




# flory: A Python package for finding coexisting phases in multicomponent mixtures

Yicheng Qiang <sup>1</sup> and David Zwicker <sup>1</sup>

<sup>1</sup> Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany

DOI: [10.21105/joss.07388](https://doi.org/10.21105/joss.07388)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

---

Editor: [Sarath Menon](#)  

## Reviewers:

- [@SunyongKwon](#)
- [@mastricker](#)

Submitted: 19 September 2024

Published: 11 March 2025

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Summary

Phase separation is an intrinsic property of mixtures that is widely observed in many scenarios, ranging from the simple demixing of oil and water to the condensation of biomolecules in cells ([Hyman et al., 2014](#)). In multicomponent mixtures, phase separation can lead to many coexisting phases, which is crucial in many fields. One key step to understand phase separation is to measure or predict the composition of the coexisting phases. To support such research, the flory package provides an easily accessible, performant, and extensible code that finds coexisting phases in multicomponent mixtures. The package expresses the free energy of the mixtures by several orthogonal aspects to cover a broad range of physical situations. In contrast to existing methods, the flory package implements a state-of-art method that is optimized for mixtures of many components. The package mainly focus on the mixtures with uniform and relatively simple free energies such as Flory-Huggins free energy and its generalizations, while more complicated ones can also be supported through extensions.

## Statement of need

Finding coexisting phases is a common task in many fields, such as chemical engineering ([Lukas et al., 2007](#)) and soft matter physics ([Jacobs, 2023](#)). The coexisting phases can be theoretically predicted by solving the balance equations between phases ([Zwicker & Laan, 2022](#)), or equivalently minimizing the total free energy of the whole mixture ([Lukas et al., 2007](#)). Other strategies include direct spatially-resolved simulations ([Shrinivas & Brenner, 2021](#)) and the construction of the convex hull of the free energy landscape ([Mao et al., 2019](#)). There are a few open-source packages that implement these strategies. Most notably, Calphad packages, including Equilipy ([Kwon et al., 2024](#)), pycalphad ([Otis & Liu, 2017](#)) and OpenCalphad ([Sundman et al., 2015](#)), combine a database of candidate phases and the strategies above to compute phase diagrams of mixtures with few components. In addition, SurfInPy ([Tse et al., 2022](#)) applies the free energy minimization strategy to surface phases.

In general, finding coexisting phases is challenging for mixtures with a large number of the components,  $N_C$ . This is because the number of degrees of freedom (e.g., to describe the composition of the phases) increases with larger  $N_C$ . Moreover, the possible number of coexisting phases also increases with  $N_C$  according to Gibbs phase rule, implying that the free energy of the entire system comprises roughly  $N_C^2$  free variables. This high-dimensional space needs to be sampled to find the global minimum with multiple coexisting phases, which is infeasible for some of the strategies mentioned above since they become prohibitively expensive. For example, the cost of the convex hull strategy increases exponentially with  $N_C$  since it requires sampling the entire free energy landscape. The existing Calphad packages address this challenge by taking the advantage of several strategies ([Lukas et al., 2007](#)). For example, the free energy minimization strategy can refine the results obtained from the convex hull strategy. Besides, Calphad packages usually provide high flexibility on candidate phases, allowing each

phase to have different free energies to model realistic systems (Sundman et al., 2015). In contrast, flory focuses on the general physics of multicomponent phase separation, and thus assumes that all candidate phases share the same free energy function, e.g., the simple Flory-Huggins free energy, similar to a recent submodule of the OpenCalphad package (Li et al., 2020). These simple models are more common in liquid systems such as polymer mixtures, and have recently been considered relevant to phase separation in biological cells. The restrictions of the physical model simplifies the user interface and allow for an efficient optimization algorithm. For example, flory implements an algorithm that automatically satisfies the constraints of ensembles, thus reducing the Lagrange multipliers required (White et al., 1958). To obtain the coexisting phases in equilibrium without the prior knowledge of the compositions of the phases, flory starts from many phases initially and then clusters the equivalent phases afterwards. As the result, the flory package can efficiently determine the multiple coexisting phases in equilibrium in a range of multicomponent mixtures with large number of the components  $N_C$ .

## Methods

The flory package is based on the free energy minimization strategy. To reduce computation cost, the package focuses on coexisting phases in thermodynamically large systems, where the interfaces between phases become negligible. Coexisting phases can thus be found by minimizing the average free energy density  $\bar{f}$  of the entire system, which is given by

$$\bar{f}(N_P, \{J_p\}, \{\phi_{p,i}\}) = \sum_{p=1}^{N_P} J_p f(\{\phi_{p,i}\}),$$

where  $N_C$  is the number of components,  $N_P$  is the number of phases,  $J_p$  denotes the fraction of volume that phase  $p = 1, \dots, N_P$  occupies in the entire system, and  $\phi_{p,i}$  is the volume fraction of component  $i = 1, \dots, N_C$  in phase  $p$ . The physical behavior of the mixture is encoded in the free energy density  $f$ , which flory expresses using four orthogonal aspects: interaction, entropy, ensemble, and constraints. The package only imposes limits on the entropy part, which is crucial for the core algorithm, while the other three aspects are rather flexible. For instance, the interactions can be described by quadratic terms, like in the Flory-Huggins model, and the parameters can be obtained from database such as 3PDB (*Polymer Property Predictor and Database*, 2019) for realistic polymer mixtures, or freely chosen for theoretical investigations. By combining these four aspects, flory supports a broad range of free energy densities  $f$  with different ensembles and constraints. A few widely-used specializations are provided for all four aspects, while customized ones can be added easily.

The flory package is designed to deliver high performance. The core part of the flory package is the finder for coexisting phases, which can be reused when the number  $N_C$  of components is kept fixed. This design moves a significant overhead to the creation of the solver, which can be amortized in many tasks, e.g., when a phase diagram is sampled. The core methods in the finder are just-in-time (JIT) compiled using numba (Lam et al., 2015) to achieve high performance. To support different forms of the free energy  $f$ , the core method is also designed to be general. The finder fetches compiled instances of the interaction, entropy, ensemble, and constraints, where case-specific codes are inserted as methods. These methods are also compiled for performance, using the experimental jitclass feature from numba (Lam et al., 2015).

The flory package adopts state-of-the-art numerical methods to determine coexisting phases. The main idea is to represent the system by many independent compartments, which can exchange particles and volumes, obeying total constraints (Zwicker & Laan, 2022). The flory package then minimizes the full free energy  $\bar{f}$  instead of directly solving the corresponding coexistence conditions. At the free energy minimum, compartments may share similar compositions, which the package then cluster to obtain unique phases using the scipy cluster methods

(Virtanen et al., 2020). This strategy circumvents the typical challenge of multiple local minima and it avoids iterating over all possible phase counts  $N_P$ . To improve performance, the flory package implements the improved Gibbs ensemble method we developed recently (Qiang et al., 2024). This method redistributes components across all compartments simultaneously, guided by a set of conjugate variables, such that the total computation cost per step only scales linearly with the number of compartments. Beside the core function of finding coexisting phases, flory also includes convenience tools, e.g., to analyze thermodynamic properties such as chemical potentials and osmotic pressures. In summary, the flory package can typically obtain the equilibrium coexisting states even in systems with many interacting components.

## Examples

The following code illustrates the main functionality of the package by finding the two coexisting phases of a symmetric Flory-Huggins binary mixture with  $\chi = 4$ :

```
fh = flory.FloryHuggins(2, chis=[[0, 4.0], [4.0, 0]])
ensemble = flory.CanonicalEnsemble(2, phi_means=[0.5, 0.5])
finder = flory.CoexistingPhasesFinder(fh.interaction, fh.entropy, ensemble)
phases = finder.run().get_clusters()
```

Here, FloryHuggins represents the seminal Flory-Huggins free energy that creates the interaction FloryHugginsInteraction and the entropy IdealGasEntropy simultaneously, and provides tools for analyzing coexisting phases. By updating the interaction, we can then obtain the coexisting phases of another symmetric binary mixture with  $\chi = 3.5$ :

```
fh.chis = [[0, 3.5], [3.5, 0]]
finder.set_interaction(fh.interaction)
phases = finder.run().get_clusters()
```

This procedure can be repeated to sample an entire phase diagrams. For example, the following code will generate the phase diagram of a mixture of two components with identical molecular sizes,

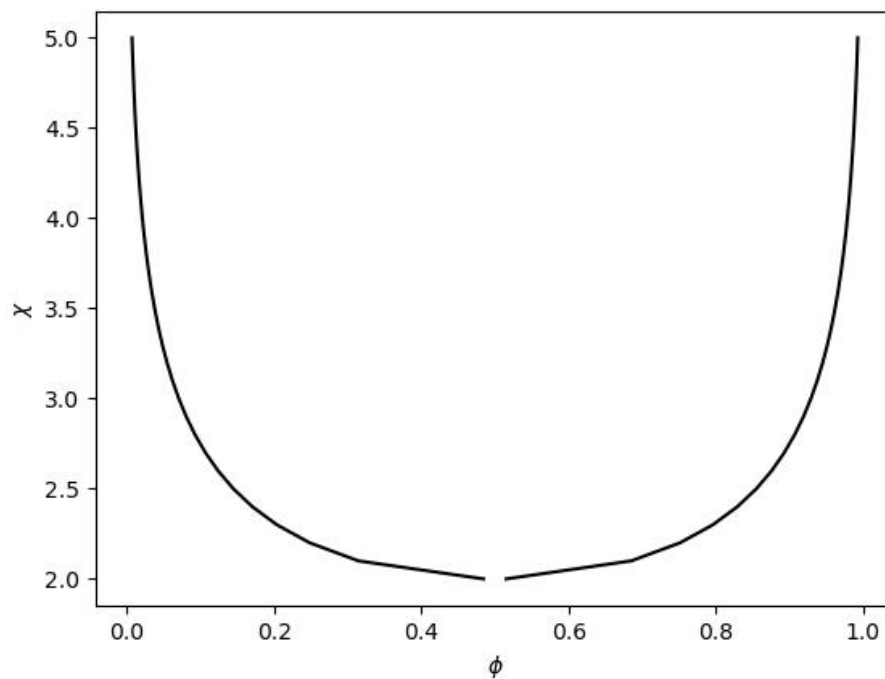
```
import matplotlib.pyplot as plt
import numpy as np
import flory

fh = flory.FloryHuggins(2, chis=[[0, 5.0], [5.0, 0]])
ensemble = flory.CanonicalEnsemble(2, phi_means=[0.5, 0.5])
finder = flory.CoexistingPhasesFinder(fh.interaction, fh.entropy, ensemble)

line_chi = []
line_l = []
line_h = []
for chi in np.arange(5.0, 1.0, -0.1): # scan chi from high value to low value
    fh.interaction.chis = [[0, chi], [chi, 0]] # set chi matrix of the finder
    finder.set_interaction(fh.interaction)
    phases = finder.run().get_clusters() # get coexisting phases
    if phases.fractions.shape[0] == 1: # stop scanning if no phase separation
        break
    line_chi.append(chi)
    line_l.append(phases.fractions[1, 0])
    line_h.append(phases.fractions[0, 0])

plt.plot(line_l, line_chi, c="black")
plt.plot(line_h, line_chi, c="black")
plt.xlabel("$\\phi$")
```

```
plt.ylabel("$\\chi$")  
plt.show()
```



**Figure 1:** Phase diagram of binary mixture. The lines show the coexisting composition  $\phi$  at a given interaction strength  $\chi$ , together known as a the binodal line.

Moreover, one can vary the type of interaction by initializing a different class or by modifying the existing one, and one could similarly change the entropy, ensemble, and constraints. Customized specialization of all four aspects can be easily implemented by deriving from the provided base classes (*“Flory” Python Package — Flory Documentation, 2025*).

## Acknowledgements

We thank Chengjie Luo for stimulating discussions. We gratefully acknowledge funding from the Max Planck Society and the European Union (ERC, EmulSim, 101044662).

## References

- “Flory” Python package — flory documentation.* (2025). <https://flory.readthedocs.io>
- Hyman, A. A., Weber, C. A., & Jülicher, F. (2014). Liquid-Liquid Phase Separation in Biology. *Annual Review of Cell and Developmental Biology*, 30(1), 39–58. <https://doi.org/10.1146/annurev-cellbio-100913-013325>
- Jacobs, W. M. (2023). Theory and Simulation of Multiphase Coexistence in Biomolecular Mixtures. *Journal of Chemical Theory and Computation*, 19(12), 3429–3445. <https://doi.org/10.1021/acs.jctc.3c00198>
- Kwon, S. Y., Thibodeau, E., Plotkowski, A., & Yang, Y. (2024). Equilipy: A python

- package for calculating phase equilibria. *Journal of Open Source Software*, 9(100), 6875. <https://doi.org/10.21105/joss.06875>
- Lam, S. K., Pitrou, A., & Seibert, S. (2015). Numba: A LLVM-based Python JIT compiler. *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, 1–6. <https://doi.org/10.1145/2833157.2833162>
- Li, J., Sundman, B., Winkelman, J. G. M., Vakis, A. I., & Picchioni, F. (2020). Implementation of the UNIQUAC model in the OpenCalphad software. *Fluid Phase Equilibria*, 507, 112398. <https://doi.org/10.1016/j.fluid.2019.112398>
- Lukas, H. L., Fries, S. G., & Sundman, B. (2007). *Computational thermodynamics: The Calphad method*. Cambridge University Press. ISBN: 978-0-521-86811-2
- Mao, S., Kuldinow, D., Haataja, M. P., & Košmrlj, A. (2019). Phase behavior and morphology of multicomponent liquid mixtures. *Soft Matter*, 15(6), 1297–1311. <https://doi.org/10.1039/C8SM02045K>
- Otis, R., & Liu, Z.-K. (2017). Pycalphad: CALPHAD-based Computational Thermodynamics in Python. *Journal of Open Research Software*, 5(1). <https://doi.org/10.5334/jors.140>
- Polymer Property Predictor and Database*. (2019). <https://pppdb.uchicago.edu>
- Qiang, Y., Luo, C., & Zwicker, D. (2024). *Scaling of phase count in multicomponent liquids* (No. arXiv:2405.01138). arXiv. <https://arxiv.org/abs/2405.01138>
- Shrinivas, K., & Brenner, M. P. (2021). Phase separation in fluids with many interacting components. *Proceedings of the National Academy of Sciences*, 118(45), e2108551118. <https://doi.org/10.1073/pnas.2108551118>
- Sundman, B., Kattner, U. R., Palumbo, M., & Fries, S. G. (2015). OpenCalphad - a free thermodynamic software. *Integrating Materials and Manufacturing Innovation*, 4(1), 1–15. <https://doi.org/10.1186/s40192-014-0029-1>
- Tse, J. S., Molinari, M., Parker, S. C., & Symington, A. R. (2022). SurfinPy 2.0: A Phase Diagram Generator for Surfaces and Bulk Phases. *Journal of Open Source Software*, 7(71), 4014. <https://doi.org/10.21105/joss.04014>
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... van Mulbregt, P. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, 17(3), 261–272. <https://doi.org/10.1038/s41592-019-0686-2>
- White, W. B., Johnson, S. M., & Dantzig, G. B. (1958). Chemical Equilibrium in Complex Mixtures. *The Journal of Chemical Physics*, 28(5), 751–755. <https://doi.org/10.1063/1.1744264>
- Zwicker, D., & Laan, L. (2022). Evolved interactions stabilize many coexisting phases in multicomponent liquids. *Proceedings of the National Academy of Sciences*, 119(28), e2201250119. <https://doi.org/10.1073/pnas.2201250119>