

lcs4Foam – An OpenFOAM Function Object to Compute Lagrangian Coherent Structures

Constantin Habes^{1,*}, Alexandra von Kameke^{2,*}, Mohammed Elwardi Fadeli¹, and Holger Marschall¹

¹Mathematical Modeling and Analysis, Technical University of Darmstadt, 64287 Darmstadt, Germany
Email address: constantin.habes@tu-darmstadt.de, elwardi.fadeli@tu-darmstadt.de,
holger.marschall@tu-darmstadt.de

²Department of Mechanical Engineering and Production Management, Hamburg University of Applied Sciences, 20099
Hamburg, Germany
Email address: alexandra.vonkameke@haw-hamburg.de

DOI: <https://doi.org/10.51560/ofj.v4.141>

Results with version(s): foam-extend 4.1, OpenFOAM® v2212

Repository: <https://github.com/CoHabes/lcs4foam>

Abstract. To facilitate the understanding and to quantitatively assess the material transport in fluids, a modern characterisation method has emerged in fluid dynamics in the last decades that is based on dynamical systems theory. It allows to examine the most influential material lines which are called Lagrangian Coherent Structures (LCS) and order the material transport into dynamically distinct regions at large scales which resist diffusion or mixing. LCS reveal the robust skeleton of material surfaces and are essential to assess material transport in time-dependent flows quantitatively. Candidates of LCS can be estimated and visualised from finite-time stretching and folding fields by calculating the Finite-Time Lyapunov Exponents (FTLE).

In this contribution, we provide a function object to compute FTLE during CFD simulation by coupling OpenFOAM to the third-party library *libcfd2lcs*. This enables the OpenFOAM community to assess the geometry of the material transport in any flow quantitatively on-the-fly using virtually any OpenFOAM flow solver.

1. Introduction

Material transport and mixing in fluids are primarily driven by advection, which is typically described using an Eulerian perspective, where the time-dependent velocity field $\mathbf{u}(\mathbf{x}, t)$ governs fluid motion. The Eulerian approach has been extensively used to derive key fluid mechanical properties, such as the relationship between high Reynolds numbers (indicative of high velocities) and improved mixing efficiency. However, this intuition can sometimes be misleading, as demonstrated by the study in [1] of a rising bubble. In that case, a coherent structure formed at intermediate Reynolds numbers, causing material to cluster and locally hinder mixing, leading to increased residence times near the rear of the bubble. This example highlights the importance of examining coherent structures to fully understand material transport in specific flow conditions. Lagrangian Coherent Structures (LCS) often manifest in fluid flows, as evidenced by the patterns formed by passive tracers such as plankton in ocean currents [2] or dissolved oxygen behind a rising bubble.

The limitations of the classical Eulerian framework in addressing these issues were first recognized in oceanography and atmospheric science [3, 4]. This led to the adoption of a Lagrangian perspective, in which the movement of individual fluid parcels is tracked, allowing for more accurate insights into material transport. Unlike the Eulerian view, where the observer analyzes fluid motion at fixed locations, the Lagrangian view focuses on the trajectories of the fluid parcels themselves. Advances in computational techniques and dynamical systems theory now enable the analysis of time-dependent flows and the identification of structures that govern transport.

* Corresponding author

Received: 27 May 2024, Accepted: 22 October 2024, Published: 18 November 2024

In time-independent incompressible flows, the velocity field serves as the dynamical system itself, with streamlines representing the trajectories of fluid parcels. In such flows, stable and unstable manifolds divide the flow into coherent subdomains [5], which are structures that dictate material transport. However, in time-dependent flows, the instantaneous streamlines no longer coincide with the trajectories of fluid parcels. This makes it impractical to rely on streamlines or other steady-state flow representations to infer material transport, as the resulting structures may not reflect the actual behavior of the fluid.

Lagrangian Coherent Structures provide a more accurate picture of transport barriers in time-dependent flows. These structures are derived from the trajectories of particles within the time-dependent velocity field $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$, and they delineate regions of fluid with different fates or histories. Various methods have been developed in recent years to calculate and evaluate LCS [6]. LCS have proven useful in a variety of applications, including the study of mesoscale ocean eddies that affect nutrient transport and pollutant dispersion [2, 7, 8], the analysis of atmospheric transport processes that influence weather patterns [9–11], and the examination of vortex dynamics in turbulent flows [12], such as those encountered in aerodynamics and industrial processes.

In this contribution, we present an OpenFOAM function object that computes three-dimensional Finite Time Lyapunov Exponents (FTLE) on-the-fly using the general-purpose numerical library *libcf2lcs*, developed by Justin Finn et al. [13] who explain the main computational details in [14]. Ridges in the FTLE field are considered candidates for LCS and can be assumed to coincide with LCS when certain conditions are met [15]. However, as noted by [13], these conditions are difficult to evaluate in three-dimensional flows, so the FTLE field is treated as an approximate representation of the 3D LCS. Details on the FTLE calculation and its mathematical foundation are provided in Section 2.

2. Theoretical background of LCS calculations

From time-resolved CFD simulations, the time-dependent velocity field $\mathbf{u}(\mathbf{x}, t)$ is known in space and time. From this information the fluid parcel or passive particle trajectories

$$\mathbf{x}(\mathbf{x}_0, t) = \mathbf{x}_0 + \int_{t_0}^t \mathbf{u}(\mathbf{x}(\tau), \tau) d\tau \quad (1)$$

can be calculated, where \mathbf{x}_0 is the starting point of a trajectory in 3D space at a starting time t_0 . Note, that each trajectory is now labelled by its start location in space and time. If a set of initially close passive particles is released at the same time the distances between them change over time due to the fluid motion. Passive particles initially forming a tiny sphere will undergo a linear deformation towards an ellipse for short times as would occur in a solid body under stress before it breaks. Certainly, in a fluid, the deformation will progress, and non-linear higher-order terms will play a role in causing stretching and folding, which is crucial for mixing. However, as a first approximation and for short times these higher-order terms are neglected for the analysis of the deformation. If we consider infinitesimal spheres of initially close particles around all mesh cell centres of our simulation starting at the same initial time t_0 , we obtain a set of different ellipsoids. All these ellipsoids have differently stretched and contracted principal axes which point in different directions at a slightly later time t_1 . The principal axes of each ellipse denominate the final directions of maximal stretching (major axis) and maximal contraction (minor axis) of the initially spherical particle blob. The stretching factor S is the length of the major axis of the final ellipse divided by the initial radius of the sphere. If this stretching factor at each initial grid point is plotted, a 3D stretching field results revealing the regions at which stretching and thus particle separation for the time interval of interest $[t_0, t_1]$ is largest due to the local flow conditions. Typically, the evolution of the stretching factor over time is described by

$$S = e^{\sigma_{t_0}^{t_1} |t_1 - t_0|}, \quad (2)$$

with $\sigma_{t_0}^{t_1}$ being the Finite-Time Lyapunov Exponent [16]. Connected areas or lines of large FTLE values characterise the fluid transport as these denote the areas or lines along which deformation and thus particle separation is largest. All these geometrical considerations have their mathematical counterparts. The stretching factor as described is the square root of the maximal eigenvalue of the right Cauchy-Green deformation tensor $\mathbf{C}_{t_0}^{t_1}$. This tensor can be calculated for every mesh cell as envisioned above for the ellipsoid. As its name reveals it includes all the information about the deformation of the fluid masses at this point for the short time interval $t_1 - t_0$, and notably it is an objective tensor such that high stretching values and candidates for LCS derived from it will persist regardless of the motion of the observer (invariant to a time-dependent translation and rotation of the coordinate system of the observer) [16, 17].

The governing ordinary differential equation (ODE) for the evolution of a fluid parcel or a passive particle reads

$$\dot{\mathbf{x}} = \mathbf{u}(\mathbf{x}(t), t). \quad (3)$$

Therefore, the infinitesimal separation $\gamma = \mathbf{x} - \mathbf{x}^*$ of the passive particle, imagined in the centre of a infinitesimal sphere, to a particle on the surface of the sphere will be governed by the ODE

$$\delta\dot{\mathbf{x}} = \nabla\mathbf{u}\gamma. \quad (4)$$

The solution of this ODE is an exponential function, which explains why the FTLE is defined as the logarithm of the stretching factor.

To analyse the stretching during short but finite time intervals, particles distributed on a mesh are advected with the flow from an initial time t_0 over the time interval $T = |t_1 - t_0|$ to t_1 . From the integral version of the governing ODE (Eqn. 3) we obtain the definition of the flow map, $\Phi_{t_0}^{t_1}$, which maps all the particles from their initial positions onto their final positions at time t_1 , viz.

$$\Phi_{t_0}^{t_1} : \mathbb{R}^n \rightarrow \mathbb{R}^n; \quad \mathbf{x}_0 \mapsto \mathbf{x}_0 + \int_{t_0}^{t_1} \mathbf{u}(\mathbf{x}(\tau), \tau) d\tau. \quad (5)$$

To obtain the separation of two initially close particles after this time interval a Taylor series

$$\delta\mathbf{x}(t_1) = \Phi_{t_0}^{t_1}(\mathbf{x}_0) - \Phi_{t_0}^{t_1}(\mathbf{x}_0 + \delta\mathbf{x}(t_0)) = \mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)\delta\mathbf{x}(t_0) + \mathcal{O}(|\delta\mathbf{x}(t_0)|^2) \quad (6)$$

around the initial position can be employed. Where $\mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)$ is the gradient (Jacobian) of the flow map with regard to the initial separation and is also the normalised fundamental matrix solution of the equation of variations above (Eqn. 4) [17]. Therefore, the particle separation at time t_1 can be written as

$$\|\delta\mathbf{x}(t_1)\| = \sqrt{\left\langle \delta\mathbf{x}(t_0), [\mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)]^* [\mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)] \delta\mathbf{x}(t_0) \right\rangle}. \quad (7)$$

The right Cauchy-Green deformation tensor is then defined as

$$\mathbf{C}_{t_0}^{t_1}(\mathbf{x}_0, t_0) = [\mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)]^* [\mathbf{D}\Phi_{t_0}^{t_1}(\mathbf{x}_0, t_0)]. \quad (8)$$

In this way the Finite-Time Lyapunov Exponent $\sigma_{t_0}^{t_1}$ for the time interval t_0 to t_1 can now be defined on the basis of this tensor in a more thorough, mathematical way. Therefore, it is now defined by

$$\sigma_{t_0}^{t_1}(\mathbf{x}_0, t_0) = \frac{1}{|t_1 - t_0|} \log \sqrt{\lambda_{\max}(\mathbf{C}_{t_0}^{t_1}(\mathbf{x}_0, t_0))}. \quad (9)$$

Here λ_{\max} is the maximum eigenvalue of the right Cauchy-Green deformation tensor and can be calculated using standard solvers. In the picture of the small ellipsoid, the square root of the eigenvalue is just the stretching rate S in Eqn. (2).

3. Computational details

The computation of flow maps within *libcfd2lcs* is described thoroughly in [14]. The following section presents a brief overview of how the computation is done in practice and which different timescales play a role in the calculations. Hereafter, we describe the structure and functionality of the newly developed function object. We will focus on how the function object acts as an interface between OpenFOAM and *libcfd2lcs*, how parallelisation is ensured and what has to be considered for the output of the generated data.

3.1. Numerical flow map computation in libcfd2lcs. *libcfd2lcs* is able to calculate both forward-time and backward-time FTLE fields. However, it uses two very different approaches for calculating the respective flow maps. The general approach used for the computation of the forward time flow-map $\Phi_{t_0}^{t_0+T}$ and the resulting forward-time FTLE field is very straightforward. A set of tracer particles is initialised on a grid with spacing Δx_{lcs} by setting each initial tracer coordinate to the cell centre coordinate of a corresponding mesh cell. Then the flow map at each cell centre is computed by passively advecting these tracers with the flow, which mathematically corresponds to an integration of equation

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}, t) \quad (10)$$

over the time interval T . Numerically this integration is done by utilising Runge-Kutta methods, with step size Δt_{lcs} . The time and space dependent velocity field $\mathbf{u}(\mathbf{x}, t)$ results from the specific fluid simulation under consideration and is passed to *libcfd2lcs* after each simulation time step Δt_{sim} (see Section 3.2). In order to save the flow map $\Phi_{t_0}^{t_0+T}$, the location of each particle after the integration is stored at its initial position.

As the evaluation of FTLE fields, indicating LCS candidates, is mainly relevant for time-dependent flows, it is often important to animate their evolution. At first glance, this would mean that a sequence of large particle sets would have to be integrated, requiring a great amount of computation. This problem is solved using a method developed by Brunton and Rowley [18]. With this method a flow map of the interval T can be constructed from a sequence of k flow maps over a smaller interval h , where $T = kh$. Following the notation of [18] this can be expressed as

$$\Phi_{t_0}^{t_0+T} = \Phi_{t_0+(k-1)h}^{t_0+kh} \circ \dots \circ \Phi_{t_0+h}^{t_0+2h} \circ \Phi_{t_0}^{t_0+h} . \quad (11)$$

In practical terms, this means that the particle grid is reinitialised for every new time interval h after which they are advected again with the flow. Then the sub-step flow map is stored and the complete flow map is constructed when all needed sub-step flow maps are available. It is important to note that since a discrete particle grid is used for the sub-step flow map computation, interpolation of the sub-step flow maps is needed in order to match the trajectories at different timelevels when reconstructing the flow map $\Phi_{t_0}^{t_0+T}$ (see [18] for more details).

A different approach for constructing the backward-time flow maps is used. This is due to the fact that using the Lagrangian approach would require to store all computed velocity fields in the subset interval h before the integration of the tracers from $t_0 + h$ to t_0 could be done backward in time. Although this already includes Brunton’s and Rowley’s method for the flow map construction, the Lagrangian approach would be “cumbersome and resource intensive” [14, p. 4]. Therefore, *libcfd2lcs* uses an Eulerian approach for the flow map computation proposed by Leung [19]. In contrast to the forward-time flow map, the backward-time flow map $\Phi_{t_0+T}^{t_0}$ describes for each grid point where a particle, that is at that point at time $t_0 + T$, originally was at time t_0 . With Leung’s Eulerian approach this backward-time flow map at time $t_0 + T$ is computed by initialising a vector field $\Psi(\mathbf{x}, t_0)$ on a grid with the cell centre coordinates at time t_0 . The advection of this so called “takeoff coordinate field” in an Eulerian reference frame is then described by the level set equation

$$\frac{\partial \Psi(\mathbf{x}, t)}{\partial t} + (\mathbf{u} \cdot \nabla) \Psi(\mathbf{x}, t) = \mathbf{0} . \quad (12)$$

Solving this equation over the time Interval $[t_0, t_0+T]$ in forward time gives $\Psi(\mathbf{x}, t_0+T)$, which represents the takeoff coordinates of a Lagrangian particle at t_0 reaching \mathbf{x} at time t_0+T . Thus, the backward-time flow map $\Phi_{t_0+T}^{t_0}$ is equivalent to $\Psi(\mathbf{x}, t_0+T)$. *libcfd2lcs* solves Eqn. (12) by using a semi-Lagrangian advection approach with the time step size

$$\Delta t_{lcs} = \frac{c_{cfl} \Delta x_{lcs}}{\mathbf{u}(\mathbf{x}, t)} \quad (13)$$

of this procedure being restricted by the CFL condition $c_{cfl} < 1$ (see [14] and [19] for more details). Furthermore, Brunton’s and Rowley’s flow map construction method is also applied to the backward-time flow maps computed with the Eulerian method. Hence, the takeoff coordinate field is reinitialised after every sub-step time interval h and the backward-time flow map

$$\Phi_{t_0+T}^{t_0} = \Phi_{t_0+h}^{t_0} \circ \Phi_{t_0+2h}^{t_0+h} \circ \dots \circ \Phi_{t_0+kh}^{t_0+(k-1)h} \quad (14)$$

is constructed from k sub-step backward-time flow maps.

Since a lot of different timescales are relevant in the practical FTLE field computation described above, we try to differentiate and order them in the following, before describing the structure and functionality of the newly developed function object in the next section. The basis of the on-the-fly LCS evaluation is a parallel running simulation that provides the velocity fields. Here, three intervals are of interest (see Fig. 1): the overall simulation time that spans from the simulation start time t_{sim_start} to the simulation end time t_{sim_end} , the time step size of the simulation Δt_{sim} and the write time interval of the simulation results Δt_{sim_write} . The computed velocity fields represent a fluid flow for which a reference timescale Δt_{ref} can be identified. This reference timescale characterises the dominant hydrodynamic timescale of the flow and is typically larger than the simulation time step size. In order to save computing resources the LCS evaluation of the simulated flow does not necessarily have to start and end at the same time as the simulation. Therefore, a separate start and end time for the LCS evaluation denoted as t_{lcs_start} and t_{lcs_end} can be defined (see Fig. 1). During the LCS evaluation, a series of FTLE fields are computed. These FTLE fields are calculated from time T flow maps, which themselves are calculated as described earlier in this section. This means storing and constructing the time T flow maps from multiple sub-step flow maps after each LCS sub-step integration interval h . Calculating the sub-step flow maps in turn requires to numerically solve Eqns. (10) or (12) using the finite time step Δt_{lcs} . While Δt_{lcs} is set automatically according to Eqn. (13) and a specified CFL number, T and h have to be defined by the user. In order to detect all LCS candidates, T is usually chosen to be larger than Δt_{ref} of the

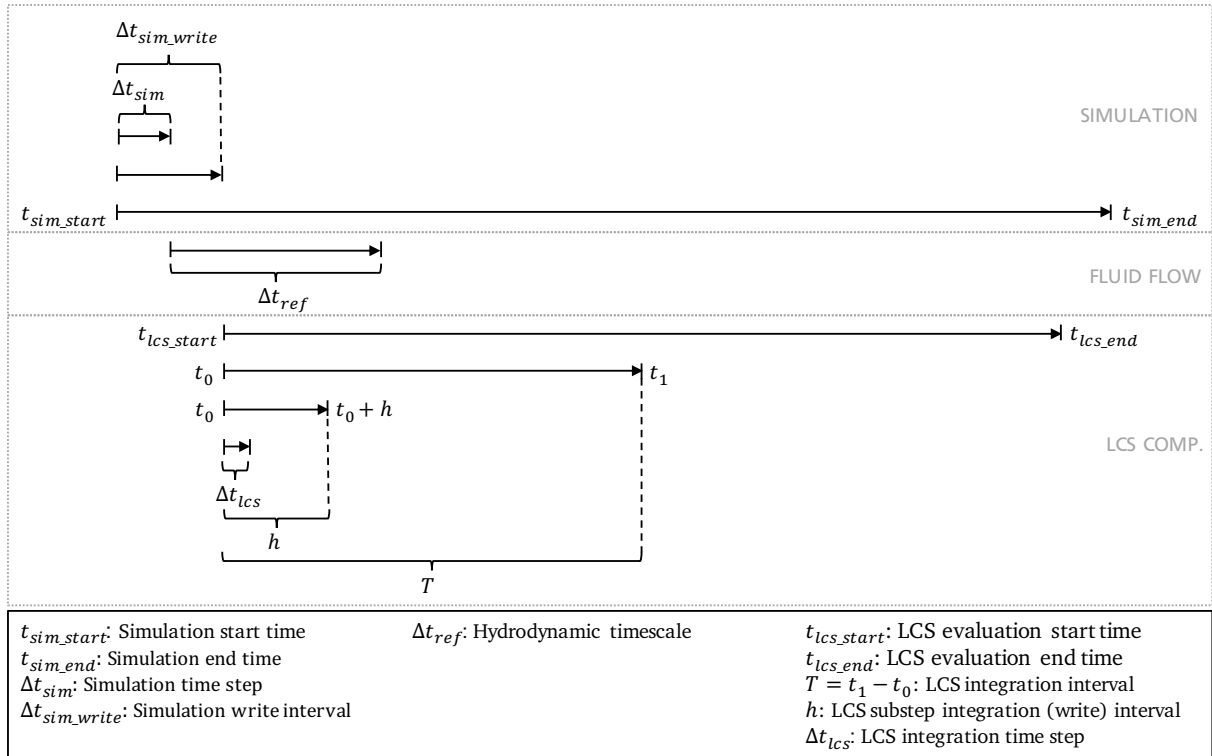


Figure 1. Different timescales relevant for the LCS evaluation

investigated flow [14]. With the aim of animating the evolution of the FTLE field, h is typically set significantly smaller than Δt_{ref} while being in the order of magnitude of Δt_{sim_write} .

3.2. Structure and functionality of the function object. In general, function objects can be used to generate additional data at runtime of the simulation. In doing so, function objects can access data generated by the flow solver at runtime, which offers a great advantage over classical post-processing since it can only utilise the stored fields or logged information. The newly developed function object incorporates the functionalities of *libcfd2lcs* into OpenFOAM at runtime while acting as an interface between both. This is achieved by processing the data generated by OpenFOAM and the subsequent exchange of this data via the *libcfd2lcs* API (see [13] for a detailed description of the *libcfd2lcs* API).

The calculation of the flow maps, the calculation of the resulting FTLE fields and the subsequent saving of these fields is completely handled by *libcfd2lcs*. The basic task of the function object is to pass the cell centre position vectors of the computational grid as well as the velocity field calculated by OpenFOAM to *libcfd2lcs*. Due to the very strict data structure requirements of *libcfd2lcs* this is not a trivial task. *libcfd2lcs* can only use static rectilinear grids for the calculation of forward-time and backward-time flow maps and therefore needs the velocity fields on these grids. This means that the mesh and velocity data has to be globally organised in an (i, j, k) structured format [13, p. 6]. Since the LCS evaluation should also be available for simulations on moving grids with general topology and adaptive grid refinement, the function object offers several possibilities to deal with this problem.

In the simplest case, where the simulation mesh is already a static rectilinear mesh, the function object does not need to process the grid and velocity data, but can directly transfer it to *libcfd2lcs* as basic C++ arrays. This is the preferred method when the flow domain can be represented by a static rectilinear mesh and e.g. immersed boundary methods are used. If a moving mesh, a mesh of general topology or adaptive mesh refinement is used for the simulation a different approach is needed in order to prepare the data for its use in *libcfd2lcs*. Here, an additional static rectilinear mesh needs to be constructed in the preprocessing step, which can be done e.g. by using the `blockMesh` utility. This mesh has to contain the region for which the LCS diagnostic should be performed, meaning that it can cover the whole simulation domain as well as only a part of it. However, since *libcfd2lcs* also requires boundary conditions for the FTLE field calculations, the boundary patches of the additional LCS mesh must be set accordingly. The user can choose between `empty`, `symmetryPlane`, `wedge`, `cyclic` and the generic `patch` patch types which the function objects translates into the corresponding *libcfd2lcs* boundary types. Then, during runtime, the velocity fields are mapped from the simulation mesh of general topology to the static rectilinear LCS

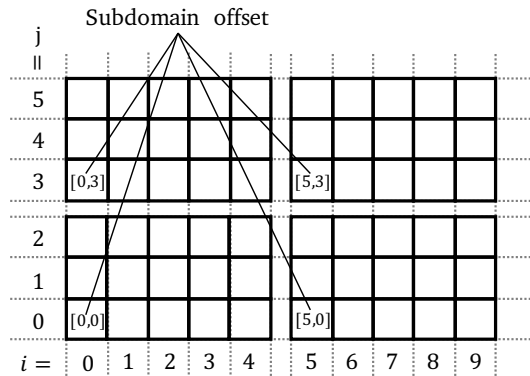


Figure 2. Subdomain offset of a i, j , ($k = 0$) structured rectilinear mesh

mesh, from which the data can again be transferred to *libcfd2lcs* as basic C++ arrays. Although this implies that interpolation errors are made during the mapping process, the LCS evaluation is hardly affected by this. Haller showed in [20] that LCS are very robust against errors in the velocity field. Also, the additional computational overhead due to the mapping can be neglected compared to the overhead caused by the flow map computations. The function object also implements a third approach in which no additional LCS mesh is needed. This approach utilises the ability to construct complex, moving mesh geometries out of simple unconnected mesh regions in OpenFOAM with the `oversetMesh` approach. Using this approach the function object can utilise any specified static rectilinear mesh region of the `oversetMesh` for the LCS evaluation, meaning that the background mesh as well as any other static rectilinear mesh region can be used. In doing so, the function object extracts the mesh and velocity data from the specified mesh region of the `oversetMesh` and passes it to *libcfd2lcs* analogously to the previous approaches. Here the `overset` type patches are generally passed on as inlet or outlet, as they are treated the same by *libcfd2lcs*.

As *libcfd2lcs* also uses the domain decomposition approach and MPI for the parallelisation of the computations, the integration within the parallelisation of OpenFOAM is done in a straightforward manner. The local subdomains of the rectilinear LCS mesh and its velocity data are passed to *libcfd2lcs* together with an offset, which describes the position of the cell data in the globally (i, j, k) structured data array (see Fig. 2). For the MPI communication, the same MPI communicator as used for OpenFOAM is shared with *libcfd2lcs*. Therefore, the function object can be used for simulations running in parallel or serial. However, if the approach involving an additional LCS mesh is used, special attention is required for the domain decomposition in the preprocessing step. Here the simulation mesh, as well as the LCS mesh, must be cut along the same surfaces to make sure that the mapping of the velocity fields from one mesh to the other works properly.

As already mentioned, the data output of the flow-map and FTLE field data is completely handled by *libcfd2lcs*. This is due to the fact that the data output interval defined by h can differ from the solver write interval Δt_{sim_write} (see Section 3.1). Therefore, the results generated by the function object are not stored in corresponding time directories but in a separate folder in the case directory called `cf2lcs_output`. Additionally, a directory named `cf2lcs_temp` is created inside of which all the sub-step data is stored. All data is stored in the Tecplot ASCII data file format (*.dat) and therefore can be visualised in ParaView when opened with its internal Tecplot reader or other common visualisation programs. In addition to this data, the computational overhead generated by the use of the function object with respect to the actual simulation is also output in the solver log file after each simulation time step. This enables the user to examine the computational costs of the LCS evaluation.

4. Examples of usage

In this section a few examples are presented which are designed to show the functionality and capabilities of the function object. Therefore, example cases are presented in which only a rectilinear simulation mesh, a separate simulation and LCS mesh and a single `oversetMesh` are used.

4.1. Steady ABC flow. The Arnold-Beltrami-Childress (ABC) flow is an exact periodic solution of the Euler equations and is often used in the literature to verify LCS calculation methods. Therefore this case

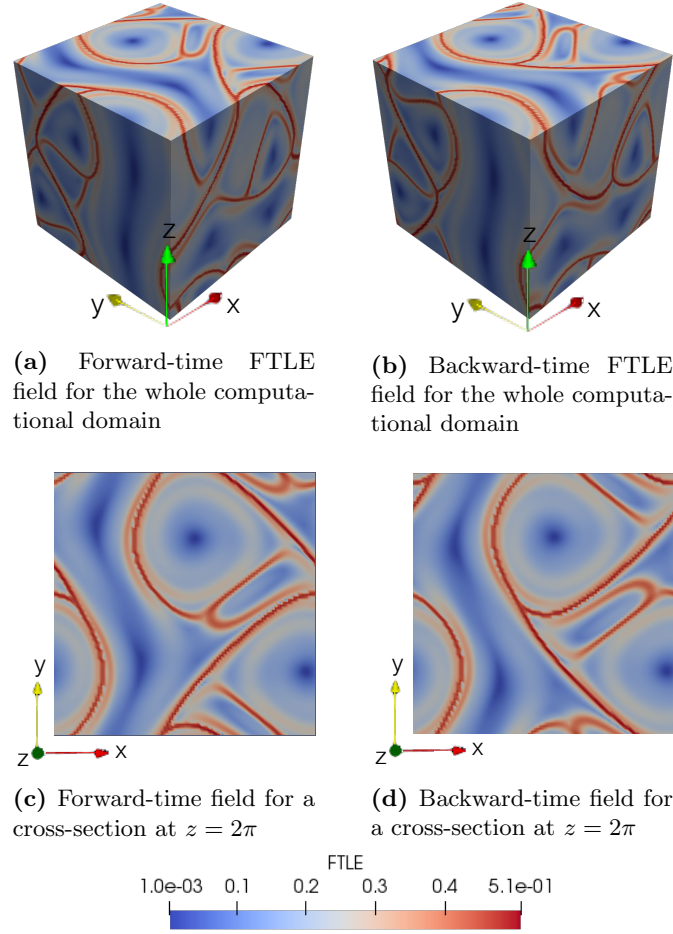


Figure 3. FTLE fields of a stationary ABC flow

is also being reviewed here. The velocity field

$$\mathbf{u} = \nabla \times [-\Psi \mathbf{k} + \nabla \times (\Phi \mathbf{k})] \quad (15)$$

of the ABC flow can be described using 2 scalar potentials Ψ and Φ [21] which themselves are defined as

$$\begin{aligned} \Psi &= -[C \sin(y) + B \cos(x)] , \\ \Phi &= A[-x \cos(z) + y \sin(z)] - \Psi . \end{aligned} \quad (16)$$

In Eqn. (15), \mathbf{k} can be any unit vector but is commonly chosen to be the vertical unit vector. This leads to the three expressions of the velocity components

$$\begin{aligned} u &= A \sin(z) + C \cos(y) , \\ v &= B \sin(x) + A \cos(z) , \\ w &= C \sin(y) + B \cos(x) . \end{aligned} \quad (17)$$

The parameters A , B and C can be freely selected and influence the properties of the ABC flow. In order to create comparability with literature values, $A = 0.5$, $B = 0.8$, $C = 0.8$ is chosen. In order to test the newly developed function object on this flow configuration a dedicated ABC flow OpenFOAM solver was written. This solver does not solve the Euler equations in the usual sense, but sets the velocity components on a given computational mesh according to Eqn. (17). Due to the periodicity of the flow solution, the dimensions of the computational mesh used in this case setup are specified as $x, y, z \in [0, 2\pi]$ with a mesh size of $100 \times 100 \times 100$. Since the described mesh is rectilinear no additional LCS mesh is used. Again for reasons of comparability, a LCS integration time of $T = 10s$ is selected for the LCS evaluation. The results of the LCS evaluation, both in forward- and backward-time, can be seen in Fig. 3.

In these results the FTLE ridges, which indicate the LCS candidates in the ABC flow, can be seen very clearly. Furthermore, the results agree very well with the results from [21], both qualitatively and

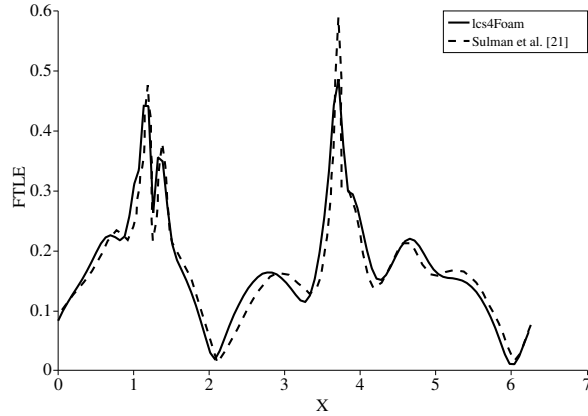


Figure 4. Cross-sections of the forward-time FTLE values in the steady ABC flow along the x-direction at $z = 2\pi$, $y = \pi/2$ computed by Sulman et al. [21] and the lcs4Foam function object

quantitatively as can be seen in Fig. 4, which suggests that the new function object calculates the FTLE ridges reliably.

4.2. Time dependent double gyre. Another frequently used flow for the verification of LCS computing algorithms is the time periodic Rayleigh-Bénard convection flow, or often called double gyre, proposed by Solomon and Gollub [22]. The velocity field of this flow can be described by using a stream function ψ

$$\begin{aligned} u &= -\frac{\partial\psi}{\partial y}, \\ v &= \frac{\partial\psi}{\partial x}. \end{aligned} \quad (18)$$

Here ψ is defined by

$$\psi(x, y, t) = A \sin(\pi f(x, t)) \sin(\pi y) \quad (19)$$

with

$$\begin{aligned} f(x, t) &= a(t)x^2 + b(t)x, \\ a(t) &= \epsilon \sin(\omega t), \\ b(t) &= 1 - 2\epsilon \sin(\omega t). \end{aligned} \quad (20)$$

This leads to the expressions for two-dimensional velocity components

$$\begin{aligned} u &= -\pi A \sin(\pi f(x)) \cos(\pi y), \\ v &= \pi A \cos(\pi f(x)) \sin(\pi y) \frac{df}{dx}. \end{aligned} \quad (21)$$

As the name double gyre suggests, this model defines the flow of two two-dimensional gyres enclosed in a rectangle which expand and contract periodically along the x-axis. Therefore, the periodic motion is controlled by ϵ if $\epsilon \neq 0$. Then ϵ describes approximately how far the line separating the gyres moves to the left or right from its centre position [16]. Otherwise ($\epsilon = 0$), no periodic motion is happening. Furthermore, A specifies the magnitude of the velocity vectors and $\omega/2\pi$ determines the oscillation frequency of the gyres.

Similar to the ABC flow example, a dedicated OpenFOAM solver was written for this case, which sets the velocity field on a given computational mesh according to Eqn. (21). For comparability, a mesh with the same specifications as in [23], [19] and [16] was used. It has the dimensions $[0, 2] \times [0, 1] \times [0, 0.1]m$ and a resolution of $512 \times 256 \times 1$ cells. As this mesh is also static and rectilinear no additional LCS mesh was used. For the mathematical model of the flow the parameter values are chosen to be $\epsilon = 0.1$, $A = 0.1m s^{-1}$ and $\omega = 2\pi/10s$. Since the oscillation frequency is known, the hydrodynamic time scale can be easily determined by $t_{ref} = 2\pi/\omega = 10s$. As described in Section 3.1, the LCS integration time interval T should be set larger than t_{ref} . Therefore, it is set to $T = 1.5 \cdot t_{ref} = 15s$. Figure 5 shows the forward- and backward-time FTLE fields at $t = 15s$ of the previously described double gyre flow. Again,

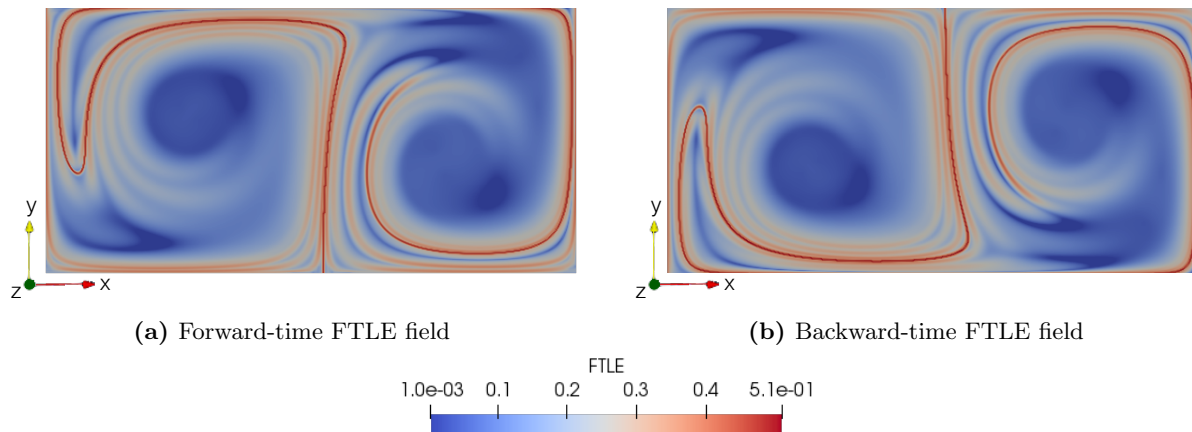


Figure 5. FTLE fields of a time-dependent double gyre at $t = 15s$.

the results match very well with the results from [23], [19] and [16]. This confirms that the function object is able to calculate the correct FTLE fields from velocity fields generated by OpenFOAM.

4.3. Flow around cylinder. As it has already been shown in the previous examples that the function object can calculate the correct FTLE fields from velocity fields provided by OpenFOAM, this example will focus on how to deal with non-rectilinear simulation meshes. For this purpose, a standard flow problem is selected that is very well suited for an LCS evaluation: the flow around an infinitely long cylinder. The general case setup contains a fluid domain with size $[-20, 30] \times [-20, 20] \times [-0.5, 0.5]m$ that surrounds a cylinder with diameter $D = 2m$ and its centre axis at $x = y = 0m$. The free-stream velocity and the fluids kinematic viscosity are set to $\mathbf{u}^T = (1 \ 0 \ 0)m s^{-1}$ and $\nu = 0.01m^2s^{-1}$, respectively. This results in a Reynolds number of $Re = 200$ which indicates that vortex shedding behind the cylinder occurs in a barely laminar regime. If we also assume a Strouhal number of $St = 0.2$ at $Re = 200$, the hydrodynamic time scale of this flow is $t_{ref} = D/(u \cdot St) = 10s$.

Because of the cylinder in the middle of the domain, a computational mesh discretising this domain is no longer rectilinear. Therefore, we consider two different procedures in the LCS evaluation, the first of which is carried out in two different ways.

4.3.1. Additional LCS mesh enclosing the whole domain. For the first approach an additional rectilinear computational mesh is used for the LCS evaluation and the flow domain is discretised with a simulation mesh consisting of 9200 hexahedra (see upper left mesh in Fig. 6). The flow solver that is used to simulate the previously described flow from $t = 0s$ to $t = 120s$ is `pimpleFoam` with the initial conditions being calculated by `potentialFoam`. The first additional LCS mesh that is used within this procedure encloses the whole flow domain (see upper right mesh in Fig. 6). In order to minimise the loss of information during the mapping of the velocity fields between the two grids, the resolution of the LCS mesh is chosen in a way that it corresponds approximately to the finest resolution in the simulation mesh. This leads to a LCS mesh with $200 \times 160 \times 1$ hexahedra. The boundary patch types are set to `patch` for the left and right patch (inlet,outlet), to `symmetryPlane` for the bottom and top patch and to `empty` for the front and back patch. The LCS integration time T is again based on t_{ref} and is set to $T = 1.5 \cdot t_{ref} = 15s$. For a good animation of the dynamics of the FTLE fields h is chosen to be $h = T/10 = 1.5s$. The results of the forward- and backward-time FTLE fields can be seen in Fig. 6. They show how the vortices behind the cylinder form large coherent structures, where the FTLE ridges of the backward-time FTLE fields separate different fluid packages that do not mix in the vortex street.

4.3.2. Additional LCS mesh enclosing only a part of the domain. Since the FTLE ridges only appear in a fraction of the overall domain and the LCS evaluation is computation-wise a quite costly operation, a second LCS mesh is prepared. This second LCS mesh is a lot smaller than the first one and encloses only the fraction of the flow domain where the FTLE ridges are expected to show up (see Fig. 7). The boundary patches on the smaller LCS mesh and its spacial resolution are also set analogous to its bigger counterpart, leading to a LCS mesh of size $[-13, 27] \times [-7.5, 7.5] \times [-0.5, 0.5]$ containing $160 \times 60 \times 1$ hexahedra. Repeating the computations with the use of the smaller LCS mesh gives the results which are displayed in Fig. 7 and are found to match with the results from the bigger LCS mesh (see Fig. 9). This shows that the LCS evaluation, when done with a separate LCS mesh, can be used in a very

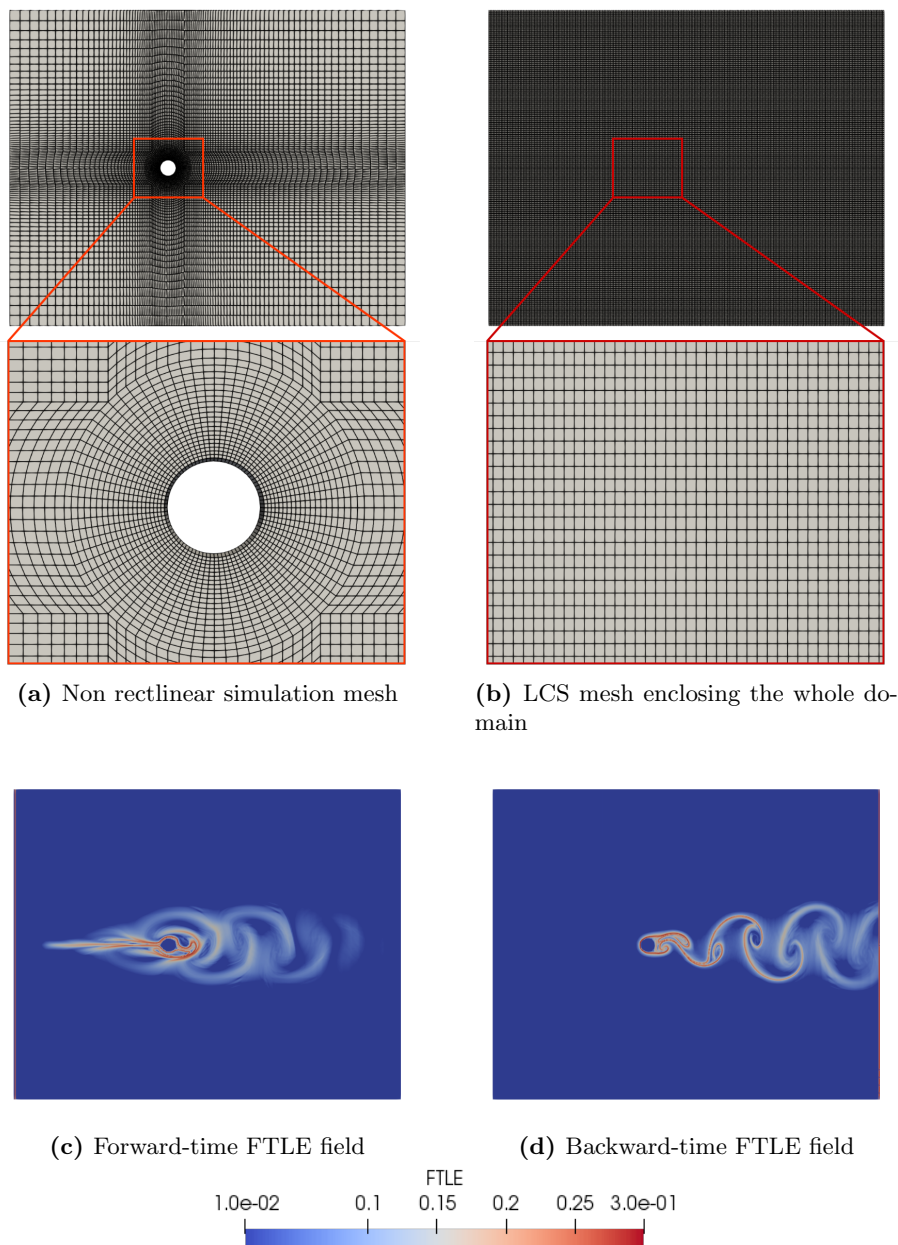


Figure 6. Simulation mesh, LCS mesh spanning the whole domain and FTLE fields of a flow around a cylinder with $Re = 200$ at $t = 118s$.

targeted way. The advantages this brings in terms of computational costs are discussed after considering the second procedure for the LCS evaluation of this flow problem.

4.3.3. Utilizing mesh zones of overset meshes as LCS mesh. Besides using an additional LCS mesh on problems where no single static rectilinear mesh can be constructed, a second procedure exists that utilises OpenFOAM's `oversetMesh` functionalities. With regard to the flow problem considered here, an `oversetMesh` is constructed with the same dimensions as the simulation mesh used previously. It consists of three mesh zones, namely a rectilinear background mesh zone that spans the whole fluid domain, another finer and smaller mesh zone that is used for a finer resolution of the flow and a cylindrical mesh that surrounds the cylinder (see Fig. 8). For comparability reasons the finer rectilinear mesh zone has the same dimensions in the y -direction and is just $0.5m$ shorter in the x -direction as the smaller additional LCS mesh considered previously. It also has the same resolution as the smaller additional LCS mesh and is therefore specified as the cell zone for the LCS evaluation. Also all other LCS evaluation settings are adopted. The only difference to the previously considered simulations is the used flow solver. Here the flow solver is `icoOversetFoam` from `foam-extend 4.1` due to the used `oversetMesh`. The resulting

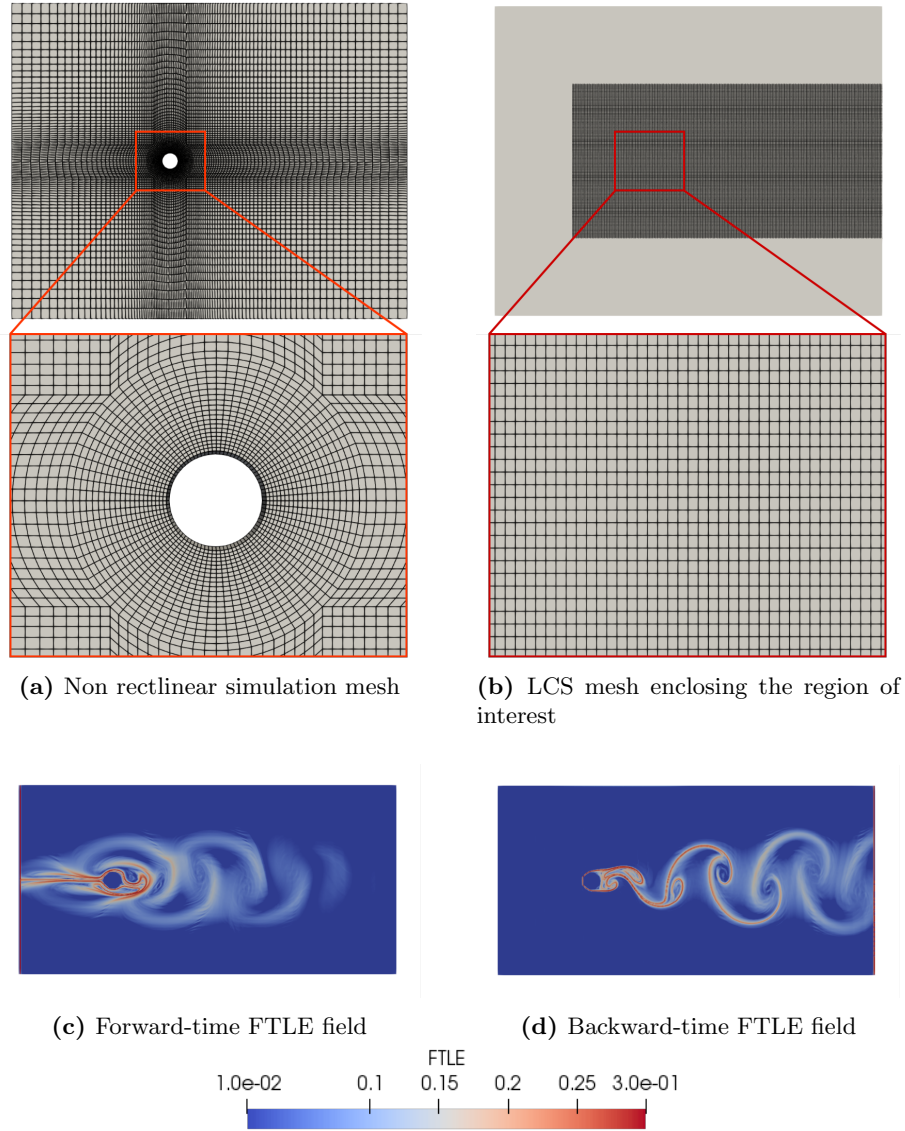
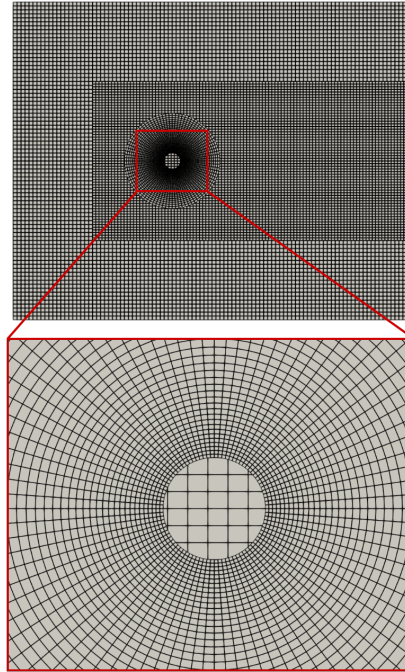


Figure 7. Simulation mesh, small LCS mesh only spanning the region of interest and FTLE fields of a flow around a cylinder with $Re = 200$ at $t = 118s$.

forward- and backward-time FTLE fields of this simulation can be found in Fig. 8. They match with the results from the previously considered procedure (see Fig. 9) which shows that both approaches can be used equally well. The only thing that stands out are the high FTLE values along some boundaries in the studied solutions. These occur because of the way *libcfd2lcs* handles its inlet and outlet boundary conditions. It fixes out-flowing Lagrangian particles/takeoff coordinates on “open” boundaries and cannot generate new in-flowing particles during the flow map computation. Therefore, high FTLE values occur in the forward-time FTLE fields at “open” boundaries where inflow occurs, since there the most “stretching” happens. Vice versa, high FTLE values occur in the backward-time FTLE fields at “open” boundaries where outflow occurs, since there the most “folding” happens. These high values at “open” boundaries are just artefacts and have to be neglected. The reason they appear more in the *oversetMesh* approach is that all *overset* type patches are passed to *libcfd2lcs* as “open” boundaries whereas the user can specify all patches problem dependent in the additional LCS mesh approach.

4.3.4. Computing time comparison. Looking at the computation times of the flow calculations including the LCS evaluation, it becomes evident that LCS evaluation is a very costly operation (see Tab. 1). When using the “large” additional LCS mesh the simulation takes approximately 30 time longer than without the LCS evaluation. This can be improved by using the smaller additional LCS mesh. Here the simulation takes 9 times longer than without the LCS evaluation. Since the costs for the LCS evaluations are almost independent of the underlying simulation for a constant grid size, this factor becomes smaller



(a) `oversetMesh` containing a rectilinear background mesh zone, a rectilinear refinement zone which is also used for the LCS evaluation, and a cylindrical mesh zone containing the cylinder.

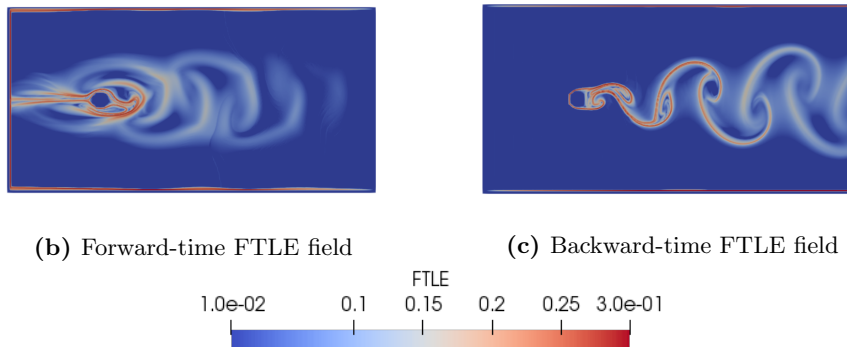


Figure 8. `oversetMesh` and FTLE fields of a flow around a cylinder with $Re = 200$ at $t = 118$ s.

and smaller for more complex simulations. This can also be seen from the fact that the factor is only 2.5 when the `oversetMesh` approach is used because the computations of the pressure and velocity fields take longer on an `oversetMesh`. At this point, however, it must be emphasised that the flow considered here is not a highly complex problem, which can also be seen from the simulation time of 1.5 min on a normal mesh and 8 min on an `oversetMesh`.

5. Summary & Conclusion

We provide an OpenFOAM function object based on `libcfd2lcs` to compute Finite-Time Lyapunov Exponent (FTLE) fields that indicate candidates of Lagrangian Coherent Structures (LCS) and allow to visualise finite-time stretching and folding fields. LCS reveal the robust skeleton of material surfaces and are key to quantitatively assess material transport in time-dependent flows. This enables the OpenFOAM community to assess the geometry of the material transport in any flow quantitatively on-the-fly using principally any OpenFOAM flow solver.

Focusing on the practical aspects, we only give a brief overview of the mathematical foundation as well as how the computation is done in practice. We describe the structure and functionality of the newly developed function object. Further focus is laid on how the function object acts as an interface between

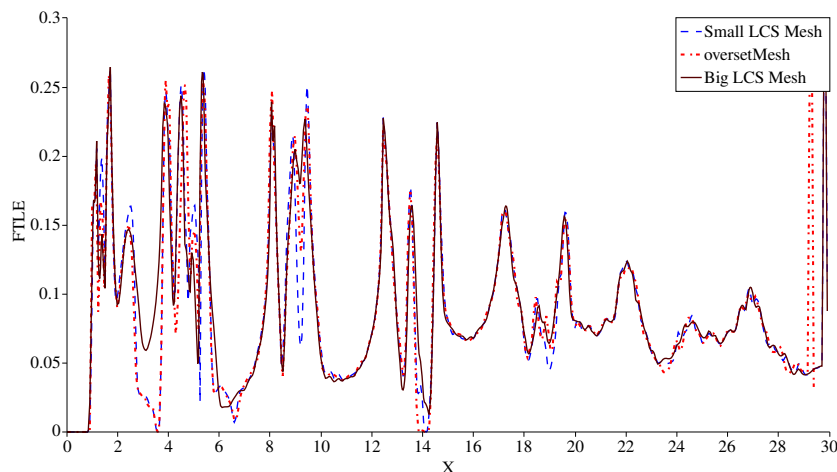


Figure 9. Cross-sections of the backward-time FTLE values of the flow around a cylinder with $Re = 200$ at $t = 118s$ along the x -direction starting from the center of the cylinder. Compared are the values computed with an additional LCS mesh enclosing the whole domain (“Big LCS Mesh”), enclosing only a part of the domain (“Small LCS Mesh”) and computed by utilizing a mesh zones of an `oversetMesh` as LCS mesh.

	“large” add. LCS mesh	“small” add. LCS mesh	<code>oversetMesh</code>
Total time	44	13.5	20
Simulation time	1.5	1.5	8
Total FTLE comp. time	42.5	12	12
Fwd.-FTLE comp. time	8.5	2	2
Bkwd.-FTLE comp. time	21	6	6
I/O, flow map const., ...	13	4	4

Table 1. Computing times in minutes for the flow simulation including LCS evaluation using FOAM-extend 4.1 on 2 cores of a Intel Core i5 processor

OpenFOAM and `libcfd2lcs`, how parallelisation is ensured and what has to be considered for the output of the generated data.

From validation of the presented function object using simple benchmark problems, a notable computational overhead has been recognised. However, if LCS evaluations are used for much more complex problems as the ones used here, the computational overhead significantly drops and the LCS evaluation no longer accounts for the largest proportion of the computation time. Nevertheless, the user should be aware that the calculation of FTLE fields is expensive and should therefore think carefully about the size and position of the LCS mesh. In addition, consideration should also be given to whether both forward and backward-time FTLE calculations are required or if one of them is sufficient.

Author Contributions: Conceptualisation, C.H. and H.M.; methodology, C.H. and A.K.; software, C.H. and M.E.F.; validation, C.H.; formal analysis, C.H. and A.K.; investigation, C.H.; resources, C.H. and A.K.; data curation, C.H.; writing—original draft preparation, C.H. and A.K.; writing—review and editing, C.H. and A.K.; visualisation, C.H.; supervision, H.M.; project administration, H.M.; funding acquisition, H.M. All authors have read and agreed to the published version of the manuscript.

References

- [1] A. Kameke, S. Kastens, S. Rüttinger, S. Herres-Pawlis, and M. Schlüter, “How coherent structures dominate the residence time in a bubble wake: An experimental example,” *Chemical Engineering Science*, vol. 207, pp. 317–326, Nov. 2019. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0009250919305366>
- [2] F. Huhn, A. von Kameke, V. Pérez-Muñuzuri, M. J. Olascoaga, and F. J. Beron-Vera, “The impact of advective transport by the South Indian Ocean Countercurrent on the Madagascar plankton bloom: Jet in Madagascar Plankton Bloom,” *Geophysical Research Letters*, vol. 39, no. 6, pp. n/a–n/a, Mar. 2012. [Online]. Available: <http://doi.wiley.com/10.1029/2012GL051246>

- [3] F. d’Ovidio, V. Fernández, E. Hernández-García, and C. López, “Mixing structures in the Mediterranean Sea from finite-size Lyapunov exponents: Mixing Structures in the Mediterranean Sea,” *Geophysical Research Letters*, vol. 31, no. 17, pp. n/a–n/a, Sep. 2004. [Online]. Available: <http://doi.wiley.com/10.1029/2004GL020328>
- [4] J. B. Weiss and A. Provenzale, Eds., *Transport and Mixing in Geophysical Flows: Creators of Modern Physics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. [Online]. Available: <http://link.springer.com/10.1007/978-3-540-75215-8>
- [5] S. Balasuriya, N. T. Ouellette, and I. I. Rypina, “Generalized Lagrangian coherent structures,” *Physica D: Nonlinear Phenomena*, vol. 372, pp. 31–51, Jun. 2018. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167278917302750>
- [6] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, and G. Haller, “A critical comparison of Lagrangian methods for coherent structure detection,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 27, no. 5, p. 053104, May 2017. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.4982720>
- [7] M. J. Olascoaga and G. Haller, “Forecasting sudden changes in environmental pollution patterns,” *Proceedings of the National Academy of Sciences*, vol. 109, no. 13, pp. 4738–4743, Mar. 2012. [Online]. Available: <https://pnas.org/doi/full/10.1073/pnas.1118574109>
- [8] J. H. Bettencourt, C. López, and E. Hernández-García, “Oceanic three-dimensional Lagrangian coherent structures: A study of a mesoscale eddy in the Benguela upwelling region,” *Ocean Modelling*, vol. 51, pp. 73–83, Jul. 2012. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S1463500312000649>
- [9] J. Curbelo and M. Linz, “Lagrangian Coherent Structures to Examine Mixing in the Stratosphere,” May 2024. [Online]. Available: <https://egusphere.copernicus.org/preprints/2024/egusphere-2024-1348/>
- [10] D. Garaboa-Paz, J. Eiras-Barca, F. Huhn, and V. Pérez-Muñuzuri, “Lagrangian coherent structures along atmospheric rivers,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, no. 6, p. 063105, Jun. 2015. [Online]. Available: <https://pubs.aip.org/cha/article/25/6/063105/322277/Lagrangian-coherent-structures-along-atmospheric>
- [11] W. Tang, M. Mathur, G. Haller, D. C. Hahn, and F. H. Ruggiero, “Lagrangian Coherent Structures near a Subtropical Jet Stream,” *Journal of the Atmospheric Sciences*, vol. 67, no. 7, pp. 2307–2319, Jul. 2010. [Online]. Available: <https://journals.ametsoc.org/doi/10.1175/2010JAS3176.1>
- [12] M. A. Green, C. W. Rowley, and G. Haller, “Detection of Lagrangian coherent structures in three-dimensional turbulence,” *Journal of Fluid Mechanics*, vol. 572, pp. 111–120, Feb. 2007. [Online]. Available: https://www.cambridge.org/core/product/identifier/S0022112006003648/type/journal_article
- [13] Justin Finn, Romain Watteaux, and Andrew Lawrie, “libcfd2lcs: A general purpose library for computation of Lagrangian coherent structures during CFD simulation,” Feb. 2017. [Online]. Available: <https://www.archer.ac.uk/training/virtual/2016-06-22-libcfd2lcs/ArcherVT.pdf>
- [14] J. Finn and S. V. Apte, “Integrated computation of finite-time Lyapunov exponent fields during direct numerical simulation of unsteady flows,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 23, no. 1, p. 013145, Mar. 2013. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.4795749>
- [15] M. Farazmand and G. Haller, “Computing Lagrangian coherent structures from their variational theory,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 22, no. 1, p. 013128, Mar. 2012. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.3690153>
- [16] 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*, vol. 212, no. 3–4, pp. 271–304, Dec. 2005. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167278905004446>
- [17] G. Haller, “Lagrangian Coherent Structures,” *Annual Review of Fluid Mechanics*, vol. 47, no. 1, pp. 137–162, Jan. 2015. [Online]. Available: <https://www.annualreviews.org/doi/10.1146/annurev-fluid-010313-141322>
- [18] S. L. Brunton and C. W. Rowley, “Fast computation of finite-time Lyapunov exponent fields for unsteady flows,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 1, p. 017503, Mar. 2010. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.3270044>
- [19] S. Leung, “An Eulerian approach for computing the finite time Lyapunov exponent,” *Journal of Computational Physics*, vol. 230, no. 9, pp. 3500–3524, May 2011. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0021999111000799>
- [20] G. Haller, “Lagrangian coherent structures from approximate velocity data,” *Physics of Fluids*, vol. 14, no. 6, pp. 1851–1861, Jun. 2002. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.1477449>
- [21] M. H. Sulman, H. S. Huntley, B. Lipphardt, and A. Kirwan, “Leaving flatland: Diagnostics for Lagrangian coherent structures in three-dimensional flows,” *Physica D: Nonlinear Phenomena*, vol. 258, pp. 77–92, Sep. 2013. [Online]. Available: <https://linkinghub.elsevier.com/retrieve/pii/S0167278913001450>
- [22] T. Solomon and J. Gollub, “Chaotic Transport in Time-Dependent Rayleigh-Bénard Convection,” *Physical review. A*, vol. 38, pp. 6280–6286, Jan. 1989.
- [23] D. Lipinski and K. Mohseni, “A ridge tracking algorithm and error estimate for efficient computation of Lagrangian coherent structures,” *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 20, no. 1, p. 017504, Mar. 2010. [Online]. Available: <http://aip.scitation.org/doi/10.1063/1.3270049>