

# TransiFlow: A Python package for performing bifurcation analysis on fluid flow problems

Sven Baars<sup>1</sup>, David Nolte<sup>2</sup>, Fred W. Wubs<sup>2</sup>, and Henk A. Dijkstra<sup>1</sup>

<sup>1</sup> Institute for Marine and Atmospheric Research (IMAU), Utrecht University, The Netherlands <sup>2</sup> Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, University of Groningen, The Netherlands

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

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

Editor: [Jack Atkinson](#)

## Reviewers:

- [@YidanXue](#)
- [@svchb](#)
- [@cwentland0](#)

Submitted: 28 February 2025

Published: unpublished

## License

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

## Summary

Dynamical systems derived from models of fluid flow show transition behaviors associated with fluid flow instabilities (Wubs & Dijkstra, 2023). For example, dynamical systems from ocean models in which transition behavior is caused by slow changes in parameters representing the surface forcing (Westen et al., 2024). A flow transition is a qualitative change in the flow when a specific parameter is varied, e.g. a transition from a no-flow heat-conducting fluid to a heat-transporting flow (as in Rayleigh-Bénard convection), or a steady flow which has a steady forcing and steady boundary conditions that turns into a transient flow which may even produce sound (as in the von Kármán vortex street). In both these examples the qualitatively different solutions do not appear out of the blue.

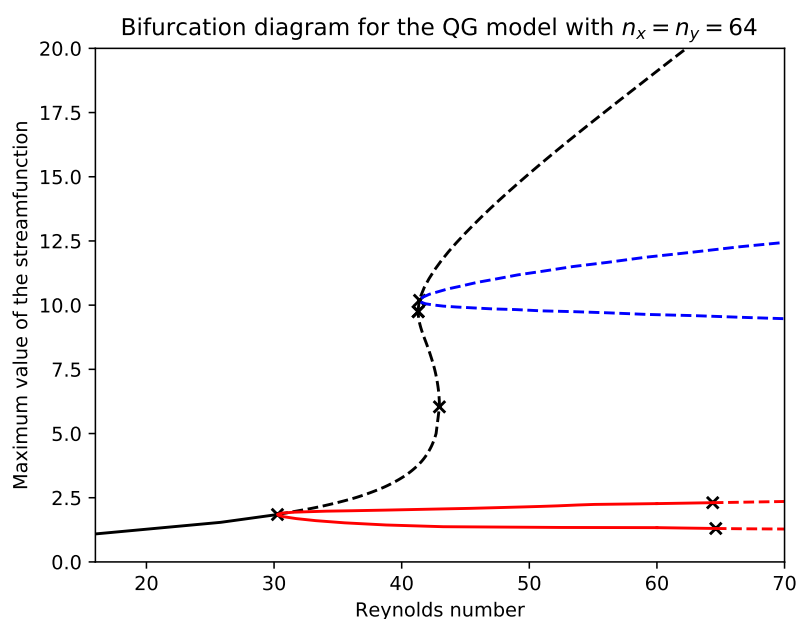
If one perturbs a steady solution before the transition point then the perturbation will die out and recover to the steady solution. This means that the steady solution is stable. However, this perturbed flow already reveals the shape that will occur after the transition point. For forcing beyond this point that shape will grow into a new steady flow (the stable solution becomes unstable) for Rayleigh-Bénard convection and into a transient flow for the von Kármán vortex street (the stationary flow becomes unstable). The parameter value for which the transition occurs is called a bifurcation point, sometimes also referred to as a transition or tipping point.

Studying these phenomena (bifurcation analysis) can be done by performing numerical simulations and observing transient behavior after a certain time. This is, however, computationally expensive and in many cases infeasible. Instead, so-called continuation methods are able to trace stable and unstable steady states in parameter space, obviating expensive transient simulations (Dijkstra, 2005). The TransiFlow Python package implements a continuation framework in which fluid flow problems can be studied with the help of several computational back-ends that can, based on the needs of the user, be easily switched between.

One motivation behind TransiFlow is that writing research software that works efficiently on a parallel computer is a challenging task. Numerical models are often developed as a sequential code with parallelization as an afterthought, making them difficult to parallelize, or as a parallel code from the start, making them complicated to work with. This is especially problematic since people that work with these codes generally only use with them for the duration of a project. If there is insufficient documentation and continuity between the projects then knowledge of how to use or develop the codes may get lost, rendering the software useless.

In climate modelling, this is a prominent issue, since the models are complex, are usually intercoupled with other models (e.g., ocean, atmosphere, ice), take a very long time to run (i.e., multiple months) and require large amounts of parallelism to reach a sufficient resolution (i.e., using thousands of cores for a single run) (Mulder et al., 2021; Thies et al., 2009). Therefore, ease of developing and using the parallel software is crucial.

We achieve this by abstracting away the computational back-end, which allows the user to adjust a model to their own needs on their own machine (e.g., a laptop) in Python using the SciPy back-end. Assuming it is implemented on a Cartesian grid, the user can then run a large-scale simulation on an HPC system using, e.g., the Trilinos back-end, which can use a combination of OpenMP, MPI, and potentially GPUs, without requiring any changes to the model code. The computationally expensive parts of the program are implemented by these libraries so one does not have to worry about the efficiency of the Python implementation of the model. Initial tests indicate that the overhead of using Python is small when compared to the expensive linear system solves that are handled by the external libraries that are implemented in e.g. C++.



**Figure 1:** Bifurcation diagram of the double-gyre wind-driven circulation configuration that is included in TransiFlow. The markers indicate pitchfork, Hopf and saddle-node bifurcations that were automatically detected by the software. Solid lines indicate stable steady states of the system; dashed lines indicate unstable steady states. A more extensive description of the bifurcation diagram and steps to reproduce it can be found in (Sapsis & Dijkstra, 2013).

## Statement of need

TransiFlow aims to be an easy-to-use tool for performing bifurcation analysis on fluid flow problems that can be used in combination with fast parallel solvers with minimal additional effort. TransiFlow implements pseudo-arclength continuation and implicit time integration methods, as well as finite-volume discretizations for the incompressible Navier-Stokes equations with optional heat and salinity transport. We also provide implementations of various canonical fluid flow problems such as lid-driven and differentially heated cavities, Rayleigh-Bénard convection and Taylor-Couette flow, a feature none of its competitors provide.

The main competitors are AUTO (Doedel et al., 2007), MatCont (Dhooge et al., 2008) BifurcationKit.jl (Veltz, 2020) and pde2path (Uecker, 2014). These packages are widely used, and they are much more feature complete in terms of bifurcation analysis. However, they lack the discretized fluid flow models and parallel solver interfaces that make TransiFlow easy to use.

66 A continuation package that allows for easy coupling with parallel solvers is the [LOCA](#) package in  
67 Trilinos ([The Trilinos Project Team, 2024](#)). Using LOCA, however, requires a vast knowledge  
68 of C++ and any change to the continuation algorithm requires a full stack of C++ templates  
69 to be reimplemented. Moreover, LOCA has not seen any active development in over 10 years.

70 An alternative Python package for performing bifurcation analysis is [PyNCT](#) ([Draelants et al.,](#)  
71 [2015](#)). It is, however, difficult to extend, the latest version of the software is not freely available,  
72 and it can only be used for systems that are symmetric positive-definite.

## 73 Past and ongoing research

74 TransiFlow has been used to generate results in ([Wubs & Dijkstra, 2023](#)) and ([Bernuzzi et al.,](#)  
75 [2024](#)) and has been used in courses at Utrecht University and the University of Groningen (NL).  
76 Earlier versions have also been used to generate results in ([Song, 2019](#)) and ([Baars et al., 2020](#)).  
77 The code is currently being used in various projects at Utrecht University, such as a project  
78 for coupling climate models and a project for studying transitions in the Atlantic Meridional  
79 Overturning Circulation. There is also ongoing work on adding 3D ocean and atmosphere  
80 discretizations to TransiFlow. TransiFlow is also used by several external researchers.

## 81 Acknowledgements

82 H.A.D. and S.B. are funded by the European Research Council through the ERC-AdG project  
83 TAOC (project 101055096). S.B. and F.W.W. were supported by funding from the SMCM  
84 project of the Netherlands eScience Center (NLeSC) with project number 027.017.G02. We  
85 would like to thank Lourens Veen for his contributions in providing guidance in setting up the  
86 documentation and continuous deployment.

## 87 References

- 88 Baars, S., Klok, M. der, Thies, J., & Wubs, F. W. (2020). A staggered-grid multilevel incomplete  
89 LU for steady incompressible flows. *International Journal for Numerical Methods in Fluids*,  
90 93(4), 909–926. <https://doi.org/10.1002/flid.4913>
- 91 Bernuzzi, P., Dijkstra, H., & Kuehn, C. (2024). *Warning signs for boundary noise and their*  
92 *application to an ocean boussinesq model*. <https://doi.org/10.2139/ssrn.4845893>
- 93 Dhooge, A., Govaerts, W., Kuznetsov, Yu. A., Meijer, H. G. E., & Sautois, B. (2008).  
94 New features of the software MatCont for bifurcation analysis of dynamical systems.  
95 *Mathematical and Computer Modelling of Dynamical Systems*, 14(2), 147–175. <https://doi.org/10.1080/13873950701742754>
- 96  
97 Dijkstra, H. A. (2005). *Nonlinear Physical Oceanography: A Dynamical Systems Approach*  
98 *to the Large Scale Ocean Circulation and El Niño*, 2nd Revised and Enlarged edition. In  
99 *Atmospheric and Oceanographic Sciences Library*. Springer Netherlands. <https://doi.org/10.1007/1-4020-2263-8>
- 100  
101 Doedel, E. J., Champneys, A. R., Dercole, F., Fairgrieve, T. F., Kuznetsov, Y. A., Oldeman, B.,  
102 Paffenroth, R., Sandstede, B., Wang, X., & Zhang, C. (2007). *AUTO-07P: Continuation*  
103 *and bifurcation software for ordinary differential equations*.
- 104 Draelants, D., Kłosiewicz, P., Broeckhove, J., & Vanroose, W. (2015). Solving general auxin  
105 transport models with a numerical continuation toolbox in python: PyNCT. In *Hybrid*  
106 *systems biology* (pp. 211–225). Springer International Publishing. [https://doi.org/10.1007/978-3-319-26916-0\\_12](https://doi.org/10.1007/978-3-319-26916-0_12)
- 107  
108 Mulder, T. E., Goelzer, H., Wubs, F. W., & Dijkstra, H. A. (2021). Snowball earth bifurcations

- 109 in a fully-implicit earth system model. *International Journal of Bifurcation and Chaos*,  
110 31(06), 2130017. <https://doi.org/10.1142/s0218127421300172>
- 111 Sapsis, T. P., & Dijkstra, H. A. (2013). Interaction of additive noise and nonlinear dynamics in  
112 the double-gyre wind-driven ocean circulation. *Journal of Physical Oceanography*, 43(2),  
113 366–381. <https://doi.org/10.1175/jpo-d-12-047.1>
- 114 Song, W. (2019). *Matrix-based techniques for (flow-)transition studies* [PhD thesis]. University  
115 of Groningen; University of Groningen. ISBN: 978-94-034-1353-2
- 116 The Trilinos Project Team. (2024). *The Trilinos Project Website*. <https://trilinos.github.io>
- 117 Thies, J., Wubs, F., & Dijkstra, H. A. (2009). Bifurcation analysis of 3D ocean flows  
118 using a parallel fully-implicit ocean model. *Ocean Modelling*, 30(4), 287–297. <https://doi.org/10.1016/j.ocemod.2009.07.005>
- 120 Uecker, H. (2014). pde2path - a Matlab package for continuation and bifurcation in 2D  
121 elliptic systems. *Numerical Mathematics: Theory, Methods and Applications*, 7(1), 58–106.  
122 <https://doi.org/10.4208/nmtma.2014.1231nm>
- 123 Veltz, R. (2020). *BifurcationKit.jl*. Inria Sophia-Antipolis. <https://hal.archives-ouvertes.fr/hal-02902346>
- 125 Westen, R. M. van, Kliphuis, M., & Dijkstra, H. A. (2024). Physics-based early warning signal  
126 shows that AMOC is on tipping course. *Science Advances*, 10(6). <https://doi.org/10.1126/sciadv.adk1189>
- 128 Wubs, F. W., & Dijkstra, H. A. (2023). *Bifurcation analysis of fluid flows*. Cambridge  
129 University Press. <https://doi.org/10.1017/9781108863148>

DRAFT