# Mid-year 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

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

December 8th, 2015

# Project Overview

- Project goal: Create tools for Lagrangian analysis of oceanic flow (2D or 3D), given only velocity data on spatio-temporal grid (e.g. from model output)
- Track vast network of particles through flow, use trajectories to bring out underlying Lagrangian coherent structures, as well as stable and unstable manifolds separating these structures
- Test tools on velocity output from Chesapeake Bay model

# Project Overview

Example: Kuroshio current, northwest Pacific Ocean

- Red indicates faster-moving regions, blue slower
- Thin yellow lines represent stable and unstable manifolds

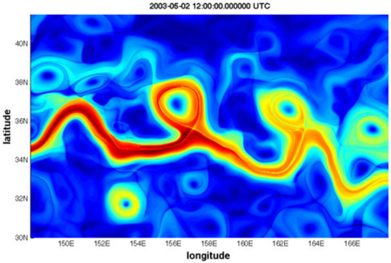



Figure: Coherent structures in the Kuroshio current [1]

# Project Overview

- Velocity field $\boldsymbol{u}(x_i, y_j, z_k, t_l) = (u, v, w)$ given at discrete points in space and time
- Want to compute particle trajectory $\boldsymbol{X}(\boldsymbol{X}_0, t)$ given initial position $\boldsymbol{X}_0 \in \mathbb{R}^2$ or $\mathbb{R}^3$
- Two numerical tasks: *interpolation* and *time integration*
  - *Interpolation*: Particle velocities $\boldsymbol{u}(\boldsymbol{X}(\boldsymbol{X}_0, t))$ must be interpolated from grid velocities $\boldsymbol{u}_{i,j,k,l}$
  - *Integration*: Velocity must be integrated in time to obtain position as time evolves:

$$\boldsymbol{X}(\boldsymbol{X}_0, t) = \int_0^t \boldsymbol{u}(\boldsymbol{X}(\boldsymbol{X}_0, t'), t')\, dt'$$

# Project Overview



- Given trajectories, two main tools for Lagrangian analysis: *M-function* and *largest finite-time Lyapunov exponent (FTLE)*
- *M-function*: distance traveled by particle within some fixed time interval (e.g. $\tau$ forward and backward from current time):

$$M_{\boldsymbol{u},\tau}(\boldsymbol{X}_0, t) = \int_{t-\tau}^{t+\tau} \left( \sum_{i=1}^{2 \text{ or } 3} \left( \frac{dX_i(\boldsymbol{X}_0, t')}{dt'} \right)^2 \right)^{\frac{1}{2}} dt'$$

- Coloring by $M$-function brings out boundaries between coherent structures moving at different speeds

# Project Overview



- Given trajectories, two main tools for Lagrangian analysis: *M-function* and *largest finite-time Lyapunov exponent (FTLE)*
- *FTLE*: exponential growth rate along maximal growth axis of an infinitesimal parcel of fluid:

$$\lambda(t) = \frac{1}{2t} \ln \left( \rho(L^T L)\} \right)$$

  where $\rho$ denotes the spectral radius and $L(t) = \frac{\partial \mathbf{X}(\mathbf{X}_0, t)}{\partial \mathbf{X}_0}$.
- Coloring by FTLE highlights bifurcation regions in flow

# Implementation Basics

- Store particle positions in column vectors `Xp`, `Yp`, `Zp`
- Basic principle: vectorize all operations (all particles at once rather than looping through them)
- Input velocity data in Arakawa C-grid format ($u$ given at east and west sides of grid box, $v$ at north and south, $w$ at top and bottom):

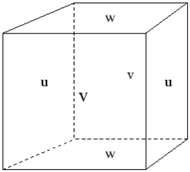



Figure: Arakawa c-grid box [5]

# 2D Interpolation Algorithms



- Compare two methods for 2D spatial interpolation: piecewise *bilinear* and piecewise *bicubic* functions (splines)
  - Bilinear: Within each grid box, fit function of form $p(x,y) = \sum_{i,j=0}^{1} a_{ij} x^i y^j = a_{00} + a_{10}x + a_{01}y + a_{11}xy$ to four corner values of $u$
  - Bicubic: Fit function of form $p(x,y) = \sum_{i,j=0}^{3} a_{ij} x^i y^j$ to four corner values of $u$ and estimates of its derivatives $u_x$, $u_y$, and $u_{xy}$
- Bicubic should be slower but more accurate

# Bilinear Interpolation



- Approximate $u(x, y)$ on $[x_i, x_{i+1}] \times [y_j, y_{j+1}]$ by

$$p(x, y) = a_{00} + a_{10}x + a_{01}y + a_{11}xy$$

- Explicit formula:

$$\begin{aligned} p(x, y) = {} & w_y \left( w_x u_{i,j} + (1 - w_x) u_{i+1,j} \right) \\ & + (1 - w_y) \left( w_x u_{i,j+1} + (1 - w_x) u_{i+1,j+1} \right) \end{aligned}$$

with weights

$$w_x = \frac{x_{i+1} - x}{x_{i+1} - x_i}$$

$$w_y = \frac{y_{j+1} - y}{y_{j+1} - y_j}$$

# Bicubic Interpolation

- Approximate $u(x, y)$ on $[0, 1]^2$ by

$$p(x, y) = \sum_{i,j=0}^{3} a_{ij} x^i y^j$$

- Find $a_{ij}$'s by matching function to known values $u(0, 0), u(0, 1), u(1, 0), u(1, 1)$ and finite difference approximations for partial derivatives $u_x$, $u_y$, and $u_{xy}$ at corners
- Sixteen parameters, sixteen unknowns
- Scale inputs from grid box to $[0, 1]^2$, solve, scale back

# Bicubic Interpolation



- Second-order centered finite difference approximations for *interior* grid points:

$$(u_x)_{i,j} \approx \frac{1}{2\Delta x}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix} u_{i-1,j} \\ u_{i,j} \\ u_{i+1,j} \end{bmatrix} = \frac{u_{i+1,j} - u_{i-1,j}}{2\Delta x}$$

$$(u_y)_{i,j} \approx \begin{bmatrix} u_{i,j-1} & u_{i,j} & u_{i,j+1}\end{bmatrix} \cdot \frac{1}{2\Delta y}\begin{bmatrix}-1 \\ 0 \\ 1\end{bmatrix} = \frac{u_{i,j+1} - u_{i,j-1}}{2\Delta y}$$

$$(u_{xy})_{i,j} \approx \frac{1}{2\Delta x}\begin{bmatrix}-1 & 0 & 1\end{bmatrix} \cdot \begin{bmatrix} u_{i-1,j-1} & u_{i-1,j} & u_{i-1,j+1} \\ u_{i,j-1} & u_{i,j} & u_{i,j+1} \\ u_{i+1,j-1} & u_{i+1,j} & u_{i+1,j+1} \end{bmatrix} \cdot \frac{1}{2\Delta y}\begin{bmatrix}-1 \\ 0 \\ 1\end{bmatrix}$$
$$= \frac{u_{i+1,j+1} - u_{i+1,j-1} - u_{i-1,j+1} + u_{i-1,j-1}}{4\Delta x \Delta y}$$

# Bicubic Interpolation



- Second-order one-sided finite difference approximations for *boundary* grid points:

$$(u_x)_{1,j} \approx \frac{1}{2\Delta x}\begin{bmatrix}-3 & 4 & -1\end{bmatrix} \cdot \begin{bmatrix} u_{1,j} \\ u_{2,j} \\ u_{3,j} \end{bmatrix}$$

$$(u_{xy})_{1,j} \approx \frac{1}{2\Delta x}\begin{bmatrix}-3 & 4 & -1\end{bmatrix} \cdot \begin{bmatrix} u_{1,j-1} & u_{1,j} & u_{1,j+1} \\ u_{2,j-1} & u_{2,j} & u_{2,j+1} \\ u_{3,j-1} & u_{3,j} & u_{3,j+1} \end{bmatrix} \cdot \frac{1}{2\Delta y}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

$$(u_{xy})_{1,1} \approx \frac{1}{2\Delta x}\begin{bmatrix}-3 & 4 & -1\end{bmatrix} \cdot \begin{bmatrix} u_{1,1} & u_{1,2} & u_{1,3} \\ u_{2,1} & u_{2,2} & u_{2,3} \\ u_{3,1} & u_{3,2} & u_{3,3} \end{bmatrix} \cdot \frac{1}{2\Delta y}\begin{bmatrix} -3 \\ 4 \\ -1 \end{bmatrix}$$

$$= \frac{9u_{1,1} - 12u_{1,2} + 3u_{1,3} - 12u_{2,1} + 16u_{2,2} - 4u_{2,3} + 3u_{3,1} - 4u_{3,2} + u_{3,3}}{4\Delta x \Delta y}$$

...and likewise for other boundary points

# Bicubic Interpolation

- Determine $a_{ij}$ from these values via

$$\begin{bmatrix} a_{00} \\ a_{10} \\ a_{20} \\ a_{30} \\ a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \\ a_{02} \\ a_{12} \\ a_{22} \\ a_{32} \\ a_{03} \\ a_{13} \\ a_{23} \\ a_{33} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -3 & 3 & 0 & 0 & -2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 & -2 & 0 & 0 & 1 & 1 & 0 & 0 \\ -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -3 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & -2 & 0 & -1 & 0 \\ 9 & -9 & -9 & 9 & 6 & 3 & -6 & -3 & 6 & -6 & 3 & -3 & 4 & 2 & 2 & 1 \\ -6 & 6 & 6 & -6 & -3 & -3 & 3 & 3 & -4 & 4 & -2 & 2 & -2 & -2 & -1 & -1 \\ 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & -2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ -6 & 6 & 6 & -6 & -4 & -2 & 4 & 2 & -3 & 3 & -3 & 3 & -2 & -1 & -2 & -1 \\ 4 & -4 & -4 & 4 & 2 & 2 & -2 & -2 & 2 & -2 & 2 & -2 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} u(0,0) \\ u(1,0) \\ u(0,1) \\ u(1,1) \\ u_x(0,0 \\ u_x(1,0 \\ u_x(0,1 \\ u_x(1,1 \\ u_y(0,0 \\ u_y(1,0 \\ u_y(0,1 \\ u_y(1,1 \\ u_{xy}(0,0 \\ u_{xy}(1,0 \\ u_{xy}(0,1 \\ u_{xy}(1,1 \end{matrix}$$

- Equations were hard-coded to allow for simultaneous operation on all particles

# Vertical and Temporal Interpolation

- For simplicity, treat these as one-dimensional interpolation problems to follow two-dimensional problem
- Again, compare two methods for each: piecewise linear (faster) and piecewise cubic (more accurate)
- Linear: given $z \in [z_k, z_{k+1}]$ and 2D-interpolated values $u_k = u(x, y, z_k)$ and $u_{k+1} = u(x, y, z_{k+1})$, approximate $u(x, y, z)$ by

$$p(z) = w \cdot u_k + (1 - w) \cdot u_{k+1}$$

where

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

# Vertical and Temporal Interpolation



- Cubic: given $z \in [z_k, z_{k+1}]$, interpolate cubic polynomial through four points $u_{k-1} = u(x, y, z_{k-1})$, $u_k = u(x, y, z_k)$, $u_{k+1} = u(x, y, z_{k+1})$, $u_{k+2} = u(x, y, z_{k+2})$
- Lagrange form:

$$
\begin{aligned}
p(z) = & \frac{(z - z_k)(z - z_{k+1})(z - z_{k+2})}{(z_{k-1} - z_k)(z_{k-1} - z_{k+1})(z_{k-1} - z_{k+2})} u_{k-1} \\
& + \frac{(z - z_{k-1})(z - z_{k+1})(z - z_{k+2})}{(z_k - z_{k-1})(z_k - z_{k+1})(z_k - z_{k+2})} u_k \\
& + \frac{(z - z_{k-1})(z - z_k)(z - z_{k+2})}{(z_{k+1} - z_{k-1})(z_{k+1} - z_k)(z_{k+1} - z_{k+2})} u_{k+1} \\
& + \frac{(z - z_{k-1})(z - z_k)(z - z_{k+1})}{(z_{k+2} - z_{k-1})(z_{k+2} - z_k)(z_{k+2} - z_{k+1})} u_{k+2}
\end{aligned}
$$

# Time Integration



- Runge-Kutta fourth order for now (simple, explicit, relatively accurate)
- Eventually will implement Milne-Hamming multistep predictor-corrector scheme in accordance with ROMS standard:

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

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

# 2D Interpolation Accuracy

- To validate and check accuracy of 2D interpolation algorithms, applied them to analytic function $u(x, y) = e^y \sin x$ on $[0, 1]^2$
- Approximation theory in 1D:
  - $n$th-order interpolating polynomial $p$ through $x_0, x_1, \ldots, x_n$ satisfies

$$u(x) - p(x) = \frac{u^{n+1}(\xi)}{(n+1)!}(x - x_0)(x - x_1)\ldots(x - x_n)$$
$$\implies \|u - p\|_\infty = O(h^{n+1})$$

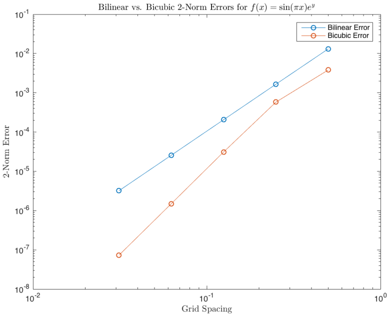
    for some $\xi \in \text{int}(x_0, x_1, \ldots, x_n)$, assuming $u \in C^{n+1}$
  - Splines: also $O(h^{n+1})$, also assuming sufficiently high-order derivative approximations used?
- Also $\|u - p\|_\infty = O(h^{n+1})$ in 2D?

# Bicubic vs. Bilinear Accuracy for $u(x,y) = e^y \sin x$



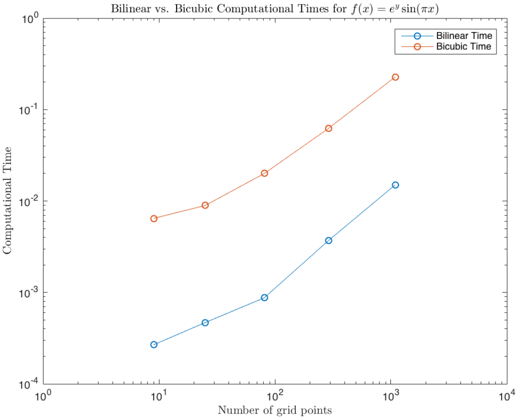
- Order of accuracy estimates ($\|u - p\|_2 = O(h^q)$):

| Bilinear | Bicubic |
|---|---|
| $q \approx 3.0029$ | $q \approx 4.3569$ |



# Bicubic vs. Bilinear Times for $u(x, y) = e^y \sin x$





# Test Problem: Unforced, Undamped Duffing Oscillator



- Unforced, undamped Duffing oscillator

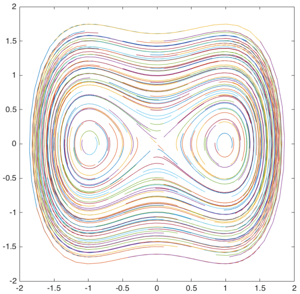
$$\frac{dx}{dt} = y$$
$$\frac{dy}{dt} = x - x^3$$

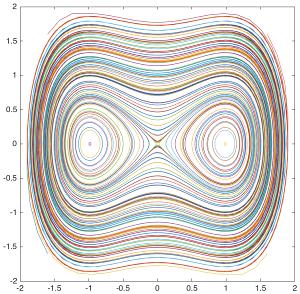
- Has exact solutions in terms of Jacobi elliptic functions

# Computed Trajectories for Duffing Oscillator (RK4)



- Stable and unstable manifolds form figure-eight around fixed points
- Still need to validate against exact solution

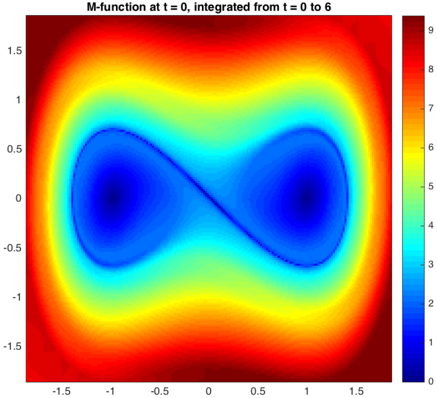


# M-Function for Duffing oscillator



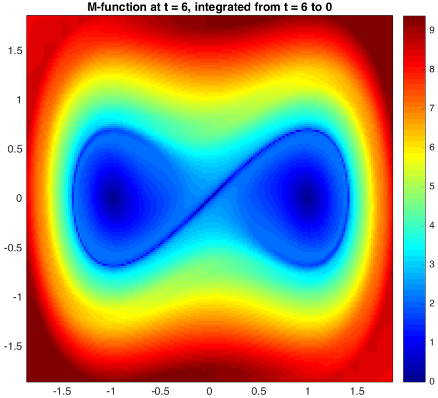
- Slower regions in blue, faster in red (integrated from $t = 0$ to $t = 6$)
- Blue line is *stable* manifold (particles slowing down as they approach fixed point at center)



# M-Function for Duffing oscillator (integrated backwards in time)



- Slower regions in blue, faster in red (integrated from $t = 0$ to $t = 6$)
- Blue line is *unstable* manifold (particles slowing down as they approach fixed point)



# Schedule

First semester:

- First half: October - Mid-November
  - Project proposal presentation and paper (done)
  - 2D and 3D interpolation
    - Need to implement cubic interpolation in vertical and time
- Second half: Mid-November - December
  - 2D trajectory implementation and validation
    - Need to validate against analytical solutions to Duffing oscillator
    - Need to validate on forced and rotating Duffing oscillator
  - M function implementation and validation
  - Mid-year report and presentation

# Schedule

- Second Semester
  - First half: January - February
    - 3D trajectory implementation and validation
    - FTLE implementation
  - Second half: March - April
    - Application to ROMS dataset
    - Visualizations and further analysis
    - Final presentation and paper

# References I



[1] Mendoza, C. & Mancho, A. M. (2010). Hidden geometry of ocean flows. *Physical Review Letters, 105* (038501), pp. 1-4.

[2] Bicubic Interpolation. (n.d.). In *Wikipedia*. Retrieved October 5, 2015 from https://en.wikipedia.org/wiki/Bicubic_interpolation.

[3] 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.

[4] Duffing Map. (n.d.). In *Wikipedia*. Retrieved October 5, 2015 from https://en.wikipedia.org/wiki/Duffing_map.

# References II



[5] [Online image of Arakawa C-grid box]. Retrieved October 5, 2015 from http://mitgcm.org/sealion/online_documents/node45.html.

[6] 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.