libcfd2lcs: A general purpose library for computing Lagrangian coherent structures during CFD simulations

Justin Finn

School of Engineering, University of Liverpool

Email: J.Finn@liverpool.ac.uk ● **URL:** `http://pcwww.liv.ac.uk/~finnj/`

ARCHER Virtual Tutorial

June 22, 2016

Lagrangian Coherent Structures (LCS)

“The structures are invisible because they often exist only as dividing lines between parts of a flow that are moving at different speeds and in different direction... They arent something you can walk up to and touch but they are not purely mathematical constructions, either... The line is not a fence or a road, but it still marks a physical barrier.”

-Prof. Jerry Marsden discussing LCS, New York Times, September 28, 2009

Lagrangian Coherent Structures are “The skeleton of water”

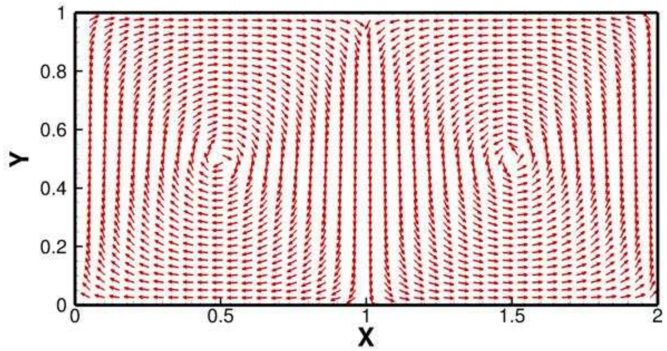
-The Economist, 2009, discussing work titled “Uncovering the Lagrangian Skeleton of Turbulence” [Mathur et al. PRL 2009]

Flow Coherence: Eulerian & Lagrangian

Even chaotic flows display remarkable levels of coherence. How this coherence looks, and what it means for transport and mixing, depends on our frame of reference.

- **Eulerian**: Observe the flow from a fixed location
- **Lagrangian**: Follow fluid parcels through space/time

Example: Time dependent "double gyre" flow [Solomon and Gollub, 1988]



Flow Coherence: Eulerian & Lagrangian

Even chaotic flows display remarkable levels of coherence. How this coherence looks, and what it means for transport and mixing, depends on our frame of reference.

- **Eulerian**: Observe the flow from a fixed location
- **Lagrangian**: Follow fluid parcels through space/time

Example: Time dependent "double gyre" flow [Solomon and Gollub, 1988]

Figure : Steady flow, no oscillation

Figure : Small transverse oscillation

Flow Coherence: Eulerian & Lagrangian

Even chaotic flows display remarkable levels of coherence. How this coherence looks, and what it means for transport and mixing, depends on our frame of reference.

- **Eulerian**: Observe the flow from a fixed location
- **Lagrangian**: Follow fluid parcels through space/time

Example: Time dependent "double gyre" flow [Solomon and Gollub, 1988]

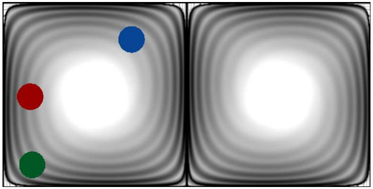


Figure : Steady flow, no oscillation

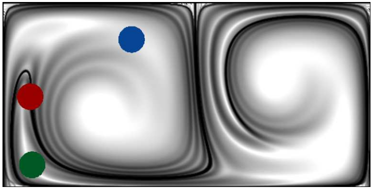


Figure : Small transverse oscillation

If we want to understand time dependent transport patterns, we should consider *Lagrangian coherent structures* (LCS).

Outline

This talk...

libcfd2lcs: A general purpose library for computing Lagrangian coherent structures during CFD simulations

1. Background: What are LCS,and how do we compute them?
2. The libcfd2lcs approach: Integrating LCS computation with CFD simulation
   - How to develop applications that use the library
   - Key library functions that every libcfd2lcs application will use.
3. Demonstrations of what the library can do

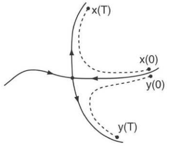
What this talk wont be...

- Detailed overview of LCS theory & applications: Several excellent reviews available:
  [Haller, 2015, Peacock et al., 2015, Peacock and Dabiri, 2010, Samelson, 2013]
- An complete, step by step guide for using the library. See the libcfd2lcs user's manual and example programs:
  `http://pcwww.liv.ac.uk/~finnj/code`

Properties of LCS

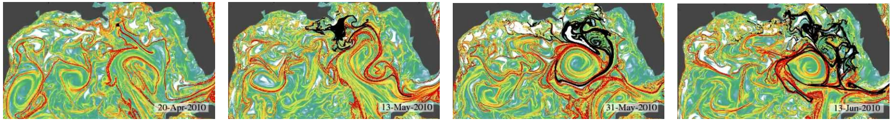
Starting roughly with [Haller, 2001], the notion of LCS has been refined

- Unsteady and turbulent flows have a **hidden skeleton** which defines mixing patterns.
- This skeleton defines boundaries of **dynamically distinct regions** within which tracer patterns behave similarly.
- This skeleton is defined by the locally most attracting or repelling **hyperbolic material lines** (or surfaces in 3D), across which **no transport** can occur.
- Can be constructed with **detailed simulation or experimental data.**

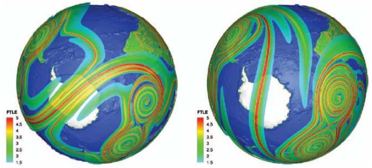


[Shadden et al., 2005

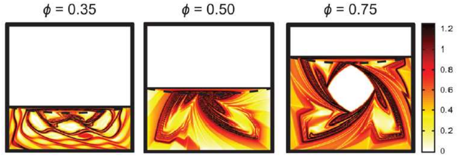
Lots of new insights:



(a) Hind-casting the movement of the Deepwater Horizon oil slick [Olascoaga and Haller, 2012]



(b) Atmospheric events [Lekien and Ross, 2010]



(c) Granular mixing [Christov et al., 2011]

Computing LCS diagnostics: Example, FTLE

A number of LCS *diagnostics* have been proposed: FTLE is imperfect, but common

1. **Release a grid of tracers and see where they go:** Provides the flow map, $\Phi(\mathbf{x}_0, t_0,)$

$$\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0) = \mathbf{x}_0 + \int_{t_0}^{t_1} \mathbf{u}(\mathbf{x}(\tau), \tau) d\tau \qquad (1)$$

2. **Differentiate the flow map W.R.T. $\mathbf{x}_0$:** compose right C.G. deformation tensor, Δ

$$\Delta_{t_0}^{t_1}(\mathbf{x}_0, t_0) = \left[\frac{d\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)}{d\mathbf{x}_0}\right]^* \left[\frac{d\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)}{d\mathbf{x}_0}\right], \qquad (2)$$

$\Downarrow$ Most (all?) LCS diagnostics

---

3. **Compute the maximum finite-time Lyapunov exponent (FTLE) field**.

$$\sigma_{t_0}^{t_1}(\mathbf{x_0}, t_0) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{max}(\Delta_{t_0}^{t_1}(\mathbf{x}_0, t_0))}, \qquad T = t_1 - t_0 \qquad (3)$$

4. **Detect FTLE ridges visually or analytically:** Forward time ridges $\Rightarrow$ repelling LCS. Backward time ridges $\Rightarrow$ attracting LCS [Haller, 2001, Shadden et al., 2005].

More recent theory has moved beyond FTLE based diagnostics, but preliminary (expensive) steps remain.

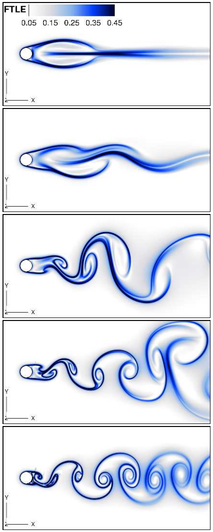
Computational challenges

Typical Approach: Algorithmically simple, *computationally expensive post-processing*

- Program reads velocity fields produced from exp/sim
- Potentially enormous number of particle advections
- Need to store lots of velocity fields
- Need to re-compute flow map for every instant LCS are needed (think of animating their movement)

Speedup possible:

- AMR [Miron et al., 2012]
- Ridge tracking [Lipinski and Mohseni, 2010]
- GPU acceleration [Conti et al., 2012]
- Re-use of flow maps [Brunton and Rowley, 2010]
- Integration with CFD [Finn and Apte, 2013]



LCS during startup of vortex shedding behind a cylinder

The integrated approach [Finn & Apte, Chaos 2013]

Ditch post processing and compute LCS on-the-fly!

- Fwd/Bkwd FTLE computed as simulation evolves
- Requires taking advantage of a few tricks
  - Flow map composition rule [Brunton and Rowley, 2010]
  - Eulerian computation of backward time flow maps [Leung, 2011]
- Modest overhead relative to cost of unsteady CFD (20-30% of simulation)

Advantages:

- Harness large scale parallelism available on HPC systems like ARCHER
- Excellent space/time resolution (minimal interpolation errors)
- Tackle larger, more complex flows using LCS in greater detail than before
- CFD and HPC capabilities will continue to grow!

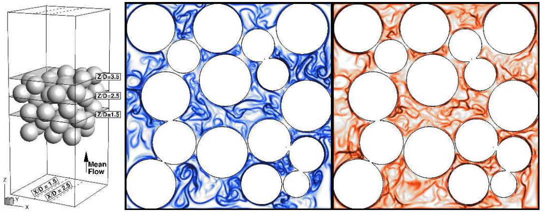


Figure : FTLE field computed from a turbulent flow through a packed bed

libcfd2lcs: General purpose numerical library for integrated LCS computations

Key capabilities:

- Easy to use, flexible API, requires little modification to many existing CFD solvers
- Allow any number of LCS diagnostics to be computed simultaneously.
- Compatible with structured grids (orthogonal, or non-orthogonal)
- User specified grid refinement
- Can be called from C/C++/Fortran
- Distributed memory, has been tested on up to 4096 cores on ARCHER.
- Add new LCS diagnostic types with modular, extensible source code

Getting libcfd2lcs

libcfd2lcs is installed on ARCHER and available to all users

- `>> module swap PrgEnv-cray PrgEnv-gnu` (Most tested environment)
- `>> module load cray-hdf5-parallel` (For parallel I/O)
- `>> module load libcfd2lcs/1.0` (Load libcfd2lcs)

Non ARCHER users...

- The library, users manual, and example programs (C & F90) available at: `http://pcwww.liv.ac.uk/~finnj/code.html`
- Minimal dependencies to install on your own workstation/HPC system.
  - `mpif90, mpicc`
  - `liblapack`
  - `hdf5` (optional, but recommended)
- Basic build instructions and Makefile provided
- Distributed under the terms of the GNU Public License

Compiling your application with libcfd2lcs

Environment variables for easy compilation

- Include the libcfd2lcs environment definitions in your application's Makefile
  `include $(CFD2LCS_HOME)/Makefile.in` (On ARCHER)
  `include /path/to/your/libcfd2lcs/Makefile.in` (In General)

- Add the include path `$(CFD2LCS_INC)` when compiling object files. For example:
  ```
  mpif90 -c -O3 $(CFD2LCS_INC) your_code.f90
  ```

- Add the single (SP) or double (DP) precision libraries to your link line:
  ```
  mpif90 -o YOUR_APPLICATION your_code.o $(CFD2LCS_SP_LIBS)
  mpif90 -o YOUR_APPLICATION your_code.o $(CFD2LCS_DP_LIBS)
  ```

- Include one of the libcfd2lcs header files in the area of your source code that interfaces with libcfd2lcs:

F90 Syntax (single prec):	`INCLUDE cfd2lcs_inc_sp.f90`
F90 Syntax (double prec):	`INCLUDE cfd2lcs_inc_dp.f90`
C Syntax (single prec):	`#include "cfd2lcs_inc_sp.h"`
C Syntax (double prec):	`#include "cfd2lcs_inc_dp.h"`

Developing applications that use libcfd2lcs

Easiest way is to follow the examples in `/libcfd2lcs/examples` directory

Pseudo-code:

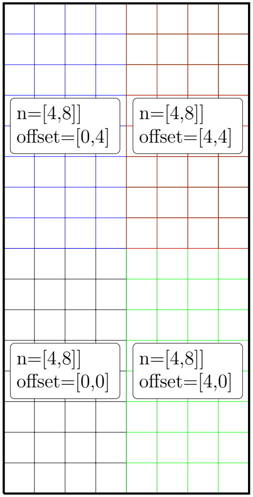
```
Start Of User's Application
INCLUDE cfd2lcs_inc_sp.f90
Establish user's grid coordinates
Establish user's boundary conditions
Establish user's domain decomposition
call cfd2lcs_init(mpicomm,n,offset,x,y,z,bcflag)
call cfd2lcs_set_option(option,val)
call cfd2lcs_set_param(param,val)
call cfd2lcs_diagnostic_init(id,type,res,T,H,label)
$t = t_{start}$;
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
```

`cfd2lcs_init`: Initialize communications & memory

F90 Syntax: `call cfd2lcs_init(comm,n,offset,x,y,z,bcflag)`
C/C++ Syntax: `cfd2lcs_init_c(comm,n,offset,x,y,z,bcflag);`

Arguments

- `comm`: Global mpi communicator in the user's application
- `n`: Vector of 3 integers defining the local number of grid points in X,Y,Z directions
- `offset`: Vector of 3 integers defining each processors offset in the the globally structured array.
- `x, y, z`: Arrays of size `n[1]*n[2]*n[3]` containing the Cartesian coordinates of each grid point.
- `bcflag`: Array of size `n[1]*n[2]*n[3]` containing a per-grid point boundary condition. `LCS_INTERNAL, LCS_WALL, LCS_SLIP, LCS_MASK, LCS_OUTFLOW, LCS_INFLOW.`

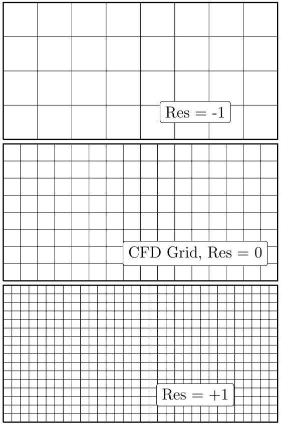


`cfd2lcs_diagnostic_init`: Initializes an LCS diagnostic to compute

F90 Syntax: `call cfd2lcs_diagnostic_init(id,type,res,T,h,label)`
C/C++ Syntax: `id = cfd2lcs_diagnostic_init_c(type,res,T,h,label);`

Arguments

- `id`: An integer handle created by the library for each diagnostic (output arg)
- `type`: The type of LCS diagnostic to initialize. Present options are `FWD_FTLE`, `BKWD_FTLE`, `LP_TRACER`
- `res`: The resolution of the LCS tracer grid, relative to the application's grid.
  - `res` = -ve integers ⇒ remove grid points
  - `res` = +ve integers ⇒ insert grid points
  - `res` = 0 ⇒ Use the application's grid
- `T`: The LCS diagnostic integration time
- `h`: Interval to compute/write the LCS diagnostic
- `label`: A string to identify the diagnostic



Example:

```
call cfd2lcs_diagnostic_init(id_1, 'BKWD_FTLE', 8.0, 1.0, 'LCS_1')
```

`cfd2lcs_set_option`: Set the value of a libcfd2lcs option

F90 Syntax: `call cfd2lcs_set_option(option,value)`
C/C++ Syntax: `cfd2lcs_set_option_c(option,value);`

Arguments

- `option`: A string matching one of the user accessible libcfd2lcs options
- `val`: An integer constant corresponding to a possible option

Example

```
call cfd2lcs_set_option('INTEGRATOR', RK3)
call cfd2lcs_set_option('INTERPOLATOR',LINEAR)
```

All user options and possible values are explained in the libcfd2lcs user's manual!

`cfd2lcs_update`: Update all LCS diagnostics

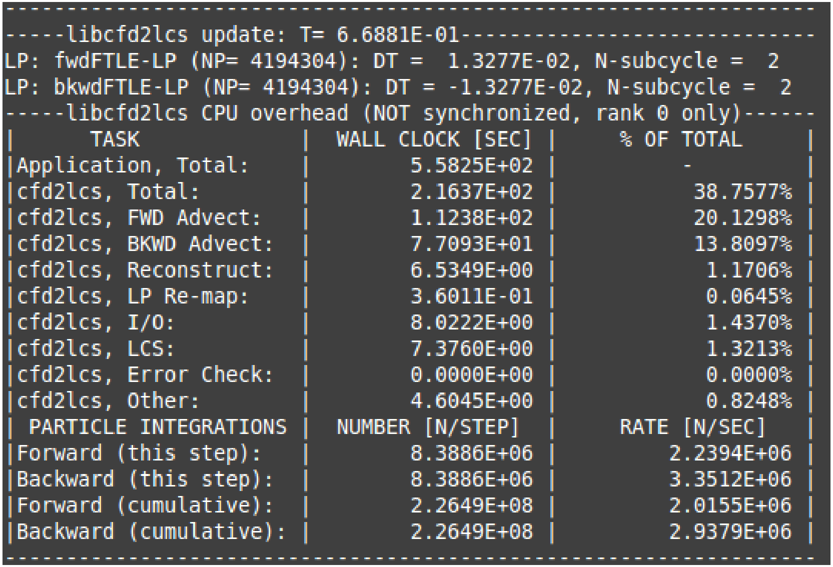
F90 Syntax: `call cfd2lcs_update(n,u,v,w,time)`
C/C++ Syntax: `id = cfd2lcs_update_c(n,u,v,w,time);`

Arguments

- `n`: Vector of 3 integers defining the local number of grid points for each processor.
- `u,v,w,`: Arrays of size `n[1]*n[2]*n[3]` containing the X,Y,Z components of velocity.
- `time`: The current simulation time

Runtime Output

Some information provided to stdout after every call to `cfd2lcs_update`



```
--------------------------------------------------------------------
-----libcfd2lcs update: T= 6.6881E-01-------------------------------
LP: fwdFTLE-LP (NP= 4194304): DT =  1.3277E-02, N-subcycle =  2
LP: bkwdFTLE-LP (NP= 4194304): DT = -1.3277E-02, N-subcycle =  2
-----libcfd2lcs CPU overhead (NOT synchronized, rank 0 only)------
|      TASK                |  WALL CLOCK [SEC] |     % OF TOTAL    |
|Application, Total:       |       5.5825E+02 |          -        |
|cfd2lcs, Total:           |       2.1637E+02 |         38.7577%  |
|cfd2lcs, FWD Advect:      |       1.1238E+02 |         20.1298%  |
|cfd2lcs, BKWD Advect:     |       7.7093E+01 |         13.8097%  |
|cfd2lcs, Reconstruct:     |       6.5349E+00 |          1.1706%  |
|cfd2lcs, LP Re-map:       |       3.6011E-01 |          0.0645%  |
|cfd2lcs, I/O:             |       8.0222E+00 |          1.4370%  |
|cfd2lcs, LCS:             |       7.3760E+00 |          1.3213%  |
|cfd2lcs, Error Check:     |       0.0000E+00 |          0.0000%  |
|cfd2lcs, Other:           |       4.6045E+00 |          0.8248%  |
| PARTICLE INTEGRATIONS    |  NUMBER [N/STEP]  |     RATE [N/SEC]  |
|Forward (this step):      |       8.3886E+06 |       2.2394E+06 |
|Backward (this step):     |       8.3886E+06 |       3.3512E+06 |
|Forward (cumulative):     |       2.2649E+08 |       2.0155E+06 |
|Backward (cumulative):    |       2.2649E+08 |       2.9379E+06 |
--------------------------------------------------------------------
```

Data Output

- All results are written to a directory called `cfd2lcs_output`.
- A new file is written for each LCS diagnostic at every time interval, *h*.
- Two I/O models are possible:
    - Parallel I/O with hdf5 (single file, multiple writers)
      Set `HDF5_SUPPORT = ``TRUE''` in `Makefile.in` when building libcfd2lcs.
    - Serial I/O (Single file, single writer)
      Set `HDF5_SUPPORT = ``FALSE''` in `Makefile.in` when building libcfd2lcs.
- Both file formats readable by common visualization programs (matlab, tecplot, etc)

Many new flows can be studied with libcfd2lcs!

Straightforward integrations into a variety of flow solvers

Successful integrations:

- Pseudo-spectral turbulence simulations (With R. Watteaux)
- Rayleigh-Taylor Instability (With A. Lawrie)
- Sediment Transport (U. Liverpool)
- Post-processing of Regional Ocean Model simulations (ROMS) (With R. Watteaux)
- LCS overhead is roughly 30-50% of the simulation (depending on parameters, solver, etc).

Figure : 128x128x256 Jet simulation with MOBILE

Many new flows can be studied with libcfd2lcs!

Straightforward integrations into a variety of flow solvers

Successful integrations:

- Pseudo-spectral turbulence simulations (With R. Watteaux)
- Rayleigh-Taylor Instability (With A. Lawrie)
- Sediment Transport (U. Liverpool)
- Post-processing of Regional Ocean Model simulations (ROMS) (With R. Watteaux)
- LCS overhead is roughly 30-50% of the simulation (depending on parameters, solver, etc).

Figure : Tyrrhenian sea ROMS sea surf. dataset

Summary

libcfd2lcs

- A new platform for computing Lagrangian coherent structures
- Can be integrated into your own MPI CFD solver or post processing utility
- Simple, easy to use interface can be called from C/C++/F90
- Lots of new applications to explore with HPC and new CFD codes.

Future plans

- Inertial particles
- New Diagnostics, keeping up with rapidly developing LCS theory.

More Information `http://pcwww.liv.ac.uk/~finnj/code.html`

Acknowledgments

- This work was funded under the embedded CSE programme of the ARCHER UK National Supercomputing Service (`http://www.archer.ac.uk`).

- Collaboration with Romain Watteaux (Stazione Zoologica Anton Dohrn, Naples) and Andrew Lawrie (University of Bristol).

Bibliography I

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

[Christov et al., 2011] Christov, I., Ottino, J., and Lueptow, R. (2011).
From streamline jumping to strange eigenmodes: Bridging the Lagrangian and Eulerian pictures of the kinematics of mixing in granular flows.
Physics of Fluids, 23:103302.

[Conti et al., 2012] Conti, C., Rossinelli, D., and Koumoutsakos, P. (2012).
GPU and APU computations of finite time Lyapunov exponent fields.
Journal of Computational Physics, 231(5):2229–2244.

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

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

[Haller, 2015] Haller, G. (2015).
Lagrangian coherent structures.
Annual Review of Fluid Mechanics, 47:137–162.

[Lekien and Ross, 2010] Lekien, F. and Ross, S. (2010).
The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds.
Chaos, 20(1):017505–1.

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

[Lipinski and Mohseni, 2010] Lipinski, D. and Mohseni, K. (2010).
A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures.
Chaos, 20(1):017504–1.

[Miron et al., 2012] Miron, P., Vétel, J., Garon, A., Delfour, M., and Hassan, M. (2012).
Anisotropic mesh adaptation on Lagrangian coherent structures.
Journal of Computational Physics.

Bibliography II

[Olascoaga and Haller, 2012] Olascoaga, M. J. and Haller, G. (2012).
Forecasting sudden changes in environmental contamination patterns.
Proc. National Acad. Sci., 109:4738–4743.

[Peacock and Dabiri, 2010] Peacock, T. and Dabiri, J. (2010).
Introduction to focus issue: Lagrangian coherent structures.
Chaos, 20(1):017501.

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

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

[Shadden et al., 2005] Shadden, S., Lekien, F., and Marsden, J. (2005).
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.

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