cloning. Seven ELP sequences were ordered to vary hydrophobicity and charge. All genes except for ELP-N were ordered in pET28b(+) from Genscript as shown in supplemental information. ELP-N was ordered in pET28a(+). All genes except for ELP-N were cloned into pET15b with restriction sites NdeI/XhoI using a standard restriction cloning protocol as outlined in previous publications. All sequences were confirmed via Sanger sequencing (Genewiz, USA).

Table 1. ELP sequences designed in the study to reflect the interactions such as hydrophobicity and charge.

Sequence Abbreviation	Guest Residue (X)	Interaction	Mole Fraction of X	Gene Sequence 
ELP-I	Isoleucine	Hydrophobic	0.2	$[(VPGVG)_2VPG\mathbf{I}G(VPGVG)_2]_{10}$
ELP-V	Valine	Hydrophobic	1 (control)	$[(VPGVG)_2VPG\mathbf{V}G(VPGVG)_2]_{10}$
ELP-N	Asparagine	Hydrophilic	0.2	$[(VPGVG)_2VPG\mathbf{N}G(VPGVG)_2]_{10}$
ELP-S	Serine	Hydrophilic	0.2	$[(VPGVG)_2VPG\mathbf{S}G(VPGVG)_2]_{10}$
ELP-G	Glycine	Hydrophilic	0.2	$[(VPGVG)_2VPG\mathbf{G}G(VPGVG)_2]_{10}$
ELP-K	Lysine	Positively Charged	0.2	$[(VPGVG)_2VPG\mathbf{K}G(VPGVG)_2]_{10}$
ELP-D	Aspartic Acid	Negatively Charged	0.2	$[(VPGVG)_2VPG\mathbf{D}G(VPGVG)_2]_{10}$

5.3.3 Protein Expression and Purification. The seven genes were transformed into various *E. coli* strains. The strains for large scale expression were selected based on the ELP yields from a high throughput small scale expression protocol as published previously[31]. ELP-S and ELP-I were transformed into C43(DE3). ELP-K, ELP-N, and ELP-D were transformed into Tuner(DE3). ELP-V and ELP-G were transformed into T7 Express lysY/I^{α}. After sequence confirmation, single colonies were then used to seed starter cultures, which were grown in 50 mL of LB media with corresponding antibiotics (50 μg/mL kanamycin for ELP-N or 100 μg/mL ampicillin for others) in baffled shake-flasks at 37 °C for 8–16 h. The starter culture (50 mL) was used to inoculate 5 L of TB expression media supplemented with 50 μg/mL kanamycin (ELP-N) or 100 μg/mL ampicillin (all other ELPs) depending upon the antibiotic resistance imparted by the selected plasmid. Cells were grown at 37 °C and induced with 1 mM isopropyl-β-D-thiogalactoside (IPTG) once the expression media reached an OD_{600} of 0.6–1.0. After induction, cells were allowed to express protein at 37 °C for an additional 3 hours for ELP-K, 6–9 h for ELP-S, ELP-N, ELP-V, ELP-D, and 20 h for ELP-I. Cells were harvested by centrifugation and frozen at -20 °C until they could be processed further, at least overnight. After thawing, cells were resuspended in 100 mL of lysis buffer (100mM NaH_2PO_4, 10mM Tris Cl, 8M urea, 0.6M NaCl, 20mM beta-mercaptoethanol, pH = 8) for every 30 g of wet cell mass. The resuspended cells were sonicated using a tip sonicator. This lysate was clarified by high-speed centrifugation at 4 °C for 30 mins. Ni-NTA affinity chromatography is chosen as the first round of purification over inverse thermal cycling (ITC) to ensure that ELPs with higher transition temperature can be purified. 20mL Ni-NTA resins were added to the clarified lysate and let the mixture bind overnight at 4°C shaker. The Ni-NTA affinity column was washed with 4 column volume buffer B twice at pH = 8, 7.3, 6.3. The protein was eluted from the column with buffer B at pH = 5.9 twice and 4.3 five times to ensure complete

elution of protein. The purification was confirmed with SDS-PAGE, which can be found in Supplementary Information. The eluted proteins were dialyzed three times in EDTA buffer and five times in MillQ water. The purified proteins were dialyzed three times in EDTA buffer and five times in MillQ water. If the resulting protein still contains impurity, inverse thermal cycling is used for further purification with subsequent dialysis and lyophilization.

5.3.4 Thrombin Cleavage. Thrombin was purchased from PROSPEC BIO (Catalogue Number: PRO-447). Lyophilized proteins were dissolved at 2.5mg/mL in cleavage buffer (20 mM tris, 0.15 M NaCl, 2.5 mM $CaCl_2$, pH = 8.4). Thrombin is added to the solution at 10 U/mg protein, and the mixture was shaken at room temperature for 40-48 hours for complete cleavage based on kinetic study of ELP-S as shown in Figure C-11. Phenylmethylsulfonyl fluoride (PMSF) dissolved in isopropanol at 10mM is added to the solution to achieve 1mM final concentration in order to stop the cleavage reaction[32, 33]. Protein and thrombin were separated by inverse thermal cycling and dialyzed in water six times to remove salt. The purified proteins were then lyophilized and the cleavage was verified with MALDI-TOF as shown in Supplementary Information.

5.3.5 Sample Preparation. Lyophilized proteins were dissolved at 5 mg/mL in MillQ water overnight at 4 °C. Samples were then diluted with a combination of water, alcohol, and 0.5 M sodium chloride or 5 M sodium chloride to bring the solution to the correct solvent content and salt concentration and to bring the total protein concentration in solution to 1 mg/mL.

5.3.6 Turbidimetry. Samples were loaded into a quartz cuvette and capped. Dry air was flown into the holder and around the cuvette to prevent condensation. The cuvette temperature was ramped at a rate of 1°C/min, and transmittance at 600 nm was monitored. Transition temperature was taken as the point where a 50% change in transmittance was reached.[26, 34] Error bars on transition temperatures correspond to standard deviations over three measurements.

5.4 Results and Discussion

The hydrophobic and hydrophilic ELPs exhibit four characteristic regions on the phase diagram: LCST-like transitions at low ethanol concentrations, complete miscibility at low-medium ethanol concentrations, UCST-like transitions at medium ethanol concentrations, and complete miscibility at high ethanol concentrations as shown in Figure 5-1(a-e). At low ethanol concentrations (0-5 mol%), the LCST transition temperature increases as ethanol concentration increases. As the sequence becomes more hydrophilic, the solubility increases more significantly with ethanol addition. As the ethanol concentrations increases up to 10 mol%, the solutions become soluble across all temperature measured from 0-90 °C. At medium ethanol concentrations (12-24 mol%), UCST transitions occur at low temperatures. As the ethanol concentration further increases (25-40 mol%), the solution becomes soluble across a large temperature range again. These differences reflect underlying thermodynamic driving forces**.** In the LCST regime, entropy gain from dehydration and release of structured water drives phase separation via chain collapse into β turns[35-37] or β hairpin[38] structures . However, in the UCST regime, enthalpy-entropy compensation likely governs the transition: ethanol disrupts hydrogen bonding within and around the ELP, and the entropic penalty of restricted chain conformations competes with the enthalpic stabilization from ethanol–polymer interactions.

Hydrophobic ELP sequences have higher UCST transition temperatures than hydrophilic ELP sequences in general. Hydrophilic ELPs exhibit higher solubility in water/ethanol mixtures because their polar residues favor interactions with the polar solvents, stabilizing the chains in solution. Among the hydrophilic sequences, ELP-N exhibits the narrowest two-phase regions and the lowest UCST transitions, indicating good solubility in water ethanol mixtures as shown in Figure 5-2. This is due to the side chain amide group in asparagine being a strong hydrogen bond

donor and acceptor[39], which can solvate ELPs even at various ethanol concentrations and low temperatures. In comparison with the other hydrophilic guest residues, serine has a hydroxyl group and enables hydrogen bonding; however, it does not stabilize the solvated structures as effectively as asparagine due to the shorter side chain and less polarity. Unlike asparagine and serine, glycine lacks polar groups but shows slightly lower UCST transitions in ELP-G than ELP-S. This might be contributing to the chain flexibility with VPGGG[40], which allows entropic contributions to increase solubility than ELP-S.

Consistent with synthetic polymers, charge can suppress the cononsolvency region in ELPs. Both positively and negatively charged ELPs do not exhibit any UCST-like transitions in water/ethanol mixtures. The Debye screening length in 200mM NaCl is 0.68 nm, indicating that the electrostatic repulsions are screened and cannot explain the full miscibility. Therefore, the full miscibility of charged ELPs in all measured solvent conditions is likely due to the hydrophilic nature of those charged amino acids and the large entropic gain arising from counterions being released into solution, promoting polyelectrolyte solubility[41]. This observation is consistent with literature of polyelectrolyte solubility in dilute solution.

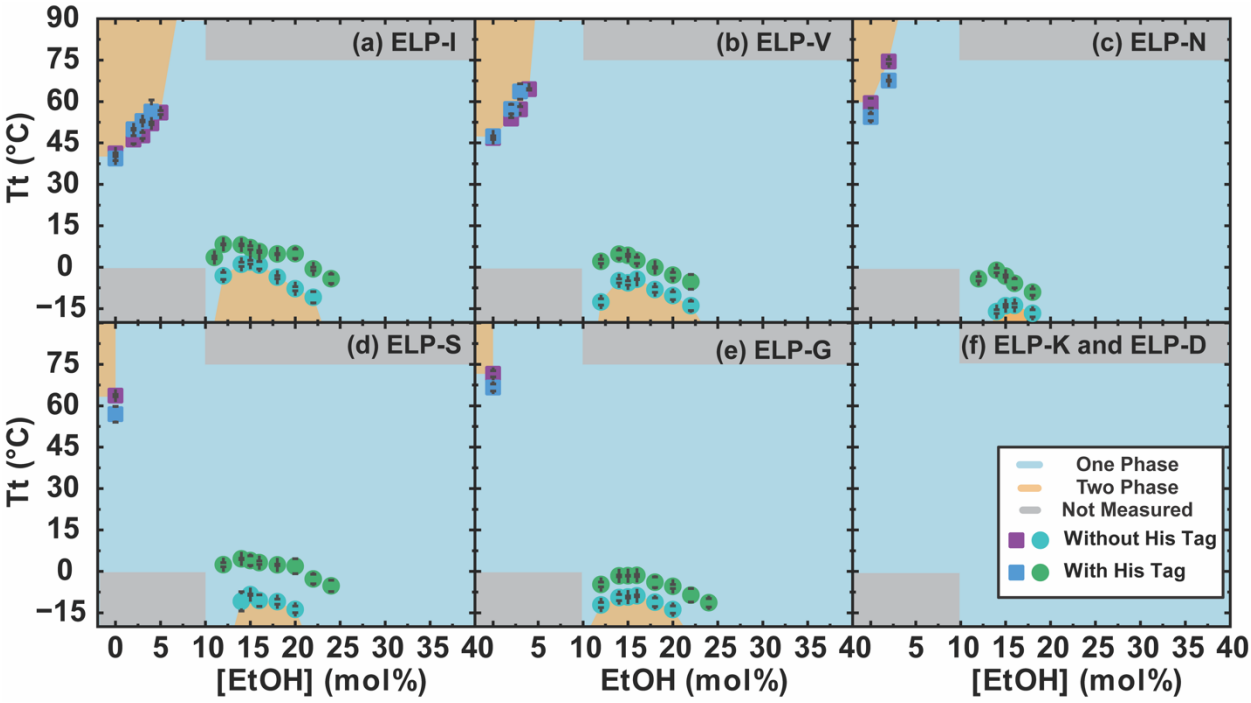


Figure 5-1. The phase diagram of ELPs with 6x His tag and without 6xHis tag in water/ethanol mixtures. The coloring reflects the phase boundaries for without 6X His tag. (a) ELP-I. (b) ELP-V. (c) ELP-N. (d) ELP-S. (e) ELP-G. (f) ELP-K and ELP-D.

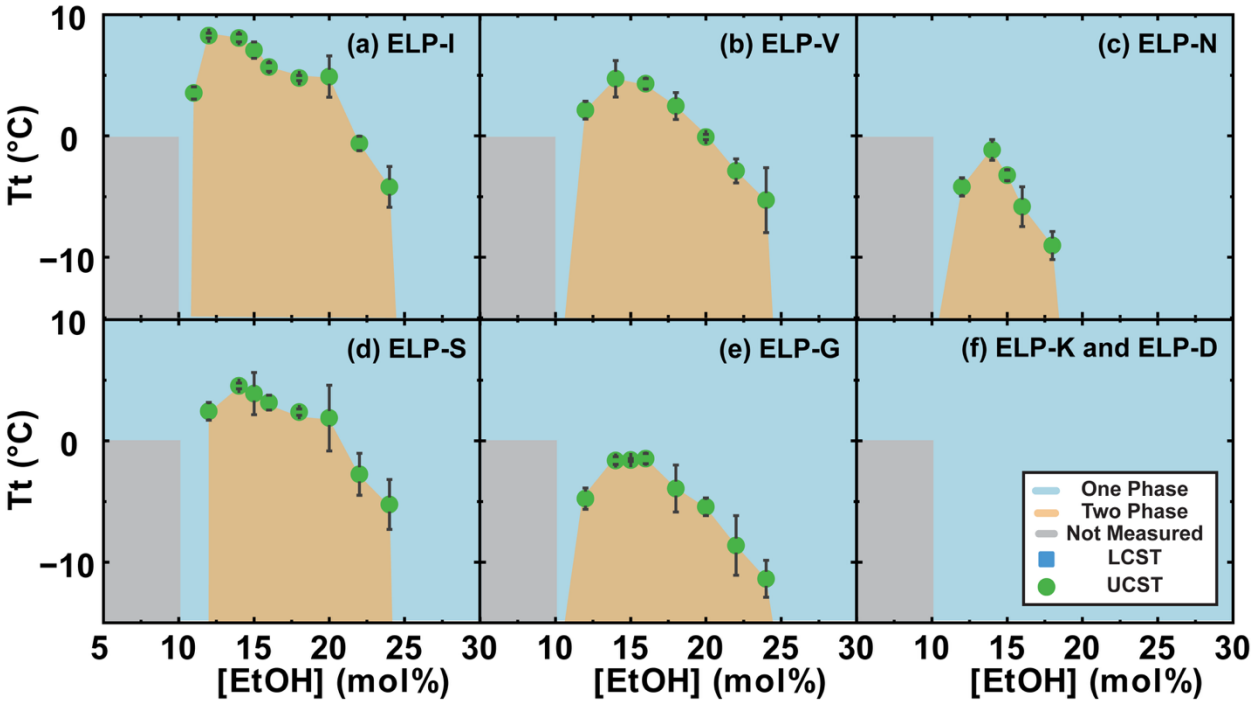


Figure 5-2. Zoomed in phase diagram of ELPs with His tag in the study in water/ethanol mixtures. (a) ELP-I. (b) ELP-V. (c) ELP-N. (d) ELP-S. (e) ELP-G. (f) ELP-K and ELP-D. The grey areas are not measured due to either freezing or significant sample evaporation.

Subtle differences in hydrophobic sequence patterning can significantly alter ELP phase behavior in water and ethanol mixtures. In Figure 5-3(a), both the ELP-I studied in this work and previous studied sequence $[(VPGVG)_2VPGIG(VPGVG)_2]_{50}$ in literature exhibit a phase behavior with both lower and upper critical solution temperature (LCST and UCST) features. As shown in Figure 5-3(a), for ELP-I, at low ethanol concentrations (< 8 mol%), the Tt increases with ethanol. With further ethanol addition, the ELP transitions into a two-phase regime, which narrows and disappears above ~24 mol% ethanol, consistent with UCST-like transitions. Figure 5-3(b) shows a literature-reported ELP, $[(VPGVG)_2IPGVG(VPGVG)_2]_{100}$, that contains the same amino acids but differs in the local arrangement of hydrophobic residues. Unlike ELP-I, the LCST-like

transition temperature slightly decreases as ethanol concentration increases, and transitions to UCST-like transitions around similar temperature range (~30°C).

These results show that change in sequence and molecular weight are crucial in the phase behavior and self-assembly properties in sequence-defined polymers. The lower-molar-mass polymer shows a two-phase region that extends to higher ethanol concentrations. If chain length alone were responsible, the higher-molar-mass polymer would display the broader two-phase region, but that is not observed. This indicates that molar mass cannot fully explain the behavior. Instead, both the positioning of Ile within the repeat unit and the overall chain length shape the phase boundaries. Ile placement (first versus fourth position) alters the distribution of hydrophobicity along the chain, which influences flexibility, hydrogen bonding, and the balance between solvent interactions and chain aggregation. Consistent with this, ELP-I exhibits sharper transitions and a higher transition temperature, suggesting stronger intermolecular interactions arising from its specific sequence. Increasing molar mass lowers the LCST-like transition temperature and can promote UCST-like demixing at high ethanol. The combined effects of sequence patterning and chain length therefore govern the distinct phase behaviors.

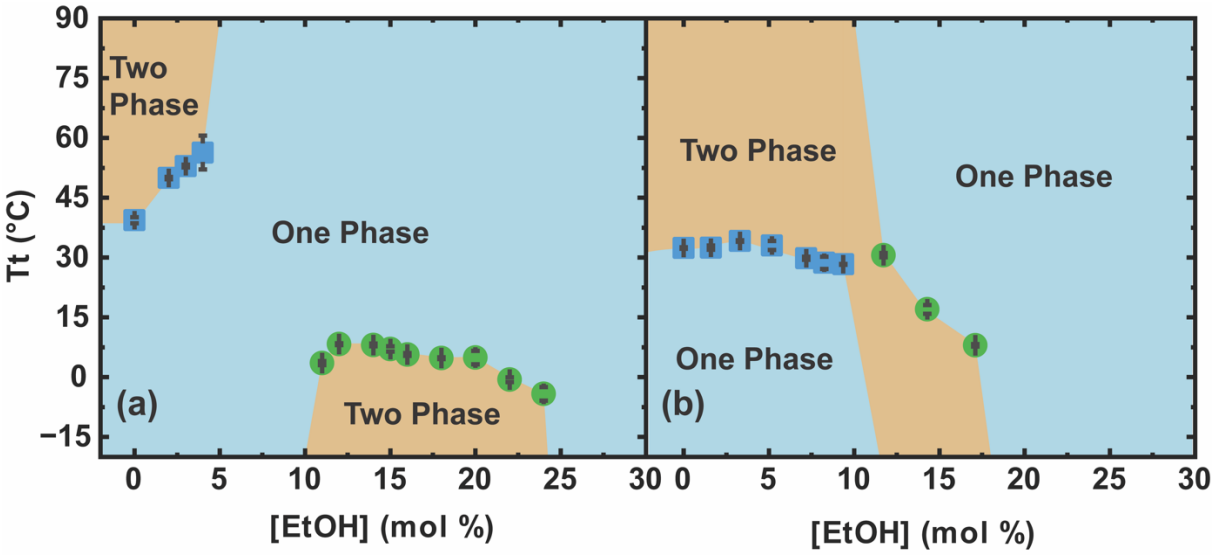


Figure 5-3. Comparison of ELP Phase Diagram in water/ethanol mixtures. (a) ELP-I in this work with sequence $[(VPGVG)_2VPGIG(VPGVG)_2]_{50}$. (b) $[(VPGVG)_2IPGVG(VPGVG)_2]_{100}$ from literature.

The effect of ionic strength on the LCST and UCST-like transitions is further investigated in ternary solutions consisting of water/ethanol/sodium chloride. All sequences remain in one phase with the addition of sodium chloride up to 200 mM at 10 mol%, 30 mol%, and 40 mol% ethanol; UCST transitions only are observed at 20 mol% ethanol in the presence of NaCl. At 10 mol%, the LCST-like transitions are extremely high, likely above experimentally measurable range based on the trend from 0-5 mol%. At 30-40 mol% ethanol solution, the chains are soluble due to alcohol solvation instead of hydrophobic hydration[26]. Increasing sodium chloride concentration did not induce LCST or UCST-like transitions at 10 mol%, 30 mol%, and 40 mol% ethanol solution.

The UCST-like transition temperature generally decreases with increasing sodium chloride concentration in 20 mol% ethanol concentration in both hydrophobic and hydrophilic sequences. UCST-like transitions are generally expected in the intermediate ethanol concentrations (12-24

mol%) and the effect of sodium chloride on the UCST-like transitions was investigated at 20 mol%. As shown in Figure 5-4, the hydrophobic sequences, ELP-I and ELP-V, in 20 mol% ethanol exhibit gradual decrease as sodium chloride concentration increases. The hydrophilic sequences, ELP-N, ELP-S, and ELP-G, shows a gradual decrease at low sodium chloride concentration, and becomes soluble even at -20 °C above 150 mM, 100 mM, 75 mM sodium chloride concentration. For the positively and negatively charged ELPs, no transition is observed in the ternary mixture system, similar to the binary water/ethanol or water/sodium chloride mixtures. This shows that electrostatic repulsions dominate hydrophobic interactions and can effectively solubilize ELPs across a wide range of solution conditions.

Because the 6xHis purification tag and scar peptides from restriction enzyme cloning has a substantially different amino acid composition than the ELP, it may influence the thermodynamic behavior of the ELP polymers. To test this hypothesis, all of the protein polymers were studied both as expressed and with 6xHis purification tag and scar peptides cleaved. Water/ethanol mixtures revealed a modest influence of the 6xHis tag and scar peptides on both LCST- and UCST-like transitions (Figure 5-1). At low ethanol concentrations (0–5 mol%), the transition temperature decreased by 1-2 °C in hydrophobic sequences upon tag removal, likely due to the loss of the charged His motif that enhances solubility. Conversely, some hydrophilic sequences exhibited increased transition temperatures by 5-7 °C after tag cleavage, indicating a slight decrease in net hydrophobicity. Since the entire ELP sequences are already very hydrophilic, the removal of extra hydrophobic scar peptides during thrombin cleavage might dominate the effect of 6x His Tag. At intermediate ethanol concentrations (12–24 mol%), UCST-like transitions were universally depressed in untagged sequences. This observation suggests that the tag itself and the cleaved scar

peptides have low solubility in ethanol-water mixtures and contribute to the overall insolubility of the protein-polymer.

Because the 6xHis tag and scar peptides alter phase behavior in water/ethanol mixtures, their influence was next examined in ternary water/ethanol/sodium chloride solutions. After removal, all sequences remain in one phase with the addition of sodium chloride at 10 mol%, 30 mol%, and 40 mol%, similar to original sequences. This shows that small number of amino acids do not alter the secondary structures and induce any phase transitions. Removing the 6xHis tag and scar peptides lowers the UCST-like transition temperature in mixtures of water, sodium chloride, and ethanol. As shown in Figure 5-1, the phase boundaries of ELPs without 6xHis tags decrease significantly compared to with 6xHis tags. The UCST-like transitions at 20 mol% remain relatively constant as sodium chloride concentration increases for the hydrophobic sequences (ELP-V and ELP-I). The trend of ELP-N with and without 6xHis tags in ternary mixtures is very different. ELP-N without a 6xHis tag, it does not show any UCST-like transitions at 20 mol% even at -20 °C. This increased solubility arises from the amide side chains, which enhance compatibility in the ternary mixtures, combined with the absence of the 6xHis tag and its associated hydrophobic amino acid residues after cleavage. For ELP-S and ELP-G, the two-phase region still exists, but it is much smaller than the region for proteins with 6xHis tags. This shows that incorporating a 6xHis tag and scar peptides can decrease the solubility and hence increases UCST-like transition temperature in water/sodium chloride/ethanol mixtures. Charged ELPs do not show any transitions across the ternary phase diagram conditions measured regardless of whether a 6xHis tag is attached or cleaved.

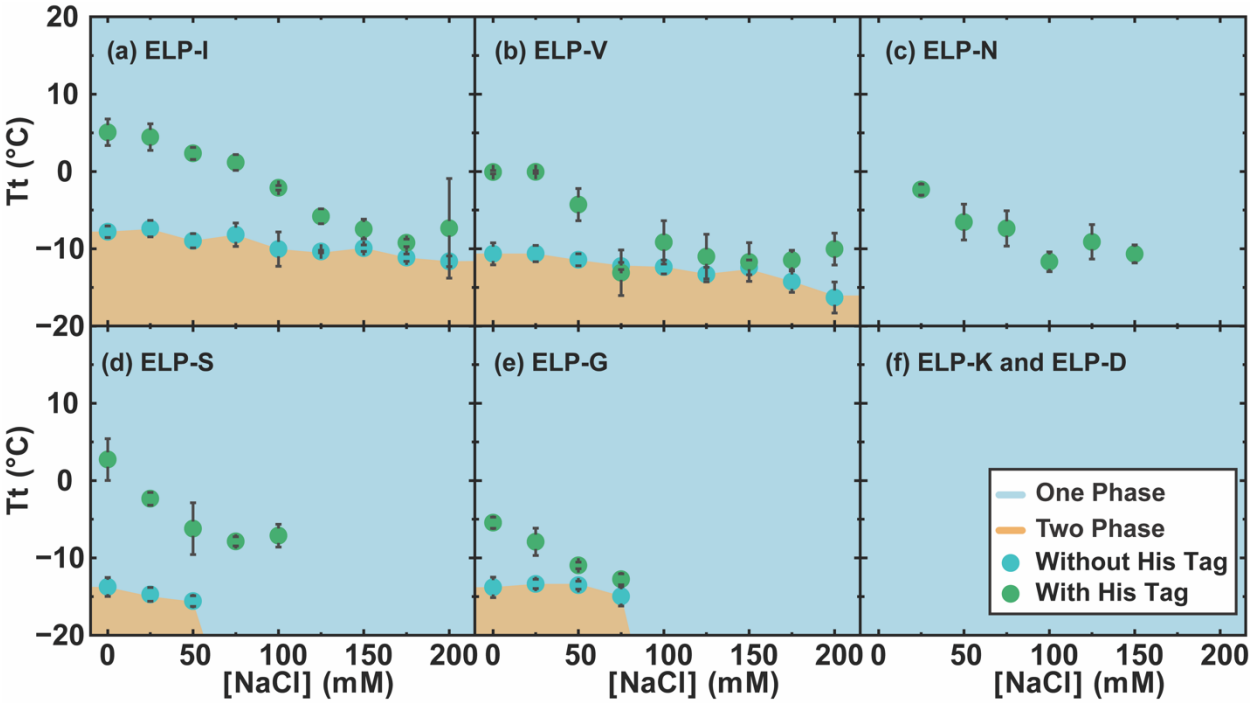


Figure 5-4. The phase diagram of ELPs with 6xHis tag and without 6xHis tag in the study in water/ethanol/sodium chloride mixtures at 20 mol% ethanol. The coloring reflects the phase boundaries for ELPs without 6xHis tag. (a) ELP-I. (b) ELP-V. (c) ELP-N. (d) ELP-S. (e) ELP-G. (f) ELP-K and ELP-D.

5.5 Conclusion

This study systematically examined the effects of primary sequence variations, solvent composition, and salt concentration on the cononsolvency behavior of elastin-like polypeptides (ELPs). Universal cononsolvency behaviors were found in uncharged hydrophobic and hydrophilic ELPs. Sequence variations in both the guest residue and the patterning in the pentapeptides in ELPs can govern both lower critical solution temperature (LCST)-type transitions in aqueous solutions and upper critical solution temperature (UCST)-like behavior in water/ethanol mixtures.

Charged ELPs, including ELP-K and ELP-D, remained fully soluble under all solvent and salt conditions, highlighting the dominant role of electrostatic interactions in suppressing phase separation. Increasing sodium chloride concentration generally decreased both LCST and UCST-like transition temperatures, with more pronounced effects observed for hydrophilic sequences. The presence of a 6xHis-tag caused a modest elevation of UCST-like transition temperatures, likely due to additional hydrophobic and electrostatic interactions from the tag sequence. Removal of the His-tag improved solubility, particularly in ternary water/ethanol/salt systems.

Overall, these findings demonstrate universal cononsolvency behavior across diverse ELP sequences and elucidate how change in sequence, side chain chemistry, and terminal modifications can be strategically utilized to control solubility and phase behavior. The insights provided by this study contribute to a broader understanding of sequence–solvent–structure relationships in intrinsically disordered biopolymers and inform the rational design of bioinspired polymeric materials with programmable phase properties.

5.6 Acknowledgement

We thank the Department of Energy Office of Basic Energy Sciences (Grant: DE-SC0007106) for supporting this research. We thank the MIT Koch Biopolymers & Proteomics for their assistance with MALDI-TOF measurements.

5.7 References

1. DeStefano, A. J.; Mengel, S. D.; Bates, M. W.; Jiao, S.; Shell, M. S.; Han, S.; Segalman, R. A., Control over Conformational Landscapes of Polypeptoids by Monomer Sequence Patterning. *Macromolecules* **2024,** *57* (4), 1469-1477.
2. Jin, T.; Coley, C. W.; Alexander-Katz, A., Sequence-Sensitivity in Functional Synthetic Polymer Properties. *Angewandte Chemie International Edition* **2025,** *64* (2), e202415047.
3. O'Toole, E. M.; Panagiotopoulos, A. Z., Effect of sequence and intermolecular interactions on the number and nature of low-energy states for simple model proteins. *The Journal of Chemical Physics* **1993,** *98* (4), 3185-3190.
4. Hartmann, L.; Börner, H. G., Precision Polymers: Monodisperse, Monomer-Sequence-Defined Segments to Target Future Demands of Polymers in Medicine. *Advanced Materials* **2009,** *21* (32-33), 3425-3431.
5. Perry, S. L.; Sing, C. E., 100th Anniversary of Macromolecular Science Viewpoint: Opportunities in the Physics of Sequence-Defined Polymers. *ACS Macro Letters* **2020,** *9* (2), 216-225.
6. Anfinsen, C. B., Principles that Govern the Folding of Protein Chains. *Science* **1973,** *181* (4096), 223-230.
7. Pääbo, S.; Gifford, J. A.; Wilson, A. C., Mitochondrial DNA sequences from a 7000-year old brain. *Nucleic Acids Research* **1988,** *16* (20), 9775-9787.
8. Bhattacharyya, R. P.; Reményi, A.; Yeh, B. J.; Lim, W. A., Domains, Motifs, and Scaffolds: The Role of Modular Interactions in the Evolution and Wiring of Cell Signaling Circuits. *Annual Review of Biochemistry* **2006,** *75* (Volume 75, 2006), 655-680.
9. Meier, M. A. R.; Barner-Kowollik, C., A New Class of Materials: Sequence-Defined Macromolecules and Their Emerging Applications. *Advanced Materials* **2019,** *31* (26), 1806027.
10. Buehler, M. J.; and Ackbarow, T., Nanomechanical strength mechanisms of hierarchical biological materials and tissues. *Computer Methods in Biomechanics and Biomedical Engineering* **2008,** *11* (6), 595-607.
11. Xie, O.; Olsen, B. D., A Self-Consistent Field Theory Formalism for Sequence-Defined Polymers. *Macromolecules* **2022,** *55* (15), 6516-6524.
12. Patel, R. A.; Borca, C. H.; Webb, M. A., Featurization strategies for polymer sequence or composition design by machine learning. *Molecular Systems Design & Engineering* **2022,** *7* (6), 661-676.
13. Webb, M. A.; Jackson, N. E.; Gil, P. S.; de Pablo, J. J., Targeted sequence design within the coarse-grained polymer genome. *Science advances* **2020,** *6* (43), eabc6216.
14. Curtis, K. A.; Statt, A.; Reinhart, W. F., Predicting self-assembly of sequence-controlled copolymers with stochastic sequence variation. *Soft Matter* **2025,** *21* (11), 2143-2151.
15. Lutz, J.-F.; Ouchi, M.; Liu, D. R.; Sawamoto, M., Sequence-Controlled Polymers. *Science* **2013,** *341* (6146), 1238149.
16. Yang, Y. J.; Holmberg, A. L.; Olsen, B. D., Artificially Engineered Protein Polymers. *Annu Rev Chem Biomol Eng* **2017,** *8*, 549-575.
17. Urry, D. W.; Gowda, D. C.; Parker, T. M.; Luan, C.-H.; Reid, M. C.; Harris, C. M.; Pattanaik, A.; Harris, R. D., Hydrophobicity scale for proteins based on inverse temperature transitions. *Biopolymers* **1992,** *32* (9), 1243-1250.
18. Urry, D. W.; Luan, C. H.; Parker, T. M.; Gowda, D. C.; Prasad, K. U.; Reid, M. C.; Safavy, A., Temperature of polypeptide inverse temperature transition depends on mean residue hydrophobicity. *Journal of the American Chemical Society* **1991,** *113* (11), 4346-4348.

19. Meyer, D. E.; Chilkoti, A., Genetically Encoded Synthesis of Protein-Based Polymers with Precisely Specified Molecular Weight and Sequence by Recursive Directional Ligation: Examples from the Elastin-like Polypeptide System. *Biomacromolecules* **2002,** *3* (2), 357-367.
20. MacKay, J. A.; Callahan, D. J.; FitzGerald, K. N.; Chilkoti, A., Quantitative Model of the Phase Behavior of Recombinant pH-Responsive Elastin-Like Polypeptides. *Biomacromolecules* **2010,** *11* (11), 2873-2879.
21. Li, N. K.; Roberts, S.; Quiroz, F. G.; Chilkoti, A.; Yingling, Y. G., Sequence directionality dramatically affects LCST behavior of elastin-like polypeptides. *Biomacromolecules* **2018,** *19* (7), 2496-2505.
22. Meyer, D. E.; Chilkoti, A., Quantification of the effects of chain length and concentration on the thermal behavior of elastin-like polypeptides. *Biomacromolecules* **2004,** *5* (3), 846-851.
23. McDaniel, J. R.; Radford, D. C.; Chilkoti, A., A Unified Model for De Novo Design of Elastin-like Polypeptides with Tunable Inverse Transition Temperatures. *Biomacromolecules* **2013,** *14* (8), 2866-2872.
24. Trabbic-Carlson, K.; Meyer, D. E.; Liu, L.; Piervincenzi, R.; Nath, N.; LaBean, T.; Chilkoti, A., Effect of protein fusion on the transition temperature of an environmentally responsive elastin-like polypeptide: a role for surface hydrophobicity? *Protein Engineering, Design and Selection* **2004,** *17* (1), 57-66.
25. Zhao, Y.; Singh, M. K.; Kremer, K.; Cortes-Huerto, R.; Mukherji, D., Why do elastin-like polypeptides possibly have different solvation behaviors in water–ethanol and water–urea mixtures? *Macromolecules* **2020,** *53* (6), 2101-2110.
26. Mills, C. E.; Ding, E.; Olsen, B. D., Cononsolvency of elastin-like polypeptides in water/alcohol solutions. *Biomacromolecules* **2019,** *20* (6), 2167-2173.
27. Cortez-Lemus, N. A.; Licea-Claverie, A., Poly(N-vinylcaprolactam), a comprehensive review on a thermoresponsive polymer becoming popular. *Progress in Polymer Science* **2016,** *53*, 1-51.
28. Bharadwaj, S.; Niebuur, B.-J.; Nothdurft, K.; Richtering, W.; van der Vegt, N. F. A.; Papadakis, C. M., Cononsolvency of thermoresponsive polymers: where we are now and where we are going. *Soft Matter* **2022,** *18* (15), 2884-2909.
29. Alenichev, I.; Sedláková, Z.; Ilavský, M., Swelling and mechanical behavior of charged poly(N-isopropylmethacrylamide) and poly(N-isopropylacrylamide) networks in water/ethanol mixtures. Cononsolvency effect. *Polymer Bulletin* **2007,** *58* (1), 191-199.
30. Schild, H. G.; Muthukumar, M.; Tirrell, D. A., Cononsolvency in mixed aqueous solutions of poly(N-isopropylacrylamide). *Macromolecules* **1991,** *24* (4), 948-952.
31. Morris, M. A.; Bataglioli, R. A.; Mai, D. J.; Yang, Y. J.; Paloni, J. M.; Mills, C. E.; Schmitz, Z. D.; Ding, E. A.; Huske, A. C.; Olsen, B. D., Democratizing the rapid screening of protein expression for materials development. *Molecular Systems Design & Engineering* **2023**.
32. Kaminski, M.; McDONAGH, J., Inhibited thrombins. Interactions with fibrinogen and fibrin. *Biochemical Journal* **1987,** *242* (3), 881-887.
33. Rick, M. E.; Hoyer, L. W., Thrombin activation of factor VIII: the effect of inhibitors. *British Journal of Haematology* **1977,** *36* (4), 585-597.
34. Yao, H. Driving forces of self-assembly in protein-polymer bioconjugates. Massachusetts Institute of Technology, 2020.
35. Urry, D. W., Molecular Machines: How Motion and Other Functions of Living Organisms Can Result from Reversible Chemical Changes. *Angewandte Chemie International Edition in English* **1993,** *32* (6), 819-841.

36. Urry, D. W., Physical Chemistry of Biological Free Energy Transduction As Demonstrated by Elastic Protein-Based Polymers. *The Journal of Physical Chemistry B* **1997,** *101* (51), 11007-11028.
37. Urry, D. W.; Trapane, T. L.; Sugano, H.; Prasad, K. U., Sequential polypeptides of elastin: cyclic conformational correlates of the linear polypentapeptide. *Journal of the American Chemical Society* **1981,** *103* (8), 2080-2089.
38. Prhashanna, A.; Taylor, P. A.; Qin, J.; Kiick, K. L.; Jayaraman, A., Effect of Peptide Sequence on the LCST-Like Transition of Elastin-Like Peptides and Elastin-Like Peptide–Collagen-Like Peptide Conjugates: Simulations and Experiments. *Biomacromolecules* **2019,** *20* (3), 1178-1189.
39. Yokota, A.; Tsumoto, K.; Shiroishi, M.; Nakanishi, T.; Kondo, H.; Kumagai, I., Contribution of asparagine residues to the stabilization of a proteinaceous antigen-antibody complex, HyHEL-10-hen egg white lysozyme. *J Biol Chem* **2010,** *285* (10), 7686-96.
40. Zhao, Y.; Bharadwaj, S.; Myers, R. L.; Okur, H. I.; Bui, P. T.; Cao, M.; Welsh, L. K.; Yang, T.; Cremer, P. S.; van der Vegt, N. F., Solvation Behavior of Elastin-like Polypeptides in Divalent Metal Salt Solutions. *The Journal of Physical Chemistry Letters* **2023,** *14* (45), 10113-10118.
41. G. Lopez, C.; Matsumoto, A.; Shen, A. Q., Dilute polyelectrolyte solutions: recent progress and open questions. *Soft Matter* **2024,** *20* (12), 2635-2687.

Chapter 6 Commensurability and Equilibrium Testing in Hard Sphere-Driven Phase Transitions in Coarse-Grained Simulations of Globular Protein–Polymer Block Copolymers

6.1 Abstract

A coarse-grained model using molecular-dynamics simulations was used to map the equilibrium phase behavior of globular protein–polymer block copolymers, taking mCherry-b-PNIPAM as a model system. The globular domain is modelled as a hard sphere, and the polymer blocks as soft spheres. Phase diagrams computed for three hard-soft potentials $U_{12}(0)$ across coil fractions 0.15–0.55 and volume fractions 0.05–0.45 reproduce all experimentally reported nanostructures—disordered, weakly ordered, hexagonally packed cylinders (HEX) and lamellae (LAM)—except perforated lamellae. Larger $U_{12}(0)$ values promotes ordering and kinetic arrest. Kinetically arrested regions identified by mean-square-displacement criteria. Commensurability of periodic structures was addressed using NPT ensemble with optimized pressure within 1% of the target concentration. For all lamellae phases, the percent difference in domain spacing between simulations performed in the NPT and NVT ensembles ranges from no difference to a maximum of 0.7%. For all hexagonally packed cylinder phases, the percent difference in domain spacing between simulations performed in the NPT and NVT ensembles ranges from 1.48% to a maximum of 12%. Qualitative concentration dependence of lamellar spacing observed in simulations mirrors experimental SAXS data. Different initializations (HEX→LAM, LAM→HEX, perforated LAM) and replica exchange simulations confirm the equilibrium structures and identify the concentrated weakly ordered state as an entropically driven, kinetically trapped configuration that cannot be annealed by temperature alone, consistent with observations in experiments. The model therefore captures both equilibrium and non-equilibrium features of protein–polymer conjugates and offers a first principle-based model for engineering targeted morphologies.

6.2 Introduction

Globular protein–polymer bioconjugates in which a folded protein is covalently linked to a synthetic polymer—extend the design space of soft materials beyond traditional synthetic block copolymers. By combining a compact, immiscible, and often rigid protein domain with a solvophilic/solvophobic polymer, these conjugates can self-assemble into nanostructures reminiscent of block-copolymer phases while preserving protein functionality. Model systems such as mCherry–poly(N-isopropylacrylamide) (mCherry-b-PNIPAM) demonstrate this promise: concentrated solutions and cast films exhibit disordered states, weakly ordered morphologies, and long-range-ordered phases such as lamellae and hexagonally packed cylinders, with the protein's tertiary structure retained within the nanophase-separated domains. These experimental observations motivate predictive, unified, and computationally efficient tools that can map how composition, concentration, and interaction strength control morphology and understanding the driving force of self-assembly in those systems.

Despite growing experimental insight, a quantitative, physics-based framework for phase prediction in protein–polymer conjugates remains underdeveloped. Direct atomistic simulation is prohibitive at the system sizes and timescales required to capture ordering, defect annealing, and kinetic trapping at high concentration. Mean-field field-theoretic methods excel for flexible polymers but struggle to represent the excluded-volume and shape constraints of a finite-sized, rigid globular domain without uncontrolled approximations[1]. Huang *et al.* analyzed a library of 15 protein–PNIPAM conjugates (comprising 11 distinct globular proteins tethered to PNIPAM) and looked for correlations between protein properties and self-assembly behavior[2]. Interestingly, despite the diverse nature of the proteins, many attributes of phase behavior were *similar across all conjugates* – for example, all showed disordered micellar phases at low concentration/high

temperature and lamellar ordering at intermediate conditions. This suggested that a few key variables might govern the behavior. By performing a regression analysis, the authors found that the protein's molecular weight and its secondary structure content (particularly the fraction of residues in β-sheets) were strong predictors of the quality of ordering in the conjugate. As a result, there is a gap between empirical structure–property correlations and a tractable, physics-grounded model that can both recover equilibrium phase behavior and diagnose nonequilibrium arrest.

A coarse-grained particle description offers a practical route to close this gap while retaining the essential physics. Representing the globular protein as a hard sphere and the polymer blocks as soft spheres reduces the dimensionality of the problem yet preserves the key ingredients for mesoscale self-assembly: immiscibility and excluded volume, chain-mediated elasticity, and composition and concentration control. Within this minimal representation, a tunable hard–soft interaction parameter, $U_{12}(0)$, compactly captures the effective incompatibility between the protein and polymer domains and serves as a knob to traverse ordering regimes.

In this work, coarse-grained molecular-dynamics simulations of globular protein–polymer block copolymers, using mCherry-b-PNIPAM as a representative system, were used to map equilibrium and arrested morphologies across coil fraction and volume fraction. Phase diagrams are constructed for multiple values of $U_{12}(0)$ spanning coil fractions 0.15–0.55 and volume fractions 0.05–0.45. Within this space, the model reproduces the experimentally reported nanostructures—disordered, weakly ordered, hexagonally packed cylinders (HEX), and lamellae (LAM)—with the exception of perforated lamellae. Increasing $U_{12}(0)$ strengthens segregation, expands ordered regions, and increases susceptibility to kinetic arrest.

Because boxes size can strongly influence self-assembly of periodic structures, simulations are performed in the isothermal–isobaric ensemble with pressures optimized to achieve target

concentrations within 1%. Equilibrium assignment is further validated with multiple initializations (HEX→LAM, LAM→HEX, perforated LAM seeding) and with replica-exchange to test equilibrium structure. Mean-square-displacement criteria delineate kinetically arrested states, revealing concentrated, weakly ordered configuration that is entropically stabilized yet cannot be annealed by temperature alone—behavior consistent with experimental observations.

Together, these results establish a first-principles, minimal Coarse-Grained framework that captures both equilibrium phase selection and nonequilibrium trapping in protein–polymer conjugates. This model offers a practical map for engineering targeted morphologies and a foundation for a generalizable model that incorporate anisotropic protein shapes, sequence-specific polymer effects, and solvent selectivity.

6.3 Computational Details

6.3.1 Simulation in Isothermal-Isobaric (NPT) Ensemble

Following approaches from the literature to address the commensurability challenge in simulating mesoscale periodic structures[3], the isothermal–isobaric (NPT) ensemble was employed to validate the box size and commensurability in the previous simulations, which had been performed in the canonical (NVT) ensemble. In the NPT ensemble, the concentration can fluctuate depending on thermal fluctuations. To achieve the target concentration with in 1%, the target pressure is varied based on bisection during equilibration run. Under template.in, the NPT ensemble is performed by

```
fix baro all npt temp 1.0 1.0 $(0.1*v_tLJ) aniso
###TARGET_PRESSURE### ###TARGET_PRESSURE### $(1.0*v_tLJ) couple
xy
```

`$(0.1*v_tLJ)` is the damping of temperature, which is 100 times more than the time step, where

`timestep $(0.001*v_tLJ)`

###TARGET_PRESSURE### are pressure input values replaced by the density_bisect_working.py. The `aniso` and `couple xy` allows the pressure in x and y direction to be equal during simulation since commensurability is tested along the z axis.

`$(1.0*v_tLJ)` is the damping with respect to pressure, set to 1000 times the timestep based on LAMMPS documentation.

Once the target pressure has been determined, the same pressure is passed to the production run under in.WCAcollgaussNPT.

```
fix baro all npt temp 1.0 1.0 $(100*v_dt) aniso 3.3125 3.3125
$(1000*v_dt) couple xy
```

where the pressure value `3.3125` vary depending on the $U_{12}(0)$, coil fraction, and radius of gyration of the polymer (R_{g2}).

Equilibration generally required 500,000–2,000,000 time steps, depending on the size of the system, after which morphologies and potential energies remained stable for the production run. Trajectories were saved into XYZ dump files and visualized using Ovito. Ovito was also used to slice selected simulation snapshots and calculate the radial distribution function.

The simulations with different initializations were performed the same as outlined in this section. To use the different initialized file during equilibration, the read_data was changed to the filename of desired python-built initial structures.

6.3.2 Domain Spacing Calculations

Structure factor was calculated using a MATLAB code provided in Appendix using the dump files during production run. The domain spacing of the structures are calculated as

$$d = \frac{2\pi}{q*} \tag{6.1}$$

Where q* is the q value of the primary peak from the structure factor.

6.3.4 Replica Exchange with Biased Initial Structure

Some systems are stuck in the biased initial structure during different initialization running at T=1.0. For those systems, replica exchange is performed to anneal those kinetically trapped structures using the restart.txt file from production. Replica exchange was performed using the *temper/npt* command in LAMMPS (available from the REPLICA package). Temperatures were distributed with the ratio between adjacent temperatures T_1/T_0 = 1.01. Additional temperature replicas are inserted or deleted depending on the acceptance rate. Replicas were distributed across a temperature range of 1.00 to 1.85 or 2.0, which was designed to give an acceptance probability of ≥ 20% for all swapping pairs[4]. The acceptance probability was monitored after running one sets of replica exchange simulations in NPT for 500,000 steps. Potential energy histograms were built after each stage and monitored for change in energy. Once the energy reached a minimum (i.e. the histograms stopped drifting downward), an isothermal-isobaric simulation with WCA and Gaussian potentials was run at T = 1.00 for 2,000,000 steps using the trajectories recorded in the restart file for the replica corresponding to this temperature.

6.4 Results and Discussion

6.4.1 Phase Behavior and Analysis

Four types of nanostructures—disordered, hexagonally packed cylinders, lamellae, weakly ordered—were observed across simulations run with the three intermediate values of $U_{12}(0)$, as

shown in the phase diagrams in Figure 6-1. Even with such Coarse-Grained modeling and simulation approach, those phases correspond to the experimental nanostructures that have been observed in mCherry-PNIPAM, indicating good selection of force field and parameterization. The only phase that is not observed in this model is the perforated lamellae phase, which does not form even when initializing with perforated lamellae for the phase boundaries between hexagonally packed cylinders and lamellae (Section 6.3). As the potential $U_{12}(0)$ increases, the number of ordered phases increases on the phase diagram, which indicates that the self-assembled nanostructures from microphase separation are favored when the repulsive interaction between the hard sphere and the soft spheres are strong. Even at low concentrations, the system can self-assemble into either weakly ordered phase or the lamellae phase.

As $U_{12}(0)$ increases, the regions of kinetic arrest based on MSD analysis also increases as shown in Figure 6-1(a-c). The MSD analysis allows studying the dynamics of molecules from MD simulation trajectories. If the MSD computed from Equation 2.49 is linear with a slope of $1 \pm$ the fluctuation from the low concentration base line and R^2 value of 1 with respect to τ, the lag time, the system is ergodic. This shows that it can freely explore its configurational landscape. If those criteria are not satisfied, it indicates that the system is in high probability of kinetically arrest. For those coil fraction and concentrations, further testing for equilibrium is followed on the lamellae and the hexagonally packed cylinders.

The entropic mechanism of the reentrant ODT in simulation suggests that this highly concentrated weakly ordered phase is a kinetically trapped state. Since the formation of the concentrated weakly ordered phase is driven by entropy, temperature cannot anneal out the kinetically trapped state, and this is consistent with observations in experiments. This is manifested in the model with replica exchange. When initialized from the weakly disordered phase, replica

exchange shows good acceptance rate but the only structure that is present in the replicas are the weakly ordered phases, which again validates that the model also can model similar effect in experiments. When initialized from the biased lamellar phase, the only structure that is present in the replicas are the lamellae phases, even up to T = 3.0. Transferring matrix from clustering analysis also shows that only small fraction of the molecules (<10% on average) is transitioning from the cluster to a different cluster.

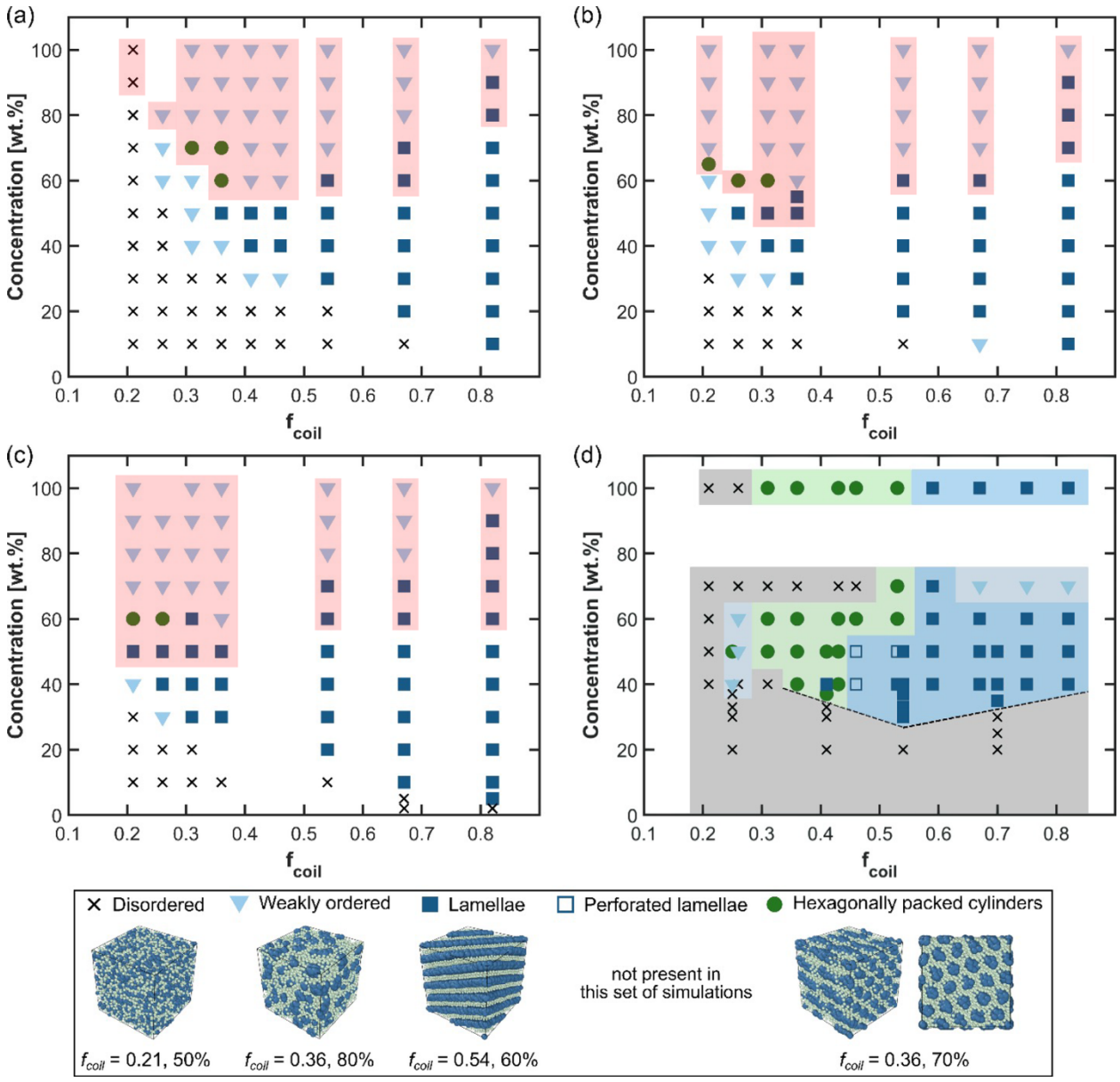


Figure 6-1. Phase diagrams from MD simulations obtained using $U_{12}(0)$ of (a) $20k_BT$, (b) $50k_BT$, and (c) $100k_BT$. The experimental phase diagram for mCherry-*b*-PNIPAM bioconjugates obtained from SAXS experiments[5, 6] is shown in panel (d). The simulation snapshots are taken from the $U_{12}(0) = 20k_BT$ phase diagram. The shaded red regions in (a-c) represent regions of potential kinetic arrest based on Mean Square Displacement (MSD) analysis.

6.4.2 Clustering Analysis

Clustering analysis was performed to evaluate molecular mobility and directional transitions (cluster hopping) in the simulations. The average number of molecules that transition was computed and normalized by the total number of molecules in the simulations. The results were consistent with the MSD analysis except for $f_{coil} = 0.82$ with $U_{12}(0) = 50k_BT$ and 100 k_BT. This was due to interconnected lamellar layers that formed under high $U_{12}(0)$ and coil fractions. The cluster analysis regarded the interconnected lamellae as a big cluster and resulted in very few small clusters.

As concentration increased, cluster hopping became less frequent, reflected by decreases in the fraction of molecules transitioning. At even higher concentrations, almost all molecules remained localized, with only a few hopping.

In the weakly ordered phase, the number of cluster transitions increased slightly. At these high concentrations, tightly packed molecules caused small clusters to merge or disappear, leading to a modest increase in the fraction of molecules that transitioned. Overall, the clustering analysis aligned with the MSD analysis in distinguishing kinetically trapped states. However, it also revealed that most of kinetically trapped molecules are not completely immobilized at lower coil fractions. As the coil fraction increased, the fraction of molecules transitioning between clusters decreased and dropped to nearly zero at coil fractions at the highest concentrations.

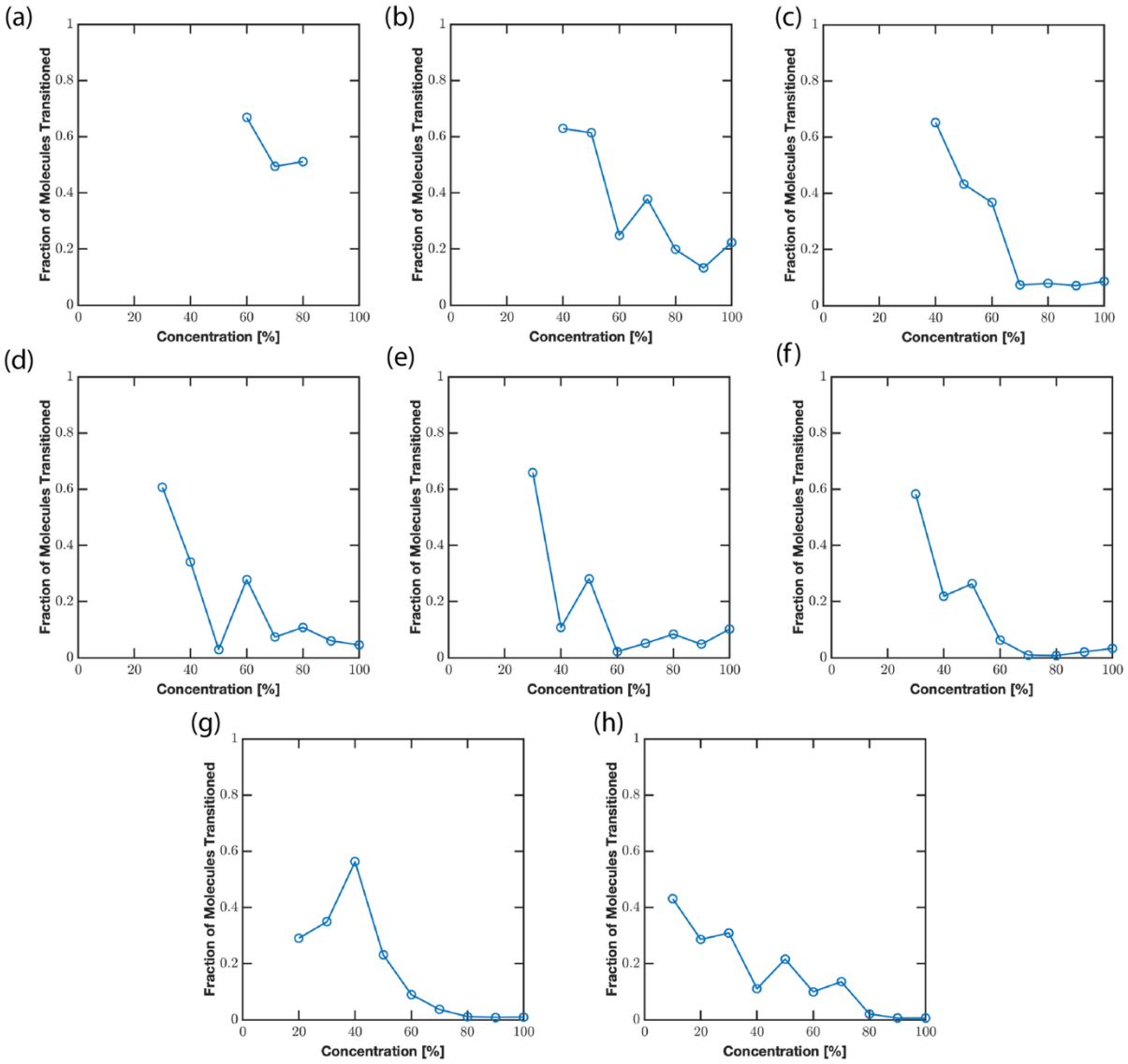


Figure 6-2. The average fraction of total number of molecules that transitioned between clusters with $U_{12}(0) = 20k_BT$ and a coil fraction of (a) 0.26, (b) 0.31, (c) 0.36, (d) 0.41, (e) 0.46, (f) 0.54, (g) 0.67, and (h) 0.82.

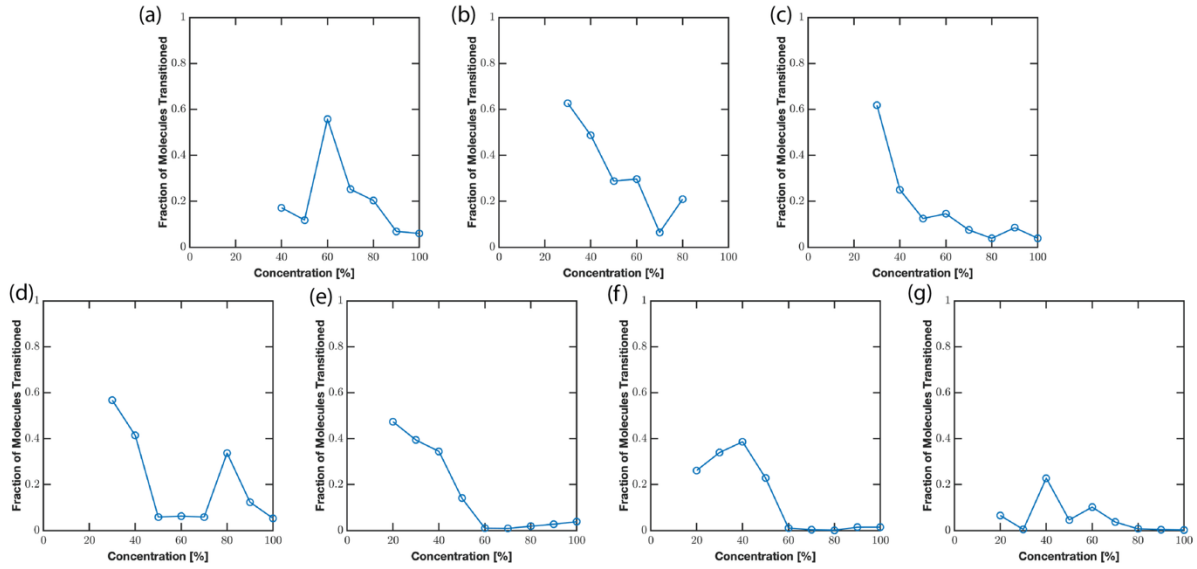


Figure 6-3. The average fraction of total number of molecules that transitioned between clusters with $U_{12}(0) = 50k_BT$ and a coil fraction of (a) 0.21, (b) 0.26, (c) 0.31, (d) 0.36, (e) 0.54, (f) 0.67, and (g) 0.82.

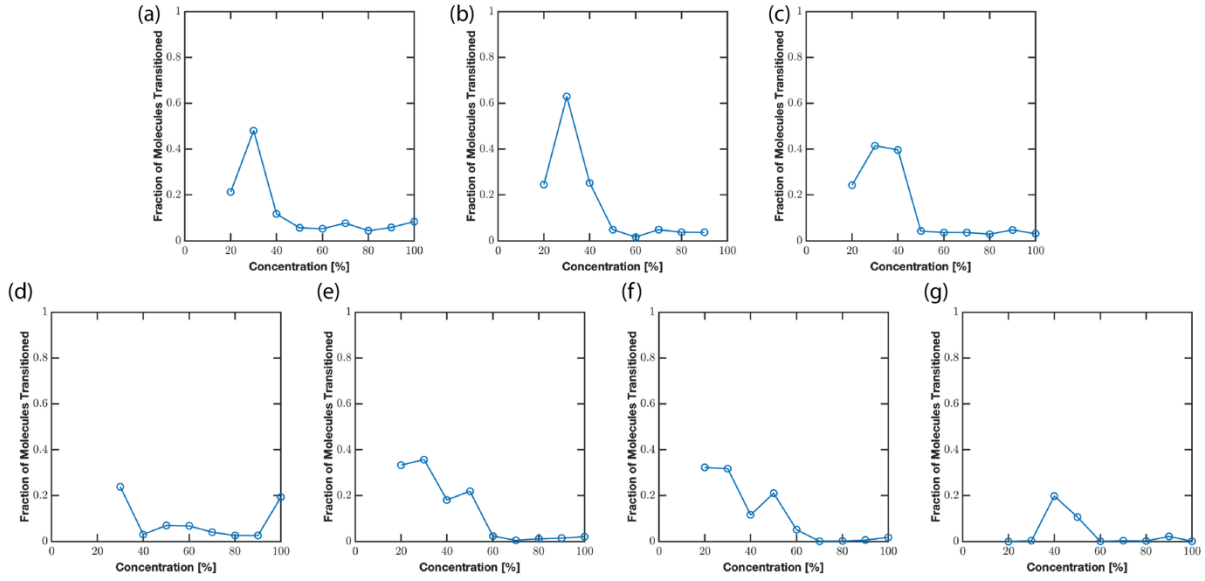


Figure 6-4. The average fraction of total number of molecules that transitioned between clusters with $U_{12}(0) = 100k_BT$ and a coil fraction of (a) 0.21, (b) 0.26, (c) 0.31, (d) 0.36, (e) 0.54, (f) 0.67, and (g) 0.82.

For all lamellar phases, the percent difference in domain spacing between simulations performed in the NPT and NVT ensembles ranges from no difference to a maximum of 0.7%. The structure factors are also nearly identical (Figure 6-5), where the peaks are all present and close in their the q value position. This suggests that the simulation box choices in the NVT ensemble were commensurate with the periodic lamellar structure. In the NPT simulations, the lamellae were aligned with the z-axis, whereas in the NVT ensemble they were free to rotate. The small observed differences therefore arise from the ability of lamellae in the NVT ensemble to reorient and adjust their internal structure to achieve commensurability.

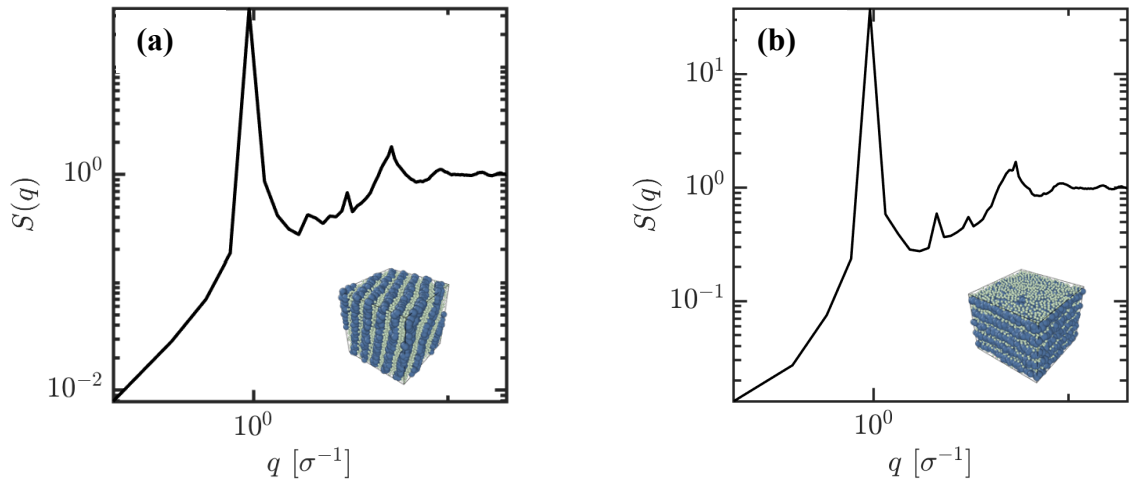


Figure 6-5. Representative Structure Factors of Lamellae Phase from (a) NVT and (b) NPT ensemble. Structure factor of $U_{12}(0) = 20k_BT$, $f_{coil} = 0.54$, 50 wt%.

The qualitative concentration dependence of lamellar spacing observed in simulations mirrors experimental SAXS data as shown in Figure 6-6. For the mCherry–PNIPAM bioconjugate with $f_{coil} = 0.67$, experiments show that the domain spacing remains constant from 40 to 60 wt%, increases slightly at 70 wt%, and then decreases at 100 wt%. Simulations at $U_{12}(0) = 100k_BT$ and $f_{coil} = 0.67$ reproduce the same non-monotonic trend: spacing is unchanged between 30 and 40

wt%, increases slightly at 50 wt%, and decreases at 70 wt%. The absolute spacings differ quantitatively between experiment and simulation, which is expected due to several factors: the coarse-grained length scale σ may not map directly to experimental dimensions; $U_{12}(0)$ is treated as state-independent despite changes in solvent quality and polymer conformation; implicit solvent neglects hydration effects that influence PNIPAM packing; and the protein is modeled as a hard sphere rather than the anisotropic mCherry structure. These approximations shift absolute values but preserve the essential physics of concentration-dependent self-assembly, suggesting that mesoscale packing constraints, rather than fine details of solvent quality or interaction strength, dominate the structural response of these bioconjugates.

Commensurability effects are more pronounced for hexagonally packed cylinders than for lamellae because cylinders require a two-dimensional hexagonal lattice to fit within a rectangular simulation box. For all hexagonally packed cylinder phases, the percent difference in domain spacing between simulations performed in the NPT and NVT ensembles ranges from 1.48% to a maximum of 12%, even though the structure factors show the same number of peaks (Figure 6-7). Small mismatches in box dimensions introduce significant strain or defects, as the hexagonal symmetry cannot be easily reconciled with orthogonal box vectors. By contrast, lamellae only require periodicity along one axis, and slight mismatches can be accommodated through layer tilting or minor compression, making them far less sensitive to box commensurability.

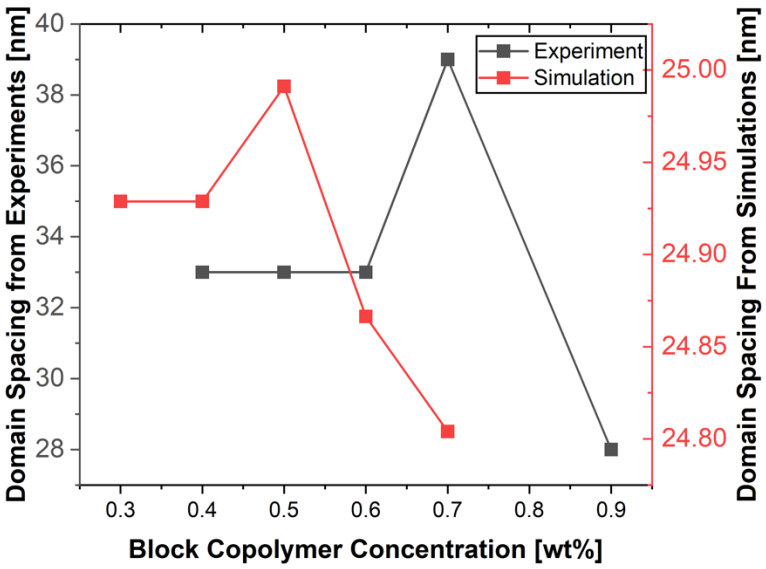


Figure 6-6. Comparison of experiment and simulation result of domain spacing as a function of block copolymer concentrations[5].

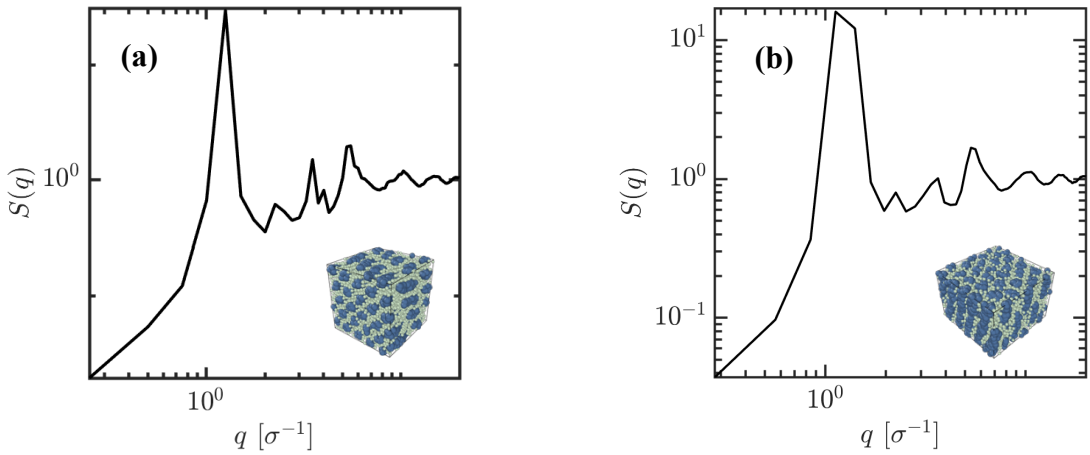


Figure 6-7. Representative Structure Factors of Hexagonally Packed Cylinder Phase from (a) NVT and (b) NPT ensemble. Structure factor of $U_{12}(0) = 20k_BT$, $f_{coil} = 0.36$, 70 wt%.

6.4.2 Confirming Equilibrium Structures

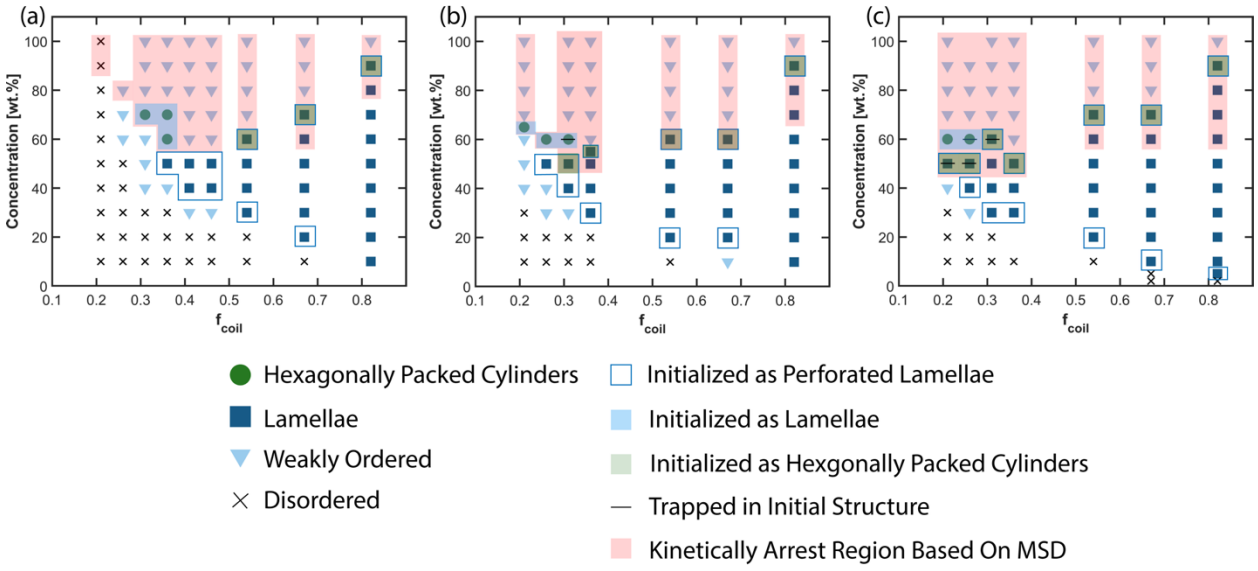


Figure 6-8. Phase Diagram illustrating different initialization.

For all the phase boundaries of the lamellae equilibrium structures, perforated lamellae were built based on the concentration required as initial structure and simulations were performed in NPT ensemble. All the tested points on the phase diagram (hollow blue squares in Figure 6-8) evolved to lamellae phases within equilibration, indicating that lamellae phase is a more favorable configuration than the perforated lamellae under this coarse-grained model.

The higher $U_{12}(0)$ is, the harder it becomes for the lamellae phase to transition to hexagonally packed cylinders. All hexagonally packed cylinder phases in Figure 6-8 were initialized with lamellae structures to test for equilibrium in NPT ensemble. At $U_{12}(0) = 20k_BT$, all three points transitioned to hexagonally packed cylinders from lamellae. At $U_{12}(0) = 50k_BT$, two out of three points transitioned. At $U_{12}(0) = 100k_BT$, one out of two points transitioned. For the points that did not transition (with the dash), they have higher coil fraction than the ones that transitioned with the same $U_{12}(0)$. Replica exchange is used for those to sample the configurational landscape effectively and escape the energy local minima.

All the initialized hexagonally packed cylinders transitioned to lamellae phases at $U_{12}(0) = 20k_BT$ and $50k_BT$ even for the systems close to the phase boundaries. At $U_{12}(0) = 100k_BT$, the systems that were trapped in their initial hexagonally packed cylinder structures were those close to the hex/lam phase boundaries due to the proximity in energy between the two phases. Therefore, replica exchange was used to anneal those systems and escape the energy minimum.

6.5 Conclusion

Coarse-grained molecular-dynamics simulations of hard–soft dumbbells—representing a globular protein as a hard sphere and the polymer blocks as soft spheres—capture the essential equilibrium and nonequilibrium features of globular protein–polymer block copolymers exemplified by mCherry-b-PNIPAM. Modified Weeks–Chandler–Andersen and Gaussian interactions were parameterized from experiments, colloid–polymer theory, and soft-sphere polymer simulations, with a single hard–soft incompatibility scale, $U_{12}(0)$, serving as a practical knob. Phase diagrams constructed across coil fractions 0.15–0.55 and volume fractions 0.05–0.45 reproduce the experimentally observed morphologies—disordered, weakly ordered, hexagonally packed cylinders (HEX), and lamellae (LAM)—while, consistent with experiment, spherical and inverse phases are absent, and perforated lamellae do not emerge. Increasing $U_{12}(0)$ expands ordered regions and simultaneously increases susceptibility to kinetic arrest.

Commensurability was achieved by running in the isothermal–isobaric ensemble and tuning pressure to match target concentrations within 1%. Equilibrium assignment was cross-validated by initializing from competing structures and by replica-exchange simulations to promote barrier crossing. These checks consistently converged to the same equilibrium morphologies where ordering is thermodynamically favored.

At high concentration, simulations recover the re-entrant order–disorder transition reported across mCherry-based conjugates. Mean-square-displacement criteria is used for identifying kinetic arrest regions and concentrated weakly ordered state is entropically stabilized and cannot be annealed by temperature alone. This identifies a kinetically trapped basin intrinsic to dense protein–polymer conjugates and explains processing-dependent ordering observed experimentally.

Taken together, the minimal hard–soft dumbbell model—controlled by composition, concentration, and a single interaction scale $U_{12}(0)$—provides a first principles-based framework for predicting phase selection and diagnosing kinetic trapping in protein–polymer conjugates. Its ability to recover the asymmetric, LAM-dominated phase map while locating re-entrant weak ordering underscores the utility of coarse-grained, physics-grounded simulation for complex macromolecular assemblies. The ability of this simple model to reproduce the most important features of the experimental phase diagram illustrate the power of coarse-grained simulation for studying complex macromolecular systems that are governed by colloidal interactions.

6.6 Acknowledgement

This work was funded by the Department of Energy Office of Basic Energy Sciences Neutron Scattering Program (award number DE-SC0007106). The simulations in this work were run using the LAMMPS platform. We thank San Diego Supercomputer Center Expanse high performance cluster for running the simulations. We acknowledge Dr. Helen Yao on her previous work on this project and thank her for the collaboration and the helpful advice. We thank Dr. Herry Jin for the helpful discussion and introduction to Molecular Dynamics Simulation.

6.7 References

1. Koski, J.; Chao, H.; Riggleman, R. A., Field theoretic simulations of polymer nanocomposites. *The Journal of chemical physics* **2013,** *139* (24).

2. Huang, A.; Paloni, J. M.; Wang, A.; Obermeyer, A. C.; Sureka, H. V.; Yao, H.; Olsen, B. D., Predicting Protein–Polymer Block Copolymer Self-Assembly from Protein Properties. *Biomacromolecules* **2019,** *20* (10), 3713-3723.
3. Ko, M. J.; Kim, S. H.; Jo, W. H., A Molecular Dynamics Simulation on the Self-Assembly of ABC Triblock Copolymers, 1. Effects of Block Composition in Symmetric Triblock Copolymers. *Macromolecular Theory and Simulations* **2001,** *10* (4), 381-388.
4. Sugita, Y.; Okamoto, Y., Replica-exchange molecular dynamics method for protein folding. *Chemical Physics Letters* **1999,** *314*, 141-151.
5. Thomas, C. S.; Olsen, B. D., Coil fraction-dependent phase behaviour of a model globular protein–polymer diblock copolymer. *Soft Matter* **2014,** *10* (17), 3093-3102.
6. Lam, C. N.; Olsen, B. D., Phase transitions in concentrated solution self-assembly of globular protein–polymer block copolymers. *Soft Matter* **2013,** *9* (8), 2393-2402.

Chapter 7 Conclusions

7.1 Summary

This thesis establishes how monomer-level sequence design governs the self-assembly, phase behavior, and structural properties of sequence-defined polymers, spanning from elastin-like polypeptides (ELPs) biopolymer to globular protein–polymer bioconjugates. A high-throughput, information-driven SANS workflow was developed to accelerate structural characterization by estimating the minimum number of counts required for parameter estimation and model differentiation with user defined uncertainty level. Using three representative polymer systems, the workflow demonstrated that accurate structural insights can be achieved with significantly reduced neutron exposure, and that weighted least squares fitting and information criteria (AIC, BIC) remain reliable tools even under count-limited conditions. This approach offers a practical framework for optimizing beamtime, enabling rapid screening of material libraries without compromising data quality.

Beyond characterization advances, this thesis elucidates how sequence variations in ELP triblock copolymers encode distinct thermoresponsive pathways. Hydrophobicity in midblock changes—from isoleucine to asparagine—led to dramatically different self-assembly behaviors, with SI75 forming syneresing micelles and SN75 evolving into a percolated gel only at high temperatures. These differences were quantitatively linked to changes in nanostructure, water partitioning, and rheology using SANS and complementary techniques.

Expanding this design space, a systematic study of ELP sequence libraries revealed universal cononsolvency behavior across uncharged sequences and emphasized the critical role of hydrophobic patterning, guest residue chemistry, and purification tags, in tuning both LCST and

UCST-type transitions. These results establish design rules for controlling phase behavior in sequence-defined polymers through monomer-level precision.

Finally, to address systems of greater architectural complexity, a coarse-grained dumbbell model was constructed to simulate the self-assembly of protein–polymer bioconjugates. Despite its simplicity, the model captured key features of experimental phase diagrams, including lamellar dominance, the suppression of highly curved morphologies, and a reentrant weakly ordered phase arising from hard-sphere packing. These results highlight the utility of minimal models in capturing the essential physical principles govern self-assembly in protein-polymer bioconjugates.

Collectively, this work advances both the theoretical understanding and practical tools required to interrogate sequence–structure–property relationships in soft materials. By bridging experiment and simulation across a broad spectrum of sequence-defined polymer architectures, it provides a foundation for the rational design of responsive biomaterials, high-throughput screening strategies, and coarse-grained models for complex self-assembling systems.

7.2 Outlook

The convergence of precision polymer design, high-throughput structural characterization, and coarse-grained simulation provides a powerful foundation for decoding and controlling the phase behavior of sequence-defined soft materials. This thesis has demonstrated how changes in primary sequence—whether in recombinant polypeptides or protein–polymer conjugates—can dictate macroscopic properties such as solubility, self-assembly, and rheological behavior. It also introduced practical tools, including an accelerated SANS workflow and a minimal coarse-grained model, that are broadly applicable across materials systems. Building on these insights, several promising avenues emerge for future exploration.

First, the high-throughput SANS framework developed here opens the door for large-scale screening of polymer libraries with minimal beamtime investment. One area can be to integrate this algorithm with existing SANS machine learning models to perform model selection when the SANS structural model is unknown. As machine learning continues to gain traction in materials design, integrating this workflow with existing machine learning model[1-4] could enable closed-loop experiments for identifying polymers with targeted structural features or phase behaviors. The bootstrapping and model selection methods employed here provide a statistical backbone for quantifying uncertainty—an essential element for trustworthy experimental automation in neutron science.

Second, the coarse-grained dumbbell model used to study protein–polymer bioconjugates alone can capture many key features of their phase behavior. However, the current model cannot predict perforated lamellae phase, which was found in experimental phase diagram[5]. To enable true predictive modeling, future versions of this framework should incorporate more complex and chemically specific interactions—electrostatics[6], hydrogen bonding[7, 8], protein surface patchiness[9], and sequence-dependent stiffness or secondary structure[10]. This is especially critical when studying systems involving charged residues, polar solvents, or functional ligands. Incorporating these interactions in a systematic and interpretable way will be necessary for generalizing phase predictions across protein–polymer conjugates of varying chemistries.

Third, a key challenge in bridging simulation and experiment is establishing reliable mappings between interaction parameters in coarse-grained models and experimentally accessible quantities. Effective interaction parameters—such as Flory–Huggins χ—are often inferred indirectly and depend sensitively on system conditions such as solvent, temperature, or

concentration. It has been found that in ELPs, the group contribution method tends to underestimate χ, and causing the models failing to predict the targeted morphologies or phases[11]. Moreover, χ might not be able to capture the different in monomer-level interactions between different hydrophobic patterns[12]. Experimental determination of these quantities typically requires model fitting to scattering data, thermodynamic measurements, or comparison to known phase diagrams, each with inherent uncertainties. Developing robust protocols for inferring effective interaction parameters from experiment or constructing hybrid data-driven potentials constrained by experiment, will be essential for making sequence defined polymer models quantitatively predictive.

Finally, this work contributes to a growing body of evidence that sequence-level design in intrinsically disordered polymers can encode programmable, emergent behavior[13, 14]. Moving forward, combining consensus-based protein engineering with block copolymer-inspired architectures may yield hybrid materials that couple biological function with tailored mechanical, optical, or transport properties[15]. The ability to tune phase transitions, micelle architecture, and solvent organization via precise sequence edits holds promise for developing responsive hydrogels, tissue scaffolds, and drug delivery carriers.

In summary, the tools and concepts presented in this thesis represent important steps toward a unified framework for understanding and engineering sequence–structure–property relationships in complex soft materials. By continuing to bridge the gaps between experiment, theory, and data-driven modeling, the field is poised to make substantial progress in rational polymer design and functional material discovery.

7.3 References

1. Do, C.; Chen, W.-R.; Lee, S., Small angle scattering data analysis assisted by machine learning methods. *MRS Advances* **2020,** *5* (29), 1577-1584.
2. Doucet, M.; Samarakoon, A. M.; Do, C.; Heller, W. T.; Archibald, R.; Tennant, D. A.; Proffen, T.; Granroth, G. E., Machine learning for neutron scattering at ORNL. *Machine Learning: Science and Technology* **2020,** *2* (2), 023001.
3. Kanazawa, T.; Asahara, A.; Morita, H., Accelerating small-angle scattering experiments with simulation-based machine learning. *Journal of Physics: Materials* **2020,** *3* (1), 015001.
4. Song, G.; Porcar, L.; Boehm, M.; Cecillon, F.; Dewhurst, C.; Le Goc, Y.; Locatelli, J.; Mutti, P.; Weber, T. In *Deep learning methods on neutron scattering data*, EPJ Web of Conferences, EDP Sciences: 2020; p 01004.
5. Thomas, C. S.; Olsen, B. D., Coil fraction-dependent phase behaviour of a model globular protein–polymer diblock copolymer. *Soft Matter* **2014,** *10* (17), 3093-3102.
6. Lam, C. N.; Yao, H.; Olsen, B. D., The effect of protein electrostatic interactions on globular protein–polymer block copolymer self-assembly. *Biomacromolecules* **2016,** *17* (9), 2820-2829.
7. Anfinsen, C. B., Principles that Govern the Folding of Protein Chains. *Science* **1973,** *181* (4096), 223-230.
8. Dill, K. A., Theory for the folding and stability of globular proteins. *Biochemistry* **1985,** *24* (6), 1501-1509.
9. Almeida, F. C. L.; Sanches, K.; Pinheiro-Aguiar, R.; Almeida, V. S.; Caruso, I. P., Protein Surface Interactions-Theoretical and Experimental Studies. *Front Mol Biosci* **2021,** *8*, 706002.
10. Huang, A.; Paloni, J. M.; Wang, A.; Obermeyer, A. C.; Sureka, H. V.; Yao, H.; Olsen, B. D., Predicting Protein–Polymer Block Copolymer Self-Assembly from Protein Properties. *Biomacromolecules* **2019,** *20* (10), 3713-3723.
11. Xie, O.; Olsen, B. D., A Self-Consistent Field Theory Formalism for Sequence-Defined Polymers. *Macromolecules* **2022,** *55* (15), 6516-6524.
12. O'Toole, E. M.; Panagiotopoulos, A. Z., Effect of sequence and intermolecular interactions on the number and nature of low-energy states for simple model proteins. *The Journal of Chemical Physics* **1993,** *98* (4), 3185-3190.
13. Oldfield, C. J.; Dunker, A. K., Intrinsically disordered proteins and intrinsically disordered protein regions. *Annual review of biochemistry* **2014,** *83* (1), 553-584.
14. Wu, K.; Jiang, H.; Hicks, D. R.; Liu, C.; Muratspahić, E.; Ramelot, T. A.; Liu, Y.; McNally, K.; Kenny, S.; Mihut, A., Design of intrinsically disordered region binding proteins. *Science* **2025,** *389* (6757), eadr8063.
15. Chang, M. P.; Huang, W.; Mai, D. J., Monomer-scale design of functional protein polymers using consensus repeat sequences. *Journal of Polymer Science* **2021,** *59* (22), 2644-2664.

Appendix A Supporting Information for Chapter 3

A.1 Error Scaling Analysis

The uncertainties (dI) in the low q and high q region decrease as $\frac{1}{\sqrt{Number\ of\ counts}}$ and in the mid q region do not follow this scaling as analyzed in Figure A-1. The scaling relationship only applies to uncertainty or error related to counting. The deviation of uncertainties from this relationship in the mid q region can be attributed to more significant error propagation from merging in the dark current, sensitivity, background, and propagation of error from binning[1].

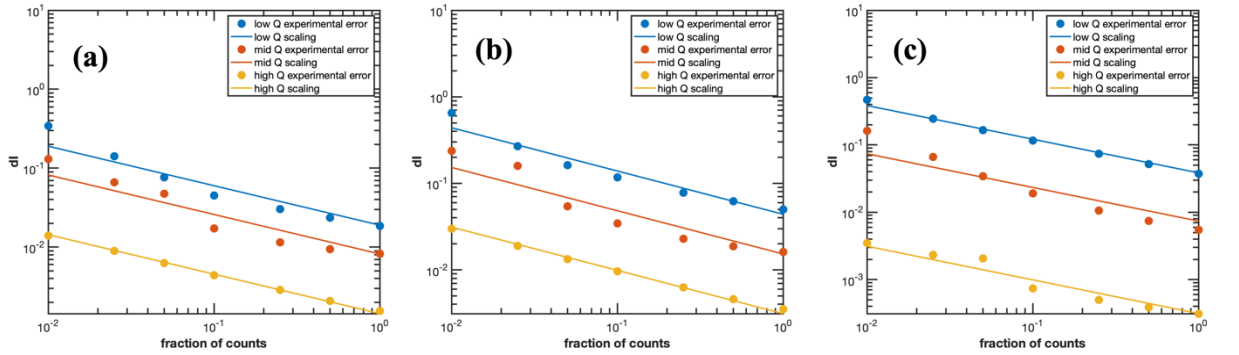


Figure A-1. The error outputted from SANS reduction file at selected q values for (a) 16mM PEG solution. (b) P4 protein hydrogel. (c) Pluronic F-127. Low q value corresponds to q = 0.0151). Mid q corresponds to q = 0.03999. High q corresponds to q = 0.250.

A.2 Fisher Information Matrix

The Broad Peak Model

The fisher information matrix of low q fitting of the broad peak model is

$$\begin{bmatrix} 5.018 * 10^{13} & 4.894 * 10^{9} \\ 4.894 * 10^{9} & 4.782 * 10^{5} \end{bmatrix}$$

Where the rows and columns represent A and n respectively. The eigenvalues are $5.018*10^{13}$ and 954.0.

The Fisher information matrix of the full fitting of broad peak model by fixing *A* and *n* is

$$\begin{bmatrix} 2.717 * 10^5 & -5.925 * 10^3 & 1.635 * 10^6 & -2.052 * 10^3 \\ -5.925 * 10^3 & 7.817 * 10^3 & -3.738 * 10^4 & 348.6 \\ 1.635 * 10^6 & -3.738 * 10^4 & 1.781 * 10^7 & -1.752 * 10^4 \\ -2.052 * 10^3 & 348.6 & -1.752 * 10^4 & 34.27 \end{bmatrix}$$

Where the rows and columns represent *C, m,* q_0, ξ respectively. The eigenvalues are $1.796*10^7$, $1.206*10^5$, $7.695*10^3$, 3.489.

A.3 Correlation Coefficient and Function Analysis

An alternative hypothesis of larger variance of parameter distribution in MC bootstrapping is that the errors associated with intensities are spatially correlated. Analysis on the correlation of the residual of intensities at each q value shows the assumption of MC bootstrapping on independent Gaussian distribution at each q value is valid. The Pearson correlation coefficient does not show any correlation coefficient greater than 0.5 as shown in Figure A-2. The correlation function calculated from both low q and high q range spikes at $\Delta q = 0$ and returns to zero as the distance Δq increases. This is strong evidence of no linear correlation in intensity between neighboring q values for SANS data as shown in Figure A-3. From the correlation coefficient and the correlation function analysis, the Gaussian distribution assumption of MC bootstrapping holds and is not the source of higher variance in parameter distribution.

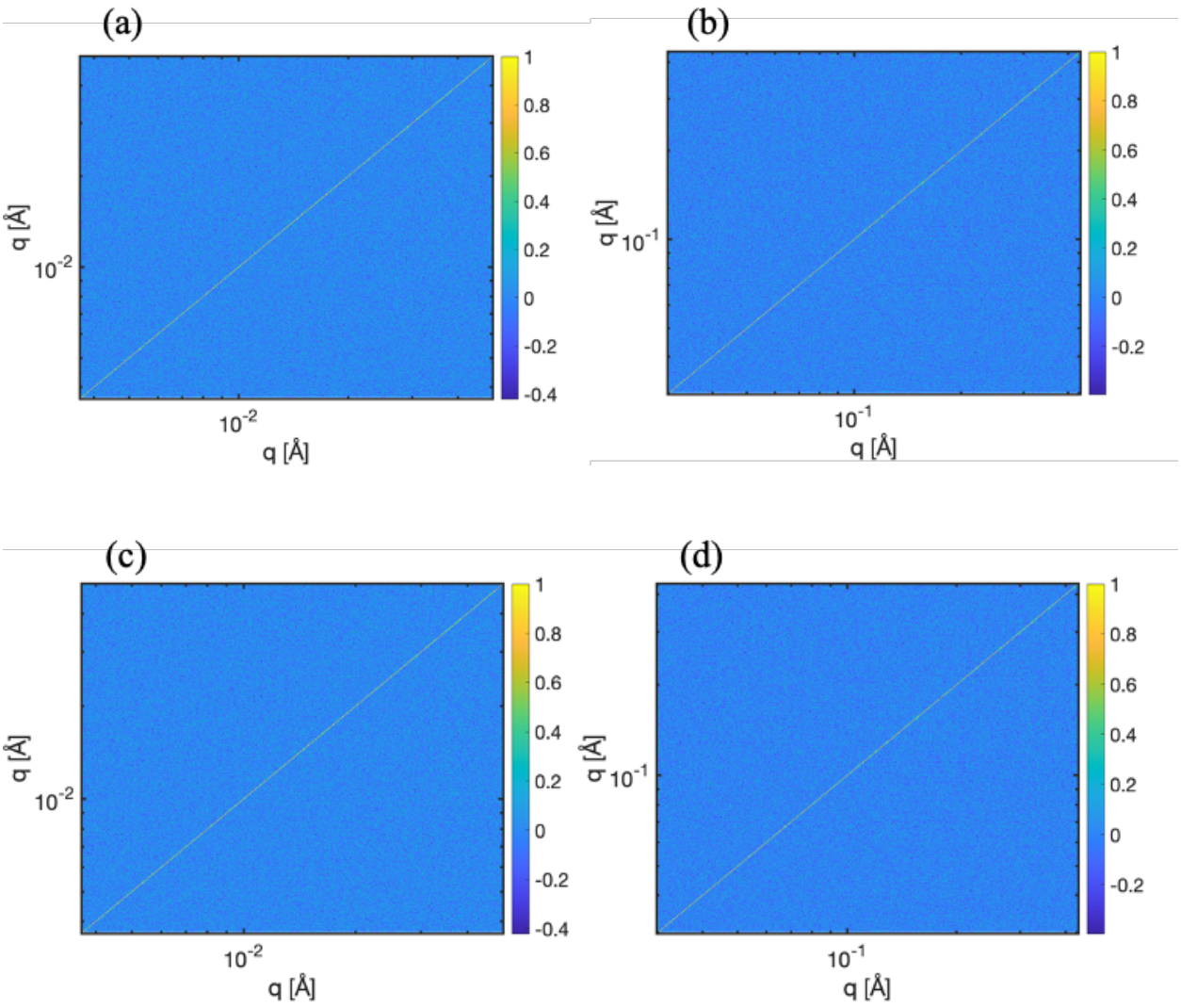


Figure A-2. Correlation coefficient of *dI* for (a) P4 protein hydrogel at low q region. (b) P4 protein hydrogel at high q region. (c) Pluronic F-127 at low q region. (d) Pluronic F-127 at high q region.

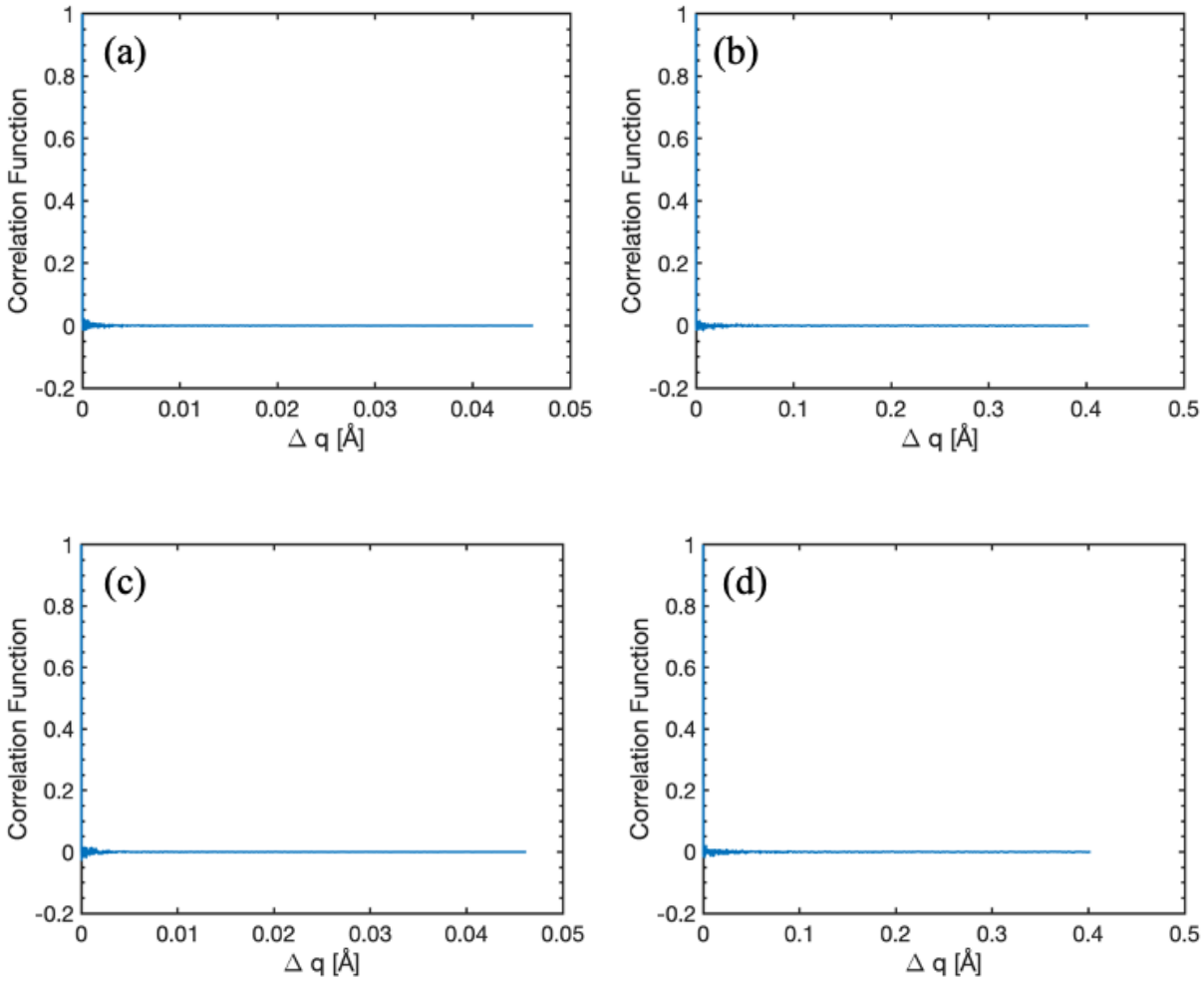


Figure A-3. Correlation function G(Δq) of Δ*I* for (a) P4 protein hydrogel at low q region. (b) P4 protein hydrogel at high q region. (c) Pluronic F-127 at low q region. (d) Pluronic F-127 at high q region.

A.4 Monte Carlo Bootstrapping

From simulated SANS result using MC bootstrapping in Figure A-4, parameter estimation can be achieved down to 0.005 fraction of total counts with error within 5% tolerance of the parameter obtained from full counts. Parameter estimation can be achieved within 5% error tolerance from reduced counts SANS data.

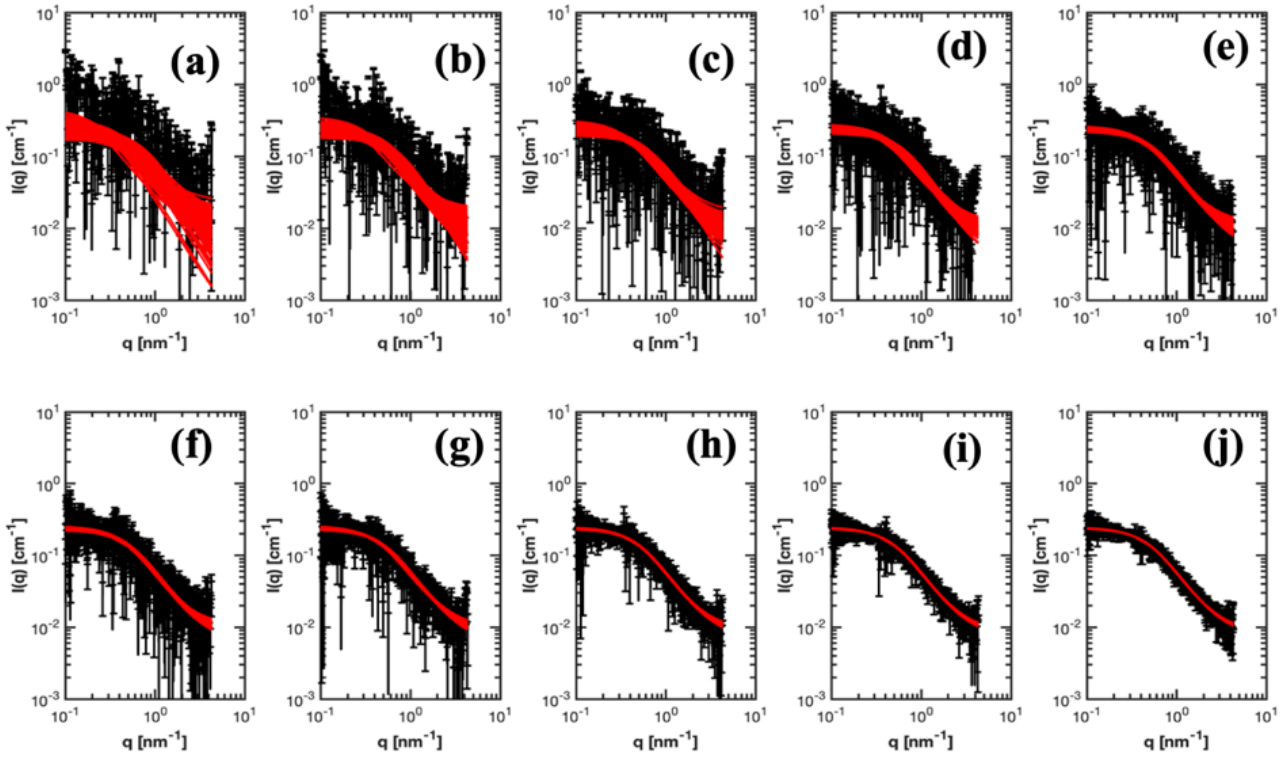


Figure A-4. (a)-(j) 16mM PEG solution SANS data from Monte Carlo Bootstrapping for fraction of counts 0.001, 0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1.

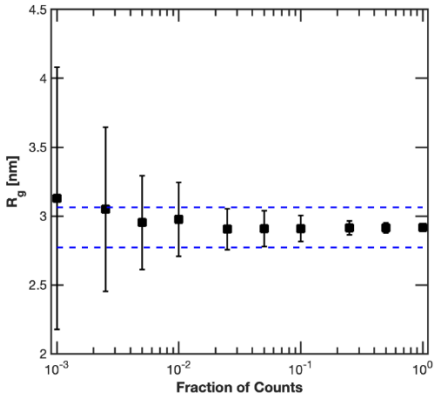


Figure A-5. The radius of gyration (R_g) from fitting the simulated MC bootstrapping data to Debye model as a function of counts. The error bars are standard deviation of the R_g from bootstrapping replicates. The dashed blue lines indicate 5% of the error from the parameter estimated from full counts.

Similar simulation was performed on the associative protein hydrogel P4 as shown in Figure A-6(a-j) with increasing number of counts. Both parameter estimation and model differentiation were performed at reduced number of counts. From Figure A-7(a), the correlation length (ξ) is within 5% of the ξ from full counts starting 0.005 fraction of the total counts. From Figure A-7(b), the best fit models were selected from the 100 MC bootstrapping replicates and plotted the frequency of the best fit for three candidate SANS models for polymer gels. The fine scale polymer gel model fits better than the broad peak model at lower counts from 0.001 to 0.1. At 0.25 fraction of counts and above, the broad peak model outperforms the rest of the models. It shows that more counts are required to perform model differentiation than parameter estimation.

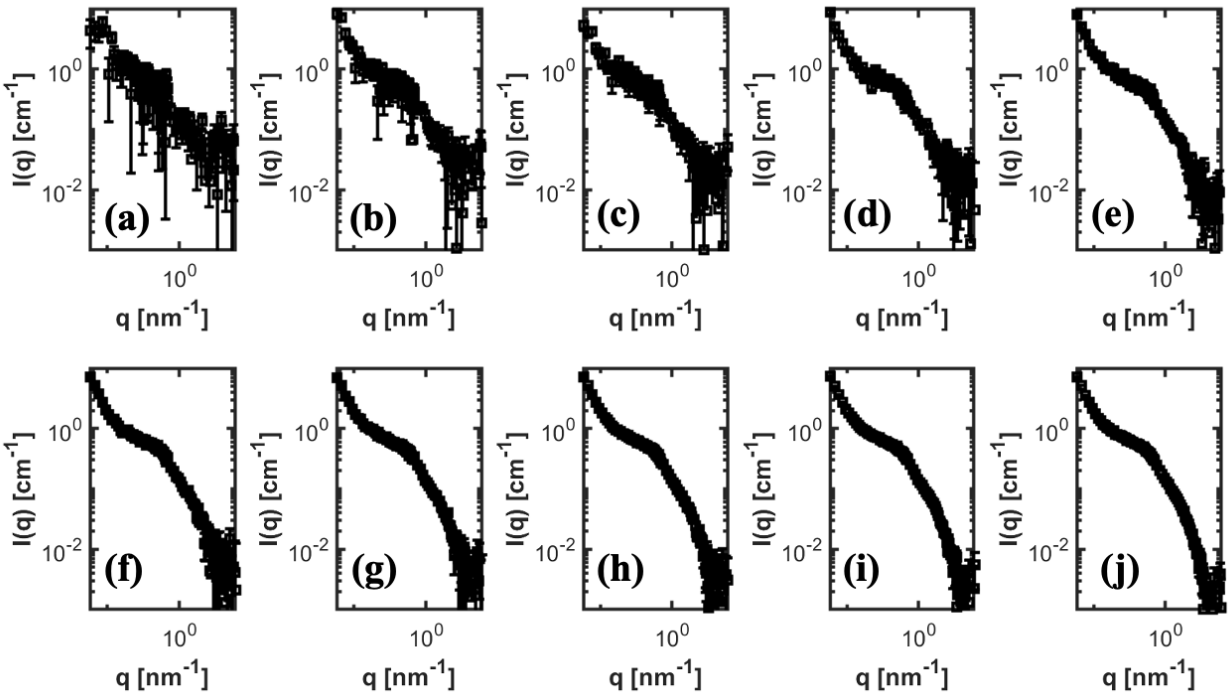


Figure A-6. (a-j) Simulated P4 SANS data from Monte Carlo Bootstrapping for fraction of counts 0.001,0.0025, 0.005, 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1. (j) represents the full total counts.

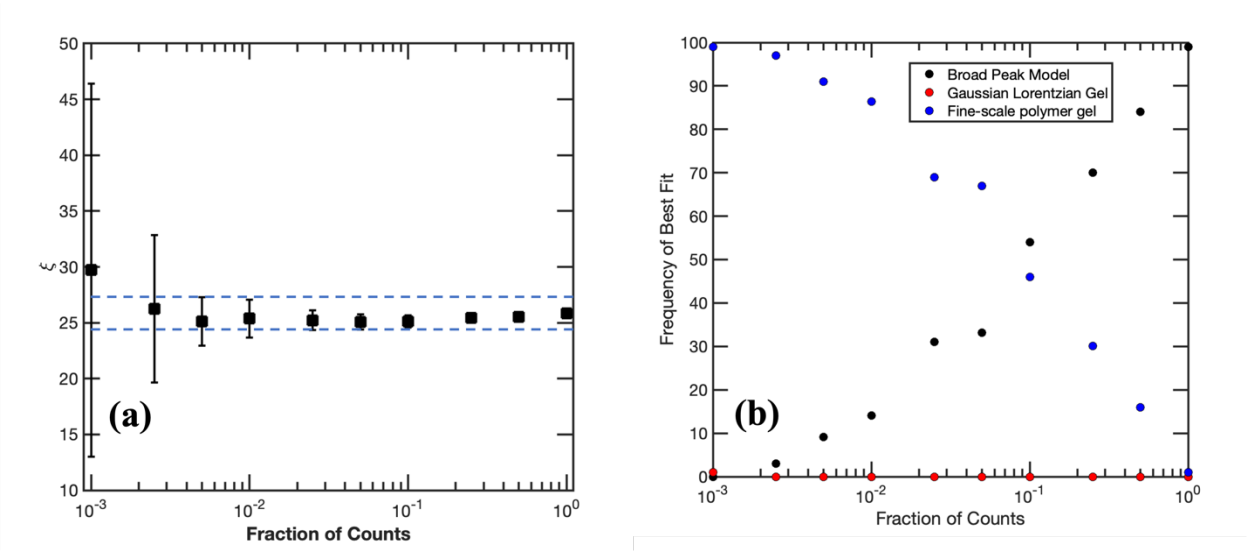


Figure A-7. (a) The correlation length (ξ) as a function of counts. The error bars are standard deviation of the R_g from bootstrapping replicates. The dashed blue lines indicate 5% of the error from the parameter estimated from full counts. (b) The number of best fit from fitting the MC bootstrapping replicates for P4 protein hydrogel for the broad peak mode, the Gauss Lorenz Gel model, the fine scale polymer gel model.

A.5 SANS Model Functions

Fine Scale Polymer Gel Model

$$I(q) = I_L(0)\frac{1}{(1+[\left(D+\frac{1}{3}\right)q^2a_1^2]^{\frac{D}{2}}} + I_G(0)\exp(-q^2a_2^2) + b \qquad (A.1)$$

Where $a_2^2 \approx \frac{Rg^2}{3}$, and R_g is the radius of gyration. $I_L(0)$ is the Lorentz term scale and $I_G(0)$ is the Guinier term scale. D is the fractal component. a_1 is the correlation length. b is the background. All of the parameters except for q and I(q) were fitting parameters.

Gauss Lorentz Gel Model

$$I(q) = I_G(0)\exp\left(-\frac{q^2\Xi^2}{2}\right) + \frac{I_L(0)}{1+q^2\xi^2} \quad (A.2)$$

Ξ represents the length scale associated with the static correlations in the gel. ξ refers to the dynamic correlation length. The factors $I_G(0)$ and $I_L(0)$ serve as scaling coefficients for these respective structural components. All of the parameters except for q and I(q) were fitting parameters.

Debye Bueche Gel Model

$$I(q) = scale * \frac{L^3}{(1+(qL)^2)^2} + b \quad (A.3)$$

where scale is the scale factor or volume fraction. L is the correlation length. b is the background. All of the parameters except for q and I(q) were fitting parameters.

Ornstein Zernike and Squared Lorentz Model

$$I(q) = \frac{I_L(0)}{1+\xi^2q^2} + \frac{I_{SL}(0)}{(1+\xi^2q^2)^2} + background \quad (A.4)$$

where ξ is the correlation length. $I_L(0)$ is the intensity at zero scattering vector q=0. It represents the strength of the scattering signal and is related to the contrast and volume fraction of the scatterers. Background represents the baseline scattering, accounting for any constant background noise or signal that is not related to the scattering from the material of interest. All the parameters except for q and I(q) were fitting parameters.

Sphere Model

$$I(q) = \frac{scale}{V}\left[3V(\Delta\rho)\frac{\sin(qR) - qrcos(qR)}{(qr)^3}\right]^2 + b \qquad (A.5)$$

Where the $scale$ parameter is a multiplicative factor that normalizes the scattering intensity. The volume V of the sphere is calculated using the formula $V = \frac{4}{3}\pi R^3$, where R is the radius of the spherical particles. The scattering length density contrast Δρ represents the difference in neutron scattering length density between the spherical particles and the surrounding solvent. The R of the sphere is a key parameter that indicates the size of the spherical particles. The size of the particles influences the scattering pattern, particularly the position and shape of features such as peaks and troughs in the scattering curve. The b background parameter accounts for any constant scattering that is not directly related to the spherical particles. All the parameters except for q and I(q) were fitting parameters.

Fuzzy Sphere Model

$$I(q) = \frac{scale}{V}(\Delta\rho)^2 A(q)^2 S(q) + b \qquad (A.6)$$

$$A(q) = \frac{3[\sin(qR) - qrcos(qR)]}{(qR)^3}\exp\left(-\frac{\left(\sigma_{fuzzy}q\right)^2}{2}\right) \qquad (A.7)$$

Here, scale is the is a multiplicative factor that normalizes the scattering intensity. The volume V of the sphere is calculated using the formula $V = \frac{4}{3}\pi R^3$, where R is the radius of the spherical particles. The scattering length density contrast Δρ represents the difference in neutron scattering length density between the spherical particles and the surrounding solvent. The R of the sphere is

a key parameter that indicates the size of the spherical particles. σ_{fuzzy} is a factor that accounts for the fuzziness of the particles. The *b* background parameter accounts for any constant scattering that is not directly related to the fuzzy spherical particles. Since the polymer solution is relatively dilute, the structure factor *S(q)* is 1. All the parameters except for q and I(q) were fitting parameters.

A.6 Model Fits to SANS Data

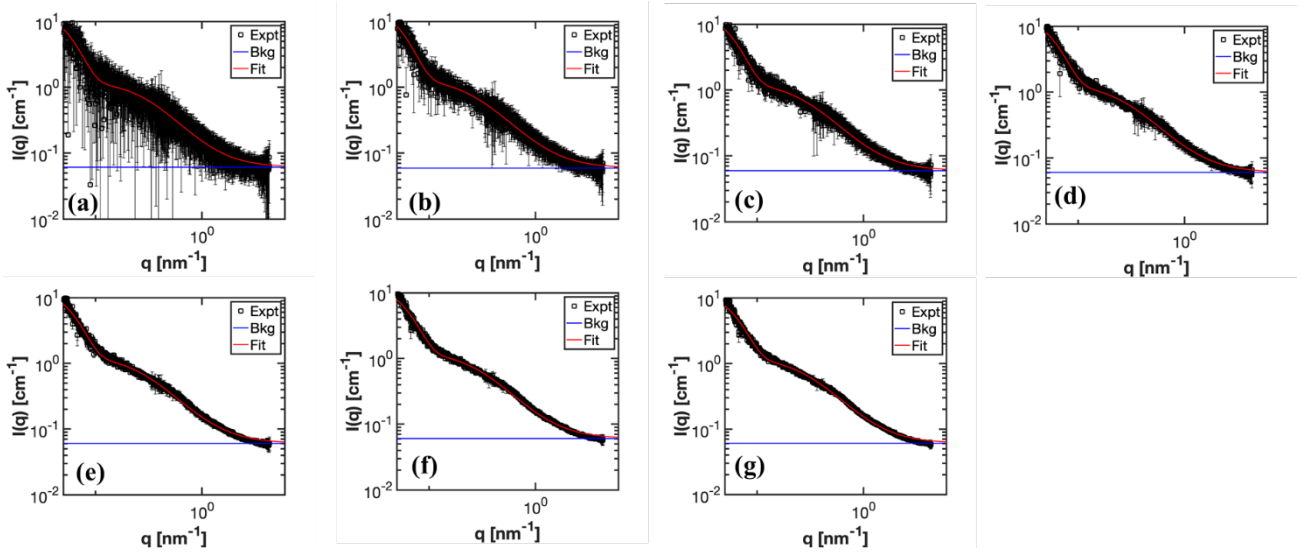


Figure A-8. SANS intensity curves for P4 hydrogel in deuterated 100mM phosphate buffer, pD = 7.4. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the fine scale polymer model.

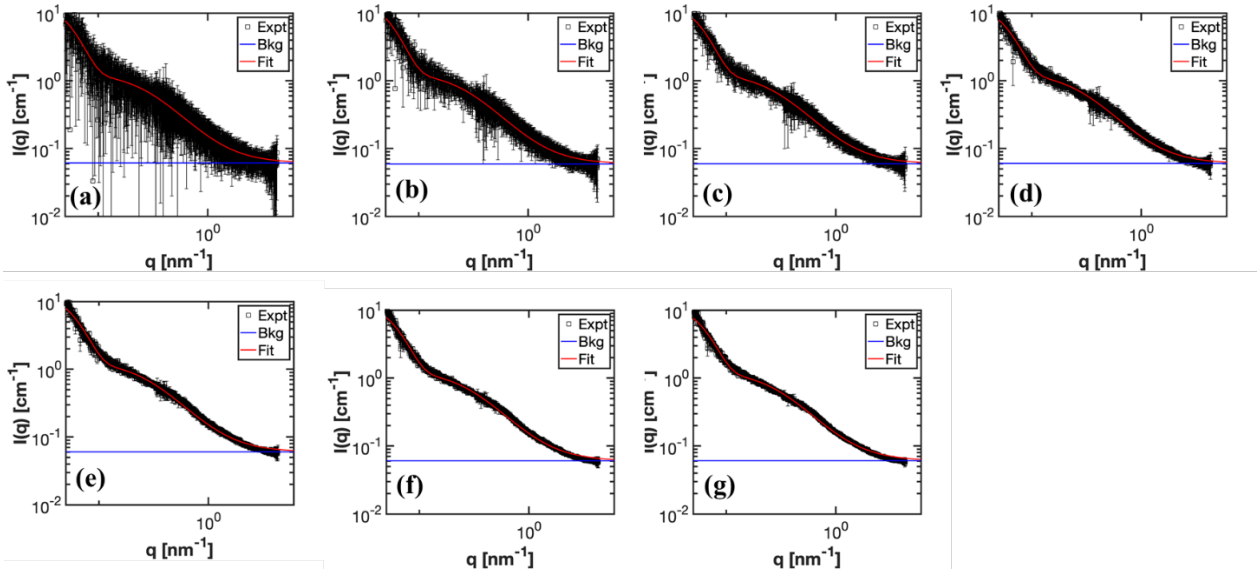


Figure A-9. SANS intensity curves for P4 hydrogel in deuterated 100mM phosphate buffer, pD = 7.4. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the Gauss Lorentz model.

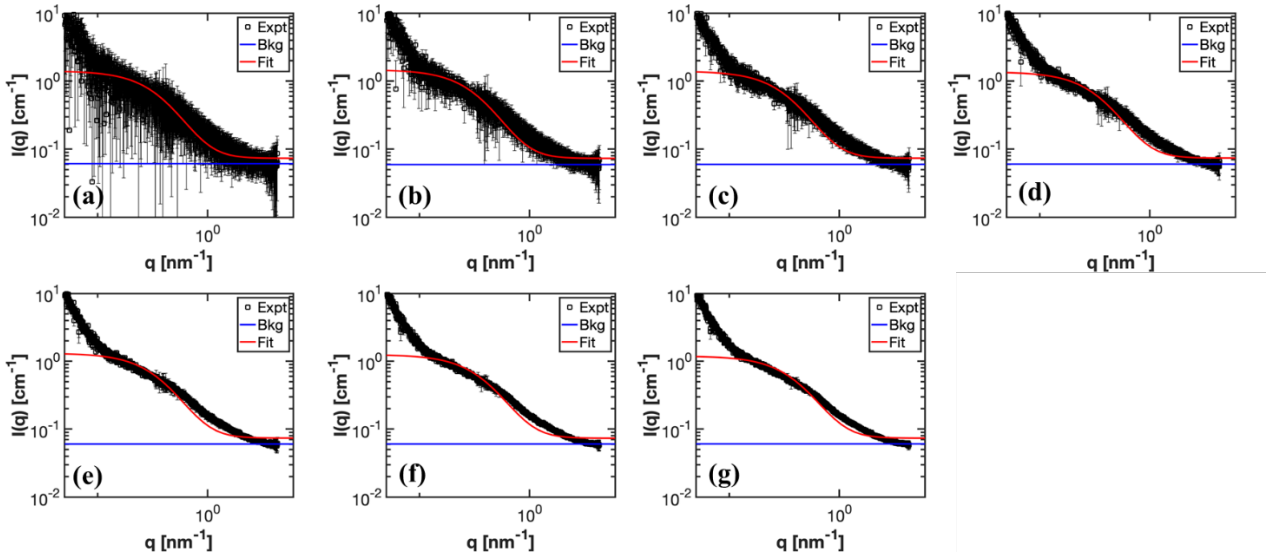


Figure A-10. SANS intensity curves for P4 hydrogel in deuterated 100mM phosphate buffer, pD = 7.4. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the Debye Bueche model.

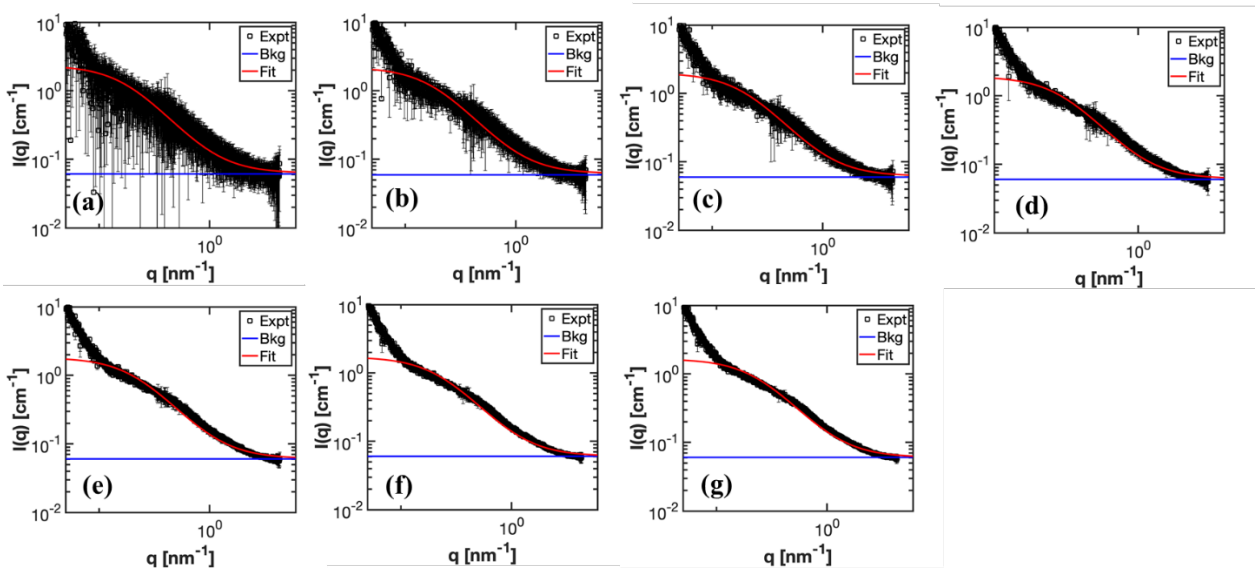


Figure A-11. SANS intensity curves for P4 hydrogel in deuterated 100mM phosphate buffer, pD = 7.4. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the Ornstein Zernike and squared Lorentz model.

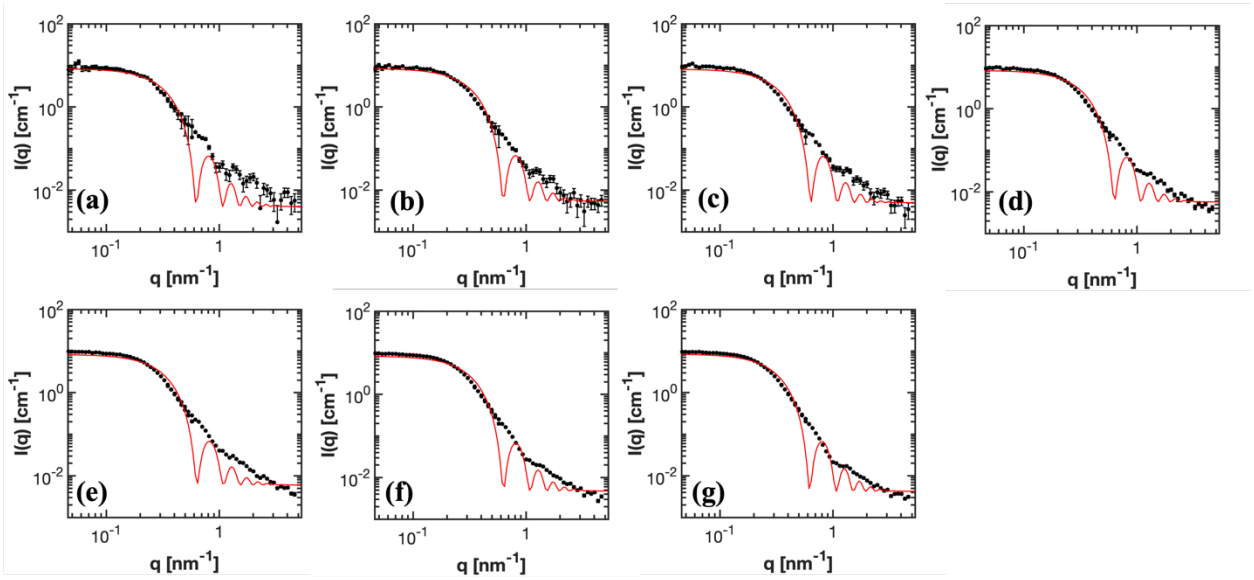


Figure A-12. SANS intensity curves for Pluronic F-127 in deuterated water. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the sphere model.

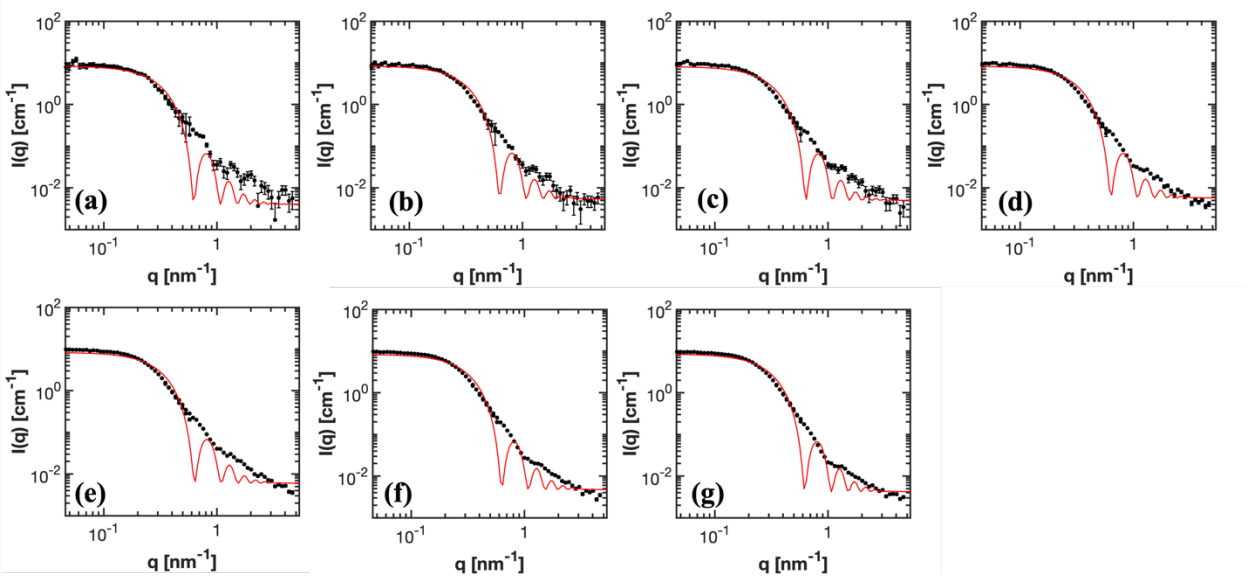


Figure A-13. SANS intensity curves for Pluronic F-127 in deuterated water. (a-g) 0.01, 0.025, 0.05, 0.1, 0.25, 0.5, 1 fraction of the total counts. The red line illustrates the fit of each dataset to the fuzzy sphere model.

A.7 Covariance Analysis of Broad Peak Model

Similar to the simulated SANS data used in bias testing, 100 simulated P4 SANS replicates with added Gaussian noise with a mean of 0 and standard deviation of dI were used for the covariance analysis. The replicates were fitted to the broad peak model and the covariance values of each parameter fits were computed in MATLAB. Figure A-14 visualizes the relationship of each parameter in the broad peak model. Table A-1 shows the covariance matrix.

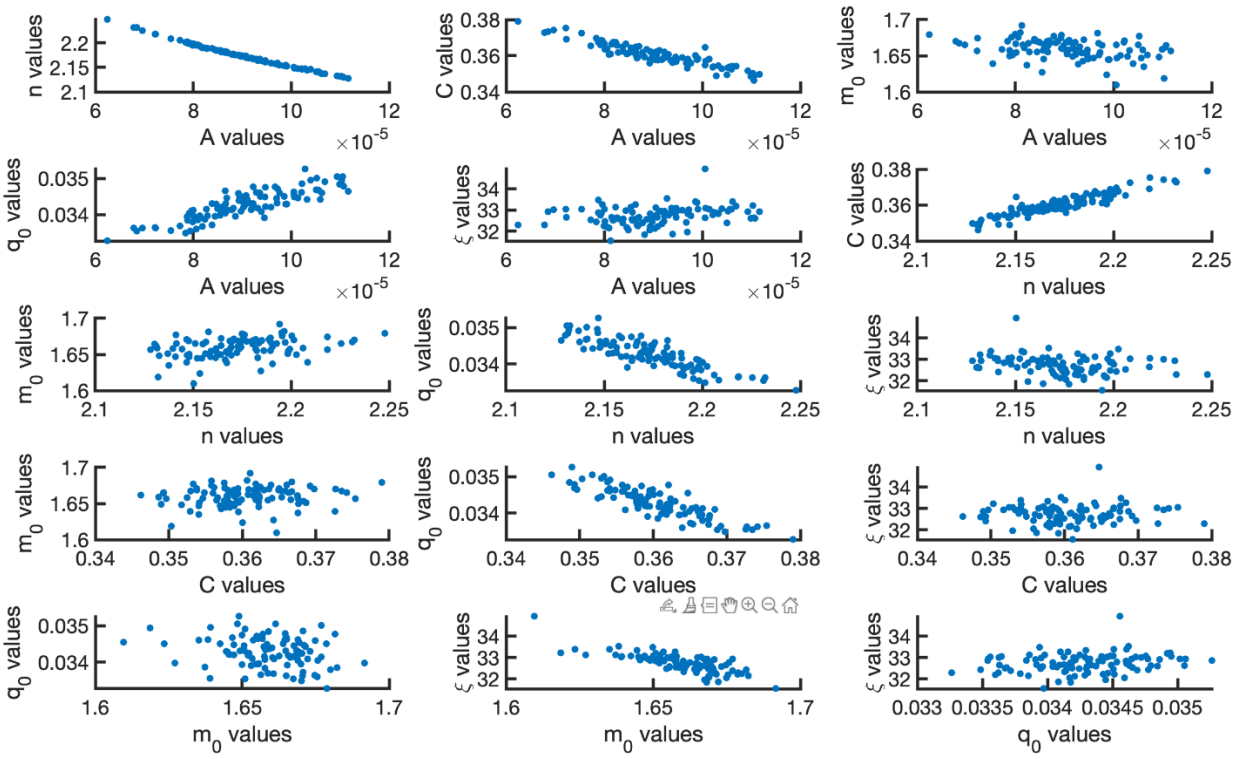


Figure A-14. Covariance Analysis on the parameters in the broad peak model.

Table A-1. Covariance values of all fitting parameters in the broad peak model.

Parameter	A	n	C	m0	q0	ξ
A	1.11×10^{-10}	-2.57×10^{-7}	-6.09×10^{-8}	-5.08×10^{-8}	3.71×10^{-9}	1.25×10^{-6}
n	-2.57×10^{-7}	6.01×10^{-4}	1.41×10^{-4}	1.16×10^{-4}	-8.55×10^{-6}	-2.76×10^{-3}
C	-6.09×10^{-8}	1.4137×10^{-4}	3.93×10^{-5}	1.17×10^{-5}	-2.21×10^{-6}	7.83×10^{-5}
m0	-5.08×10^{-8}	1.16×10^{-4}	1.17×10^{-5}	1.93×10^{-4}	-1.36×10^{-6}	-4.84×10^{-3}
q0	3.71×10^{-9}	-8.55×10^{-6}	-2.21×10^{-6}	-1.36×10^{-6}	1.67×10^{-7}	3.96×10^{-5}
ξ	1.25×10^{-6}	-2.76×10^{-3}	7.83×10^{-5}	-4.84×10^{-3}	3.96×10^{-5}	1.97×10^{-1}

A.8 Estimator Bias Testing

After performing parameter estimation with weighted nonlinear least square fitting using full count experimental SANS data, the best fit parameters were used to generate an analytical fitted SANS intensity curve. To simulate measurement noise, Gaussian noise with a mean of 0 and

standard deviation of *dI* was added to the fitted intensity curve. Following this workflow, 100 replicates with Gaussian noise were generated to simulate SANS data and fitted to the same model. Figure A-15 shows the distribution of parameter values from fitting the 100 simulated P4 SANS replicates to the broad peak model.

The bias testing of the estimator is also performed on the Debye model and the spherical micelle model. For the Debye model, the bias for R_g is less than 0.01% and less than 0.1% for A. For the spherical micelle model with polydispersity, the bias for R is -0.02% and the bias for R_g is 0.19%. The small biases for those two systems show that the weighted nonlinear least square estimator can work well for the Debye model and the spherical micelle model. The bias testing result for those two systems are consistent with the convergence behavior of parameters from fitting MC bootstrapping replicates and experimental data as the number of counts increases.

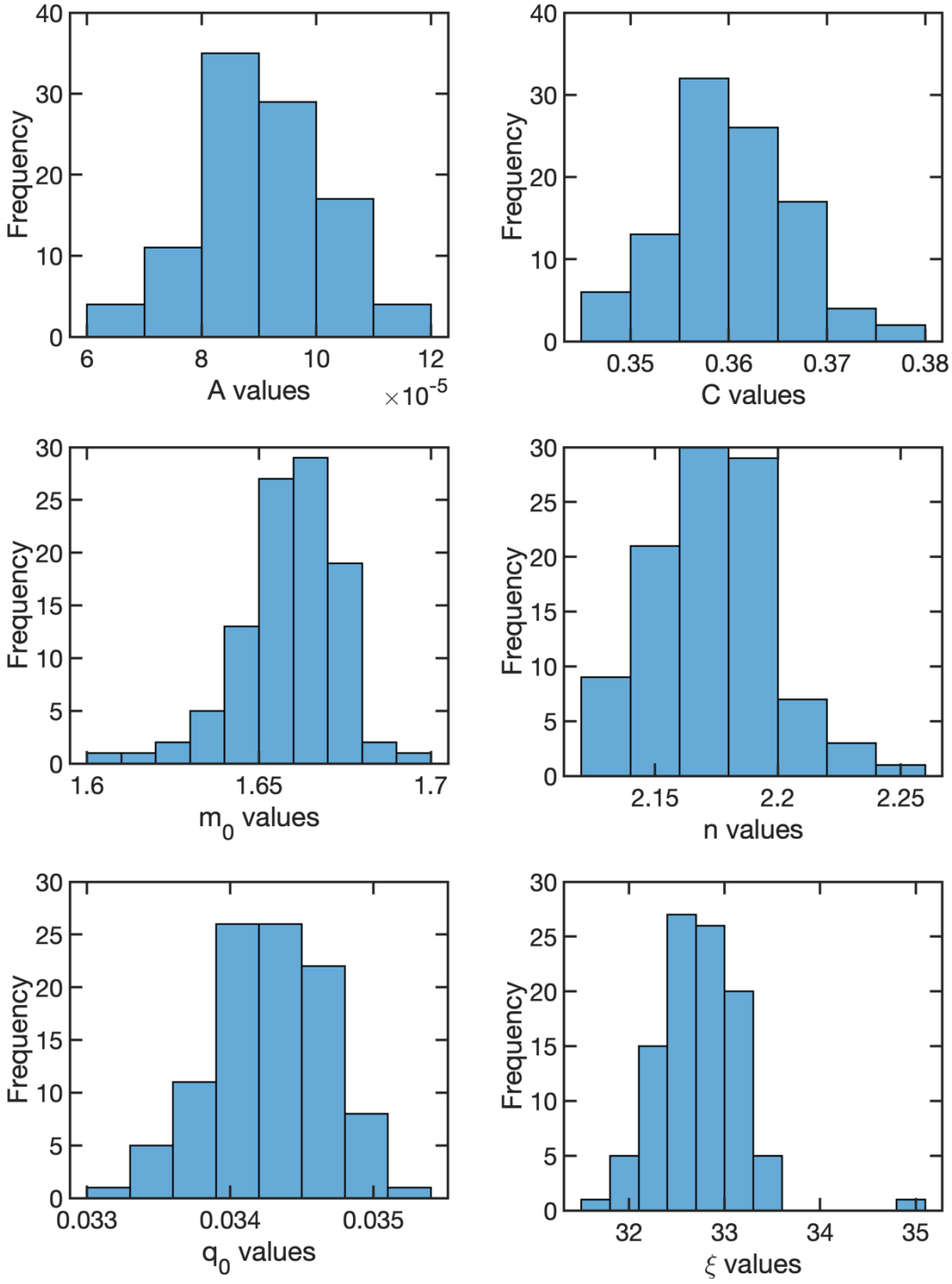


Figure A-15. Histograms for fit parameters from fitting 100 simulated P4 SANS data using analytical expression with added Gaussian noise (mean of 0 and standard deviation of dI) to the broad peak model.

The largest bias is in parameter *A* in the broad peak model. The *E[A]* is 350% higher than A_{true}. With reduction of bias in A, the bias in other parameters can potentially be reduced to different extent. In order to test this, A is set to $2.00 * 10^{-5}$ and the rest of the parameters are fitting parameters in the broad peak model. Bias reduction is the difference between the bias from fitting all parameters (Table 3-1) and bias from setting A to fixed value (Table A-2).

The largest bias occurs in parameter *A* of the broad peak model, with *E[A]* being 350% higher than A_{true}. Reducing the bias in *A* could also decrease the bias in other parameters to varying degrees. To test this, *A* was fixed at $2.00 * 10^{-5}$, while the remaining parameters were fitted. The results were summarized in Table A-2.

Table A-2. Comparison between true parameter X_{true} from fitting the P4 experimental data and the expectation value from fitting simulated data *E[X]* to the broad peak model. The fitting is done by setting parameter A to A_{true}. The bias is the percent difference between X_{true} and *E[X]*.

Parameter X	X_{true}	$E[X]$	**Bias from Fixing *A* (%)**	**Bias from Table 3-1 (%)**
A	$2.00 * 10^{-5}$	$2.00 * 10^{-5}$	0	350
n	2.46	2.481	1.04	-11.5
C	0.45	0.4283	-4.66	-19.62
m_0	1.87	1.7412	-6.74	-11.17
q_0	0.025	0.0296	17.71	36.42
ξ	28.04	30.98	10.50	16.94

A.9 References

1. Heller, W. T.; Hetrick, J.; Bilheux, J.; Calvo, J. M. B.; Chen, W.-R.; DeBeer-Schmitt, L.; Do, C.; Doucet, M.; Fitzsimmons, M. R.; Godoy, W. F.; Granroth, G. E.; Hahn, S.; He, L.; Islam, F.; Lin, J.; Littrell, K. C.; McDonnell, M.; McGaha, J.; Peterson, P. F.; Pingali, S. V.; Qian, S.; Savici, A. T.; Shang, Y.; Stanley, C. B.; Urban, V. S.; Whitfield, R. E.; Zhang, C.; Zhou, W.; Billings, J. J.; Cuneo, M. J.; Leal, R. M. F.; Wang, T.; Wu, B., drtsans: The data reduction toolkit for small-angle neutron scattering at Oak Ridge National Laboratory. *SoftwareX* **2022,** *19*, 101101.

Appendix B Supporting Information for Chapter 4

B.1 Interaction Parameter χ Calculation

Using Equation 24-26 and Table 3 in the referenced paper[1], we calculated the dispersive, polar, and hydrogen bonding solubility parameters for each amino acid to translate them to χ values as shown in Table B-1.

Table B-1. Hansen solubility parameters of amino acids from group contribution theory.

Amino acid	δ_d	δ_p	δ_{hb}
V	16.282	16.1344	19.6951
G	16.8619	17.8194	20.9345
P	19.2187	18.2254	18.774
R	17.1925	20.6019	25.6781
H	18.0501	18.7419	23.0489
K	16.2253	18.6188	22.7027
D	17.0927	21.5876	22.4853
E	17.0658	21.2831	22.0734
S	16.5157	18.9598	28.1253
T	16.2162	18.2686	27.5541
N	16.793	23.8888	26.1625
Q	16.7661	23.5843	25.7506
C	18.1685	17.3016	25.811
A	15.9174	16.4791	20.5651
I	15.5641	15.1789	19.1701
L	15.5641	15.1789	19.1701
M	16.95	16.9276	19.9245
F	18.1346	16.1241	19.0564
Y	18.5529	17.7554	26.4449
W	20.0456	15.17	21.2366

The overall solubility parameter δ is calculated as[2]

$$\delta = \sqrt{\delta_d^2 + \delta_p^2 + \delta_{hb}^2} \quad (S1)$$

The interaction parameter χ is calculated as the following.

$$\chi = \frac{V_S}{RT}(\delta - \delta_{H2O}) \quad (S2)$$

Where V_s is the molar volume of the solvent, R is the ideal gas constant, T is the absolute temperature, δ_{H2O} is the Hansen solubility parameter of water.

B.2 SDS-PAGE

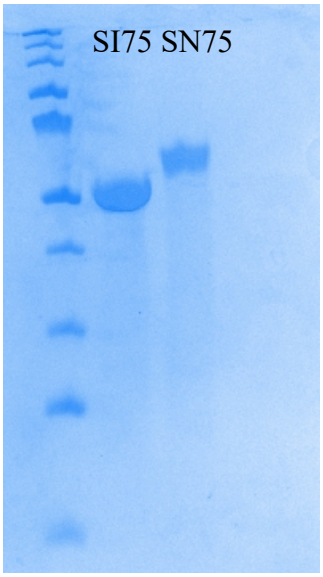


Figure B-1. SDS-PAGE gel confirming the purity of protein.

B.3 Birefringence Raw Data

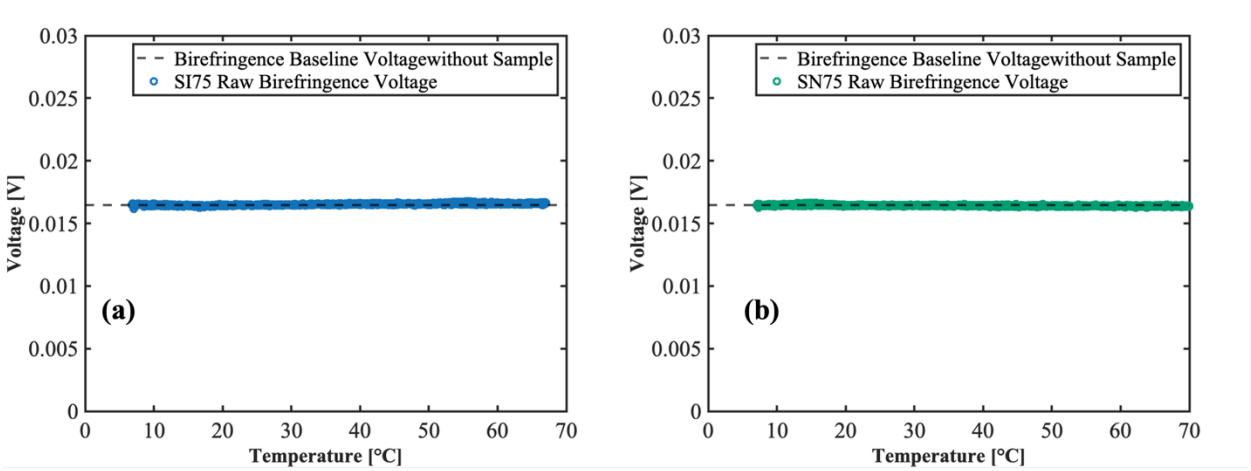


Figure B-2. Raw birefringence data as a function of temperature for (a) 40w/v% SI75 and (b) 40w/v% SN75.

B.4 Rheology

Strain Sweep

Strain sweep was performed for both SN75 and SI75 at various temperatures to determine the strain for linear viscoelastic regime. The optimal strain for subsequent temperature sweep and frequency sweep is chosen as 1% for SI75 and 0.1% for SN75.

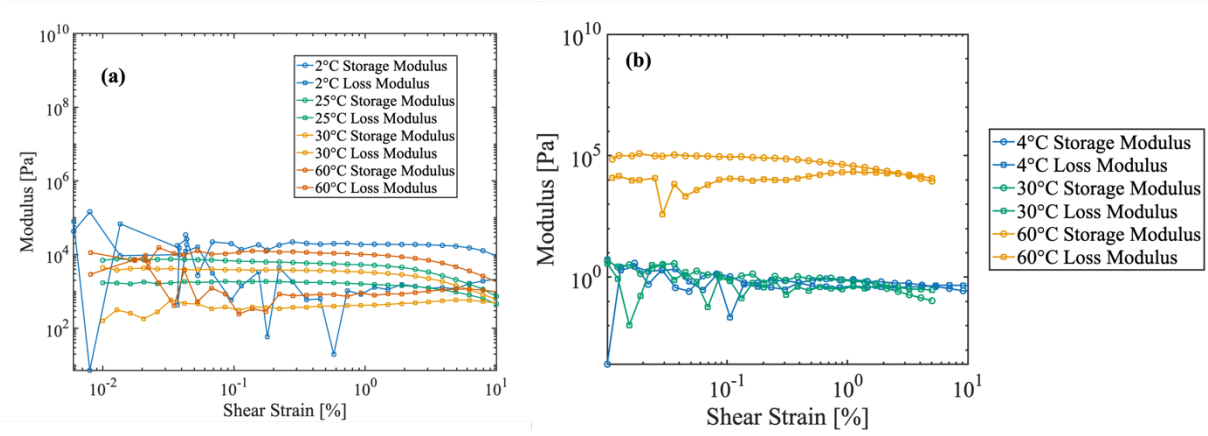


Figure B-3. Strain Sweep at various temperature for (a) SI75 and (b) SN75.

Frequency Sweep

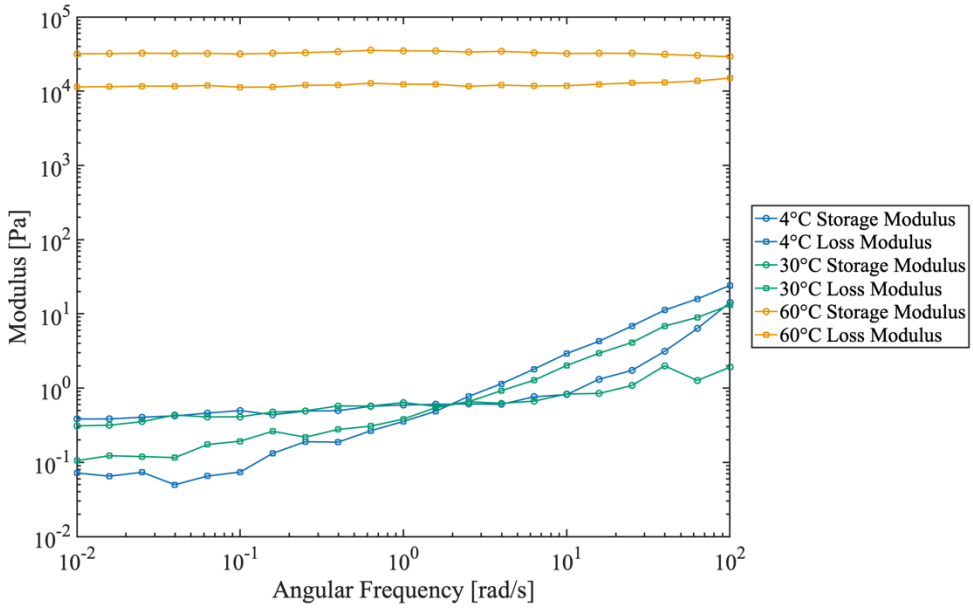


Figure B-4. Frequency Sweep at various temperatures for SN75.

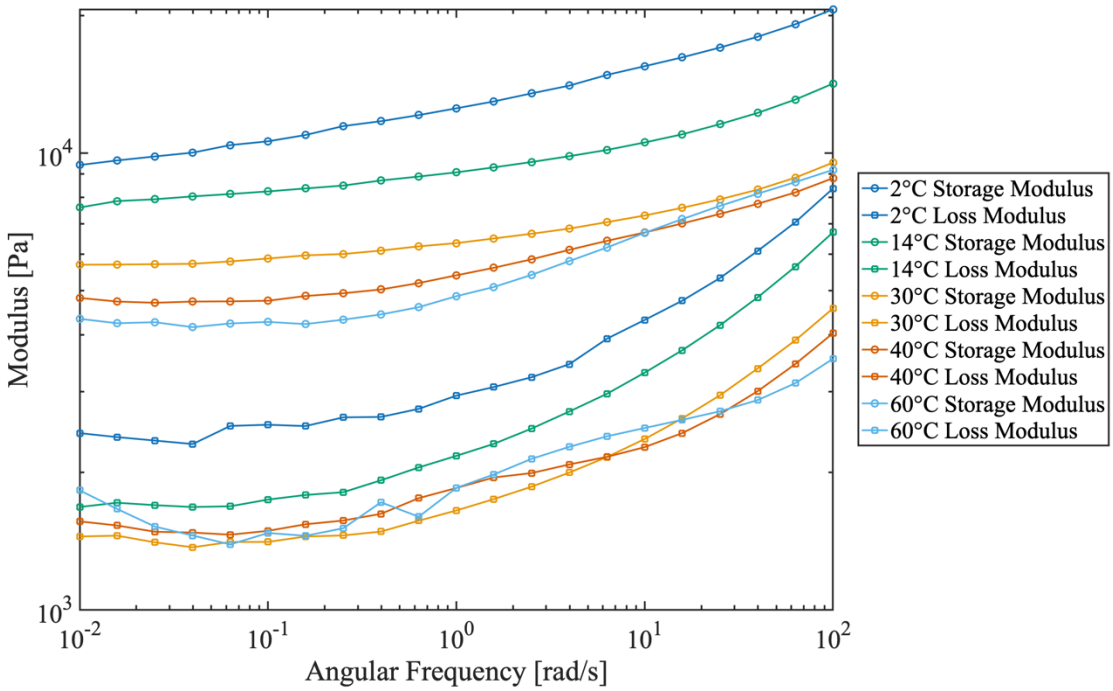


Figure B-5. Frequency Sweep at various temperatures for SI75.

B.5 Water Partitioning SANS Data and Analysis

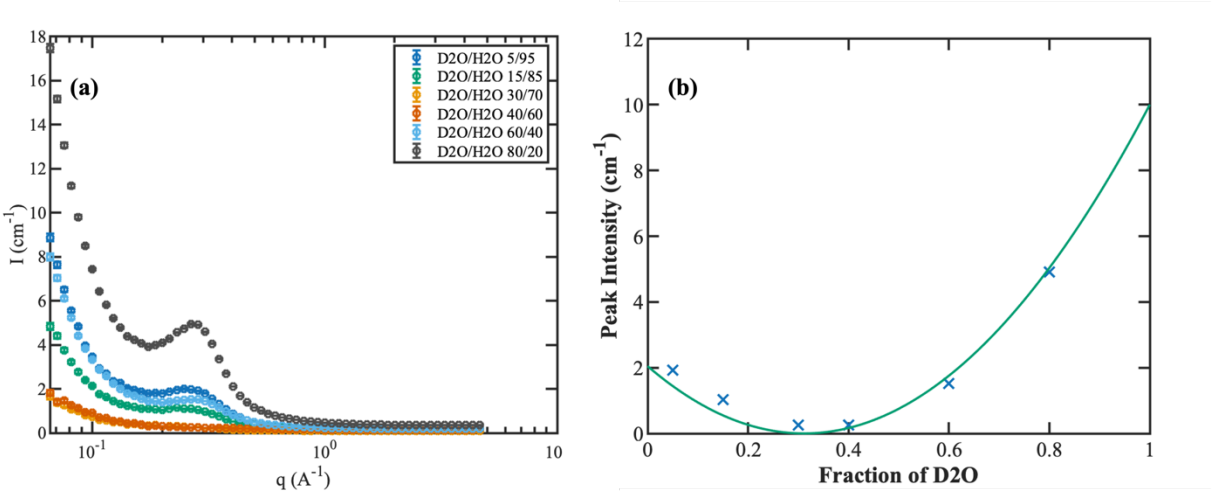


Figure B-6. SANS water partition analysis for SI75 at 14°C. (a) Contrast Variation results. (b) Water partitioning fitting results.

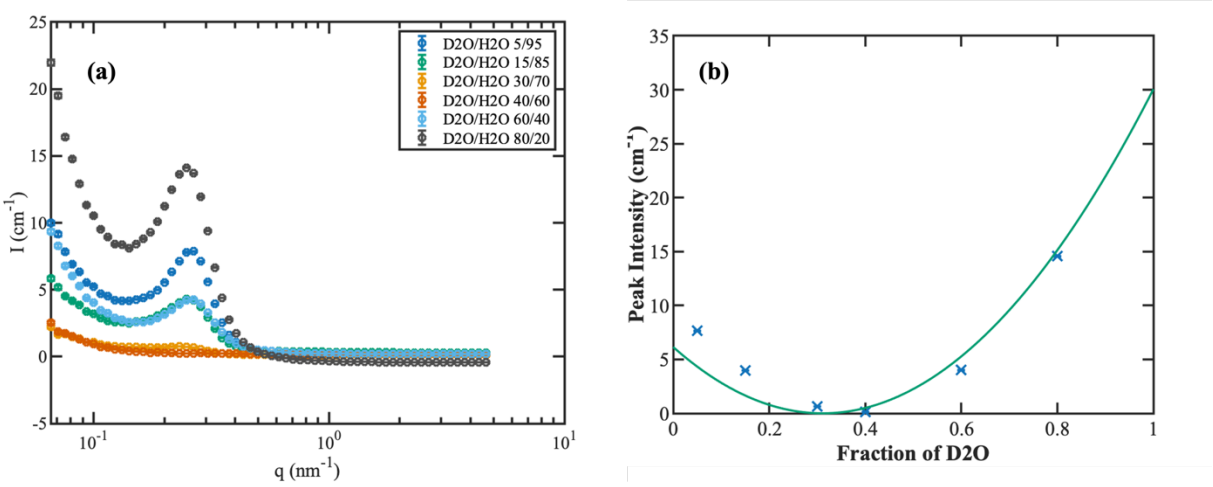


Figure B-7. SANS water partition analysis for SI75 at 25°C. (a) Contrast Variation results. (b) Water partitioning fitting results.

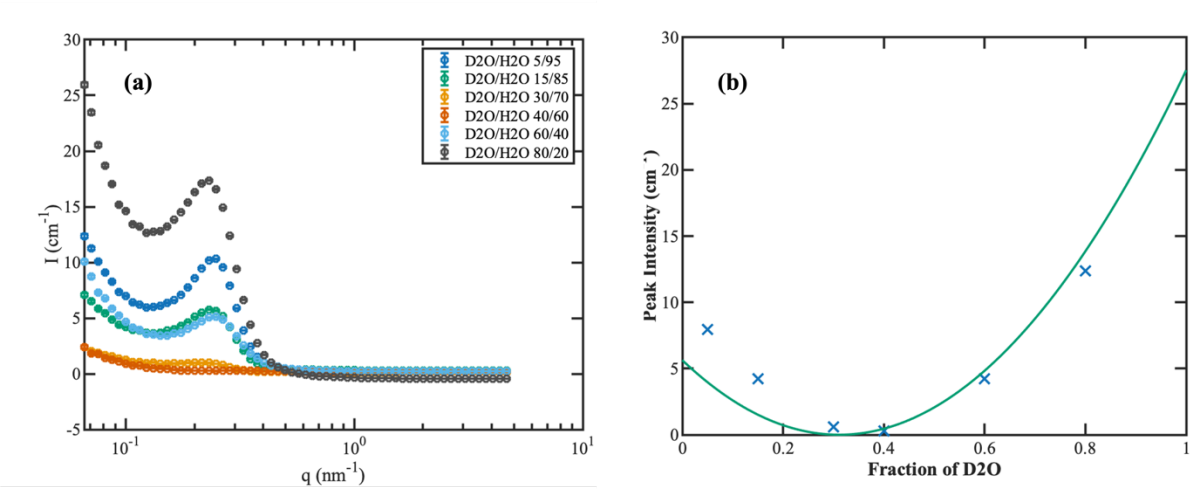


Figure B-8. SANS water partition analysis for SI75 at 30°C. (a) Contrast Variation results. (b) Water partitioning fitting results.

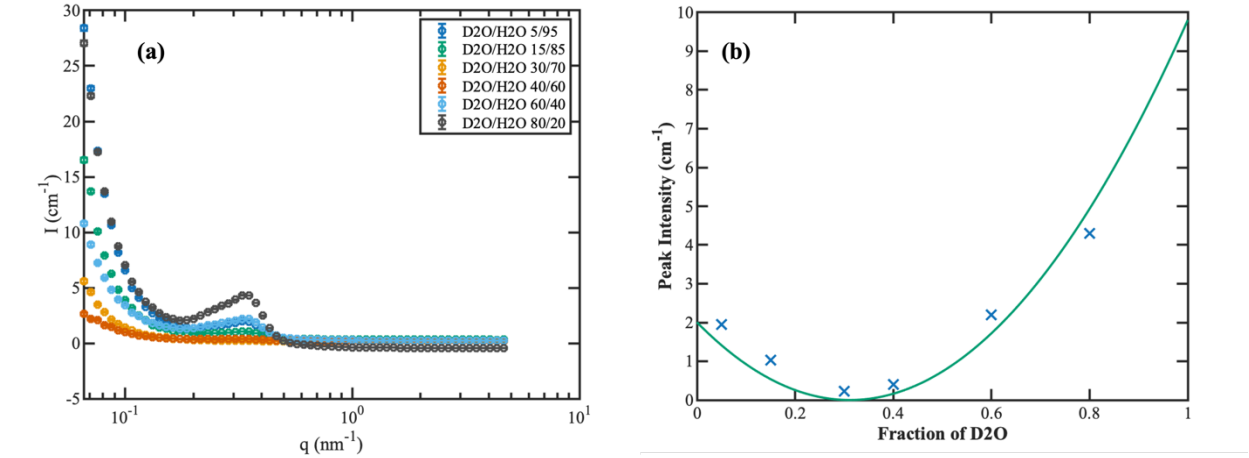


Figure B-9. SANS water partition analysis for SI75 at 60°C. (a) Contrast Variation results. (b) Water partitioning fitting results.

B.6 References

1. Stefanis, E.; Panayiotou, C., Prediction of Hansen solubility parameters with a new group-contribution method. *International Journal of Thermophysics* **2008,** *29*, 568-585.
2. Hansen, C. M., *Hansen solubility parameters: a user's handbook*. CRC press: 2007.

Appendix C Supporting Information for Chapter 5

C.1 SDS-PAGE

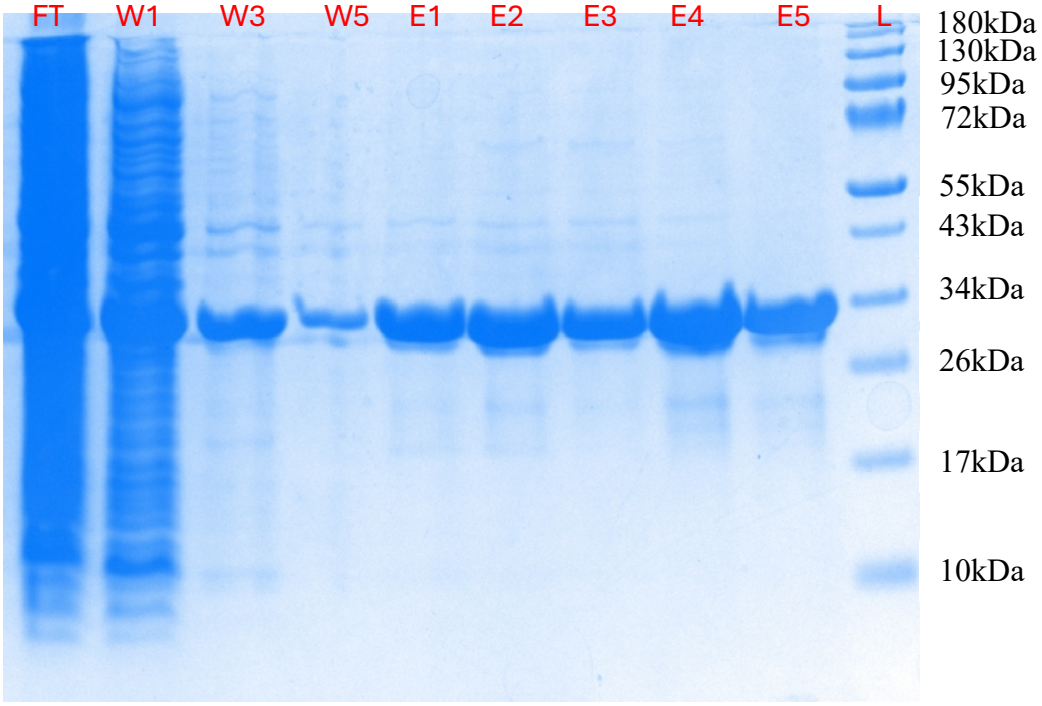


Figure C-1. Ni-NTA purification of ELP-I. The target protein molecular weight is 27.2kDa.

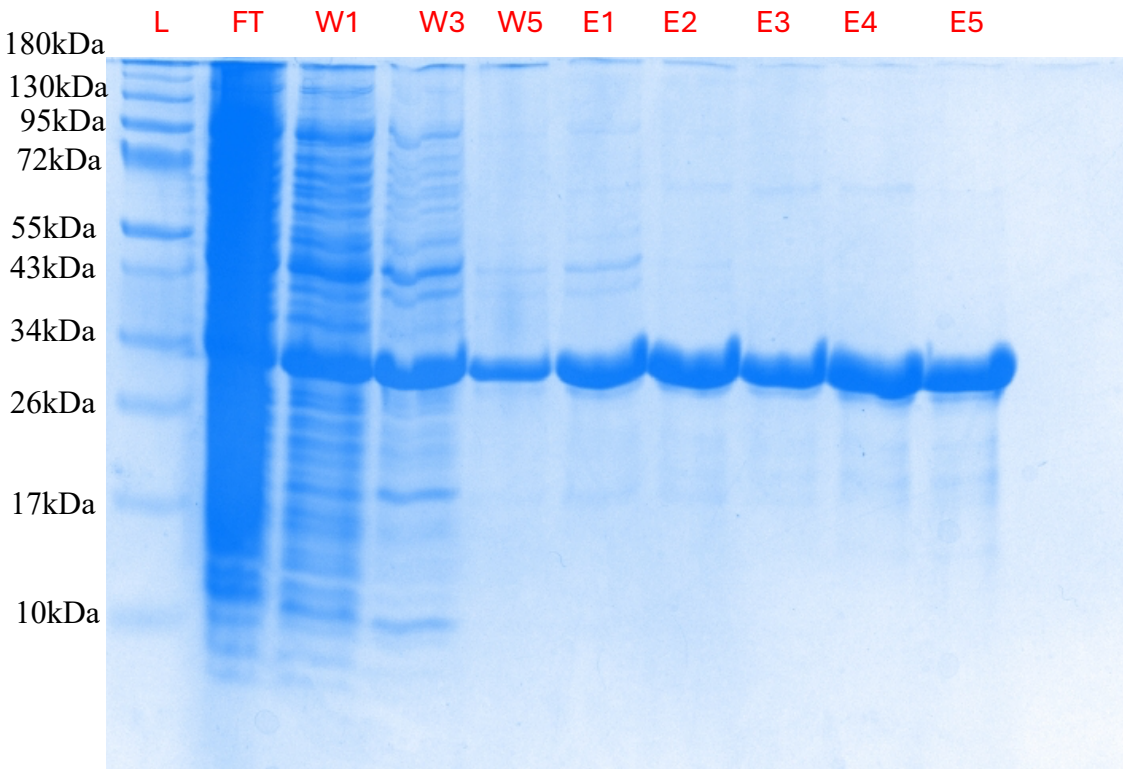


Figure C-2. Ni-NTA purification of ELP-V. The target protein molecular weight is 27.2kDa.

ELP-N

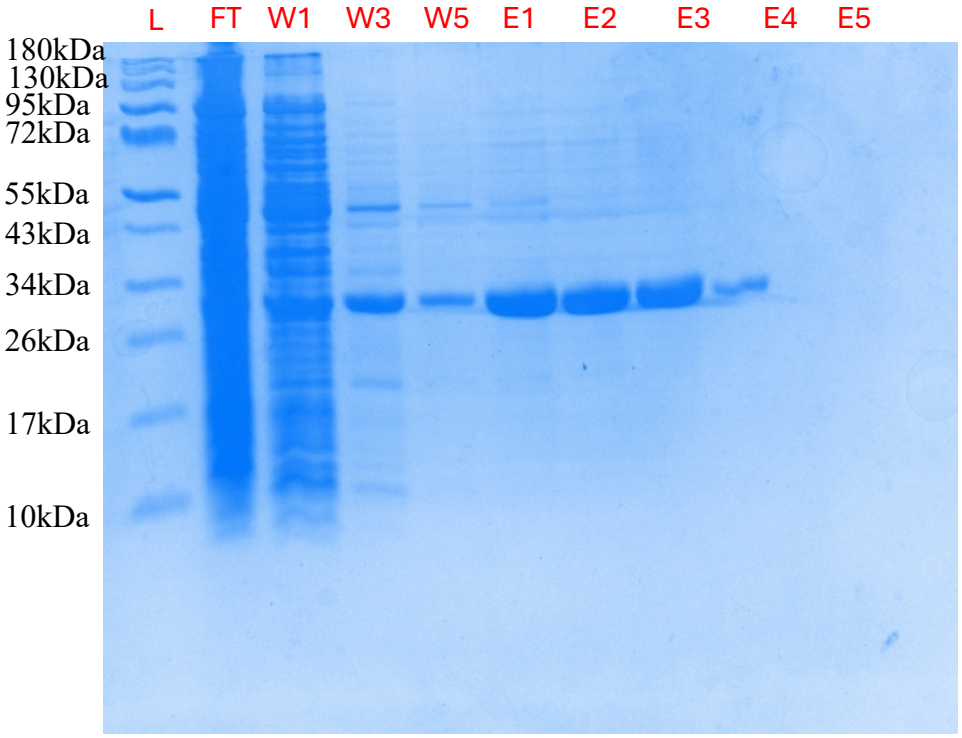


Figure C-3. Ni-NTA purification of ELP-N. The target protein molecular weight is 25.7kDa.

ELP-S

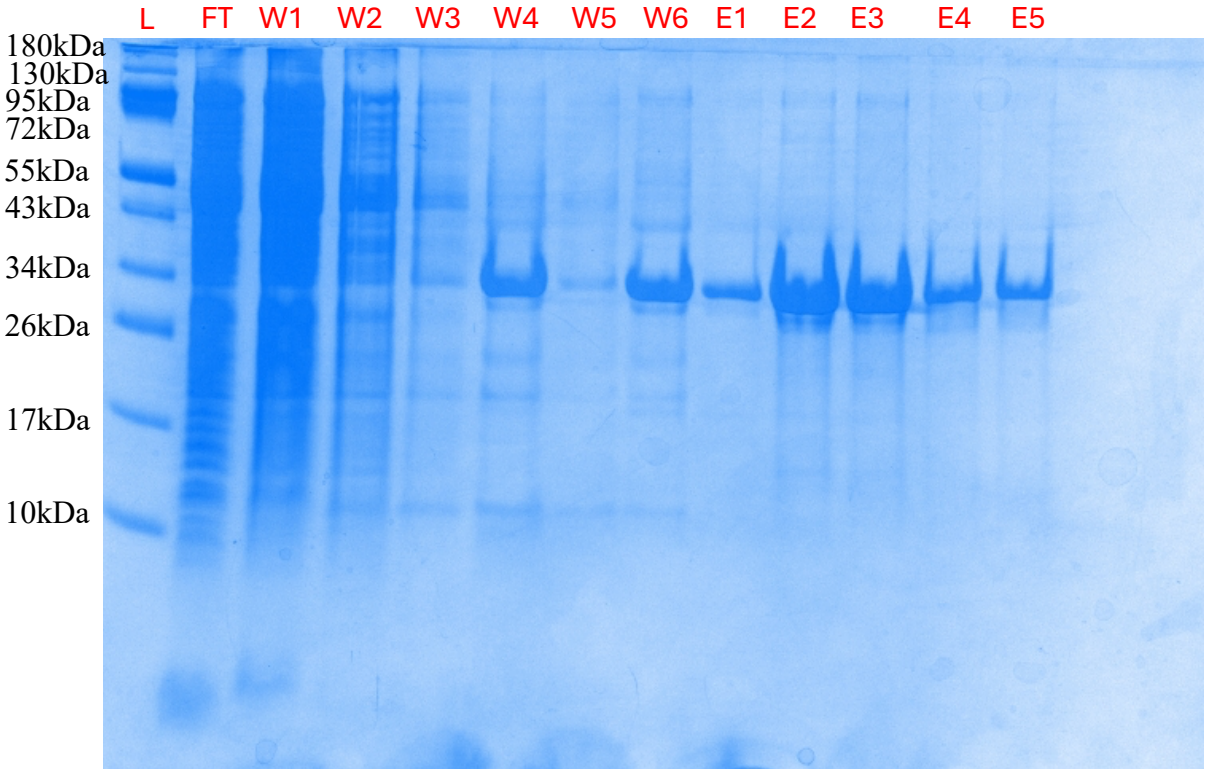


Figure C-4. Ni-NTA purification of ELP-S. The target protein molecular weight is 27.2kDa.

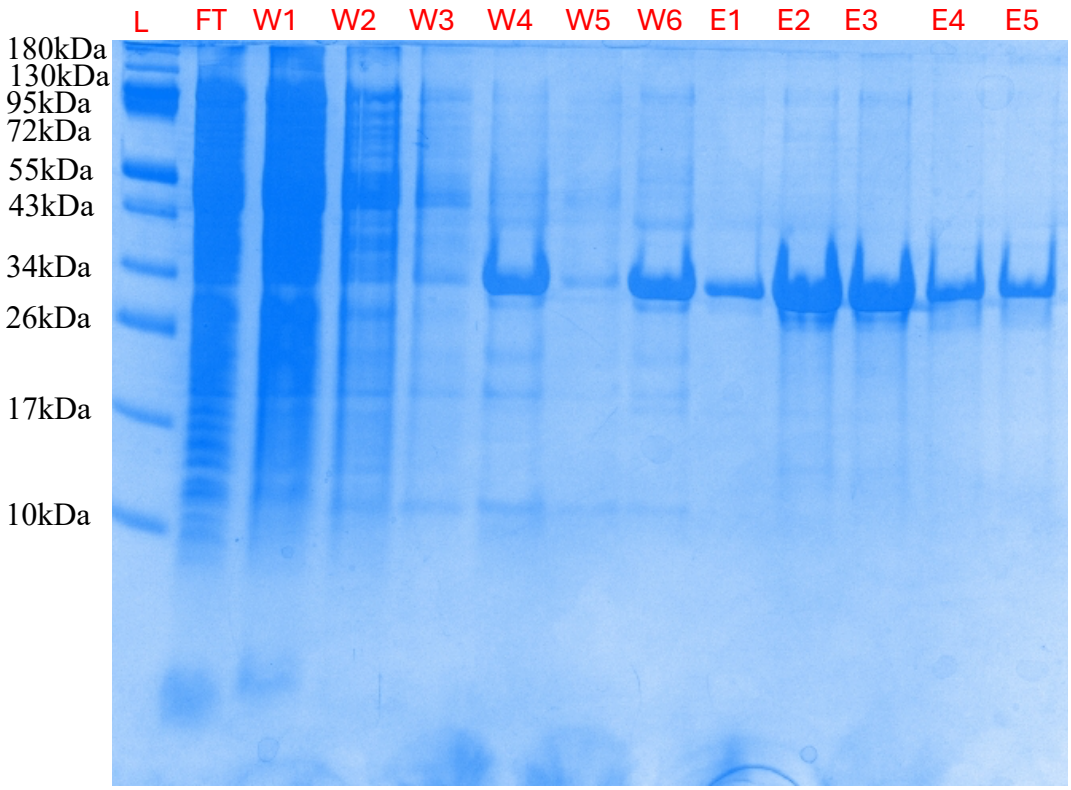


Figure C-5. Ni-NTA purification of ELP-G. The target protein molecular weight is 27.2kDa.

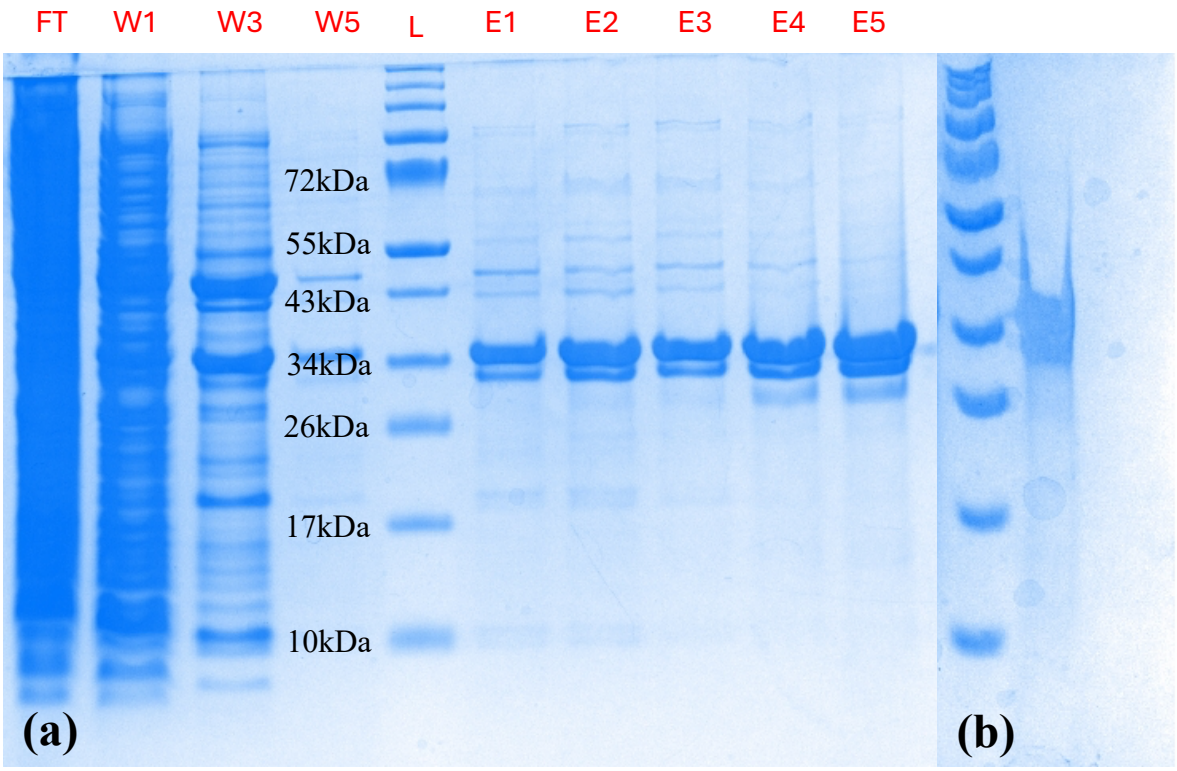


Figure C-6. (a) Ni-NTA purification of ELP-K. The target protein molecular weight is 27.2kDa. (b) Purified ELP-K after ITC. The molecular weight on SDS-PAGE is higher than target due to the lysine rich protein having less effective binding to SDS and showing slower migration.

ELP-D

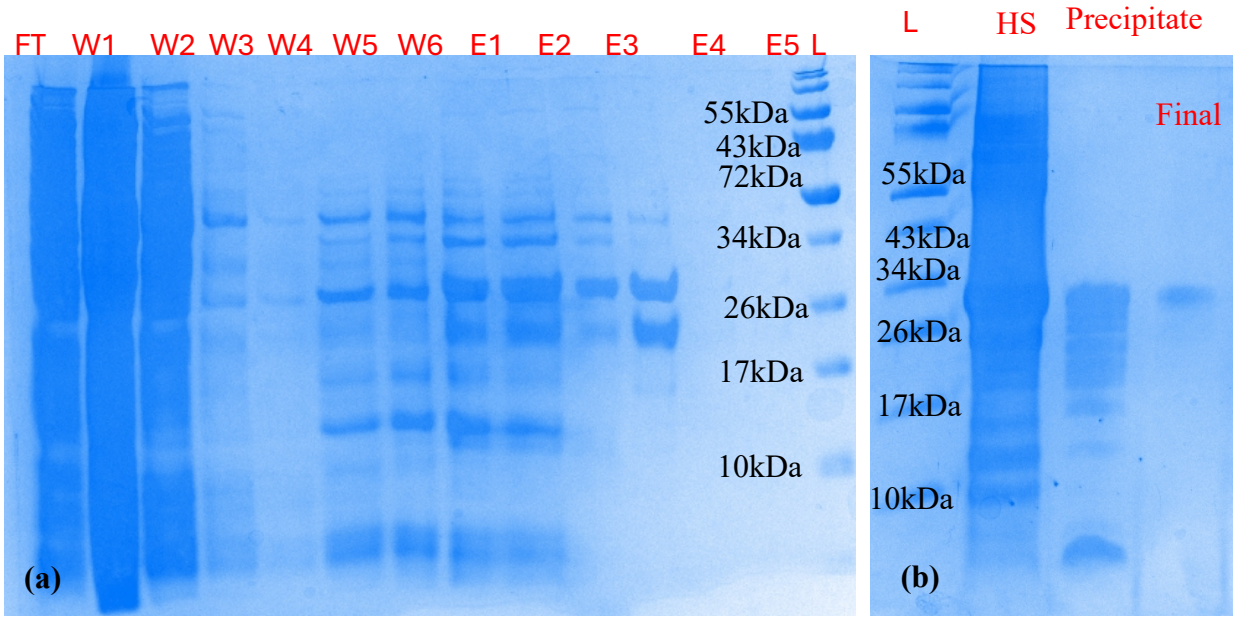


Figure C-7. (a) Ni-NTA purification of ELP-D. The target protein molecular weight is 27.2kDa. (b) ITC purification of ELP-D. The target protein molecular weight is 27.2kDa. HS stands for the pellet from ITC hot spin. Precipitate stands for the precipitate from cold spin. Final is the purified product.

C.2 ELP NaCl/water Binary Phase Diagrams

The transition temperature (Tt) of ELPs in dilute solution is strongly governed by the hydrophobicity of the guest residue, with increasing hydrophobicity leading to lower Tt values. Hydrophobic residues such as isoleucine (ELP-I) and valine (ELP-V) exhibited the lowest Tt values (~42–45°C), whereas glycine (ELP-G), serine (ELP-S), and asparagine (ELP-N) produced significantly elevated Tt values (~55–65°C). Charged residues, including lysine (ELP-K) and aspartate (ELP-D), did not induce any measurable phase transition up to 90°C, consistent with the high hydration and polarity of their side chains due to ionization. High charge repulsion between chains **prevents collapse and aggregation,** so **no LCST-type transition** is detected. Even though salt NaCl is added up to 200mM, it is not enough to screen for electrostatic interactions given the

relatively low concentration of ELPs. These findings align with the experimental hydrophobicity scale developed by Urry et al., which defines Tt as a function of guest residue hydrophobicity in inverse temperature transition systems[16]. As shown in Figure 1, salt concentration had a modest secondary effect, slightly reducing Tt values across all sequences, likely due to salting out effect[23].

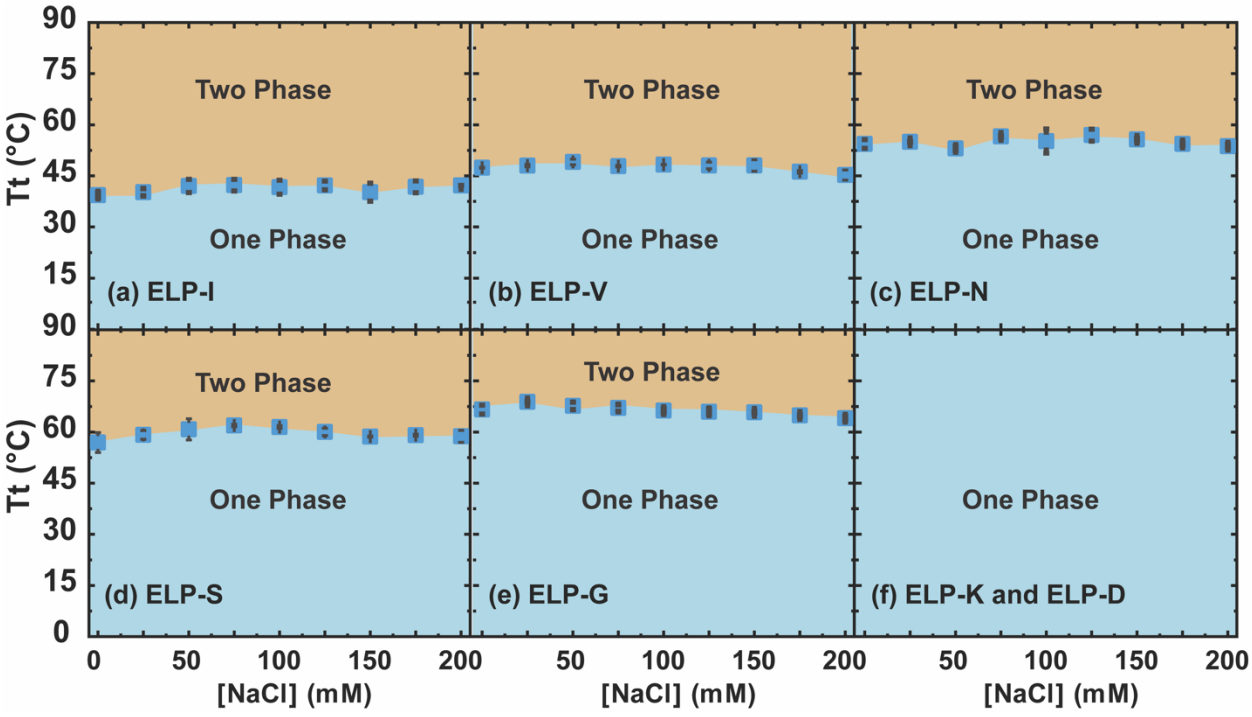


Figure C-8. The phase diagram of ELPs with His tag in the study in water/sodium chloride mixtures. (a) ELP-I. (b) ELP-V. (c) ELP-N. (d) ELP-S. (e) ELP-G. (f) ELP-K and ELP-D.

The absence of a 6xHis-tag does not significantly alter the transition temperature or phase boundaries for most of the sequences studied in sodium chloride/water mixtures as shown in Figure 5. The difference in transition temperature ranges from 1-8°C, where the largest difference is observed in ELP-S at 100-200mM sodium chloride concentration. Most transition temperatures demonstrate nearly complete overlap between tagged and untagged constructs as expected, suggesting that the short and charged His-tag does not substantially impact intermolecular

interactions, such as hydrophobic interactions, hydrogen bonding, or electrostatic interactions. Charged ELPs shows complete solubility across all the sodium chloride concentrations measured regardless of Tag presence.

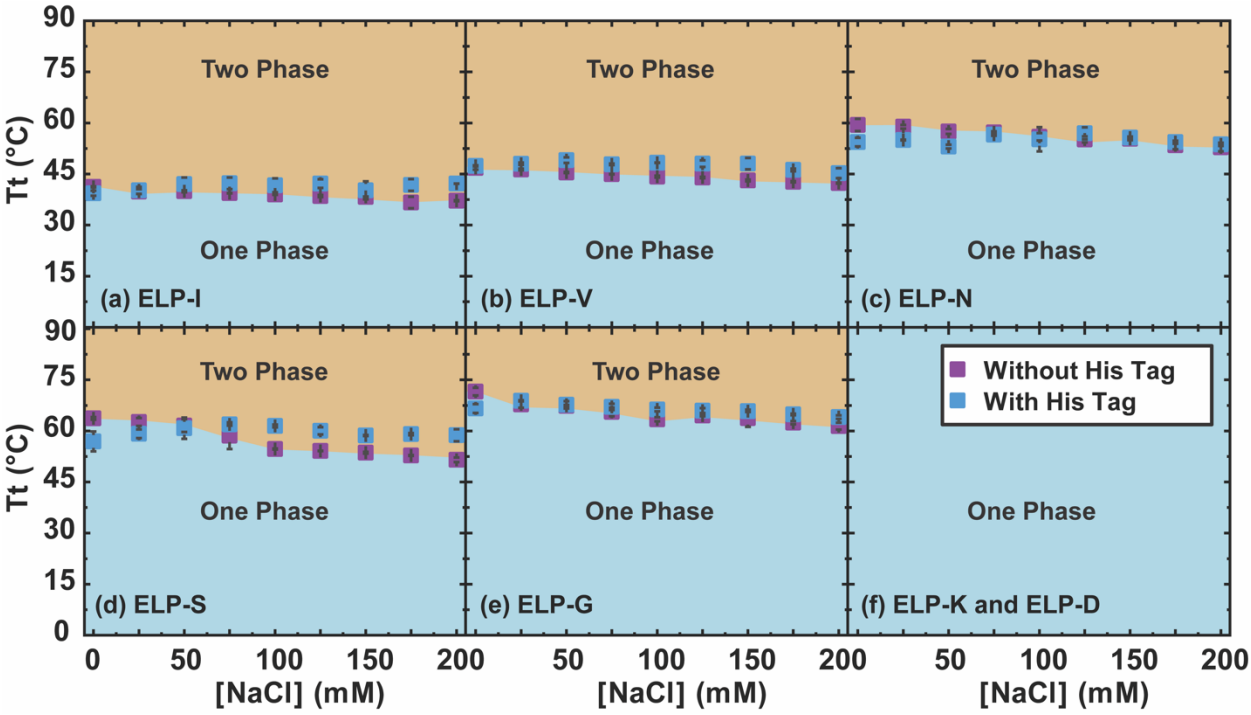


Figure C-9. The phase diagram of ELPs with 6X His and without 6X His tag in water/sodium chloride mixtures. The coloring reflects the phase boundaries for without 6X His tag. (a) ELP-I. (b) ELP-V. (c) ELP-N. (d) ELP-S. (e) ELP-G. (f) ELP-K and ELP-D. The grey areas are not measured due to either freezing or significant sample evaporation.

C.3 Thrombin Cleavage Kinetic Study

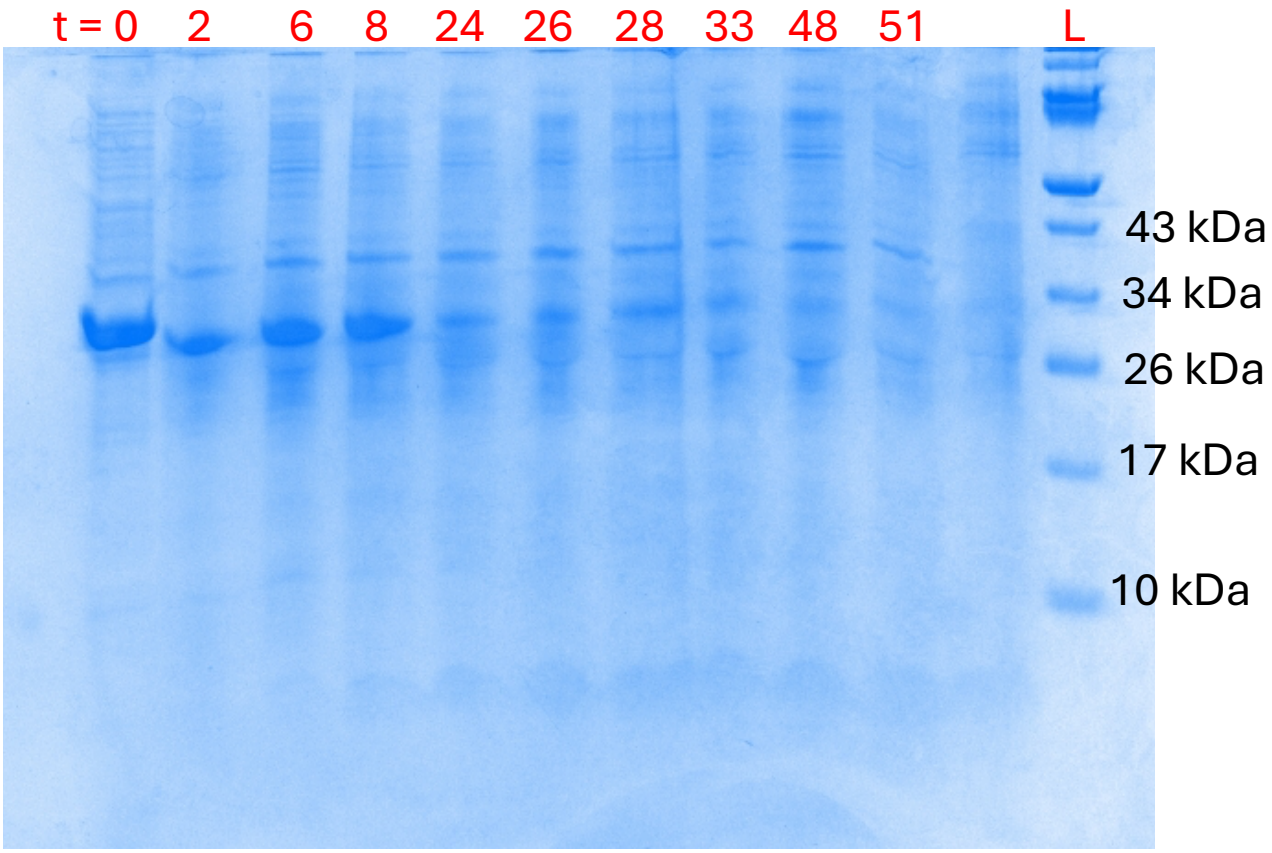


Figure C-10. SDS-PAGE of thrombin Kinetic Study of ELP-S. The time *t* are in hours. L stands for ladder.

C.4 MALDI-TOF

MALDI-TOF is used for verifying molecular weight of ELP-K (doubled molecular weight on SDS-PAGE) and successful thrombin cleavage. The MALDI matrix is sinapinic acid. Each sample is dissolved in 20 μL of 70% acetonitrile, 0.1% TEA, 29.9% H2O. 1 μL of HFIP was added to each dissolved sample. 1uL of each mixture was mixed with 4uL of matrix solution, 1 μL spotted and analyzed.

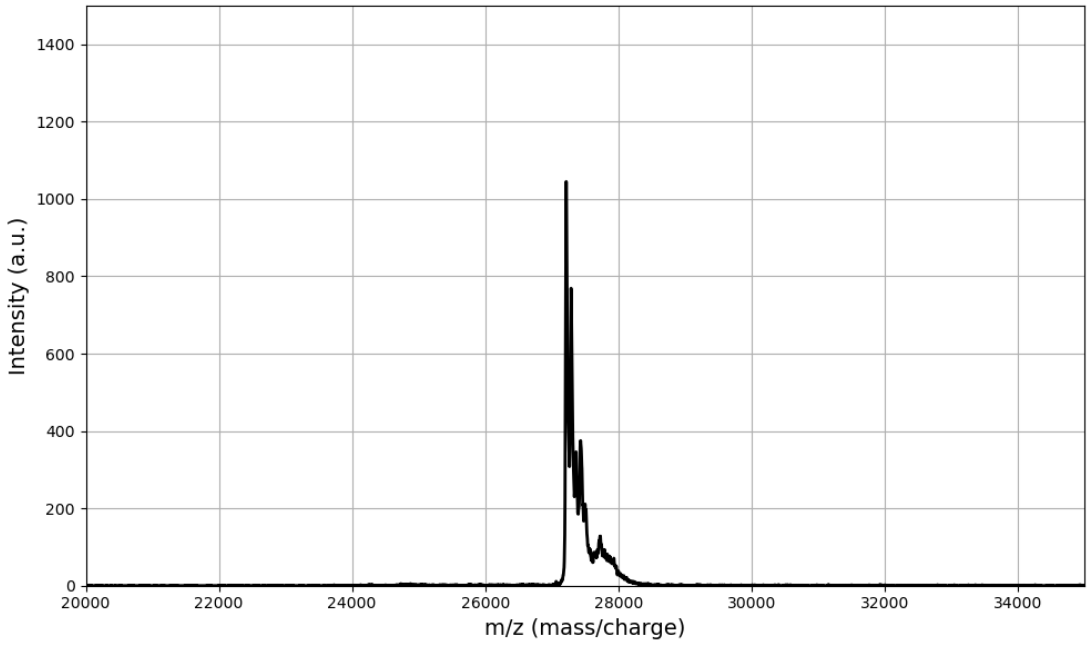


Figure C-11. MALDI-TOF mass spectroscopy for ELP-K. The peak molecular weight is 27.2kDa.

Table C-1. Summary of MALDI-TOF results of target molar mass vs achieved molar mass.

Sequence	Target Molar Mass (kDa)	Achieved Molar Mass (kDa)
ELP-I	22.4	22.5
ELP-V	22.4	22.4
ELP-S	22.4	21.4
ELP-N	20.8	20.8
ELP-G	22.4	22.1
ELP-K	22.4	22.8
ELP-D	22.4	22.5

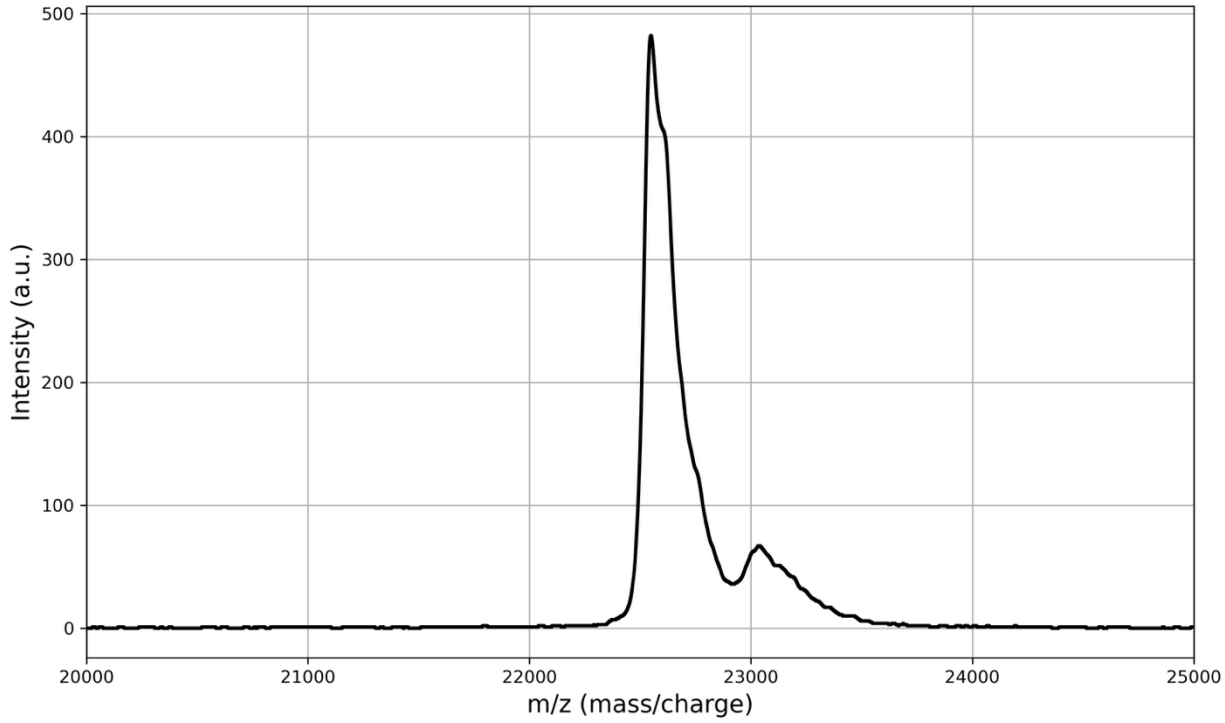


Figure C-12. MALDI-TOF mass spectroscopy for ELP-I after cleavage.

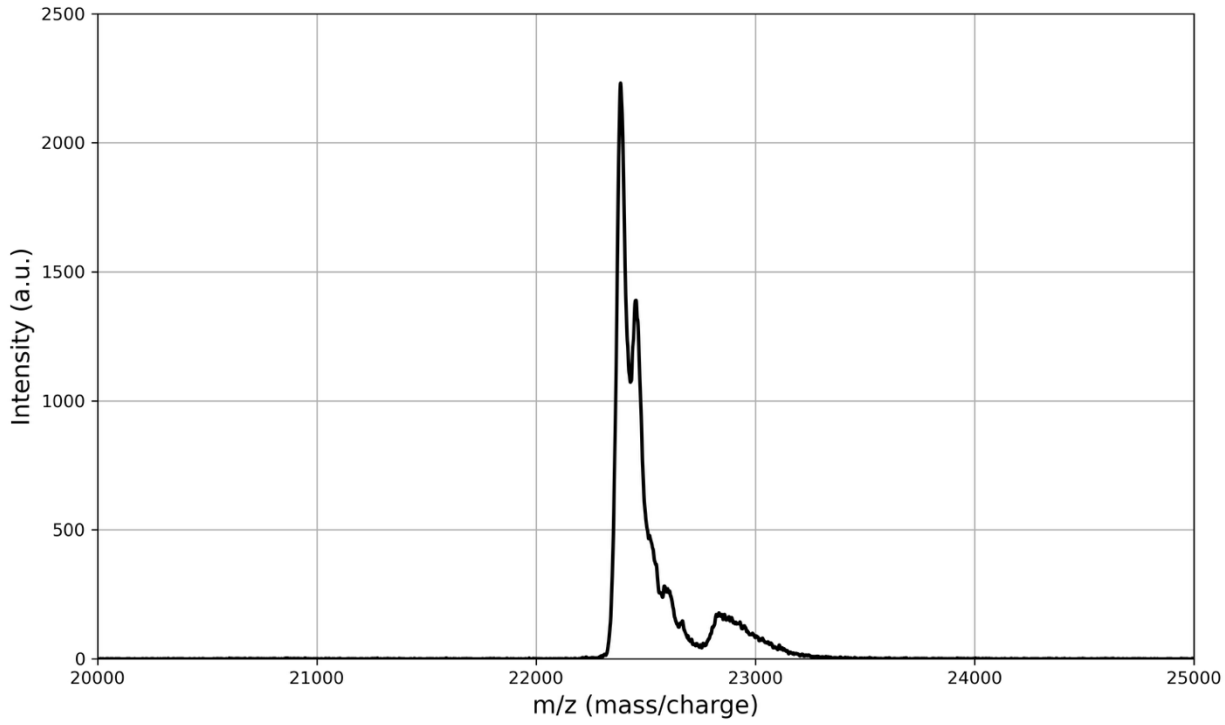


Figure C-13. MALDI-TOF mass spectroscopy for ELP-V after cleavage.

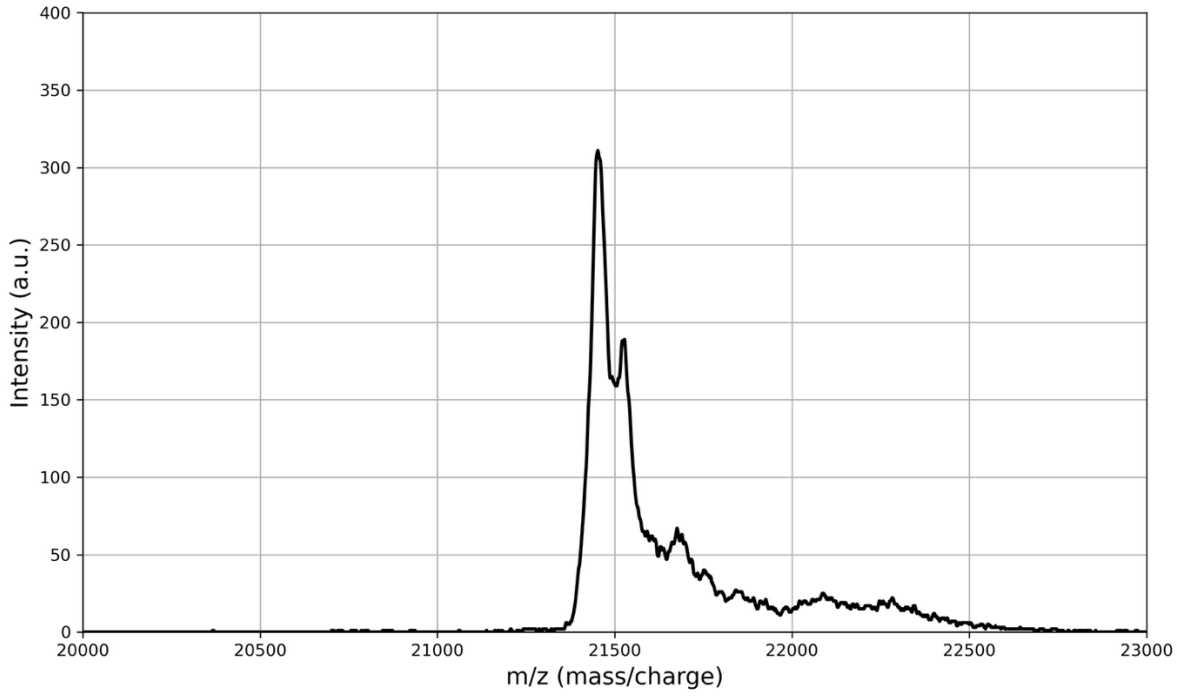


Figure C-14. MALDI-TOF mass spectroscopy for ELP-S after cleavage.

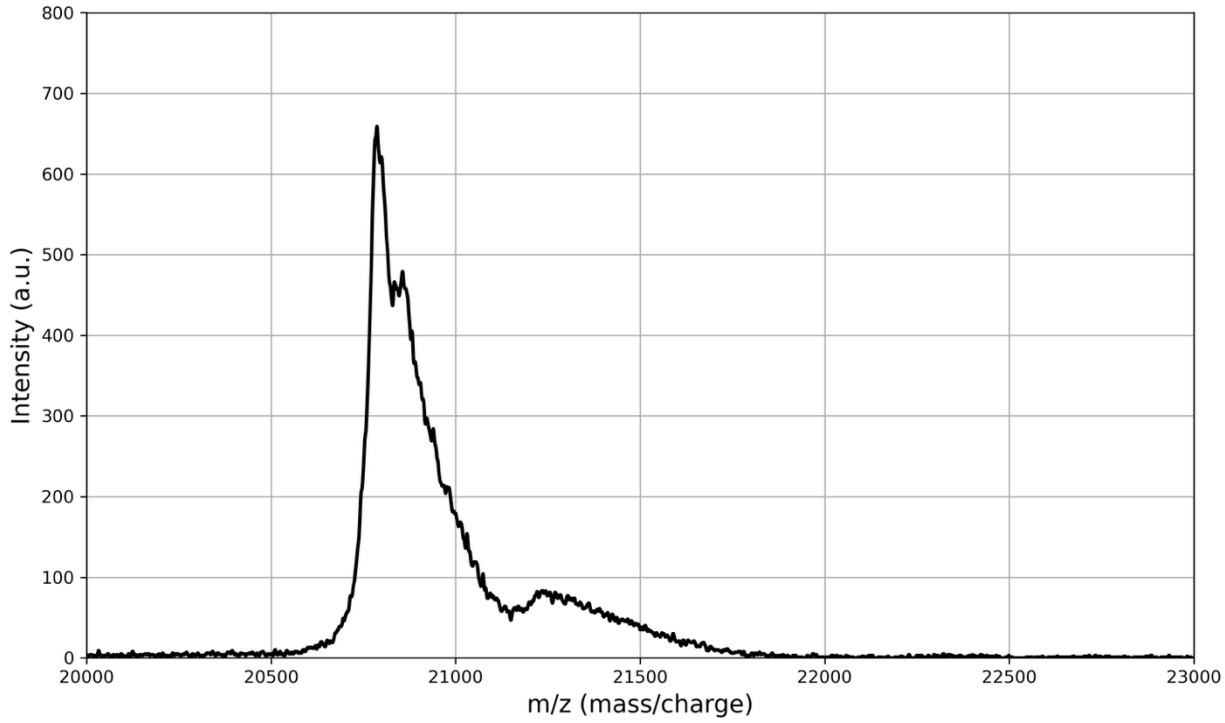


Figure C-15. MALDI-TOF mass spectroscopy for ELP-N after cleavage.

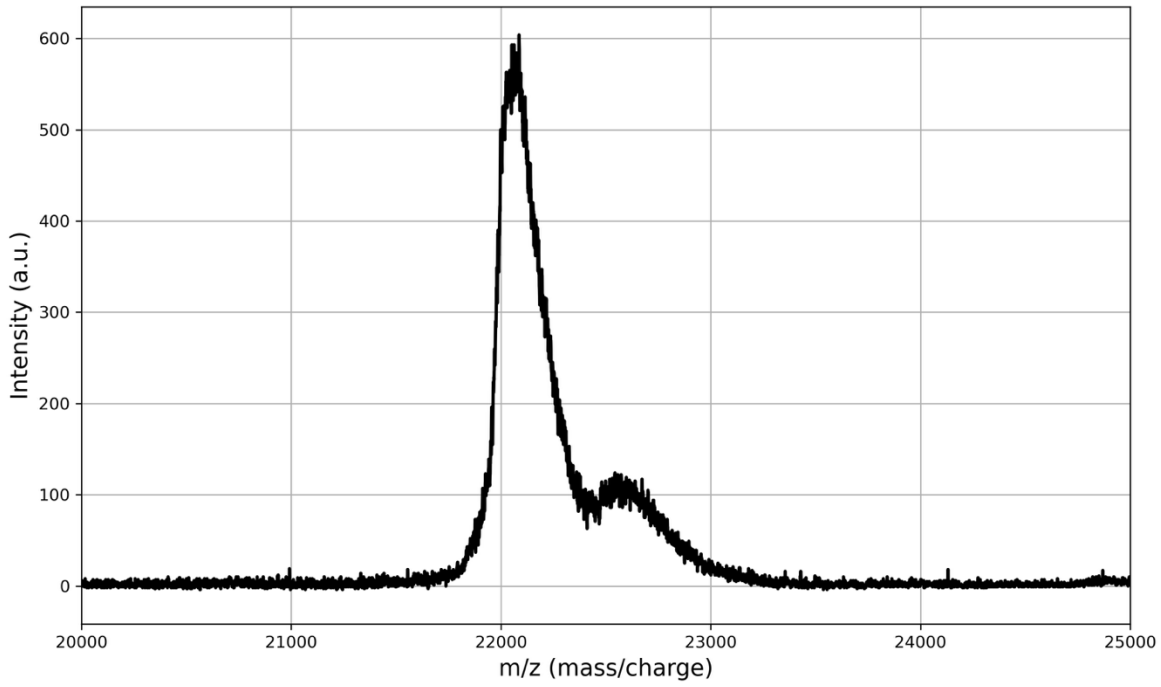


Figure C-15. MALDI-TOF mass spectroscopy for ELP-G after cleavage.

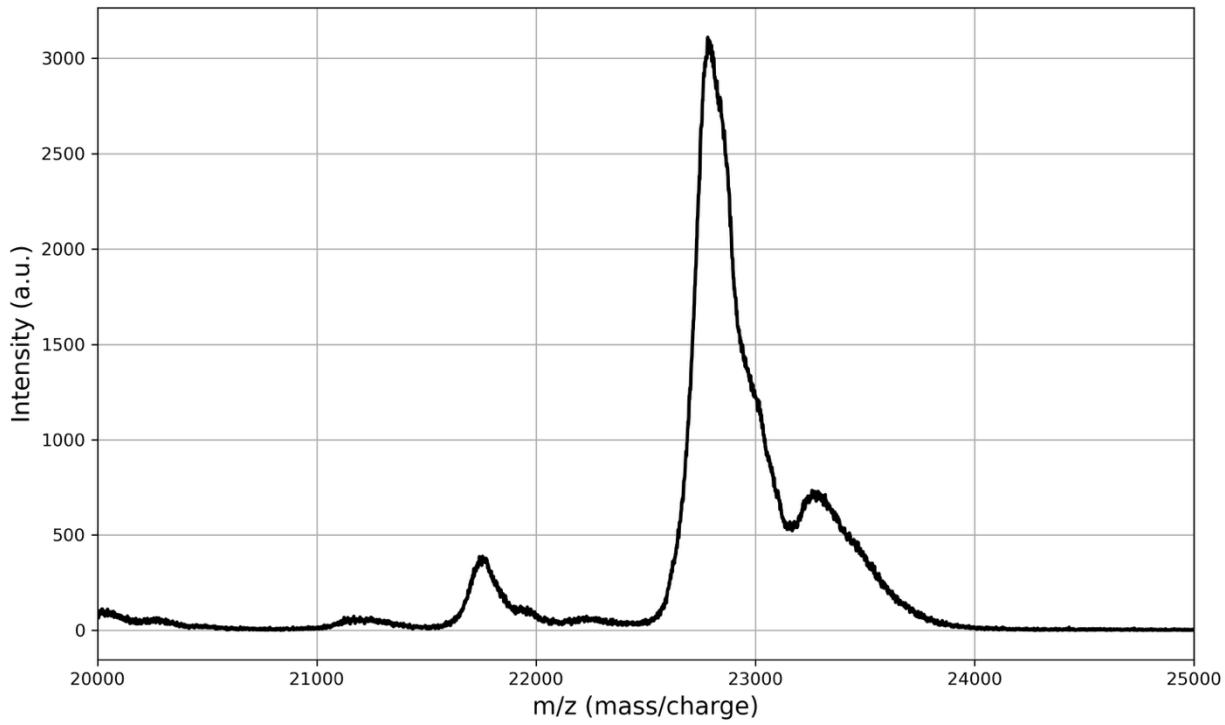


Figure C-16. MALDI-TOF mass spectroscopy for ELP-K after cleavage.

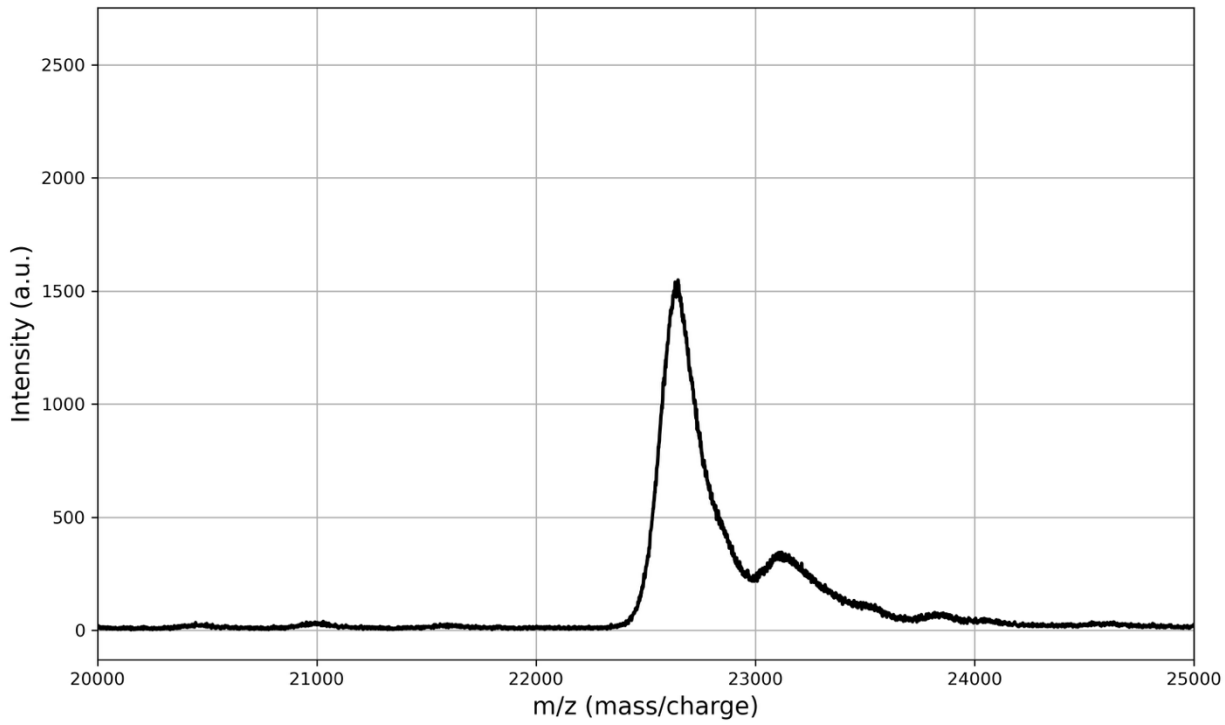


Figure C-17. MALDI-TOF mass spectroscopy for ELP-D after cleavage.

C.5 Turbidimetry

Given the large number of turbidimetry data for this article, the data is available at CRIPT database (https://www.criptapp.org).

Appendix D Supporting Information for Chapter 6

D.1 Python Scripts for Building Initial Structures

D.1.1 Build_Lamellar_Stack.py

```
import numpy as np

import math

import random

import matplotlib.pyplot as plt


# --- Simulation Parameters ---

# Simulation box dimensions

Lx, Ly, Lz = 25.0, 25.0, 25.0


# Total number of molecules (each molecule is a pair of atoms)

total_molecules = 3511

n_layers = 10


# Optional scaling factors to slightly expand or contract the
generated coordinates

# before centering. 1.0 means no scaling.

scale_factors = np.array([1.0, 1.0, 1.0])



# The z-spacing is determined by the FCC stacking of layers.

# The height of a tetrahedron with side length 'a' is
a*sqrt(6)/3.

# Here, the z-distance between the centers of two stacked
bilayers is related to this geometry.

spacing_z = Lz / (n_layers * (1 + math.sqrt(6) / 3))
```

```
# --- Lattice and Molecule Generation Functions ---

def create_hexagonal_plane(starting_xyz, spacing, n, m):
    """
    Creates a hexagonal lattice plane using basis vectors.

    Args:
        starting_xyz (np.array): The starting coordinate for the plane.
        spacing (float): The distance between adjacent atoms in the lattice.
        n (int): The number of unit cells to tile along the first basis vector.
        m (int): The number of unit cells to tile along the second basis vector.

    Returns:
        list: A list of atom coordinate arrays (np.array).
    """
    atoms = []
    # Define the hexagonal lattice basis vectors in the xy-plane
    a1 = np.array([spacing, 0, 0])
    a2 = np.array([spacing / 2.0, spacing * math.sqrt(3) / 2.0, 0])

    for i in range(n):
        for j in range(m):
```

```
            # Calculate the position as a linear combination of
basis vectors
            position = starting_xyz + i * a1 + j * a2
            atoms.append(position)
    return atoms


def create_bilayer(starting_xyz, spacing, n, m, is_even_layer):
    """
    Creates two parallel hexagonal planes, forming a bilayer.

    The 'is_even_layer' flag determines the orientation of the
molecule pairs.
    In even layers, atom 1 is below atom 2. In odd layers, it's
flipped.
    This helps create an alternating A-B-A-B pattern between
layers.
    """
    plane_one_start = starting_xyz
    plane_two_start = starting_xyz + np.array([0, 0, spacing_z])

    atoms_one = create_hexagonal_plane(plane_one_start, spacing,
n, m)
    atoms_two = create_hexagonal_plane(plane_two_start, spacing,
n, m)

    # Zip the two planes together to form molecule pairs.
    if is_even_layer:
        return list(zip(atoms_one, atoms_two))
    else:
```

```
        return list(zip(atoms_two, atoms_one))


def create_stacked_bilayer(starting_xyz, spacing, n, m):
    """Creates an FCC-like stacked bilayer (an 'A' layer and a
'B' layer)."""
    atoms = []
    # Create the 'A' layer
    atoms += create_bilayer(starting_xyz, spacing, n, m, True)


    # Define the offset for the 'B' layer for FCC (ABC)
stacking.
    # The offset shifts the next layer to sit in the hollows of
the layer below.
    # z_offset is derived from tetrahedral geometry for close-
packing.
    z_offset_dist = spacing_z * (1 + math.sqrt(6) / 3)
    xy_offset_dist = math.sqrt(3) / 6 * spacing
    offset = np.array([0.5 * spacing, xy_offset_dist,
z_offset_dist])


    # Create the 'B' layer, typically with slightly fewer atoms
to avoid edge issues
    atoms += create_bilayer(starting_xyz + offset, spacing, n -
1, m - 1, False)
    return atoms


def create_fcc_bilayers(starting_xyz, spacing, n, m,
num_stacked_layers):
    """Creates a full stack of FCC bilayers."""
    all_molecules = []
```

```
    # The z-distance to start the next A-B stack
    z_stack_offset = np.array([0, 0, 2 * spacing_z * (1 +
math.sqrt(6) / 3)])


    for i in range(num_stacked_layers):
        layer_start_pos = starting_xyz + i * z_stack_offset
        all_molecules += create_stacked_bilayer(layer_start_pos,
spacing, n, m)
    return all_molecules


def rescale_and_center_coordinates(molecules, scale_factors,
box_dims):
    """
    Rescales and centers the entire system of molecules in the
box.
    """
    # 1. Apply scaling factors to all atoms first
    scaled_molecules = []
    for atom1, atom2 in molecules:
        scaled_molecules.append((atom1 * scale_factors, atom2 *
scale_factors))


    # 2. Calculate the Center of Mass (CoM) of the now-scaled
system
    total_atoms = 0
    com = np.zeros(3)
    if not scaled_molecules:
        return []
```

```
    for atom1, atom2 in scaled_molecules:
        com += atom1
        com += atom2
        total_atoms += 2

    if total_atoms > 0:
        com /= total_atoms

    # 3. Determine the vector needed to move the CoM to the box
center
    box_center = box_dims / 2.0
    translation_vector = box_center - com

    # 4. Apply the translation to all atoms
    centered_molecules = []
    for atom1, atom2 in scaled_molecules:
        centered_molecules.append((atom1 + translation_vector,
atom2 + translation_vector))

    return centered_molecules



def wrap_coordinates(molecules, box_dims):
    """
    Applies periodic boundary conditions to all atom
coordinates.

    Args:
```

```
        molecules (list): A list of molecule tuples, where each
tuple contains
                          two np.array coordinates.
        box_dims (np.array): An array with the [Lx, Ly, Lz]
dimensions.

    Returns:
        list: A new list of molecules with wrapped coordinates.
    """
    wrapped_molecules = []
    for atom1, atom2 in molecules:
        # Use the modulo operator to wrap each coordinate into
the periodic box
        wrapped_atom1 = atom1 % box_dims
        wrapped_atom2 = atom2 % box_dims
        wrapped_molecules.append((wrapped_atom1, wrapped_atom2))
    return wrapped_molecules


# --- Main Script Logic ---

# Determine the number of atoms per plane (n, m) to generate
more
# molecules than required. We need to slightly overfill the box
to account
# for the hexagonal packing not perfectly fitting a square.
# n=20, m=23 should provide enough atoms.
n, m = 21, 24
box_dimensions = np.array([Lx, Ly, Lz])
```

```
# Define starting position and lattice spacing to fit the box
# Spacing is based on the number of unit cells in the x-direction
starting_xyz = np.array([0.0, 0.0, 0.5])
spacing = Lx / n


# 1. Generate the full stack of molecules (atom pairs)
fcc_bilayer_molecules = create_fcc_bilayers(starting_xyz, spacing, n, m, n_layers // 2)


# 2. Rescale and center the coordinates
centered_molecules = rescale_and_center_coordinates(fcc_bilayer_molecules, scale_factors, box_dimensions)


# 3. Wrap all generated coordinates into the periodic box
wrapped_fcc_molecules = wrap_coordinates(centered_molecules, box_dimensions)


# 4. Randomly remove extra molecules to match the desired total
print(f"Generated, centered, and wrapped {len(wrapped_fcc_molecules)} molecules.")
print(f"Downsampling to {total_molecules} molecules.")
downsampled_molecules = random.sample(wrapped_fcc_molecules, total_molecules)


# --- Write LAMMPS Data File ---
```

```
# Dynamically create the header with correct atom and bond
counts

header = f"""LAMMPS data file: Hexagonal Lamellar Stack
(Centered & Wrapped)


{total_molecules * 2} atoms

{total_molecules} bonds

2 atom types

1 bond types


0.0 {Lx:.2f} xlo xhi

0.0 {Ly:.2f} ylo yhi

0.0 {Lz:.2f} zlo zhi


Atoms # bond

"""


output_filename = "initial_lamellar_55P.data"

print(f"Writing {total_molecules} molecules to
{output_filename}...")


with open(output_filename, "w") as f:

    f.write(header)

    f.write("\n")

    # Write atom coordinates

    for i, (atom1, atom2) in enumerate(downsampled_molecules):

        molecule_id = i + 1

        atom1_id = 2 * i + 1
```

```
        atom2_id = 2 * i + 2
        # Format: atom-ID molecule-ID atom-type x y z
        f.write(f"{atom1_id} {molecule_id} 1 {atom1[0]:.4f}
{atom1[1]:.4f} {atom1[2]:.4f}\n")
        f.write(f"{atom2_id} {molecule_id} 2 {atom2[0]:.4f}
{atom2[1]:.4f} {atom2[2]:.4f}\n")

    f.write("\nBonds\n\n")
    # Write bond information
    for i in range(total_molecules):
        bond_id = i + 1
        atom1_id = 2 * i + 1
        atom2_id = 2 * i + 2
        # Format: bond-ID bond-type atom1-ID atom2-ID
        f.write(f"{bond_id} 1 {atom1_id} {atom2_id}\n")

print("Done.")
```

D.1.1 Build_HEX.py

```
import numpy as np
import math
import itertools
import random

L_x = 25
L_y = 25
L_z = 25
num_molecules = 4915
```

```
def create_single_vector(side_length=1.0, rotation=0, z=0):
    coordinates = np.array(
        [math.cos(rotation) * side_length, math.sin(rotation) *
side_length, z]
    )
    normal_vector = np.array([math.cos(rotation),
math.sin(rotation), 0])
    return (coordinates, normal_vector)



def create_triangle(side_length=1.0, rotation=0, z=0):
    output = []
    for i in range(3):
        output.append(
            create_single_vector(
                side_length=side_length, rotation=rotation + 2 *
i * math.pi / 3, z=z
            )
        )
    return output



def create_cylinder(side_length=1.0, initial_z=0,
initial_rotation=0, num_rows=18):
    output = []
    z_spacing = L_z / num_rows
    rotation = initial_rotation
```

```
    for i in range(num_rows):
        output.append(
            create_triangle(
                side_length=side_length, rotation=rotation,
z=initial_z + z_spacing * i
            )
        )
        rotation += math.pi / 3
    return list(itertools.chain.from_iterable(output))



def create_rotating_cylinder_row(
    side_length=1.0,
    initial_z=0,
    initial_rotation=0,
    num_rows=18,
    distance_between_cylinders=5,
    num_cylinders_per_row=4,
    initial_x=0,
):
    output = []
    x = initial_x
    rotation = initial_rotation
    for i in range(num_cylinders_per_row):
        for triangle_coordinates, normal_vector in
create_cylinder(
            side_length=side_length,
            initial_z=initial_z,
```

```
            initial_rotation=rotation,
            num_rows=num_rows,
        ):
            output.append((triangle_coordinates + np.array([x,
0, 0]), normal_vector))
        x += distance_between_cylinders
        rotation += math.pi / 6
    return output



def create_identical_cylinder_plane(
    side_length=1.0,
    initial_z=0,
    initial_rotation=0,
    num_rows=18,
    distance_between_cylinders=5,
    num_cylinders_per_row=5,
    num_cylinder_columns=5,
    initial_x=0,
    initial_y=0,
):
    output = []
    y = initial_y
    rotation = initial_rotation
    for i in range(num_cylinder_columns):
        num_cylinders_this_row = num_cylinders_per_row - (1 if
i % 2 == 1 else 0)
```

```
        this_x = initial_x + (distance_between_cylinders / 2 if
i % 2 == 1 else 0)
        for triangle_coordinates, normal_vector in
create_rotating_cylinder_row(
            side_length=side_length,
            initial_z=initial_z,
            initial_rotation=rotation,
            num_rows=num_rows,

distance_between_cylinders=distance_between_cylinders,
            num_cylinders_per_row=num_cylinders_this_row,
            initial_x=this_x,
        ):
            output.append((triangle_coordinates + np.array([0,
y, 0]), normal_vector))
        y += distance_between_cylinders * math.sqrt(3) / 2
        rotation += math.pi / 3
    return output



def create_lammps_file(molecule_inputs, side_length):
    header = f"""LAMMPS data file: Lamellar Stack\n
{len(molecule_inputs) * 2} atoms
2 atom types
{len(molecule_inputs)} bonds
1 bond types\n
0.0 {L_x:.2f} xlo xhi
0.0 {L_y:.2f} ylo yhi
0.0 {L_z:.2f} zlo zhi\n
```

```
Atoms # bond\n
"""
    with open("downsampled_cylinders.data", "w") as f:
        f.write(header)
        for i, single_molecule in enumerate(molecule_inputs):
            b = single_molecule[0]
            normal_vector = single_molecule[1]
            a = b + normal_vector * side_length
            f.write(f"{2*i+1} {i+1} 1 {a[0]:.4f} {a[1]:.4f}
{a[2]:.4f} 0 0 0\n")
            f.write(f"{2*i+2} {i+1} 2 {b[0]:.4f} {b[1]:.4f}
{a[2]:.4f} 0 0 0\n")
        f.write("\nBonds\n\n")
        for i in range(1, len(molecule_inputs) + 1):
            f.write(f"{i} 1 {2*i-1} {2*i}\n")



output = create_identical_cylinder_plane(
    initial_x=1.7,
    initial_y=1.5,
    initial_z=1.0,
    side_length=0.65,
    num_cylinder_columns=8,
    distance_between_cylinders=3.6,
    num_cylinders_per_row=7,
    num_rows=34, # change this
)
print(len(output))
```

```
assert len(output) >= num_molecules
downsampled = random.sample(output, num_molecules)
create_lammps_file(downsampled, side_length=0.65)
```

D.1.3 Build_Perforated_Lamellar.py

```
import numpy as np
import math
import itertools
import random
import matplotlib.pyplot as plt

# --- Simulation Parameters ---
# Simulation box dimensions
Lx, Ly, Lz = 25.0, 25.0, 25.0

# Total number of molecules (each molecule is a pair of atoms)
total_molecules = 3511
n_layers = 10 # Number of A-B stacked bilayer units

# Optional scaling factors to slightly expand or contract the
generated coordinates
# before centering. 1.0 means no scaling.
scale_factors = np.array([1.0, 1.0, 0.98])

# Perforation parameters
perforation_radius = 2.5
cylinder_num_rows = 35
cylinder_side_length = 0.65 # This will now be primarily used to
define how far atom2 is from the cylinder's central axis
```

```
# --- NEW PARAMETERS: Atom Radii and Buffer ---
atom_radius_1 = 0.5 # Radius of atom type 1
atom2_radius = 1.0 # Radius of atom type 2 (can be different from atom_radius_1)
buffer_from_boundary = max(atom_radius_1, atom2_radius) * 1.3 # Add a small buffer, e.g., 10% more than the larger radius


# --- Input Validation for Simulation Parameters ---
if Lx <= 0 or Ly <= 0 or Lz <= 0:
    raise ValueError("Box dimensions (Lx, Ly, Lz) must be positive.")
if total_molecules <= 0:
    raise ValueError("Total molecules must be a positive integer.")
if n_layers <= 0:
    raise ValueError("Number of layers (n_layers) must be a positive integer.")
if not all(s > 0 for s in scale_factors):
    raise ValueError("Scale factors must be positive.")
if perforation_radius <= 0:
    raise ValueError("Perforation radius must be positive.")
if cylinder_num_rows <= 0:
    raise ValueError("Cylinder number of rows must be positive.")
if cylinder_side_length <= 0:
    raise ValueError("Cylinder side length must be positive.")
if atom_radius_1 <= 0:
    raise ValueError("Atom 1 radius must be positive.")
```

```
if atom2_radius <= 0:
    raise ValueError("Atom 2 radius must be positive.")
if buffer_from_boundary < max(atom_radius_1, atom2_radius):
    raise ValueError("Buffer from boundary should at least be
equal to the maximum atom radius.")



# The z-spacing is determined by the FCC stacking of layers.
# The height of a tetrahedron with side length 'a' is
a*sqrt(6)/3.
# Here, the z-distance between the centers of two stacked
bilayers is related to this geometry.
spacing_z = Lz / (n_layers * (1 + math.sqrt(6) / 3))


# --- Lattice and Molecule Generation Functions ---


def create_hexagonal_plane(starting_xyz, spacing, n, m):
    """
    Creates a hexagonal lattice plane using basis vectors.


    Args:
        starting_xyz (np.array): The starting coordinate for the
plane.
        spacing (float): The distance between adjacent atoms in
the lattice.
        n (int): The number of unit cells to tile along the
first basis vector.
        m (int): The number of unit cells to tile along the
second basis vector.
```

```
    Returns:
        list: A list of atom coordinate arrays (np.array).
    """
    if not (isinstance(starting_xyz, np.ndarray) and
starting_xyz.shape == (3,)):
        raise TypeError("starting_xyz must be a 3-element numpy
array.")
    if spacing <= 0:
        raise ValueError("Spacing must be positive.")
    if n <= 0 or m <= 0:
        raise ValueError("n and m (number of unit cells) must be
positive integers.")

    atoms = []
    # Define the hexagonal lattice basis vectors in the xy-plane
    a1 = np.array([spacing, 0, 0])
    a2 = np.array([spacing / 2.0, spacing * math.sqrt(3) / 2.0,
0])

    for i in range(n):
        for j in range(m):
            position = starting_xyz + i * a1 + j * a2
            atoms.append(position)
    return atoms

def create_bilayer(starting_xyz, spacing, n, m, is_even_layer):
    """
    Creates two parallel hexagonal planes, forming a bilayer.
```

```
    The 'is_even_layer' flag determines the orientation of the
molecule pairs.

    In even layers, atom 1 is below atom 2. In odd layers, it's
flipped.

    This helps create an alternating A-B-A-B pattern between
layers.

    """

    plane_one_start = starting_xyz

    plane_two_start = starting_xyz + np.array([0, 0, spacing_z])


    atoms_one = create_hexagonal_plane(plane_one_start, spacing,
n, m)

    atoms_two = create_hexagonal_plane(plane_two_start, spacing,
n, m)


    # Ensure planes have the same number of atoms before zipping

    if len(atoms_one) != len(atoms_two):

        raise RuntimeError("Mismatch in atom count between
planes during bilayer creation.")


    if is_even_layer:

        return list(zip(atoms_one, atoms_two))

    else:

        return list(zip(atoms_two, atoms_one))


def create_stacked_bilayer(starting_xyz, spacing, n, m):

    """Creates an FCC-like stacked bilayer (an 'A' layer and a
'B' layer)."""

    atoms = []

    # Create the 'A' layer
```

```
    atoms += create_bilayer(starting_xyz, spacing, n, m, True)


    # Define the offset for the 'B' layer for FCC (ABC)
stacking.
    # The offset shifts the next layer to sit in the hollows of
the layer below.
    # z_offset is derived from tetrahedral geometry for close-
packing.
    z_offset_dist = spacing_z * (1 + math.sqrt(6) / 3)
    xy_offset_dist = math.sqrt(3) / 6 * spacing
    offset = np.array([0.5 * spacing, xy_offset_dist,
z_offset_dist])


    # Create the 'B' layer, typically with slightly fewer atoms
to avoid edge issues
    n_b = max(1, n - 1)
    m_b = max(1, m - 1)
    atoms += create_bilayer(starting_xyz + offset, spacing, n_b,
m_b, False)
    return atoms


def create_fcc_bilayers(starting_xyz, spacing, n, m,
num_stacked_layers):
    """Creates a full stack of FCC bilayers."""
    if num_stacked_layers <= 0:
        raise ValueError("Number of stacked layers must be
positive.")


    all_molecules = []
    # The z-distance to start the next A-B stack
```

```
    z_stack_offset = np.array([0, 0, 2 * spacing_z * (1 +
math.sqrt(6) / 3)])


    for i in range(num_stacked_layers):
        layer_start_pos = starting_xyz + i * z_stack_offset
        all_molecules += create_stacked_bilayer(layer_start_pos,
spacing, n, m)
    return all_molecules


def rescale_and_center_coordinates(molecules, scale_factors,
box_dims):
    """
    Rescales and centers the entire system of molecules in the
box.
    """
    if not molecules:
        print("Warning: No molecules to rescale and center.
Returning empty list.")
        return []
    if not (isinstance(scale_factors, np.ndarray) and
scale_factors.shape == (3,) and np.all(scale_factors > 0)):
        raise ValueError("scale_factors must be a 3-element
numpy array with positive values.")
    if not (isinstance(box_dims, np.ndarray) and box_dims.shape
== (3,) and np.all(box_dims > 0)):
        raise ValueError("box_dims must be a 3-element numpy
array with positive values.")


    scaled_molecules = []
    for atom1, atom2 in molecules:
```

```
        atom1_np = np.asarray(atom1)
        atom2_np = np.asarray(atom2)
        scaled_molecules.append((atom1_np * scale_factors,
atom2_np * scale_factors))

    total_atoms = 0
    com = np.zeros(3)

    for atom1, atom2 in scaled_molecules:
        com += atom1
        com += atom2
        total_atoms += 2

    if total_atoms == 0:
        print("Warning: No atoms found after scaling for CoM
calculation.")
        return []

    com /= total_atoms

    box_center = box_dims / 2.0
    translation_vector = box_center - com

    centered_molecules = []
    for atom1, atom2 in scaled_molecules:
        centered_molecules.append((atom1 + translation_vector,
atom2 + translation_vector))
```

```
    return centered_molecules



def wrap_coordinates(molecules, box_dims):
    """
    Applies periodic boundary conditions to all atom
coordinates.

    Args:
        molecules (list): A list of molecule tuples, where each
tuple contains
                          two np.array coordinates.
        box_dims (np.array): An array with the [Lx, Ly, Lz]
dimensions.

    Returns:
        list: A new list of molecules with wrapped coordinates.
    """
    if not molecules:
        print("Warning: No molecules to wrap. Returning empty
list.")
        return []
    if not (isinstance(box_dims, np.ndarray) and box_dims.shape
== (3,) and np.all(box_dims > 0)):
        raise ValueError("box_dims must be a 3-element numpy
array with positive values.")

    wrapped_molecules = []
    for atom1, atom2 in molecules:
```

```
        wrapped_atom1 = atom1 % box_dims
        wrapped_atom2 = atom2 % box_dims
        wrapped_molecules.append((wrapped_atom1, wrapped_atom2))
    return wrapped_molecules


def create_perforation(starting_xyz, spacing, n, m,
num_stacked_layers, perforations, perforation_radius,
box_dimensions, scale_factors):
    """
    Creates a perforated lamellar structure by generating a full
stack of FCC bilayers
    and then adding perforations at specified coordinates.
    The perforations are defined as a list of (x,y) coordinates
where the perforations should be placed.
    """
    if not isinstance(perforations, list) or not
all(isinstance(p, np.ndarray) and p.shape == (2,) for p in
perforations):
        raise ValueError("Perforations must be a list of 2-
element numpy arrays (x,y coordinates).")


    print("Generating full FCC bilayers...")
    all_molecules = create_fcc_bilayers(starting_xyz, spacing,
n, m, num_stacked_layers)
    print(f"Generated {len(all_molecules)} molecules
initially.")


    print("Rescaling and centering coordinates...")
    centered_molecules =
rescale_and_center_coordinates(all_molecules, scale_factors,
box_dimensions)
```

```
    print("Wrapping coordinates into the periodic box...")
    wrapped_fcc_molecules = wrap_coordinates(centered_molecules,
box_dimensions)


    def is_within_perforation(atom_coords, perforations,
radius):
        """Helper function to check if an atom is within any
perforation radius."""
        for perforation_center_xy in perforations:
            distance_sq = (atom_coords[0] -
perforation_center_xy[0])**2 + \
                          (atom_coords[1] -
perforation_center_xy[1])**2
            if distance_sq < radius**2:
                return True
        return False


    print(f"Applying {len(perforations)} perforations with
radius {perforation_radius}...")
    perforated_molecules = []
    for blues, reds in wrapped_fcc_molecules:
        if not (is_within_perforation(blues, perforations,
perforation_radius) or
                is_within_perforation(reds, perforations,
perforation_radius)):
            perforated_molecules.append((blues, reds))


    print(f"Remaining molecules after perforation:
{len(perforated_molecules)}")
```

```
    return perforated_molecules


def create_single_vector(side_length=1.0, rotation=0, z=0):
    """Creates a 3D coordinate and a 2D normal vector for a
point on a circle."""
    if side_length <= 0:
        raise ValueError("Side length must be positive.")
    coordinates = np.array(
        [math.cos(rotation) * side_length, math.sin(rotation) *
side_length, z]
    )
    normal_vector = np.array([math.cos(rotation),
math.sin(rotation), 0])
    return (coordinates, normal_vector)



def create_triangle(side_length=1.0, rotation=0, z=0):
    """Creates the three points of a triangle in the xy-plane at
a given z."""
    output = []
    for i in range(3):
        output.append(
            create_single_vector(
                side_length=side_length, rotation=rotation + 2 *
i * math.pi / 3, z=z
            )
        )
    return output
```

```
def create_cylinder(side_length=1.0, initial_z=0,
initial_rotation=0, num_rows=18):
    """
    Creates a stack of triangles forming a cylinder-like
structure.
    Each 'triangle' here is a set of three points around a
central axis.
    """
    if num_rows <= 0:
        raise ValueError("Number of cylinder rows must be
positive.")
    output = []
    z_spacing = Lz / num_rows
    rotation = initial_rotation
    for i in range(num_rows):
        output.append(
            create_triangle(
                side_length=side_length, rotation=rotation,
z=initial_z + z_spacing * i
            )
        )
        rotation += math.pi / 3
    return list(itertools.chain.from_iterable(output))


def fill_perforation(
    starting_xyz,
    spacing,
    n,
```

```
    m,
    num_stacked_layers,
    perforations,
    perforation_radius,
    num_rows,
    side_length, # This side_length refers to the radius of the
circle on which atom2s are placed in the cylinder.
    initial_rotation,
    box_dimensions,
    scale_factors,
    atom_radius_1 # Pass atom_radius_1 to determine bond length
):
    """
    Fills the perforation with a cylindrical structure.
    The cylinder is created with a specified number of rows and
side length.
    """
    print("Creating cylinder for filling...")
    cylinder = create_cylinder(
        side_length=side_length, # This side_length determines
atom2's radial position
        initial_z=starting_xyz[2],
        initial_rotation=initial_rotation,
        num_rows=num_rows,
    )

    print("Creating perforated lamellar structure...")
    perforated_molecules = create_perforation(
```

```
        starting_xyz, spacing, n, m, num_stacked_layers,
perforations, perforation_radius, box_dimensions, scale_factors
    )


    print(f"Adding cylinder atoms to fill perforations (approx.
{len(cylinder) * len(perforations)} new atoms)...")
    filled_molecules = list(perforated_molecules)


    for atom_point, normal_vec in cylinder:
        for perforation_center_xy in perforations:
            perforation_center_3d =
np.array([perforation_center_xy[0], perforation_center_xy[1],
0.0])
            # atom2_coords are placed at the cylinder points
            atom2_coords = perforation_center_3d + atom_point
            # atom1_coords are placed atom_radius_1 distance
away from atom2_coords along the normal vector
            atom1_coords = atom2_coords + normal_vec *
atom_radius_1


            # The z-coordinate from atom_point (relative to
cylinder's initial_z) is maintained
            # The global z position is atom_point[2] +
starting_xyz[2] for the base cylinder.
            # However, since the whole system will be re-
centered and wrapped later,
            # we just place them relative to
perforation_center_3d and let the final
            # scaling/wrapping handle their ultimate position
within the global box.
```

```
            filled_molecules.append((atom1_coords,
atom2_coords))

    return filled_molecules



# --- Main Script Logic ---
def main():
    print("--- Starting Molecular Structure Generation ---")

    global Lx, Ly, Lz, total_molecules, n_layers, scale_factors
    global perforation_radius, cylinder_num_rows,
cylinder_side_length
    global atom_radius_1, atom2_radius, buffer_from_boundary #
Access new parameters

    box_dimensions = np.array([Lx, Ly, Lz])

    n_guess = 21
    m_guess = 24

    spacing = Lx / n_guess
    print(f"Calculated lattice spacing: {spacing:.4f}")

    # MODIFICATION: Offset starting_xyz for Z to account for
atom radius and boundary buffer
    # The lowest Z coordinate should be at least
'buffer_from_boundary'
    starting_xyz = np.array([0.0, 0.0, buffer_from_boundary])
```

```
    print(f"Adjusted starting_xyz for Z-offset:
{starting_xyz[2]:.4f}")



    try:
        perforated_bilayer_molecules = fill_perforation(
            starting_xyz, spacing, n_guess, m_guess, n_layers //
2,
            perforations, perforation_radius, cylinder_num_rows,
cylinder_side_length,
            initial_rotation=0, box_dimensions=box_dimensions,
scale_factors=scale_factors,
            atom_radius_1=atom_radius_1 # Pass atom_radius_1 to
the function
        )
    except Exception as e:
        print(f"Error during perforation filling: {e}")
        return


    print("Final rescaling, centering, and wrapping for all
molecules...")
    final_molecules =
rescale_and_center_coordinates(perforated_bilayer_molecules,
scale_factors, box_dimensions)
    final_wrapped_molecules = wrap_coordinates(final_molecules,
box_dimensions)



    print(f"Generated and processed
{len(final_wrapped_molecules)} total molecules before
downsampling.")
```

```
    if len(final_wrapped_molecules) < total_molecules:
        print(f"Warning: Only {len(final_wrapped_molecules)}
molecules generated, which is less than the target
{total_molecules}.")
        actual_total_molecules = len(final_wrapped_molecules)
    else:
        actual_total_molecules = total_molecules


    print(f"Downsampling to {actual_total_molecules}
molecules...")
    try:
        downsampled_molecules =
random.sample(final_wrapped_molecules, actual_total_molecules)
    except ValueError as e:
        print(f"Error during downsampling: {e}. This usually
means the generated count is less than target.")
        print("Using all generated molecules instead of
downsampling.")
        downsampled_molecules = final_wrapped_molecules


    # --- Verify coordinates are within bounds before writing --
-
    print("Verifying all final atom coordinates are within box
bounds...")
    for mol_idx, (atom1, atom2) in
enumerate(downsampled_molecules):
        for atom_idx, atom in enumerate([atom1, atom2]):
            # IMPORTANT: The check for radius is NOT on the
center coordinate itself,
```

```
            # but that the range (center - radius) to (center +
radius) does not cross the boundary.

            # For periodic boundary conditions (which
wrap_coordinates handles), this is less of an issue,

            # but for initial placement, having a buffer is
good.

            # The current check (0 <= coord < Dim) is for the
center only, which is what LAMMPS uses.

            if not (0 <= atom[0] < Lx and 0 <= atom[1] < Ly and
0 <= atom[2] < Lz):

                print(f"ERROR: Atom {mol_idx*2 + atom_idx + 1}
({atom_idx+1} of molecule {mol_idx+1}) "

                      f"coordinates {atom} are outside box [0,
{Lx}), [0, {Ly}), [0, {Lz})!")

                exit(1)

    print("All atom coordinates confirmed to be within box
bounds.")



    # --- Write LAMMPS Data File ---

    output_filename = "perforated_lamellar_50P.data"

    print(f"Writing {actual_total_molecules} molecules to
{output_filename}...")


    header = f"""LAMMPS data file: Hexagonal Lamellar Stack
(Centered & Wrapped)


{actual_total_molecules * 2} atoms

{actual_total_molecules} bonds

2 atom types

1 bond types
```

```
0.0 {Lx:.4f} xlo xhi

0.0 {Ly:.4f} ylo yhi

0.0 {Lz:.4f} zlo zhi


Atoms # bond

"""


    try:

        with open(output_filename, "w") as f:

            f.write(header)

            f.write("\n")

            for i, (atom1, atom2) in
enumerate(downsampled_molecules):

                molecule_id = i + 1

                atom1_id = 2 * i + 1

                atom2_id = 2 * i + 2

                f.write(f"{atom1_id} {molecule_id} 1
{atom1[0]:.4f} {atom1[1]:.4f} {atom1[2]:.4f}\n")

                f.write(f"{atom2_id} {molecule_id} 2
{atom2[0]:.4f} {atom2[1]:.4f} {atom2[2]:.4f}\n")


            f.write("\nBonds\n\n")

            for i in range(actual_total_molecules):

                bond_id = i + 1

                atom1_id = 2 * i + 1

                atom2_id = 2 * i + 2

                f.write(f"{bond_id} 1 {atom1_id} {atom2_id}\n")
```

```
        print(f"Successfully wrote {actual_total_molecules}
molecules to {output_filename}.")
    except IOError as e:
        print(f"Error writing LAMMPS data file: {e}")
        return

    print("--- Simulation Setup Complete ---")

# --- Perforation Coordinates ---
perforations = [
    np.array([5.0, 5.0]),
    np.array([5.0, 12.5]),
    np.array([5.0, 20.0]),
    np.array([12.5, 8.75]),
    np.array([12.5, 16.25]),
    np.array([20.0, 5.0]),
    np.array([20.0, 12.5]),
    np.array([20.0, 20.0])
]

if __name__ == "__main__":
    main()
```

D.2 LAMMPS Template Script for Bisection

in.template

```
# 2-sphere system: NPT equilibration to test z-axis
commensurability
```

```
# ------------------------------------------------------------
--------------

#  This script continues directly from the Stage-4 output

#  written by your NVT run (restart_dumbbell_lamellar.data).

#  It lets only the box length in z fluctuate so the lamellar

#  repeat distance can settle on a commensurate value.

# ------------------------------------------------------------
--------------


# -------------------- Variable inputs (same as NVT file) ------
--------------

variable        sigma1 equal 1.0

variable        Delta  equal "0.2821*v_sigma1"

variable        Rg1    equal "0.5*v_sigma1"

variable        Rg2    equal 1.0

variable        U012   equal 20

variable        U022   equal 2

variable        Ucut   equal 0.001

variable        eps11  equal 1.54177


# -------------------- Derived quantities ---------------------
-------------

variable       sigmah22   equal "((v_Rg2^2 + v_Rg2^2)/3)^(1/2)"

variable       sigmah12   equal "((v_Rg1^2 + v_Rg2^2)/3)^(1/2)"

variable       H22        equal
"((2*PI)^(1/2))*v_sigmah22*v_U022"

variable       H12        equal
"((2*PI)^(1/2))*v_sigmah12*v_U012"

variable       bondlength equal "0.5*v_sigma1 + v_Rg2"
```

```
variable       rgausscut  equal "(-
2*v_sigmah22^2*ln((v_Ucut*v_sigmah22*(2*PI)^(1/2))/v_H22))^(1/2)
"


# -------------------- Mass for bead-spring map ----------------
-------------

if "${Rg2} == 0.69" then "variable mass2 equal 0.2026"

if "${Rg2} == 1.0"  then "variable mass2 equal 0.4443"

if "${Rg2} == 1.45" then "variable mass2 equal 1.0557"

if "${Rg2} == 1.9"  then "variable mass2 equal 1.5817"

if "${Rg2} == 2.8"  then "variable mass2 equal 3.4584"


# -------------------- Initialise simulation -------------------
-------------

clear

log log.npt_###TARGET_PRESSURE### append


units           lj

dimension       3

boundary        p p p

atom_style      bond

bond_style      harmonic

# from built molecular file

read_data       perforated_lamellar_50P.data


# -------------------- Force-field -----------------------------
-------------

mass 1 1.0

mass 2 ${mass2}
```

```
velocity        all create 1.0 54654 mom yes # created some
velocity randomly instead of using 0

pair_style  hybrid gauss/cut 4.0 lj/expand $(2^(1/6))

pair_modify pair lj/expand shift yes

pair_coeff 1 1 lj/expand ${eps11} ${sigma1} ${Delta}
$((2^(1/6))*v_sigma1)

pair_coeff 2 2 gauss/cut ${H22} 0.0 ${sigmah22} ${rgausscut}

pair_coeff 1 2 gauss/cut ${H12} 0.0 ${sigmah12} ${rgausscut}


bond_coeff 1 500 ${bondlength}

special_bonds lj/coul 0 1 1


neighbor     0.4 multi

neigh_modify check yes delay 0

comm_modify  cutoff 0


# -------------------- Time-integration & barostat --------------
-------------

variable tLJ equal "sqrt(v_sigma1^2/v_eps11)"   # characteristic
time


timestep $(0.001*v_tLJ)

reset_timestep 0


# One fix does everything (integration + Nose-Hoover thermostat
+ barostat)

#fix baro all npt temp 1.0 1.0 $(0.1*v_tLJ) z 1.0 1.0
$(1.0*v_tLJ)
```

```
# --- If you prefer x and y to relax together with z, comment
the line above

 fix baro all npt temp 1.0 1.0 $(0.1*v_tLJ) aniso
###TARGET_PRESSURE### ###TARGET_PRESSURE### $(1.0*v_tLJ) couple
xy


thermo_style  custom step temp press density pxx pyy pzz lx ly
lz vol pe ke epair emol atoms

thermo        10000


dump          1 all custom 100000 equilnpt.xyz   id type mol x
y  z

dump          2 all custom 100000 equilnptuw.xyz id type mol xu
yu zu


run           500000


write_data    restart_dumbbell_npt.data nocoeff


# ---------------------------------------------------------------
-------------

# End of file
```

D.2 Python Script for Iterating Pressure

Density_bisect_working.py

```
#!/usr/bin/env python3

# density_bisect_slurm.py

import subprocess, pathlib, time, shutil

from statistics import mean
```

```
# ------------------------------------------------------------
-----------

# 1. USER SETTINGS

# ------------------------------------------------------------
-----------

# modify this based on the density desired. Can be found from
Round 5 NVT.

target_density = 0.32441338

tolerance      = 0.01 * target_density            # 1 % window

# change this based on the NVT pressure.

P_low, P_high  = 3.9375, 3.9375                         #
Initial pressure bracket

max_iter       = 20


template_in    = pathlib.Path("template.in")      # Contains
###TARGET_PRESSURE###

# Assumes template.in exists and has the placeholder.

# Your template.in should now have a line like:

# log log.npt_###TARGET_PRESSURE###

if not template_in.exists():

    # Create a dummy template.in if it doesn't exist.

    template_in.write_text("log
log.npt_###TARGET_PRESSURE###\nfix 1 all npt temp 300.0 300.0
100.0 iso ###TARGET_PRESSURE### ###TARGET_PRESSURE###
1000.0\nrun 10000\n")



lmp_exec       = "/expanse/lustre/projects/itm108/kdai/lammps-
7Aug19/build/lmp" # Full path if not in $PATH

poll_seconds   = 60                               # How often to
check the queue
```

```
# ------------------------------------------------------------
-----------



# ------------------------------------------------------------
-----------
# 2. HELPERS
# ------------------------------------------------------------
-----------
def write_lammps_input(P: float) -> pathlib.Path:
    """Return path of newly written LAMMPS input file."""
    Pstr = f"{P:.4f}"
    content =
template_in.read_text().replace("###TARGET_PRESSURE###", Pstr)
    name = pathlib.Path(f"in_P{Pstr}.lmp")
    name.write_text(content)
    return name


#
____________________________________________________________
_____________
def write_job_script(input_fn: pathlib.Path) -> pathlib.Path:
    """
    Writes a SLURM job script that runs LAMMPS.
    """
    script = f"""#!/bin/bash
#SBATCH --nodes=1
#SBATCH --ntasks=4
#SBATCH --cpus-per-task=2
```

```
#SBATCH --mem-per-cpu=2G
#SBATCH --constraint="lustre"
#SBATCH --time=5:00:00
#SBATCH --partition=shared
#SBATCH --account=TG-CHM240092
#SBATCH --output=slurm-%j.out

module load cpu/0.17.3b
module load gcc/10.2.0
module load openmpi/4.1.1

export OMP_NUM_THREADS=$SLURM_CPUS_PER_TASK

# Run LAMMPS. The log file name is controlled by the 'log'
command
# inside the LAMMPS input script itself.
mpirun -n 4 {lmp_exec} -in {input_fn}
"""

    job_fn = pathlib.Path(f"job_{input_fn.stem}.sh")
    job_fn.write_text(script)
    job_fn.chmod(0o755)
    return job_fn
#
_____
_____

def submit_and_wait(job_fn: pathlib.Path, job_name: str):
```

```
    """
    Submits a SLURM job and waits for it to complete.
    """
    # --- Submit the job ---
    try:
        completed = subprocess.run(
            ["sbatch", str(job_fn)],
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE,
            universal_newlines=True,  # Use
universal_newlines=True for Python 3.6
            check=True
        )
    except (subprocess.CalledProcessError, FileNotFoundError) as
e:
        print(f"sbatch command failed: {e}")
        if hasattr(e, 'stderr'):
            print(f"Stderr: {e.stderr}")
        return


    # stdout is something like: "Submitted batch job 40175802\n"
    try:
        jobid = int(completed.stdout.strip().split()[-1])
        print(f"  > Submitted job {jobid} for {job_name}.
Waiting...")
    except (ValueError, IndexError):
        print(f"Error: Could not parse job ID from sbatch
output: {completed.stdout}")
        return
```

```
    # --- Wait for the job to finish by polling the queue ---
    while True:
        try:
            # Check the queue for the job ID
            queue_check = subprocess.run(
                ["squeue", "-j", str(jobid)],
                stdout=subprocess.PIPE,
                stderr=subprocess.PIPE,
                universal_newlines=True # Use
universal_newlines=True for Python 3.6
            )
            # If the job ID is not in the output, it's done
            if str(jobid) not in queue_check.stdout:
                print(f"  > Job {jobid} finished.")
                break
        except FileNotFoundError:
            print("Error: 'squeue' command not found. Cannot
check job status.")
            print("Assuming job finished after a delay.")
            time.sleep(poll_seconds * 2) # Wait for a fixed time
as a fallback
            break

        # Wait before checking again
        time.sleep(poll_seconds)
```

```
def density_stats(log_file: pathlib.Path):
    """Return (rho_last, rho_mean) from last run block in a
LAMMPS log."""
    idx, this_run, last_run = None, [], []
    with log_file.open() as fh:
        for line in fh:
            line = line.strip()
            if line.startswith("Step"):
                if this_run:
                    last_run = this_run
                this_run = [] # Start a new block
                try:
                    cols = line.split()
                    idx = cols.index("Density")
                except ValueError:
                    idx = None # Density not in this header
                continue

            if idx is not None and line:
                try:
                    cols = line.split()
                    # Ensure the line looks like a data line
(starts with a number)
                    int(cols[0])
                    density_val = float(cols[idx])
                    this_run.append(density_val)
                except (ValueError, IndexError):
                    # Skip lines that are not valid data rows
```

```
                    continue

    # The final run block might be in 'this_run' if it's the
last thing in the file
    if this_run:
        last_run = this_run

    if not last_run:
        raise RuntimeError(f"No valid density data found in
{log_file}")

    return last_run[-1], mean(last_run)
# ------------------------------------------------------------
-----------



# ------------------------------------------------------------
-----------
# 3.  MAIN LOOP
# ------------------------------------------------------------
-----------
print("Starting bisection search for target density...")
print(f"Target Density: {target_density:.6f} (tolerance:
{tolerance:.6f})")
print(f"Initial Pressure Bracket: [{P_low:.4f}, {P_high:.4f}]")
print("-" * 50)

for it in range(max_iter):
    P_mid = 0.5 * (P_low + P_high)
```

```
    Pstr  = f"{P_mid:.4f}"
    print(f"[{it+1:02d}/{max_iter}] Testing P = {Pstr}")

    # --- Prepare input & job files ---
    inp = write_lammps_input(P_mid)
    job = write_job_script(inp)

    # --- Submit job and wait for it to complete ---
    submit_and_wait(job, job_name=f"P={Pstr}")

    # --- Define the expected log file name and check for it ---
    # This name must match the name generated by the 'log'
command in your LAMMPS script
    log_to_parse = pathlib.Path(f"log.npt_{Pstr}")

    # --- Analyze density from the log file ---
    if not log_to_parse.exists() or log_to_parse.stat().st_size
== 0:
        print(f"✗ Warning: Log file '{log_to_parse}' not found
or is empty. SLURM job may have failed.")
        print("  > Assuming pressure was too high and adjusting
bracket.")
        P_high = P_mid # A common failure is unstable dynamics
from high P
        continue

    try:
        rho_last, rho_mean = density_stats(log_to_parse)
        print(f"  > Result: ρ_mean = {rho_mean:.6f}")
```

```
    except RuntimeError as e:
        print(f"✗ Error analyzing log file '{log_to_parse}':
{e}")
        print("  > Assuming pressure was too low or run failed.
Adjusting bracket.")
        P_low = P_mid
        continue

    # --- Bisection update ---
    if abs(rho_mean - target_density) < tolerance:
        print("-" * 50)
        print(f"✓ Convergence successful on iteration {it+1}!")
        print(f"  Final Pressure: {Pstr}")
        print(f"  Final Density:  {rho_mean:.6f}")
        break
    elif rho_mean > target_density:
        print("  > Density is too high. Setting P_high = P_mid")
        P_high = P_mid
    else: # rho_mean < target_density
        print("  > Density is too low. Setting P_low = P_mid")
        P_low  = P_mid

    print("-" * 50)

else: # This 'else' belongs to the 'for' loop
    print("✗ Reached max_iter without converging to the target
density.")
```

```
# ------------------------------------------------------------
-----------
```

D.3 LAMMPS Production Script

in.WCAcollgaussNPT

```
########################################
###       GCBC Dumbbell Model       ###
########################################

# Base units
variable        temp index 1.0
variable        dt equal 0.005
variable        sigma1 equal 1.0
variable        Delta equal "0.2821*v_sigma1"
variable        Rg1 equal "0.5*v_sigma1"
variable        Rg2 equal 0.69
variable        U012 equal 100
variable        U022 equal 2
variable        Ucut equal 0.001
variable        eps11 equal 1.54177

# Calculated quantities
variable        Tdamp equal "(v_dt*1e2)"
variable        sigmah22 equal "((v_Rg2^2 + v_Rg2^2)/3)^(1/2)"
variable        sigmah12 equal "((v_Rg1^2 + v_Rg2^2)/3)^(1/2)"
variable        H22 equal "((2*PI)^(1/2))*v_sigmah22*v_U022"
variable        H12 equal "((2*PI)^(1/2))*v_sigmah12*v_U012"
variable        bondlength equal "0.5*v_sigma1 + v_Rg2"
```

```
variable        rgausscut equal "(-
2*v_sigmah22^2*ln((v_Ucut*v_sigmah22*(2*PI)^(1/2))/v_H22))^(1/2)
"



# Masses
if "${Rg2} == 0.69" then &
     "variable mass2 equal 0.2026" &
elif "${Rg2} == 0.79" &
     "variable mass2 equal 0.2666" &
elif "${Rg2} == 0.89" &
     "variable mass2 equal 0.3483" &
elif "${Rg2} == 1.0" &
     "variable mass2 equal 0.4443" &
elif "${Rg2} == 1.11" &
     "variable mass2 equal 0.6469" &
elif "${Rg2} == 1.23" &
     "variable mass2 equal 0.6718" &
elif "${Rg2} == 1.45" &
     "variable mass2 equal 1.0557" &
elif "${Rg2} == 1.9" &
     "variable mass2 equal 1.5817" &
elif "${Rg2} == 2.8" &
     "variable mass2 equal 3.4584"

# Units and force field
units           lj
atom_style      bond
```

```
pair_style      hybrid gauss/cut 4.0 lj/expand $(2^(1/6))
pair_modify     pair lj/expand shift yes
bond_style      harmonic
special_bonds   lj/coul 0 1 1

# System setup
boundary        p p p
read_data       restart_dumbbell_npt.data nocoeff
group           hard type 1
group           soft type 2

pair_coeff      1 1 lj/expand ${eps11} ${sigma1} ${Delta}
$((2^(1/6))*v_sigma1)
pair_coeff      2 2 gauss/cut ${H22} 0.0 ${sigmah22}
${rgausscut}
pair_coeff      1 2 gauss/cut ${H12} 0.0 ${sigmah12}
${rgausscut}
bond_coeff      1 500 ${bondlength}

mass 1 1.0
mass 2 ${mass2}

# Neighbor settings
neighbor        0.4 multi
neigh_modify    every 1 delay 0 check yes
velocity        all create ${temp} 54654 mom yes
timestep        ${dt}
```

```
# === NPT FIX ===
 fix baro all npt temp 1.0 1.0 $(v_dt*1e2) aniso 3.9375 3.9375
$(v_dt*1e3) couple xy


# Output folders
shell "mkdir dump_GCBC"
shell "rm dump_GCBC/*"
shell "mkdir dump_GCBCuw"
shell "rm dump_GCBCuw/*"

# Averaging and observables
if "${dt} == 0.005" then &
     "variable Nevery equal 10000" &
     "variable Nfreq equal 50000" &
     "variable Ndump equal 100000" &
     "variable Nrun equal 2000000" &
elif "${dt} == 0.0005" &
     "variable Nevery equal 100000" &
     "variable Nfreq equal 500000" &
     "variable Ndump equal 1000000" &
     "variable Nrun equal 20000000"
variable        Nrepeat equal  5
variable        PotentialEnergy equal pe
variable        KineticEnergy equal ke
variable        MolecularEnergy equal emol
variable        PairwiseEnergy equal epair
variable        Pressure equal press
variable        Temperature equal temp
```

```
variable        Density equal density

fix             3 all ave/time ${Nevery} ${Nrepeat} ${Nfreq} &

                v_PotentialEnergy v_KineticEnergy
v_MolecularEnergy v_PairwiseEnergy v_Temperature v_Pressure
v_Density &

                file TimeAvgQtys.out format %.4g


# RDF

compute myRDF all rdf 50

fix 1 all ave/time ${Nevery} ${Nrepeat} ${Nfreq} c_myRDF[*] file
GCBCs.rdf mode vector


# Thermo output

thermo          ${Nevery}

thermo_style    custom step temp press pxx pyy pzz vol density
pe ke epair emol atoms

thermo_modify   norm no


# Dump coordinates

dump            x1 all custom ${Ndump} dump_GCBC/dump*.xyz id
type mol x y z

dump            x2 all custom ${Ndump} dump_GCBCuw/dump*.xyz id
type mol xu yu zu


# Run MD

reset_timestep  0

run             ${Nrun}


# Save final data
```

```
write_data      restart.txt nocoeff
```

D.4 LAMMPS Replica Exchange Script

in.temperNPT

```
#GCBCs: Hard-Soft Coarse Grained Model

#This is a template that should contain most of the relevant
syntax

#Always start from here when coding.


#------------------Initialize Simulation--------------------

clear


# Base units

# 1 epsilon = 1 kBT

# 1 sigma = 3.12 nm (diameter of hard sphere)

# 1 T = 298 K (room temperature)


# Variable inputs

variable       temp index 1.0 # we will use temperature of 1 for
25 C

variable       dt equal 0.005 # size of timestep! damping factor
Tdamp will depend on this

variable       sigma1 equal 1.0 # diameter of hard sphere, where
1 sigma = dpro = 3.12 nm

variable       Delta equal "0.2821*v_sigma1" # Delta = 0.88 nm

variable       Rg1 equal "0.5*v_sigma1" # radius of gyration of
hard sphere without its hydration shell (can also do with
hydration shell)

variable       Rg2 equal 1.0 # radius of gyration of soft sphere
```

```
variable       U012 equal 50 # U(r=0) for hard-soft potential
[kbT]

variable       U022 equal 2 # U(r=0) for soft-soft potential
[kbT]

variable       Ucut equal 0.0001 # energy at cutoff [kbT] for
Gaussian interactions

variable       eps11 equal 1.54177 # epsilon for hard-hard
potential [kbT] from 2nd virial coefficient

variable       fswap equal 50 # I need to understand what fswap
is doing.

variable       Ndump equal 500000

variable       Nrun equal 500000


# --- NEW VARIABLES FOR NPT ENSEMBLE ---

# Target pressure for NPT ensemble (in LJ units). Adjust as
needed.

variable        pressure_val equal 8.5000

# Damping factor for the barostat (pressure control). Typically
1000*dt.

variable        Pdamp equal "(v_dt*1e3)"


# Variable calculations

variable       t world 1.0 1.01 1.0201 1.0303010000000001
1.04060401 1.0510100501000001 1.0615201506010001
1.07213535210701 1.0828567056280802 1.0936852726843609
1.1046221254112045 1.1156683466653166 1.1268250301319698
1.1380932804332895 1.1494742132376226 1.1609689553699987
1.1725786449236988 1.1843044313729356 1.196147475686665
1.2081089504435316 1.220190039947967 1.2323919403474468
1.2447158597509211 1.2571630183484304 1.2697346485319148
1.2824319950172338 1.2952563149674063 1.3082088781170804
1.321290966898251 1.3345038765672337 1.347848915332906
1.3613274044862351 1.3749406785310974 1.3886900853164084
```

```
1.4025769861695725 1.4166027560312682 1.430768783591581
1.4450764714274968 1.4595272361417717 1.4741225085031895
1.4888637335882213 1.5037523709241036 1.5187898946333447
1.5339777935796781 1.549317571515475 1.5648107472306296

variable       Tdamp equal "(v_dt*1e2)" # damping factor of
Langevin thermostat is typically 2 orders of magnitude larger
than the size of the timestep

variable       sigmah22 equal "((v_Rg2^2 + v_Rg2^2)/3)^(1/2)" #
sigmah = sqrt((Rgi^2 + Rgj^2)/3), a term in the Gaussian
potential

variable       sigmah12 equal "((v_Rg1^2 + v_Rg2^2)/3)^(1/2)"

variable       H22 equal "((2*PI)^(1/2))*v_sigmah22*v_U022" # H
term in Gaussian potential

variable       H12 equal "((2*PI)^(1/2))*v_sigmah12*v_U012" # H
term in Gaussian potential

variable       bondlength equal "0.5*v_sigma1 + v_Rg2" # bond
length

variable       rgausscut equal "(-
2*v_sigmah22^2*ln((v_Ucut*v_sigmah22*(2*PI)^(1/2))/v_H22))^(1/2)
"


# Mass Conditionals for Soft Sphere

if "${Rg2} == 0.69" then &

     "variable mass2 equal 0.2026" &

elif "${Rg2} == 0.79" &

     "variable mass2 equal 0.2666" &

elif "${Rg2} == 0.89" &

     "variable mass2 equal 0.3483" &

elif "${Rg2} == 1.0" &

     "variable mass2 equal 0.4443" &

elif "${Rg2} == 1.11" &
```

```
     "variable mass2 equal 0.6469" &
elif "${Rg2} == 1.23" &
     "variable mass2 equal 0.6718" &
elif "${Rg2} == 1.45" &
     "variable mass2 equal 1.0557" &
elif "${Rg2} == 1.9" &
     "variable mass2 equal 1.5817" &
elif "${Rg2} == 2.8" &
     "variable mass2 equal 3.4584"

#Create a log file
log log.temper append

# Interaction-potential models
units           lj
atom_style      bond
pair_style      hybrid gauss/cut 4.0 lj/expand $(2^(1/6))
pair_modify     pair lj/expand shift yes
bond_style      harmonic
special_bonds   lj/coul 0 1 1

read_data restart.txt nocoeff

#---------------Define Interatomic Potential----------------
# Force-fields
# For lj/expand (colloidal WCA/LJ): type1 type2 lj/expand
epsilon sigma delta cutoff [!!! this cutoff does not include
delta!]
```

```
# For lj/cut (WCA/LJ): type1 type2 lj/cut epsilon sigma cutoff1
(cutoff2)

# For gauss/cut (Gaussian): H [energy*distance] r_mh [distance]
sigma_h [distance] cutoff [distance]

pair_coeff 1 1 lj/expand ${eps11} ${sigma1} ${Delta}
$((2^(1/6))*v_sigma1)

pair_coeff 2 2 gauss/cut ${H22} 0.0 ${sigmah22} ${rgausscut}

pair_coeff 1 2 gauss/cut ${H12} 0.0 ${sigmah12} ${rgausscut}

bond_coeff 1 500 ${bondlength}


mass 1 1.0

mass 2 ${mass2}


# MD settings - Equilibration

neighbor        0.4 multi

neigh_modify    every 1 delay 0 check yes

velocity        all create ${temp} 54654 mom yes

timestep        ${dt}

run_style verlet #verlet integration, standard


# --- NPT ENSEMBLE FIX ---

# Replaces 'fix nve' and 'fix langevin'

# Uses Nose-Hoover thermostat and barostat for constant NPT.

# The temperature for each replica is set by the 'temper'
command's world variable '$t'.

# The pressure is set to 'pressure_val' isotropically.

fix ensemble_npt all npt temp $t $t ${Tdamp} aniso
${pressure_val} ${pressure_val} ${Pdamp} couple xy
```

```
shell "rm dump_temper/*"
shell "mkdir dump_temper"



thermo         ${fswap}
thermo_style    custom step temp press density pe epair atoms
thermo_modify   norm no


#Dump File syntax:
#dump ID group-ID style N file args
variable       r world 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37
38 39 40 41 42 43 44 45
dump           x1 all xyz ${Ndump} dump_temper/dump*.$r.xyz


# --- UPDATED TEMPER COMMAND ---
# The 'temper' command now uses the 'ensemble_npt' fix for
temperature and pressure control.
temper/npt          ${Nrun} ${fswap} $t ensemble_npt 1206 1825
${pressure_val}


#Create desired restart files here
write_data     restart_dumbbell.$r.data
```

D.5 MATLAB Code for Structure Factor Calculations

```
% Author: Zachary M. Sherman
% Modified by: Helen Yao
%
```

```
% Computes and plots the static structure factor for a snapshot
of
% particles.
%


close all
clc
clear


%% Inputs
% %% Input Parameters
t0 = 10000000;
tN = 20000000;
time_vector = t0:1000000:tN;
Ngrid = 256; % number of grid points to divide box into
unwrapped = 0; %if unwrapped, turn on to 1
com = 0; % 1: count all particles to be the same, 0: mark
protein and polymer differently (use 0)


% Folder information
stage = 'Production';
U12 = 'U0_100kT';
Rg12 = 'Rg_05_28';
conc = 5; % change this
alg = 'FFT'; % FFT or GRD (when was time avg calculated; FFT
includes "debye-waller factor" from flucs)
hires = 'N'; % save hi-res figures or not
savefiles = 'N'; % save the SF data files?
```

```
% Base colors
Dk = [12, 44, 132];
Mid = [65,182,196];
Lt = [237, 248, 177];
MyColors = Color_Gradient(100,Dk,Mid,Lt);

%% Where to save stuff
fullpath = fullfile(pwd,stage,'StructureFactors',U12,Rg12);
if ~exist(fullpath, 'dir')
    mkdir(fullpath);
end

hirespath =
fullfile(pwd,stage,'StructureFactors','HiRes',U12,Rg12);
if ~exist(hirespath, 'dir')
    mkdir(hirespath);
end

%% Set box size (Round 5 NPT)
box_dimension = [-6.7717130536954073e-01 5.4677171305372170e+01
-6.7717130536954073e-01 5.4677171305372170e+01
1.2996210607562801e+00 5.2700378939238554e+01];
box = [(box_dimension(1,2) - box_dimension(1,1)),
    (box_dimension(2,2) - box_dimension(2,1)),
    (box_dimension(3,2) - box_dimension(3,1)),]';
%% Extract the structure factor, along with all wave vectors
% [Smag,q,qx,qy,qz,S] = StructureFactor1D(box,Ngrid,alg);
```

```
if strcmp(alg,'FFT')

    [S,qx,qy,qz,Smag,q] =
timeavgfft(time_vector,box,stage,U12,Rg12,conc,unwrapped,com,Ngr
id,savefiles);

elseif strcmp(alg,'GRD')

    [S,qx,qy,qz,Smag,q] =
timeavggrid(time_vector,box,stage,U12,Rg12,conc,unwrapped,com,Ng
rid,savefiles);

end


%% Plot spherically averaged S(q)

figure(1)

plot(q(2:end),Smag(2:end),'LineWidth',2,'Color','k')

xlabel('$q$ $[\sigma^{-1}]$','Interpreter','latex')

ylabel('$S(q)$','Interpreter','latex')

xlim([0,20])

ylim([0,round(max(Smag(2:end))+1)])

% set(gca,'LineWidth',2,'FontSize',16,'Box','on');

set(gca,'LineWidth',2,'FontUnits','normalized','FontSize',0.07,.
..


'LabelFontSizeMultiplier',1,'FontWeight','bold','PlotBoxAspectRa
tio',[1,1,1],'Box','on',...

    'TickLength',[0.03,
0.03],'TickLabelInterpreter','latex','xscale','log','yscale','lo
g')

set(gcf,'Color','w','units','pixels','outerposition',[10 200 500
500]);

saveas(1,fullfile(fullpath,strcat('Sq_',alg,'_mag_',num2str(conc
),'P.png')));

if strcmp(hires,'y')
```

```
export_fig(fullfile(hirespath,strcat('Sq_',alg,'_mag_',num2str(c
onc),'P')),'-r600','-png');

end


%% Plot the qz = 0 2D diffraction pattern

% Isolate the qz = 0 plane

ind = find(qz == 0);

qx_0 = qx(ind);

qy_0 = qy(ind);

S_0 = S(ind);


% The previous calculations return vectors.  Reshape the vectors
into

% grids.

Ngrid = size(qx,1);

qx_0 = reshape(qx_0,Ngrid,Ngrid);

qy_0 = reshape(qy_0,Ngrid,Ngrid);

S_0 = reshape(S_0,Ngrid,Ngrid);


% Filter out values larger than some tolerance

tol = 15;

S_0(S_0 >= tol) = nan;


% Plot S(q) in the qz = 0 plane

figure(2)

% colormap hot

colormap(MyColors)
```

```
h = pcolor(qx_0,qy_0,S_0);
h.EdgeColor = 'none';
colorbar('TickLabelInterpreter','latex')
xlabel('$q_x$ $[\sigma^{-1}]$','Interpreter','latex')
ylabel('$q_y$ $[\sigma^{-1}]$','Interpreter','latex')
title('$q_z = 0$ $\sigma^{-1}$','Interpreter','latex')
xlim([-20 20]);
ylim([-20 20]);
set(gca,'LineWidth',2,'FontUnits','normalized','FontSize',0.05,...

'LabelFontSizeMultiplier',1.5,'FontWeight','bold','PlotBoxAspectRatio',[1,1,1],'Box','on',...
    'TickLabelInterpreter','latex')
set(gcf,'Color','w','units','pixels','outerposition',[500 350 500 500]);
saveas(2,fullfile(fullpath,strcat('Sq_',alg,'_z0_',num2str(conc),'P.png')));
if strcmp(hires,'y')

export_fig(fullfile(hirespath,strcat('Sq_',alg,'_z0_',num2str(conc),'P')),'-r600','-png');
end


%% Plot the qy = 0 2D diffraction pattern
% Isolate the qy = 0 plane
ind = find(qy == 0);
qx_0 = qx(ind);
qz_0 = qz(ind);
S_0 = S(ind);
```

```
% The previous calculations return vectors.  Reshape the vectors
into
% grids.
qx_0 = reshape(qx_0,Ngrid,Ngrid);
qz_0 = reshape(qz_0,Ngrid,Ngrid);
S_0 = reshape(S_0,Ngrid,Ngrid);

% Filter out values larger than some tolerance
tol = 15;
S_0(S_0 >= tol) = nan;

% Plot S(q) in the qy = 0 plane
figure(3)
% colormap hot
colormap(MyColors)
h = pcolor(qx_0,qz_0,S_0);
h.EdgeColor = 'none';
colorbar('TickLabelInterpreter','latex')
xlabel('$q_x$ $[\sigma^{-1}]$','Interpreter','latex')
ylabel('$q_z$ $[\sigma^{-1}]$','Interpreter','latex')
xlim([-20 20]);
ylim([-20 20]);
title('$q_y = 0$ $\sigma^{-1}$','Interpreter','latex')
set(gca,'LineWidth',2,'FontUnits','normalized','FontSize',0.05,.
..

'LabelFontSizeMultiplier',1.5,'FontWeight','bold','PlotBoxAspect
Ratio',[1,1,1],'Box','on',...
```

```
    'TickLabelInterpreter','latex')
set(gcf,'Color','w','units','pixels','outerposition',[1000 350
500 500]);
saveas(3,fullfile(fullpath,strcat('Sq_',alg,'_y0_',num2str(conc)
,'P.png')));
if strcmp(hires,'y')

export_fig(fullfile(hirespath,strcat('Sq_',alg,'_y0_',num2str(co
nc),'P')),'-r600','-png');
end


%% Plot the qx = 0 2D diffraction pattern
% Isolate the qx = 0 plane
ind = find(qx == 0);
qy_0 = qy(ind);
qz_0 = qz(ind);
S_0 = S(ind);

% The previous calculations return vectors.  Reshape the vectors
into
% grids.
qy_0 = reshape(qy_0,Ngrid,Ngrid);
qz_0 = reshape(qz_0,Ngrid,Ngrid);
S_0 = reshape(S_0,Ngrid,Ngrid);

% Filter out values larger than some tolerance
tol = 15;
S_0(S_0 >= tol) = nan;
```

```
% Plot S(q) in the qz = 0 plane

figure(4)

% colormap pink

colormap(MyColors)

h = pcolor(qy_0,qz_0,S_0);

h.EdgeColor = 'none';

colorbar('TickLabelInterpreter','latex')

xlim([-20 20]);

ylim([-20 20]);

xlabel('$q_y$ $[\sigma^{-1}]$','Interpreter','latex')

ylabel('$q_z$ $[\sigma^{-1}]$','Interpreter','latex')

title('$q_x = 0$ $\sigma^{-1}$','Interpreter','latex')

set(gca,'LineWidth',2,'FontUnits','normalized','FontSize',0.05,.
..


'LabelFontSizeMultiplier',1.5,'FontWeight','bold','PlotBoxAspect
Ratio',[1,1,1],'Box','on',...

    'TickLabelInterpreter','latex')

set(gcf,'Color','w','units','pixels','outerposition',[700 10 500
500]);

saveas(4,fullfile(fullpath,strcat('Sq_',alg,'_x0_',num2str(conc)
,'P.png')));

if strcmp(hires,'y')


export_fig(fullfile(hirespath,strcat('Sq_',alg,'_x0_',num2str(co
nc),'P')),'-r600','-png');

end
```

D.6 Python Code for MSD Analysis

D.6.1 Msd_com_multi_origin_new.py

```
import argparse, glob, re, itertools, sys
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

def sorted_files(pattern):
    def step(name):
        m = re.search(r"(\d+)", name)
        return int(m.group(1)) if m else 0
    return sorted(glob.glob(pattern), key=step)

def parse_frame(f):
    if (line := f.readline()) == "":
        return None
    if not line.startswith("ITEM: TIMESTEP"):
        raise ValueError("Expected ITEM: TIMESTEP")
    ts = int(f.readline())
    f.readline(); natoms = int(f.readline().strip())
    f.readline(); [f.readline() for _ in range(3)]   # BOX lines
    cols_line = f.readline().strip()
    if "mol" not in cols_line.split():
        raise ValueError("Dump must include 'mol' column.")
    # figure out column order
    cols = cols_line.split()[2:]   # drop 'ITEM: ATOMS'
    data = np.loadtxt(itertools.islice(f, natoms))
    col_idx = {name: i for i, name in enumerate(cols)}
    coords = data[:, [col_idx.get(c) for c in ('xu','yu','zu')
if c in col_idx]]
```

```
    if coords.size == 0:
        coords = data[:, [col_idx[c] for c in ('x','y','z')]]
    mol_ids = data[:, col_idx['mol']].astype(int)
    types = data[:, col_idx['type']].astype(int)
    return ts, mol_ids, types, coords

def read_trajectory(input_path=None, pattern=None):
    frames = []
    if input_path:
        with open(input_path) as f:
            while (fr := parse_frame(f)) is not None:
                frames.append(fr)
    else:
        for fname in sorted_files(pattern):
            with open(fname) as f:
                fr = parse_frame(f)
                if fr is None:
                    raise ValueError(f"{fname} missing data.")
                frames.append(fr)
    return frames

def compute_com(coords, mol_ids, types, masses):
    # returns array (Nmol, 3) COM positions
    unique = np.unique(mol_ids)
    com = np.zeros((unique.size, 3))
    for i, mid in enumerate(unique):
        mask = mol_ids == mid
```

```
        m = masses[types[mask]]
        weighted = (coords[mask] * m[:,None]).sum(axis=0)
        com[i] = weighted / m.sum()
    return com

def compute_msd_multi_origin(frames, masses):
    # Build 3-D array (F, Nmol, 3)
    com_list = []
    for _, mol_ids, types, coords in frames:
        com = compute_com(coords, mol_ids, types, masses)
        com_list.append(com)
    com_arr = np.array(com_list)                                    #
(F,N,3)
    F = com_arr.shape[0]
    msd = np.zeros(F)
    for lag in range(1, F):
        disp = com_arr[lag:] - com_arr[:-lag]                       #
(F-lag,N,3)
        sq = (disp**2).sum(axis=-1)
        msd[lag] = sq.mean()
    return msd

def main():
    p = argparse.ArgumentParser(description="MSD of molecular
COM (multi-origin)")
    g = p.add_mutually_exclusive_group(required=True)
    g.add_argument("--input", help="single dump file with many
frames")
```

```
    g.add_argument("--pattern", help="glob pattern for many dump
files")

    p.add_argument("--dt", type=float, default=1.0,
help="timestep spacing between frames")

    p.add_argument("--mass", type=float, nargs='+', help="masses
for atom types 1,2,3… (default 1.0)")

    args = p.parse_args()


    frames = read_trajectory(args.input, args.pattern)

    n_types_in_dump = int(max(frames[0][2]))                  #
types array is third element

    masses = np.ones(n_types_in_dump+1)

    if args.mass:

        for i,m in enumerate(args.mass, start=1):

            if i <= n_types_in_dump:

                masses[i] = m


    timesteps = np.array([ts for ts, *_ in frames], float)

    msd = compute_msd_multi_origin(frames, masses)


    lags = np.arange(len(msd))

    times = lags * args.dt


    # Fit a linear regression model to the log-log plot

    log_t   = np.log(times[1:])

    log_msd = np.log(msd[1:])

    slope, intercept, r_value, p_value, std_err =
stats.linregress(log_t, log_msd)

    r_squared = r_value**2
```

```
    # Print the slope and R^2 value
    print(f"Fitted Slope: {slope:.4f}")
    print(f"R^2 value: {r_squared:.4f}")

    # Save the slope and R^2 value to a file
    with open("msd_fit_parameters.txt", "w") as f:
        f.write(f"Fitted Slope: {slope:.4f}\n")
        f.write(f"R^2 value: {r_squared:.4f}\n")
    print("Wrote msd_fit_parameters.txt")

    # Plot the data and the fit
    plt.figure()
    plt.loglog(times[1:], msd[1:], marker='o', label='MSD data')
    plt.loglog(times[1:], np.exp(intercept) * times[1:]**slope,
linestyle='--', color='red', label=f'Fit: slope={slope:.2f},
$R^2$={r_squared:.2f}')
    plt.xlabel("Time lag Δt")
    plt.ylabel("MSD")
    plt.legend()
    plt.savefig("msd_com_multi.png", dpi=300)

    # Save the data
    np.savetxt("msd_com_multi.dat", np.column_stack([times,
msd]), header="time MSD_COM", comments="")

    print("Wrote msd_com_multi.dat and msd_com_multi.png")
```

```
if __name__ == "__main__":
    main()
```

D.6.2 run_msd.sh

```
#!/usr/bin/env bash

EXT="xyz"

DT=100000

MASS_ARGS="--mass 1.0 0.4443"

FILES=$(seq -f "dump%.0f.${EXT}" 0 ${DT} 4000000)

echo "Files to process:"

echo ${FILES}

python msd_com_multi_origin_new.py --pattern "dump*.xyz" --dt
"${DT}" ${MASS_ARGS}
```

D.7 Python Code for Cluster Analysis

Clustering_Analysis_working.py

```
import numpy as np

from sklearn.cluster import DBSCAN

from collections import defaultdict, Counter

import matplotlib.pyplot as plt

import os

import traceback


# --- Configuration Parameters (YOU MUST ADJUST THESE) ---

DUMP_FILE_PREFIX = "dump"

DUMP_FILE_START_INDEX = 700000 # this is lamellar starting this
frame

DUMP_FILE_END_INDEX = 2000000

DUMP_FILE_INCREMENT = 100000
```

```
# Hard sphere diameter (e.g., from your simulation units)
SIGMA = 1.0

# DBSCAN parameters for INITIAL CLUSTERING (t=0)
DBSCAN_EPS_INITIAL = SIGMA * 2.5 # This requires tuning.
DBSCAN_MIN_SAMPLES_INITIAL = 5 # Set this to your desired value

# --- NEW: DBSCAN parameter for subsequent frame propagation ---
# This may need to be different from the initial EPS. Often slightly smaller,
# as layers might become more defined over time. This requires tuning.
DBSCAN_EPS_PROPAGATE = SIGMA * 2.5



# --- Data Structures to Store Cluster Information ---
molecule_cluster_transitions = defaultdict(list)
current_cluster_centers = {}
cluster_lifespans = defaultdict(int)

# --- Helper Function for Minimum Image Convention Distance ---
def minimum_image_distance_squared(p1, p2, box_dimensions):
    delta = np.abs(p1 - p2)
    delta = np.where(delta > 0.5 * box_dimensions, delta - box_dimensions, delta)
    return np.sum(delta**2)

# --- Helper Function to check initial DBSCAN results (Complete version) ---
```

```
def check_initial_dbscan_results(labels,
num_molecules_processed):

    unique_labels = set(labels)

    num_clusters = len(unique_labels) - (1 if -1 in
unique_labels else 0)

    noise_points = list(labels).count(-1)

    clustered_points = num_molecules_processed - noise_points


    print("\n--- Initial DBSCAN Clustering Summary (t=0) ---")

    print(f"Total molecules processed in initial frame:
{num_molecules_processed}")

    print(f"Number of identified clusters: {num_clusters}")

    print(f"Number of noise points (-1 label): {noise_points}")

    print(f"Number of molecules assigned to clusters:
{clustered_points}")

    print(f"Percentage of molecules in clusters:
{clustered_points / num_molecules_processed * 100:.2f}%")


    if num_clusters == 0:

        print("\n* Recommendation: No clusters found.
`DBSCAN_EPS_INITIAL` is likely too small or
`DBSCAN_MIN_SAMPLES_INITIAL` is too high.")

    elif noise_points / num_molecules_processed > 0.5:

        print("\n* Recommendation: More than 50% of molecules
are noise.")

        print("  Consider increasing `DBSCAN_EPS_INITIAL` or
decreasing `DBSCAN_MIN_SAMPLES_INITIAL`.")

    elif num_clusters > num_molecules_processed / 2:

        print("\n* Recommendation: Many small clusters
detected.")
```

```
        print("  Consider increasing `DBSCAN_EPS_INITIAL` or
`DBSCAN_MIN_SAMPLES_INITIAL`.")
    else:
        print("\n* Initial DBSCAN results seem reasonable.
Proceeding with propagation.")

    print("-----------------------------------------\n")

# --- Function to parse LAMMPS dump file (Complete version) ---
def parse_lammps_dump_and_get_coms(filepath):
    molecule_atoms_data = defaultdict(list)
    box_dimensions = np.zeros(3)
    atom_coords = defaultdict(list)

    with open(filepath, 'r') as f:
        line = f.readline()
        while not line.startswith('ITEM: NUMBER OF ATOMS'):
            line = f.readline()
        num_atoms = int(f.readline().strip())

        line = f.readline()
        for i in range(3):
            bounds = f.readline().split()
            low = float(bounds[0])
            high = float(bounds[1])
            box_dimensions[i] = high - low

        atom_header_line = f.readline().strip()
```

```
        if not atom_header_line.startswith('ITEM: ATOMS'):
            raise ValueError(f"Expected 'ITEM: ATOMS' line but
got: {atom_header_line}")

        column_names = atom_header_line.split()[2:]

        try:
            id_idx = column_names.index('id')
            type_idx = column_names.index('type')
            mol_idx = column_names.index('mol')
            x_idx = column_names.index('x')
            y_idx = column_names.index('y')
            z_idx = column_names.index('z')
        except ValueError as e:
            raise ValueError(f"Missing expected column in 'ITEM:
ATOMS' header: {e}. Found: {column_names}")

        for _ in range(num_atoms):
            line_parts = f.readline().split()
            atom_id = int(line_parts[id_idx])
            atom_type = int(line_parts[type_idx])
            mol_id = int(line_parts[mol_idx])
            x = float(line_parts[x_idx])
            y = float(line_parts[y_idx])
            z = float(line_parts[z_idx])
            molecule_atoms_data[mol_id].append((atom_type, x, y,
z))
            atom_coords[mol_id].append((atom_id, atom_type, x,
y, z))
```

```
    molecule_coms = {}
    for mol_id, atoms_list in molecule_atoms_data.items():
        found_atom2 = False
        for atom_type, x, y, z in atoms_list:
            if atom_type == 2:
                molecule_coms[mol_id] = np.array([x, y, z])
                found_atom2 = True
                break
        if not found_atom2:
            print(f"Warning: Molecule {mol_id} does not contain
atom type 2. Skipping it.")

    return molecule_coms, box_dimensions, atom_coords

# --- Function to write XYZ file ---
def write_xyz_with_cluster_ids(filepath, atom_coords_dict,
molecule_cluster_labels, mol_id_map, frame_comment):
    all_atoms_data = []
    mol_to_cluster_id = {mol_id_map[i]:
molecule_cluster_labels[i] for i in range(len(mol_id_map))}

    for mol_id, atoms_list in atom_coords_dict.items():
        cluster_id_for_mol = mol_to_cluster_id.get(mol_id, -1) +
1
        for atom_id, original_atom_type, x, y, z in atoms_list:
            if original_atom_type == 2:
                all_atoms_data.append((cluster_id_for_mol, x, y,
z))
```

```
    num_atoms = len(all_atoms_data)
    with open(filepath, 'w') as f:
        f.write(f"{num_atoms}\n")
        f.write(f"{frame_comment}\n")
        for effective_type, x, y, z in all_atoms_data:
            f.write(f"{effective_type} {x:.6f} {y:.6f}
{z:.6f}\n")



# --- Main Processing Loop ---
print("Starting time-resolved clustering and tracking...")

all_molecule_ids_ever_seen = set()
persistent_cluster_id_counter = 0

OUTPUT_XYZ_DIR = "cluster_snapshots_atom2_only"
os.makedirs(OUTPUT_XYZ_DIR, exist_ok=True)
print(f"Cluster-colored XYZ snapshots for atom type 2 will be
saved to: '{OUTPUT_XYZ_DIR}/'")

for frame_idx_counter, frame_idx in
enumerate(range(DUMP_FILE_START_INDEX, DUMP_FILE_END_INDEX + 1,
DUMP_FILE_INCREMENT)):
    dump_file_path = f"{DUMP_FILE_PREFIX}{frame_idx}.xyz"

    if not os.path.exists(dump_file_path):
        print(f"Warning: File not found: {dump_file_path}.
Skipping.")
```

```
        continue


    print(f"\nProcessing timestep {frame_idx_counter}:
{dump_file_path}")


    try:
        current_molecule_coms_dict, box_dimensions,
original_atom_data =
parse_lammps_dump_and_get_coms(dump_file_path)


        current_mol_ids_list =
sorted(current_molecule_coms_dict.keys())
        current_com_positions =
np.array([current_molecule_coms_dict[mol_id] for mol_id in
current_mol_ids_list])


        all_molecule_ids_ever_seen.update(current_mol_ids_list)


        if len(current_com_positions) == 0:
            print(f"No molecular COMs found in {dump_file_path}.
Skipping clustering for this frame.")
            continue


        current_frame_cluster_labels =
np.full(len(current_mol_ids_list), -1, dtype=int)


        if frame_idx_counter == 0:
            # --- Step 1: Initialization at t = 0 (Initial
DBSCAN) ---
            print(f"  Performing initial DBSCAN clustering at
t=0...")
```

```
            dbscan = DBSCAN(eps=DBSCAN_EPS_INITIAL,
min_samples=DBSCAN_MIN_SAMPLES_INITIAL)
            labels = dbscan.fit_predict(current_com_positions)


            check_initial_dbscan_results(labels,
len(current_mol_ids_list))


            temp_to_persistent_id_map = {}
            for label in np.unique(labels):
                if label != -1:
                    temp_to_persistent_id_map[label] =
persistent_cluster_id_counter
                    persistent_cluster_id_counter += 1


            for i, mol_id in enumerate(current_mol_ids_list):
                dbscan_label = labels[i]
                if dbscan_label != -1:
                    persistent_cluster_id =
temp_to_persistent_id_map[dbscan_label]
                    current_frame_cluster_labels[i] =
persistent_cluster_id


            new_cluster_centers_this_frame = {}
            for persistent_id in
temp_to_persistent_id_map.values():
                cluster_molecules_coms =
current_com_positions[current_frame_cluster_labels ==
persistent_id]
                if len(cluster_molecules_coms) > 0:
```

```
new_cluster_centers_this_frame[persistent_id] =
np.mean(cluster_molecules_coms, axis=0)
                    cluster_lifespans[persistent_id] += 1


            current_cluster_centers =
new_cluster_centers_this_frame


        else:
            # --- MODIFIED Step 2: Time Propagation using DBSCAN
+ Matching ---
            print(f"  Propagating labels using DBSCAN +
Matching...")


            if not current_cluster_centers:
                print("    No active clusters from previous
frame. All molecules will be noise.")
            else:
                # Step A: Re-cluster the current frame with
DBSCAN to find dense regions
                dbscan_propagator =
DBSCAN(eps=DBSCAN_EPS_PROPAGATE,
min_samples=DBSCAN_MIN_SAMPLES_INITIAL)
                temp_labels =
dbscan_propagator.fit_predict(current_com_positions)


                # Step B: Calculate centers of these newly found
temporary clusters
                new_temp_clusters = {}
                for label in set(temp_labels):
                    if label != -1:
```

```
                        mols_in_new_cluster =
current_com_positions[temp_labels == label]
                        new_temp_clusters[label] =
np.mean(mols_in_new_cluster, axis=0)


                # Step C: Match new temporary clusters to old
persistent clusters
                persistent_id_map = {}
                if new_temp_clusters:
                    old_ids =
list(current_cluster_centers.keys())
                    old_centers =
np.array(list(current_cluster_centers.values()))


                    for temp_label, new_center in
new_temp_clusters.items():
                        distances_sq =
[minimum_image_distance_squared(new_center, old_center,
box_dimensions) for old_center in old_centers]
                        closest_old_id_index =
np.argmin(distances_sq)
                        persistent_id =
old_ids[closest_old_id_index]


                        # Check for conflicts: if an old cluster
is closest to two new ones, assign it to the nearer one
                        if persistent_id not in
persistent_id_map.values():
                            persistent_id_map[temp_label] =
persistent_id
                        else:
```

```
                            existing_temp_label = [k for k, v in
persistent_id_map.items() if v == persistent_id][0]
                            existing_dist_sq =
[minimum_image_distance_squared(new_temp_clusters[existing_temp_
label], old_center, box_dimensions) for old_center in
old_centers][closest_old_id_index]


                            if
distances_sq[closest_old_id_index] < existing_dist_sq:

persistent_id_map.pop(existing_temp_label)
                                persistent_id_map[temp_label] =
persistent_id


                # Step D: Assign the final, persistent labels to
all molecules
                for i, temp_label in enumerate(temp_labels):
                    if temp_label in persistent_id_map:
                        current_frame_cluster_labels[i] =
persistent_id_map[temp_label]


                # Step E: Update cluster centers and lifespans
for the next frame
                new_cluster_centers_this_frame = {}
                active_persistent_ids =
set(persistent_id_map.values())
                for persistent_id in active_persistent_ids:
                    cluster_molecules_coms =
current_com_positions[current_frame_cluster_labels ==
persistent_id]
                    if len(cluster_molecules_coms) > 0:
```

```
new_cluster_centers_this_frame[persistent_id] =
np.mean(cluster_molecules_coms, axis=0)
                        cluster_lifespans[persistent_id] += 1


                current_cluster_centers =
new_cluster_centers_this_frame


        # --- Analysis and Output ---
        for i, mol_id in enumerate(current_mol_ids_list):
            cluster_id = current_frame_cluster_labels[i]

molecule_cluster_transitions[mol_id].append(cluster_id)


        cluster_counts = Counter(current_frame_cluster_labels)
        print(f"  Cluster sizes for {dump_file_path}:")
        for cluster_id, count in sorted(cluster_counts.items()):
            if cluster_id == -1:
                print(f"    Noise (-1): {count} molecules")
            else:
                print(f"    Persistent Cluster {cluster_id}:
{count} molecules")


        output_xyz_filename = os.path.join(OUTPUT_XYZ_DIR,
f"atom2_clusters_frame_{frame_idx}.xyz")
        write_xyz_with_cluster_ids(
            output_xyz_filename,
            original_atom_data,
            current_frame_cluster_labels,
```

```
            current_mol_ids_list,
            f"LAMMPS Timestep: {frame_idx}, Atom 2 Only, Type =
Cluster ID"
        )
        print(f"  Saved atom 2 cluster snapshot to
{output_xyz_filename}")

    except Exception as e:
        print(f"AN ERROR OCCURRED while processing
{dump_file_path}: {e}")
        traceback.print_exc()
        continue

print("\nFinished processing all dump files.")

# --- Final Analysis and Plotting ---
print("\nAnalyzing molecular transitions...")

total_molecules_processed = len(all_molecule_ids_ever_seen)
if total_molecules_processed == 0:
    print("No molecules found or processed.")
    exit()

molecules_transitioned = 0
molecules_trapped = 0
molecules_always_noise = 0

for mol_id in sorted(list(all_molecule_ids_ever_seen)):
```

```
    cluster_history = molecule_cluster_transitions.get(mol_id,
[])

    if not cluster_history:
        molecules_always_noise += 1
        continue

    unique_clusters = set([c for c in cluster_history if c != -
1])

    if len(unique_clusters) > 1:
        molecules_transitioned += 1
    elif len(unique_clusters) == 1:
        molecules_trapped += 1
    else:
        molecules_always_noise += 1

print(f"\nTotal molecules tracked: {total_molecules_processed}")
print(f"Molecules observed transitioning between multiple
*persistent* clusters: {molecules_transitioned}")
print(f"Molecules observed trapped in a single *persistent*
cluster: {molecules_trapped}")
print(f"Molecules observed as always noise (-1) or un-clustered:
{molecules_always_noise}")

tracked_in_clusters = molecules_transitioned + molecules_trapped
if tracked_in_clusters > 0:
    print(f"Percentage of *tracked in clusters* molecules
transitioning: {molecules_transitioned / tracked_in_clusters *
100:.2f}%")
```

```
else:
    print("No molecules were observed in defined clusters.")


# Directory for saving figures and data (added this for saving
consistency)
OUTPUT_FIGURE_DIR = "cluster_analysis_results"
os.makedirs(OUTPUT_FIGURE_DIR, exist_ok=True)
print(f"\nFigures and analysis data will be saved to:
'{OUTPUT_FIGURE_DIR}/'")


# --- Visualization ---
unique_clusters_visited = []
for mol_id, history in molecule_cluster_transitions.items():
    filtered_history = [c for c in history if c != -1]
    if filtered_history:

unique_clusters_visited.append(len(set(filtered_history)))


if unique_clusters_visited:
    plt.figure(figsize=(10, 6))
    bins = np.arange(0.5, max(unique_clusters_visited) + 1.5, 1)
    plt.hist(unique_clusters_visited, bins=bins, rwidth=0.8,
edgecolor='black', alpha=0.7)
    plt.title('Distribution of Unique Persistent Clusters
Visited per Molecule (Excluding Noise)')
    plt.xlabel('Number of Unique Persistent Clusters Visited')
    plt.ylabel('Number of Molecules')
    plt.xticks(range(1, int(max(unique_clusters_visited)) + 1))
    plt.grid(axis='y', alpha=0.75)
```

```
    plt.tight_layout()
    #plt.show()
    plt.savefig(os.path.join(OUTPUT_FIGURE_DIR,
'unique_clusters_visited_histogram.tif'), dpi=300)
else:
    print("\nNo molecules visited a valid cluster to plot the
histogram.")


if cluster_lifespans:
    lifespans = list(cluster_lifespans.values())
    plt.figure(figsize=(10, 6))
    plt.hist(lifespans, bins=range(1, max(lifespans) + 2),
align='left', rwidth=0.8, edgecolor='black', alpha=0.7)
    plt.title('Distribution of Cluster Lifespans (Number of
Frames Existed)')
    plt.xlabel('Lifespan (Number of Frames)')
    plt.ylabel('Number of Clusters')
    plt.xticks(range(1, max(lifespans) + 1))
    plt.grid(axis='y', alpha=0.75)
    plt.tight_layout()
    #plt.show()
    plt.savefig(os.path.join(OUTPUT_FIGURE_DIR,
'cluster_lifespans_histogram.tif'), dpi=300)



# --- START ADDING TRANSFER MATRIX CODE HERE ---


# Directory for saving figures and data (added this for saving
consistency)
```

```
OUTPUT_FIGURE_DIR = "cluster_analysis_results"

os.makedirs(OUTPUT_FIGURE_DIR, exist_ok=True)

print(f"\nFigures and analysis data will be saved to:
'{OUTPUT_FIGURE_DIR}/'")



print("\n--- Calculating Transfer Matrix ---")


# Get all unique states (cluster IDs and -1 for noise) that ever
existed

all_states = set([-1])

for history in molecule_cluster_transitions.values():

    all_states.update(history)


# Sort the states so the matrix is ordered (e.g., -1, 0, 1,
2...)

# This is crucial for consistent matrix representation across
runs

sorted_states = sorted(list(all_states))

state_to_index = {state: i for i, state in
enumerate(sorted_states)}

num_states = len(sorted_states)


transitions_per_history = np.zeros(len(history)-1)

if num_states > 1:

    # Initialize a matrix to store raw transition counts

    transition_counts = np.zeros((num_states, num_states),
dtype=int)
```

```
    # Populate the transition counts
    for mol_id, history in molecule_cluster_transitions.items():
        for i in range(len(history) - 1):
            from_state = history[i]
            to_state = history[i+1]

            from_idx = state_to_index[from_state]
            to_idx = state_to_index[to_state]

            transition_counts[from_idx, to_idx] += 1
            if from_idx != to_idx:
                transitions_per_history[i] += 1

    # Normalize the counts to get the transfer probability
matrix
    # T_ij = probability of going from state i to state j
    row_sums = transition_counts.sum(axis=1, keepdims=True)
    # Avoid division by zero for states that were never departed
from
    transfer_matrix = np.divide(transition_counts, row_sums,
out=np.zeros_like(transition_counts, dtype=float),
where=row_sums!=0)

    # --- Print and Plot the Matrix ---
    print("Transfer Matrix (Rows: From State, Columns: To
State)")
    print("States:", [f"{s}" for s in sorted_states])
    np.set_printoptions(precision=3, suppress=True)
    print(transfer_matrix)
```

```
    np.set_printoptions() # Reset to default


    # Print dataframe of transitions per time slice
    import pandas as pd
    df = pd.DataFrame()
    df["Time Slice"] = np.arange(len(transitions_per_history))
    df["Transitions"] = transitions_per_history
    df.to_csv(os.path.join(OUTPUT_FIGURE_DIR,
'transitions_per_history.csv'), index=False)
    print(df)


    # Save the transfer matrix to a text file
    transfer_matrix_filename = os.path.join(OUTPUT_FIGURE_DIR,
'transfer_matrix.txt')
    header_line = "States: " + ", ".join([f"{s}" for s in
sorted_states]) + "\n"
    header_line += "Rows: From State, Columns: To State"
    np.savetxt(transfer_matrix_filename, transfer_matrix,
fmt='%.4f', header=header_line, comments='# ')
    print(f"Saved transfer matrix to
{transfer_matrix_filename}")


    # Plotting the heatmap
    fig, ax = plt.subplots(figsize=(max(8, num_states*0.8),
max(6, num_states*0.6)))
    cax = ax.imshow(transfer_matrix, cmap='viridis',
interpolation='nearest')


    # Add a color bar
    fig.colorbar(cax, label="Transition Probability")
```

```
    # Set labels for axes
    state_labels = ['Noise' if s == -1 else f'C{s}' for s in
sorted_states]
    ax.set_xticks(np.arange(num_states))
    ax.set_yticks(np.arange(num_states))
    ax.set_xticklabels(state_labels)
    ax.set_yticklabels(state_labels)
    ax.set_xlabel("To State")
    ax.set_ylabel("From State")
    ax.set_title("Cluster Transition Probability Matrix")

    # Rotate x-axis labels for better readability if many
clusters
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
rotation_mode="anchor")

    # Add text annotations for each cell
    for i in range(num_states):
        for j in range(num_states):
            prob = transfer_matrix[i, j]
            if prob > 0:
                # Use a heuristic for text color based on
background luminance
                # A simple approximation: if prob is dark, use
white text, else black
                text_color = "white" if prob > 0.6 else "black"
                ax.text(j, i, f"{prob:.2f}", ha="center",
va="center", color=text_color, fontsize=8)
```

```
    fig.tight_layout()

    # Save the heatmap. Adjusted DPI to prevent "image too
large" error with many clusters.

    plt.savefig(os.path.join(OUTPUT_FIGURE_DIR,
'cluster_transition_matrix_heatmap.tif'), dpi=150)

    #plt.show()


else:

    print("Not enough state transitions to build a transfer
matrix.")


# --- END ADDING TRANSFER MATRIX CODE HERE ---
```