Lagrangian Analysis of Two- and Three-Dimensional Oceanic Flows from Eulerian Velocity Data

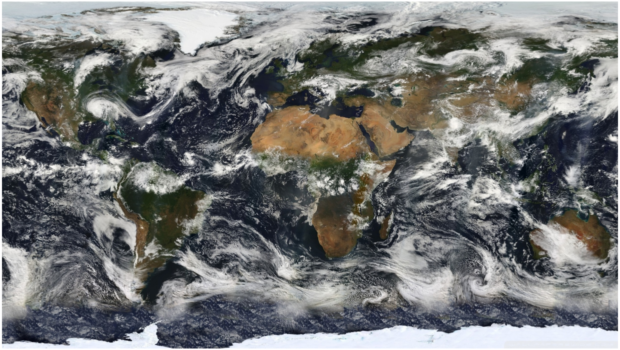
David Russell
Second-year Ph.D. student, Applied Math and Scientific Computing

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

October 6, 2015

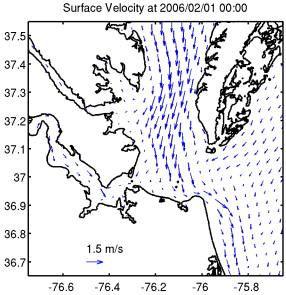
Introduction

- Atmospheric flows have various coherent structures, such as fronts, jet streams, and hurricanes
- Clouds often provide a ready means of visually tracking these flows



Introduction

- The oceans have no built-in "tracers" to track the flow, so their structure remains largely hidden
- Grid-based computer models can generate velocity data at certain discrete points in space and time, but these data alone do not reveal coherent structures in the flow



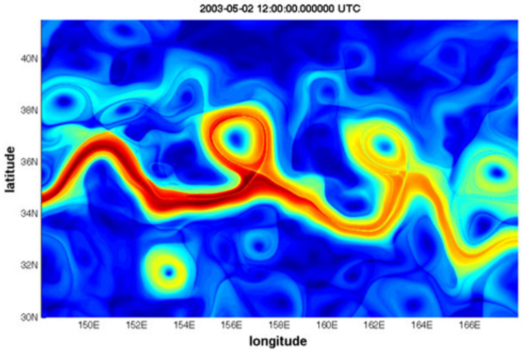
Introduction

- To better visualize the underlying structures, we can lay down a network of particles and track them as they move through the flow
- Using certain quantitative analysis tools to color the particles, we can clearly delineate coherent structures, as well as the manifolds (boundaries) that separate them
- We can also use these tools as a launching point for a deeper investigation of the mixing and transport properties of a given flow

Introduction

Example: Kuroshio current, northwest Pacific Ocean

- Red indicates faster-moving regions, blue slower
- Thin yellowish lines represent stable and unstable manifolds
- Coherent structures clearly visible



Introduction

- This approach entails a shift from an “Eulerian” to a “Lagrangian” perspective:
  - Eulerian: Track fluid velocity from a fixed point in space (e.g. an observing station)
  - Lagrangrian: Track fluid velocity following a tiny parcel of fluid as it moves through the flow

Introduction

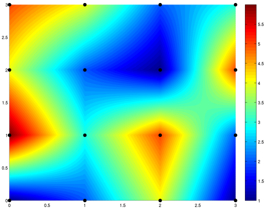
- Our goals for this project are:
  - Given a two- or three-dimensional velocity dataset in this Eulerian form, convert it to a Lagrangian form by calculating trajectories for a large number of particles laid out in a lattice
  - Design tools to visualize and analyze the flow structures based on this Lagrangian data
  - Apply these tools to a flow field from a computer model of the Chesapeake Bay, see what we can learn about its transport and mixing properties

Approach

- $\boldsymbol{X}(\boldsymbol{X}_0, t)$ is (2D or 3D) position of particle at time t that began at position $\boldsymbol{X}_0$
- $\boldsymbol{u}(\boldsymbol{x}, t)$ is velocity field at position $\boldsymbol{x}$ at time t
  - 2D: $\boldsymbol{u}(\boldsymbol{x}, t) = (u(x, y, t), v(x, y, t))$
  - 3D: $\boldsymbol{u}(\boldsymbol{x}, t) = (u(x, y, z, t), v(x, y, z, t), w(x, y, z, t))$
- These quantities must be related by $\frac{d\boldsymbol{X}}{dt}(\boldsymbol{X}_0, t) = \boldsymbol{u}(\boldsymbol{X}_0, t)$
- In practice, however, model only gives $\boldsymbol{u}$ at discrete grid points, say (x_i, y_j, z_k, t_l) in 3D
- Particles travel between grid points (in time and space), so we need a way to interpolate $\boldsymbol{u}$ to these non-grid points

Velocity Interpolation

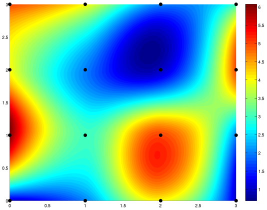
- Horizontal spatial interpolation: Bilinear interpolation (simplest method)



- Approximate velocity within each grid box by a function of the form $f(x, y) = c_1 + c_2 x + c_3 y + c_4 xy$ (four constants determined by known values at corners)
- Not very accurate (first-order), not smooth, but fast

Velocity Interpolation

- Horizontal spatial interpolation: Bicubic interpolation (higher-order method)



- Approximate by a function of the form $f(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} c_{ij} x^i y^j$
- 16 unknowns, so we also need approximations of f_x, f_y, and f_{xy} at the corners (can get these using centered difference approximations)
- Smoother and more accurate, but slower

Velocity Interpolation

- Vertical spatial interpolation (for 3D)
  - Low-order method: Interpolate linearly from horizontal interpolation
  - Higher-order method: Use cubic polynomial passing through four nearest points?
- Time interpolation
  - Low-order method: Interpolate linearly from spatial interpolation
  - High-order method: Use cubic polynomial passing through four nearest points?

Time Integration

- After interpolating $\boldsymbol{u}$ to a desired particle position, we must evolve this position in time by solving the system

$$\frac{dX}{dt} = u(X, Y, t) \qquad \frac{dY}{dt} = v(X, Y, t)$$

or

$$\frac{dX}{dt} = u(X, Y, Z, t) \qquad \frac{dY}{dt} = v(X, Y, Z, t) \qquad \frac{dZ}{dt} = w(X, Y, Z, t)$$

- Low-order method: Forward Euler (first-order)
- High-order method: 4th-order predictor-corrector using Milne for predictor and Hamming for corrector
- RK4 also a simpler higher-order option

Lagrangian Analysis Tools

- Once we have our particle trajectories, we can calculate some useful functions for analyzing the flow
- M-function: Calculates the arc length of a given trajectory over a prescribed time period (forward and backward by time τ):

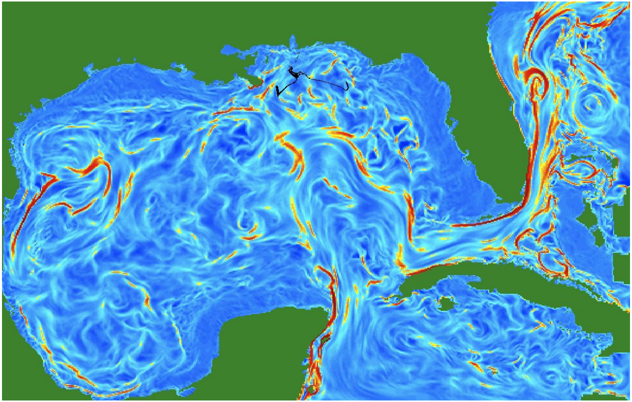
$$M_{\boldsymbol{u},\tau}(\boldsymbol{X}_0^*, t^*) = \int_{t^*-\tau}^{t^*+\tau} \left(\sum_{i=1}^{2 \text{ or } 3} \left(\frac{dX_i(t)}{dt} \right)^2 \right)^{\frac{1}{2}} dt$$

- M-function proportional to average speed near time t^*, so if we color particles by M-function, different colors will indicate different flow speeds

Lagrangian Analysis Tools

- Finite-Time Lyapunov Exponent (FTLE): Measures the degree to which two nearby trajectories diverge over time
- if $\delta\boldsymbol{X}_0$ represents an infinitesimal displacement between two nearby particles, then for certain directions, $|\delta\boldsymbol{X}_0|$ will grow or shrink exponentially in time, i.e. $|\delta\boldsymbol{X}(t)| \approx e^{\lambda t}|\delta\boldsymbol{X}_0|$
- (Maximum) FTLE is defined as the largest such growth rate λ
- FTLE grows large when flow bifurcates, so coloring by FTLE should also reveal coherent structures

Finite-Time Lyapunov Exponent



Finite-Time Lyapunov Exponent

- To calculate the FTLE at a given time t, let $L = \frac{\partial \mathbf{X}(t)}{\partial \mathbf{X}_0}$ be the transition matrix (Jacobian of position with respect to initial position)
- FTLE is given by

$$\begin{aligned} \lambda &\approx \frac{1}{t} \ln(\text{largest singular value of L}) \\ &= \frac{1}{2t} \ln\left(\text{largest eigenvalue of } L^T L\right) \end{aligned}$$

- So to calculate λ, we need to approximate L as well as find the eigenvalues of a 2x2 or 3x3 matrix

Finite-Time Lyapunov Exponent

- To approximate L at time t, we begin by placing particles very close to given particle in each coordinate direction (4 for 2D, 6 for 3D) at specified starting time
- We find their positions at time t by calculating trajectories
- Using these final positions, we can approximate L using finite differences, e.g. in the 2D case we have

$$L \approx \begin{bmatrix} \frac{\Delta X_x(t)}{\Delta X_0} & \frac{\Delta X_y(t)}{\Delta Y_0} \\ \\ \frac{\Delta Y_x(t)}{\Delta X_0} & \frac{\Delta Y_y(t)}{\Delta Y_0} \end{bmatrix}$$

where $\Delta X_x(t)$ is the final x-separation of particles that started out separated in only the x-direction, etc.

Finite-Time Lyapunov Exponents

- To calculate the eigenvalues of the 2x2 or 3x3 matrix L, we must solve its characteristic equation
- 2D: The characteristic equation is quadratic, so we can simply use the quadratic formula
- 3D: The characteristic equation is cubic, so we must write a cubic solver (based on Newton's method?)

Implementation

- Software: MATLAB
- Hardware: MacBook Pro laptop (mid-2014), 2.6 GHz Intel Core i5, 8 GB RAM (can access Deepthought2 HPC cluster on campus if necessary)
- Circa 800,000 particles to track, so trajectory computations will get very expensive
- However, each trajectory is independent, so all calculations can be done in parallel
- Can investigate accuracy vs. speed tradeoff for interpolation and integration algorithms

Validation

- Validate our tools by testing them on a few dynamical systems with well known phase portraits (stable and unstable manifolds, typical trajectories, etc.)
- Evaluate velocity for each of these systems on a fine enough grid, then feed this velocity data into our Lagrangian routines
- Routines should reproduce the known phase portraits

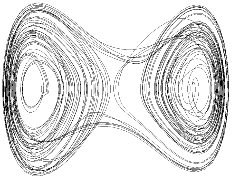
Test Problems

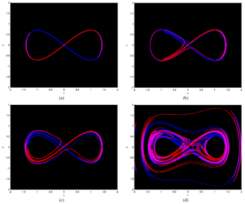
- Forced and unforced Duffing oscillator

$$\frac{dx}{dt} = y$$

$$\frac{dy}{dt} = x - x^3 + \epsilon \sin t$$

(set $\epsilon = 0$ for unforced)

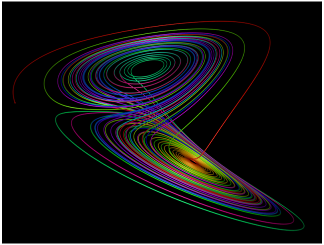




Test Problems

- Three-variable Lorenz equations

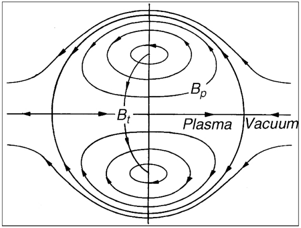
$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = rx - y - xz$$
$$\frac{dz}{dt} = xy - bz$$



Test Problems

- Hill's spherical vortex

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



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

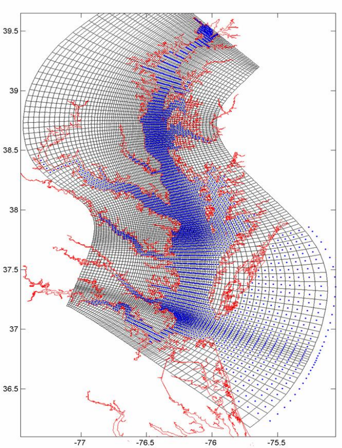
$$u_\theta = -\frac{1}{r \sin \theta} \frac{\partial \psi}{\partial r}$$

Application

- Apply tools to velocity data for the Chesapeake Bay
- Data is output from a ROMS (Regional Ocean Modeling System) model of the bay
- Tools must be able to handle ROMS specifications

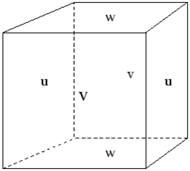
Regional Ocean Modeling System

- ROMS is a free-surface, terrain-following, primitive equations ocean model that can be adapted to various regions
- Chesapeake ROMS uses curvilinear coordinates tailored to geography
- All algorithms will be applied in this coordinate system



Regional Ocean Modeling System

- Grid type is so-called Arakawa C-grid (u, v, and w evaluated on different faces of each grid box)

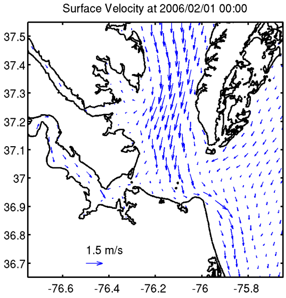


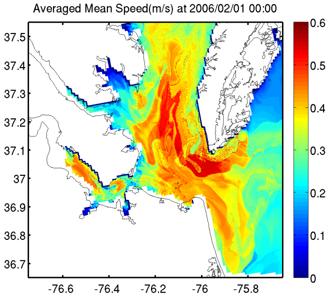
- This means u, v, and w will be interpolated relative to different grids
- Boundary conditions: no-slip ($\boldsymbol{u} = 0$ at boundary) or free-slip ($\boldsymbol{u} \cdot \hat{\boldsymbol{n}} = 0$ at boundary)

Application

- Expect M-function and FTLE data to reveal coherent structures in Chesapeake Bay dataset
- Use Chesapeake ROMS data as one more validation test (test everything against results of another researcher in group)
- Time permitting, use Lagrangian tools to quantitatively investigate transport and mixing processes, e. g.:
  - What percentage of the water entering the bay from rivers/ocean also exits through the rivers/ocean within a certain timeframe?
  - Can we observe and quantify dynamic effects of Coriolis force, density differences between ocean and fresh water, etc.

Expected Visualization Results





Schedule

- 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
  - Second half: March - April
    - Visualizations and further analysis
    - Final presentation and paper

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

Deliverables

- Reports
  - Project proposal and presentation
  - Mid-year progress report and presentation
  - Final paper and presentation
- Databases
  - Chesapeake Bay ROMS dataset

Bibliography

- Mancho, A. M., Wiggins S., Curbelo J. & Mendoza, C. (2013). Lagrangian descriptors: A method for revealing phase space structures of general time dependent dynamical systems. *Communications in Nonlinear Science and Numerical Simulation, 18*, pp. 3530-3557.
- Mendoza, C. & Mancho, A. M. (2010). Hidden geometry of ocean flows. *Physical Review Letters, 105*(038501), pp. 1-4.
- Shadden, S. C., Lekien, F. & Marsden, J. (2005). Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D, 212*, pp. 271-304
- Xu, J. et al. (2012). Climate forcing and salinity variability in Chesapeake Bay, USA. *Estuaries and Coasts, 35*, pp. 237-261.