DOI: 10.14529/jsfi230307

MOUSE2: Molecular Ordering Utilities for Simulations, Edition 2

Mikhail K. Glagolev[1], *Anna A. Glagoleva*[1],
Valentina V. Vasilevskaya[1,2]

The progress in spatial and temporal scales of molecular simulations attainable with modern supercomputers makes the processing of the simulation data a challenging task in itself. One of the most important applications is the simulation of living systems, which are based on polymers, as well as simulation of polymer systems in material sciences. The behavior of many polymer systems is determined by local ordering of polymer chains, which on many occasions contain helical motifs. This ordering can be hard to quantify visually and using standard tools. To overcome these problems, we have developed an original toolkit to look into orientational and especially chiral ordering in polymer systems, which can quantify the orientational ordering of polymers based on their spatial proximity as well as assess the stiffness, helical and superhelical ordering based on polymer connectivity. The proposed software is aimed at balancing flexibility and computational efficiency. The quantitative order parameters can be useful to quantify various types of self-organization observed in coarse-grained as well as all-atom particle simulations. The utilities can be tailored to meet specific user requirements.

Keywords: molecular simulation, polymers, helicity, simulation data processing, order parameters, performance optimization.

Introduction

In recent decades, particle-based computer simulations have become a vital tool to advance scientific knowledge and get valuable insights into the behavior of complex systems [18]. Simulations can be used for high-throughput screening, to check the influence of conditions that cannot be easily achieved experimentally, and are also important for validation of the models.

As most of the considered systems are chaotic [64], meaningful results cannot be obtained only from the visual analysis of the snapshots, but require considering a statistically representative ensemble of configurations [17]. Therefore, the data from particle simulations has to be quantified, i.e. reduced to a relatively small set of comprehensible parameters.

Molecular ordering and self-assembly are the focus of computer simulations due to their great importance to many areas of research, including physics, chemistry, biology and medicine, and because computer simulations are the perfect tool to explore the relationships between molecular-level phenomena and practically important behavior of the systems.

Analysis of the results of particle simulations is implemented in multiple software tools, including those with open-source code. For example, one can use the algorithms included in popular simulation engines such as LAMMPS [57] or GROMACS [1] to calculate the required quantities in the course of the simulation or process the trajectories afterwards. In the case of post-processing, the range of possible tools is wider and includes multiple toolkits and libraries. Among the latter are MDAnalysis [45], VMD [37], Travis [10], PyLAT [36], Pteros [62] and MDTraj [44]. However, in many cases, the data analysis is performed by the unpublished in-house code, which hinders the reproducibility of the results.

In this manuscript we share a set of software tools which were used to describe the behavior of various complex polymer systems [2, 19–27], mostly representing the class of so-called conformationally asymmetric polymers. The parameters calculated by our tools exploit the ability of

[1] A. N. Nesmeyanov Institute for Organoelement Compounds RAS, Moscow, Russian Federation
[2] Chemistry Department, M. V. Lomonosov Moscow State University, Moscow, Russian Federation

computer simulations to directly access the structure of the systems. This way, one can quantify the patterns that can be observed only in computer simulations and explore the relationship between the parameters of the system, its molecular structuring and its experimentally observed properties. This can reveal underlying processes responsible for complex phenomena observed in polymer systems.

The order parameters which are calculated by the proposed tools are independent of the atomistic details of the system and capture its general behavior. To analyze the results of detailed (e.g. atomistic) simulations, one can either select specific particles to represent the spatial positions of monomer units or apply a systematic coarse-graining procedure, i.e. consider the centers of mass of the units as their positions. The latter option can be implemented through available open-source coarse-graining software, for example, the VOTCA toolkit [54], or done within the MDAnalysis framework.

The order parameters are also designed to work without prior knowledge about the ordering in the system. For example, the estimation of helical parameters based on the autocorrelation function does not depend on the orientation of the helical fragments. The local ordering parameter and lamellar ordering parameter also do not depend on the direction of ordering in the system. However, some parameters, such as the backbone correlation distance or the local ordering cutoff range, are left to the user's discretion. It is worth noting that these spatial parameters can be efficiently employed to assess the scale of the spatial correlation in the systems and can be an important tool in assessing the soundness of the results in terms of size-related artifacts [17].

The utilities are written in Python3 [58], which is an easy-to-learn and flexible interpreted programming language. To make our code convenient for a wide range of researchers, we have refactored it and moved to use the popular and actively developed MDAnalysis library [45] to process the data input, providing an interface to input the data from many simulation packages, including LAMMPS, GROMACS, AMBER, HOOMD, Tinker, as well as data in the standard PDB format. The format of the data is recognized automatically by MDAnalysis based on filename extensions. The MDAnalysis also provides intrinsic support for processing of the time series. For the analysis of aggregation, we use the NetworkX library [32]. The input parameters are processed by the Argparse library. The output is provided in standard Python dictionary structures with a short description and results for all of the timesteps. It can be printed out in JSON [52] data format, to be assessed directly or processed further. In some of the utilities, Matplotlib [38] is used to plot the results.

Computing local structural properties is challenging because they scale as $O(N^2)$, where the number of particles N can be as large as 10^5–10^6 [4]. While some of the calculation-intensive algorithms could be reasonably implemented only using NumPy [34], we tried to optimize the code and use NumPy-based vectorisation wherever possible. For example, to calculate the list of neighbors for a selected atom, the distances between the atom and all of the other atoms are calculated, with the values exceeding the cutoff masked using the numpy.ma module. The indices of the atoms that do not satisfy the neighbor criteria are then filtered out using the numpy.ma.compress function. Similar approach, implying vectorized computations with masking out the irrelevant values was used to calculate the bond autocorrelation function as well as the superhelical twist parameters.

In the following sections, we describe the individual algorithms and their applicability to structural analysis, and in the end, we describe the performance optimization techniques employed.

1. Algorithms

1.1. Backbone Bonds Autocorrelation and Helicity

Helix is one of the main ways of polymer packing which is widespread in synthetic and biological macromolecules [15, 39, 48]. The helical packing combines the connectivity of the monomer units in the macromolecule with a high density [48]. The helical symmetry can be determined by optimization of the contacts between the polymer chains [11, 39], by steric limitations on the dihedral angles between the neighboring units along the backbone [20], and can emerge spontaneously [13, 16]. Many pure synthetic polymers, such as polylactic acid, crystallize in a helical form, which determines their physical and chemical properties. In biological macromolecules, the helical secondary structure serves as a basis for self-organization and determines the structuring at the higher levels [29, 30, 35, 43, 60]. During the protein folding, the formation of the local structure, including the helical motifs, provides the entrance to the so-called “folding funnel”, immensely reducing the entropy of the macromolecule [3, 6, 7]. It was shown that local helical structure of a macromolecule can significantly affect the probability of knot formation [63].

The analysis of the secondary helical structures in the soft matter is complicated due to the local nature of the correlations. Therefore, the algorithm must be agnostic in terms of the orientation of the helical fragments, and the length of the analyzed correlations could be limited.

In [21] we have proposed the autocorrelation function of the backbone bonds to quantify the local helical ordering in the macromolecules:

$$C\left(k\right)=\frac{1}{N-k}\sum_{i=1}^{N-k}\left|\left(\boldsymbol{d}_i\boldsymbol{d}_{i+k}\right)\right|, \tag{1}$$

where $\boldsymbol{d}_i$ designates the normalized vector, connecting the atoms of the i-th backbone bond, and N is the length of the backbone. From the analysis of the resulting autocorrelation functions, it is possible to determine the actual number of monomer units per helix turn and to evaluate the stability of the structure quantitatively.

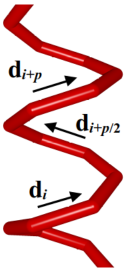


Figure 1. The bond vectors in a helix with an even number of monomer units per turn

For a perfect macromolecular helix with p monomer units per helix turn, bond length b, and pitch d, the $C(k)$ would depend on those values according to the following relationship:

$$C\left(k\right)=\left(1-d^2/b^2p^2\right)\cos\left(2\pi k/p\right)+d^2/b^2p^2. \tag{2}$$

To evaluate the spatial parameters of the helical structure and its stability, one can fit $C(k)$ with a function in the form

$$y\left(x\right)=\left(A\cdot\cos\left(2\pi x/p\right)+B\right)\cdot e^{-\beta x} \tag{3}$$

to obtain the number of monomer units per turn and the persistence length $1/\beta$. It is worth noting that the amplitude of the oscillations and the shift correspond to the squared cosine and sine of the helix angle respectively:

$$1 - d^2/b^2p^2 = cos^2(\alpha), \tag{4}$$

$$d^2/b^2p^2 = sin^2(\alpha). \tag{5}$$

The decay parameter β can be used as a measure of the stability of the helical structure. Alternatively, one can use the value of $C(p)$, which corresponds to correlations between the backbone at the neighboring turns of the helix. The observed values can also be compared to the perfect crystalline form of the polymer, as was done in [28] for polylactic acid.

To use the implementation, the user can call the bond_autocorrelations.py tool, provide the input data file, the maximum value of k, and specify whether the correlations between the bonds connecting the beads with different residue IDs shall be taken into account. With the --plot option the graph is shown, and with the --fit option, the curve is fitted with the SciPy [59] curve_fit function, and the values of p, B and β from (3) are calculated.

1.2. Local Bond Orientations

Due to entanglement, the polymers can crystallize only partially, forming separate crystalline domains with different orientations. The degree of crystallinity together with the size of the domains and their mutual orientation can greatly affect the thermal, mechanical and barrier properties of the polymers.

In many biological systems, a complex local arrangement of the polymer chains is observed, when the backbones align at a certain non-zero angle to each other, forming a helix-coil structure or a helical multiplet. Such an arrangement can be caused by steric reasons [55], solvophobic [12] or electrostatic [40] interactions. Through these mechanisms, which make the basis of the so-called “chirality transfer”, the local helicity can cause the twisting of the polymer filaments [50, 61], which, in its turn, can limit the growth of the bundles composed of such filaments [30, 61].

Therefore, studying the ordering effects at the molecular level can provide valuable insights. To establish the validity of computer simulations [5, 17] it is important to compare the characteristic scale of the ordering with the size of the simulation cell.

To evaluate the local ordering, we have implemented the algorithm to calculate the angles χ between the bonds, when the distance between the midpoints of the bonds lies in the interval between the user-defined minimum r_{min} and maximum r_{max} distances.

To avoid the complexity, we suggest using the bonds in the molecular dataset as the vectors. In the case of polymers, these can be the vectors connecting the positions of 1-st or higher-order neighbors along the polymer chain. To analyze the macromolecules which have a helical structure, one can determine the vectors as the ones connecting the i-th and $i+p$-th monomer units, where p is the number of monomer units per helix turn, and p can be determined from bond autocorrelation analysis as described above. Alternatively, the ordering of helical macromolecules can be studied using the vectors, connecting the centers of mass of each p monomer units along the polymer backbone.

The values can then be used to calculate the 2-nd Legendre polynomial:

$$S = \frac{3 < cos^2\chi > -1}{2} \tag{6}$$

which can serve as the order parameter. To calculate the spatial scale of ordering, one can check the dependence of the value of the parameter on the cutoff radius. The minimum reasonable value of the cutoff radius is determined by the density of the system and can be of the order of the first coordination sphere. To avoid the finite-size effects, the maximum value must not exceed half the size of the simulation cell. The values of the angles between the neighboring bonds and helical segments were used in [23] to investigate the mechanically induced ordering of polylactic acid, and in [2] to study the structuring of polymer globules.

To use the implementation, the user can call the local_alignment.py tool, and provide the input data files, the minimum (--r_min) and maximum (--r_max) distance between the midpoints of the vectors which will be taken into account. With the --histogram option, the distributions of the parameters in terms of $cos^2\chi$, $cos\chi$ and χ, will be calculated. The distributions for $cos^2\chi$, normalized by the total area and by the solid angle (so that the distribution is uniform for the isotropic case) can be plotted by using the --plot option. Studying the distributions of χ can be insightful for many biological and biomimetic systems, where the neighboring helical segments prefer to align at a certain angle to each other [55].

1.3. Orientation of Copolymer Blocks in Lamellae

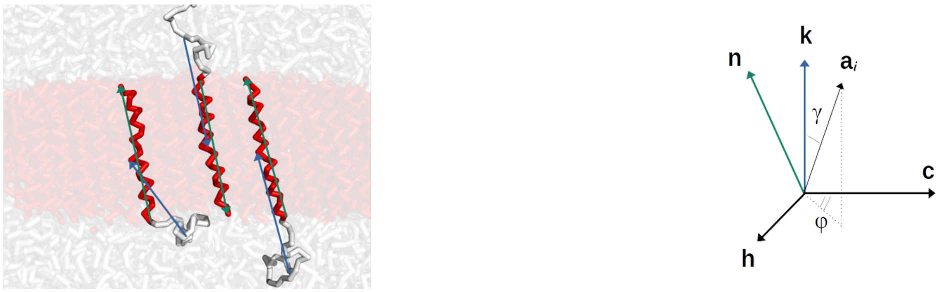
The procedure described in this section is applied to the systems of diblock copolymers with a stiff block which undergoes microphase separation into the lamellar phase. Microphase separation is an important phenomenon which occurs in melts and concentrated solutions of block copolymers with incompatible blocks and results in the formation of micro- and nanodomains of various shapes [8, 33, 42]. This phenomenon finds its practical application in a wide range of nanostructured materials where the domains are endowed with predefined properties: those can be electronic or ionic conductivity, optical properties, etc. [41, 53].

Block copolymers containing a stiff block and a flexible block demonstrate the properties of both flexible coil-coil diblock copolymers and liquid crystals. The self-assembly of such block copolymers results in novel unique morphologies distinct from the classic morphologies formed by coil-coil diblock copolymers [14, 47] and can include various forms of lamellae. In particular, it was found that for block copolymers with stiff blocks the latter can be smectically or nematically ordered [46]. However, the orientation of the blocks is meaningful in relation to the lamellar plane. To calculate the orientational order parameters of the stiff blocks relative to the normal vector of the lamellae plane, we have implemented the algorithm described in [49] and used it to study the ordering of helix-coil block-copolymers [19].

First, it should be taken into account that the lamellae can take random orientation in the simulation cell, so the procedure of determining the vector normal to the lamellar planes is applied. For each polymer chain, the vectors connecting the centers of mass of incompatible blocks are calculated, normalized to have a unit length and placed to the origin of coordinates. The resulting unit vectors are denoted as $\boldsymbol{e}_i = (e_{ix}, e_{iy}, e_{iz})(|\boldsymbol{e}_i| = 1)$. We find the gyration tensor of those vectors as follows:

$$I_{\alpha\beta} = \frac{1}{n_{ch} * N_{stiff}} \sum_{i=1}^{n_{ch} * N_{stiff}} \boldsymbol{e}_{i\alpha} \boldsymbol{e}_{i\beta}. \tag{7}$$

In this equation, n_{ch} is the number of polymer chains in the system, N_{stiff} is the number of units in the stiff block. The eigenvalues and corresponding eigenvectors are calculated for this tensor. The lamellar normal vector $\boldsymbol{k}$ is denoted as the eigenvector which corresponds to the largest eigenvalue.



(a) Schematic representation of several molecules of the helix-coil diblock copolymer within the lamella, obtained in [19]

(b) Coordinate system related to lamellar domains as proposed in [49]

Figure 2. Schematic representation of the lamellar domain and the corresponding coordinate system. Blue arrows are the vectors connecting the centers of mass of the coil and helix block (in Fig. 2a) and the derived lamellar normal (in Fig. 2b). Green arrows are the vectors connecting the ends of the helix block (in Fig. 2a) and the director of these blocks (in Fig. 2b)

The orientational order parameter L_k is introduced as the average orientation of the stiff blocks with respect to the lamellar normal:

$$L_k = \left\langle \frac{3}{2} (\boldsymbol{a} \cdot \boldsymbol{k})^2 - \frac{1}{2} \right\rangle . \tag{8}$$

Here, $\boldsymbol{a}$ is the unit vector showing the direction of each stiff block, i.e. the vector between the stiff block ends, and the averaging is performed over all the polymer chains.

The modulus of L_k tends to zero if the orientations of the stiff blocks are not correlated and is close to unity when the stiff blocks orient perpendicularly to the lamellae surface.

We also introduce the director $\boldsymbol{n}$ of all the stiff blocks. It is defined via the same procedure as $\boldsymbol{k}$, using the vectors connecting the ends of the stiff blocks $\boldsymbol{a}_i$ normalized to the unity length ($|\boldsymbol{a}_i| = 1$) instead of $\boldsymbol{e}_i$ in (7). The plane comprising the vectors $\boldsymbol{n}$ and $\boldsymbol{k}$ is denoted as a tilt plane. We calculate the tilt angle θ of the stiff blocks using three order parameters:

$$\tan 2\theta = \frac{V}{L_k - 0.5 P_k}, \tag{9}$$

$$P_k = \left\langle \sin^2 \gamma \cos 2\varphi \right\rangle , \tag{10}$$

$$V = \left\langle \sin 2\gamma \cos \varphi \right\rangle . \tag{11}$$

In (10) and (11), γ is the angle between the lamellar normal $\boldsymbol{k}$ and $\boldsymbol{a}_i$; φ is the angle between the vector $\boldsymbol{c}$, which belongs to the tilt plane and is perpendicular to the lamellar normal ($\boldsymbol{c} \perp \boldsymbol{k}$), and the projection of the vector $\boldsymbol{a}_i$ on the plane formed by the vector $\boldsymbol{c}$ and the normal to the tilt plane $\boldsymbol{h}$ (Fig. 2 and ref. [49]).

The parameter P_k describes the biaxial distribution of the stiff blocks within the lamellae; V is a coefficient in the non-diagonal term of the tensor order parameter of the system [49] and describes the tilt of its main axis with respect to the lamellar normal. The algorithm is implemented in lamellar_alignment.py tool. The user should provide an input data file and, optionally, the names of the block types.

The output of the calculation includes the order parameter L_k which is the measure of the correlation between the directions of the stiff blocks with each other and with the lamellar plane and the tilt angle θ of the stiff blocks.

1.4. Superhelical Twist

The formation of superhelical structures is common in the self-organization of biological macromolecules. The local helical structure of polymer chains can lead to a preferential alignment angle between the neighboring polymer chains, as discussed in section 1.2. In helical multiplets, this causes the twisting of the helical polymer chains around each other. The spatial dimensions of this type of structure depend on the spatial parameters of individual helices [31]. To assess the handedness of these superhelices, one can connect uniformly points of the helical tube with the vectors and calculate the dihedral angles formed by these vectors in the range $(-\pi; \pi]$. If the length of the vectors is commensurate to the period of the superhelix, the dihedral angles will have preferentially negative values, corresponding to the left-hand superhelices, or positive values, corresponding to right-hand superhelices [2]. The algorithm quantitatively supported the visual observations reported in [2], and the results were presented in the Supporting Information of [2].

For a set of the integer values of k, provided by the user, the algorithm calculates the dihedral angles between the triplets of consecutive vectors connecting the monomer units of the backbone: (r_i, r_{i+k}), (r_{i+k}, r_{i+2k}), (r_{i+2k}, r_{i+3k}), where r_j is the position of the j-th monomer unit. By default, the dihedral angle values are calculated using those values of i, where the start points and end points of all three vectors belong to the same molecule.

To calculate the values of the described dihedral angles, the user can call the backbone_twist.py utility, and provide the list of the values of k, and the criteria for backbone atom selection in terms of MDAnalysis select_atoms function. With the --plot option, the distributions for all of the values of k will be plotted using the NumPy histogram function and PyPlot.

1.5. Aggregation

The algorithm calculates the list of the aggregates at each timestep. The atom is defined as belonging to a particular aggregate if it has at least one neighboring atom belonging to the same aggregate.

At a given timestep, a list of neighbors for each atom is created. The atoms are considered neighbors if they lie within the cutoff distance from each other. The user can switch between the distance matrix calculation function of the MDAnalysis library or our algorithm based on NumPy (see the Performance Optimization section). The algorithm then loops through the atoms and for each atom loops through its neighbors, adding the pairs of the atoms and their neighbors as the edges of the NetworkX graph object [32].

The iterative deepening depth-first search algorithm is then applied to the graph [56]. The atom indices returned by this search are then stored as an individual aggregate, and removed from the list of all atom indices. The procedure is repeated until the list of all atom indices is empty.

The function can be called using aggregates.py tool with the data file and the cutoff distance as input. The output contains lists of the aggregates in a form of the lists of atoms in each aggregate. Such lists are generated for each timestep.

2. Performance Optimization

The performance of the toolkit is achieved by employing NumPy vectorization for computationally intensive operations. For example, to calculate the list of neighbors for a selected atom, the distances between the atom and all of the other atoms are calculated. The resulting squared distances are then compared to the maximum distance between the neighbors, and the mask for the *numpy.ma* module is generated. The indices of the atoms that do not satisfy the neighbor criteria are then filtered out using the *numpy.ma.compress* function.

To vectorize the calculation of the bond autocorrelation functions, two arrays of bond vector coordinates are calculated, which are shifted by the autocorrelation distance k:

```
b1 = np.pad(b, ((0, k), (0, 0)), constant_values = 1.)
b2 = np.pad(b, ((k, 0), (0, 0)), constant_values = 1.)
```

The residue identifiers, associated with the bonds, are then also padded, to check whether the result is meaningful, and the autocorrelation function is calculated for all of the bonds:

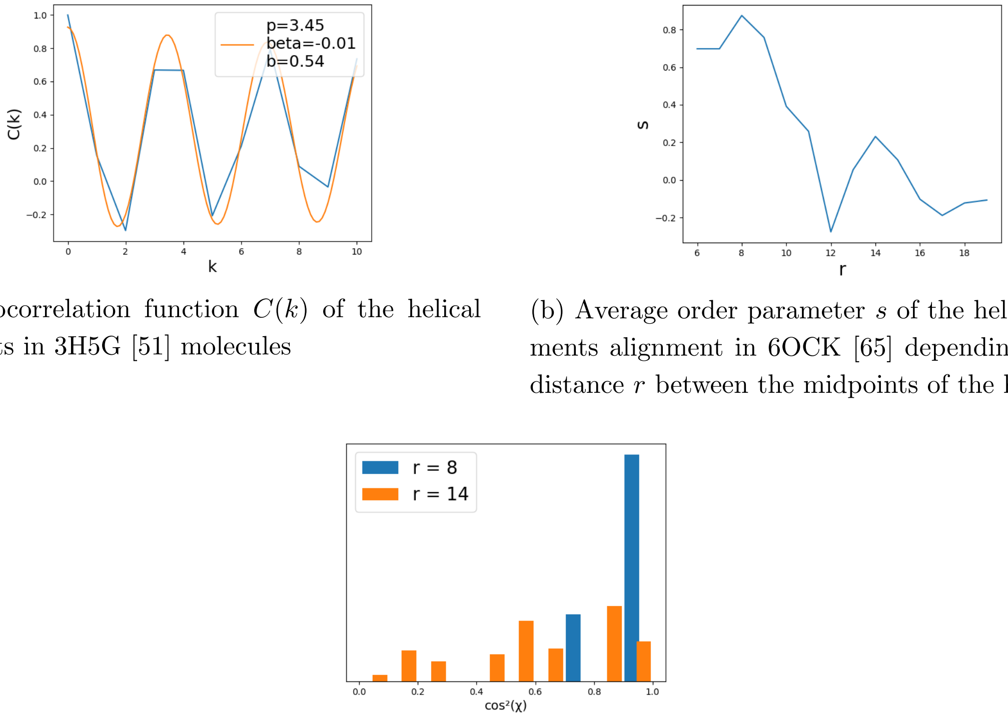
```
if not different_molecules:
        # Pad the residue id arrays
        resid1 = np.pad(bond_resids, (0, k), constant_values = 0)
        resid2 = np.pad(bond_resids, (k, 0), constant_values = 0)
        # Take into account only molecules with same residue id
        valid = np.logical_and(valid, np.equal(resid1, resid2))
# Mask is True for the values that are not valid
mask = np.logical_not(valid)
# Calculate the correlation values for all the bonds
c = (np.sum(np.multiply(b1, b2), axis = 1)
        / np.linalg.norm(b1, axis = 1)
        / np.linalg.norm(b2, axis = 1))
```

To calculate the dihedral angles associated with the superhelical twist, a similar scheme is performed first to calculate the vectors forming the dihedral angles, and then the values of the dihedral angles themselves.

3. Usage Examples

The examples directory of the project repository currently contains two Python scripts, which demonstrate using some of the functions. The systems for analysis are taken from the RSCB Protein Data Bank (PDB) [9]. We use the bond_autocorrelations.py tool to detect the parameters of the helices in a de novo protein with the PDB identifier 3H5G [51]. The script invokes the bond_autocorrelations.py command through the *os.system*() call, corresponding to the command line usage scenario. The bond_autocorrelations.py finds the average number of residues per helix turn as $p = 3.45$, which is close to the values for the α-helix, and the positive shift of the autocorrelation function, which depends on the elongation of the helix, as $b = 0.54$.

The very low value of the decay parameter $\beta = -0.01$ is indicative of a well-defined helical structure, which is consistent with the authors' conclusions about its stability.



(a) Autocorrelation function $C(k)$ of the helical fragments in 3H5G [51] molecules

(b) Average order parameter s of the helical fragments alignment in 6OCK [65] depending on the distance r between the midpoints of the helices

(c) Distributions of the angles between the helical fragments in 6OCK [65] at $r = 8$ and $r = 14$

Figure 3. Results for processing example proteins with helical structures from the RSCB PDB [9]

For the second protein from the PDB, with the identifier 6OCK [65], the example script invokes the local_alignment.py function, corresponding to the scenario where the user can build her/his own tools upon the functions provided by our package. The script calculates the average order parameter s, corresponding to the alignment of the helices, depending on the distance between their midpoints. The helices are taken according to the information provided in the original data bank record, and only those helices that are equal or longer than 10 residues are taken into account.

The distributions of the angles between the helices are then calculated for the main maximum at $r = 8$ and a small maximum at $r = 14$. The plots show that the helices located closely and belonging to the same domain are mostly aligned with each other, while at larger distances there is a wide distribution of mutual orientations of the helices, which can result in fluctuations of the average local alignment parameter s, leading to local extrema.

4. Availability and System Requirements

The tools described above can be downloaded free of charge, used and modified according to the conditions of the standard GNU GPL v.3 license. To use the software, the user needs to download it to her/his computer. Currently we provide the option to download the software from the main GitHub repository, or, alternatively to use the PyPI package. In the latter case the pip package manager automatically installs the libraries and packages upon which our software is built: MDAnalysis, NumPy, NetworkX, SciPy, Matplotlib. The instructions on how to install the software and a short user guide are available in the README.md file, and are displayed at the

GitHub page of the project. The unit tests for the algorithms are provided with the package and are located in the tests directory of the GitHub repository. We have developed, tested and used the software with the Linux platforms used by our group, and have not tested the installation and functioning of the software on other platforms. In case of the Windows operating system, the installation shall be possible using the PyPI and/or Conda package managers, and we will be happy to cooperate with the users to make the software work on other platforms.

Conclusions

Our package offers several tools for the analysis of molecular dynamics simulations and is available for download from GitHub under the GNU General Public License, and from PyPI. With the help of MDAnalysis library, it can process the output of multiple simulation software. The algorithms were optimized for performance and with the modern hardware can process the systems containing up to 10^6 particles.

Although it is impossible to implement all of the usage scenarios, the tools can be extended and the functions can be called from the Python code. We are open to cooperation and will consider implementing other usage scenarios.

We are planning to build on these utilities and add more functionality in the course of our future research.

Acknowledgements

This work was supported by the Ministry of Science and Higher Education of the Russian Federation (Contract No. 075-03-2023-642).

References

1. Abraham, M.J., Murtola, T., Schulz, R., *et al.*: GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX 1-2, 19–25 (Sep 2015). https://doi.org/10.1016/j.softx.2015.06.001

2. Abramova, A., Glagolev, M., Vasilevskaya, V.: Structured globules with twisted arrangement of helical blocks: Computer simulation. Polymer 253, 124974 (Jun 2022). https://doi.org/10.1016/j.polymer.2022.124974

3. Adamcik, J., Mezzenga, R.: Amyloid Polymorphism in the Protein Folding and Aggregation Energy Landscape. Angewandte Chemie International Edition 57(28), 8370–8382 (Jul 2018). https://doi.org/10.1002/anie.201713416

4. Allen, M.P., Tildesley, D.J.: Computer simulation of liquids. Oxford University Press, Oxford, United Kingdom, second edition edn. (2017)

5. Arora, A., Morse, D.C., Bates, F.S., Dorfman, K.D.: Commensurability and finite size effects in lattice simulations of diblock copolymers. Soft Matter 11(24), 4862–4867 (2015). https://doi.org/10.1039/C5SM00838G

6. Baldwin, R.L., Rose, G.D.: Is protein folding hierarchic? II. Folding intermediates and transition states. Trends in Biochemical Sciences 24(2), 77–83 (Feb 1999). https://doi.org/10.1016/S0968-0004(98)01345-0

7. Baldwin, R.L., Rose, G.D.: Is protein folding hierarchic? II. Folding intermediates and transition states. Trends in Biochemical Sciences 24(2), 77–83 (Feb 1999). https://doi.org/10.1016/S0968-0004(98)01345-0

8. Bates, F.S., Fredrickson, G.H.: Block Copolymer Thermodynamics: Theory and Experiment. Annual Review of Physical Chemistry 41(1), 525–557 (Oct 1990). https://doi.org/10.1146/annurev.pc.41.100190.002521

9. Berman, H.M., Westbrook, J., Feng, Z., *et al.*: The Protein Data Bank. Nucleic Acids Research 28(1), 235–242 (01 2000). https://doi.org/10.1093/nar/28.1.235

10. Brehm, M., Thomas, M., Gehrke, S., Kirchner, B.: TRAVIS—A free analyzer for trajectories from molecular simulation. The Journal of Chemical Physics 152(16), 164105 (Apr 2020). https://doi.org/10.1063/5.0005078

11. Cajamarca, L., Grason, G.M.: Geometry of flexible filament cohesion: Better contact through twist? The Journal of Chemical Physics 141(17), 174904 (Nov 2014). https://doi.org/10.1063/1.4900983

12. Cejas, M.A., Kinney, W.A., Chen, C., *et al.*: Thrombogenic collagen-mimetic peptides: Self-assembly of triple helix-based fibrils driven by hydrophobic interactions. Proceedings of the National Academy of Sciences 105(25), 8513–8518 (Jun 2008). https://doi.org/10.1073/pnas.0800291105

13. Chaudhuri, D., Mulder, B.M.: Spontaneous Helicity of a Polymer with Side Loops Confined to a Cylinder. Physical Review Letters 108(26), 268305 (Jun 2012). https://doi.org/10.1103/PhysRevLett.108.268305

14. Chen, J.T., Thomas, E.L., Ober, C.K., Mao, G.p.: Self-Assembled Smectic Phases in Rod-Coil Block Copolymers. Science 273(5273), 343–346 (Jul 1996). https://doi.org/10.1126/science.273.5273.343

15. Claessens, M.M.A.E., Semmrich, C., Ramos, L., Bausch, A.R.: Helical twist controls the thickness of F-actin bundles. Proceedings of the National Academy of Sciences 105(26), 8819–8822 (Jul 2008). https://doi.org/10.1073/pnas.0711149105

16. Ermilov, V.A., Vasilevskaya, V.V., Khokhlov, A.R.: Secondary globular structure of copolymers containing amphiphilic and hydrophilic units: Computer simulation analysis. Polymer Science Series A 49(1), 89–96 (Jan 2007). https://doi.org/10.1134/S0965545X07010129

17. Frenkel, D.: Simulations: The dark side. The European Physical Journal Plus 128(1), 10 (Jan 2013). https://doi.org/10.1140/epjp/i2013-13010-8

18. Gartner, T.E., Jayaraman, A.: Modeling and Simulations of Polymers: A Roadmap. Macromolecules 52(3), 755–786 (Feb 2019). https://doi.org/10.1021/acs.macromol.8b01836

19. Glagolev, M.K., Glagoleva, A.A., Vasilevskaya, V.V.: Microphase separation in helix-coil block copolymer melts: computer simulation. Soft Matter 17(36), 8331–8342 (2021). https://doi.org/10.1039/D1SM00759A

20. Glagolev, M.K., Vasilevskaya, V.V.: Liquid-Crystalline Ordering of Filaments Formed by Bidisperse Amphiphilic Macromolecules. Polymer Science, Series C 60(1), 39–47 (Sep 2018). https://doi.org/10.1134/S1811238218010046

21. Glagolev, M.K., Vasilevskaya, V.V., Khokhlov, A.R.: Compactization of rigid-chain amphiphilic macromolecules with local helical structure. Polymer Science Series A 52(7), 761–774 (Jul 2010). https://doi.org/10.1134/S0965545X10070102

22. Glagolev, M.K., Vasilevskaya, V.V., Khokhlov, A.R.: Formation of fibrillar aggregates in concentrated solutions of rigid-chain amphiphilic macromolecules with fixed torsion and bend angles. Polymer Science Series A 53(8), 733–743 (Aug 2011). https://doi.org/10.1134/S0965545X11080037

23. Glagolev, M.K., Vasilevskaya, V.V.: Coarse-grained simulation of molecular ordering in polylactic blends under uniaxial strain. Polymer 190, 122232 (Mar 2020). https://doi.org/10.1016/j.polymer.2020.122232

24. Glagolev, M.K., Vasilevskaya, V.V., Khokhlov, A.R.: Effect of Induced Self-Organization in Mixtures of Amphiphilic Macromolecules with Different Stiffness. Macromolecules 48(11), 3767–3774 (Jun 2015). https://doi.org/10.1021/acs.macromol.5b00188

25. Glagolev, M.K., Vasilevskaya, V.V., Khokhlov, A.R.: Induced liquid-crystalline ordering in solutions of stiff and flexible amphiphilic macromolecules: Effect of mixture composition. The Journal of Chemical Physics 145(4), 044904 (Jul 2016). https://doi.org/10.1063/1.4959861

26. Glagolev, M.K., Vasilevskaya, V.V., Khokhlov, A.R.: Domains in mixtures of amphiphilic macromolecules with different stiffness of backbone. Polymer 125, 234–240 (Sep 2017). https://doi.org/10.1016/j.polymer.2017.08.009

27. Glagolev, M. K., Vasilevskaya, V.V., Khokhlov, A.R.: Self-organization of amphiphilic macromolecules with local helix structure in concentrated solutions. The Journal of Chemical Physics 137(8), 084901 (Aug 2012). https://doi.org/10.1063/1.4745480

28. Glova, A.D., Melnikova, S.D., Mercurieva, A.A., *et al.*: Grafting-Induced Structural Ordering of Lactide Chains. Polymers 11(12), 2056 (Dec 2019). https://doi.org/10.3390/polym11122056

29. Grason, G.M.: Chirality Transfer in Block Copolymer Melts: Emerging Concepts. ACS Macro Letters 4(5), 526–532 (May 2015). https://doi.org/10.1021/acsmacrolett.5b00131

30. Grason, G.M.: Chiral and achiral mechanisms of self-limiting assembly of twisted bundles. Soft Matter 16(4), 1102–1116 (2020). https://doi.org/10.1039/C9SM01840A

31. Grosberg, A.Y.: Theory of the cholesteric mesophase in a solution of chiral macromolecules. Soviet Physics Doklady 25, 638 (1980)

32. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring Network Structure, Dynamics, and Function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference. pp. 11–15. Pasadena, CA, USA (2008)

33. Hajduk, D.A., Harper, P.E., Gruner, S.M., *et al.*: The Gyroid: A New Equilibrium Morphology in Weakly Segregated Diblock Copolymers. Macromolecules 27(15), 4063–4075 (Jul 1994). https://doi.org/10.1021/ma00093a006

34. Harris, C.R., Millman, K.J., van der Walt, S.J., *et al.*: Array programming with NumPy. Nature 585(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2

35. Ho, R.M., Li, M.C., Lin, S.C., *et al.*: Transfer of Chirality from Molecule to Phase in Self-Assembled Chiral Block Copolymers. Journal of the American Chemical Society 134(26), 10974–10986 (Jul 2012). https://doi.org/10.1021/ja303513f

36. Humbert, M.T., Zhang, Y., Maginn, E.J.: PyLAT: Python LAMMPS Analysis Tools. Journal of Chemical Information and Modeling 59(4), 1301–1305 (Apr 2019). https://doi.org/10.1021/acs.jcim.9b00066

37. Humphrey, W., Dalke, A., Schulten, K.: VMD: Visual molecular dynamics. Journal of Molecular Graphics 14(1), 33–38 (Feb 1996). https://doi.org/10.1016/0263-7855(96)00018-5

38. Hunter, J.D.: Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering 9(3), 90–95 (2007). https://doi.org/10.1109/MCSE.2007.55

39. Ji, X.Y., Zhao, M.Q., Wei, F., Feng, X.Q.: Spontaneous formation of double helical structure due to interfacial adhesion. Applied Physics Letters 100(26), 263104 (Jun 2012). https://doi.org/10.1063/1.4731199

40. Kornyshev, A.A., Lee, D.J., Leikin, S., Wynveen, A.: Structure and interactions of biological helices. Reviews of Modern Physics 79(3), 943–996 (Aug 2007). https://doi.org/10.1103/RevModPhys.79.943

41. Lazzari, M., Liu, G., Lecommandoux, S. (eds.): Block copolymers in nanoscience. Wiley-VCH ; John Wiley [distributor], Weinheim : Chichester (2006), oCLC: ocm69486595

42. Leibler, L.: Theory of Microphase Separation in Block Copolymers. Macromolecules 13(6), 1602–1617 (Nov 1980). https://doi.org/10.1021/ma60078a047

43. Li, M.C., Ousaka, N., Wang, H.F., *et al.*: Chirality Control and Its Memory at Microphase-Separated Interface of Self-Assembled Chiral Block Copolymers for Nanostructured Chiral Materials. ACS Macro Letters 6(9), 980–986 (Sep 2017). https://doi.org/10.1021/acsmacrolett.7b00493

44. McGibbon, R., Beauchamp, K., Harrigan, M., *et al.*: MDTraj: A Modern Open Library for the Analysis of Molecular Dynamics Trajectories. Biophysical Journal 109(8), 1528–1532 (Oct 2015). https://doi.org/10.1016/j.bpj.2015.08.015

45. Michaud-Agrawal, N., Denning, E.J., Woolf, T.B., Beckstein, O.: MDAnalysis: A toolkit for the analysis of molecular dynamics simulations. Journal of Computational Chemistry 32(10), 2319–2327 (Jul 2011). https://doi.org/10.1002/jcc.21787

46. Olsen, B., Segalman, R.: Self-assembly of rod-coil block copolymers. Materials Science and Engineering: R: Reports 62(2), 37–66 (Jul 2008). https://doi.org/10.1016/j.mser.2008.04.001

47. Olsen, B.D., Segalman, R.A.: Nonlamellar Phases in Asymmetric Rod-Coil Block Copolymers at Increased Segregation Strengths. Macromolecules 40(19), 6922–6929 (Sep 2007). https://doi.org/10.1021/ma070976x

48. Olsen, K., Bohr, J.: The generic geometry of helices and their close-packed structures. Theoretical Chemistry Accounts 125(3-6), 207–215 (Mar 2010). https://doi.org/10.1007/s00214-009-0639-4

49. Osipov, M.A., Gorkunov, M.V., Berezkin, A.V., *et al.*: Molecular theory of the tilting transition and computer simulations of the tilted lamellar phase of rod-coil diblock copolymers. The Journal of Chemical Physics 152(18), 184906 (May 2020). https://doi.org/10.1063/5.0005854

50. Paavilainen, S., Rg, T., Vattulainen, I.: Analysis of Twisting of Cellulose Nanofibrils in Atomistic Molecular Dynamics Simulations. The Journal of Physical Chemistry B 115(14), 3747–3755 (Apr 2011). https://doi.org/10.1021/jp111459b

51. Peacock, A., Stuckey, J., Pecoraro, V.: Switching the chirality of the metal environment alters the coordination mode in designed peptides. Angewandte Chemie International Edition 48(40), 7371–7374 (2009). https://doi.org/10.1002/anie.200902166

52. Pezoa, F., Reutter, J.L., Suarez, F., *et al.*: Foundations of JSON Schema. In: Proceedings of the 25th International Conference on World Wide Web. pp. 263–273. International World Wide Web Conferences Steering Committee, Montral, Qubec, Canada (Apr 2016). https://doi.org/10.1145/2872427.2883029

53. Ruokolainen, J., Mkinen, R., Torkkeli, M., *et al.*: Switching Supramolecular Polymeric Materials with Multiple Length Scales. Science 280(5363), 557–560 (Apr 1998). https://doi.org/10.1126/science.280.5363.557

54. Rhle, V., Junghans, C., Lukyanov, A., *et al.*: Versatile Object-Oriented Toolkit for Coarse-Graining Applications. Journal of Chemical Theory and Computation 5(12), 3211–3223 (Dec 2009). https://doi.org/10.1021/ct900369w

55. Straley, J.P.: Theory of piezoelectricity in nematic liquid crystals, and of the cholesteric ordering. Physical Review A 14(5), 1835–1841 (Nov 1976). https://doi.org/10.1103/PhysRevA.14.1835

56. Tarjan, R.: Depth-First Search and Linear Graph Algorithms. SIAM Journal on Computing 1(2), 146–160 (Jun 1972). https://doi.org/10.1137/0201010

57. Thompson, A.P., Aktulga, H.M., Berger, R., *et al.*: LAMMPS – a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. Computer Physics Communications 271, 108171 (Feb 2022). https://doi.org/10.1016/j.cpc.2021.108171

58. Van Rossum, Guido, Drake, Fred L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA

59. Virtanen, P., Gommers, R., Oliphant, T.E., *et al.*: SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature Methods 17(3), 261–272 (Mar 2020). `https://doi.org/10.1038/s41592-019-0686-2`

60. Wang, H.F., Yang, K.C., Hsu, W.C., *et al.*: Generalizing the effects of chirality on block copolymer assembly. Proceedings of the National Academy of Sciences 116(10), 4080–4089 (Mar 2019). `https://doi.org/10.1073/pnas.1812356116`

61. Yang, Y., Meyer, R.B., Hagan, M.F.: Self-Limited Self-Assembly of Chiral Filaments. Physical Review Letters 104(25), 258102 (Jun 2010). `https://doi.org/10.1103/PhysRevLett.104.258102`

62. Yesylevskyy, S.O.: Pteros 2.0: Evolution of the fast parallel molecular analysis library for C++ and python. Journal of Computational Chemistry 36(19), 1480–1488 (Jul 2015). `https://doi.org/10.1002/jcc.23943`

63. Zhao, Y., Rothrl, J., Besenius, P., *et al.*: Can Polymer Helicity Affect Topological Chirality of Polymer Knots? ACS Macro Letters 12(2), 234–240 (Feb 2023). `https://doi.org/10.1021/acsmacrolett.2c00600`

64. Zhou, H.b., Wang, L.: Chaos in Biomolecular Dynamics. The Journal of Physical Chemistry 100(20), 8101–8105 (Jan 1996). `https://doi.org/10.1021/jp953409x`

65. Zielinski, K., Sekula, B., Bujacz, A., Szymczak, I.: Structural investigations of stereoselective profen binding by equine and leporine serum albumins. Chirality 32(3), 334–344 (2020). `https://doi.org/10.1002/chir.23162`