libcfd2lcs: A general purpose library for computation of Lagrangian coherent structures during CFD simulation

Justin Finn, School of Engineering, University of Liverpool, Liverpool UK

Romain Watteaux, Stazione Zoologica Anton Dohrn, Naples Italy

Andrew Lawrie, School of Engineering, University of Bristol, Bristol UK

February 16, 2017

Abstract

This report describes *libcfd2lcs*, a general purpose numerical library that performs the calculations needed to extract *Lagrangian Coherent Structures* (LCS) from time dependent flows. *libcfd2lcs* is designed to work with two or three dimensional hydrodynamic datasets produced by computational fluid dynamics (CFD) simulations or experimental measurements. Critically, it is capable of interfacing directly with a distributed memory CFD application, allowing for parallel, *on-the-fly* calculation of LCS. The library has been integrated with several research flow solvers and tested for a variety of flows. The additional computational overhead introduced by the library can vary significantly (4-44% additional wall clock time in our tests) depending on the test/code, which we consider as not prohibitive. The library shows good scaling efficiency (> 80%) for up to 1024 compute cores on ARCHER.

1. Introduction

A number of diagnostics have been proposed in recent years that allow transport and mixing patterns to be objectively defined in unsteady or chaotic flows. One such family of diagnostics, coined *Lagrangian coherent structures* (LCS) [8], represent the skeleton of tracer trajectories and separate time dependent flows into regions of dynamically distinct behavior [6, 7, 17]. When properly defined, they act as the locally most attracting or repelling material surfaces in a flow, across which no mixing or transport occurs. In addition, as boundaries of dynamically distinct regions they provide an exceptional tool with which to visualize and understand coherence in complex fluid motion. Although definitions and techniques continue to evolve [2, 7], identification of LCS from experimentally measured or numerically simulated flows has most often relied on the relation of these special material surfaces to ridges in the finite-time Lyapunov exponent (FTLE) field. The FTLE is an Eulerian measure that quantifies the local rate of fluid stretching between initially adjacent Lagrangian trajectories. As a LCS diagnostic, FTLE ridges have led to new understanding of a variety of geophysical, biological, aerospace, and industrial flows [9, 14, 15, 16].

Typical computation of the FTLE requires the advection of Lagrangian tracer particles in the time dependent flow in order to compute a *flow map* for a given time interval, $T = [t_0,\ t_1]$. The flow map is then differentiated and used to compose the Cauchy-Green deformation tensor, whose principal eigenvalue is related to the FTLE field [17]. For large-scale, complex, three dimensional flows of general engineering interest, an enormous number of tracer particles can be required to provide a high quality flow map from which well-defined LCS can be extracted. In addition, for aperiodic, time dependent flows, the LCS are also time dependent and should be recomputed with a new set of tracer integrations in order to reveal their motion in time. Further complicating matters is the fact that the LCS should be computed in both forward ($t_1 > t_0$) and backward ($t_1 < t_0$) time in order to reveal both stable (repelling) and unstable (attracting) surfaces in the flow. When done as part of a post-processing procedure, these factors can demand a significant computational resource, and have so far limited the application of LCS to mostly academic problems.

These challenges help to illustrate several scientific and computational benefits of the integrated approach, recently proposed by Finn and Apte [4], where LCS diagnostics are computed *on-the-fly* during a computational fluid dynamics (CFD) simulation. The key benefit of this approach is that the LCS computation has access to the full spatio-temporal resolution of the simulation, thereby reducing integration errors and producing LCS with very high detail. Their technique utilized advanced techniques for flow map composition [1, 11] in order to incorporate the computation of LCS within the CFD solver in a minimally intrusive way. Crucially, the additional overhead relative to the flow solver alone (an unstructured DNS/LES code [12]) was modest (15-30%), and it was possible to maintain good parallel scalability of the overall simulation with the added LCS computations.

This report describes *libcfd2lcs*, a new computational library that provides a general implementation this integrated approach, allowing it to be utilized by a variety of different hydrodynamic solvers. Users of the library create *LCS diagnostics* through a simple but flexible Application Program Interface (API), and the library updates the diagnostics as the user's CFD simulation evolves. By harnessing the large scale parallelism of platforms

like ARCHER, *libcfd2lcs* enables CFD practitioners to make LCS a standard analysis tool, even for large scale, three dimensional data sets.

2. Computing the FTLE

To compute the FTLE, *libcfd2lcs* works with velocity fields, $\boldsymbol{u}(\boldsymbol{x}, t)$, defined in two or three dimensional domains. If we consider the time interval, $t \in (t_0, t_1)$, the flow map, $\boldsymbol{\Phi}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)$ is the function that integrates passive tracers from their initial position, $\boldsymbol{x}_0$, at time t_0 along pathlines to their "advected" position, $\boldsymbol{x}$, at time t_1,

$$\boldsymbol{\Phi}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0) = \boldsymbol{x}_0 + \int_{t_0}^{t_1} \boldsymbol{u}(\boldsymbol{x}(\tau), \tau) d\tau. \tag{1}$$

The flow map is then differentiated and used to compose the right Cauchy-Green deformation tensor,

$$\boldsymbol{C}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0) = \left[\boldsymbol{D}\boldsymbol{\Phi}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)\right]^* \left[\boldsymbol{D}\boldsymbol{\Phi}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)\right]. \tag{2}$$

Here, $\boldsymbol{D}\boldsymbol{\Phi}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)$ is the Jacobian of the flow map evaluated at the initial tracer coordinate, $\boldsymbol{x}_0$ and * denotes the transpose. The principal eigenvalue of $\boldsymbol{C}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)$ characterizes the amount of stretching over this time interval of initially adjacent trajectories at $\boldsymbol{t}_0$, and the average rate of repulsion is given by the corresponding FTLE,

$$\sigma_{t_0}^{t_1}(\boldsymbol{x}_0, t_0) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{max}\left(\boldsymbol{C}_{t_0}^{t_1}(\boldsymbol{x}_0, t_0)\right)}. \tag{3}$$

The FTLE may be computed in *forward time* ($t_1 > t_0$), or *backward time* ($t_1 < t_0$) in order to reveal repelling and attracting LCS respectively. To compute the backward time FTLE during a CFD simulation that progresses only forward in time, an Eulerian level-set treatment [4, 11], is used to extract the backward time flow map in *libcfd2lcs*.

In practice, it is useful to construct the time T flow maps, $\boldsymbol{\Phi}_{t_0}^{t_0+T}(\boldsymbol{x}_0, t_0)$, from a sequence of N smaller time h flow maps, where h=T/N [1, 13]. This removes the need to perform redundant tracer integrations when many concurrently evolving FTLE fields must be computed (for example, to animate their evolution). Rather than store these flow

map sub steps in memory, they are written to temporary binary files on hard disk and used to construct the time T flow map as needed [1].

In view of more recent theoretical developments [7, 9], FTLE ridges should be viewed strictly as *LCS candidates*, which can have the full properties of LCS under sufficient additional conditions. Nonetheless, they have been broadly utilized and can be computed in both two or three dimensions, whereas more recent definitions of LCS are not easily computed in 3D [2]. Future extensions of *libcfd2lcs* to include more rigorous LCS diagnostics should be possible, since most new definitions appear to also be based on properties of the Cauchy-Green deformation tensor.

3. Implementation

Although the work of Finn and Apte [4] laid the foundation for the present work, *libcfd2lcs* was implemented from scratch, and represented a significant code development effort. The goal was to make *libcfd2lcs* a flexible but high performance "black box" for computing LCS diagnostics, and to make the library compatible with a wide variety of existing flow solvers. The library's full API is accessible from user applications written in either Fortran or C, and the *libcfd2lcs* user's manual [3] contains a complete syntax specification and detailed description of each user accessible function. During development, the library was tested using several analytically defined unsteady flows in two and three dimensions, including Arnold-Beltrami-Childress (ABC) flow and double gyre flow. These flows provided the basis for a suite of example programs (written in both C and Fortran) that are distributed with the library, which should help new users to develop their own applications with *libcfd2lcs*.

3.1. The *libcfd2lcs* API

Algorithm 1 shows a typical usage pattern of *libcfd2lcs* by a CFD application. After the user performs the usual initialization of their flow solver, a call is first made to `cfd2lcs_init` to initialize the library's data structure and parallel communications. The next calls to `cfd2lcs_set_option` and `cfd2lcs_set_param` are used to set user-

accessible options (ie, the interpolation function, or the integration scheme). The user may then initialize any number of LCS diagnostics (ie. forward or backward FTLE fields, particle tracers, etc) using calls to `cfd2lcs_diagnostic_init`. Once the user's main flow solver loop begins, calls to `cfd2lcs_update` utilize the latest velocity field to update all LCS diagnostics. The user can stop the computation of specific diagnostics during the simulation using a call to `cfd2lcs_diagnostic_destroy`, and can gracefully stop all *libcfd2lcs* functionality using a call to `cfd2lcs_finalize`.

Algorithm 1: Typical usage of the *libcfd2lcs* API.

```
! Start Of User's Application
Establish user's grid coordinates
Establish user's boundary conditions
Establish user's domain decomposition
call cfd2lcs_init(mpicomm,n,offset,x,y,z,bcflag)
call cfd2lcs_set_option(option,val)
call cfd2lcs_set_param(param,val)
call cfd2lcs_diagnostic_init(id,type,res,T,H,label)
! User's main flow solver loop
t = t_start;
while t < t_finish do
    Establish new velocity field at time t
    call cfd2lcs_update(n,u,v,w,t,cfl)
    if Done With Diagnostic then
        call cfd2lcs_diagnostic_destroy(id);
    t = t + dt
call cfd2lcs_finalize
! End Of User's Application
```

3.2. Parallelization and data structure

The library is fully parallelized using a distributed memory model and MPI. To accomplish this, it was necessary to place some restrictions on the type of grid and velocity data that the library can work with. Specifically, the data must be:

- **Globally structured**: Data must be organized globally in an i, j, k structured format. Non-rectangular domains (corners, holes, etc), non rectilinear grids, and

non-Cartesian geometries (ie, spherical coordinates) can be handled. An example of using the library with non-rectangular boundaries and spherical coordinates can be found in the `/examples/ROMS/` directory.

- **Co-located**: All components of the velocity (u, v, w) must be located at the same points in space. Staggered grid velocity data should therefore be interpolated to the cell centers before passing to *libcfd2lcs*.

- **Block-partitioned**: Distributed memory partitions must be subdomains with edges along constant i, j, or k indices.

Future extensions will hopefully relax some of these restrictions.

3.3. I/O

The library supports parallel read/write operations using either the HDF5 library or its own native ASCII read/write routines. The HDF5 routines offer large scale parallel scalability using a single file / multiple writers strategy and are recommended. For systems without HDF5 support, or for debugging purposes, the ASCII I/O routines (single file /single writer) may be used. Both file formats are readable by common visualization programs (Matlab, Tecplot, etc).

4. Results

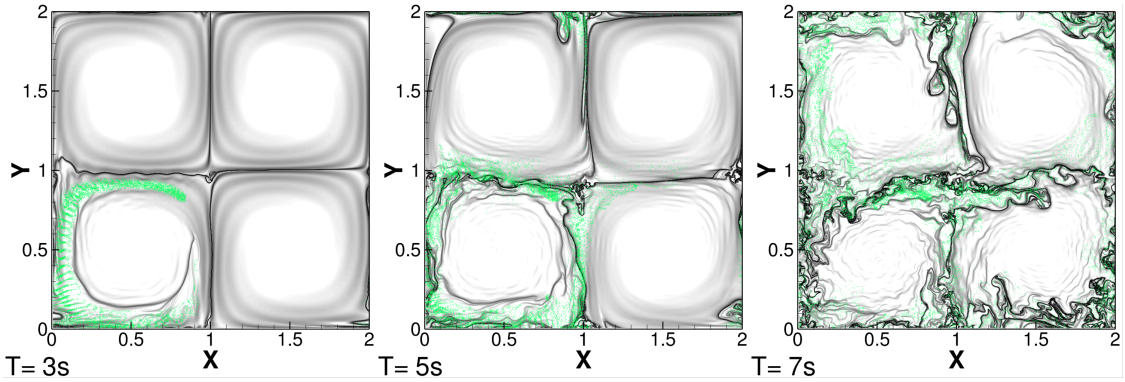
The library was tested with three different flow solvers to ensure that *libcfd2lcs* and its API are compatible with a variety of scientific codes. In Table 1 and Figure 1 below, we provide an overview of case studies that were completed. The overhead (wall clock) associated with the *libcfd2lcs* computations relative to the entire simulation is reported in Table 1, and can vary significantly depending on the problem/code considered. The decay of Taylor vortices induced by entrainment of inertial particles (Figure 1a) was computed using the CGS-DEM code [5]. This is a relatively expensive simulation, involving computation of particle-particle collisions and particle-fluid coupling. Computing both

forward and backward time FTLE fields for integration time $T = 3s$ (3 times the integral timescale of the flow) and frequency $T/h = 10$ consumes only 4% of the total simulation time. On the other hand, the turbulent jet simulation (Fig. 1b) performed with MOBILE [10] and three dimensional isotropic turbulence simulation (Fig. 1c) performed with TurboTrack3D [22] are themselves much faster. This means the LCS related computations for similar T/h consume roughly 40% of the simulation time in these cases. Although this is certainly not a negligible added overhead, it should be viewed in the context of added value that high quality LCS results can provide.

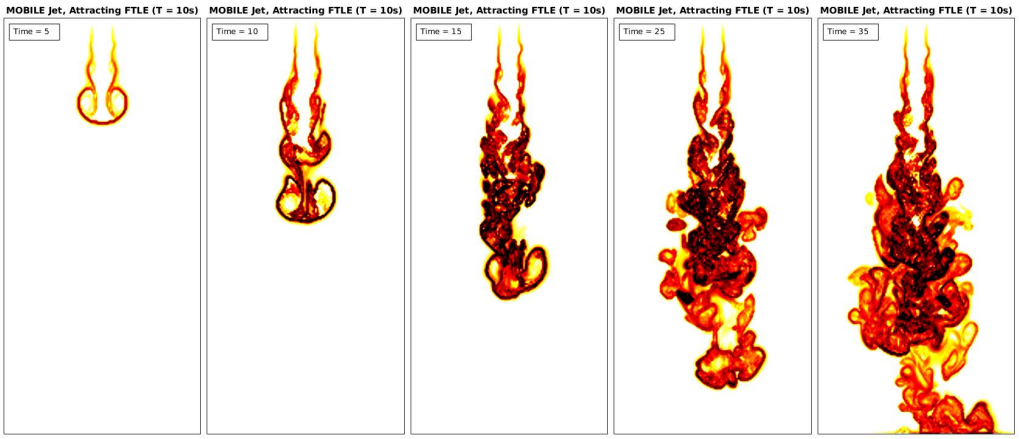
Table 1: Summary of *libcfd2lcs* test cases, including the library overhead (% of simulation wall clock) where applicable.

Flow solver	Test case	*libcfd2lcs* overhead
Analytic flow field	ABC flow	-
Analytic flow field	Double gyre flow	-
CGS-DEM (U. of Liverpool)	Particle laden Taylor vortices	4%
MOBILE (U. of Bristol)	Turbulent Jet	37%
TurboTrack3D (KTH, Stockholm & SZN, Naples)	3D Isotropic Turbulence	44%
Dataset (SZN, Naples)	Tyrrhenian sea dataset	-

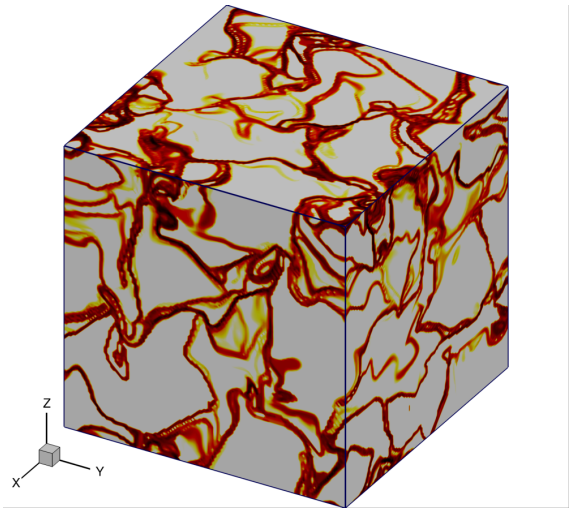
Additionally, the library was used to compute LCS from a regional ocean model simulation (ROMS) [18, 19] dataset of the Tyrrhenian sea (Catanese et al, Submitted). Snapshots of the sea surface velocity at 1 day resolution were used to compute the attracting FTLE fields for an integration time of $T = 7$days. A spherical to Cartesian coordinate transformation was applied to the data before passing it to the library, and the sea-land boundary was handled using the library's `MASK` boundary type. The computation was distributed on 16 cores. The sharp ridges shown in the FTLE field (Fig. 1d) illustrate the capability of *libcfd2lcs* for post-processing existing datasets.



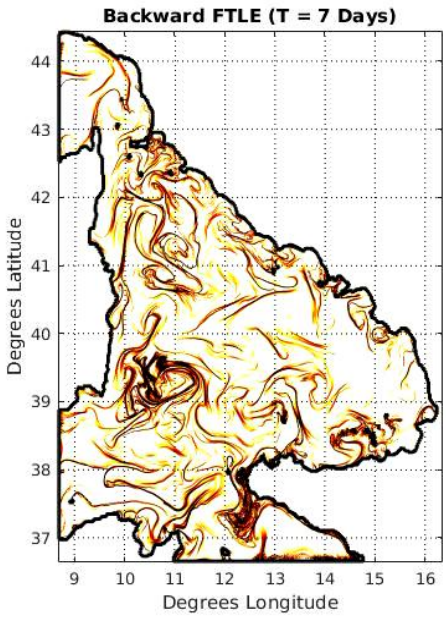
(a) Turbulent breakdown of 2D Taylor vortices induced by inertial particles (green points) (CGS-DEM)



(b) Three dimensional turbulent Jet, center plane shown (MOBILE)



(c) 3D isotropic turbulence (TTrack3D)

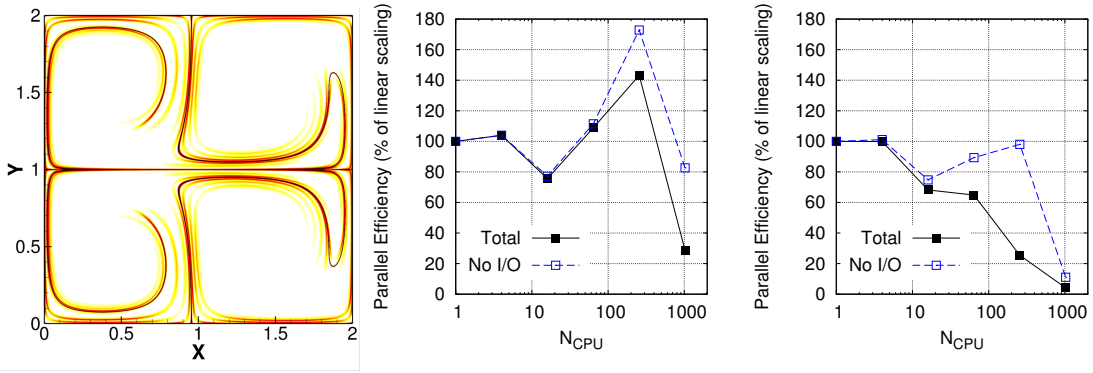


(d) Tyrrhenian Sea surface currents (ROMS dataset)

Figure 1: Overview of results obtained with *libcfd2lcs*. Each figure shows contours of the backward time FTLE field. Dark ridges indicate location of attracting LCS candidates.

4.1. Parallel performance

The parallel scalability of *libcfd2lcs* was evaluated on ARCHER using two dimensional analytic double gyre flow [20]. The domain is discretized with a uniform, 1024^2 Cartesian mesh in the region $[0, 2] \times [0, 2]$. The flow is started at at time $t = 0$, and at advanced to $t = 25s$, with a constant time step of $\Delta t = 0.01$. In two separate trials, forward and backward time FTLE fields are computed with a temporal resolution $T/h = 1$ and $T/h = 10s$ for $T = 15s$. A representative forward time FTLE field is shown in Figure 2a



(a) Double Gyre Flow (b) Strong scaling: $T/h = 1$ (c) Strong scaling: $T/h = 10$

Figure 2: Parallel performance of *libcfd2lcs*. (a) shows contours of forward time FTLE field for the analytic double gyre flow. (b) and (c) show strong scaling efficiency as a percentage of linear efficiency for a 1024^2 problem with $T/h = 1$ and $T/h = 10$ respectively. Both total efficiency and efficiency excluding I/O are shown.

Each trial was repeated using 1, 4, 16, 64, 256, and 1024 processors on ARCHER. The strong scaling efficiency, $\mathrm{E} = t_1 / (N_{CPU} \cdot t_N) \times 100$ is plotted vs the number of cores, N_{CPU}, for the two different temporal resolutions in Figure 2b-c, with and without I/O costs. In both cases, excellent efficiency is obtained up to 256 cores when I/O is excluded from the figure. We presently cannot explain the drop in efficiency at 16 cores, or the super-linear speedup observed for the $T/h = 1$ trial at 256 cores. In the case of $T/h = 1$, greater than 80% scaling efficiency is maintained up to 1024 cores. For $T/h = 10$ (Fig 2c), it appears that the degradation in the total efficiency for $N_{CPU} > 100$ is due in large part to I/O. Future work is needed in order to determine the optimal domain size per core for read/write operations, and optimal systems settings such as the file striping parameters.

The poor scaling for $T/h = 10$ at $N_{CPU} = 1024$ is also due in part to costs associated with constructing the time T flow map from time h substeps. It appears that for large core counts, interpolation errors between subsequent flow-map substeps can sometimes result in particles being erroneously tracked to the wrong processor's subdomain, and the procedure for correcting this does not scale well. A robust solution to this problem will be the focus of future efforts.

5. Obtaining a copy of *libcfd2lcs*

libcfd2lcs is installed as a public module on ARCHER and can be loaded with the command `module load libcfd2lcs/1.0`. The source code is also available under the terms of the GNU public license and can be downloaded from pcwww.liv.ac.uk/~finnj/code. A minimal list of prerequisites for building the library include i) MPI Fortran and C compiler, ii) `liblapack`, iii) `libhdf5` (optional). GNU Make is used for building the library and platform dependent settings are defined in a `Makefile.in` for portability. A user manual [3] and several example programs written in F90 and C are included in the distribution.

6. Conclusions and future work

libcfd2lcs is a new computational library that enables researchers to integrate LCS analysis in their fluid dynamics research by providing a toolbox for computing LCS on-the-fly while a CFD simulation runs. This approach has several advantages over the traditional post-processing approach to computing LCS, and should remove a significant barrier to entry for deciphering unsteady and chaotic flow dynamics using LCS. The library has been designed to be a general purpose tool, with a limited but easy to use API, and has been tested with several research flow solvers. The relative overhead associated with the library's computations is problem specific and can vary with the flow solver, but has not been found to be prohibitive. Good parallel scaling efficiency has been demonstrated for up to 1024 cores, and further improvements in the flow-map construction algorithm as

well as I/O tuning should enable larger scale-ups.

Development of *libcfd2lcs* is an ongoing effort, and we welcome contributions from collaborators wanting to further develop its capabilities. Planned extensions include computation of *inertial LCS* for particles with finite size and density, computations of *strainlines* and *stretchlines* following the recent theory described in [9], and generalization of the approach to work with fully unstructured flow-solvers.

7. Acknowledgements

This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service http://www.archer.ac.uk. The high order interpolation methods in the library are based on a slightly modified version of the multidimensional `BSPLINE-FORTRAN` library, provided by Jacob Williams [21]

References

[1] S.L. Brunton and C.W. Rowley. Fast computation of finite-time Lyapunov exponent fields for unsteady flows. *Chaos*, 20(1):017503, 2010.

[2] M. Farazmand and G. Haller. Computing Lagrangian coherent structures from their variational theory. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22 (1):013128–013128, 2012.

[3] J. Finn, R. Watteaux, and A. Lawrie. libcfd2lcs user's manual. *Available online at http://pcwww.liv.ac.uk/~finnj/code.html*, 2016.

[4] Justin Finn and Sourabh V. Apte. Integrated computation of finite time Lyapunov exponent fields during direct numerical simulation of unsteady flows. *Chaos*, 23(1): 013145, 2013.

[5] Justin R Finn, Ming Li, and Sourabh V Apte. Particle based modelling and simula-

tion of natural sand dynamics in the wave bottom boundary layer. *Journal of Fluid Mechanics*, 796:340–385, 2016.

[6] G. Haller. Distinguished material surfaces and coherent structures in three-dimensional fluid flows. *Physica D: Nonlinear Phenomena*, 149(4):248–277, 2001.

[7] G. Haller. A variational theory of hyperbolic Lagrangian coherent structures. *Physica. D*, 240(7):574–598, 2011.

[8] G. Haller and G. Yuan. Lagrangian coherent structures and mixing in two-dimensional turbulence. *Physica D: Nonlinear Phenomena*, 147(3-4):352–370, 2000.

[9] George Haller. Lagrangian coherent structures. *Annual Review of Fluid Mechanics*, 47:137–162, 2015.

[10] Andrew G W Lawrie. Mobile, 2008.

[11] S. Leung. An Eulerian approach for computing the finite time Lyapunov exponent. *Journal of Computational Physics*, 230(9):3500–3524, 2011.

[12] P. Moin and SV Apte. Large-eddy simulation of realistic gas turbine-combustors. *AIAA Journal*, 44(4):698–708, 2006. ISSN 0001-1452.

[13] J.M. Ottino. *The kinematics of mixing: stretching, chaos, and transport*. Cambridge University Press, 1989.

[14] T Peacock, G Froyland, and G Haller. Introduction to focus issue: Objective detection of coherent structures. *Chaos*, 25(8):7201, 2015.

[15] Thomas Peacock and John Dabiri. Introduction to focus issue: Lagrangian coherent structures. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(1): 017501, 2010.

[16] RM Samelson. Lagrangian motion, coherent structures, and lines of persistent material strain. *Annual review of marine science*, 5:137–163, 2013.

[17] S.C. Shadden, F. Lekien, and J.E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3-4):271–304, 2005.

[18] Alexander F Shchepetkin and James C McWilliams. A method for computing horizontal pressure-gradient force in an oceanic model with a nonaligned vertical coordinate. *Journal of Geophysical Research: Oceans*, 108(C3), 2003.

[19] Alexander F Shchepetkin and James C McWilliams. The regional oceanic modeling system (roms): a split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, 9(4):347–404, 2005.

[20] TH Solomon and JP Gollub. Chaotic particle transport in time-dependent Rayleigh-Bénard convection. *Physical Review A*, 38(12):6280, 1988.

[21] Jacob Williams. The BSPLINE-FORTRAN library. *Available online at https://github.com/jacobwilliams/bspline-fortran.*, 2016.

[22] Caijuan Zhan, Gaetano Sardina, Enkeleida Lushi, and Luca Brandt. Accumulation of motile elongated micro-organisms in turbulence. *Journal of Fluid Mechanics*, 739:22–36, 2014.