The Journal of Open Source Software

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

Sven Baars [1], **David Nolte** [2], **Fred W. Wubs** [2], and **Henk A. Dijkstra** [1]

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

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Jack Atkinson

Reviewers:

- @YidanXue
- @svchb
- @cwentland0

Submitted: 28 February 2025
Published: unpublished

Summary

Dynamical systems such as those derived from models of fluid flows show transition behavior associated with fluid flow instabilities (Wubs & Dijkstra, 2023). Examples include dynamical systems from ocean models in which transition behavior is caused by slow changes in parameters representing the surface forcing (Westen et al., 2024). By flow transition one means a qualitative change in the flow when a specific parameter is changed, e.g. a transition from a no-flow heat-conducting fluid to a heat-transporting flow (as in Rayleigh-Bénard convection), or a flow which has a steady forcing and steady boundary conditions that turns from a steady flow into a transient flow which may even produce sound (as in the von Kármán vortex street). In the foregoing two examples both qualitatively different solutions do not just appear out of the blue. If one perturbs the steady solution before the transition point then with time the perturbation will die out and we will recover the steady solution. This means that the steady solution is stable. However, the perturbation already reveals the shape of the flow that will occur after the transition point. After this point that shape will grow into a steady flow (the no-flow solution becomes unstable) for the first example and a transient flow in the second example (the stationary flow becomes unstable).The parameter value for which the transition sets in is called a bifurcation point, which, in certain cases, is also referred to as tipping point.

Studying these phenomena (bifurcation analysis) can be done by performing numerical simulations with a model and observing its transient behavior after a certain time. This is, however, computationally very 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, easily be switched between.

One motivation behind `TransiFlow` is that writing research software that works efficiently on a parallel computer is a challenging task. Therefore, numerical models are often developed as a sequential code with parallelization as an afterthought, which makes them very difficult to parallelize, or as a parallel code from the start, which makes them complicated to work with for researchers. This is especially problematic since people that work with these codes generally only work with them for the duration of their project. If there is insufficient continuity between the projects, knowledge of how to use the codes or work on them may get lost, which renders the developed 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.

By abstracting away the computational back-end from the user, the user can adjust a model to their own needs on their own machine (e.g., a laptop) in Python using the SciPy back-end, and once the model works, run a large scale simulation on a supercomputer, e.g., using the `Trilinos` back-end, which can use a combination of OpenMP, MPI and potentially GPUs, without requiring any changes to the 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 less than 1% of the total computational cost.

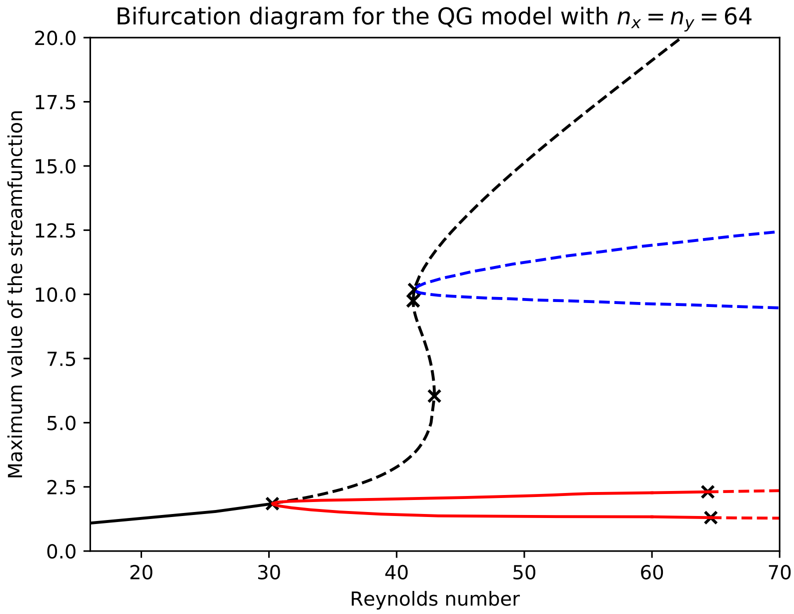

Figure 1: Bifurcation diagram of the double-gyre wind-driven circulation configuration that is included in `TransiFlow`. The markers indicate pitchfork, Hopf and saddle-node bifurcations that were automatically detected by the software. Solid lines indicate stable steady states of the system; dashed 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 without any additional effort. For this purpose, `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. They do, however, lack the discretized fluid flow models and parallel solver interfaces that make `TransiFlow` easy to use.

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

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

Past and ongoing research

`TransiFlow` has been used to generate results in (Wubs & Dijkstra, 2023) and (Bernuzzi et al., 2024) and has been used in courses at Utrecht University and the University of Groningen (NL). Earlier versions have also been used to generate results in (Song, 2019) and (Baars et al., 2020). The code is currently being used in various projects at Utrecht University and by several external researchers.

Acknowledgements

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

References

Baars, S., Klok, M. der, Thies, J., & Wubs, F. W. (2020). A staggered-grid multilevel incomplete LU for steady incompressible flows. *International Journal for Numerical Methods in Fluids*, *93*(4), 909–926. https://doi.org/10.1002/fld.4913

Bernuzzi, P., Dijkstra, H., & Kuehn, C. (2024). *Warning signs for boundary noise and their application to an ocean boussinesq model*. https://doi.org/10.2139/ssrn.4845893

Dhooge, A., Govaerts, W., Kuznetsov, Yu. A., Meijer, H. G. E., & Sautois, B. (2008). New features of the software MatCont for bifurcation analysis of dynamical systems. *Mathematical and Computer Modelling of Dynamical Systems*, *14*(2), 147–175. https://doi.org/10.1080/13873950701742754

Dijkstra, H. A. (2005). Nonlinear Physical Oceanography: A Dynamical Systems Approach to the Large Scale Ocean Circulation and El Niño, 2nd Revised and Enlarged edition. In *Atmospheric and Oceanographic Sciences Library*. Springer Netherlands. https://doi.org/10.1007/1-4020-2263-8

Doedel, E. J., Champneys, A. R., Dercole, F., Fairgrieve, T. F., Kuznetsov, Y. A., Oldeman, B., Paffenroth, R., Sandstede, B., Wang, X., & Zhang, C. (2007). *AUTO-07P: Continuation and bifurcation software for ordinary differential equations*.

Draelants, D., Kłosiewicz, P., Broeckhove, J., & Vanroose, W. (2015). Solving general auxin transport models with a numerical continuation toolbox in python: PyNCT. In *Hybrid systems biology* (pp. 211–225). Springer International Publishing. https://doi.org/10.1007/978-3-319-26916-0_12

Mulder, T. E., Goelzer, H., Wubs, F. W., & Dijkstra, H. A. (2021). Snowball earth bifurcations in a fully-implicit earth system model. *International Journal of Bifurcation and Chaos*, *31*(06), 2130017. https://doi.org/10.1142/s0218127421300172

Sapsis, T. P., & Dijkstra, H. A. (2013). Interaction of additive noise and nonlinear dynamics in the double-gyre wind-driven ocean circulation. *Journal of Physical Oceanography*, *43*(2), 366–381. https://doi.org/10.1175/jpo-d-12-047.1

Song, W. (2019). *Matrix-based techniques for (flow-)transition studies* [PhD thesis]. University of Groningen; University of Groningen. ISBN: 978-94-034-1353-2

The Trilinos Project Team. (2024). *The Trilinos Project Website*. https://trilinos.github.io

Thies, J., Wubs, F., & Dijkstra, H. A. (2009). Bifurcation analysis of 3D ocean flows using a parallel fully-implicit ocean model. *Ocean Modelling*, *30*(4), 287–297. https://doi.org/10.1016/j.ocemod.2009.07.005

Uecker, H. (2014). pde2path - a Matlab package for continuation and bifurcation in 2D elliptic systems. *Numerical Mathematics: Theory, Methods and Applications*, *7*(1), 58–106. https://doi.org/10.4208/nmtma.2014.1231nm

Veltz, R. (2020). *BifurcationKit.jl*. Inria Sophia-Antipolis. https://hal.archives-ouvertes.fr/hal-02902346

Westen, R. M. van, Kliphuis, M., & Dijkstra, H. A. (2024). Physics-based early warning signal shows that AMOC is on tipping course. *Science Advances*, *10*(6). https://doi.org/10.1126/sciadv.adk1189

Wubs, F. W., & Dijkstra, H. A. (2023). *Bifurcation analysis of fluid flows*. Cambridge University Press. https://doi.org/10.1017/9781108863148