AMSC 663-664 Final Report: Lagrangian Analysis of Two- and Three-Dimensional Oceanic Flows from Eulerian Velocity Data

David Russell
Second-year Ph.D. student, Applied Math and Scientific Computing
russelld@umd.edu

Project Advisor: Kayo Ide
Department of Atmospheric and Oceanic Science
Center for Scientific Computation and Mathematical Modeling
Earth System Science Interdisciplinary Center
Institute for Physical Science and Technology
ide@umd.edu

May 13th, 2016

Abstract

In this project, we will design and build a set of Lagrangian analysis tools for an oceanic flow whose velocity is proscribed on a spatio-temporal grid. The main tools will be the so-called M-function (arc-length over a fixed time interval) and the maximal finite-time Lyapunov exponent, both of which help elucidate the underlying coherent structures of the flow. After validating them, we will test these tools on a dataset coming from a modeled flow of the Chesapeake Bay.

1 Introduction

Ocean currents have all sorts of large-scale coherent structures that are generally invisible to the naked eye. By coherent structure, we mean a "blob" of fluid that moves as one [10]. Finding ways to unveil these structures is of general interest to those who study mixing and transport; for example, the boundaries between these coherent structures may serve as rough barriers to pollutant transport.

Viewing the ocean as a dynamical system, we can gain access to some useful theoretical tools for our task, most notably the notions of stable and unstable manifolds. Suppose we are given a system of ordinary differential equations (or *flow*) $\dot{\mathbf{x}} = \mathbf{v}(\mathbf{x}, t)$, where $\mathbf{x} \in \mathbb{R}^N$ is position, t is time, $\mathbf{v} : \mathbb{R}^N \times \mathbb{R} \to \mathbb{R}^N$ is velocity, and $\dot{\mathbf{x}}$ denotes $\frac{d\mathbf{x}}{dt}$. Let $\mathbf{X}(\mathbf{X}_0, t)$ denote the position at time t of

the trajectory starting at $\mathbf{X}_0$ at time t_0, so that $\dot{\mathbf{X}}(\mathbf{X}_0, t) = \mathbf{v}(\mathbf{X}(\mathbf{X}_0, t))$ for all $t \in \mathbb{R}$. A *hyperbolic fixed point* $\mathbf{x}^*$ is a point such that $\mathbf{v}(\mathbf{x}^*, t) \equiv 0$ and the Jacobian operator $\frac{d\mathbf{v}}{d\mathbf{x}}(\mathbf{x}^*, t)$ has eigenvalues with both positive and negative real parts[1]. (In two dimensions, this means that locally, trajectories near $\mathbf{x}^*$ look like hyperbolas.) The *stable manifold* of $\mathbf{x}^*$ is

$$\left\{\mathbf{X}_0 \in \mathbb{R}^N : \lim_{t\to\infty} \mathbf{X}(\mathbf{X}_0, t) = \mathbf{x}^*\right\}$$

and the *unstable manifold* of $\mathbf{x}^*$ is

$$\left\{\mathbf{X}_0 \in \mathbb{R}^N : \lim_{t\to-\infty} \mathbf{X}(\mathbf{X}_0, t) = \mathbf{x}^*\right\}.$$

Thus, points on the stable manifold eventually approach the fixed point, while those on the unstable manifold have "originated from" the fixed point. These manifolds are important because they serve as boundaries between exactly the type of coherent structures we are interested in.

Although our discussion has applied to the case of a true hyperbolic fixed point, in ocean applications, nothing is truly fixed, and so a more appropriate notion is that of a *distinguished hyperbolic trajectory*, or *DHT*, which we will not define here [6].

Velocity data from ocean models give us the raw material to bring out these structures. To do so, it is useful to move from the grid-based (Eulerian) viewpoint to a flow-following (Lagrangian) one. That is, set up a vast network of fluid "particles" to be tracked through the flow, simulate their trajectories as the flow evolves, and then analyze those trajectories to get structural information. In Lagrangian terms, a coherent structure is nothing more than a group of particles whose trajectories "go together" in some sense.

2 Approach

In our effort to numerically elucidate the coherent structures and stable and unstable manifolds in an ocean flow, we will use two Lagrangian descriptor functions. The first is the so-called M-function, proposed by Mendoza and Mancho [8]:

$$M(\mathbf{X}_0, t_0, \tau) = \int_{t_0-\tau}^{t_0+\tau} \left|\dot{\mathbf{X}}(\mathbf{X}_0, t)\right| dt = \int_{t_0-\tau}^{t_0+\tau} \left(\sum_{i=1}^{2 \text{ or } 3} \left(\dot{X}_i(\mathbf{X}_0, t)\right)^2\right)^{\frac{1}{2}} dt,$$

which is simply the distance traveled by the particle with initial position $\mathbf{X}_0$ over the time interval spanning forward and backward time τ from time t_0. Sometimes we will be interested in looking only forward or backward in time, and will define M accordingly by

$$M(\mathbf{X}_0, t_0, \tau) = \int_{t_0}^{t_0+\tau} \left|\dot{\mathbf{X}}(\mathbf{X}_0, t)\right| dt$$

or

$$M(\mathbf{X}_0, t_0, \tau) = \int_{t_0-\tau}^{t_0} \left|\dot{\mathbf{X}}(\mathbf{X}_0, t)\right| dt$$

The choice should be clear from the context. The logic here is that a structural boundary will be evident where a region of fast-moving particles abuts a slow-moving region, whether this difference in speed is happening at the time t_0 or further along in our domain of integration.

The other, more traditional descriptor function we use is the maximal finite-time Lyapunov exponent of the flow, defined as

$$\mathrm{FTLE}(\mathbf{X}_0, t_0, \tau) = \frac{1}{\tau} \ln\left(\sigma_{\max}(L(\mathbf{X}_0, t_0, \tau)\right)$$

where

$$L(\mathbf{X}_0, t, \tau) = \frac{\partial \mathbf{X}(\mathbf{X}_0, t_0 + \tau)}{\partial \mathbf{X}_0}$$

is the so-called transition operator, representing a linearization of the flow about time t_0 and position $\mathbf{X}_0$, and $\sigma_{\max}$ denotes the largest singular value of L. We allow τ to be negative, which would give a "backward" FTLE. The idea here is that, for the locally linearized flow, its singular values (Lyapunov exponents) represent exponential growth/decay rates of an infinitesimal sphere in the directions of the corresponding singular vectors. Thus a large Lyapunov exponent indicates a strong stretching along some axis. We expect the strongest such stretching to occur along manifolds where the flow (eventually) bifurcates, as we expect along our stable (forward FTLE) and unstable (backward FTLE) manifolds.

Thus, given a velocity dataset, our general method is: lay down a fine grid of particles at time t_0, calculate their trajectories, use this information to obtain M and FTLE for each particle, and produce color plots of these M and FTLE fields at t_0, for τ long enough to bring out the manifolds. A previous result showing success with this method for M is shown in figure 1, in which the DHT and stable and unstable manifolds are clearly visible.

3 Algorithms

Since our velocity data will only be available on a spatio-temporal grid, we need to interpolate it to particle locations in time and space. Then we must integrate these velocities to obtain trajectories. Finally, we must numerically calculate our M and FTLE fields. Each of these tasks requires a choice of algorithms, and in some cases we chose more than one method for comparison, with an eye toward highlighting the speed vs. accuracy tradeoff often inherent in these judgments.

3.1 Interpolation

Suppose we are given velocity component u on a spatio-temporal grid, i.e. in three dimensions, $u_{ijkl} = u(x_i, y_j, z_k, t_l)$ for a suitable domain of i, j, k, and l. To

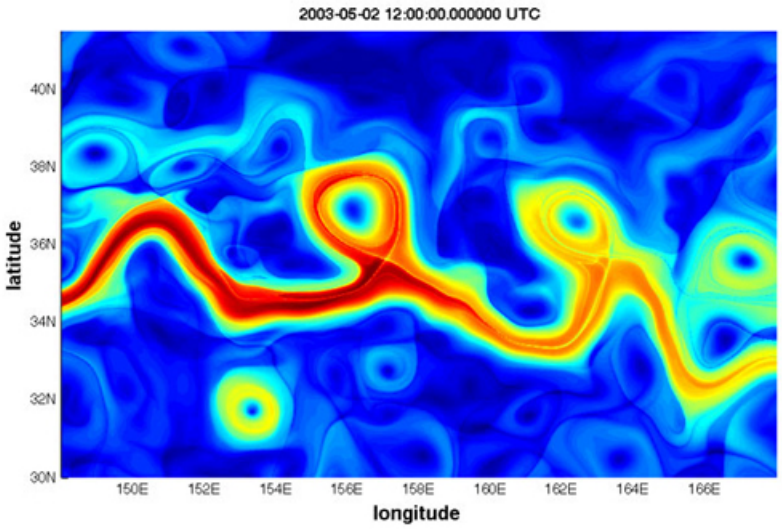


Figure 1: M-function coloring for a dataset from the Kuroshio current [8]. Red represents fast-moving regions, blue slow. Stable and unstable manifolds appear as thin yellow lines.

interpolate this velocity to an arbitrary point $\mathbf{x} = (x, y, z, t)$ in $\mathbb{R}^4$, we first locate the particular grid box $[x_i, x_{i+1}] \times [y_j, y_{j+1}] \times [z_k, z_{k+1}] \times [t_l, t_{l+1}]$ containing this point. Then we interpolate in phases: first horizontally (two dimensions), then vertically (one dimension), and finally temporally (one dimension). Thus we need algorithms for one- and two-dimensional interpolation.

For the one-dimensional interpolation problem (for z and t), we used Lagrange polynomials, of degrees 1 and 3 for comparison. That is, we interpolate a linear function through the two nearest neighboring points, or we interpolate a cubic polynomial through the four nearest neighbors (note that this is different from cubic splines, in which we only use two neighbors but match derivatives). Thus, for example, in the linear case we can approximate $u(x_i, y_j, z, t_l)$ by

$$f(z) = w \cdot u_{i,j,k,l} + (1 - w) \cdot u_{i,j,k+1,l}$$

where

$$w = \frac{z_{k+1} - z}{z_{k+1} - z_k}$$

represents a weight from 0 to 1 (for bottom to top). For the cubic case we use

$$\begin{aligned} f(z) = &\frac{(z-z_k)(z-z_{k+1})(z-z_{k+2})}{(z_{k-1}-z_k)(z_{k-1}-z_{k+1})(z_{k-1}-z_{k+2})} u_{i,j,k-1,l} \\ &+ \frac{(z-z_{k-1})(z-z_{k+1})(z-z_{k+2})}{(z_k-z_{k-1})(z_k-z_{k+1})(z_k-z_{k+2})} u_{i,j,k,l} \\ &+ \frac{(z-z_{k-1})(z-z_k)(z-z_{k+2})}{(z_{k+1}-z_{k-1})(z_{k+1}-z_k)(z_{k+1}-z_{k+2})} u_{i,j,k+1,l} \\ &+ \frac{(z-z_{k-1})(z-z_k)(z-z_{k+1})}{(z_{k+2}-z_{k-1})(z_{k+2}-z_k)(z_{k+2}-z_{k+1})} u_{i,j,k+2,l}. \end{aligned}$$

Approximation theory gives $O(h^2)$ accuracy for the linear case and $O(h^4)$ for the cubic case (REFERENCE?). As for smoothness, both methods are only C^0, unlike cubic splines which are C^1 by design.

For the two-dimensional interpolation problem, we again implement two different-order methods for comparison: bilinear splines and bicubic splines. A bilinear spline is a function of the form

$$f(x,y) = \sum_{m,n=0}^{1} c_{mn} x^m y^n = c_{00} + c_{10}x + c_{01}y + c_{11}xy$$

defined on a single two-dimensional grid box, where the parameters $c_{00}, c_{10}, c_{01}, c_{11}$ are chosen so that the values of f at the corners match those of the given function. In other words, given data u, the interpolant f on $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ must satisfy

$$\begin{aligned} f(x_i, y_j) &= u_{i,j} \\ f(x_{i+1}, y_j) &= u_{i+1,j} \\ f(x_i, y_{j+1}) &= u_{i,j+1} \\ f(x_{i+1}, y_{j+1}) &= u_{i+1,j+1}. \end{aligned}$$

A bicubic spline is a function of the form

$$f(x,y) = \sum_{m,n=0}^{3} c_{mn} x^m y^n$$

defined on a grid box. To solve for the sixteen unknowns c_{mn}, we must fit not only to the given values of u at the four corners, but also to estimated values of the partial derivatives $\partial_x u$, $\partial_y u$, and $\partial_{xy} u$ at those corners. By analogy with the one-dimensional case, we expect $O(h^2)$ accuracy for bilinear and $O(h^4)$ for bicubic. As for smoothness, bilinear is again C^0, while bicubic is at least C^1.

3.2 Integration

Assuming $\mathbf{v}$ is known (or interpolated) at all points of interest, we obtain a trajectory $\mathbf{X}(\mathbf{X}_0, t)$ by integrating the velocity:

$$\mathbf{X}(\mathbf{X}_0, t) = \int_{t_0}^{t} \mathbf{v}(\mathbf{X}(\mathbf{X}_0, t'), t')\, dt'.$$

For our numerical integration, we use the famous fourth-order Runge Kutta method, a classic high-order method that is nonetheless explicit and single-step, and therefore fast and easy to implement:

$$\begin{aligned}
\mathbf{k}_1 &= \Delta t \cdot \mathbf{v}\left(\mathbf{X}_n, t_n\right) \\
\mathbf{k}_2 &= \Delta t \cdot \mathbf{v}\left(\mathbf{X}_n + \frac{\mathbf{k}_1}{2}, t_n + \frac{\Delta t}{2}\right) \\
\mathbf{k}_3 &= \Delta t \cdot \mathbf{v}\left(\mathbf{X}_n + \frac{\mathbf{k}_2}{2}, t_n + \frac{\Delta t}{2}\right) \\
\mathbf{k}_4 &= \Delta t \cdot \mathbf{v}\left(\mathbf{X}_n + \mathbf{k}_3, t_n + \Delta t\right) \\
\mathbf{X}_{n+1} &= \mathbf{X}_n + \frac{1}{6}\left(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4\right).
\end{aligned}$$

3.3 Lagrangian Descriptor Functions

3.3.1 M Calculation

The M-function can be calculated in parallel with, and using the same integration schemes as, the trajectory computation.

3.3.2 FTLE Calculation

The two main aspects of the maximum Lyapunov exponent calculation are (i) approximating the transition matrix L and (ii) calculating the eigenvalues of $L^T L$. For the two-dimensional case, we have

$$L(\mathbf{X}(\mathbf{X}_0, t), t) = \begin{bmatrix} \frac{\partial X(\mathbf{X}_0, t)}{\partial X_0} & \frac{\partial X(\mathbf{X}_0, t)}{\partial Y_0} \\ \\ \frac{\partial Y(\mathbf{X}_0, t)}{\partial X_0} & \frac{\partial Y(\mathbf{X}_0, t)}{\partial Y_0} \end{bmatrix},$$

and the idea is to approximate these partial derivatives using finite differences. To that end, we initially lay four particles directly to the left and right of the initial position of interest (with sufficiently small separation ΔX_0) and up and down (separation ΔY_0), calculate their trajectories up to the current time, and approximate L by

$$\begin{bmatrix} \frac{X(X_0+\frac{\Delta X_0}{2}, Y_0, t) - X(X_0-\frac{\Delta X_0}{2}, Y_0, t)}{\Delta X_0} & \frac{X(X_0, Y_0+\frac{\Delta Y_0}{2}, t) - X(X_0, Y_0-\frac{\Delta Y_0}{2}, t)}{\Delta Y_0} \\ \\ \frac{Y(X_0+\frac{\Delta X_0}{2}, Y_0, t) - Y(X_0-\frac{\Delta X_0}{2}, Y_0, t)}{\Delta X_0} & \frac{Y(X_0, Y_0+\frac{\Delta Y_0}{2}, t) - Y(X_0, Y_0-\frac{\Delta Y_0}{2}, t)}{\Delta Y_0} \end{bmatrix}$$

(A higher-order method could presumably be used here as well, but this was not discussed.) The three-dimensional case is entirely analogous.

Calculating the eigenvalues of $L^T L$ is relatively simple, since it is either a 2×2 or 3×3 matrix, leading to a characteristic polynomial that is either quadratic or cubic. Solving the characteristic equation thus amounts to finding the roots of a quadratic or cubic polynomial. Both of these problems have tractable closed-form solutions (e.g. the quadratic formula in the 2×2 case), and this is the approach we use.

4 Implementation

All programs are written in MATLAB (version 2015b). The majority of runs were executed on a MacBook Pro laptop with a 2.6 GHz Intel Core i5 processor and 8 GB of memory. However, beyond $\tau = 6$ hours or so for the Chesapeake ROMS data, these runs were moved to the Deepthought2 computing cluster at the University of Maryland, mostly because the data storage capacities of the laptop were too limited.

Despite the essentially parallel nature of the algorithms (each particle trajectory depends only on the field, not on the other particles), no explicit parallelization was written into the code. However, all particle operations were completely vectorized, to make use of MATLAB's fast handling of vector operations. Thus, for example, at each time step, a single call to the interpolating function was made, feeding in the particle positions as column vectors (rather than a unique call for each particle).

5 Validation

To validate our interpolation, integration, and Lagrangian analysis tools, we applied them to some simple analytical functions to see if they produced the correct results.

5.1 Interpolation

To validate our two-dimensional interpolation routine, we applied it to the function $f(x, y) = e^y \sin(\pi x)$ on the domain $[-1, 1]^2$. We discretized this function on an Arakawa C-grid (using $\Delta x = \Delta y$), interpolated f as $\hat{f}$ on a finer grid ($\Delta x_{\text{fine}} = \frac{\Delta x}{6}$), took $\Delta x, \Delta y \to 0$ (maintaining $\Delta x_{\text{fine}} = \frac{\Delta x}{6}$ as the field was re-interpolated at each step), and verified that the error went to 0 at the expected rate. The appropriate error here is in the grid-function p-norm:

$$\|e\|_p = \|f - \hat{f}\|_p = \left(\sum_{i=1}^{M} \sum_{j=1}^{N} \|f(x_i, y_j) - \hat{f}(x_i, y_j)\|^p \cdot \Delta x_{\text{fine}} \Delta y_{\text{fine}} \right)^{\frac{1}{p}}$$

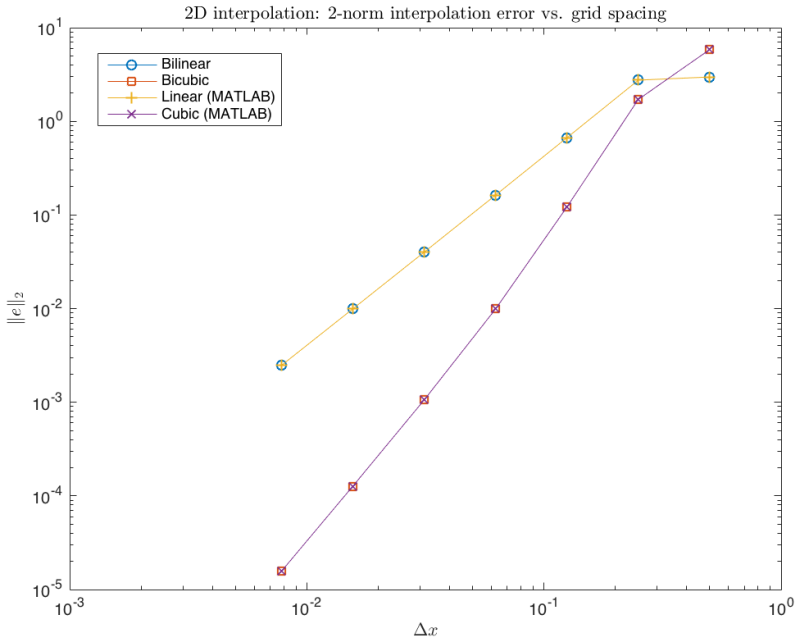


Figure 2: $\|\hat{f} - f\|_2$ vs. Δx for bilinear and bicubic interpolation, $f(x, y) = e^y \sin(\pi x)$.

where the sum is over an $M \times N$ fine grid. We tried $p = 1, 2, \infty$, but show here only the results of $p = 2$, though the others were similar. The errors are shown in figure 2) along with estimated convergence rates in table 1, using the two smallest values of Δx. The plots confirm that i) the interpolation error does indeed go to 0 with Δx, ii) our algorithms closely match those of MATLAB's `interp2` function, and iii) bilinear interpolation is second-order accurate, while bicubic is third-order.

Similar procedures were used to validate our three-dimensional interpolation methods for the function $u(x, y, z) = e^x \cos 2\pi y \cos 2\pi z$ on $[-1, 1]^3$. In this case, because the horizontal dimensions are decoupled from the vertical dimension, we validated those two cases separately. For the horizontal dimensions, we fixed Δz while $\Delta x = \Delta y \to 0$, comparing our interpolated function $\hat{f}$ to the horizontally-

	Convergence rate ($\|e\|_2 \sim h^p$)
Bilinear	$p \approx 2.0064$
Bicubic	$p \approx 3.0175$

Table 1: Convergence rates for bilinear and bicubic interpolation

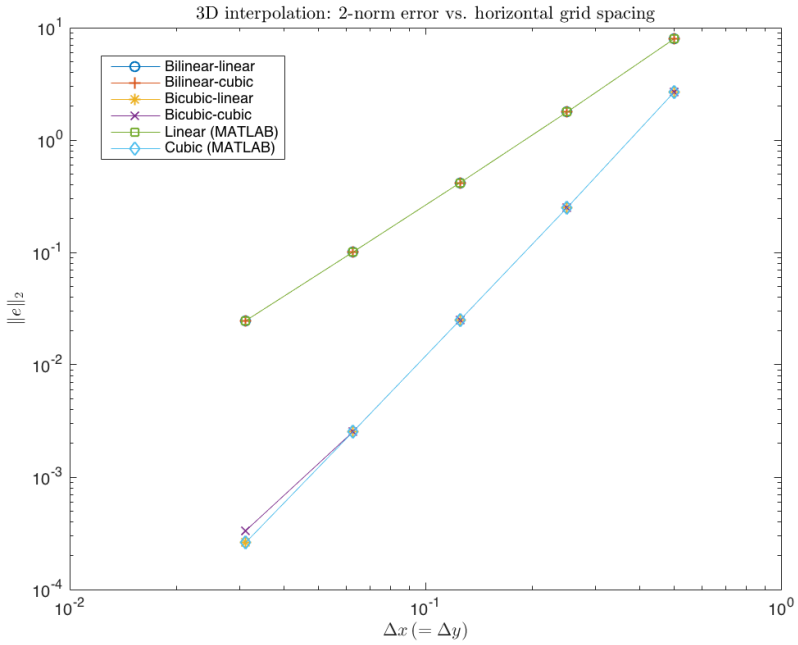


Figure 3: $\|\hat{f} - \tilde{f}\|_2$ vs. Δx for bilinear and bicubic interpolation, $f(x, y) = e^y \sin(\pi x)$.

true, vertically-interpolated function $\tilde{f}$ (using the same vertical interpolation method for $\hat{f}$ and $\tilde{f}$, so that $\hat{f}$ should approach $\tilde{f}$ as $\Delta x \to 0$). Results are shown in figure **??** and table 2. Here again we see second-order accuracy for bilinear horizontal interpolation (no matter the vertical method), and third-order for bicubic horizontal interpolation. The inclusion of MATLAB's linear and cubic algorithms for its `interp3` function is a bit messy: while our bilinear-linear interpolation method seems to be equivalent to MATLAB's three-dimensional linear interpolation, its three-dimensional *cubic* interpolation is not decoupled in the way that ours is, and so it is compared against the completely true solution here, which somewhat muddies the interpretation (IS $\Delta z \to 0$.

The vertical component of our three-dimensional interpolation was validated likewise, by fixing $\Delta x = \Delta y$ and taking $\Delta z \to 0$. The results (figure 4 and table 3) now show second-order convergence for linear vertical interpolation and fourth-order for cubic.

The timing of these routines, in two and three dimensions, is shown in figures 5 and 6. Unsurprisingly, cubic and bicubic take significantly longer than linear and bilinear. Interestingly, MATLAB significantly outperforms our implementations, in some cases by as much as a factor of 60! For this reason, we switched to MATLAB's innate interpolation for our more expensive runs, where

Horizontal(-vertical) method	Convergence rate ($\|e\|_2 \sim h^p$)
Bilinear(-linear)	$p \approx 2.0253$
Bicubic(-linear)	$p \approx 3.2675$
Bicubic(-cubic)	$p \approx 2.9192$
MATLAB Cubic	$p \approx 3.2675$

Table 2: Convergence rates for horizontally bilinear and bicubic interpolation.

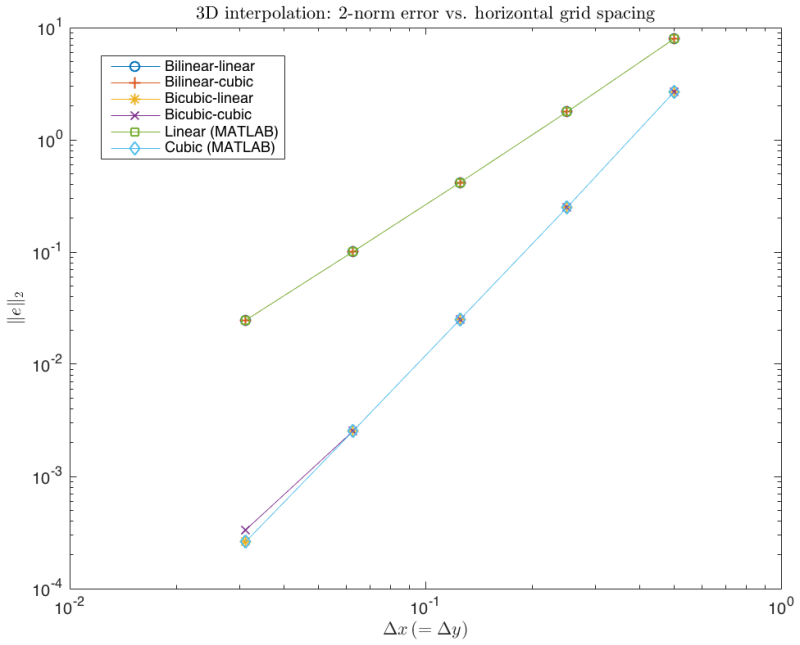


Figure 4: $\|\hat{f} - \tilde{f}\|_2$ vs. Δx for bilinear and bicubic interpolation, $f(x, y) = e^y \sin(\pi x)$.

(Horizontal-)vertical method	Convergence rate ($\|e\|_2 \sim h^p$)
(Bilinear-)linear	$p \approx 2.0101$
(Bicubic-)cubic	$p \approx 4.0484$
MATLAB cubic	$p \approx 3.2836$

Table 3: Convergence rates for vertically linear and cubic interpolation.

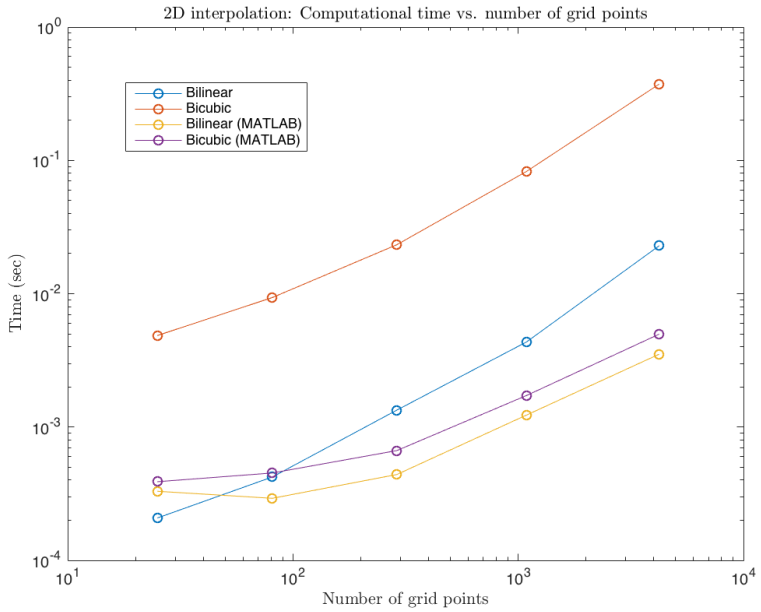


Figure 5: CPU time vs. Δx for two-dimensional interpolation methods

speed became essential.

5.2 Integration

To validate our integration methods and our analysis, we tested them on a few ODE systems with well-known dynamics. In two dimensions, our primary test system was the undamped Duffing oscillator, and in three dimensions, we used Hill's spherical vortex.

The Duffing oscillator [3] is an example of a simple nonlinear oscillator, wherein the restoring force has a cubic as well as a linear term. The specific version we will implement here has equations:

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= x - x^3 + \epsilon \sin t \end{aligned}$$

where ϵ is a forcing parameter. The unforced case ($\epsilon = 0$) is an integrable system with Hamiltonian $H(x, y) = \frac{1}{2}y^2 - \frac{1}{2}x^2 + \frac{1}{4}x^4$, so that H is conserved along trajectories. A phase portrait of several of these trajectories (level sets of H) is shown in figure 7. Trajectories can also be explicitly given as functions of

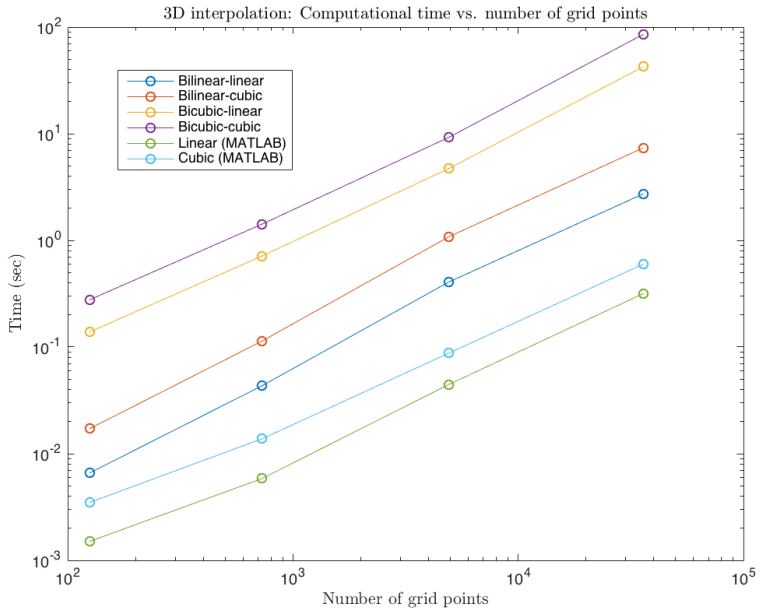


Figure 6: CPU time vs. Δx for three-dimensional interpolation methods

time [9] via

$$x(t) = x_0 \cdot \mathrm{cn}\left(t\sqrt{x_0^2 - 1}, \sqrt{\frac{x_0^2}{2(x_0^2 - 1)}}\right)$$

$$y(t) = -x_0\sqrt{x_0^2 - 1} \cdot \mathrm{sn}\left(t\sqrt{x_0^2 - 1}, \sqrt{\frac{x_0^2}{2(x_0^2 - 1)}}\right) \cdot \mathrm{dn}\left(t\sqrt{x_0^2 - 1}, \sqrt{\frac{x_0^2}{2(x_0^2 - 1)}}\right)$$

where cn, sn, and dn are the three types of Jacobi elliptic functions. These exact trajectories provide a benchmark for comparison of our numerically integrated trajectories. A plot of three such trajectories against their analytical counterparts is shown in figure 8. The trajectory beginning at $(-\sqrt{2}, 0)$ is notable because it starts on the stable/unstable manifold of the hyperbolic fixed point at the origin, and should thus remain there forever (which the numerical version evidently does not). Despite the clear drift from the true solution with time, we nonetheless expect the numerical trajectories to approach truth as the interpolation mesh size Δx goes to zero. In fact, this does happen for the sample trajectory $\mathbf{X}_0 = (-1.5, 0)$, as in figure 10, where we see the root-mean-square error decrease with Δx, for fixed Δt and t_f. We also observe that H becomes closer and closer to constant in time as $\Delta x \to 0$ (figure **??**).

To validate our three-dimensional trajectory integration, we use Hill's spherical vortex [5] as a model system. This three-dimensional axisymmetric flow field

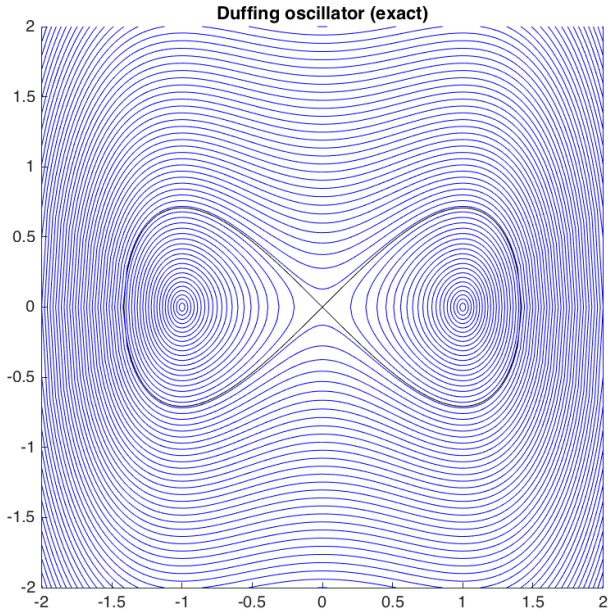


Figure 7: Exact trajectories for the autonomous Duffing oscillator (stable/unstable manifold in black)

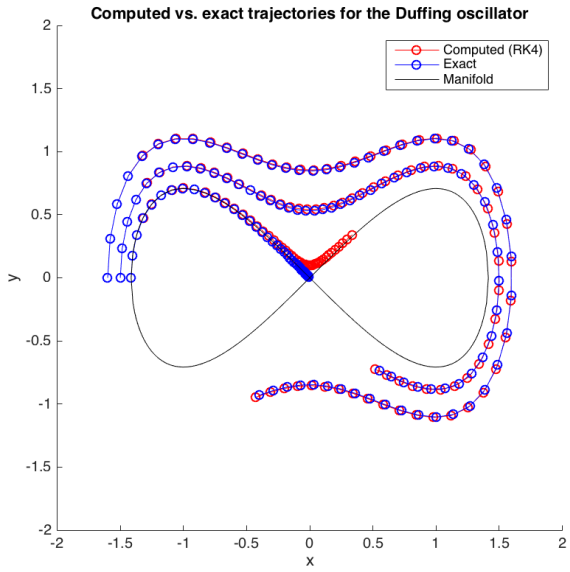


Figure 8: Computed vs. exact trajectories for the autonomous Duffing oscillator with $\mathbf{X}_0 = (-1.6, 0)$, $(-1.5, 0)$, and $(-\sqrt{2}, 0)$.

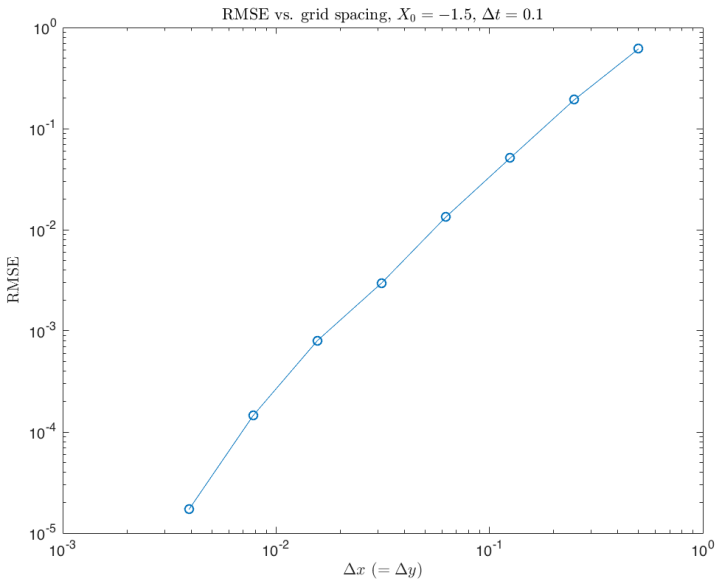


Figure 9: RMSE vs. Δx for the autonomous Duffing oscillator, trajectory $\mathbf{X}_0 = (-1.5, 0)$, $t_f = 6$, $\Delta t = 0.1$

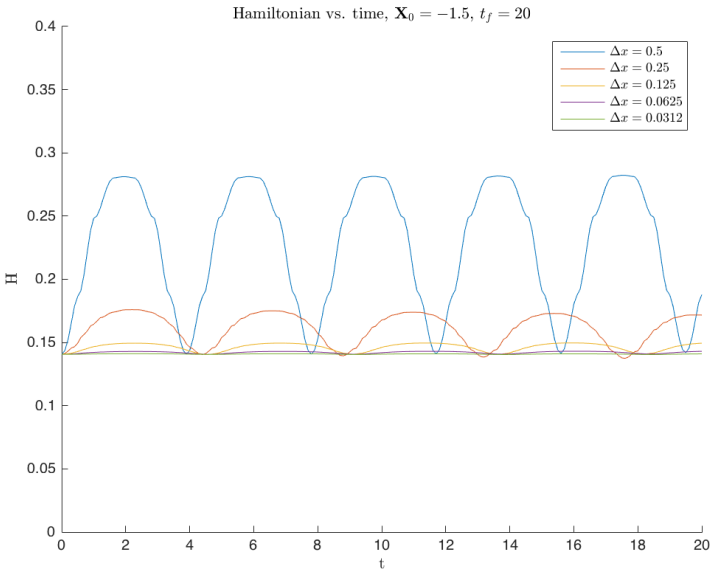


Figure 10: Hamiltonian vs. time for autonomous Duffing oscillator, $\mathbf{X}_0 = (-1.5, 0)$, $t_f = 20$, $\Delta t = 0.1$, for various values of Δx.

serves as a simple model for fluid flow in and around a sphere (figure 11). It has a streamfunction given by

$$\psi(r, \theta) = -\frac{3}{4} U r^2 \left(1 - \frac{r^2}{a^2}\right) \sin^2 \theta$$

where θ is the polar angle (measured from the positive z-axis), r is the distance to the origin, U is a velocity parameter, and a is the radius. From this streamfunction, radial and azimuthal velocities can be generated via

$$u_r = \frac{1}{r^2 \sin\theta} \frac{\partial \psi}{\partial \theta}$$
$$u_\theta = -\frac{1}{r \sin\theta} \frac{\partial \psi}{\partial r}.$$

which, after some manipulation, gives Cartesian velocities

$$\dot{x} = -\frac{3Uxz}{2a^2}$$
$$\dot{y} = -\frac{3Uyz}{2a^2}$$
$$\dot{z} = \frac{3U(2x^2 + 2y^2 + z^2 - a^2)}{2a^2}$$

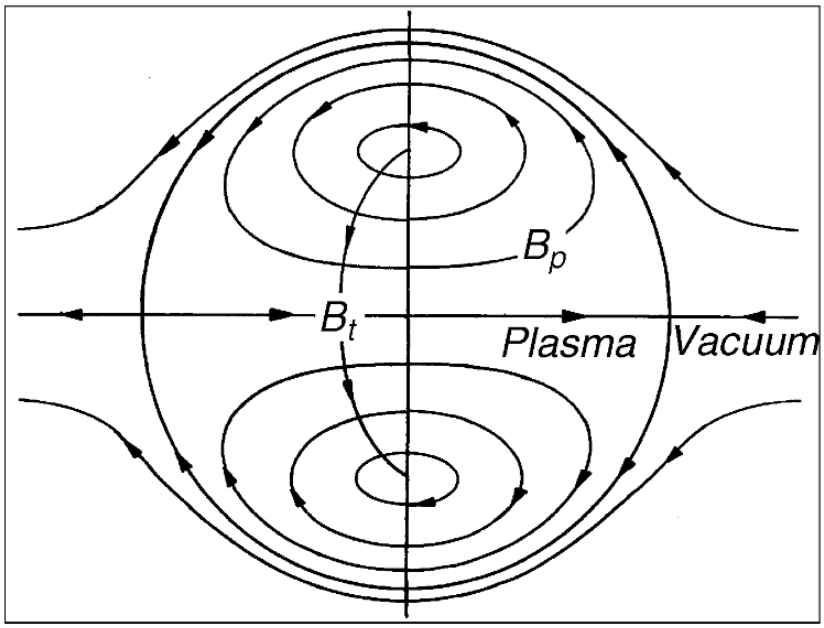


Figure 11: Hill's spherical vortex [11], z-axis pointing to the right. In our version there is no flow into or out of the page for this cross-section.

The streamfunction ψ now plays the role of the Hamiltonian, that is, it is conserved along trajectories. Thus, for $U = a = 1$, we looked at the Hamiltonian of the trajectory $\mathbf{X}_0 = (0.4, 0.3, 0)$ (bound inside the sphere) as $\Delta x = \Delta y = \Delta z \to 0$. Indeed, the Hamiltonian becomes constant in time in the limit (figure 12). Additional validation came from a visual check (not shown) that every trajectory was more or less confined to a plane containing the z-axis, as is required of an axisymmetric flow.

5.3 Lagrangian Descriptor Functions

Beyond the properties of their trajectories, the Duffing and Hill systems have well-known stable and unstable manifolds, which our M and FTLE Lagrangian descriptor functions should both be able to "find" and bring out. We can also validate these attempts against previous work in the same vein by Mancho et. al. [7].

As alluded to before, the autonomous Duffing oscillator has a single hyperbolic fixed point at the origin, and its stable and unstable manifolds are both given by the figure-eight shape $\frac{1}{2}y^2 - \frac{1}{2}x^2 + \frac{1}{4}x^4 = 0$. This shape be broken into two S-shapes, one of which spreads quickly away from the origin, and one quickly toward the origin. Thus, as τ increases, we should see the former structure first when plotting the M and FTLE forward in time, and the latter first when plotting backward. Figure 13 shows exactly that, with the low-M ridges

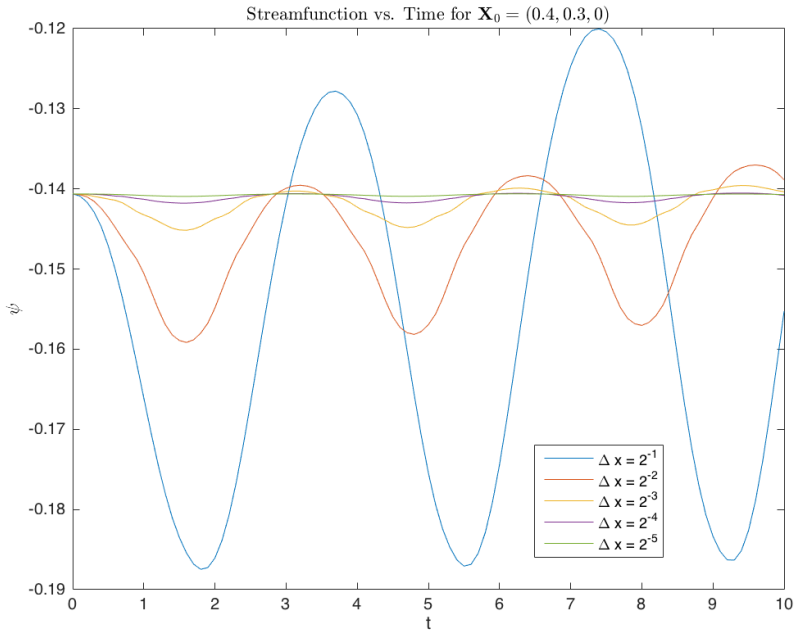
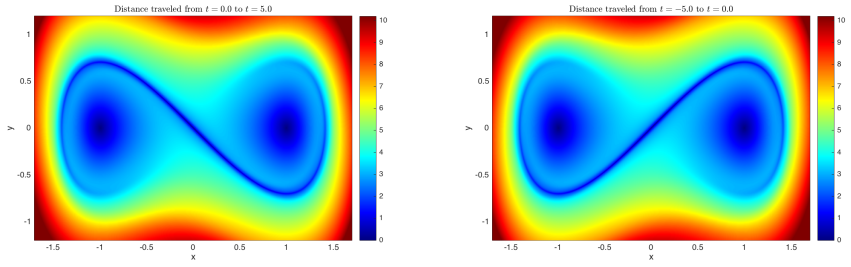
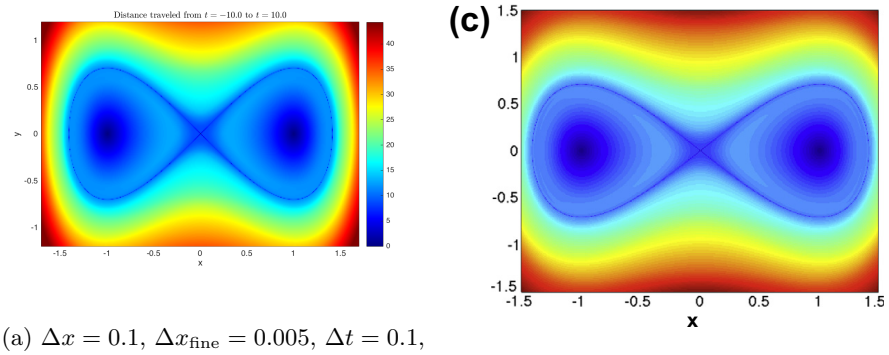


Figure 12: Hamiltonian vs. time for Hill's spherical vortex, $\mathbf{X}_0 = (0.4, 0.3, 0)$, $t_f = 10$, $\Delta t = 0.1$, for various values of Δx.



(a) Forward (b) Backward

Figure 13: M for Duffing oscillator, $\tau = 5$, $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation



(a) $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation

(b) Mancho et al [7]

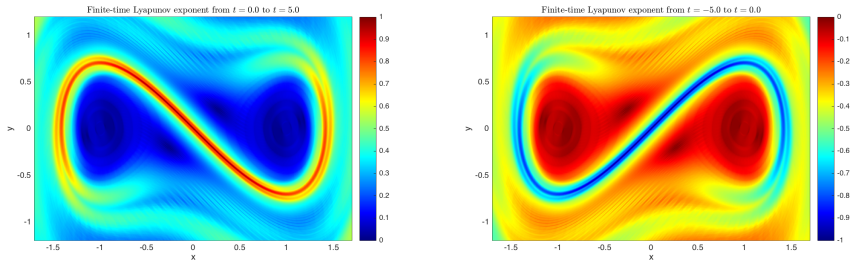
Figure 14: M for Duffing oscillator, $\tau = 10$, forward and backward together

around the origin highlighting its stable and unstable manifolds, although we also see low values of M near the elliptic fixed points at $(-1, 0)$ and $(1, 0)$, since the cyclic trajectories around those points move slowly. The choice of $\tau = 5$ here was deliberate to highlight these separate components. If we let the integration run long enough, both forward and backward, we can capture the whole stable-unstable manifold in a single plot, as in figure 14. Our results correspond closely to those of Mancho et al [7].

The corresponding FTLE plots for this case are shown in figures 15 and 16. The highlighting of the FTLE against the background is particularly stark here–indeed, since the FTLE measures deformation in the flow, we see that it is negligible near the elliptic fixed points, where there is almost no deformation. One caveat of the FTLE, however, is that the spiraling structure coming off the stable and unstable manifolds are neither part of those manifolds, nor are they aligned with the trajectories themselves. Another drawback is that if we simply overly the forward and backward FTLE fields in one figure, we get a partial cancellation which tends to obscure both stable and unstable manifolds (thus we use only the forward FTLE for validation against Mancho et al in figure 16).

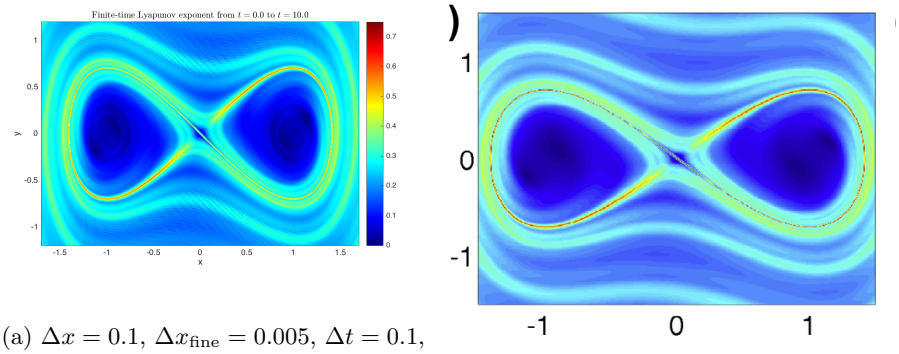
To validate our Lagrangian descriptors on a time-dependent system, we set $\epsilon = 0.1$ in the Duffing equations. Adding this small forcing term perturbs the stable and unstable manifolds in interesting ways. The results for $\tau = 10$ are shown alongside Mancho et al in figures 17 (M forward and backward) and 18 (FLTE forward). Note that the M plot contains the stable manifold highlighted in the FTLE plot.

Finally we move to three-dimensional validation on Hill's vortex. The fixed points are $\mathbf{x}_1 = (0, 0, -a)$ and $\mathbf{x}_2 = (0, 0, a)$, and the sphere $|\mathbf{x}| = a$, minus the two points $\mathbf{x}_1$ and $\mathbf{x}_2$, forms part of the unstable manifold of $\mathbf{x}_1$ and the stable manifold of $\mathbf{x}_2$. The remaining parts of these manifolds are merely portions of the z-axis. Figures 19a and 19b show the results for M and FTLE after a suitable τ. In both cases, the descriptor function "finds" the sphere, and even



(a) Forward (b) Backward

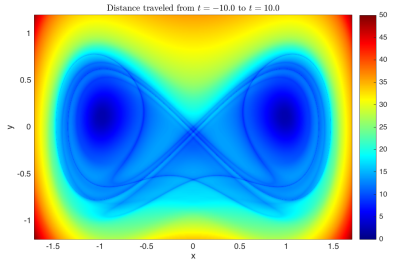
Figure 15: FTLE for Duffing oscillator, $\tau = 5$, $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation



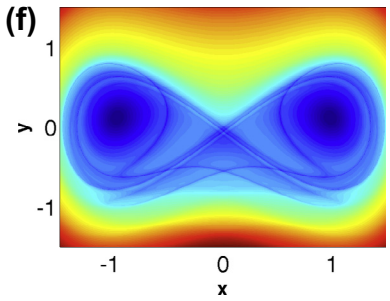
(a) $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation

(b) Mancho et al [7]

Figure 16: FTLE for Duffing oscillator, $\tau = 10$, forward and backward together

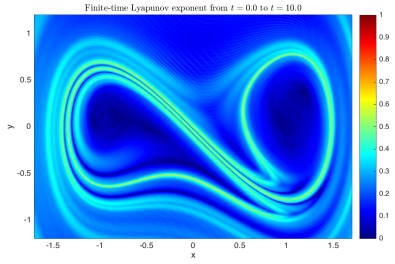


(a) $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation

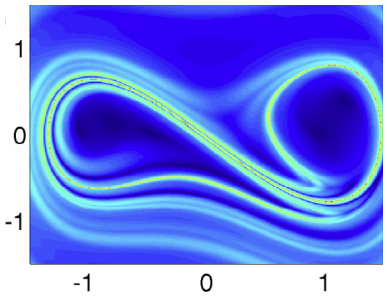


(b) Mancho et al [7]

Figure 17: M for forced Duffing oscillator, $\tau = 10$, both forward and backward



(a) $\Delta x = 0.1$, $\Delta x_{\text{fine}} = 0.005$, $\Delta t = 0.1$, bilinear interpolation



(b) Mancho et al [7]

Figure 18: FTLE for forced Duffing oscillator, $\tau = 10$, forward

the z-axis portion of the manifold, in the case of M. The white space outside the sphere comes from the exterior trajectories exiting the domain (and becoming `NaN`) on a time scale faster than the period of the inner trajectories. Again, for the FTLE, we see a slight spiraling effect near the manifold which is not indicative of actual trajectories.

6 Application

We applied our Lagrangian analysis tools to a velocity dataset coming from a modeled flow of the Chesapeake Bay. The Chesapeake Bay is an estuary in the mid-Atlantic region of the east coast of the United States, surrounded by the states of Maryland and Virginia. It is fed by an extensive river system including the Potomac and Susquehanna rivers, and its outlet is the Atlantic Ocean. The irregular topography, mixing of fresh and salt water, and tidal effects make it

an interesting area for modeling.

The particular model used was the Chesapeake Bay ROMS Community model, or ChesROMS for short (http://www.myroms.org). ChesROMS is an implementation of the more general ROMS (Regional Ocean Modeling System) developed by researchers at NOAA, the University of Maryland, CRC (Chesapeake Research Consortium) and MD DNR (Maryland Department of Natural Resources). ROMS is an ocean-modeling platform that comes with a set of specifications for grid type, coordinates, time-stepping schemes, boundary conditions, and other parameters and methods of interest. In addition to generating a grid-based velocity dataset by solving its equations of motion, it can be used to create particle trajectories through this velocity field, which will serve as a benchmark for our own trajectory calculations.

The ChesROMS model covers the whole bay, though not all of its river system, as well as a large adjoining section of the Atlantic Ocean (see figure 20b). The extent of this area is roughly 135 km by 482 km by 2.5 - 50 m, discretized to a 150 x 480 x 50 grid, so the horizontal length scale is on the order of ~ 1 km per grid box. The vertical coordinate is stretched extensively so that the lowest coordinate surface follows the ocean floor, and the highest one follows the free surface. The horizontal coordinates are aligned roughly with the Bay rather than the Earth's latitude and longitude, which in combination with the Earth's curvature leads to a slight stretching of these coordinates as well.

The Arakawa C-grid format means that horizontal indices (ξ_u, η_u) and (ξ_v, η_v), for the u-grid and v-grid respectively, are offset by $\frac{1}{2}$ from each other. All data comes in the format of netCDF files, a common one for storing geophysical data, which gives many field variables at each grid point. Of primary interest to us are physical velocity components $u := \mathbf{v} \cdot \hat{\xi}$ and $v := \mathbf{v} \cdot \hat{\eta}$ (given in m/s). Also of interest are longitude (λ), latitude (ϕ), local scale factors $m := \frac{\partial \xi}{\partial (S \cdot \hat{\xi})}$ and $n := \frac{\partial \eta}{\partial (S \cdot \hat{\eta})}$ (in m^{-1}, S representing physical arc length), and depth h (in m).

The curvilinear coordinate system necessitated a few changes to our toolset, most notably in tweaks to the interpolation and integration algorithms, which are most easily implemented on a rectangular grid. Thus the interpolation was performed in index space (using (ξ, η) as coordinates), though applied to the given physical velocity components u and v. However, these interpolated physical velocities were then normalized to index space via mu and nv (units s^{-1}), so that their integration could also be done in that space. Since the M-function is a physical quantity (m), its own integration (embedded in the RK4 time-stepping) had to use the true physical velocity components. Finally, the FTLE calculation, which requires only initial and final positions, used an approximation of physical distances via

$$\begin{aligned} \Delta x &\approx R_E \cos \phi_{\mathrm{avg}} \Delta \lambda \\ \Delta y &\approx R_E \Delta \phi \end{aligned}$$

where $R_E \approx 6371$ km is the mean Earth radius and ϕ_{avg} represents an average latitude for the domain.

The particular data used for our application was from a model flow dated February 2006, of which a total of up to 8 days worth of data was used, coming in time slices separated by two minutes. Different integration time steps ranging from 15 seconds to 2 minutes were used.

To validate the integration in the ChesROMS domain, trajectories of nine particles scattered throughout the domain were obtained from a ROMS run of 4 days backward and forward in a static flow field, and compared with our computed trajectories for the same velocity and initial position data (figures 21 and 22). The integration time step here was 2 minutes, the same as the interpolation time step, so no actual temporal interpolation should have been performed. The forward trajectories look qualitatively correct, although many seem to slightly overshoot the ROMS trajectories, indicating a possible scaling problem. Most trajectories are never more than about 5 km off the mark during this time, although several do exit the domain. The largest error comes from a trajectory in the eastern part of the bay, where the flow seems to bifurcate around an island.

The primary source of this relatively small error is unknown as of writing. One possible culprit is that there is a slight misalignment in grids, since the ROMS trajectories seem to slightly impede on the land mask in the figures. Other possibilities are that ROMS is using higher-order (likely bicubic) interpolation in the horizontal, or a higher-order integration scheme.[1]

The M and FTLE function were calculated for a time-varying flow field (starting at a different time now) for values of τ ranging from 6 hours to 4 days. (Unfortunately, time did not allow for a similar validation of the M and FTLE fields for this domain, which would have been hampered by the inaccurate trajectories anyway.) Results for $\tau = 6$ hours, time step 45 s, are shown, for backward integration only in figures 23 and 24, and forward only in figures 25 and 26. The general theme is that stable and unstable manifolds are revealed by ridges of the FTLE field in forward time, and troughs in backward time, while these same structures are revealed as sharp gradients in the M-field. Thus the snaking lines that start to show up in the FTLE plots tend to outline regions of relatively constant M. However, the density of those lines is rather sensitive to the color axis used for the plot (figure 27), leading to possibly "phantom" structures.

As the time interval is stretched to a day and beyond, the coherent structures become even clearer (figure **??**). For example, around $\tau = 2$ days backwards, the high-M structure around $\xi = 115$, $\eta = 200$ is clearly delineated from the

[1] ROMS documentation seems to indicate that ROMS uses a predictor-corrector method [4] with fourth-order Milne for the predictor:

$$\hat{\mathbf{X}}_{n+1} = \mathbf{X}_{n-3} + \frac{4\Delta t}{3}\left(2\mathbf{v}(\mathbf{X}_n, t_n) - \mathbf{v}(\mathbf{X}_{n-1}, t_{n-1}) + 2\mathbf{v}(\mathbf{X}_{n-2}, t_{n-2})\right)$$

and fourth-order Hamming for the corrector:

$$\mathbf{X}_{n+1} = \frac{9}{8}\mathbf{X}_n - \frac{1}{8}\mathbf{X}_{n-2} + \frac{3\Delta t}{8}\left(\mathbf{v}(\hat{\mathbf{X}}_{n+1}, t_{n+1}) + 2\mathbf{v}(\mathbf{X}_n, t_n) - \mathbf{v}(\mathbf{X}_{n-1}, t_{n-1})\right)$$

for the corrector. Unfortunately, ROMS documentation was not forthcoming with its interpolation methods for Lagrangian drifters.

low-M structures nearby, particularly to the northeast. This indication of a group of particles eventually moving together fast in backward time (presumably as they enter the mouth of the bay together) is further confirmed in the $\tau = 4$ days backward plot (figure 29), in which these particles disappear entirely, indicating that they eventually exit the domain en masse. The corresponding FTLE fields (figure 30) are in close correspondence with these M-fields, in particular highlighting the northeast boundary of the aforementioned structure, as well as structural boundaries around $\xi = 95, \eta = 105$, $\xi = 100, \eta = 155$, and $\xi = 135, \eta = 180$, all of which are visible as sharp gradients in M. It's worth noting, however, that the color scales are not constant in time in these FTLE plots, rather they were specifically adjusted to bring out these very structures, while suppressing some of the less obvious ones. This choice of color scale, as well the corresponding colormap, is clearly an important aspect of any such plot, and should be carefully considered and noted.

This disappearance of particles over long enough time scales (e.g. 4 days) is obviously problematic for visualization. Of course, this is inevitable with the current decision to ignore particles outside the domain, since more and more will in fact come into contact with boundaries eventually. In the future, it would be worth exploring how these plots change with different boundary conditions. For example, a free-slip condition would allow particles to drift along coastlines with the local tangential velocity, but this was not considered in the project.

7 Conclusions

We created tools for the M and FTLE and applied these tools to successfully reveal the dynamics of simple ODE systems in two and three dimensions. We then applied slightly altered versions of these tools to a model two-dimensional flow of the Chesapeake Bay and, although our trajectory integration had some potential inaccuracies which merit further exploration, we nevertheless succeeded in calculating these trajectories over time spans up to 4 days, and visualizing the corresponding M and FTLE fields. As in the toy examples, these visualizations revealed the stable and unstable manifolds in the flow usually via ridges and troughs of the FTLE field and sharp gradients in the M field. Immediate future work would involve diagnosing the possible errors in the ChesROMS trajectory calculation (especially backwards), as well as applying the tools to the full three-dimensional flow to see, how the coherent structures develop as τ increases in a cross-section of the mouth of the bay. We would expect to identify clear structures indicating the boundaries between inflowing saltwater and outgoing freshwater, and observe how their spatial placement is affected by the Coriolis force. In addition, we might hope to implement more realistic boundary conditions for particles, disabling the current tendency for white space to grow over time.

8 Timeline

- First Semester
  - First half: October - Mid-November
    - Project proposal presentation and paper
    - 2D and 3D interpolation
  - Second half: Mid-November - December
    - 2D trajectory implementation and validation
    - M function implementation and validation
    - Mid-year report and presentation
- Second Semester
  - First half: January - February
    - 3D trajectory implementation and validation
    - FTLE implementation
    - Tailor all existing code to work with ROMS data
  - Second half: March - April
    - Apply tools to Chesapeake dataset
    - Further analysis of Chesapeake dataset?
    - Final presentation and paper

9 Deliverables

- Code
  - Routines that lay down particle lattice and calculate trajectories from velocity data
  - Routines that calculate M-function and FTLE based on trajectories
- Results
  - Series of visualizations (images, movies, graphs) based on these functions, for Chesapeake Bay data and test problems
- Reports
  - Project proposal and presentation
  - Mid-year progress report and presentation
  - Final paper and presentation
- Databases
  - Chesapeake Bay ROMS dataset

References

[1] Kathleen T Alligood, Tim D Sauer, and James A Yorke. *Chaos.* Springer, 1997.

[2] Wikimedia Commons. Landsat image of chesapeake bay (cropped). `https://commons.wikimedia.org/wiki/File:Chesapeakelandsat.jpeg`. Accessed: 2016-05-13.

[3] Georg Duffing. *Erzwungene Schwingungen bei veränderlicher Eigenfrequenz und ihre technische Bedeutung.* Number 41-42. R, Vieweg & Sohn, 1918.

[4] Richard Wesley Hamming. Stable predictor-corrector methods for ordinary differential equations. *Journal of the ACM (JACM)*, 6(1):37–47, 1959.

[5] Micaiah John Muller Hill. On a spherical vortex. *Proceedings of the Royal Society of London*, 55(331-335):219–224, 1894.

[6] K. Ide, D. Small, and S. Wiggins. Distinguished hyperbolic trajectories in time-dependent fluid flows: analytical and computational approach for velocity fields defined as data sets. *Nonlinear Processes in Geophysics*, 9(3/4):237–263, 2002.

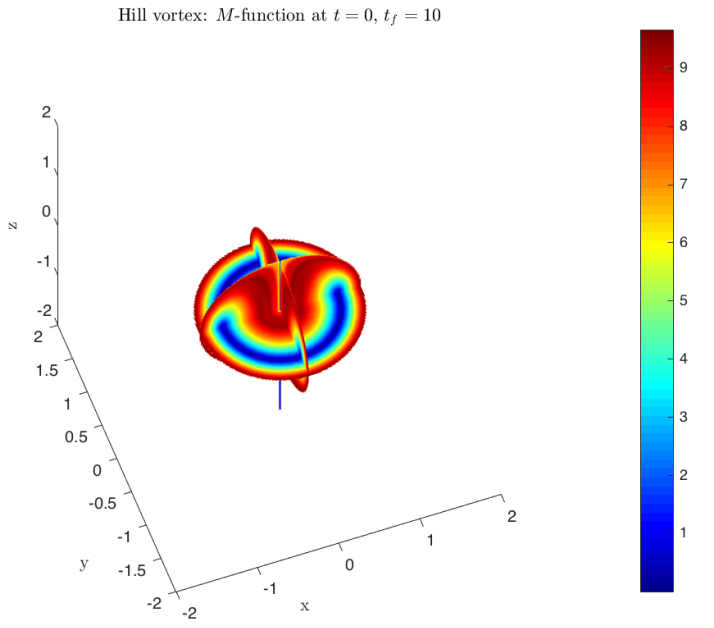
[7] Ana M Mancho, Stephen Wiggins, Jezabel Curbelo, and Carolina Mendoza. Lagrangian descriptors: A method for revealing phase space structures of general time dependent dynamical systems. *Communications in Nonlinear Science and Numerical Simulation*, 18(12):3530–3557, 2013.

[8] Carolina Mendoza and Ana M Mancho. Hidden geometry of ocean flows. *Physical review letters*, 105(3):038501, 2010.

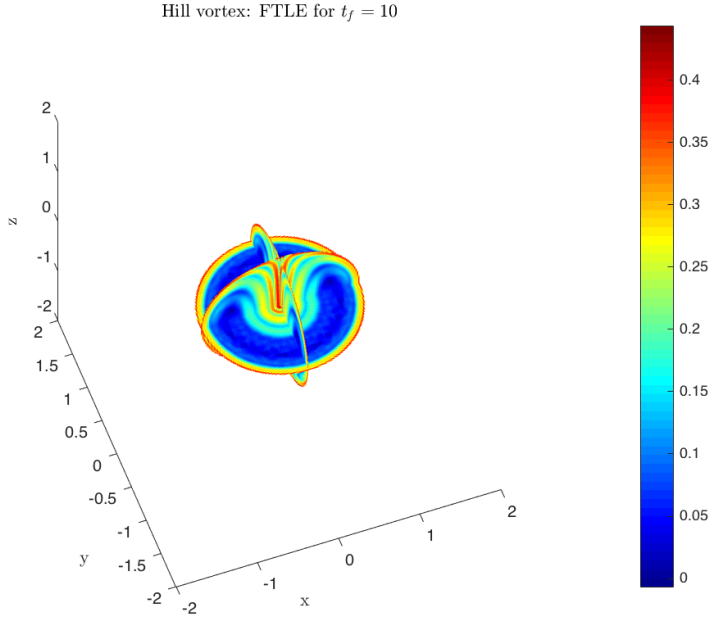
[9] Alvaro H Salas. Exact solution to duffing equation and the pendulum equation. *Applied Mathematical Sciences*, 8(176):8781–8789, 2014.

[10] Shawn C Shadden, Francois Lekien, and Jerrold E Marsden. Definition and properties of lagrangian coherent structures from finite-time lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3):271–304, 2005.

[11] Reiner L. Stenzel. Online image of hill's spherical vortex. `http://www.physics.ucla.edu/plasma-exp/conferences/TopicalConferences/IPELS97/Spheromak.html`. Accessed: 2016-05-13.

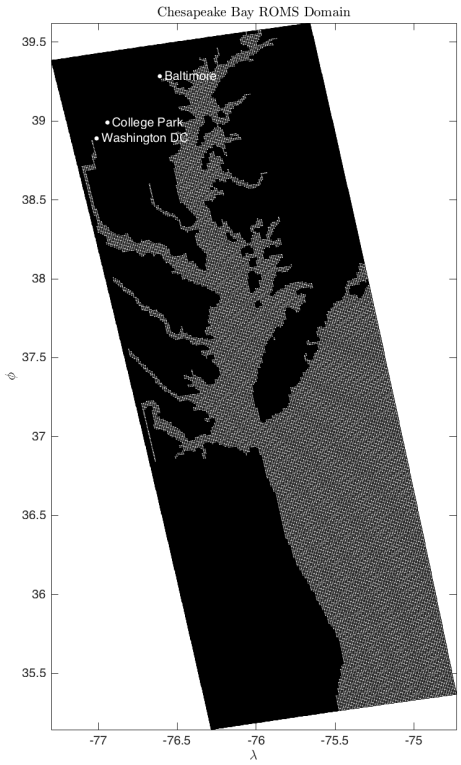


(a) Hill's vortex, M, $\tau = 10$, forward

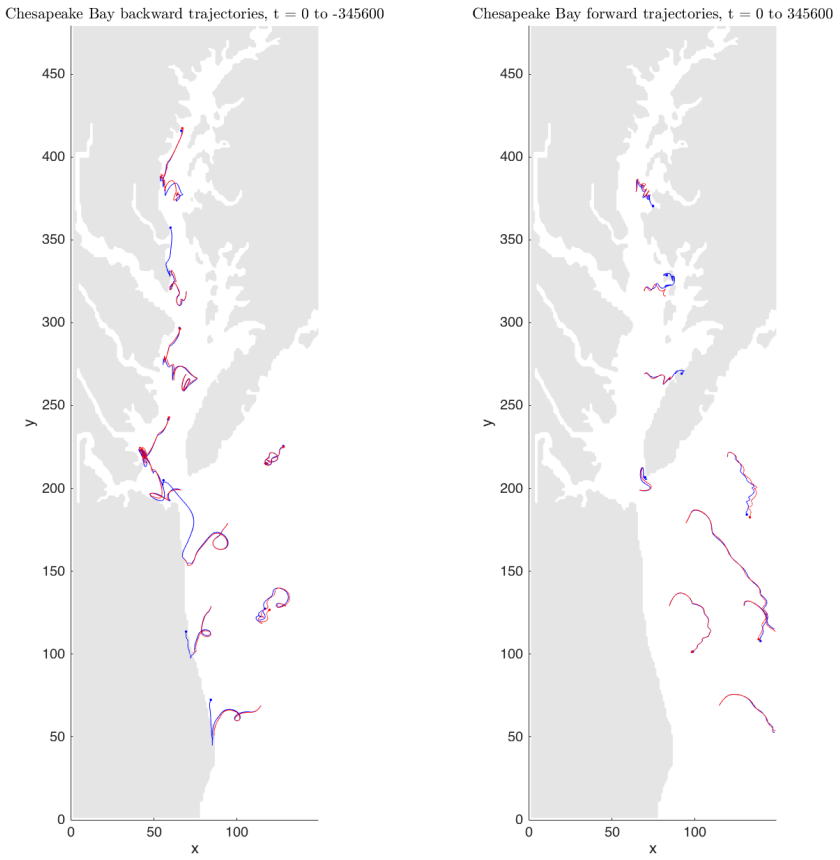


(b) Hill's vortex, FTLE, $\tau = 10$, forward

(a) The Chesapeake Bay [2]

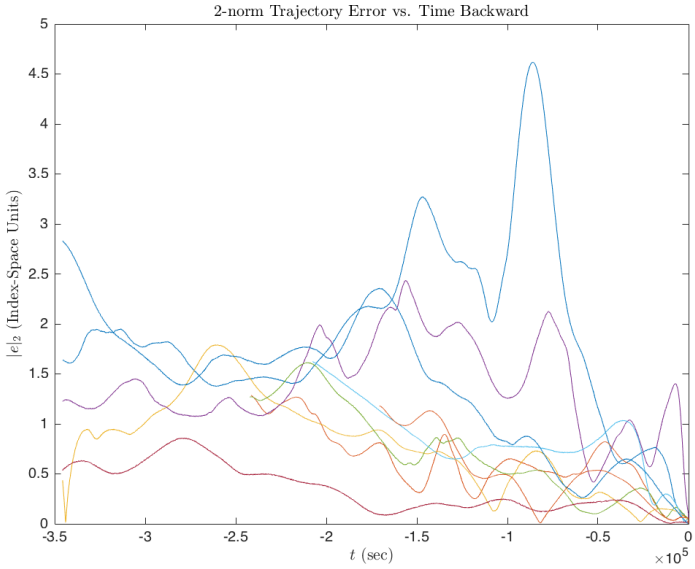


(b) The horizontal domain of the ChesROMS dataset, latitude vs. longitude, with land areas blacked out.

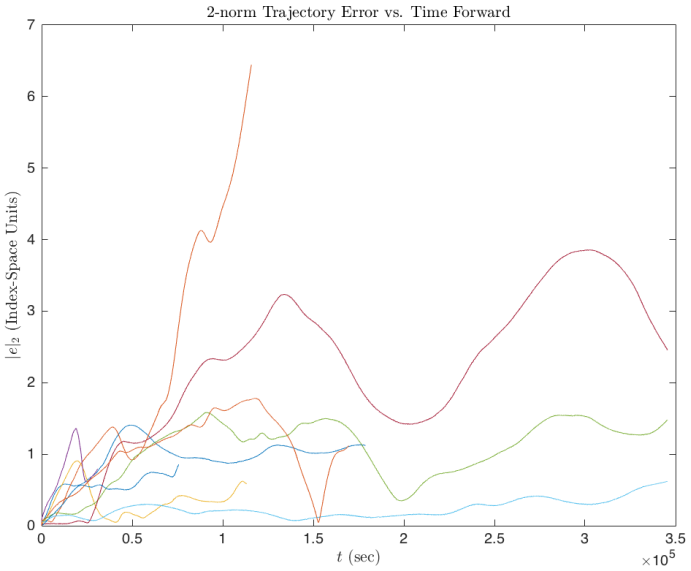


(a) Backward (b) Forward

Figure 21: Our computed trajectories (red) vs. ROMS trajectories (blue), $\tau =$ 4 days, backward and forward (ROMS trajectories courtesy of Bin Zhang, Cooperative Institute for Climate and Satellites-Maryland)

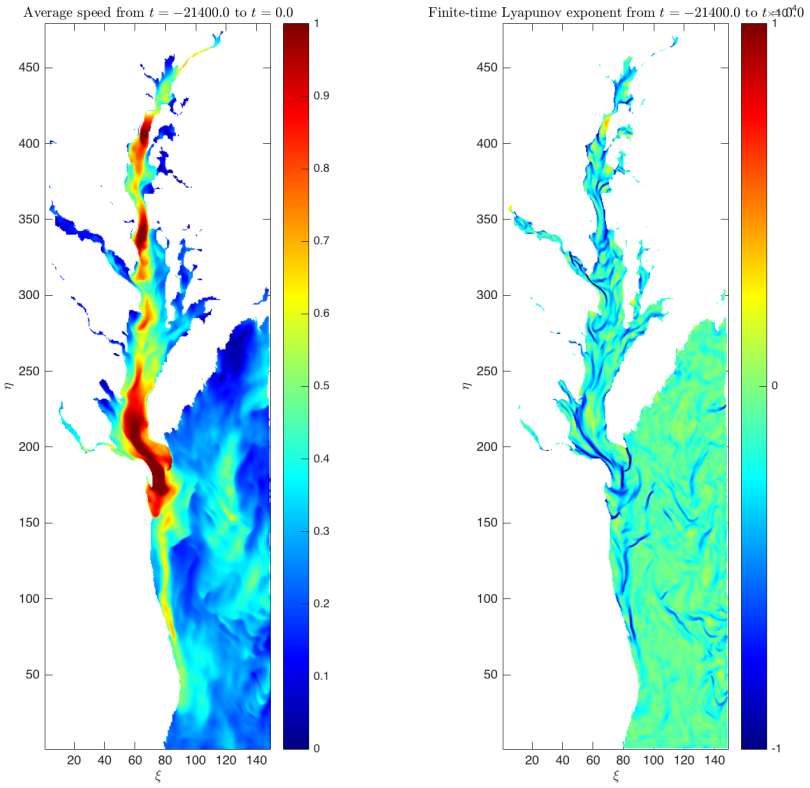


(a) Backward



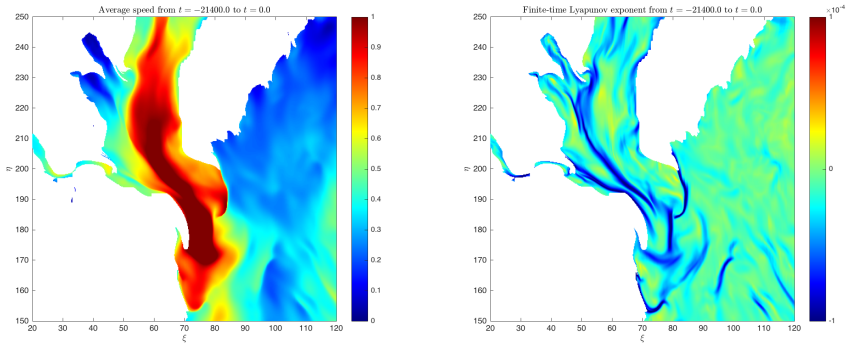
(b) Forward

Figure 22: 2-norm index-space Chesapeake trajectory error (compared to ROMS) vs. time, $\tau = 4$ days, forward and backward. Each line represents one of nine particles. Vertical units are roughly km.



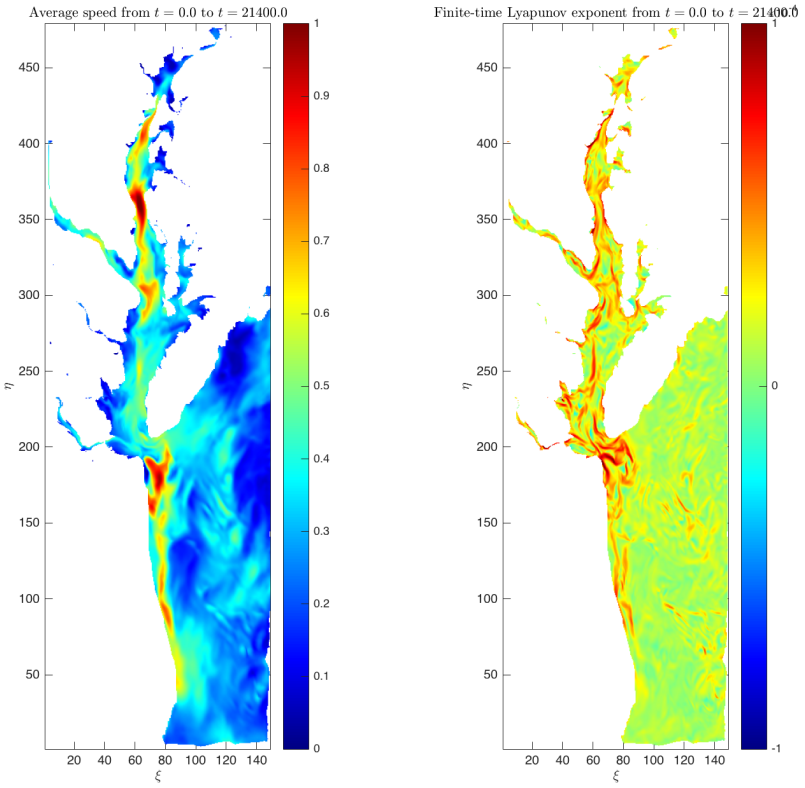
(a) $\frac{M}{\tau}$ (b) FTLE

Figure 23: Comparison of average speed ($\frac{M}{\tau}$) vs. FTLE for $\tau = 6$ hours, backward only



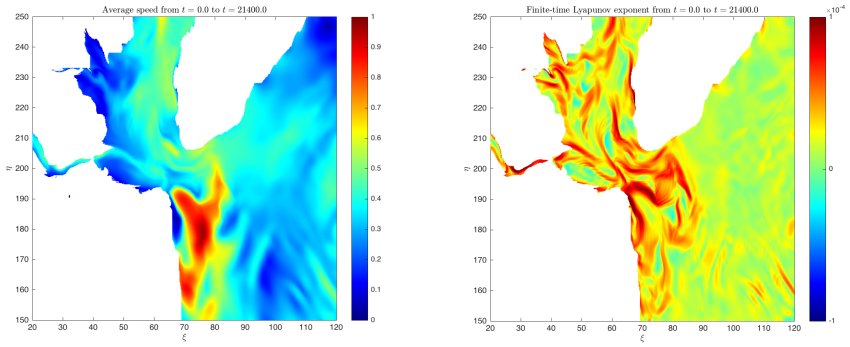
(a) $\frac{M}{\tau}$ (b) FTLE

Figure 24: Closeup of figure 23, mouth of bay



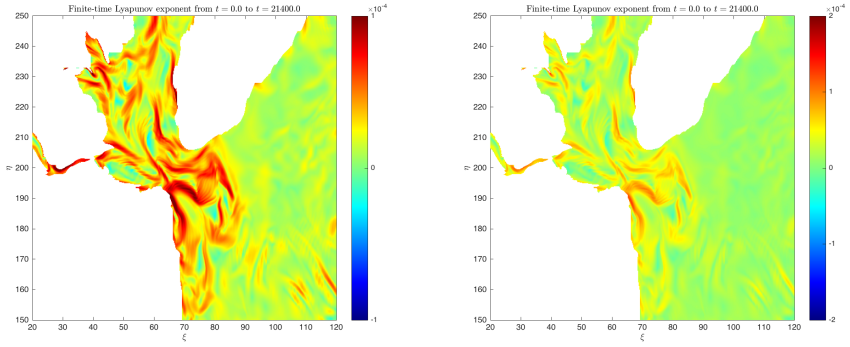
(a) $\frac{M}{\tau}$ (b) FTLE

Figure 25: Comparison of average speed ($\frac{M}{\tau}$) vs. FTLE for $\tau = 6$ hours, forward only



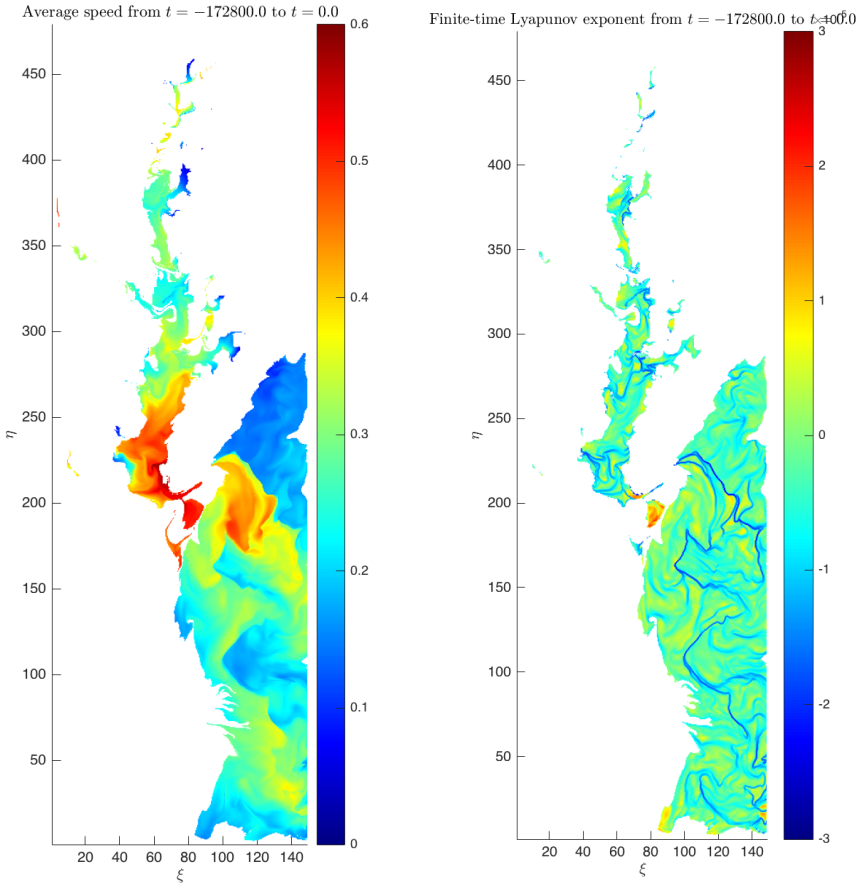
(a) $\frac{M}{\tau}$ (b) FTLE

Figure 26: Closeup of figure 25, mouth of bay



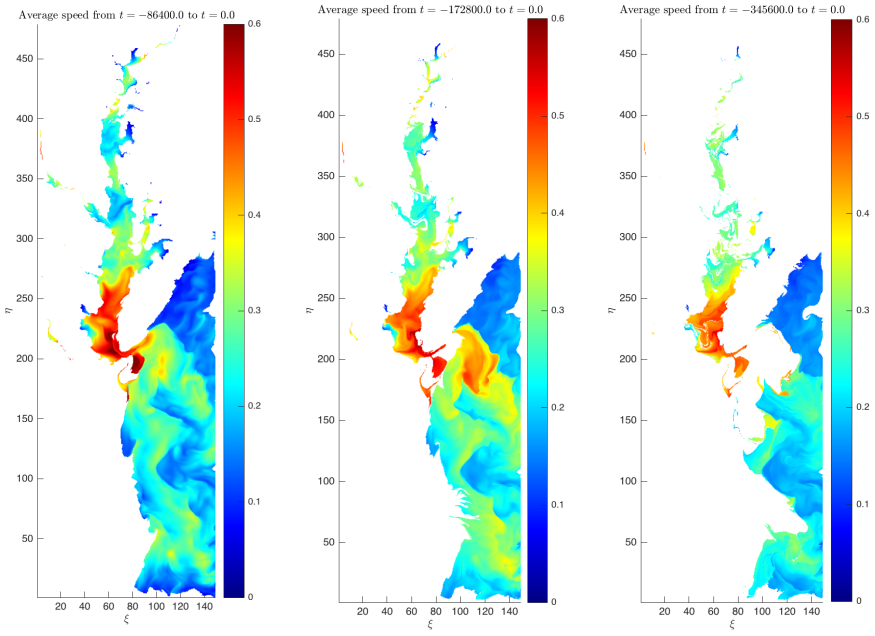
(a) FTLE, color axis up to 10^{-4} (b) FTLE, color axis up to $2.2 \cdot 10^{-4}$

Figure 27: Changing the color axis can affect one's perception of the structure of an FTLE field.



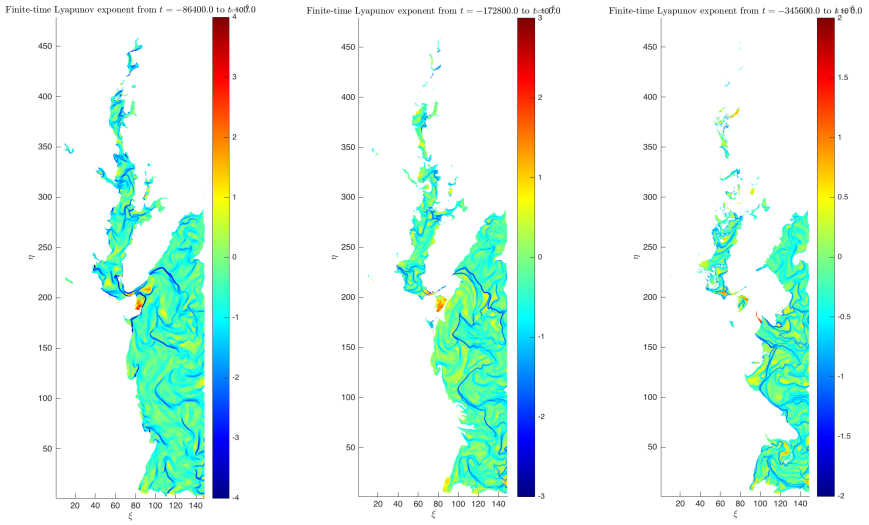
(a) $\frac{M}{\tau}$ (b) FTLE

Figure 28: Comparison of $\frac{M}{\tau}$ vs. FTLE for $\tau = 2$ days, backward only



(a) $\tau = 1$ day (b) $\tau = 2$ days (c) $\tau = 4$ days

Figure 29: Evolution of $\frac{M}{\tau}$ for τ from 1 to 4 days, backward only



(a) $\tau = 1$ day (b) $\tau = 2$ days (c) $\tau = 4$ days

Figure 30: Evolution of FTLE for τ from 1 to 4 days, backward only