NumbaCS: A fast Python package for coherent structure analysis

Albert Jarvis [1] **and Shane D. Ross** [2]

1 Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA, USA **2** Department of Ocean and Aerospace Engineering, Virginia Tech, Blacksburg, VA, USA

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor:

Submitted: 12 October 2024
Published: unpublished

Summary

NumbaCS (Numba Coherent Structures) is a Python package that implements a variety of coherent structure methods in an efficient and user-friendly manner. "Coherent structure methods" refer to any method which can be used to infer or extract Lagrangian and objective Eulerian coherent structures. The theory behind these methods have been developed over the last few decades with the aim of extending many of the important invariant objects from time-independent dynamical systems theory to the more general setting where a system may have arbitrary time dependence and may only be known or defined for some finite time. These time-dependent systems are ubiquitous in the context of geophysical and engineering flows where the evolution of the velocity field depends on time and velocity data representing these flows is not available for all time. By extending the ideas from the time-independent setting to the more general time-dependent setting, important transient objects (coherent structures) can be identified which govern how material is transported within a flow. Understanding material transport in flows is of great importance for applications ranging from monitoring the transport of a contaminant in the ocean or atmosphere to informing search and rescue strategies for persons lost at sea.

Statement of need

As theory and implementations of coherent structures have been developed (Farazmand & Haller, 2012; Haller, 2011; Haller et al., 2016; Haller & Beron-Vera, 2013; Haller & Poje, 1998; Mathur et al., 2007; Nolan, Serra, et al., 2020; Schindler et al., 2012; Serra & Haller, 2016; Shadden et al., 2005) and the utility of these tools has been demonstrated over the last two decades (Curbelo & Rypina, 2023; Du Toit & Marsden, 2010; Günther et al., 2021; Liu et al., 2018; Nolan, Foroutan, et al., 2020; Peacock & Haller, 2013; Pretorius et al., 2023; Rutherford et al., 2012; Serra et al., 2017), there has been a steadily growing interest in using these methods for real-world applications. Early on, software implementations were largely contained to in-house packages developed by applied mathematicians and engineers advancing the theory. Over the years, there have been a number of software packages developed in an attempt to provide implementations of some of these methods for practitioners outside of the field. Some provide a friendly interface for users (`Dynlab` – Nolan (2024); `LCS MATLAB Kit` – Dabiri (2009)), others aim to provide efficient implementations of specific methods (sometimes in specific circumstances) (`Lagrangian` – Briol & d'Ovidio (2011); `Newman` – Du Toit (2010); `Aquila-LCS` – Lagares & Araya (2023)), and a few implement a variety of methods (`Tbarrier` – Bartos et al. (2022); `LCS Tool` - Onu et al. (2015)). NumbaCS intends to unite these aims by providing efficient and user-friendly implementations of a variety of coherent structure methods. By doing this, the hope is to provide a powerful tool for experienced practitioners and a low barrier of entry for newcomers. In addition, as new methods/implementations arise, the framework laid out in `NumbaCS` provides a straightforward environment for contributions and maintenance. Also of note is another package called `CoherentStructures.jl` (Junge et al., 2020), which is fast, user-friendly, and implements a variety of methods. This package has some overlap with `NumbaCS` but they both implement methods which the other does not. `CoherentStructures.jl` is a powerful tool that should be considered by users who perhaps prefer Julia to Python or are interested in computing some of the methods not implemented in `NumbaCS`. For a more detailed breakdown of how all of the mentioned packages compare with `NumbaCS`, see the documentation.

Functionality

`NumbaCS` implements the following features for both analytical and numerical flows:

- Standard flow map computation
- Flow map composition method (Brunton & Rowley, 2010)
- Finite time Lyapunov exponent (FTLE) (Shadden et al., 2005)
- instantaneous Lyapunov exponent (iLE) (Nolan, Serra, et al., 2020)
- Lagrangian averaged vorticity deviation (LAVD) (Haller et al., 2016)
- Instantaneous vorticity deviation (IVD) (Haller et al., 2016)
- FTLE ridge extraction (Schindler et al., 2012; Steger, 1998)
- Variational hyperbolic LCS (Farazmand & Haller, 2012; Haller, 2011)
- Variational hyperbolic OECS (Serra & Haller, 2016)
- LAVD-based elliptic LCS (Haller et al., 2016)
- IVD-based elliptic OECS (Haller et al., 2016)

For flows defined by numerical velocity data:

- Simple creation of JIT compiled linear and cubic interpolants

All of these implementations are relatively straightforward to use and quite efficient. This is due to three key dependencies `NumbaCS` utilizes to speed up computations. The first is `Numba` (Lam et al., 2015), a JIT compiler for Python which can drastically speed up numerical operations and provides a simple framework for parallelizing tasks. Next, `numbalsoda` (Wogan, 2021) is a Python wrapper to ODE solvers in both C++ (LSODA) and FORTRAN (DOP853) that bypasses the Python interpreter and can be used within Numba functions (standard Python ODE solvers cannot be executed within `Numba` functions). This package is crucial to the efficiency of NumbaCS as particle integration is often the most costly part of finite-time coherent structure methods. Finally, the `interpolation` package (Winant et al., 2017) provides optimized interpolation in Python and is utilized in `NumbaCS` to create JIT compiled interpolant functions, producing efficient implementations of methods even for flows defined by numerical data. By taking advantage of these packages behind the scenes, `NumbaCS` is able to maintain the simplicity and readability of an interpreted language while achieving runtimes closer to that of a compiled language.

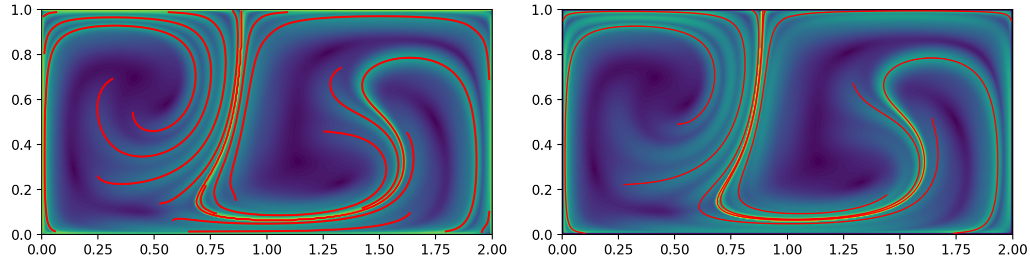
Examples

A User Guide is provided which details the workflow in NumbaCS and a number of examples demonstrating the functionality are covered in the Example Gallery. Here we show the output of a few examples, provide the runtime of each, and breakdown the runtime based on the parts of each method. "Flowmap" refers to the particle integration step, "C eig" and "S eig" refer to the eigenvalue/vector step for Lagrangian and Eulerian methods respectively (this time will be roughly equal to the FTLE and iLE times), and the last is the extraction time for a given method. For examples that require particle integration, the default solver (DOP853) was used with the default error tolerances (relative tolerance = 1e-6, absolute tolerance = 1e-8). All runs were performed on an Intel$^{(R)}$ Core^{TM} i7-3770K CPU @ 3.50GHz (which has 4 cores and

The Journal of Open Source Software

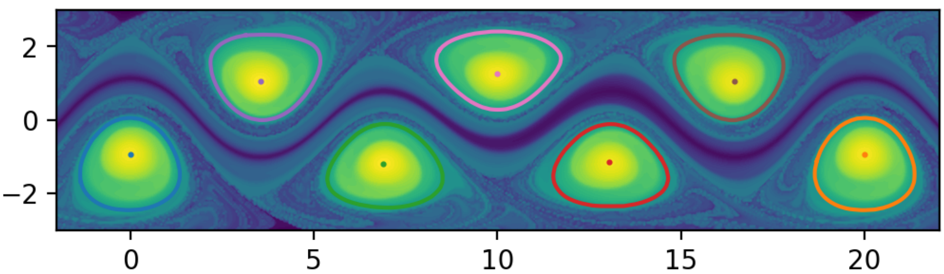
8 total threads). Warm-up time[1] is not included in the timings.

Analytical Flow (Double Gyre)



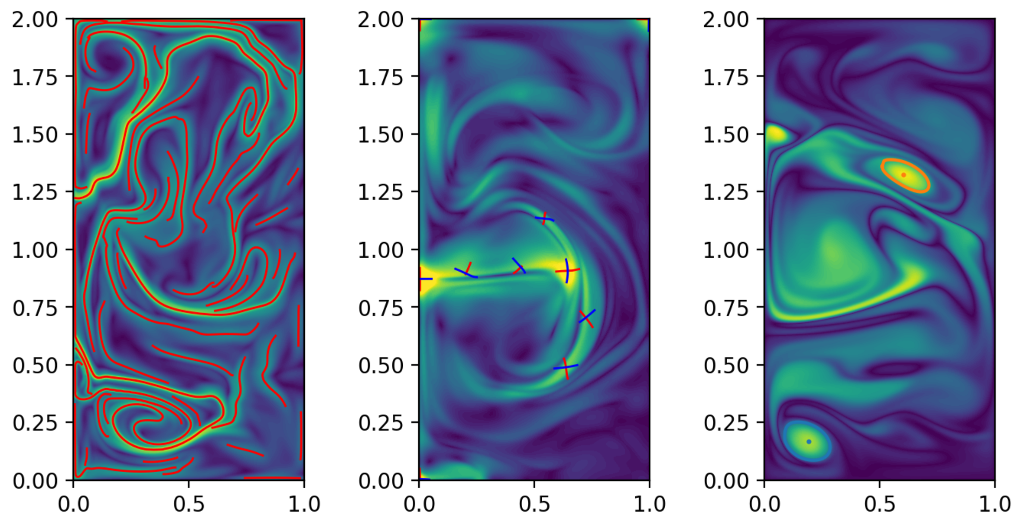
Left: DG FTLE ridges at $t_0 = 0$, integration time T = -10. Total runtime per iterate: ~0.424s (flowmap: ~0.390s; C eig: ~0.025s; FTLE ridge extraction: ~0.009s). Right: DG hyperbolic LCS at $t_0 = 0$, integration time T = -10. Total runtime per iterate: ~5.219s (flowmap (aux grid): ~1.83s; C eig (aux grid): ~0.039s; hyperbolic LCS extraction: ~3.350s). Both are computed over a 401x201 grid.

Analytical Flow (Bickley jet)



Bickley jet elliptic LCS at $t_0 = 0$, integration time T = 40 days. Total runtime per iterate: ~9.200s (flowmap: ~5.050s; LAVD: ~4.140s; elliptic LCS extraction: ~0.010s). Computed over 482x121 grid.

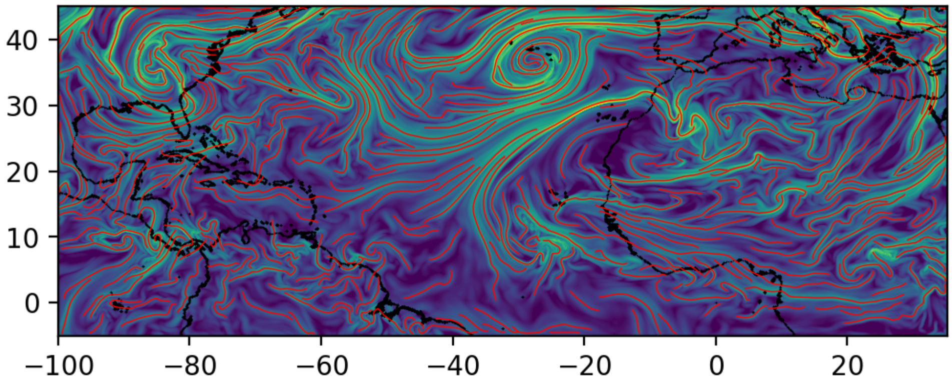
Numerical Flow (QGE)



[1] Since many functions in `NumbaCS` are JIT compiled, these functions are optimized and compiled into machine code on the first function call. This initial delay is often referred to as "warm-up time". After the first call, subsequent function calls are much faster.

Left: QGE FTLE ridges at $t_0 = 0$, integration time $T = 0.1$. Total runtime per iterate: ~2.461s (flowmap: ~2.400s; C eig: ~0.038s; FTLE ridge extraction: ~0.023s). Middle: QGE hyperbolic OECS at $t_0 = 0.15$. Total runtime per iterate: ~2.238s (S eig: ~0.038s; hyperbolic OECS extraction: ~2.200s). Right: QGE elliptic OECS at $t_0 = 0.5$. Total runtime per iterate: ~0.0452s (IVD: ~0.0002s; elliptic OECS extraction: ~0.045s). All are computed over a 257x513 grid.

Numerical Flow (MERRA-2)



MERRA-2 FTLE ridges at t_0 = 06/16/2020-00:00, integration time T = -72hrs. Total runtime per iterate: ~7.835s (flowmap: ~7.480s; C eig: ~0.085s; FTLE ridge extraction: ~0.27s). Computed over 902x335 grid.

Datasets

Two datasets are provided with NumbaCS to test the functionality for flows defined by numerical velocity data. One is a numerical simulation of the quasi-geostrophic equations (QGE). We thank the authors of Mou et al. (2021) for providing us with this dataset, which was used extensively during development, and allowing a piece of the dataset to be included in the package. The full dataset was over the time span [10,81] with dt = 0.01. We provide the velocity fields over the much shorter time span of [10,11] with the same dt. For details on parameters used in the simulation, refer to the cited paper. The other dataset is a MERRA-2 vertically averaged reanalysis dataset (Gelaro et al., 2017; GMAO, 2015), which was used as part of a paper (Jarvis et al., 2024) coauthored by the authors of this paper. Wind velocity fields were vertically averaged over pressure surfaces ranging from 500 hPa to 800 hPa. The corresponding latitude, longitude, and date arrays are also provided. All data can be downloaded from the data folder on the GitHub page.

Usage in ongoing research

As of the writing of this paper, `NumbaCS` has not been public for long but has been utilized in one publication (Jarvis et al., 2024) where it was the computational tool for all coherent structure methods. In addition, it is currently being used in an ongoing project focused on airborne invasive species traveling from Australia to New Zealand titled "Protecting Aotearoa from wind-dispersed pests". This is a five year (October 2023 - October 2028) Scion-led and Ministry of Business, Innovation and Employment (MBIE)-supported program.

Acknowledgments

This work was partially supported by the National Science Foundation (NSF) under grant number 1821145 and the National Aeronautics and Space Administration (NASA) under grant number 80NSSC20K1532 issued through the Interdisciplinary Research in Earth Science (IDS) and Biological Diversity & Ecological Conservation programs.

References

Bartos, A. P. E., Kaszás, B., & Haller, G. (2022). *Haller-group/TBarrier: TBarrier* (Version v1.0.0). Zenodo. https://doi.org/10.5281/zenodo.6779400

Briol, F., & d'Ovidio, F. (2011). Lagrangian. In *GitHub repository*. GitHub. https://github.com/CNES/aviso-lagrangian

Brunton, S. L., & Rowley, C. W. (2010). Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *20*(1), 017503. https://doi.org/10.1063/1.3270044

Curbelo, J., & Rypina, I. I. (2023). A Three Dimensional Lagrangian Analysis of the Smoke Plume From the 2019/2020 Australian Wildfire Event. *Journal of Geophysical Research: Atmospheres*, *128*(21), e2023JD039773. https://doi.org/10.1029/2023JD039773

Dabiri, J. (2009). *LCS MATLAB Kit*. https://dabirilab.com/software

Du Toit, P. C. (2010). *Transport and Separatrices in Time-Dependent Flows* [PhD thesis, California Institute of Technology]. https://resolver.caltech.edu/CaltechTHESIS:10072009-165901284

Du Toit, P. C., & Marsden, J. E. (2010). Horseshoes in hurricanes. *Journal of Fixed Point Theory and Applications*, *7*(2), 351–384. https://doi.org/10.1007/s11784-010-0028-6

Farazmand, M., & Haller, G. (2012). Computing Lagrangian coherent structures from their variational theory. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *22*(1), 013128. https://doi.org/10.1063/1.3690153

Gelaro, R., McCarty, W., Suarez, M. J., Todling, R., Molod, A., Takacs, L., Randles, C. A., Darmenov, A., Bosilovich, M. G., Reichle, R., Wargan, K., Coy, L., Cullather, R., Draper, C., Akella, S., Buchard, V., Conaty, A., Silva, A. M. da, Gu, W., … Zhao, B. (2017). The Modern-Era Retrospective Analysis for Research and Applications, Version 2 (MERRA-2). *Journal of Climate*, *30*(14), 5419–5454. https://doi.org/10.1175/JCLI-D-16-0758.1

GMAO. (2015). *Global Modeling and Assimilation Office, MERRA-2 inst3_3d_asm_Np: 3d, 3-Hourly, Instantaneous, Pressure-Level, Assimilation, Assimilated Meteorological Fields V5.12.4, Greenbelt, MD, USA, Goddard Earth Sciences Data and Information Services Center (GES DISC), Accessed on 09-22-2023*. https://doi.org/10.5067/QBZ6MG944HW0

Günther, T., Horváth, Á., Bresky, W., Daniels, J., & Buehler, S. A. (2021). Lagrangian Coherent Structures and Vortex Formation in High Spatiotemporal-Resolution Satellite Winds of an Atmospheric Kármán Vortex Street. *Journal of Geophysical Research: Atmospheres*, *126*(19), e2021JD035000. https://doi.org/10.1029/2021JD035000

Haller, G. (2011). A variational theory of hyperbolic Lagrangian Coherent Structures. *Physica D: Nonlinear Phenomena*, *240*(7), 574–598. https://doi.org/10.1016/j.physd.2010.11.010

Haller, G., & Beron-Vera, F. J. (2013). Coherent Lagrangian vortices: the black holes of turbulence. *Journal of Fluid Mechanics*, *731*, R4. https://doi.org/10.1017/jfm.2013.391

Haller, G., Hadjighasem, A., Farazmand, M., & Huhn, F. (2016). Defining coherent vortices objectively from the vorticity. *Journal of Fluid Mechanics*, *795*, 136–173. https://doi.org/

10.1017/jfm.2016.151

Haller, G., & Poje, A. C. (1998). Finite time transport in aperiodic flows. *Physica D: Nonlinear Phenomena*, *119*(3), 352–380. https://doi.org/10.1016/S0167-2789(98)00091-8

Jarvis, A., Hossein Mardi, A., Foroutan, H., & Ross, S. D. (2024). Atmospheric transport structures shaping the "Godzilla" dust plume. *Atmospheric Environment*, *333*, 120638. https://doi.org/10.1016/j.atmosenv.2024.120638

Junge, O., Diego, A. de, Karrasch, D., & Schilling, N. (2020). CoherentStructures.jl. In *GitHub repository*. GitHub. https://github.com/CoherentStructures/CoherentStructures.jl

Lagares, C., & Araya, G. (2023). A GPU-Accelerated Particle Advection Methodology for 3D Lagrangian Coherent Structures in High-Speed Turbulent Boundary Layers. *Energies*, *16*(12), 4800. https://doi.org/10.3390/en16124800

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*. https://doi.org/10.1145/2833157.2833162

Liu, Y., Wilson, C., Green, M. A., & Hughes, C. W. (2018). Gulf Stream Transport and Mixing Processes via Coherent Structure Dynamics. *Journal of Geophysical Research: Oceans*, *123*(4), 3014–3037. https://doi.org/10.1002/2017JC013390

Mathur, M., Haller, G., Peacock, T., Ruppert-Felsot, J. E., & Swinney, H. L. (2007). Uncovering the Lagrangian Skeleton of Turbulence. *Phys. Rev. Lett.*, *98*, 144502. https://doi.org/10.1103/PhysRevLett.98.144502

Mou, C., Wang, Z., Wells, D. R., Xie, X., & Iliescu, T. (2021). Reduced Order Models for the Quasi-Geostrophic Equations: A Brief Survey. *Fluids*, *6*(1). https://doi.org/10.3390/fluids6010016

Nolan, P. (2024). Dynlab. In *GitHub repository*. GitHub. https://github.com/hokiepete/dynlab

Nolan, P., Foroutan, H., & Ross, S. D. (2020). Pollution Transport Patterns Obtained Through Generalized Lagrangian Coherent Structures. *Atmosphere*, *11*(2). https://doi.org/10.3390/atmos11020168

Nolan, P., Serra, M., & Ross, S. D. (2020). Finite-time Lyapunov exponents in the instantaneous limit and material transport. *Nonlinear Dyn*, *100*, 3825–3852. https://doi.org/10.1007/s11071-020-05713-4

Onu, K., Huhn, F., & Haller, G. (2015). LCS Tool: A computational platform for Lagrangian coherent structures. *Journal of Computational Science*, *7*, 26–36. https://doi.org/10.1016/j.jocs.2014.12.002

Peacock, T., & Haller, G. (2013). Lagrangian coherent structures: The hidden skeleton of fluid flows. *Physics Today*, *66*(2), 41–47. https://doi.org/10.1063/PT.3.1886

Pretorius, I., Schou, W. C., Richardson, B., Ross, S. D., Withers, T. M., Schmale III, D. G., & Strand, T. M. (2023). In the wind: Invasive species travel along predictable atmospheric pathways. *Ecological Applications*, *33*(3), e2806. https://doi.org/10.1002/eap.2806

Rutherford, B., Dangelmayr, G., & Montgomery, M. T. (2012). Lagrangian coherent structures in tropical cyclone intensification. *Atmospheric Chemistry and Physics*, *12*(12), 5483–5507. https://doi.org/10.5194/acp-12-5483-2012

Schindler, B., Peikert, R., Fuchs, R., & Theisel, H. (2012). Ridge Concepts for the Visualization of Lagrangian Coherent Structures. In R. Peikert, H. Hauser, H. Carr, & R. Fuchs (Eds.), *Topological methods in data analysis and visualization II: Theory, algorithms, and applications* (pp. 221–235). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-23175-9_15

Serra, M., & Haller, G. (2016). Objective Eulerian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *26*(5), 053110. https://doi.org/10.1063/1.4951720

Serra, M., Sathe, P., Beron-Vera, F., & Haller, G. (2017). Uncovering the Edge of the Polar Vortex. *Journal of the Atmospheric Sciences*, *74*(11), 3871–3885. https://doi.org/10.1175/JAS-D-17-0052.1

Shadden, S. C., Lekien, F., & Marsden, J. E. (2005). Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, *212*(3), 271–304. https://doi.org/10.1016/j.physd.2005.10.007

Steger, C. (1998). An unbiased detector of curvilinear structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(2), 113–125. https://doi.org/10.1109/34.659930

Winant, P., Coleman, C., & Lyon, S. (2017). Interpolation.py. In *GitHub repository*. GitHub. https://github.com/EconForge/interpolation.py

Wogan, N. (2021). Numbalsoda. In *GitHub repository*. GitHub. https://github.com/Nicholaswogan/numbalsoda