Journal of Computational Science 65 (2022) 101883

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

An improved numerical method for hyperbolic Lagrangian Coherent Structures using Differential Algebra

Jack Tyler *, Alexander Wittig

Astronautics Research Group, University of Southampton, Southampton, SO17 1BJ, United Kingdom

ARTICLE INFO

MSC:
65P40
65L15

Keywords:
Lagrangian Coherent Structures
Differential Algebra
Transport barriers
Automatic differentiation

ABSTRACT

In dynamical systems, it is advantageous to identify regions of flow which can exhibit maximal influence on nearby behaviour. Hyperbolic Lagrangian Coherent Structures have been introduced to obtain two-dimensional surfaces which maximise repulsion or attraction in three-dimensional dynamical systems with arbitrary time-dependence. However, the numerical method to compute them requires obtaining derivatives associated with the system, often performed through the approximation of divided differences, which can lead to significant numerical error and numerical noise. In this paper, we introduce a novel method for the numerical calculation of hyperbolic Lagrangian Coherent Structures using Differential Algebra called DA-LCS. As a form of automatic forward differentiation, it allows direct computation of the Taylor expansion of the flow, its derivatives, and the eigenvectors of the associated strain tensor, with all derivatives obtained algebraically and to machine precision. It does so without *a priori* information about the system, such as variational equations or explicit derivatives. We demonstrate that this can provide significant improvements in the accuracy of the Lagrangian Coherent Structures identified compared to finite-differencing methods in a series of test cases drawn from the literature. We also show how DA-LCS uncovers additional dynamical behaviour in a real-world example drawn from astrodynamics.

1. Introduction

In dynamical systems, it is often useful to identify surfaces which separate or maximally influence regions of qualitatively different flow. For time-independent systems, one often determines the geometric location of the invariant manifolds, which partition phase space and are found by studying the system's behaviour over infinite time scales [1]. However, in time-aperiodic flows, such infinite-time behaviour is not always well defined. Instead, the behaviour of these systems is typically studied over fixed time-scales chosen to match some practical period of interest [2,3].

To overcome this problem, several methods for identifying analogous structures to the invariant manifolds in temporally aperiodic systems have been suggested. For example, one may study a number of heuristic flow diagnostics [4,5], such as the Finite-Time Lyapunov exponent (FTLE) which quantifies the separation between two trajectories which start out infinitesimally close. However, many of these methods are only effective for simple flows and are dependent on the reference frame [2]. Being heuristic, they also often lack a proper theoretical foundation as to exactly what they are indicating.

Lagrangian Coherent Structures (LCS) have been proposed to solve this problem [6]. A particular type of LCS, the hyperbolic LCS, is locally the most repelling or attracting surface in a given region of flow, and plays an analogous role to the stable and unstable manifolds. Several equivalent definitions of LCS have arisen in the literature (for a review, see [7]).

A global, objective approach to the practical construction of Lagrangian Coherent Structures based on their variational theory was presented in [8]. The authors provide both the theoretical underpinning and a practical algorithm to directly construct LCS as parameterised surfaces by growing material surfaces which impose locally extreme deformation on nearby sets of initial conditions. These surfaces are shown to be necessarily orthogonal to certain eigendirections of the Cauchy–Green strain tensor, $C_{t_0}^T$, and further satisfy a certain criterion involving the curl of the eigenvectors of $C_{t_0}^T$ to ensure the surface is maximally repelling or attracting. This approach is valid for three-dimensional flows with general time-dependence and over arbitrarily-chosen time periods of observation.

However, there are several computational complexities associated with computing LCS using this approach [2], such as the need to account for degenerate points and orientational discontinuities in the eigenvector field of $C_{t_0}^T$. More importantly, the eigenvectors of $C_{t_0}^T$ must be computed precisely, yet are very sensitive to numerical errors. These

* Corresponding author.
E-mail addresses: jack.tyler@soton.ac.uk (J. Tyler), a.wittig@soton.ac.uk (A. Wittig).

https://doi.org/10.1016/j.jocs.2022.101883
Received 24 June 2022; Received in revised form 7 October 2022; Accepted 10 October 2022
Available online 23 October 2022

errors are particularly troublesome near regions of intense attraction or repulsion, since large errors in $C^T_{t_0}$ can quickly accumulate, yet these are also the exact regions where one would expect a hyperbolic LCS. The approximation of the derivatives of a flow using finite differencing is often used [2,9–11], but this method is particularly sensitive to the grid size chosen, which must be carefully selected to account for flow behaviour over different spatial scales, which is generally difficult to determine *a priori* and often selected through trial-and-error. Other such methods for approximating derivatives exist, such as the use of variational equations, where one manually derives and implements a set of adjoint differential equations that are propagated along with a reference trajectory [12]. While this approach yields derivatives as accurate as the propagation along the reference trajectory, it requires the derivation, implementation and integration of n^2 additional equations for the first derivatives of a n dimensional flow, and another $n^2(n+1)/2$ equations for the second flow derivatives. An alternative Eulerian approach for approximating $C^T_{t_0}$ without the need for divided differences was presented in [13] by the solution of a set of partial differential equations (PDEs). However, this does not extend to the computation of the derivatives of the eigenvectors of $C^T_{t_0}$ and in some cases the Eulerian approach via the solution of PDEs may be more computationally expensive than the equivalent Lagrangian approach.

Separately, Differential Algebra (DA) was originally introduced to compute high-order transfer maps for particle accelerator systems [14]. This approach constructs a Taylor series representation of an arbitrary map in a dynamical system, and has since seen widespread use in the study of non-linearities [15–17], the management of uncertainties [18–21], and as a form of automatic differentiation [22] in a wide variety of fields. Unlike other numerical methods such as divided differences, the derivatives found using DA are accurate to machine precision, and since it is a form of automatic differentiation there is no need to derive or implement any additional equations beyond the system itself. However, unlike standard automatic differentiation packages, we have additional access to a Taylor expansion about the reference point, which can be manipulated directly including by partial derivative operators (see Section 3.2), as suggested by the name Differential Algebra [18].

In this paper we introduce DA-LCS, which uses DA to improve the numerical method presented in [8] for determining hyperbolic LCS. Firstly, in Section 3.1 we briefly review how polynomial expansions of arbitrary flows of an ordinary differential equation (ODE) can be calculated, with applications to obtaining flow derivatives of arbitrary systems to machine precision. Next, in Section 3.2 we introduce a novel use of DA to construct algebraic expansions of the leading eigenvector of a matrix of polynomials. Both of these techniques are then combined to form the DA-LCS algorithm for computing LCS in three-dimensional flows. In Section 5, we demonstrate that this method works well in reproducing results for commonly-used 'toy' problems from the literature. Lastly, in Section 6 we present the application of DA-LCS to a more complex system from astrodynamics where the traditional method of divided differences fails to produce usable results in the literature [24,25].

2. Mathematical background and notation

We study the behaviour of a dynamical system

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x},\ t), \boldsymbol{x} \in D \subset \mathbb{R}^n,\ t \in \left[t_0,\ t_0 + T\right] \tag{1}$$

where f is a smooth vector field considered over some time T starting at time t_0. Denoting a trajectory of the dynamical system starting at position $\boldsymbol{x}_0$ at time t_0 as $\boldsymbol{x}\left(t_0,\ \boldsymbol{x}_0;\ T\right)$, the flow map of Eq. (1) is given by

$$\boldsymbol{F}^T_{t_0} : \begin{cases} D \to D \\ \boldsymbol{x}_0 \mapsto \boldsymbol{x}\left(t_0,\ \boldsymbol{x}_0;\ T\right) \end{cases} \tag{2}$$

which is assumed to be at least k-times continuously differentiable. The Jacobian of this flow map, $\nabla \boldsymbol{F}^T_{t_0}$, defines the right Cauchy–Green Strain Tensor (CGST) $C^T_{t_0}$, which describes the local deformation of the flow at the end of a given trajectory.

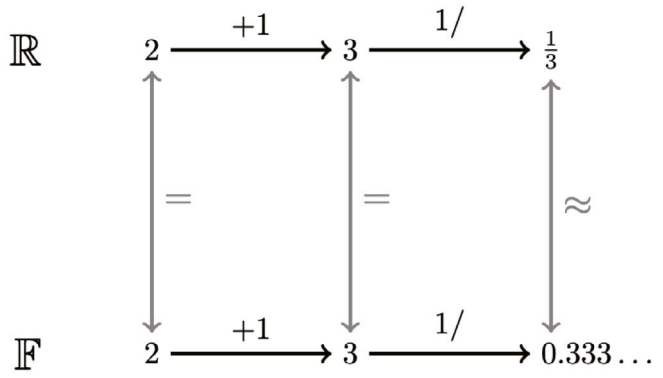
$$C^T_{t_0} = \left(\nabla \boldsymbol{F}^T_{t_0}\right)^\top \left(\nabla \boldsymbol{F}^T_{t_0}\right) \tag{3}$$

with $^\top$ denoting the matrix transpose. $C^T_{t_0}$ is positive-definite and symmetric, with real eigenvalues $\lambda_1 \le \lambda_2 \le \cdots \le \lambda_n$ and associated real eigenvectors $\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \ldots, \boldsymbol{\zeta}_n$.

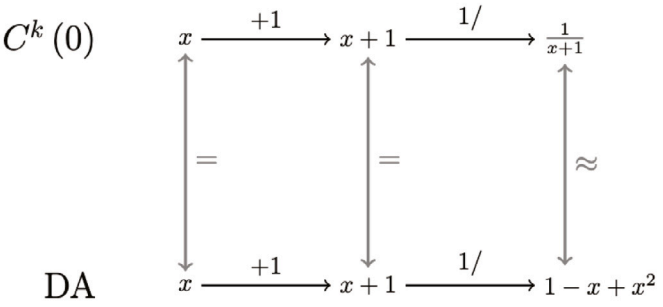
The dominant eigenvalue λ_n can be used to calculate the finite-time Lyapunov exponent (FTLE), a measure of maximum separation of two particles advected forward under Eq. (1) that start out infinitesimally close to each other:

$$\sigma^T_{t_0} = \frac{1}{2}\frac{\log \lambda_n}{T}. \tag{4}$$

Many previous studies have leveraged the FTLE field as a heuristic indication of separation in the flow. While the FTLE has been shown to be insufficient to indicate LCS alone [26], the FTLE is a commonly-used metric and is thus used in this paper to preliminarily highlight system behaviour.



(a) Evaluation of $1/(2+1)$ in the field of real numbers $\mathbb{R}$ (top) and in the floating-point approximation to $\mathbb{R}$, $\mathbb{F}$ (bottom). Each operation in $\mathbb{R}$ has a corresponding operation in $\mathbb{F}$.



(b) Evaluation of $1/(x+1)$ in the k-times differentiable functions C^k (top) and truncated polynomials of order 2 represented by DA (bottom). Each operation in $C^k(0)$ has a corresponding operation in DA, approximating the resulting function in $C^k(0)$ by its Taylor expansion around 0.

Fig. 1. Comparison between the field of real numbers $\mathbb{R}$ and function space C^k, and their respective computer representations.
Source: The subfigures are taken from [23].

3. Differential algebra

In the following, we give a very brief introduction to Differential Algebra. For a more comprehensive treatment, the reader is referred to the literature [27].

Differential Algebra can be used as a tool to compute the derivatives of functions within a computer environment [27,28]. Similar to how

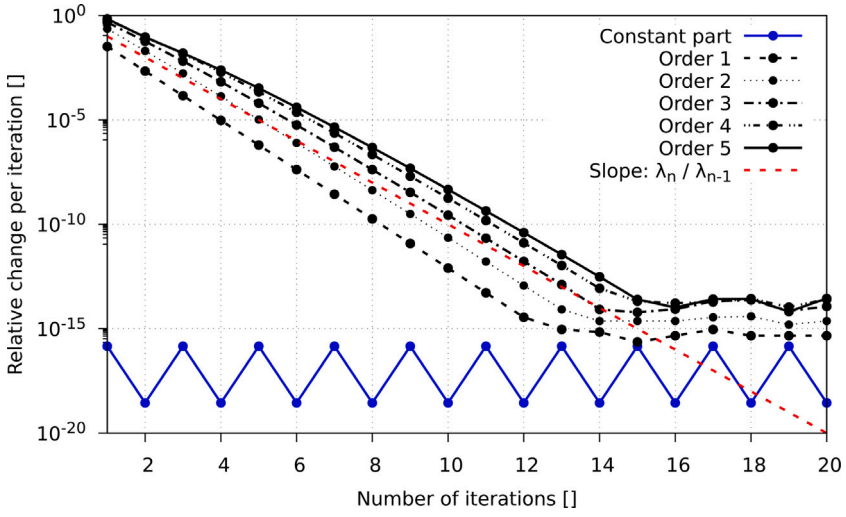


Fig. 2. Relative error across all polynomial orders in successive applications of $\left[C_{t_0}^T\right]$ to an initial guess containing only the floating-point dominant eigenvector at the expansion point as the constant part. Higher expansion orders (black) can all be seen converging at around the expected convergence rate λ_n/λ_{n-1} (dashed red) towards the floating-point floor.

computers represent the field of real numbers as floating-point numbers, DA allows the representation and manipulation of functions in a computer [29].

Consider two real numbers a and $b \in \mathbb{R}$. The approximation to a and b in a computational environment is their floating-point representation $\bar{a}, \bar{b} \in \mathbb{F}$, which essentially stores a set number of digits of its binary expansion. Any operation defined in $\mathbb{R}$, $\square$, has a corresponding operation in $\mathbb{F}$, $\boxtimes$, defined such that the result is another floating-point approximation of the operation on the real numbers a and b, i.e. $\bar{a} \times \bar{b}$ commutes with the floating-point representation of $a \times b$, $\overline{a \times b}$.

Similarly, now consider two functions, c and d, which are sufficiently smooth, $k-$differentiable functions of n variables: $c, d : \mathbb{R}^n \to \mathbb{R}$. In the DA framework, a computer operates on the multivariate Taylor expansion of c and d, $[c]$ and $[d]$, with corresponding operations to those defined in the real function space, such that the operation of $[c] \cdot [d]$ commutes with the DA representation of the product $[c \cdot d]$.

An example to demonstrate how real numbers are approximated in a computer environment is provided in Fig. 1(a) for the evaluation of the expression $1/(x+1)$ for $x = 2$ in $\mathbb{F}$ and $\mathbb{R}$. In Fig. 1(a), we begin with $x = 2$, perform the operation $+1$ to obtain 3, and then perform the operation $1/$ to compute the final expression. In $\mathbb{R}$, we obtain the solution $1/3$, and in $\mathbb{F}$ we obtain the solution $0.333\ldots$ up to the limit of precision of the type. The final result of the evaluation in floating-point arithmetic is an approximation of the real computation.

Analogously, in Fig. 1(b) we evaluate the expression $1/(1+x)$ in the space $C^k(0)$ of real functions, and a DA representation of expansion order 2. We begin with the function $c(x) = x$, and perform the operation $+1$ followed by the operation $1/$, yielding $1/(x+1)$ in the real function space, and $1 - x + x^2$ in the DA arithmetic. The result of the DA arithmetic is the Taylor expansion of $1/(x+1)$ which represents the function exactly at $x = 0$, and approximates the function locally near $x = 0$ with an error of $\mathcal{O}\left(x^3\right)$. The coefficients of the expansion are computed automatically without any further input from the user.

Differential Algebra comprises the full set of elementary operations to efficiently operate on multivariate expansions, including operations for common intrinsic functions such as division, square roots, trigonometric functions, and exponentials, as well as operations for differentiation and integration. An important application of DA widely used in both the literature and this paper is the high-order expansion of the solution of an ODE as a function of the initial conditions [19,21], which is discussed in more detail in Section 3.1. In this paper, we use the Differential Algebra Computational Engine [22] (DACE) to operate on polynomial expansions ('DA objects' or 'DAs').

3.1. Flow expansions to arbitrary order using differential algebra

A key advantage of using DA is that the derivatives of flows with respect to the initial conditions can be obtained automatically and without any further effort from the user, beyond implementing the system's governing equations and the numerical integration scheme in DA arithmetic.

To illustrate this concept of flow expansion, suppose we solve the following initial value problem (IVP) numerically using a forward Euler scheme, the simplest of the Runge–Kutta family of numerical integrators

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x},\ t) \\ \boldsymbol{x}(t_i) = \boldsymbol{x}_i. \end{cases} \tag{5}$$

A single step in this scheme is given explicitly by

$$\boldsymbol{x}_i = \boldsymbol{x}_{i-1} + \Delta t f\left(\boldsymbol{x}_{i-1}\right) \tag{6}$$

which can be expressed as a function of the initial condition $\boldsymbol{x}_0$,

$$\boldsymbol{x}_f = \boldsymbol{x}_0 + \sum_{i=0}^{n} \Delta t\, f\left(\boldsymbol{x}_0 + i \cdot \Delta t\right) \tag{7}$$

i.e. the initial condition is simply a sequence of operations on the initial condition, which is true for any numerical integrator of the Runge–Kutta family.

If we set $\boldsymbol{x}_0$ to be a DA representation of the initial condition by substituting the initial value with the DA identity, $\left[\boldsymbol{x}(t_0)\right] = \boldsymbol{x}(t_0) + \delta\boldsymbol{x}$, then $\boldsymbol{x}_f$ becomes a DA representation of the final condition as a function of the initial condition, $\left[\boldsymbol{x}_f\right]$. Differentiating the polynomial thus yields the derivatives of the final condition with respect to the initial condition completely algebraically.

As mentioned, the numerical integrator must support DA operations. Using Boost C++, which has operator overloading to operate on any type, this is relatively straightforward and its 7th/8th order Dormand–Prince method is used in this paper. However, care must be taken when calculating norms for error estimation in the integrator when using DA. Evaluating the usual L_2 norm of a vector $|\boldsymbol{x}| = \sqrt{\sum_{i=0}^{n} x_i^2}$ in DA yields another DA object representing a polynomial. As there is no ordering on the space of polynomials, this cannot be directly compared to some tolerance. Instead, we have to define the norm of a DA object which maps it into the non-negative real numbers. In this application, the norm of a DA object is taken to be the largest absolute value of any coefficient of the expansion in any order. Considering all orders in the

norm allows the usual step-size control algorithms of embedded Runge–Kutta methods to control the error in all orders of the expansion, rather than just the constant part.

3.2. Polynomial expansions of leading eigenvectors of $C^T_{t_0}$ to arbitrary order

Since derivatives of polynomials are straightforward to compute, we can apply the partial derivative operator to differentiate the $j-$th variable of an expansion, ∂_j, making it particularly easy to assemble an expansion of $C^T_{t_0}$. This means we can directly evaluate the Jacobian as

$$\left[\nabla \boldsymbol{F}^T_{t_0}\right]_{ij} = \partial_j \left[\boldsymbol{x}\right]^T_{t_0,\,i} \tag{8}$$

from which a polynomial expansion of $C^T_{t_0}$ can be assembled

$$\left[C^T_{t_0}\right] = \left[\nabla \boldsymbol{F}^T_{t_0}\right]^\top \left[\nabla \boldsymbol{F}^T_{t_0}\right]. \tag{9}$$

Note that the constant part of $\left[C^T_{t_0}\right]$ is the CGST at the expansion point accurate to machine precision, that is it is the same as would be approximated with divided differences. The remaining higher order terms represent an expansion of the CGST in the neighbourhood around the expansion point.

To compute the LCS, the derivatives of the leading eigenvector of the Cauchy–Green strain tensor with respect to position are required. While divided differences can in principle again be used to obtain these derivatives, the method is susceptible to numerical noise and it is difficult to determine the most appropriate grid sizes to use. Moreover, eigenvectors are only defined up to a sign, and thus care must be taken when taking the derivatives that nearby eigenvectors have 'smooth' changes in orientation.

Instead, we use a novel application of DA to obtain an expansion of the leading eigenvector of a matrix of DAs, which then can once again be differentiated directly in DA. We simply use power (von Mises) iteration [30] performed in DA, which is a well-established algorithm in standard floating-point operations [31].

Power iteration performs the repeated evaluation of an arbitrary starting vector $\boldsymbol{b}_0$ through a matrix A to obtain an approximation to its dominant unit eigenvector $\boldsymbol{b}$ through the recurrence relation

$$\boldsymbol{b}_{m+1} = \frac{A\boldsymbol{b}_m}{\|A\boldsymbol{b}_m\|} \tag{10}$$

where $\|\cdot\|$ represents a vector norm, here taken to be the L_2 norm, and m is the number of iterations. The vector $\boldsymbol{b}$ will converge provided that the starting vector $\boldsymbol{b}_0$ has a nonzero component in the direction of the dominant eigenvector, and A has a unique largest eigenvalue by absolute value. The theoretical convergence rate of the method between successive iterations is the ratio of the dominant eigenvalue to the second dominant eigenvalue. Practically, the recurrence relation is iterated until the stopping condition $\|\boldsymbol{b}_{m+1} - \boldsymbol{b}_m\| \leq \varepsilon$ is valid, where $\varepsilon > 0$ is a pre-set tolerance and the norm is again taken to be an L_2 norm.

To convert this algorithm to DA, let A now be a DA matrix with DA objects in each entry, $[A]$. Iterating it on a DA vector $[\boldsymbol{b}_0]$ will yield a DA vector $[\boldsymbol{b}]$ corresponding to the dominant eigenvector of $[A]$ with a polynomial expansion in each entry, that is it is the recurrence relation

$$[\boldsymbol{b}]_{m+1} = \frac{[A]\,[\boldsymbol{b}]_m}{\|\,[A]\,[\boldsymbol{b}]_m\,\|}. \tag{11}$$

Note that here the norm in the denominator is simply a DA evaluation of the L_2 (Euclidean) norm $[|\boldsymbol{x}|] = \sqrt{\sum_{i=0}^{n} [x]_i^2}$. We generalise the stopping condition from floating-point computation such that we iterate until there is no relative change in any order in any entry of $\left([\boldsymbol{b}]_{m+1} - [\boldsymbol{b}]_m\right)$ above a pre-set tolerance $\varepsilon > 0$. We set ε to be 10^{-12} in this paper.

To speed up convergence, and because eigenvector solvers for floating-point computations are readily available and highly efficient, we set the initial guess for $[\boldsymbol{b}_0]$ to have a constant part equal to the dominant eigenvector of the constant part of $[A]$, since we know by construction that this will be the constant part of the expansion of the dominant eigenvector.

An example of the convergence of this method is illustrated in Fig. 2, which shows the maximum relative change of coefficients in $[\boldsymbol{b}]$ separated by their expansion order over repeated application of $\left[C^T_{t_0}\right]$ to the initial guess of the dominant eigenvector of a trajectory in the periodic ABC flow (Section 5.2). The theoretically expected rate of convergence λ_n/λ_{n-1} can clearly be seen in the plot as a dashed red line. All orders converge at approximately the expected rate and the floating-point portion of the expression converges instantly as it was already set to the double-precision representation of the leading eigenvector.

Once the eigenvector $[\boldsymbol{\zeta}_n]$ is expanded to at least first order, the curl $\nabla \times \boldsymbol{\zeta}_n$ of the eigenvector field, which is used in the LCS construction (Section 4), can be computed by simply applying the DA partial derivative operator ∂_j again.

To obtain the value of $\nabla \times \boldsymbol{\zeta}_n$ at the expansion point, the flow map $\boldsymbol{F}^T_{t_0}$ must be computed at least to order 2. This is because one derivative is taken in the construction of $C^T_{t_0}$ (Section 3.1), and another is then taken in $\nabla \times \boldsymbol{\zeta}_n$, both of which reduce the order of the expansion by one.

4. Lagrangian coherent structures

In this Section we give the method for computing hyperbolic LCS in three-dimensional systems. We begin by formally reviewing their mathematical construction following [8] and highlight how this is implemented algorithmically and practically following the literature [8,26,32]. We then highlight how DA is used to enhance the algorithm in DA-LCS.

The full, three-dimensional hyperbolic LCS, which is the locally maximally repelling or attracting surface over a given time interval $[t_0, t_0 + T]$, is constructed from its intersections with a family of hyperplanes S. These intersections are called *reduced strainlines* or *reduced stretchlines*. Interpolating between these intersections yields the full, three-dimensional structure of the LCS. In the following, we highlight the construction of the repelling LCS, whose structure is derived from the dominant eigenvector $\boldsymbol{\zeta}_3$ and whose intersections with S are the reduced strainlines. A similar procedure applies to $\boldsymbol{\zeta}_1$ (reduced stretchlines) to obtain attracting LCS.

Mathematically, at any point $\boldsymbol{s}$ on a hyperplane, we define the reduced strainline that passes through that point from the definition of the repelling LCS as being necessarily orthogonal to $\boldsymbol{\zeta}_3$ and of course lying within the hyperplane. This is true for any point on the strainline, allowing their parameterisation to be described by the ODE

$$\boldsymbol{s}' = \hat{\boldsymbol{n}}_S \times \boldsymbol{\zeta}_3 \tag{12}$$

where $\hat{\boldsymbol{n}}_S$ is the unit normal to the surface at $\boldsymbol{s}$. Points with zero *helicity* $H_{\boldsymbol{\zeta}_3}$ lie on surfaces which are maximisers of repulsion

$$H_{\boldsymbol{\zeta}_3} = \langle \nabla \times \boldsymbol{\zeta}_3, \boldsymbol{\zeta}_3 \rangle, \tag{13}$$

where $\langle \cdot, \cdot \rangle$ is the inner product. Reduced strainlines which begin from points with zero helicity are thus the intersection of the LCS with that hyperplane. The strainlines forming part of the LCS on each hyperplane are then interpolated to produce the full 3D structure of the LCS.

Practically, Algorithm 1 is used to solve for the two quantities above following the literature [8,26,32]. We first sample points $\boldsymbol{s}$ on each hyperplane in S on a uniformly-spaced grid and compute the derivative of the flow map $\nabla \boldsymbol{F}^T_{t_0}$ and thus $C^T_{t_0}$ at each point (Lines 3 and 4). The dominant eigenvector of $C^T_{t_0}$, $\boldsymbol{\zeta}_3$, and its curl, $\nabla \times \boldsymbol{\zeta}_3$, is obtained and then used to compute the helicity (Lines 5 and 6).

Numerically, the selection of zero-helicity initial conditions for the strainline ODE in Eq. (12) is relaxed by allowing them to begin from points where the helicity $H_{\boldsymbol{\zeta}_3}$ is below some tolerance $\alpha > 0$ (Line 7).

The ODE in Eq. (12) is then rewritten in discretised form and solved numerically as

$$\boldsymbol{s}_i' = \text{sign}\left(\boldsymbol{\zeta}_{i,3} \cdot \boldsymbol{\zeta}_{i-1,3}\right) \hat{\boldsymbol{n}}_S \times \boldsymbol{\zeta}_{i,3} \tag{14}$$

where $\boldsymbol{s}_i$ is the $i-$th point on the strainline L and the term $\text{sign}\left(\boldsymbol{\zeta}_{i,3} \cdot \boldsymbol{\zeta}_{i-1,3}\right)$ is introduced to enforce continuity in the vector field by selecting the direction most closely aligned with the previous tangent vector. Since the eigenvector is only defined up to the sign, we integrate the strainline in both directions corresponding to $\pm\boldsymbol{\zeta}_3$ to capture the entire strainline structure. The numerical integration of the ODE along the strainline continues until the sum of the helicity at each $\boldsymbol{s}_i$ divided by the number of steps performed (i) rises above α (Line 10).

Algorithm 1: High-level algorithm for computing three-dimensional LCS

Input: S, α, δ, t_0, T

1: **for** hyperplane S in S **do**
2: **for** point $\boldsymbol{s}$ on hyperplane S **do**
3: Compute $\nabla \boldsymbol{F}_{t_0}^T$ at $\boldsymbol{s}$
4: $C_{t_0}^T \leftarrow \left(\nabla \boldsymbol{F}_{t_0}^T\right)^\top \left(\nabla \boldsymbol{F}_{t_0}^T\right)$
5: $\boldsymbol{\zeta}_3 \leftarrow$ dominant eigenvector of $C_{t_0}^T$
6: $H_{\boldsymbol{\zeta}_3} \leftarrow \langle \nabla \times \boldsymbol{\zeta}_3, \boldsymbol{\zeta}_3 \rangle$
7: **if** $H_{\boldsymbol{\zeta}_3} \leq \alpha$ **then**
8: $L \leftarrow$ new strainline starting at $\boldsymbol{s}$
9: **while** average of $H_{\boldsymbol{\zeta}_3}$ along $L \leq \alpha$ **do**
10: Extend L via ODE (12)
11: **end while**
12: Add L to set of strainlines on S, $\mathcal{L}_S$
13: **end if**
14: **end for**
15: Filter $\mathcal{L}_S$ for duplicate strainlines L using distance metric and threshold δ
16: **end for**
17: Interpolate strainlines in $\mathcal{L}_S$ and $\mathcal{L}_{S'}$ for all adjacent hyperplanes S and S' in S

The trajectories obtained through the strainline ODE are segments of strainlines forming the LCS. However, since different initial points can belong to the same strainline, the trajectories often overlap. They must, therefore, be filtered to provide a single, continuous curve. Given a suitable metric d_F of how close two strainline segments are, the shorter of the two strainlines is discarded whenever d_F is below some threshold $\delta > 0$ (Line 15).

We now highlight our additions to the algorithm in DA-LCS. In previous literature, divided differences was used to numerically approximate $\nabla \boldsymbol{F}_{t_0}^T$ in Line 3, and the quantity $\nabla \times \boldsymbol{\zeta}_3$ in Line 5 [2,9–11]. This can lead to significant numerical error, particularly when computing the second derivative required for $\nabla \times \boldsymbol{\zeta}_3$. Divided differences can either be applied on the same grid on which points are sampled, or on a finer grid used solely for the purpose of approximating the derivatives. In DA-LCS, we use the technique of flow expansion described in Section 3.1 to compute $\nabla \boldsymbol{F}_{t_0}^T$ as an expansion at each grid point and around each grid point. This provides an automatic representation of this quantity accurate to machine precision, and without the need to alter grid sizes through trial and error. It thus also provides $C_{t_0}^T$ and $\boldsymbol{\zeta}_3$ in Lines 4 and 5 to high accuracy for use in the strainline ODE. To improve the numerically-challenging computation of $\nabla \times \boldsymbol{\zeta}_3$, we compute Line 6 to machine precision using the power iteration described in Section 3.2, and again without the need to alter grid sizes through trial-and-error.

In [8], the distance metric used was the Hausdorff distance, a measure of similarity between two curves. While the Fréchet distance is recognised as a better measure of similarity than the Hausdorff distance in trajectory clustering problems [33,34], we find we obtain qualitatively better strainlines when retaining the Hausdorff distance. It is defined as follows: given two curves A and B, the Hausdorff filtering distance d_F between A and B is defined such that

$$d_F = \max\left\{ \max_{x \in A}\left(\min_{y \in B} d_E(x, y) \right), \max_{x \in B}\left(\min_{y \in A} d_E(x, y) \right) \right\} \tag{15}$$

where d_E is the Euclidean distance. As per other investigations into computing LCS [32], we also enforce a minimum strainline length of δ; the rationale behind this is given in Section 5.1.

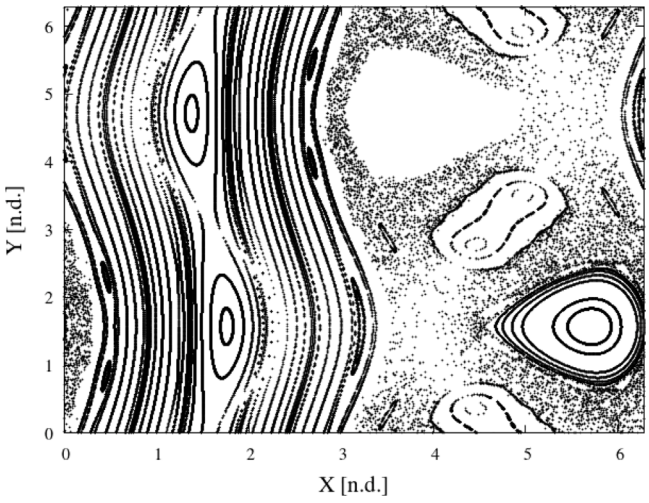


Fig. 3. Poincaré section (return map) for the steady ABC flow on the $z = 0$ plane; generated using a 15×15 grid of initial points with integration time $T = 1500$.

5. Arnold–Beltrami–Childress flows

To show that DA-LCS reproduces the results from the literature, we now apply the standard approach of divided differences and the DA-LCS method to several variations of the Arnold–Beltrami–Childress (ABC) flow, as studied in [8]. For each example, we present the equations of motion, the FTLE field using DA-LCS, and the relative error of the application of the approximation of divided differences to the FTLE field, using DA-LCS as the baseline. We also present the helicity field and the resulting strainlines. The results obtained using divided differences each use the manually-determined optimal grid size for each application that produces the qualitatively 'best' results, to allow for a fair comparison. Grid sizes between 0.1 and 5 times the nominal grid size were analysed. No such adjustments are needed when using DA-LCS.

5.1. Steady Arnold–Beltrami–Childress flow

We first consider the steady Arnold–Beltrami–Childress flow, using the problem parameters and reference planes presented in [8]. The ABC flow is an exact solution to Euler's equation, and its equations of motion in Cartesian coordinates are

$$\dot{x} = A \sin z + C \cos y \tag{16}$$

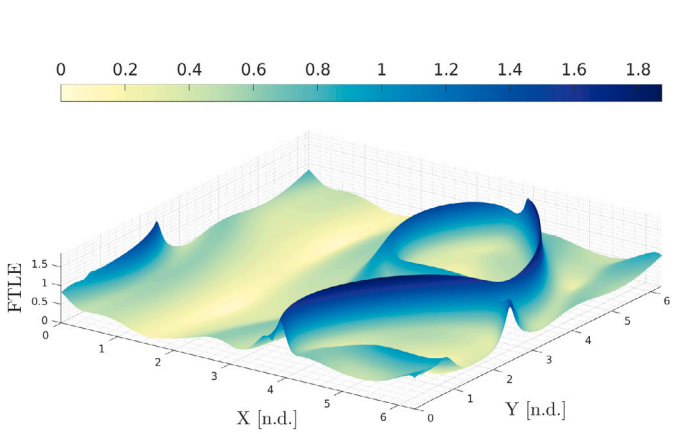
$$\dot{y} = B \sin x + A \cos z \tag{17}$$

$$\dot{z} = C \sin y + B \cos x \tag{18}$$

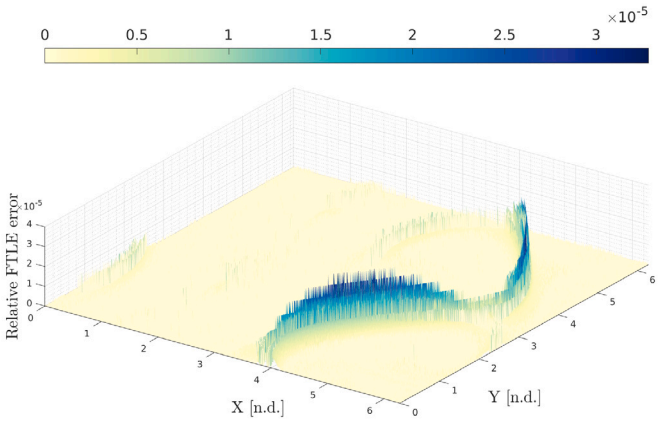
with parameter values $A = \sqrt{3}$, $B = \sqrt{2}$, $C = 1.0$. To illustrate the behaviour of this system, the Poincaré section in the x–y plane is shown in Fig. 3, computed from a regular 15×15 grid of initial points and an integration time of $T = 1500$.

For the LCS computation, matching previous literature the set of reference planes are taken to be

$$S = \{(x, y, z) \in [0, 2\pi]^3 \ : \ z \in \{0, 0.005, 0.01 \ldots, 0.1\}\},$$

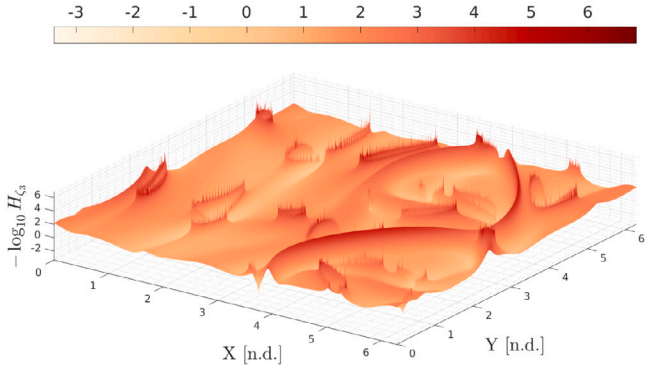


(a) FTLE field obtained using DA-LCS.

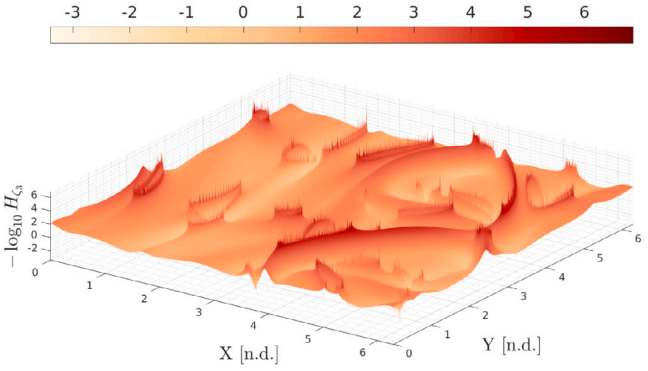


(b) Relative error between DA-LCS and the FTLE field obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in all directions.

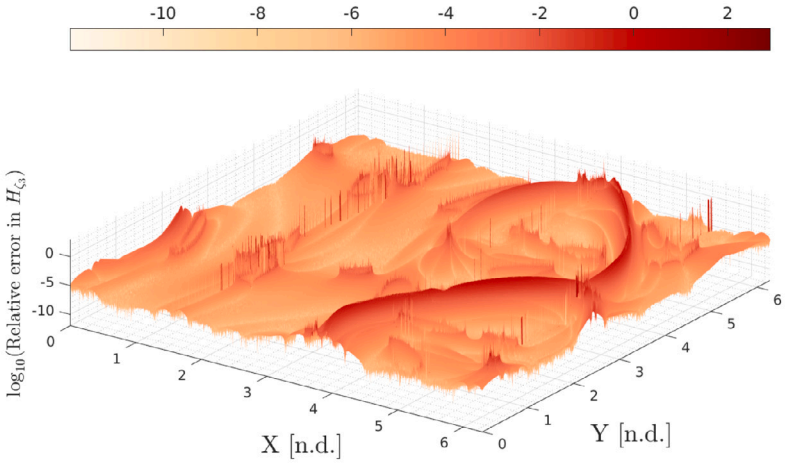
Fig. 4. Finite-time Lyapunov fields for the steady ABC flow from $t = 0$ to $T = 3$ using DA-LCS and divided differences. The relative error is below 3×10^{-5}, suggesting that computing the FTLE using divided differences is not a major source of error in this example.



(a) $-\log H_{\zeta_3}$ computed using DA-LCS.



(b) $-\log H_{\zeta_3}$ obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in both x and y. The same grid is used in computing both C_0^3 and $\nabla \times \zeta_3$.



(c) Relative error in the helicity field obtained through using DA-LCS compared to divided differences.

Fig. 5. Helicity fields for the steady ABC flow from $t_0 = 0$ to $T = 3$ using DA-LCS and divided differences. Qualitatively both strongly agree, showing that DA-LCS is working. However, the relative error on the ridges indicating very low helicity is of the order of 100, making the identification of seed points more robust with DA-LCS.

that is the x–y plane evenly spaced along the z axis. However, within each plane we alter the grid size used. [8] use a 500 × 500 grid on which to compute the underlying helicity field, and then sample seed points for the ODE in Eq. (12) on a reduced grid of 600 × 10. While

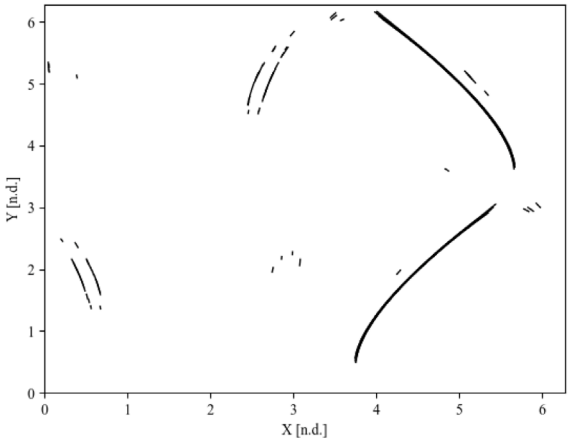


Fig. 6. Final, filtered strainlines for the steady ABC flow on the $z = 0$ plane computed using DA-LCS. The structure is formed of approximately 53 strainline segments.

Table 1
Core time required to compute the LCS on one reference plane for the steady ABC flow using divided differences and DA-LCS on Intel Xeon E5-2670 processors. While DA-LCS is slower to determine the initial H_{ζ_3} field, it is quicker at the integration of a representative set of strainlines and can grow much longer strainlines with the same number of evaluations of Eq. (12) as divided differences. Importantly, divided differences requires significant grid size tuning, which may make the time required to determine H_{ζ_3} slower overall when used practically.

Method	Time to compute H_{ζ_3} field [s]	Time to compute 100 strainlines [s]	Average function evaluations per unit length
Divided differences	224.902	4684.689	7498.171
DA-LCS	611.360	1108.571	78.392

the authors acknowledge that sampling every point on a dense grid is numerically inefficient, to simplify analysis, ensure we capture all of the flow's behaviour, and to work off of the assumption of no *a priori* knowledge we perform all stages of the analysis on a 1000×1000 grid defined for each hyperplane in $\mathcal{S}$. In practice, additional information about the system may be available to search more efficiently for LCS seed points, such as searching on a fixed line or only in a certain region of flow.

The system defined by Eqs. (16)–(18) is integrated forward for 3 non-dimensional time units using the DA-compatible numerical integrator introduced previously, with an integration tolerance of 10^{-13}. A helicity tolerance of $\alpha = 10^{-4}$ is applied to determine seed points and terminate the numerical integration. A minimum distance of $d_F = 0.04$ is used in the strainline segment filtering. These parameters are chosen from visual examination of the helicity field and resulting strainline structure for all of the examples in this paper.

The FTLE field on the $z = 0$ plane for this flow, computed using DA-LCS, is shown in Fig. 4(a). We show the relative error between the use of DA-LCS and the use of divided differences on the manually-determined 'optimal' grid-size in Fig. 4(b). The relative error is very low throughout the field, which suggests that the computation of C_0^3 and its dominant eigenvalue agrees across the two methods.

In the DA-LCS and divided difference helicity fields on the $z = 0$ plane, shown in Figs. 5(a) and 5(b) respectively, some first differences can be seen. The quantitative differences are highlighted in a plot of the relative error between them in Fig. 5(c). While the two methods qualitatively agree on the structure of the field, the DA-LCS method produces smoother peaks and ridges in the field for the primary features in the flow. This is particularly visible on the main ridge in the bottom right corner around $X = 4$ and $Y = 1$. This makes the identification of seed points in the flow more straightforward.

The resulting strainlines on the $z = 0$ plane for this flow are shown in Fig. 6, and follow the expected structure from the helicity field presented in Fig. 5(a) using 53 strainline segments. We note the existence of several 'loops' in the helicity field, particularly in the left-hand side of the field. The strainline segments at these points often grow transverse to the ridges at certain points, and do not track along the ridge as would be expected. This behaviour is also present when computing LCS with divided differences. These small strainline segments are not present in [8] due to being missed by the largely reduced 600×10 grid resolution used there. This explains their omission from the literature, and we do not investigate this issue further here, although we note the existence of similar structure in [35].

The 'spiky' nature of these loops, found partially as a result of our dense sampling of initial conditions, also means that those strainline segments that do grow are of exceedingly small length rather than being a continuous, low-helicity structure, as numerical integration of the strainline ODE is terminated immediately upon leaving the 'tip' of the spike. Ensuring we capture only strainlines which form part of a larger structure that exerts maximal influence on nearby flow, as per the definition of an LCS, motivates the use of a minimum strainline length equal to the filtering distance. In practice, one may wish to only filter the subset of longest strainlines, or only those with minimal average helicity. The use of a reduced grid, or a more sophisticated search strategy for low-helicity points, would also avoid the inclusion of such spurious points.

We now discuss the computational and numerical performance of DA-LCS, using the steady ABC flow as an example. The total time to compute the full LCS on 48 2.0 GHz Intel Xeon E5-2670 processors is given in Table 1, broken down by the time required to obtain the initial H_{ζ_3} field and then a representative set of 100 strainlines. The set of 100 strainlines is chosen to be the 100 points with lowest H_{ζ_3}, integrated until the running average of helicity rises above 10 times the initial value. Visual inspection of the initial conditions confirms that the seed points are sufficiently 'close' in both divided differences and DA-LCS that they are assumed to represent the same behaviour.

We find that DA-LCS is slower than divided differences for computing the initial helicity field since two orders are computed, requiring more CPU instructions per operation, and because fewer optimisations can be made by the compiler compared to native double-precision types. However, since DA-LCS requires no tuning of grid size, this computational deficit is eliminated as soon as more than two trial computations of the LCS using divided differences has to be performed to obtain the 'optimal' grid size in every dimension. Moreover, owing to better numerical performance, the strainline integration is approximately four times faster using DA-LCS than using divided differences, since the integrator can take larger steps than with divided differences while still controlling the error in the integration of Eq. (12). We also find that the strainlines obtained with DA-LCS are on average 10 times longer than when using divided differences for the representative set here; this may mean that more sophisticated search methods for identifying seed points, such as the method of searching on a fixed line mentioned earlier, would be more feasible in DA-LCS. Both improvements in strainline integration are due to the elimination of numerical noise introduced by divided differences, which is not present in DA.

5.2. Periodic Arnold–Beltrami–Childress flow

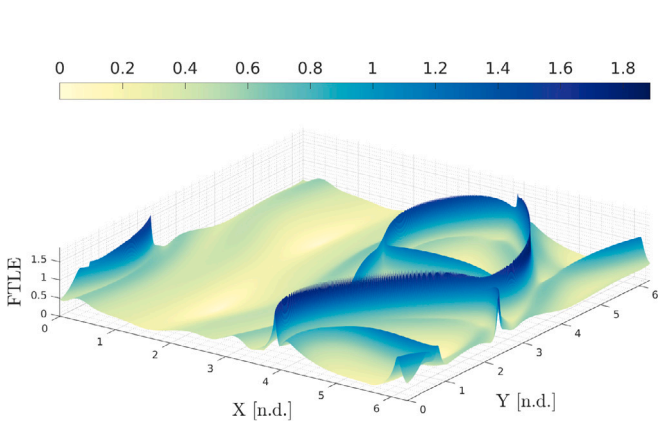
We now consider a time-periodic version of the Arnold–Beltrami–Childress flow with equations of motion

$$\dot{x} = (A + 0.1 \sin t) \sin z + C \cos y \tag{19}$$

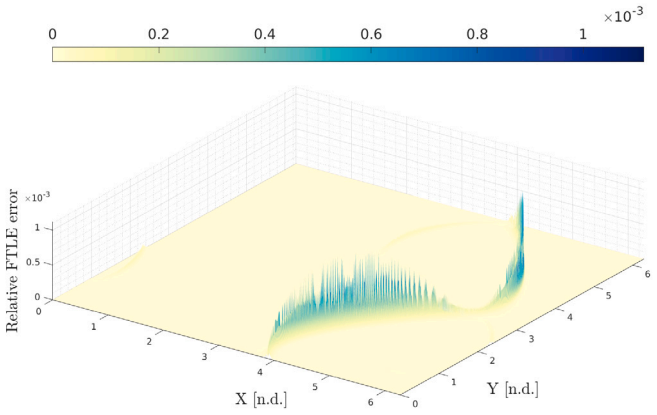
$$\dot{y} = B \sin x + (A + 0.1 \sin t) \cos z \tag{20}$$

$$\dot{z} = C \sin y + B \cos x. \tag{21}$$

The hyperplanes $\mathcal{S}$ and grids are the same as in the case of the steady ABC flow, but now with integration times $t_0 = 0$ and $T = 4$ to

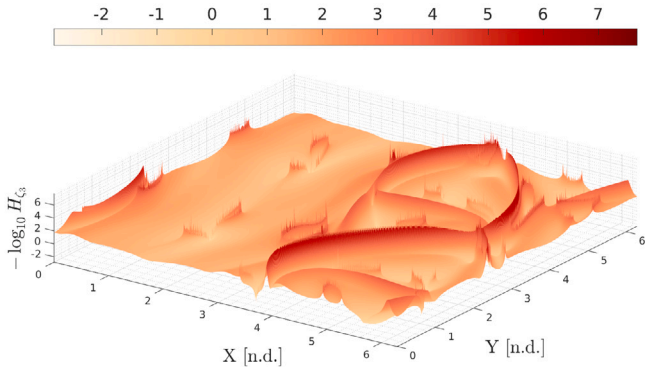


(a) Computed using DA-LCS.

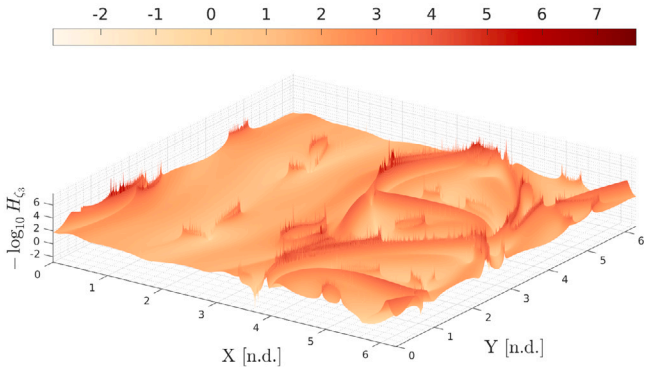


(b) Relative error between DA-LCS and the FTLE field obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in all directions.

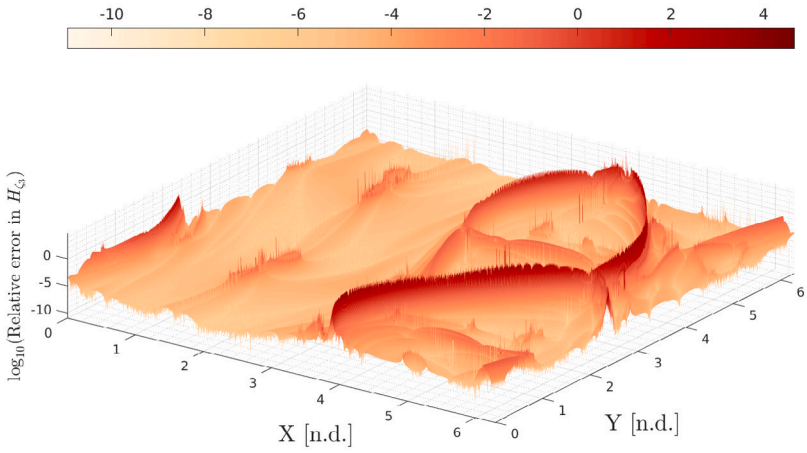
Fig. 7. Finite-time Lyapunov exponent fields for the periodic ABC flow from $t_0 = 0$ to $T = 4.0$, obtained using DA-LCS and divided differences. Again, the relative error between divided differences and DA-LCS is small, suggesting divided differences on the correct auxiliary grid in this case accurately approximates the FTLE field.



(a) $-\log H_{\zeta_3}$ computed using DA-LCS.



(b) $-\log H_{\zeta_3}$ obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in all directions. The same grid is used in computing both C_0^4 and $\nabla \times \zeta_3$.



(c) Relative error in the helicity field obtained through using DA-LCS compared to divided differences.

Fig. 8. Helicity fields for the periodic ABC flow computed using DA-LCS and divided differences from $t_0 = 0$ to $T = 4.0$. Here the DA-LCS helicity field already shows a qualitative difference in smoothness over divided differences. This is confirmed by the relative error which in this example exceeds 10^4 on the ridges.

again match the literature exactly. A helicity tolerance of $\alpha = 5 \times 10^{-5}$ is used, with a distance threshold $d_F = 0.04$.

Mirroring the analysis in the steady case, the FTLE fields for both DA-LCS and the relative error in the FTLE between DA-LCS and divided

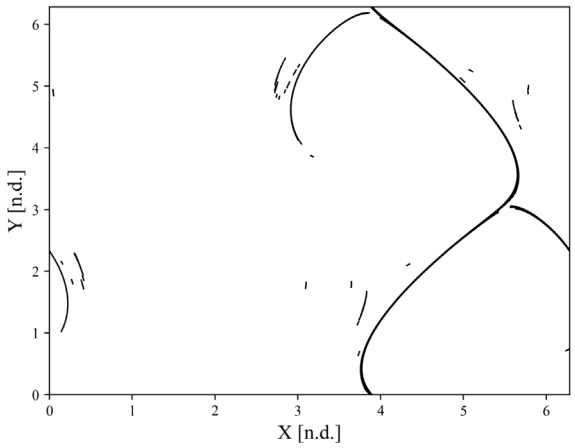


Fig. 9. Final strainlines for the periodic ABC flow on the $z = 0$ plane computed using DA-LCS, after filtering. The strainline structure is composed of approximately 50 strainline segments.

differences are shown in Figs. 7(a) and 7(b), respectively. Again, there is little qualitative and quantitative difference between the two fields. The differences in smoothness in the helicity fields are, however, more pronounced between Figs. 8(a) and 8(b), with the relative difference between them presented in Fig. 8(c). The main wishbone-like structure is particularly 'spiky' when using divided differences. With DA-LCS, there is a smooth, well-defined ridge of consistently low helicity for the algorithm to detect with much lower numerical noise; in fact, our helicity threshold is approximately two orders of magnitude lower than used in the literature but recovers qualitatively similar structures.

Finally, the strainlines on the $z = 0$ plane for this system computed using DA-LCS are shown in Fig. 9. Approximately 50 strainline segments determine the full strainline structure on the $z = 0$ plane for this example.

5.3. Chaotically-forced Arnold–Beltrami–Childress flow

Following [8], we now demonstrate that DA-LCS is robust under perturbations from a chaotic forcing function $g(t)$. The motion is forced by a chaotic Duffing oscillator, with equations of motion given by

$$\dot{x} = (A + 0.1\sin t)\sin z + C\cos y \tag{22}$$

$$\dot{y} = B\sin x + (A + 0.1g(t))\cos z \tag{23}$$

$$\dot{z} = C\sin y + B\cos x \tag{24}$$

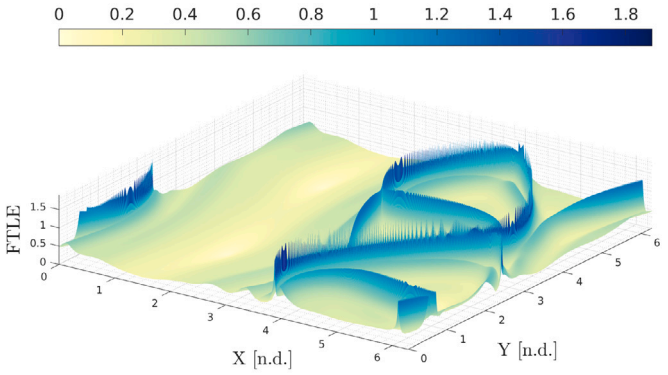
where $g(t)$ is the $x-$coordinate of the solution to the Duffing equation

$$\ddot{x} = -\delta\dot{x} - \beta x - \alpha x^3 + \gamma\cos(\omega t)\,. \tag{25}$$

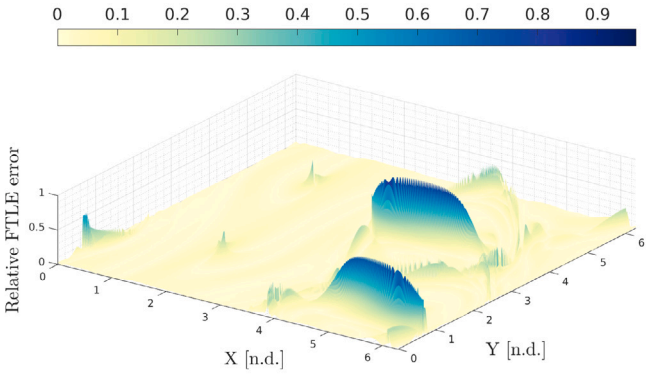
with parameters $\alpha = 1$, $\beta = -1$, $\gamma = 0.3$, $\delta = 0.2$, $\omega = 1$.

The computational grid is again the same as for the previous test cases involving the ABC flow, including the hyperplanes $\mathcal{S} = \{(x, y, z) \in [0, 2\pi]^3 \;:\; z = s_1\}, s_1 = 0.0, 0.005, 0.01, \ldots, 0.1$, but a longer integration time of $T = 5$ is used to match the literature. Again, a helicity tolerance of $\alpha = 5 \times 10^{-5}$ is used with a filtering distance of $d_F = 0.07$.

The FTLE fields computed using DA-LCS and the relative error compared to divided differences are again shown in Fig. 10(a) and Fig. 10(b), respectively. Some differences in the FTLE are beginning to emerge in portions of the main wishbone-like structure. The helicity fields for DA-LCS and divided differences are shown in Figs. 11(a) and 11(b), respectively, which now exhibits a significant difference compared to the two previous cases (Fig. 11(c)). Using DA-LCS, we are able to resolve a relatively smooth ridge of low helicity, whereas the use of divided differences leads to noticeable numerical noise throughout the field as well as an overall much higher helicity.



(a) Computed using DA-LCS.



(b) Relative error between DA-LCS and the FTLE field obtained using divided differences with auxiliary grid spacing of 0.1 of the nominal grid spacing in all directions.

Fig. 10. Finite-time Lyapunov exponent field for the chaotically-forced ABC flow and an integration time from $t_0 = 0$ to $T = 5.0$. Differences are beginning to become visible on the 'ends' of the main wishbone-like structure when using divided differences due to the spiky FTLE values when using divided differences.

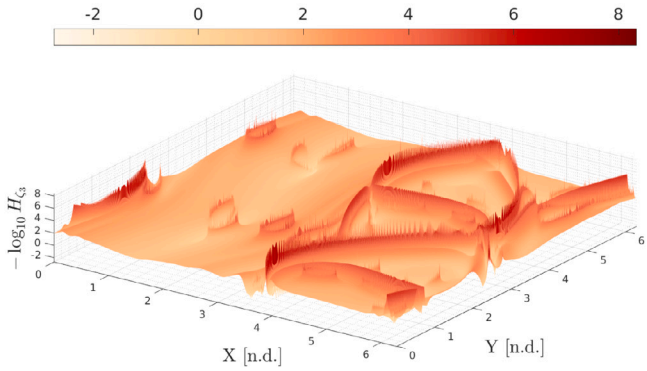
The strainlines for this system on the $z = 0$ plane computed using DA-LCS are presented in Fig. 12. A total of 57 strainline segments give the full structure on the $z = 0$ plane.

6. The elliptic-restricted three-body problem

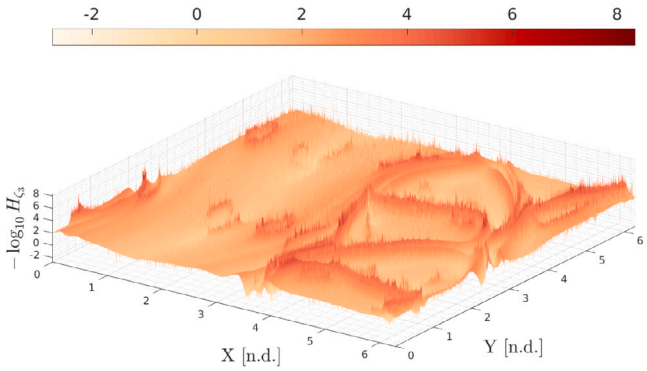
We now demonstrate the numerical out-performance of DA-LCS compared to standard approaches on a test problem from astrodynamics. The system presented in this Section is the Elliptic-Restricted Three-body Problem (ER3BP), which studies the motion of a small mass m_3 under the motion of two far larger masses m_1 and m_2 such that $m_1 \geqslant m_2 \gg m_3$. The system is parameterised by the mass parameter $\mu = m_2/(m_1 + m_2)$.

In an inertial coordinate system, m_2 and m_1 orbit their centre of mass on an ellipse of fixed eccentricity e_p, which is the second system parameter. The angle of m_2 with respect to the $+x$-axis of the inertial coordinate system is the true anomaly ν.

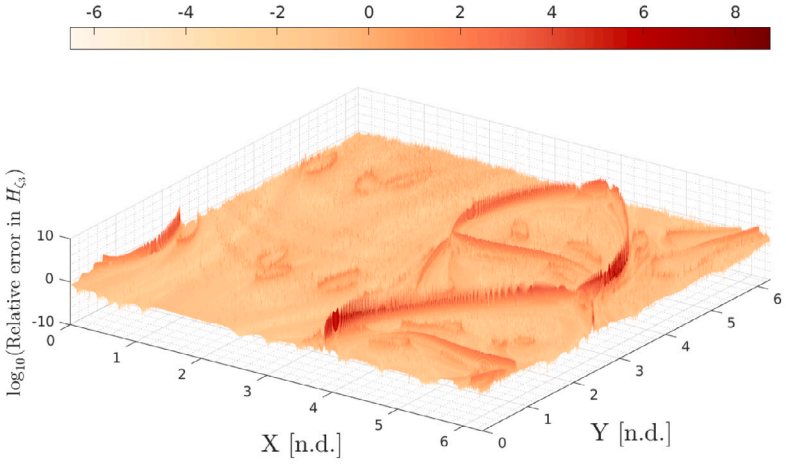
For the special case of $e_p = 0$, one recovers an autonomous dynamical system for which fixed points and invariant manifolds exist [12]; for the more general $e_p > 0$, such structures become difficult to determine. LCS have thus been suggested to analyse the behaviour for the cases of $e_p > 0$. In this example, we analyse the interesting dynamical phenomena around m_2. For small differences in initial position and velocity, orbits can vary from being bound entirely around m_2, being only temporarily captured around m_2, or escaping entirely [36]. Profiling these regions is of high importance in the design of space missions [37].



(a) $-\log H_{\zeta_3}$ computed using DA-LCS.



(b) $-\log H_{\zeta_3}$ obtained using divided differences with auxiliary grid spacing of 0.1 of the nominal grid spacing in all directions. The same grid is used for computing both C_0^5 and $\nabla \times \zeta_3$.



(c) Relative error in the helicity field obtained through using DA-LCS compared to divided differences.

Fig. 11. Helicity fields for the chaotically-forced ABC flow from $t_0 = 0$ to $T = 5.0$. DA-LCS produces visibly better-defined ridges, helping with robustly identifying seed points. However, the relative errors are very high due to the spiky nature of the ridges in both DA-LCS and divided differences.

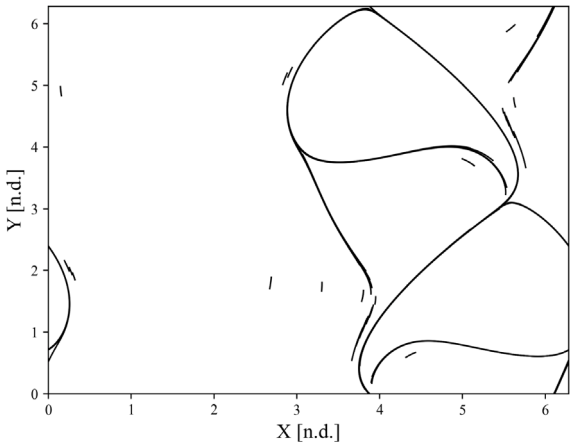


Fig. 12. Final strainline structure on the $z = 0$ plane for the chaotically-forced ABC flow computed using DA-LCS. The structure is formed of 57 individual strainline segments.

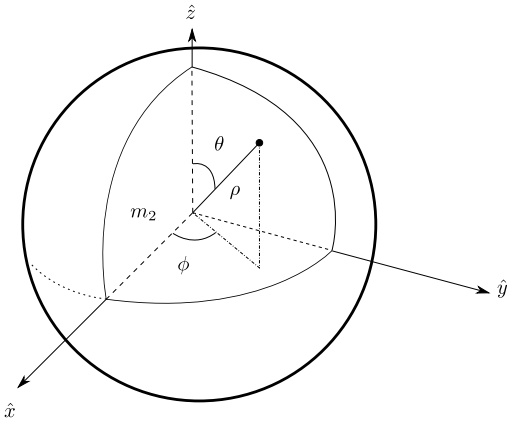


Fig. 13. The parameterisation of the space around m_2 using spherical coordinates relative to the inertial coordinate frame. By careful choice of the ranges of ρ, ϕ and θ, the reference hyperplanes can encapsulate regions of 'interesting' dynamics about m_2.

Since the ER3BP lives in a phase space defined in $\mathbb{R}^6$, but the algorithm above functions for a CGST that is 3×3 in dimension and represents a system with three-dimensional dynamics, we embed a three-dimensional submanifold in the six-dimensional phase space on which we compute the LCS. We parameterise the manifold in the three spatial directions to represent position around m_2 using spherical coordinates $\boldsymbol{\Psi} = (\rho, \phi, \theta)$ (Fig. 13). We complete the embedding by

uniquely associating a velocity $\boldsymbol{v}$ with each point in space to complete the full phase space.

Given the Cartesian position $\boldsymbol{x} = (x,\ y,\ z)^{\top}$ corresponding to $\boldsymbol{\Psi}$

$$x = \rho \cos\phi \sin\theta \tag{26}$$

$$y = \rho \sin\phi \sin\theta \tag{27}$$

$$z = \rho \cos\theta \tag{28}$$

the velocity at this point $\boldsymbol{v}(\boldsymbol{x})$ is chosen to be

$$\boldsymbol{v}(\boldsymbol{x}) = \sqrt{\mathrm{G}m_2 \frac{(1+e)}{\rho^3}} \left[\begin{pmatrix} x \\ y \\ z \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right], \tag{29}$$

where the problem parameters $\mathrm{G}m_2$ and e are the gravitational parameter of m_2 and an orbital eccentricity, respectively. Conceptually, this fixes the velocity direction tangential to a cylinder around the z-axis, while the magnitude corresponds to a Keplerian orbit of eccentricity e around m_2. Together, this choice of velocity vector reveals the 'dynamically interesting' behaviour introduced previously.

Rather than using the inertial coordinate system about m_2 to propagate the initial condition, it is beneficial to use a rotating-pulsating Cartesian coordinate system centred on the barycentre of m_1 and m_2. In this system, m_1 and m_2 are fixed, and the true anomaly ν replaces time as the independent variable. The transformation of the initial condition into this coordinate system is shown in Appendix B. In this system the equations of motion are given by

$$x'' = 2y' + \frac{\partial \Omega}{\partial x} \tag{30}$$

$$y'' = -2x' + \frac{\partial \Omega}{\partial y} \tag{31}$$

$$z'' = \frac{\partial \Omega}{\partial z} \tag{32}$$

where

$$\Omega = \frac{1}{1 + e_p \cos\nu} \left[\frac{1}{2} \left(x^2 + y^2 - z^2 e_p \cos\nu \right) + \frac{\mu}{r_1} + \frac{1-\mu}{r_2} \right] \tag{33}$$

and

$$r_1 = \sqrt{(x-\mu)^2 + y^2 + z^2} \tag{34}$$

$$r_2 = \sqrt{(x+1-\mu)^2 + y^2 + z^2}. \tag{35}$$

After propagation under the equations of motion, the transformation into the rotating coordinate system is inverted, and the final position is projected back into spherical coordinates. Another advantage of DA-LCS is that, provided the intermediate transformations are coded as DA operations, the derivatives of this process are computed fully automatically and there is no need to derive further equations for the coordinate transformations.

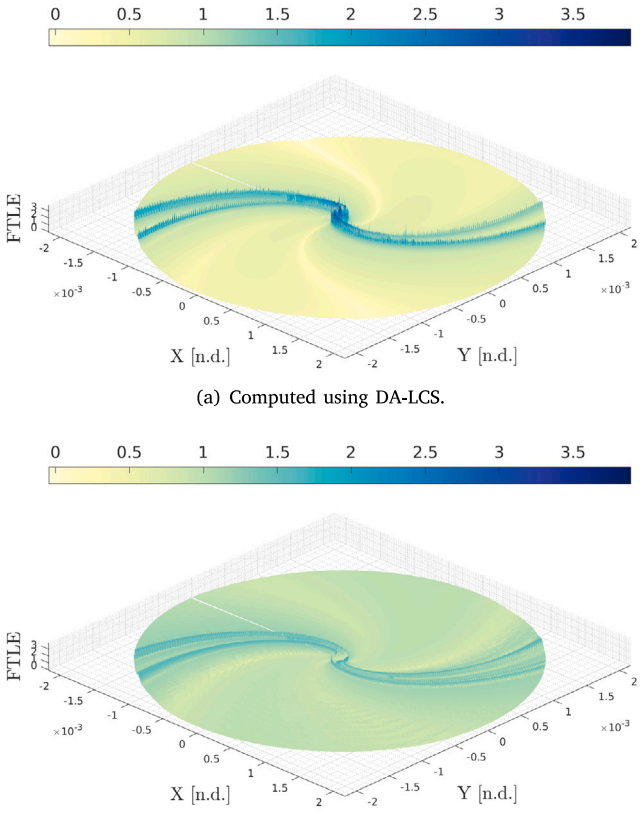
For this example, we choose m_1 to be the Sun and m_2 to be Mars, with the system parameters as given in Table 2. The set of reference hyperplanes is defined as

$$\mathcal{S} = \left\{ \boldsymbol{\Psi} \in \left[r, r_s\right] \times [0, 2\pi] \times \left[5^\circ,\ 15^\circ, \ldots, 175^\circ\right] \right\}.$$

The variables r and r_s here are the radius and the Hill sphere of Mars, respectively; the latter is the maximum distance from Mars at which it still dominates gravitational attraction. Together, the reference planes enclose the 'dynamically interesting' region around m_2. The initial integration time is set equal to $t_0 = \nu_0 = 0$ and the final time is $T = \nu = 2\pi$. The helicity tolerance α used is 4×10^{-6}.

6.1. Results

The FTLE fields computed using DA-LCS and divided differences on the $\theta = 115^\circ$ plane are presented in Figs. 14(a) and 14(b), respectively. The structure found using DA-LCS agrees with what would be expected from previous literature, with the structures in the two 'arms' being



(a) Computed using DA-LCS.

(b) FTLE field obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in r and ϕ and the nominal grid spacing in θ.

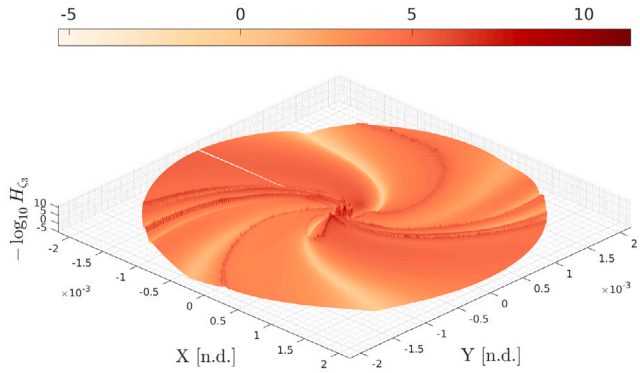
Fig. 14. Finite-time Lyapunov exponent field for the Elliptic-Restricted Three-body Problem on the $\theta = 115^\circ$ plane from $t_0 = \nu_0 = 0$ to $T = \nu = 2\pi$. While the structure is qualitatively similar, the ridges in the FTLE field are much more well-defined with DA-LCS.

Table 2
Parameter values used in the ER3BP investigation where m_1 is arbitrarily chosen to be the Sun and m_2 arbitrarily chosen to be Mars. All values are given in non-dimensional units and valid at $\nu = 2n\pi$, $n \in \mathbb{Z}$.

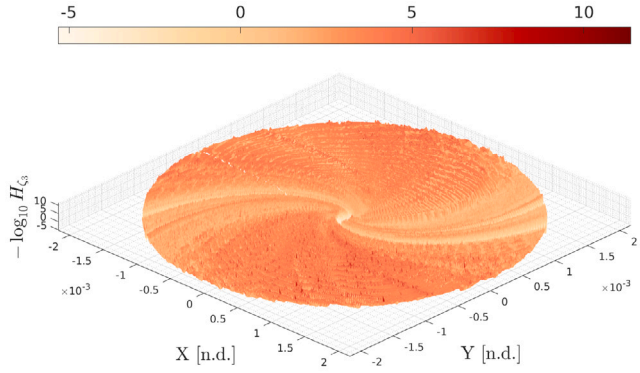
Parameter	Description	Value
e_p	Eccentricity of the orbit of m_2 about m_1	0.0935
μ	Mass parameter	3.227154×10^{-7}
e	Eccentricity of the orbit of m_3 about m_2	0.9
$\mathrm{G}m_2$	Standard gravitational parameter of m_2	1.50499×10^{-14}
r	Planetary radius of m_2	1.641×10^{-5}
r_s	Hill sphere of m_2	0.00513

consistent with the transition between orbits that escape and are permanently or temporarily captured about m_2 [36]. Similar performance, albeit with poorer definition of the FTLE ridges, can be obtained using divided differences after tuning the grid sizes used to generate the derivatives. We note that the ER3BP does admit variational equations that can be integrated with the equations of motion which may improve the quality of the derivatives used to compute $C^T_{t_0}$.

Importantly, these variational equations cannot be used to compute $\nabla \times \boldsymbol{\zeta}_3$, which must still be approximated using divided differences and appear to produce the majority of the error for this test case. This is to be expected, as the estimation of second derivatives using divided differences is numerically difficult. Fig. 15(a) presents the helicity field on the $\theta = 115^\circ$ plane for the ER3BP computed using DA-LCS, which like the FTLE field highlights the 'arms' as being influential portions of flow. Qualitative inspection of the trajectories in this region reveals the low-helicity portions of the field to separate regions of different



(a) Computed using DA-LCS.



(b) Helicity field obtained using divided differences with auxiliary grid spacing of 0.05 of the nominal grid spacing in r and ϕ and the nominal grid spacing in θ. The same grid is used for computing both $C_0^{2\pi}$ and $\nabla \times \boldsymbol{\zeta}_3$.

Fig. 15. $-\log H_{\zeta_3}$ for the Elliptic-Restricted Three-body Problem on the $\theta = 115°$ plane from $t_0 = \nu_0 = 0$ to $T = \nu = 2\pi$. No defined regions of low helicity are visible when using divided differences, whereas with DA-LCS we can readily identify low helicity regions to identify seed points.

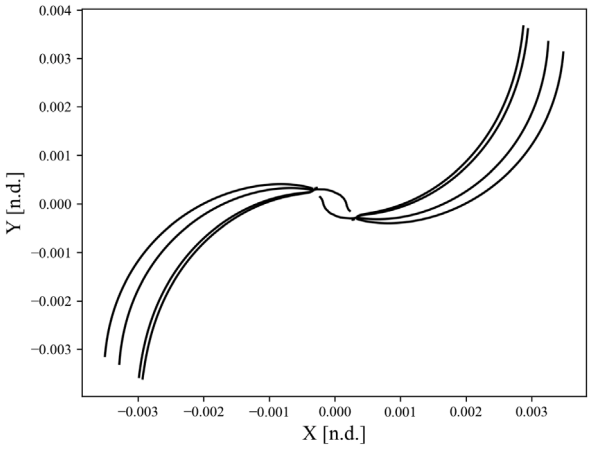


Fig. 16. Strainlines on the $\theta = 115°$ plane for the Elliptic-Restricted Three-Body Problem computed using DA-LCS. We are unable to generate any strainlines when using divided differences, but with DA-LCS we can deduce the structure of the LCS readily and with only 8 strainlines.

dynamical behaviour. However, using divided differences to compute the helicity, given in Fig. 15(b), produces no meaningful insight into the helicity field even after tuning the grid sizes used; the numerical noise in the determination of the helicity reveals no distinct ridges along which the numerical integration can begin, and the accuracy of the eigenvectors of $C_{t_0}^T$ when using divided differences yields strainlines that do not follow the expected structure in previous attempts at this topic [24,25], even after extensive tuning of the grid size used. This numerical improvement comes completely automatically, without the need to tune grid sizes and functions without any *a priori* knowledge.

The final strainlines for this flow computed using DA-LCS on the $\theta = 115°$ plane are shown in Fig. 16, and largely follow from the helicity field given earlier. We were not able to generate any meaningful strainlines using divided differences due to the poor numerical resolution of the eigenvectors and the related helicity field. A representative rendering of the full 3D LCS for this test case is shown in Fig. 17.

7. Conclusion

This paper has introduced DA-LCS, an improved numerical method for determining hyperbolic Lagrangian Coherent Structures in time-dependent dynamical systems. We showed how Differential Algebra can be used to directly construct high-order Taylor expansions of the flow, its derivatives and a field of leading eigenvectors of the flow's strain tensor, accurate to machine precision. We have shown that with this information we can construct a highly-accurate LCS based solely on the underlying dynamics of the system, even in highly complex flows. We demonstrated the effectiveness of the method through applications to common variations of the Arnold–Beltrami–Childress flow from the literature, as well as introducing a new and particularly challenging test problem from astrodynamics where the classical methods fail to produce usable results. DA-LCS also constructs the LCS automatically and without any *a priori* information, requiring no additional implementation beyond the dynamics of the system.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

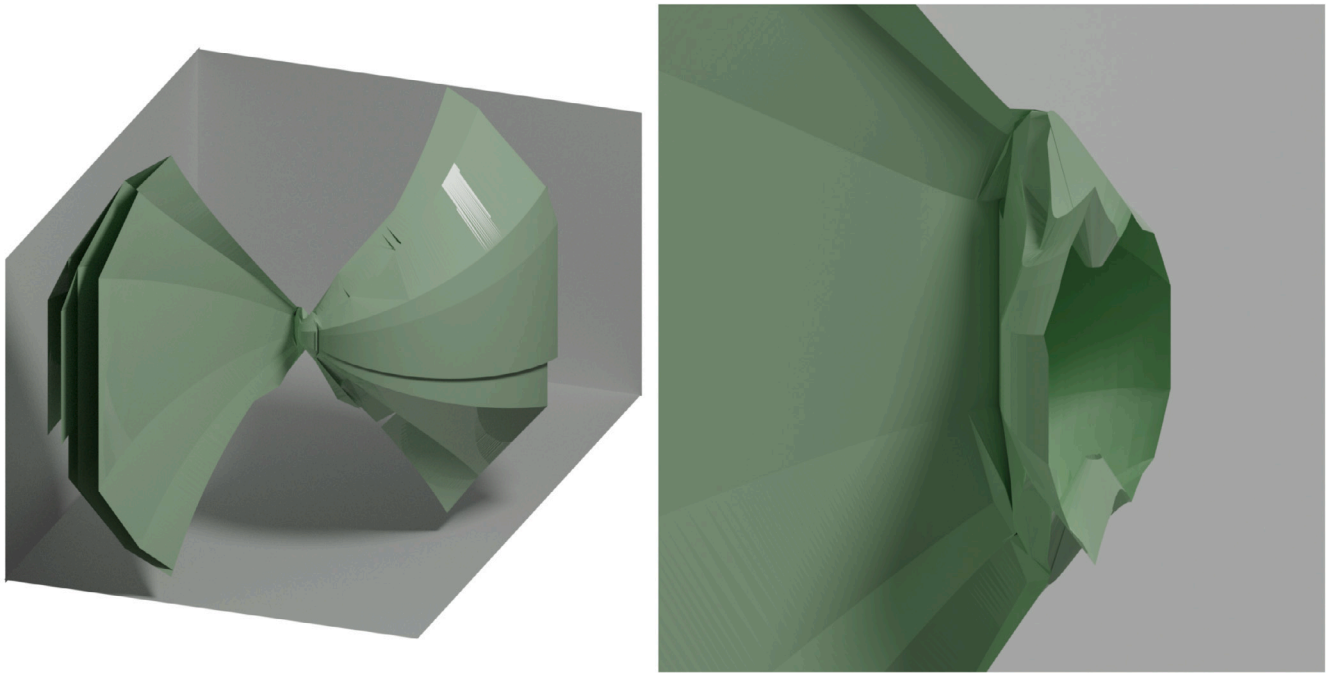
Data will be made available on request.

Acknowledgements

The authors acknowledge financial support from the EPSRC Centre for Doctoral Training in Next Generation Computational Modelling, United Kingdom grant EP/L015382/1, and the use of the IRIDIS High Performance Computing Facility and associated support services at the University of Southampton. The authors also thank Davide Lasagna for his helpful suggestions.

Appendix A. Pseudocode for the full DA-LCS algorithm

This Appendix introduces Algorithm 2, a detailed algorithmic pseudocode for the full DA-LCS numerical method to aid reproducibility. As in the main text, bold quantities refer to vectors and the notation $[\cdot]$ is used to denote a polynomial expansion using the quantity inside the square brackets as the reference point for the expansion.



(a) Full 3D structure of the LCS over the entire set of reference planes. (b) A zoomed-in section of the full LCS highlighting the interior structure.

Fig. 17. A set of representative renders of the 3D LCS for the ER3BP test case. The left figure is the full 3D LCS over all hyperplanes in S. On the right is a zoomed-in portion of the centre of the LCS, with the right half removed to highlight the internal structure.

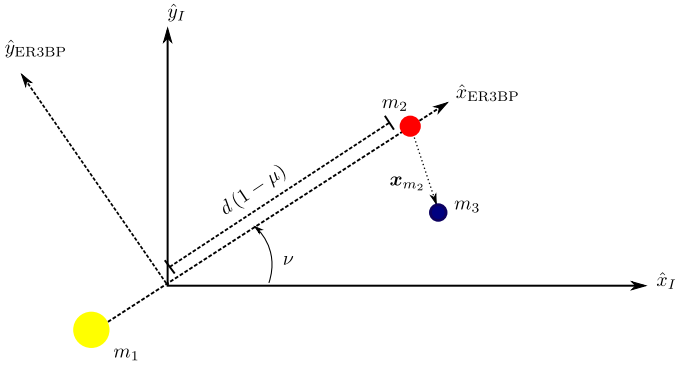


Fig. B.18. Schematic of the inertial frame (subscript $_I$) and the rotating-pulsating frame (subscript $_{\text{ER3BP}}$) for use in Appendix B. The transformation between the inertial and rotating-pulsating frame is a composite translation, rotation and normalisation.

Appendix B. Transformation into the rotating-pulsating frame of the elliptic-restricted three-body problem

As previously introduced, the Elliptic-Restricted Three-body Problem (ER3BP) models the motion of a small object m_3 under the influence of two far larger masses m_1 and m_2, such that $m_1 \geqslant m_2 \gg m_3$. The object m_3 is sufficiently small compared to m_1 and m_2 that it is considered massless. The system is parameterised by the mass parameter $\mu = m_2/(m_1 + m_2)$, and in an inertial coordinate system m_1 and m_2 orbit their centre of mass on an ellipse with fixed eccentricity e_p.

In Section 6, we chose the parameterisation of the sub-manifold to represent initial position around m_2 in the inertial frame using spherical coordinates, and the embedding to represent the initial velocity of the point in the inertial frame. This was done to simplify the problem set-up and more easily define the regions of 'interesting' dynamical behaviour. However, in the literature [38] the ER3BP is integrated in a rotating coordinate system where m_1 and m_2 are fixed on the x-axis at $(-\mu, 0, 0)$ and $(1-\mu, 0, 0)$, respectively, and the distance between them is normalised to unity. In this frame, the independent variable in the motion of m_3 is the true anomaly ν. To simplify the test case, the transformation that follows is valid only for values of ν that are scalar multiples of 2π; for an in-depth derivation of the general case of this transformation, the reader is directed to [38].

With reference to Fig. B.18, the transformation of the position from the m_2-centred inertial frame to the rotating-pulsating frame is formed of a translation to move the centre of the system to the centre of mass of m_1 and m_2, a rotation to align the $+x$ axis to the line joining m_1 and m_2, and a scaling to normalise the distance between m_1 and m_2 to unity.

We perform the translation first. Define the Cartesian position of m_3 about m_2 in the inertial frame as $\boldsymbol{x}_{m_2}$, such that the translated position around the barycentre (centre of mass) of m_1 and m_2, $\boldsymbol{x}_{\text{BC}}$, is

$$\boldsymbol{x}_{\text{BC}} = \boldsymbol{x}_{m_2} + d\,(1-\mu)\begin{pmatrix}\cos\nu\\ \sin\nu\\ 0\end{pmatrix} \tag{B.1}$$

where d is the full distance between m_1 and m_2, and $(1-\mu)$ gives the proportion of the distance d between m_2 and the centre of mass. The distance d can be retrieved from the orbit equation (more generally known as the ellipse equation)

$$d(\nu) = \frac{a\left(1-e_p^2\right)}{1+e_p\cos\nu} \tag{B.2}$$

with a the semi-major axis of m_2 about m_1. For the case of m_1 being the Sun and m_2 being Mars studied in this paper, at scalar multiples of 2π the semi-major axis $a = 1.10314$.

The coordinate axes must now be rotated such that m_1 and m_2 lie on the $+x$-axis. This is a clockwise rotation about $+z$ of an angle ν. We apply the standard Euler rotation matrix to $\boldsymbol{x}_{\text{BC}}$ to find its equivalent state in the rotated coordinate system $\boldsymbol{x}_{\text{rot}}$

$$\boldsymbol{x}_{\text{rot}} = R_z(\nu)\,\boldsymbol{x}_{\text{BC}} = \begin{pmatrix}\cos\nu & \sin\nu & 0\\ -\sin\nu & \cos\nu & 0\\ 0 & 0 & 1\end{pmatrix}\boldsymbol{x}_{\text{BC}}. \tag{B.3}$$

Algorithm 2 The DA-LCS numerical method

Input: S, α, δ, t_0, T, f

1: **for** hyperplanes S in S **do**
2: **for** points $\boldsymbol{x}_0$ on hyperplane S **do**
3: $\left[\boldsymbol{x}_0\right] \leftarrow \boldsymbol{x}_0 + \delta\boldsymbol{x}$ {Supplement initial condition with DA identity to at least 2nd order}
4: $[\boldsymbol{x}] \leftarrow$ Integration of $\left[\boldsymbol{x}_0\right]$ under f from t_0 to $t_0 + T$ {Flow expansion}
5: $\left[\nabla \boldsymbol{F}_{t_0}^T\right] \leftarrow \partial_j [\boldsymbol{x}]_i$ {Performed algebraically using DA}
6: $\left[C_{t_0}^T\right] \leftarrow \left(\left[\nabla \boldsymbol{F}_{t_0}^T\right]\right)^\top \left(\left[\nabla \boldsymbol{F}_{t_0}^T\right]\right)$
7: $C_{t_0}^T \leftarrow$ constant part of $\left[C_{t_0}^T\right]$
8: $\boldsymbol{\zeta}_3 \leftarrow$ dominant eigenvector of $C_{t_0}^T$ {Using standard floating-point techniques}
9: $\left[\boldsymbol{\zeta}_3\right] \leftarrow$ DA power law performed on $\left[C_{t_0}^T\right]$ with initial guess $\boldsymbol{\zeta}_3$
10: $\left[H_{\zeta_3}\right] \leftarrow \langle \nabla \times \left[\boldsymbol{\zeta}_3\right], \left[\boldsymbol{\zeta}_3\right] \rangle$
11: $H_{\zeta_3} \leftarrow$ constant part of $\left[H_{\zeta_3}\right]$
12: **if** $H_{\zeta_3} \leq \alpha$ **then**
13: Append $\boldsymbol{x}_0$ and H_{ζ_3} to $\mathcal{U}$
14: **end if**
15: **end for**
16: Create set of strainlines on hyperplane $\mathcal{L}$
17: **for** low-helicity points $\boldsymbol{x}_0$ and stored $H_{\zeta_{3,0}}$ in $\mathcal{U}$ **do**
18: $n \leftarrow 1$ {Number of steps}
19: $\sum H_{\zeta_3} \leftarrow H_{\zeta_{3,0}}$ {Running total}
20: $\bar{H}_{\zeta_3} \leftarrow \sum H_{\zeta_3}/n$ {Running average}
21: $\boldsymbol{s}_0 \leftarrow \boldsymbol{x}_0$
22: Add $\boldsymbol{s}_0$ to time-history of strainline L
23: **while** $\bar{H}_{\zeta_3} \leq \alpha$ **do**
24: $\boldsymbol{s}_n \leftarrow$ evaluation of strainline ODE with $\boldsymbol{s}_{n-1}$ {$\boldsymbol{\zeta}_3$ computed following Lines 3–8}
25: $n \leftarrow n + 1$
26: $\sum H_{\zeta_3} \leftarrow \sum H_{\zeta_3} + H_{\zeta_{3,n}}$ {$H_{\zeta_{3,n}}$ computed following Lines 3–11}
27: $\bar{H}_{\zeta_3} \leftarrow \sum H_{\zeta_3}/n$
28: Append $\boldsymbol{s}_n$ to L
29: **end while**
30: Append L to $\mathcal{L}$
31: **end for**
32: Filter $\mathcal{L}$ for duplicate strainlines using Hausdorff distance and threshold δ
33: **end for**
34: Interpolate between $\mathcal{L}_S$ on hyperplanes S in S to produce full LCS structure

Finally, the distance between m_1 and m_2 is normalised to 1 by scaling the length unit of the system by d. This yields the final ER3BP position $\boldsymbol{x}_{\text{ER3BP}}$

$$\boldsymbol{x}_{\text{ER3BP}} = \frac{\boldsymbol{x}_{\text{rot}}}{d}. \tag{B.4}$$

The composite transformation can be combined into a single equation for brevity:

$$\boldsymbol{x}_{\text{ER3BP}} = \frac{R_z(\nu)}{d(\nu)}\left(\boldsymbol{x}_{m_2} + d(1-\mu)\begin{pmatrix}\cos\nu\\ \sin\nu\\ 0\end{pmatrix}\right) \tag{B.5}$$

$$= \frac{R_z(\nu)}{d(\nu)}\boldsymbol{x}_{m_2} + (1-\mu)\begin{pmatrix}1\\ 0\\ 0\end{pmatrix}. \tag{B.6}$$

The equation above completes the transformation of the position from the inertial coordinate system around m_2 to the rotating coordinate system of the ER3BP. However, integrating the ER3BP equations of motion also requires the initial velocity of m_3 with respect to ν in the rotating coordinate system. Thus, the velocity in the inertial frame about m_2 with respect to time given by the embedding introduced in the main text, $\boldsymbol{v}$, must also be transformed into the ER3BP coordinate frame.

To do this, Eq. (B.6) is differentiated with respect to the true anomaly ν, which is the independent variable in the ER3BP. In the following, $\square'$ denotes derivatives with respect to ν (as in the ER3BP coordinate system), and $\dot{\square}$ denotes derivatives with respect to time (the inertial coordinate system.) Via the chain rule, the derivative of Eq. (B.6) is

$$\boldsymbol{x}'_{\text{ER3BP}} = \frac{R_z(\nu)'}{d(\nu)}\boldsymbol{x}_{m_2} + \frac{R_z(\nu)}{d(\nu)}\boldsymbol{x}'_{m_2} \tag{B.7}$$

since the quantity $(1/d(\nu))'$ is zero in the case of ν being a scalar multiple of 2π. The quantity $R_z(\nu)'$ is trivial to infer from its use previously

$$R'_z(\nu) = \begin{pmatrix}-\sin\nu & \cos\nu & 0\\ -\cos\nu & -\sin\nu & 0\\ 0 & 0 & 0\end{pmatrix}. \tag{B.8}$$

The velocity with respect to time in the inertial frame $\boldsymbol{v}$ represents $\dot{\boldsymbol{x}}_{m_2}$. To obtain $\boldsymbol{x}'_{m_2}$, we use

$$\frac{\mathrm{d}\boldsymbol{x}_{m_2}}{\mathrm{d}\nu} = \frac{\mathrm{d}\boldsymbol{x}_{m_2}}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\nu} = \boldsymbol{v}/\dot{\nu} \tag{B.9}$$

where $\dot{\nu}$ is given by considering the angular momentum of m_2 about m_1

$$\dot{\nu} = \frac{\mathrm{G}m_1^{\frac{1}{2}}\left(1+e_p\right)^2}{a^{\frac{3}{2}}\left(1-e_p^2\right)^{\frac{3}{2}}} \tag{B.10}$$

which completes the transformation of a position in the inertial frame about m_2 to the rotating coordinate system of the ER3BP for use in Section 6.

Since we are computing the LCS on a submanifold that represents the spatial dimensions about m_2, the inverse transformation need only consider the position. Eq. (B.6) is inverted to give $\boldsymbol{x}_{m_2}$ and then converted back into spherical coordinates for use in computing the LCS.

References

[1] J.D. Meiss, Symplectic maps, variational principles, and transport, Rev. Modern Phys. 64 (3) (1992) 795–848, http://dx.doi.org/10.1103/RevModPhys.64.795.

[2] G. Haller, Lagrangian coherent structures, Annu. Rev. Fluid Mech. (August 2014) (2015) 1–19, http://dx.doi.org/10.1063/1.2740025.

[3] F. Lekien, S.D. Ross, The computation of finite-time Lyapunov exponents on unstructured meshes and for non-Euclidean manifolds, Chaos 20 (1) (2010) 017505, http://dx.doi.org/10.1063/1.3278516, URL http://aip.scitation.org/doi/10.1063/1.3278516.

[4] G. Boffetta, G. Lacorata, G. Redaelli, A. Vulpiani, Detecting barriers to transport: A review of different techniques, Physica D 159 (1–2) (2001) 58–70, http://dx.doi.org/10.1016/S0167-2789(01)00330-X.

[5] A.M. Mancho, S. Wiggins, J. Curbelo, C. Mendoza, Lagrangian descriptors: A method for revealing phase space structures of general time dependent dynamical systems, Commun. Nonlinear Sci. Numer. Simul. 18 (12) (2013) 3530–3557, http://dx.doi.org/10.1016/j.cnsns.2013.05.002.

[6] G. Haller, G. Yuan, Lagrangian coherent structures and mixing in two-dimensional turbulence, Physica D 147 (3–4) (2000) 352–370, http://dx.doi.org/10.1016/S0167-2789(00)00142-1.

[7] A. Hadjighasem, M. Farazmand, D. Blazevski, G. Froyland, G. Haller, A Critical Comparison of Lagrangian Methods for Coherent Structure Detection, 2017, April.

[8] D. Blazevski, G. Haller, Hyperbolic and elliptic transport barriers in three-dimensional unsteady flows, Physica D 273–274 (2014) 46–62, http://dx.doi.org/10.1016/j.physd.2014.01.007.

[9] M. Farazmand, D. Blazevski, G. Haller, Shearless transport barriers in unsteady two-dimensional flows and maps, Physica D 278–279 (2014) 44–57, http://dx.doi.org/10.1016/j.physd.2014.03.008.
[10] C.R. Short, D. Blazevski, K.C. Howell, G. Haller, Stretching in phase space and applications in general nonautonomous multi-body problems, Celestial Mech. Dynam. Astronom. 122 (3) (2015) 213–238, http://dx.doi.org/10.1007/s10569-015-9617-4.
[11] Q. Qingyu, L. Mingpei, X. Ming, Lagrangian Coherent Structures in the Planar Parabolic/Hyperbolic Restricted Three-Body Problem, Mon. Not. R. Astron. Soc. (2020) http://dx.doi.org/10.1093/mnras/staa199.
[12] W.S. Koon, M.W. Lo, J.E. Marsden, S.D. Ross, Dynamical Systems, the Three-Body Problem and Space Mission Design, Marsden Books, 2008, URL http://www.gg.caltech.edu/~mwl/publications/papers/dynamicalThreeBody.pdf.
[13] S. Leung Shingyu, An Eulerian approach for computing the finite time Lyapunov exponent, J. Comput. Phys. 230 (9) (2011) 3500–3524, http://dx.doi.org/10.1016/j.jcp.2011.01.046, URL https://www.sciencedirect.com/science/article/pii/S0021999111000799.
[14] M. Berz, The method of power series tracking for the mathematical description of beam dynamics, Nucl. Instrum. Methods Phys. Res. A 258 (3) (1987) 431–436, http://dx.doi.org/10.1016/0168-9002(87)90927-2, URL https://www.sciencedirect.com/science/article/pii/0168900287909272.
[15] K. Makino, M. Berz, Remainder Differential Algebras and their Applications, in: Computational Differentiation: Techniques, Applications, and Tools, 1996, pp. 63–74.
[16] K. Makino, Rigorous Analysis of Nonlinear Motion in Particle Accelerators (Ph.D. thesis), (2) Michigan State University, 1998, URL http://www.bt.pa.msu.edu/pub/papers/makinophd/makinophd.ps.
[17] G. Di Mauro, M. Schlotterer, S. Theil, M. Lavagna, Nonlinear Control for Proximity Operations Based on Differential Algebra, J. Guid. Control Dyn. 38 (11) (2015) 2173–2187, http://dx.doi.org/10.2514/1.g000842.
[18] A. Wittig, P. Di Lizia, R. Armellin, F.B. Zazzera, K. Makino, M. Berzş, An automatic domain splitting technique to propagate uncertainties in highly nonlinear orbital dynamics, Adv. Astronaut. Sci. 152 (2014) 1923–1941.
[19] A. Wittig, P. Di Lizia, R. Armellin, K. Makino, F. Bernelli-Zazzera, M. Berz, Propagation of large uncertainty sets in orbital dynamics by automatic domain splitting, Celestial Mech. Dynam. Astronom. 122 (3) (2015) 239–261, http://dx.doi.org/10.1007/s10569-015-9618-3, URL http://link.springer.com/10.1007/s10569-015-9618-3.
[20] M. Massari, P. Di Lizia, M. Rasotto, Nonlinear Uncertainty Propagation in Astrodynamics Using Differential Algebra and Graphics Processing Units, J. Aerosp. Inf. Syst. 14 (9) (2017) 493–503, http://dx.doi.org/10.2514/1.i010535.
[21] R. Armellin, P. Di Lizia, F. Bernelli-Zazzera, M. Berz, Asteroid close encounters characterization using differential algebra: The case of Apophis, Celestial Mech. Dynam. Astronom. 107 (4) (2010) 451–470, http://dx.doi.org/10.1007/s10569-010-9283-5.
[22] M. Massari, P. Di Lizia, F. Cavenago, A. Wittig, Differential Algebra Software Library with Automatic Code Generation for Space Embedded Applications, no. January, 2018, http://dx.doi.org/10.2514/6.2018-0398.
[23] A. Wittig, Rigorous High-Precision Enclosures of Fixed Points and their Invariant Manifolds (Ph.D. thesis), Michigan State University, 2012, p. 158.
[24] X. Ros Roca, Computation of Lagrangian Coherent Structures with Application to Weak Stability Boundaries (M.Sc. thesis), 2015.
[25] A.S. Parkash, Application of Lagrangian Coherent Structures to the Computation and Understanding of Ballistic Capture Trajectories (M.Sc. thesis), 2019.
[26] G. Haller, A variational theory of hyperbolic Lagrangian Coherent Structures, Physica D 240 (7) (2011) 574–598, http://dx.doi.org/10.1016/j.physd.2010.11.010, URL https://www.sciencedirect.com/science/article/pii/S0167278910003143.
[27] M. Berz, Modern Map Methods in Particle Beam Physics, in: Advances in Imaging and Electron Physics, vol. 108, 1999, pp. 1–318, URL http://bt.pa.msu.edu/cgi-bin/display.pl?name=AIEP108book.
[28] F. Cavenago, P. Di Lizia, M. Massari, A. Wittig, On-board DA-based state estimation algorithm for spacecraft relative navigation, in: 7th European Conference for Aeronautics and Space Sciences, EUCASS, 2017, pp. 1–14, http://dx.doi.org/10.13009/EUCASS2017-607, URL https://www.eucass.eu/doi/EUCASS2017-607.pdf.
[29] A. Wittig, C. Colombo, R. Armellin, Long-term density evolution through semi-analytical and differential algebra techniques, Celestial Mech. Dynam. Astronom. 128 (4) (2017) 435–452.
[30] E.V. Haynsworth, A.S. Householder, The Theory of Matrices in Numerical Analysis, Amer. Math. Monthly 73 (10) (1966) http://dx.doi.org/10.2307/2314680.
[31] R.H. Chan, Y. Qiu, G. Yin, Iterative Methods for Eigenvalues/Eigenvectors, in: Encyclopedia of Social Network Analysis and Mining, 2018, http://dx.doi.org/10.1007/978-1-4939-7131-2_148.
[32] M. Farazmand, G. Haller, Computing Lagrangian coherent structures from their variational theory, Chaos 22 (1) (2012) http://dx.doi.org/10.1063/1.3690153.
[33] T. Devogele, M. Esnault, L. Etienne, F. Lardy, Optimized Discrete Fréchet Distance between trajectories, in: BigSpatial 2017 - Proceedings of the 6th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, no. November, 2017, pp. 11–19, http://dx.doi.org/10.1145/3150919.3150924.
[34] A. Driemel, A. Krivosija, C. Sohler, Clustering time series under the Fréchet distance, in: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms, Vol. 2, 2016, pp. 766–785, http://dx.doi.org/10.1137/1.9781611974331.ch55.
[35] K.L. Palmerius, M. Cooper, A. Ynnerman, Flow field visualization using vector field perpendicular surfaces, in: Proceedings - SCCG 2009: 25th Spring Conference on Computer Graphics, 2009, pp. 27–34, http://dx.doi.org/10.1145/1980462.1980471.
[36] Z.F. Luo, F. Topputo, Analysis of ballistic capture in Sun-planet models, Adv. Space Res. (2015) http://dx.doi.org/10.1016/j.asr.2015.05.042.
[37] E. Belbruno, Lunar capture orbits, a method of constructing earth moon trajectories and the lunar CAS mission, in: 19th International Electric Propulsion Conference, 1987, 1987, http://dx.doi.org/10.2514/6.1987-1054.
[38] V. Szebehely, W. Jefferys, Theory of Orbits: The Restricted Problem of Three Bodies, 36, (4) 1968, p. 375, http://dx.doi.org/10.1119/1.1974535,

Jack Tyler is a final-year Ph.D. Candidate in the Astronautics Research Group at the University of Southampton, funded by the EPSRC Centre for Doctoral Training in Next-Generation Computational Modelling. He obtained a Bachelor's of Engineering in Aeronautics and Astronautics in 2018, also from Southampton. His research interests encompass the application of high-performance computing and modern numerical methods to dynamical systems, with a particular focus on spaceflight problems.

Dr. Alexander Wittig is an Associate Professor in Astronautics at the University of Southampton. After receiving his dual Ph.D. in Mathematics and Physics from Michigan State University, he worked as an experienced researcher in the AstroNet-II Marie-Curie network at Politecnico di Milano, and as a research fellow in the Advanced Concepts Team at the European Space Agency.

His research focuses on the study of complex, nonlinear dynamical systems through the application of modern numerical methods, such as high-order differential algebra techniques and high performance computing, together with mathematical concepts from modern dynamical systems theory.