Analyzing Interfaces, Interactions and Self-Assembly in Soft Matter Simulations with PySoftK

Raquel López-Ríos de Castro,[†,‡] Alejandro Santana-Bonilla,[*,¶] Robert M. Ziolek,[‡] and Christian D. Lorenz[*,‡]

†*Department of Chemistry, King's College London, London, SE1 1DB, United Kingdom*

‡*Biological Physics and Soft Matter Group, Department of Physics, King's College London, London, WC2R 2LS, United Kingdom*

¶*Department of Physics, King's College London, London, WC2R 2LS, United Kingdom*

E-mail: alejandro.santana_bonilla@kcl.ac.uk; chris.lorenz@kcl.ac.uk

Abstract

Molecular dynamics simulations have become an essential tool in the study of soft matter and biological macromolecules. The large amount of high-dimensional data produced by such simulations does not immediately elucidate the atomistic mechanisms that underlie complex materials and molecular processes. Analysis of these simulations is complicated: the dynamics intrinsic to soft matter simulations necessitates careful application of specific (often complex) algorithms to extract meaningful molecular scale understanding. There is an ongoing need for high-quality computational workflows to facilitate this analysis in a reproducible manner with minimal user input. In this work, we introduce a series of new computational tools for analyzing soft matter interfaces, molecular interactions (including ring-ring stacking), and self-assembly. In addition,

we include a number of auxiliary tools, including a useful function to unwrap molecular structures that are greater than half the length of their corresponding simulation box. These tools are contained in the PySoftK software package, making application of these algorithms straightforward for the user. These new simulation analysis tools within PySoftK will support high-quality, reproduce analysis of soft matter and biomolecular simulations to bring about new predictive understanding in nano- and biotechnology.

Introduction

Soft matter spans materials science applications in cosmetics,[1–3] pharmaceuticals[4–7] and water decontamination[8] among many others. Advances in synthetic chemistry and formulation science have lead to the development of complex, bespoke soft matter architectures while ever-increasing computational power and simulation techniques have opened up the study of a broad range of such systems *in silico*. An understanding of the interplay of molecular structure, conformational dynamics and intermolecular interactions of the constituent molecules is required to build up generaliazble structure-property relationships to in turn support the rational design of new functional soft matter materials with PySoftK.

Molecular dynamics (MD) simulations provide the framework to investigate the structure, dynamics and interactions at a level of detail that cannot be resolved experimentally. MD simulations have been widely used in the study of soft matter self-assembly.[9–16] MD simulations generate a large amount of data from which it is typically challenging to extract meaningful predictive understanding. This difficulty arises not only from the high dimensionality of the output data, but also from the fast and complex dynamics that are intrinsic to soft matter. Interpreting the molecular mechanisms present in MD simulations typically requires bespoke computational tools to quantify such complex behavior. As a result, it is often not possible to replicate experimental findings or to reproduce quantifiable results.[17]

The computational soft matter community has invested significant effort in simplifying the creation of inputs for soft matter simulations, as exemplified by tools such as PySoftK,[18]

Polymer Structure Predictor,[19] Radonpy[20] and MoSDeF.[21] However, a comprehensive package for analyzing soft matter material properties has not yet been developed. To address this issue, PySoftK (version 1.0) now includes a toolkit designed for analysis of soft matter simulations, providing a unified computational framework in which modelling and analysis can be streamlined under modern software development standards. This feature supports both, data provenance and reproducibility of results. In line with the design commitment of PySoftK to minimize user inputs and provide highly efficient code, PySoftK v1.0 enables the analysis of large-scale soft matter systems.

In this work, new computational analysis tools will be introduced, providing illustrated case studies. The tools are divided into two analysis groups: properties of self-assembled structures and molecular-scale interaction analysis. The software can be employed to investigate different kinds of soft matter systems, since the implemented algorithms are entirely chemically agnostic. With this new release, we aim to support the acceleration computer-aided development of new materials with .

Results and Discussion

Tracking and Analyzing Nanoparticle Self-Assembly

The algorithms introduced in this section are useful for performing analysis of soft matter aggregates, including their self-assembly. The algorithms are formulated to be resolution and chemically agnostic.

Tracking Molecular Self-Assembly. The first tool presented is a method to track soft matter self-assembly. The Spatial Clustering Protocol (SCP) algorithm provides a fast way to label molecules based on the cluster or aggregate in which they reside during a self-assembly process. We make use of simple graph theory to represent molecules as a graph, where each molecule is a node and if the distance between specified atoms of any two

given molecules is less than the defined cutoff, an edge is added between these two nodes in the graph. In this representation, clusters are rapidly identifiable as connected subgraphs. This makes this analysis suitably fast, such that the dynamical self-assembly process can be quickly rationalized over an entire trajectory. The algorithms returns a pandas dataframe which contains the molecule resids for each cluster and the cluster size at each time step. More details about the application of graph theory to investigating self-assembly can be found in the Electronic Supplementary Information (ESI).

The choice of which atom to use in the identification of molecular contacts is specific to the molecule of interest and one can select as many atoms as necessary to accurately describe the self-assembly of the molecules. The SCP algorithm calculates the distances between all the selected atoms of the chosen molecules. If any of these distances is below the cutoff it will then add them to the same subgraph. It is important to note that choosing a large number of atoms will slow down this calculation. Figure 1 shows how different atom selection choices affect the output of the SCP clustering for an ABA triblock copolymer. Figure 1 (a) shows the system, two micelles formed by ABA triblock polymers (the A block is hydrophilic; the B block is hydrophobic). Figure 1 (b) displays the desired clustering output. This result is achieved by selecting the backbone C atom of the middle monomer of the hydrophobic block. Since the hydrophobic block tightly interacts with those of other polymers in the micelle, picking atoms within this domain is a reasonable selection. On the other hand, Figure 1 (c) shows the clustering performed for the same system but picking atoms at the end of the hydrophilic blocks. The hydrophilic atoms at the end of the polymer chains are not suitable choices for clustering and as such, the clusters obtained do not reflect the formation of two micelles.

Apart from selecting atoms, users must also specify a cutoff distance for clustering. This distance determines whether two molecules belong to the same cluster. The cutoff distance might be obtained from the radial distribution function (RDF) of the selected atoms (either the position of the RDF maximum or first minimum). The lack of a single clear choice for the

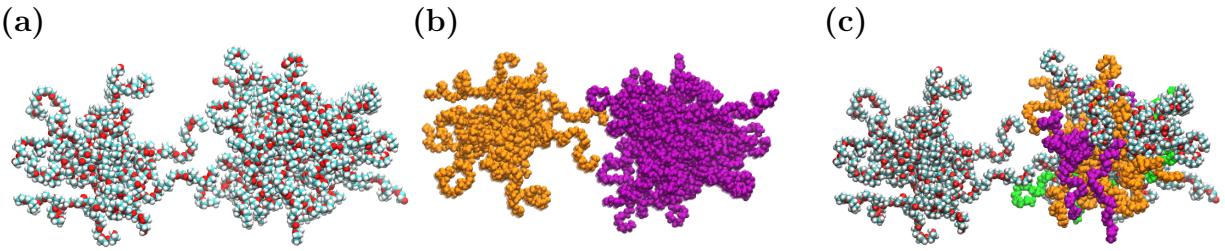

Figure 1: **Atom selection strongly influences the results of SCP clustering.** (a) Triblock hydrophilic terminated polymer system to cluster. (b) `SCP` algorithm applied on the system in (a) picking the middle hydrophobic monomer C atoms for the clustering. Polymers with the same color belong to the same cluster outputted by the algorithm. Clearly, the clustering here is done correctly. (c)`SCP` algorithm applied on the system in (a) picking the ending hydrophilic atoms. Polymers with the same color belong to the same cluster outputted by the algorithm. It is evident that the clustering is done badly when these atoms are used. The ending hydrophilic monomers do not play a key role in the intermolecular interactions with other polymers during the self-assembly of the micelle, so they are not good clustering candidates.

cutoff distance is caused by the complex structures of self-assembled structures compared to, for examples, ions in solutions (i.e., the system is inherently not well-mixed). It is necessary to consider a range of cutoff distances, in our experience typically investigating a range of cutoffs between 8 Å and 13 Å is useful. Visual inspection of the resulting clusters determines the most appropriate cutoff distance straightforwardly in most cases. As well as soft matter, the SCP algorithm can be readily applied to biological systems, highlighting its broad applicability. For example, Figure 2 shows the result of applying the SCP algorithm to a coarse-grained protein simulation to measure the aggregation of the transmembrane domains of a protein within a lipid bilayer.

Unwrapping Nanoparticles Across PBC. When soft matter self assembles, the nanoparticle that is formed can often be found to span the more than half the length of the simulation box in at least one dimension. In order to accurately analyse the nanoparticle and its environment, it is necessary to accurately represent the location of all of the molecules that make up the nanoparticle while accounting for periodic boundary conditions. Various tools have been implemented elsewhere, which can accurately reconstruct the position of molecules across the periodic boundary conditions (PBC) but fail to do so when structures

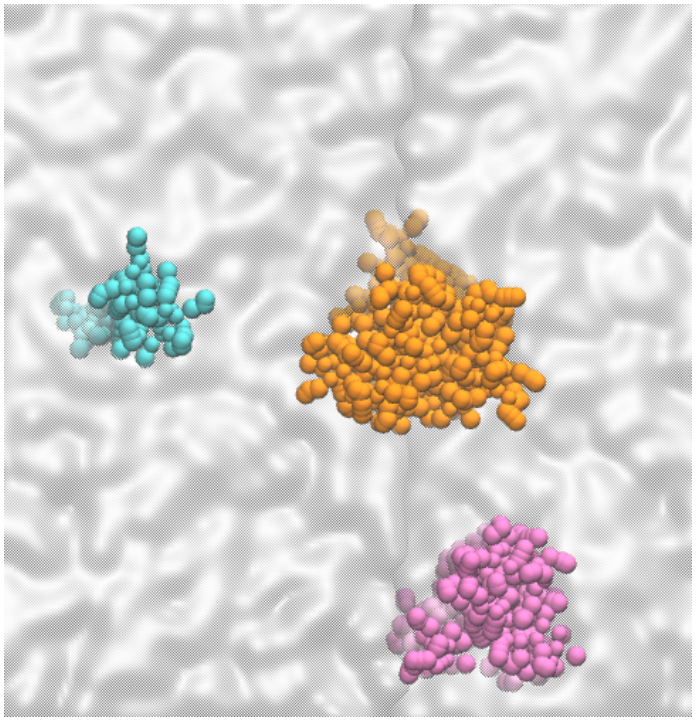

Figure 2: `SCP` **applied to a CG protein simulation to measure peptide aggregation.** Since the input needed to run this tool is not exclusive to polymers, it can be applied to any other type of system to measure molecular clustering. In this case, this is the result of the `SCP` being applied on CG transmembrane peptides inserted into a membrane. This is a top-view of the peptides-membrane system. Proteins colored in the same way belong to the same cluster. Blue cluster contains two peptides, the pink cluster has 6 peptides and the orange cluster has 8 peptides. The lipids membrane is colored in silver. Representation is not to scale.

(or indeed molecules) span more than half of the simulation box size in a given dimension.

The `make_micelle_whole` tool in PySoftK is able to successfully position the molecules within a self-assembled aggregate if it spans one or more dimensions of the simulation box and if the aggregate is larger than a half of the length of the simulation box in one or more dimensions. Figure 3 (a) shows a polymer micelle that has spanned the simulation box in at least two dimensions.Figure 3 (b) shows the effectiveness of the PySoftK tool `make_micelle_whole`, which is able to successfully reconstruct the micelle in Figure 3 (a) On the other hand, Figures 3 (c) and (d) demonstrate that MDAnalysis v2.5 and GROMACS 2023, respectively, are not able to reconstruct an accurate structure of the micelle using the same input as files as PySoftK's implementation of `make_micelle_whole`. Our algorithm

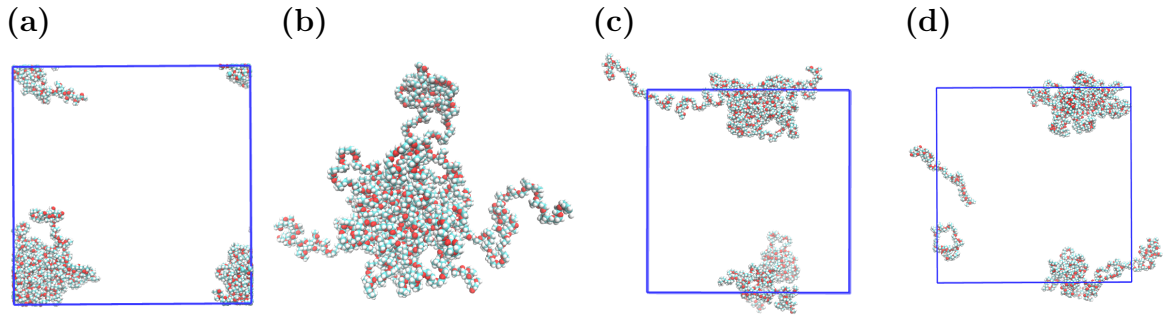

Figure 3: **Effect of `make_micelle_whole` on structure that is greater than half the box length.** Trying to make the micelle in (a) which is broken across the PBC and that is greater than half the box length in one dimension using (b) `make_micelle_whole` (c) MDAnalysis transformation (d) GROMACS 2023 `gmx trjconv -pbc mol` command. It is clear that only the PySoftK analysis tool `make_micelle_whole` is able to properly make the micelle whole across the PBC even if one of its dimensions is greater than half the box length.

offers a more reliable and robust reconstruction of structures disrupted by periodic boundary conditions, while streamlining the user's workflow and minimizing the required input.

Failing to accurately reproduce structures across the PBC can lead to inconsistencies in the analysis and interpretation of soft matter systems.[22] In contrast, the PySoftK `make_micelle_whole` tool is able to accurately represent the coordinates of molecular structures which span the PBC, even if their size is greater than half the box size. More details about this algorithm are described in the ESI. Therefore, `make_micelle_whole` provides an accurate representation of the system, which ensures that any physical properties of the self-assembled structure and its environment is calculated correctly. For example, certain functions of MDAnalysis, such as `radius_of_gyration()` or `moment_of_inertia()`, may produce erroneous results when applied to molecules and aggregates that span at least a single dimension, leading to artefacts in the simulation analysis. For instance, Figure 4 shows the difference between using solely the MDanalysis `radius_of_gyration(pbc=True)` function (Figure 4 (b)) and the MDanalysis `radius_of_gyration()` function applied on the whole structure created by `make_micelle_whole` (Figure 4 (c)) of the micelle depicted in Figure 4 (a). From Figure 4 (b), it is clear that MD-Analysis with the `pbc=True` parameter is not able to take the periodic boundary conditions into account correctly for the micelle, which leads to erroneous radius of gyration values.

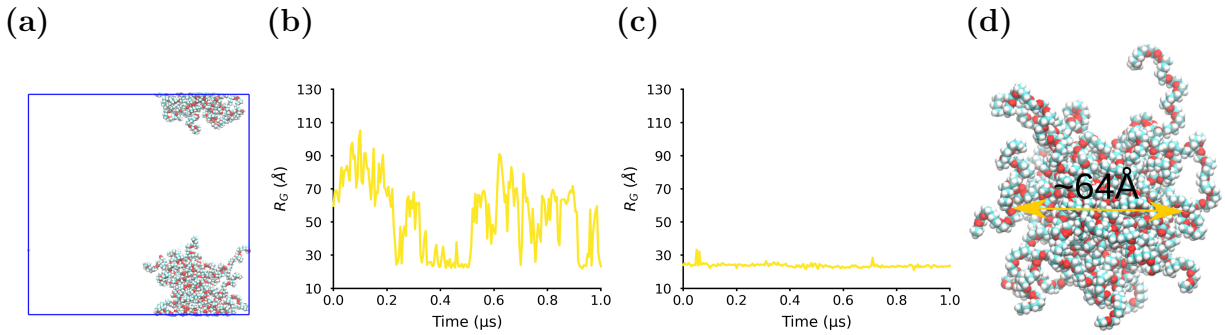

Figure 4: **Effect of `make_micelle_whole` on the calculation of the radius of gyration.** Result from running native MDAnalysis `radius_of_gyration()` on the system when broken across the PBC as depicted in (a) with (b) `pbc=True` MDAnalysis option and (c) on the whole coordinates obtained with `make_micelle_whole`. It is clear that the `radius_of_gyration(pbc=True)` implementation cannot take PBC effects properly into account during the radius of gyration calculation. (d) Snapshot of the micelle made whole with the diameter indicated. Representation is not to scale.

However, Figure 4 (c) shows that when this same function `radius_of_gyration()` is applied on the whole coordinates obtained with `make_micelle_whole` the radius of gyration is properly calculated.

Auxiliary Functions. We have included a number of smaller functions to perform simple analysis tasks in a unified way. These are briefly described below.

Radius of Gyration. This function allows the user to easily calculate the radius of gyration of a structure that is not always formed by the same molecules throughout the simulation. It utilises the MDAnalysis function `radius_of_gyration(pbc=True)`, but allows users to specify the atom positions and their corresponding resids on which to perform this calculation at each time step. Figure 4 shows the comparison between using the MDAnalysis radius of gyration function alone compared to the PySoftK `rgyr`, which captures the right radius of gyration of the micelle when computed on the whole coordinates from `make_micelle_whole`.

Eccentricity. Eccentricity, a metric quantifying a structure's deviation from a perfect sphere, serves as a useful tool for assessing the shape of spherical-like soft matter aggregates. The `ecc` tool calculates the eccentricity for any molecular structure by leveraging the MDAnalysis function `moment_of_inertia()` and employing the following formula:

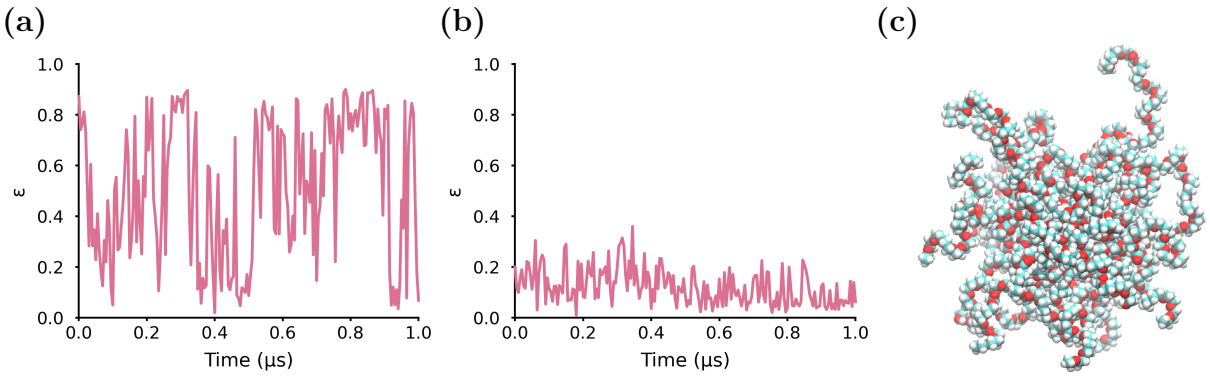

Figure 5: **Eccentricity calculation of polymer micelle.** Calculation of the eccentricity of a micelle solving Equation 1 over time with (a) only MDAnalysis `moment_of_inertia()` and (b) PySoftK `ecc` function. It is clear that `ecc` can easily use the correct polymers belonging to the cluster at each time step and the correct atom positions across the PBC to compute the eccentricity values, while the MDAnalysis function on its own, even with `pbc=True` is not able to compute them properly. (c) Snapshot of the micelle on which the eccentricity is being calculated, clearly it is slightly spherical, so the values from (b) are the correct ones. The micelle representation is not to scale.

$$\epsilon = 1 - \frac{I_{min}}{I_{mean}} \tag{1}$$

where ϵ is the eccentricity value, I_{min} is the minimum moment of inertia across all axis of the molecule(s), and I_{mean} is the mean moment of inertia over all axis of the molecule(s). A perfect sphere corresponds to $\epsilon = 0$, while increasing values indicate more oblong structures. Similar to the `rgyr` calculation, the `ecc` tool can account for varying number of molecules within the structure and by using the coordinates of the micelle having been corrected by `make_micelle_whole` as input, it ensures accurate calculations without artefacts from PBC. Figure 5 illustrates how `ecc` accurately computes the eccentricity of a micelle over time compared to the MDAnalysis strategy, since it uses the correct coordinate reconstruction across the PBC.

Spherical Density. Understanding the internal distribution of components within self-assembled aggregates is essential for characterizing their structure. For soft-matter aggregates that are approximately spherical, the density of their various components and its environments can be calculated with reference to the aggregate's center of mass ('spherical

density'). There are numerous MD studies that measure the density of components of polymer micelles using this approach,[12,23,24] but there are limited open-sourced codes that can be used to carry out such analysis. The PySoftK `spherical\_density` tool allows users to easily calculate the spherical density over time, even for structures with varying molecule numbers throughout the simulation. It computes the average density (over time) with respect to the distance from the center of mass of the molecular structure. This is achieved by dividing up the simulation space into spherical bins with reference to the origin at the center of mass of the aggregate. For each of these bins, the number of particles in them is counted and divided by the volume of the bin. This calculation is described in Equation 2.

$$\rho_{spherical\ bin} = \frac{n_{beads}}{\frac{4}{3}\pi(R_{out}^3 - {R_{in}}^3)} \tag{2}$$

Where $\rho_{spherical\ bin}$ is the spherical density of a particular bin, n_{beads} is the number of beads in the same bin, R_{out} is the outer bin radius (the upper bound of the bin) and R_{in} is the inner bin radius (the lower bound of the bin). Therefore, the output of the `spherical_density` tool is a Numpy array with the average spherical density values across time for each bin. Furthermore, it is worth noting that this algorithm can handle the calculation of spherical density of an aggregates with varying number of molecules for each time step.

Additionally, the `spherical_density_water` class provides a customized version of this algorithm to investigate only the water density. It is a separate function because to properly calculate the distances of water with respect to the center of mass of the soft-matter aggregate, the water coordinates need to be wrapped around the coordinates of the aggregate while correctly making it whole through the periodic boundary conditions. Since this process is computationally more expensive, a different class was developed which works in exactly the same way as the `spherical_density` function, but the atom names of the solvent need to be inputted by the user. For water density calculations, only the oxygen atoms need to be selected.

Intrinsic Density. If an aggregate does not have a smooth interface or surface, then spherical density approaches are not suitable for accurately determining the density of the molecular constituents of the aggregates and those surrounding it. For these cases, intrinsic interface techniques have been previously employed to analyse the distribution of components within such irregular structures.[25] A computational method that has been used to study the intrinsic density of polymer micelles with a distinct core-shell structure, is the intrinsic core-shell interface method (ICSI).[12,26] This approach divides the constituent molecules within a molecular aggregate into the core and shell region with an intermediate region in between these. The masses of the core and shell are determined, and the volumes of the core, shell, and interface are calculated. The total density of the aggregate is then computed by dividing the total mass by the sum of the volumes of the core, shell, and interface, which accounts for the varying properties of the core and shell. PySoftK's `intrinsic_density` class harnesses the ICSI method to perform intrinsic density calculations. PySoftK's implementation enables seamless processing of the entire reconstructed coordinate set provided by `make_micelle_whole` . Also, it can handle varying numbers of molecules constituting the structure of interest at each time step. The usage of this function closely resembles that of the spherical density function. Additionally, there is a `intrinsic_density_water` class for the computation of the intrinsic density of water.

The `intrinsic_density` class outputs a NumPy array with the average densities (over time) with respect to the distance to the core-shell interface, where a distance of 0 represents the location of this interface. It also outputs another NumPy array with the values of the bins used in the density calculation. Figure 6 shows density calculations of the same system using the spherical density tool and intrinsic density tool respectively. It can be seen that PySoftK spherical density tools output the density as a function of the distance from the center of mass of the aggregate(Figure 6 (a)), while the intrinsic density outputs the density as a function of the distance from the core-shell interface (Figure 6 (b)). Therefore, negative distance values in the intrinsic density represent atoms within the core. Given the spherical

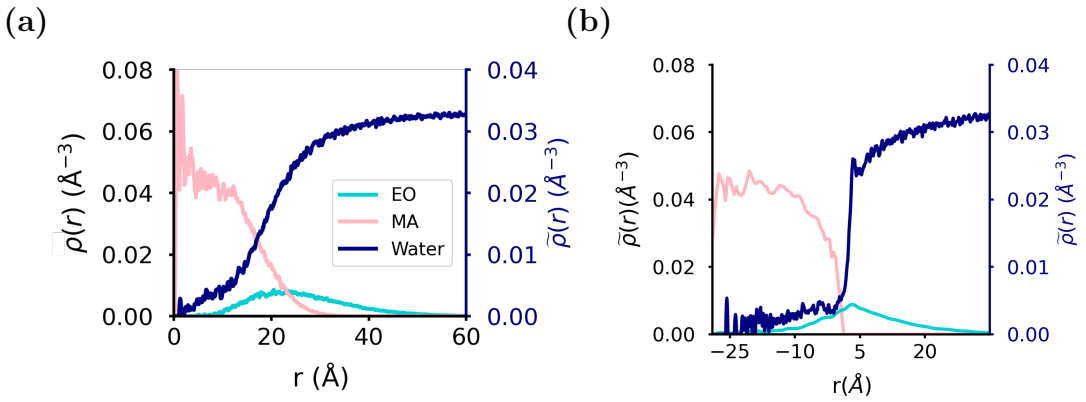

Figure 6: **Density calculation comparison.** Density calculation of a spherical micelle formed by PEO-PMA polymers using: (a) `spherical_density` (b) `intrinsic_density`. The results are very similar, since the micelle is spherical and has a clear core-shell interface. Therefore, in this case both methods can be used to determine the distribution of components within the micelle.

shape of the micelle, both density methods yield similar results.

Molecular-scale interaction analysis

This section describes the tools that we have developed to analyze different intermolecular interactions that play important roles in the self-assembly of soft matter. This includes calculations of the contacts between molecules, ring stacking interactions, and solvent interactions.

Contact map calculations. Quantifying intermolecular interactions is vital to understand the mechanisms that drive molecular-scale phenomena. By determining which parts of two different types of molecules are in contact with each other, one can then perform a more detailed analysis of the specific types of interactions (e.g. hydrogen bonding, ionic interactions, ring-ring stacking) present in the system. The `contacts` tool calculates the contacts between molecules by measuring the distance between selected atoms. If the intermolecular distance between two selected atoms is less than a user-defined cutoff, it is considered a contact. The values of the distance cutoff used to characterise the interactions of different types of molecules will vary, and can be determined from radial distribution functions between

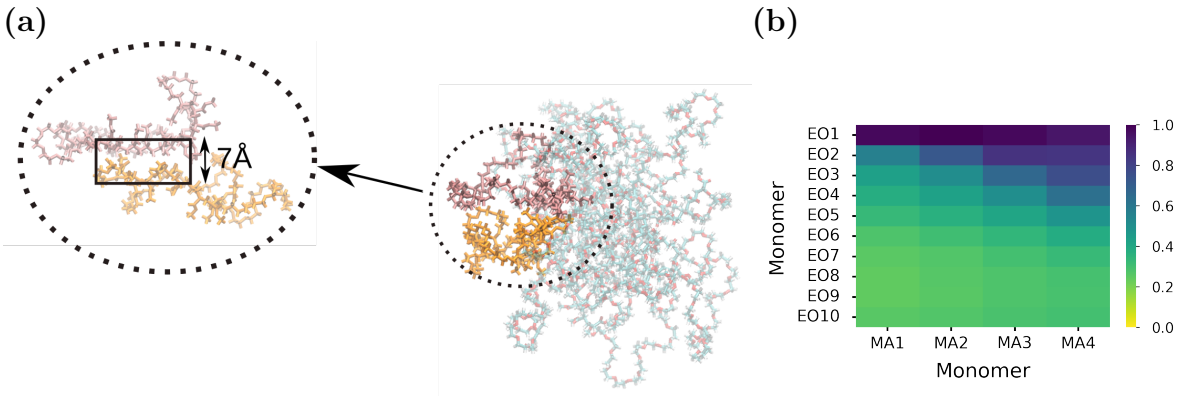

Figure 7: **Normalized `contacts` output.** (a) Snapshot of two cyclic polymer within a micelle that are in contact. The inter-molecular distance of atoms in the black rectangular box is 7 Å. These inter-molecular interactions would be picked up by the `contacts` algorithm as contacts. (b) Output of the calculation of the intermolecular contacts of PEO-PMA polymers forming a micelle. The output matrix is represented as a heatmap. From here it is clear that the rows and columns represent the contact groups of the calculation.

the specific atoms of the two molecules used in the calculation or through an analysis of the minimum distance between these atoms on two molecules that are known to have aggregated during the simulation. Values of the distance cutoff have been found to be between 4 Å and 7 Å as shown elsewhere.[16,27,28] A visual representation of intermolecular contacts between two poly(ethylene oxide) (PEO) - poly(methyl acrylate) (PMA) polymers within a micelle is shown in Figure 7 (a). The `contacts` class can utilize the output of the `make_micelle_whole` tool as an input. This ensures that the distances between atoms within the molecules are calculated correctly when accounting for the periodic boundary effects within the system. Figure 7 (b) shows a heatmap representation of the normalized matrix generated by `contacts`, which shows the intermolecular EO-MA interactions of PEO-PMA polymers.

Ring Stacking Analysis. Ring stacking interactions are the driving force behind many collective phenomena ranging from DNA base pairing,[29] protein-drug binding,[30] and through-space charge transfer in conjugated polymers.[31] A class to identify ring-ring interactions has been developed for PySoftK v1.0. Note that the software ProLIF[32] can also calculate ring stacking interactions, but it is specific to protein-ligand systems, while the tool implemented in PySoftK v1.0. can be applied to any soft matter system. PySoftK's

rin-ring interaction algorithm consists of three stages; firstly, all atoms belonging to aromatic (conjugated) rings within the chosen molecules (or parts of molecules) are detected. Secondly, pairs of molecules within the system are screened using a cutoff to define molecules (or parts of molecules) in close contact. Finally, those molecules (or parts of molecules) which are found to be in contact are selected to have the necessary geometrical properties between their aromatic units calculated. The algorithm is explained in detail in the SI. The `RSA` class allows users to identify ring stacking patterns within a simulation and calculate the network formed by the stacking interactions and their evolution over time. This tool implemented in PySoftK 1.0 enables the user to perform this analysis based on minimal input parameters. These input parameters are: the maximum distance cutoff between two rings for which the ring stacking is calculated, the angle cutoff used to determine the range for which two rings are considered to be stacked, the frames on which to run the analysis and the output file name. Default values adopted are a distance cutoff of 10 Å and an angular cutoff of 20°. This algorithm has been tested on amorphous F8BT (see Figure 8 (a)) and the protein-potein interaction complex formed between TREM2 and DAP12 (see Figure 8 (b)).

Solvation Analysis. Solvation analysis plays a crucial role in understanding the structure and dynamics of amphiphilic soft matter. This analysis allows us to quantify the solvation cells around molecules, and to predict hydrophobic interactions.[16,33] Currently, there is no readily available open-source software that simplifies solvation calculations in this context, and that can be used in any soft-matter system. MDAnalysis has the class `MDAnalysis.analysis.waterdynamics`, but it focuses on the dynamics of water and the interactions of water with other molecules via hydrogen bonds. PySoftK's `solvation` class provides a straightforward method for quantifying solvation by determining the number of solvent molecules within the first solvation shell of the specified molecules. In doing so, PySoftK will identify the solvent molecules that are hydrogen-bonded to a particular part of a molecule, as well as any that might be attracted via other types of interactions (e.g. electrostatic,

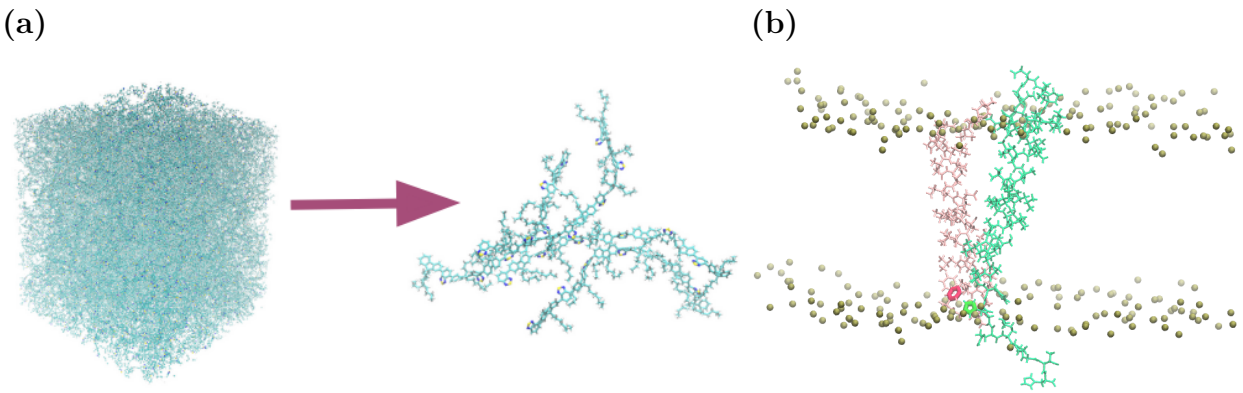

Figure 8: **Example of ring-ring stacking calculation using the `RSA` class on (a) a polymer melt and (b) on the TREM12 and DAP12 protein.** In (a) the `RSA` is able to obtain ring stacking in complicated and large system. The purple arrow points to a subset of polymers interacting in the amorphous phase via ring stacking. This cluster has been identified with the `RSA` tool. (b) `RSA` applied to determine ring stacking events that drive protein-protein interactions. TREM12 is shown in pink and DAP12 in green. An observed ring stacking interaction are denoted by the bold representation. Phosphate groups of the membrane are colored in dark green. Representations are not to scale.

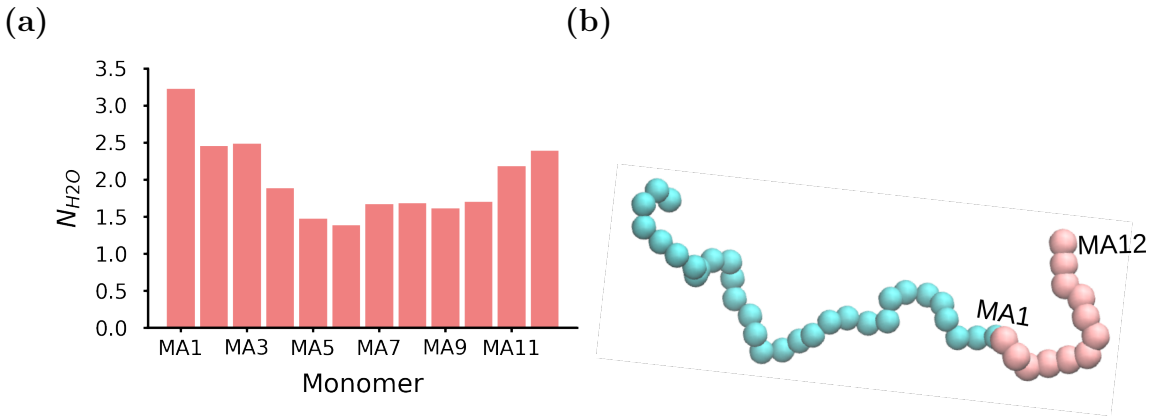

Figure 9: **Solvation calculation of polymer micelle.** Average solvation of a diblock BCP micelle. (a) Average over time of water coordination numbers for all monomers of hydrophobic block. (b) Snapshot of the diblock polymer being studied. PMA in pink and PEO in blue. Polymer representation is not to scale.

hydrophobic), and thus is a more general way of identifying the solvation of different parts of molecules. Figure 9 shows the average solvation number calculated for all hydrophobic MA monomers of a PEO-PMA polymeric micelle.

The `solvation` class operates similarly to the `contacts` class, where distances between selected atoms of the molecules and specific solvent atoms are computed. If the distance is

shorter than a user-defined cutoff, the selected atom in a molecule of interest is considered to be solvated. The inputted cut-off distance can be used to represent a given solvation shell of the molecule (e.g. first or second solvation shell) depending on what is of interest to the user. When using water as the solvent, it is recommended to select just the oxygen water atoms to speed up calculations. Consistent with the other tools in PySoftK v2.0, the solvation code seamlessly handles the varying number of molecules and the correct atomic coordinates at each time step.

Conclusion

Pysoftk v1.0 adds a complete standalone module for the analysis of soft matter systems. The new module described in this paper provides a set of interconneced tools that are useful for determining the physical properties of soft matter self-assembled aggregates as well as the molecular-scale interactions that underlie such emergent behavior. One of the key features of PySoftK v1.0 is that it properly accounts for periodic boundary conditions when determining the positions of atoms within the molecules that make up a soft matter aggregate, particularly if the aggregate is larger than half the size of the simulation box in one or multiple dimensions. Other software tools are not designed to account for molecular assemblies of such size. A thorough set of tests have been created to cover all code to ensure its correct functionality. Furthermore, PySoftK v1.0 is designed to provide maximum flexibility to the user, so most functions output the data per outputted configuration of the trajectory, so that the user can decide how to represent or further process the data. Although the initial version of PySoftK has a particular focus on polymers, the analysis module has been created such that it is fully chemically agnostic. The goal of this module is to create an open-source platform that allows users to analyse complex structures, dynamics and interactions in their simulations with minimal user input. PySoftK v1.0 contributes to the standardisation of molecular-scale simulation analysis, which will promote accurate comparisons across differ-

ent simulations to support the rational *in silico* design of new soft materials.

Acknowledgement

The authors thank the e-Research department at King's College London for computational resources.[34] We are grateful to the UK Materials and Molecular Modelling Hub for computational resources, which is partially funded by EPSRC (EP/T022213/1, EP/W032260/1 and EP/P020194/1). R. L.-R. D. C. acknowledges the support by the Biotechnology and Biological Sciences Research Council (BB/T008709/1) via the London Interdisciplinary Doctoral Programme (LIDo). R. M. Z. and C. D. L. acknowledge the Engineering and Physical Sciences Research Council (EPSRC) for funding (EP/V049771/1). For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence (where permitted by UKRI, 'Open Government Licence' or 'Creative Commons Attribution No-derivatives (CC BY-ND) public copyright licence' may be stated instead) to any Author Accepted Manuscript version arising.

Supporting Information Available

Theoretical explanations of Graph Theory in our implementation, concrete examples, and PySoftK code snippets for hands-on practice are included in the ESI.

References

(1) Mitura, S.; Sionkowska, A.; Jaiswal, A. Biopolymers for hydrogels in cosmetics: review. *J. Mater. Sci.: Mater. Med.* **2020**, *31*, 50.

(2) Ahmadi, D.; Ledder, R.; Mahmoudi, N.; Li, P.; Tellam, J.; Robinson, D.; Heenan, R. K.; Smith, P.; Lorenz, C. D.; Barlow, D. J.; Lawrence, M. J. Supramolecular architecture

of a multi-component biomimetic lipid barrier formulation. *J. Colloid Interface Sci.* **2021**, *587*, 597–612.

(3) Gupta, S.; Sharma, S.; Kumar Naada, A.; Saad Bala Husain, M.; Gupta, A. Biopolymers from waste biomass and its applications in the cosmetic industry: A review. *Mater. Today: Proc.* **2022**, *68*, 873–879.

(4) Kohli, A. G.; Kierstead, P. H.; Venditto, V. J.; Walsh, C. L.; Szoka, F. C. Designer lipids for drug delivery: From heads to tails. *J. Control. Release* **2014**, *190*, 274–287.

(5) Chen, C. H.; Liu, Y.-H.; Eskandari, A.; Ghimire, J.; Lin, L. C.-W.; Fang, Z.-S.; Wimley, W. C.; Ulmschneider, J. P.; Suntharalingam, K.; Hu, C.-M. J., et al. Integrated Design of a Membrane-Lytic Peptide-Based Intravenous Nanotherapeutic Suppresses Triple-Negative Breast Cancer. *Advanced Science* **2022**, *9*, 2105506.

(6) Ishkhanyan, H.; Rhys, N. H.; Barlow, D. J.; Lawrence, M. J.; Lorenz, C. D. Impact of drug aggregation on the structural and dynamic properties of Triton X-100 micelles. *J. Mol. Liq.* **2023**, *385*, 122376.

(7) Saaka, Y.; Allen, D.; Terry, A. E.; Lorenz, C. D.; Barlow, D. J.; Lawrence, M. J. Characterisation of the apparent aqueous solubility enhancement of testosterone analogues in micelles of dodecyl-chained surfactants with different headgroups. *J. Mol. Liq.* **2023**, *385*, 122376.

(8) Shah, A.; Shahzad, S.; Munir, A.; Nadagouda, M. N.; Khan, G. S.; Shams, D. F.; Dionysiou, D. D.; Rana, U. A. Micelles as Soil and Water Decontamination Agents. *Chemical Reviews* **2016**, *116*, 6042–6074.

(9) Srinivas, G.; Discher, D. E.; Klein, M. L. Self-assembly and properties of diblock copolymers by coarse-grain molecular dynamics. *Nature materials* **2004**, *3*, 638–644.

(10) Jorge, M. Molecular Dynamics Simulation of Self-Assembly of *n*-Decyltrimethylammonium Bromide Micelles. *Langmuir* **2008**, *24*, 5714–5725.

(11) Wang, H.; Zhang, H.; Liu, C.; Yuan, S. Coarse-grained molecular dynamics simulation of self-assembly of polyacrylamide and sodium dodecylsulfate in aqueous solution. *J. Colloid Interface Sci.* **2012**, *386*, 205–211.

(12) Ziolek, R. M.; Omar, J.; Hu, W.; Porcar, L.; Gonzalez-Gaitano, G.; Dreiss, C. A.; Lorenz, C. D. Understanding the pH-directed self-assembly of a four-arm block copolymer. *Macromolecules* **2020**, *53*, 11065–11076.

(13) Pink, D. L.; Foglia, F.; Barlow, D. J.; Lawrence, M. J.; Lorenz, C. D. The Impact of Lipid Digestion on the Dynamic and Structural Properties of Micelles. *Small* **2021**, *27*, 2004761.

(14) Pink, D. L.; Loruthai, O.; Ziolek, R. M.; Terry, A. E.; Barlow, D. J.; Lawrence, M. J.; Lorenz, C. D. Interplay of lipid and surfactant: Impact on nanoparticle structure. *J. Colloid Interface Sci.* **2021**, *597*, 278–288.

(15) Bhendale, M.; Singh, J. K. Molecular Insights on Morphology, Composition, and Stability of Mixed Micelles Formed by Ionic Surfactant and Nonionic Block Copolymer in Water Using Coarse-Grained Molecular Dynamics Simulations. *Langmuir* **2023**, *39*, 5031–5040.

(16) De Castro, R. L.-R.; Ziolek, R.; Lorenz, C. Topology-Controlled Self-Assembly of Amphiphilic Block Copolymers. **2023**,

(17) Gartner III, T. E.; Jayaraman, A. Modeling and simulations of polymers: a roadmap. *Macromolecules* **2019**, *52*, 755–786.

(18) Santana-Bonilla, A.; Lopez-Rios de Castro, R.; Sun, P.; Ziolek, R. M.; Lorenz, C. D.

Modular Software for Generating and Modeling Diverse Polymer Databases. *Journal of Chemical Information and Modeling* **2023**,

(19) Sahu, H.; Shen, K.-H.; Montoya, J. H.; Tran, H.; Ramprasad, R. Polymer structure predictor (psp): a python toolkit for predicting atomic-level structural models for a range of polymer geometries. *Journal of Chemical Theory and Computation* **2022**, *18*, 2737–2748.

(20) Hayashi, Y.; Shiomi, J.; Morikawa, J.; Yoshida, R. RadonPy: automated physical property calculation using all-atom classical molecular dynamics simulations for polymer informatics. *npj Computational Materials* **2022**, *8*, 222.

(21) Summers, A. Z.; Gilmer, J. B.; Iacovella, C. R.; Cummings, P. T.; McCabe, C. MoSDeF, a Python Framework Enabling Large-Scale Computational Screening of Soft Matter: Application to Chemistry-Property Relationships in Lubricating Monolayer Films. *J. Chem. Theory Comput.* **2020**, *16*, 1779–1793.

(22) Smith, P.; Lorenz, C. D. LiPyphilic: A Python toolkit for the analysis of lipid membrane simulations. *bioRxiv* **2021**,

(23) Kovacevic, M.; Balaz, I.; Marson, D.; Laurini, E.; Jovic, B. Mixed-monolayer functionalized gold nanoparticles for cancer treatment: Atomistic molecular dynamics simulations study. *Biosystems* **2021**, *202*, 104354.

(24) Wilkosz, N.; Łazarski, G.; Kovacik, L.; Gargas, P.; Nowakowska, M.; Jamroz, D.; Kepczynski, M. Molecular insight into drug-loading capacity of PEG–PLGA nanoparticles for itraconazole. *The Journal of Physical Chemistry B* **2018**, *122*, 7080–7090.

(25) Sega, M.; Kantorovich, S. S.; Jedlovszky, P.; Jorge, M. The generalized identification of truly interfacial molecules (ITIM) algorithm for nonplanar interfaces. *The Journal of chemical physics* **2013**, *138*.

(26) Ziolek, R. M.; Smith, P.; Pink, D. L.; Dreiss, C. A.; Lorenz, C. D. Unsupervised learning unravels the structure of four-arm and linear block copolymer micelles. *Macromolecules* **2021**, *54*, 3755–3768.

(27) Ulmschneider, M. B.; Doux, J. P.; Killian, J. A.; Smith, J. C.; Ulmschneider, J. P. Mechanism and kinetics of peptide partitioning into membranes from all-atom simulations of thermostable peptides. *Journal of the American Chemical Society* **2010**, *132*, 3452–3460.

(28) Sun, X.; Feng, Z.; Hou, T.; Li, Y. Mechanism of graphene oxide as an enzyme inhibitor from molecular dynamics simulations. *ACS applied materials & interfaces* **2014**, *6*, 7153–7163.

(29) Kool, E. T. Hydrogen bonding, base stacking, and steric effects in DNA replication. *Annual review of biophysics and biomolecular structure* **2001**, *30*, 1–22.

(30) Liu, Y.; Liu, B.-Y.; Hao, P.; Li, X.; Li, Y.-X.; Wang, J.-F. π–π Stacking mediated drug–drug interactions in human CYP2E1. *Proteins: Structure, Function, and Bioinformatics* **2013**, *81*, 945–954.

(31) Schwartz, B. J. Conjugated polymers as molecular materials: How chain conformation and film morphology influence energy transfer and interchain interactions. *Annual review of physical chemistry* **2003**, *54*, 141–172.

(32) Bouysset, C.; Fiorucci, S. ProLIF: a library to encode molecular interactions as fingerprints. *Journal of Cheminformatics* **2021**, *13*, 72.

(33) Jafari, M.; Doustdar, F.; Mehrnejad, F. Molecular self-assembly strategy for encapsulation of an amphipathic α-helical antimicrobial peptide into the different polymeric and copolymeric nanoparticles. *Journal of chemical information and modeling* **2018**, *59*, 550–563.

(34) King's College London (2022). King's Computational Research, Engineering and Technology Environment (CREATE). https://doi.org/10.18742/rnvf-m076.

TOC Graphic