SwiftPol: A Python package for building and parameterizing *in silico* polymer systems

Hannah N. Turney [1] and **Micaela Matta** [1]

**1** Department of Chemistry, King's College London Strand Campus, East Wing, 33-41 Surrey St, London, WC2R 2ND, United Kingdom ROR

Summary

A polymer sample contains a natural degree of variation in its structure and non-uniformity between its chains, which influences the bulk material properties of the sample. This innate heterogeneity is often disregarded in the *in silico* study of a polymer system, resulting in divergence from experiments. This paper presents SwiftPol, a user-guided Python software for the automated generation of polydisperse polymer ensembles which reproduce the heterogeneity observed in real materials.

Statement of need

MD simulations of polymers are often performed with uniform idealized systems that do not capture the heterogeneity of their experimental counterparts. The result of this misalignment is non-convergence between MD-derived polymer properties and experimental data, and these MD simulations can overlook key components of polymer physics such as polydispersity and semi-crystallinity (Schmid, 2023; Triandafilidi et al., 2016). Studies have demonstrated that bulk polymer material properties such as glass transition temperature, hydrophobicity, and inherent viscosity are highly sensitive to variations in polydispersity, making it essential to account for this heterogeneity to capture the true physics of polymer systems (Li et al., 2016; Ochi et al., 2021; Wan et al., 2021). Polymer MD studies showcase an assortment of approaches to manually incorporate polydispersity into their polymer chain builds (Andrews et al., 2020; Kawagoe et al., 2019; Stipa et al., 2021). Although effective for their associated applications, these manual approaches are not universally applicable to different polymer chemistries or are performed using proprietary software.

Open-source software packages designed to build *in silico* polymer chains are focused on the design of polymers at the monomer and single-chain scale (Davel et al., 2024; Klein et al., 2016; Santana-Bonilla et al., 2023).

Packages that have the capability to build multi-chain systems such as Polyply (Grünewald et al., 2022), RadonPy (Hayashi et al., 2022), and Polymer Structure Predictor (Sahu et al., 2022), do not contain in-built functions to incorporate molecular weight diversity, and put the onus on the user to generate an ensemble of polydisperse chain sequences prior to using their chain building functions. Polymatic (Abbott et al., 2013), an algorithm to simulate polymerization, can theoretically create polydisperse systems through its random bond formation scheme, but does not have to ability to simultaneously control other key attributes of polymers such as blockiness and monomer ratio. The python tool Hoobas (Girard et al., 2019) builds polydisperse systems of randomized polymer chains for coarse-grained models. However, there is not currently a software package available that contains in-built functions to integrate multiple smaller-scale characteristics (monomer ratio, blockiness, degree of polymerization, chain terminal) into **atomistic** computational polymer models, whilst effectively capturing the

heterogeneity and polydispersity of real-life samples. The development of SwiftPol was driven by the need to fill this gap in multi-scale building functionality of existing polymer building packages, to enable the simulation of realistic polymer models whilst allowing close user control of multiple system characteristics.

SwiftPol uses open-source Python libraries RDKit (Landrum et al., 2024), OpenFF-interchange (Thompson et al., 2024), and OpenFF-toolkit (Wagner et al., 2024) to promote reproducibility and portability. We have ensured that SwiftPol objects can be seamlessly integrated into existing open-source software built for parameterization and simulation, to allow the user to select their preferred force field, topology format, and engine. RDKit, OpenFF-interchange and OpenFF-toolkit enable the export of SwiftPol polymer ensembles directly to simulation engines, and to a range of MD-compatible file formats, including .pdb, .top, .prmtop, and .json.

Here, we will detail the development of SwiftPol - a user-guided Python tool for building representative polymer ensembles, and subsequent studies to show its relevance and performance.
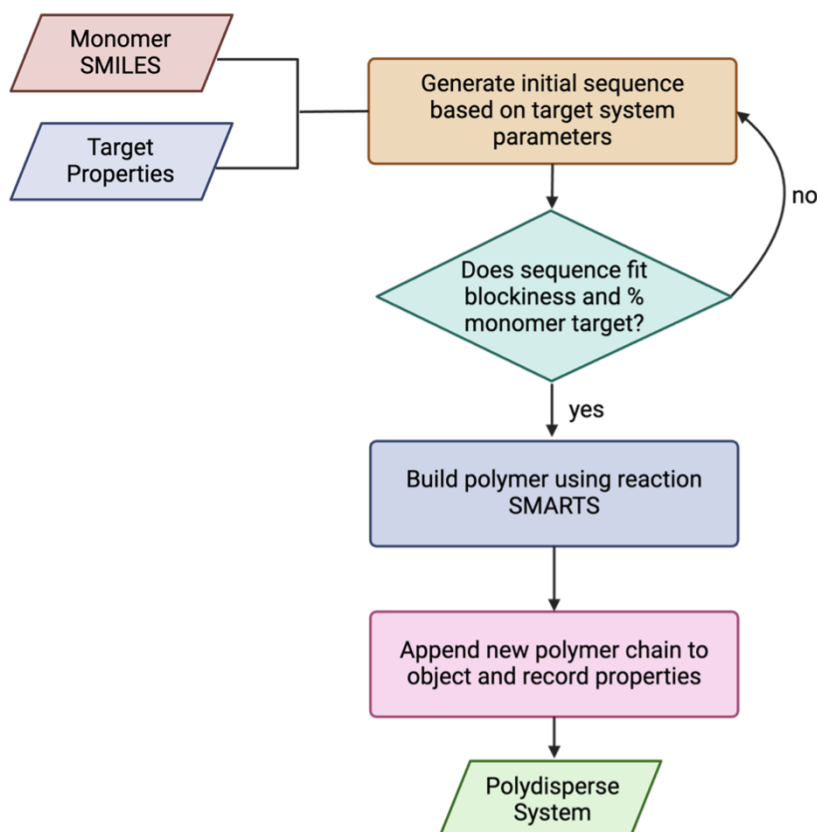
Package Overview

The SwiftPol `build` module contains Python functions to build both single polymer chains and polydisperse polymer chain ensembles.

SwiftPol takes as an input the simplified molecular-input line-entry system (SMILES)(Daylight Chemical Information Systems, Inc., 2022) string of all co-monomers, as well as values representing the target average properties of the ensemble: monomer % composition (for copolymers), length, number of chains, blockiness (for blocky copolymers), terminals, residual monomer. The user must define the reaction SMARTS (Daylight Chemical Information Systems, 2022) which describes the polymerization reaction associated with their polymer chemistry.

As depicted in Figure 1, SwiftPol generates an initial polymer chain with a chain length drawn from a normal distribution centered around the specified target length, along with a terminal group that corresponds to the chosen input. In the case of a block copolymer, a probability function is used to determine the ratio of monomers in the chain and the chain is passed to a second function which tests whether the values for blockiness and % monomer are within 10% of the input variable by default. The $+/-$ 10% acceptance margin introduces polydispersity into the ensemble by ensuring a certain level of non-uniformity between polymer chains, without straying too far from the input value. The acceptance margin can be adjusted by the user to control build stringency and polydispersity in the SwiftPol ensemble.

If all tests are passed, the chain is appended to the Python polymer ensemble build object, and the associated properties of the chain are calculated and added as ensemble attributes. Otherwise, the chain is discarded, and the process is repeated. Once the ensemble size is satisfied, average properties are calculated using built-in SwiftPol functions.

Figure 1: Flowchart showing the process of building a polymer ensemble using SwiftPol. Created in BioRender. Matta, M. (2024) https://BioRender.com/o66z317.

This approach allows for the generation of a polydisperse chain ensemble, meaning each chain displays different properties but the ensemble matches the target properties and distribution, as is observed in experimental polymer samples.

SwiftPol also contains functions to generate conformers using RDKit (Landrum et al., 2024) or OpenEye Omega(license-dependent, academic license provided by OpenEye, Cadence Molecular Sciences)(Hawkins et al., 2010; OpenEye Cadence Molecular Sciences, 2025), and assign force field parameters to the polydisperse ensembles using the openff-interchange infrastructure. The user can export the chain ensemble to existing tools such as packmol (Martínez et al., 2009) and PolyPly (Grünewald et al., 2022) to generate initial positions for molecular dynamics, the latter of which is particularly well-suited for building amorphous configurations for amorphous multi-component systems.

Application: building a poly(lactide-co-glycolide) ensemble

Using SwiftPol, we have successfully constructed polydisperse ensembles of poly(lactide-co-glycolide) (PLGA), a widely used biodegradable polymer. We used the molecular structures and properties of experimental PLGA products as input for SwiftPol building functions to create representative PLGA systems to be used for molecular dynamics simulations. By integrating experimental data, such as chain terminals, copolymer ratios of lactic and glycolic acid, and blockiness, we have been able to replicate the bulk characteristics of various commercial polymer products, namely polydispersity. A full example implementation of SwiftPol for building PLGA

systems can be found in the building a PLGA system example notebook. We used SwiftPol to build 'product X', a commercially available 75:25 LA:GA ester-terminated PLGA. Following the chain build, another SwiftPol function was used to calculate the appropriate box size for the unit cell, number of water molecules, NaCl ions, and residual monomer molecules to include in the simulation of a complete condensed polymer ensemble. The input values for the SwiftPol builder, seen in Table 1, were taken from quality assurance documents provided by the manufacturer of product X, except the value for blockiness which was measured experimentally by Sun et al (Sun et al., 2022). The system attributes assigned by SwiftPol to the completed condensed PLGA unit cell are in seen in Table 2.

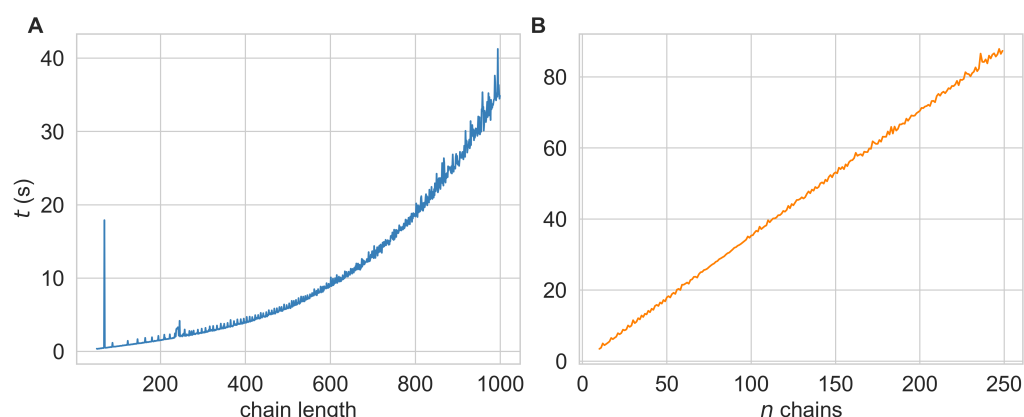**Table 1:** Input parameters for SwiftPol PLGA builder function, for the building of product X.

INPUT	VALUE
SYSTEM SIZE	3
TARGET LACTIDE PROPORTION (%)	75
DEGREE OF POLYMERIZATION (MONOMER)	50
TARGET CHAIN BLOCKINESS	1.7
TERMINAL	Ester
RESIDUAL MONOMER (% W/W)	0.05
NACL CONCENTRATION (M)	0.1

Table 2: SwiftPol system build attributes. $\bar{x}_n$ = mean value of attribute across n chains.

ATTRIBUTE	$\bar{x}_n$
SYSTEM SIZE (CHAINS)	3
ACTUAL LACTIDE PROPORTION (%)	68.9
AVERAGE CHAIN BLOCKINESS	1.65
AVERAGE MOLECULE WEIGHT (DALTON)	3370
AVERAGE CHAIN LENGTH (MONOMERS)	50
POLYDISPERSITY INDEX	1.68
BUILD TIME (S)	1.4

Performance Benchmarking

We determined whether SwiftPol can build polymer ensembles and chains with sizes that are relevant to the system scales of interest by performing a stress test. Figure 2 shows measurements of the performance benchmarking results, illustrating that SwiftPol can build large-scale systems in a realistic time frame.

Figure 2: A) Time, t, taken to build systems with a single-chain, ranging from a 10-mer to a 1000-mer. B) Time, t, taken to 50-mer chain build systems ranging from 10 chains to 250 chains.

Conclusion

We presented SwiftPol, an open-source Python package for building polydisperse *in silico* polymer ensembles. SwiftPol recreates core characteristics of bulk polymer materials like polydispersity, enabling the simulation of representative systems that capture key components of polymer physics. We have shown that building longer chains and larger systems, exceeding what would be appropriate for atomistic MD simulations, will not create a time bottleneck in the MD workflow. SwiftPol is a robust and scalable tool for the guided generation of polydisperse polymer mixtures, which can be easily integrated into existing open-source MD software, such as the OpenFF toolkit.

In future releases, we will expand SwiftPol to include options to control tacticity, provide in-package integration into widely-established packing software and offer a broader selection of solvation buffers.

Defining Polymer Properties

SwiftPol uses the following expressions to define key polymer properties.

`Monomer ratio`, R_m, is the ratio of monomer A to monomer B in an AB copolymer, shown in [Equation 1](#)

$$R_m = \frac{n(A)}{n(A+B)} \tag{1}$$

`Degree of polymerization`, *DOP*, is the mean polymer chain length in the system, shown in [Equation 2](#).

$$DOP = \overline{x}(nA + nB) \tag{2}$$

`Number of chains`, n_{chains}, is the total number of chains built by SwiftPol and appended to the object, shown in [Equation 3](#).

$$n_{chains} = \text{total number of chains built} \tag{3}$$

Turney, & Matta. (2025). SwiftPol: A Python package for building and parameterizing *in silico* polymer systems. *Journal of Open Source Software*, *10*(110), 8053. https://doi.org/10.21105/joss.08053

Blockiness, b, is a measurement of the distribution of monomers in an AB copolymer, shown in Equation 4.

$$b = \frac{nB - B\,\text{bonds}}{nA - B\,\text{bonds}}$$ (4)

Residual monomer, M_{resid}, is the % of residual monomer molecules in the system, shown in Equation 5.

$$M_{resid} = \frac{M_w(\text{resid})}{M_w(\text{carbon-containing compounds})}$$ (5)

User guidance: Dependencies

YAML environment file listing all required packages to use SwiftPol

To use SwiftPol please download the following packages:

- RDKit
- openff-interchange
- openff-toolkit
- openff-nagl
- openeye-toolkits
- openff-units==0.2.2
- dgl==2.0.0

optional dependencies:

- espaloma-charge

for quick dependency and swiftpol install using conda, run the following commands in bash

```
conda create -n swiftpol python=3.10 rdkit openff-interchange openff-toolkit openff-nagl
conda activate swiftpol
git clone https://github.com/matta-research-group/SwiftPol
cd SwiftPol
pip install -e .
```

Acknowledgements

References

Abbott, L. J., Hart, K. E., & Colina, C. M. (2013). Polymatic: A generalized simulated polymerization algorithm for amorphous polymers. *Theoretical Chemistry Accounts*, *132*(3), 1334. https://doi.org/10.1007/s00214-013-1334-z

Andrews, J., Handler, R. A., & Blaisten-Barojas, E. (2020). Structure, energetics and thermodynamics of PLGA condensed phases from Molecular Dynamics. *Polymer*, *206*, 122903. https://doi.org/10.1016/j.polymer.2020.122903

Davel, C. M., Bernat, T., Wagner, J. R., & Shirts, M. R. (2024). Parameterization of General Organic Polymers within the Open Force Field Framework. *Journal of Chemical Information and Modeling*, *64*(4), 1390–1305. https://doi.org/10.1021/acs.jcim.3c01691

Daylight Chemical Information Systems, Inc. (2022). *Daylight Theory: SMARTS - A Language for Describing Molecular Patterns*. https://daylight.com/dayhtml/doc/theory/theory.smarts.html

Daylight Chemical Information Systems, Inc. (2022). *Daylight Theory: SMILES*. https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html

Girard, M., Ehlen, A., Shakya, A., Bereau, T., & Cruz, M. O. de la. (2019). Hoobas: A highly object-oriented builder for molecular dynamics. *Computational Materials Science*, *167*, 25–33. https://doi.org/10.1016/j.commatsci.2019.05.003

Grünewald, F., Alessandri, R., Kroon, P. C., Monticelli, L., Souza, P. C. T., & Marrink, S. J. (2022). Polyply; a python suite for facilitating simulations of macromolecules and nanomaterials. *Nature Communications*, *13*(1), 68. https://doi.org/10.1038/s41467-021-27627-4

Hawkins, P. C. D., Skillman, A. G., Warren, G. L., Ellingson, B. A., & Stahl, M. T. (2010). Conformer Generation with OMEGA: Algorithm and Validation Using High Quality Structures from the Protein Databank and Cambridge Structural Database. *Journal of Chemical Information and Modeling*, *50*(4), 572–584. https://doi.org/10.1021/ci100031x

Hayashi, Y., Shiomi, J., Morikawa, J., & Yoshida, R. (2022). RadonPy: Automated physical property calculation using all-atom classical molecular dynamics simulations for polymer informatics. *Npj Computational Materials*, *8*(1), 1–15. https://doi.org/10.1038/s41524-022-00906-4

Kawagoe, Y., Surblys, D., Matsubara, H., Kikugawa, G., & Ohara, T. (2019). Construction of polydisperse polymer model and investigation of heat conduction: A molecular dynamics study of linear and branched polyethylenimine. *Polymer*, *180*, 121721. https://doi.org/10.1016/j.polymer.2019.121721

King's College London. (2024). *King's Computational Research, Engineering and Technology Environment (CREATE)*. https://doi.org/10.18742/rnvf-m076

Klein, C., Sallai, J., Jones, T. J., Iacovella, C. R., McCabe, C., & Cummings, P. T. (2016). A Hierarchical, Component Based Approach to Screening Properties of Soft Matter. In R. Q. Snurr, C. S. Adjiman, & D. A. Kofke (Eds.), *Foundations of Molecular Modeling and Simulation: Select Papers from FOMMS 2015* (pp. 79–92). Springer. https://doi.org/10.1007/978-981-10-1128-3_5

Landrum, G., Tosco, P., Kelley, B., Rodriguez, R., Cosgrove, D., Vianello, R., sriniker, Gedeck, P., Jones, G., NadineSchneider, Kawashima, E., Nealschneider, D., Dalke, A., Swain, M., Cole, B., Turk, S., Savelev, A., Vaucher, A., Wójcikowski, M., … Bisson, J. (2024). *Rdkit/rdkit: 2024_09_2 (Q3 2024) Release*. Zenodo. https://doi.org/10.5281/zenodo.13990314

Li, S.-J., Xie, S.-J., Li, Y.-C., Qian, H.-J., & Lu, Z.-Y. (2016). Influence of molecular-weight polydispersity on the glass transition of polymers. *Physical Review E*, *93*. https:

//doi.org/10.1103/PhysRevE.93.012613

Martínez, L., Andrade, R., Birgin, E. G., & Martínez, J. M. (2009). PACKMOL: A package for building initial configurations for molecular dynamics simulations. *Journal of Computational Chemistry*, *30*(13), 2157–2164. https://doi.org/10.1002/jcc.21224

Ochi, M., Wan, B., Bao, Q., & Burgess, D. J. (2021). Influence of PLGA molecular weight distribution on leuprolide release from microspheres. *International Journal of Pharmaceutics*, *599*, 120450. https://doi.org/10.1016/j.ijpharm.2021.120450

OpenEye Cadence Molecular Sciences. (2025). *OMEGA 6.0.0.1.* OpenEye, Cadence Molecular Sciences. https://www.eyesopen.com

Sahu, H., Shen, K.-H., Montoya, J. H., Tran, H., & Ramprasad, R. (2022). 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*, *18*(4), 2737–2748. https://doi.org/10.1021/acs.jctc.2c00022

Santana-Bonilla, A., López-Ríos de Castro, R., Sun, P., Ziolek, R. M., & Lorenz, C. D. (2023). Modular Software for Generating and Modeling Diverse Polymer Databases. *Journal of Chemical Information and Modeling*, *63*(12), 3761–3771. https://doi.org/10.1021/acs.jcim.3c00081

Schmid, F. (2023). Understanding and Modeling Polymers: The Challenge of Multiple Scales. *ACS Polymers Au*, *3*(1), 28–58. https://doi.org/10.1021/acspolymersau.2c00049

Stipa, P., Marano, S., Galeazzi, R., Minnelli, C., & Laudadio, E. (2021). Molecular dynamics simulations of quinine encapsulation into biodegradable nanoparticles: A possible new strategy against Sars-CoV-2. *European Polymer Journal*, *158*, 110685. https://doi.org/10.1016/j.eurpolymj.2021.110685

Sun, J., Walker, J., Beck-Broichsitter, M., & Schwendeman, S. P. (2022). Characterization of commercial PLGAs by NMR spectroscopy. *Drug Delivery and Translational Research*, *12*(3), 720–729. https://doi.org/10.1007/s13346-021-01023-3

Thompson, M., Wagner, J., Gilmer, J. B., Timalsina, U., Quach, C. D., Boothroyd, S., & Mitchell, J. A. (2024). *OpenFF Interchange*. Zenodo. https://doi.org/10.5281/zenodo.11389943

Triandafilidi, V., Rottler, J., & Hatzikiriakos, S. G. (2016). monodMolecular dynamics simulations of monodisperse/bidisperse polymer melt crystallization. *Journal of Polymer Science Part B: Polymer Physics*, *54*(22), 2318–2326. https://doi.org/10.1002/polb.24142

Wagner, J., Thompson, M., Mobley, D. L., Chodera, J., Bannan, C., Rizzi, A., trevorgokey, Dotson, D. L., Mitchell, J. A., jaimergp, Camila, Behara, P., Bayly, C., JoshHorton, Pulido, I., Wang, L., Lim, V., Sasmal, S., SimonBoothroyd, … Zhao, Y. (2024). *Openforcefield/openff-toolkit: 0.16.0 Minor feature and bugfix release*. Zenodo. https://doi.org/10.5281/zenodo.10967071

Wan, B., Andhariya, J. V., Bao, Q., Wang, Y., Zou, Y., & Burgess, D. J. (2021). Effect of polymer source on in vitro drug release from PLGA microspheres. *International Journal of Pharmaceutics*, *607*, 120907. https://doi.org/10.1016/j.ijpharm.2021.120907