# Analysis of Lagrangian Coherent Structures of the Chesapeake Bay

Stephanie A E Young
syoung3@math.umd.edu
Advisor: Kayo Ide
Ide@umd.edu
Atmospheric and Oceanic Science Department
Center for Scientific Computing and Mathematical Modeling
Applied Mathematics, Statistics and Scientific Computing Program
Earth System Science Interdisciplinary Center
Institute for Physical Science and Technology

October 15, 2013

## Abstract

Numerical lagrangian analysis of the Chesapeake Bay can reveal dynamical features not obtainable through analytical means. These features can indicate coherent structures within the Bay, revealing neighboring regions of fluid that have very different behaviors. Give the model-based discrete velocity data of the Bay, obtained through the use of the Regional Ocean Modeling System (ROMS), we will implement bilinear and bicubic spatial interpolation methods and a $3^{rd}$ order lagrangian polynomial to interpolate in time. This interpolation then allows us to calculate trajectories of $\sim$1 million initial conditions, using a $5^{th}$ order Runge Kutta Fehlberg method and a $4^{th}$ order Runge Kutta method (for comparison). From these trajectories, we will apply a qualitative analysis method along with a quantitative probabalistic approach to the lagrangian analysis.

# 1 Introduction

Studying the dynamics of the earth's oceans is of great concern to many fields of study. The water systems store and transport particles and energy around the globe affecting the lives of everyone living on this planet. [1] It is therefore important to understand how the positions of a set of particles might evolve over time, given that all of the particles originated from some enclosed region.

To do this, we must start thinking in terms of langrangian dynamics. This is a very intuitive perspective to take, as it is the perspective of a person if they were to follow some parcel of particles (air or water) through space and time. Using this approach to the dynamics allows us to study individual particles as well as how individual particles move together.

Particles that move together and share similar dynamical properties are called coherent sets and are separated by what we will call manifolds. [2] Examples of these coherent sets are hurricanes and jet streams. The problem that we plan to address in this project is how to identify these structures. Locally it is clear that one of the important waterways that affects the Maryland area is the Chesapeake Bay. Therefore we will be focused on analyzing data from this particular region.

If we are given some discrete velocity field for the Bay, we would like to be able to integrate the velocity field from some initial time $t_0$ to some final time $t_f$ and calculate the position of some particle at any time in this interval, given its initial position. Due to the discrete nature of the data, if we integrate from time $t_i$ to $t_j$ our velocity field may not be defined on the point $(x_j, y_j)$ making it impossible to move forward with the integration. That is, unless we find some way to estimate that value of the velocity at $(x_j, y_j)$. We do this using interpolation methods that will be discussed in §2.1.1. These interpolation techniques allow us to calculate the trajectories that are necessary to perform our langrangian analysis.

Once we have trajectories (§2.1.2), we will analyze the dynamics of the Chesapeake Bay data using a qualitative method [1] (§2.2.1) and then with a more quantitative probablistic method [3] (§2.2.2).

# 2 Approach and algorithms

This project is split into two parts. The first part consists of computing trajectories by numerically integrating $\frac{dx}{dt} = u(x, y, t)$ and $\frac{dy}{dt} = v(x, y, t)$. $u$ and $v$ are given on a grid and therefore any time integration of $\frac{dx}{dt}$ and $\frac{dy}{dt}$ requires that we be able to interpolate $u$ and $v$.

The second part consists of implementing lagrangian analysis methods (§2.2.1 and §2.2.2) based on the trajectories calculated in part 1.

## 2.1 Part 1. Trajectory computation

### 2.1.1 Spatial Interpolation

Given some discrete velocity field we need to be able to interpolate any off-grid velocity in order to properly calculate trajectories. To do this, we will be implementing a bilinear spatial interpolation method, as well as a bicubic spatial interpolation method. We also have a time 'dimension' which will be interpolated using a $3^{rd}$ order Lagrange polynomial in time. The interpolation of $u$ (x-directed velocity) and $v$ (y-directed velocity) will be done separately. This means the interpolation of one will not depend on the interpolation of the other. [4]

Bilinear interpolation of some point requires 4 nearest points and the velocity values at those 4 points to interpolate. Using these four points we create some surface $u = a_0 + a_1x + a_2y + a_3xy$ to approximate u(x,y) (at constant time) within the grid cell created by the 4 nearest points. [5][6]

Bicubic interpolation on the otherhand requires velocity values as well as derivatives of the velocity at each of the 4 nearest points to interpolate one velocity value. [6][7] For each of these 4 nearest neighboring point $(x_i, y_j)$ we need $u(x_i, y_j)$, $u_x(x_i, y_j)$, $u_y(x_i, y_j)$, and $u_{xy}(x_i, y_j)$. This is a total of 16 pieces of data required to interpolate each velocity value. To do this we will be approximating the derivative and cross derivatives with central difference schemes.

This interpoltion creates the surface in Equation 2.1 where the 16 $b_{ij}$ coefficients need to be determined.

$$\begin{aligned} u(x,y) = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2 + b_{21}x^2y + b_{12}xy^2 + b_{22}x^2y^2 \\ + b_{30}x^3 + b_{03}y^3 + b_{31}x^3y + b_{13}xy^3 + b_{32}x^3y^2 + b_{23}x^2y^3 + b_{33}x^3y^3 \end{aligned} \tag{2.1}$$

For the purposes of the analysis of the Chesapeake Bay data we will be using bilinear interpolation, provided the bilinear method is not unreliable for interpolation of the velocity field given by Equation 3.1. The use of bilinear (instead of bicubic) is preferred due to the computational expense of bicubic. We can forsee a difference in computational time simply from observing that we need 4 times as many function and function derivative values for bicubic as we do for bilinear. If the accuracy does not suffer too greatly on the velocity field of Equation 3.1 then bilinear will certainly be used.

Time interpolation will be done with Lagrange polynomials. For some time $t \in (t_i, t_{i+1})$ the polynomial will go through time values $t_{i-1}, t_i, t_{i+1}$, and $t_{i+2}$.

Time permitting, I will use Matlab to speed up the interpolation and trajectory calculations through parallelization.

### 2.1.2 Time integration

To calculate the trajectories, I will be using two methods. First I will use a simple $4^{th}$ order Runge Kutta method (RK4) (Equation 2.2) and then secondly the Runge Kutta Fehlberg 4(5) method (RKF) (Equations 2.3 through 2.5). [8]

In Equation 2.2 the $k_i$ and $l_i$ values are $u$ and $v$ calculated at points in within the intervals $[x_n, x_{n+1}]$ and $[x_n, x_{n+1}]$ respectively. We then use a weighted average of these values to determine the final value of $(x_{n+1}, y_{n+1})$. h is the fixed time step used.

$$
\begin{aligned}
k_1 &= u(t_n, x_n, y_n) \\
l_1 &= v(t_n, x_n, y_n) \\
k_2 &= u(t_n + h/2, x_n + hk_1/2, y_n + hl_1/2) \\
l_2 &= v(t_n + h/2, x_n + hk_1/2, y_n + hl_1/2) \\
k_3 &= u(t_n + h/2, x_n + hk_2/2, y_n + hl_2/2) \\
l_3 &= v(t_n + h/2, x_n + hk_2/2, y_n + hl_2/2) \\
k_4 &= u(t_{n+1}, x_n + hk_3, y_n + hl_3) \\
l_4 &= v(t_{n+1}, x_n + hk_3, y_n + hl_3) \\
k &= \frac{k_1 + 2k_2 + 2k_3 + k_4}{6} \\
l &= \frac{l_1 + 2l_2 + 2l_3 + l_4}{6} \\
x_{n+1} &= x_n + hk \\
y_{n+1} &= y_n + hl
\end{aligned}
\tag{2.2}
$$

For the Runge Kutta Fehlberg method, we use a $4^{th}$ order Runge Kutta method and a $5^{th}$ order Runge Kutta method in combination to create an adaptive time step method.

The set up is shown in Equation 2.3. Similar to RK4 we have a set of $k_i$ values that represent weighted function evaluations between $t_n$ and $t_{n+1}$. Not shown are the $l_i$ equations that correspond to $v$. The expressions for $l_i$ is the same as for $k_i$ only it is a weighted function evaluation of $v$ (replace $u$ with $v$ and you get the $l_i$ values).

$$
\begin{aligned}
k_1 &= u(t_n, x_n, y_n) \\
k_2 &= u(t_n + \frac{h}{4}, x_n + \frac{k_1}{4}, y_n + \frac{l_1}{4}) \\
k_3 &= u(t_n + \frac{3h}{8}, x_n + \frac{3k_1}{32} + \frac{9k_2}{32}, y_n + \frac{3l_1}{32} + \frac{9l_2}{32}) \\
k_4 &= u(t_n + \frac{12h}{13}, x_n + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3, \\
&\qquad y_n + \frac{1932}{2197}l_1 - \frac{7200}{2197}l_2 + \frac{7296}{2197}l_3) \\
k_5 &= u(t_n + h, x_n + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4, \\
&\qquad y_n + \frac{439}{216}l_1 - 8l_2 + \frac{3680}{513}l_3 - \frac{845}{4104}l_4) \\
k_6 &= u(t_n + \frac{h}{2}, x_n - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5, \\
&\qquad y_n - \frac{8}{27}l_1 + 2l_2 - \frac{3544}{2565}l_3 + \frac{1859}{4104}l_4 - \frac{11}{40}l_5)
\end{aligned}
\tag{2.3}
$$

Let $(x_{n+1}^{[4]}, y_{n+1}^{[4]})$ be the solution to the $4^{th}$ order Runge Kutta at time step $n+1$ and $(x_{n+1}^{[5]}, y_{n+1}^{[5]})$ be the solution of the $5^{th}$ order Runge Kutta method at time step $n+1$. We calculate both solutions in Equation 2.4.

$$x_{n+1}^{[4]} = x_n + \left(\frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5\right) \tag{2.4a}$$

$$y_{n+1}^{[4]} = y_n + \left(\frac{25}{216}l_1 + \frac{1408}{2565}l_3 + \frac{2197}{4104}l_4 - \frac{1}{5}l_5\right) \tag{2.4b}$$

$$x_{n+1}^{[5]} = x_n + \left(\frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{9}{50}k_5 + \frac{2}{55}k_6\right) \tag{2.4c}$$

$$y_{n+1}^{[5]} = y_n + \left(\frac{16}{135}l_1 + \frac{6656}{12825}l_3 + \frac{28561}{56430}l_4 - \frac{9}{50}l_5 + \frac{2}{55}l_6\right) \tag{2.4d}$$

We then calculate the difference in solutions. Let's define $\epsilon_x \equiv |x_{n+1}^{[5]} - x_{n+1}^{[4]}|$ and $\epsilon_y \equiv |y_{n+1}^{[5]} - y_{n+1}^{[4]}|$. If either of these $\epsilon$ values are greater than some tolerance, $tol$, we must decrease the time step and implement Equations 2.3 and 2.4 again to get a more accurate value for $(x_{n+1}, y_{n+1})$.

The method for refining the time step is given in Equation 2.5. The power of 1/4 is due to the $4^{th}$ order accuracy of the least accurate of the two methods, RK4. The factor of 2 on the bottom of the fraction is usually used to ensure the new time step is small enough.

$$h_{new} = h_{old} \left( \frac{tol}{2 \left| y_{n+1}^{[5]} - y_{n+1}^{[4]} \right|} \right)^{1/4} \tag{2.5}$$

We may also like to be able to increase the time step if the $\epsilon_x$ and $\epsilon_y$ are both smaller than some value, $tol_{min}$. To do this, we can double the time step for the next iteration through the RKF method. This means that we accept the solution $(x_{n+1}, y_{n+1})$ of the 5th order Runge Kutta method (2.4$c$ and 2.4$d$) and use $h_{new} = 2h_{old}$ to calculate $(x_{n+2}, y_{n+2})$.

For RK4 we end up with a solution that is $O(h^4)$ accurate while we end up with a solution of $O(h^5)$ accurate for RKF because we use the $5^{th}$ order solution (after accepting the time step size for each iteration).

## 2.2 Part 2. Lagrangian Analysis

In this section, we will talk about two methods: one deterministic and the other probabalistic.

With the deterministic model we know the state of the particle given its initial condition and the velocity field. It requires much computation to obtain the final state of all of the particles.

With the probabalistic model, we may know the initial state of the system (distribution of particles) and we know the probability that some particle will end up in a certain location in the final state of the system, but we do not know for certain where any individual particle will be at that later time. This means we can predict what the system might look like at some later time without knowing exactly how individual particles will behave.

### 2.2.1 Deterministic method: Lagrangian descriptor

One way to analyze the lagrangian behavior of the Bay is the use the M Function described in [1], as shown in Equation 2.6. For this analysis, we will be using a two dimensional system (x and y). Therfore, Equation 2.6 becomes Equation 2.7, which is simply the distance a particle travels during some time $2\tau$.

$$\frac{d\vec{x}}{dt} = \vec{V}(\vec{x}, t) \tag{2.6a}$$

$$M(\vec{x}_0, t_0)_{\vec{V},\tau} = \int_{t_0-\tau}^{t_0+\tau} \left[ \sum_{i=1}^{n} \left( \frac{dx_i(t)}{dt} \right)^2 \right]^{1/2} dt \tag{2.6b}$$

$$M(\vec{x}_0, t_0)_{\vec{V},\tau} = \int_{t_0-\tau}^{t_0+\tau} \sqrt{u(t)^2 + v(t)^2} \quad dt \tag{2.7}$$

To calculate the distance each particle is traveling we initially set M = 0, and then at each iteration of the time integration we update M by adding the old value to M to the distance just traveled from $(x_n, y_n)$ to $(x_{n+1}, y_{n+1})$. This means that instead of updating just $x$ and $y$ at each iteration we are also updating this third variable $M$. The updating process is given in Equation 2.8.

$$x_{n+1} = \text{update x using either RK4 or RKF method} \tag{2.8a}$$

$$y_{n+1} = \text{update y using either RK4 or RKF method} \tag{2.8b}$$

$$M_{n+1} = M_n + \sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} \tag{2.8c}$$

The idea behind this equation is that we can compare the M value of a group of nearby particles to determine where the coherent structures are and where we would expect to find a manifold within our system. We do this by plotting the M values on some color scale, as a function of the initial condition of the corresponding trajectory.

Particles from a coherent set should appear to be of the same color, as we would expect them to be traveling roughly the same distance. Any place we see sharp changes in color indicates a manifold between two structures.

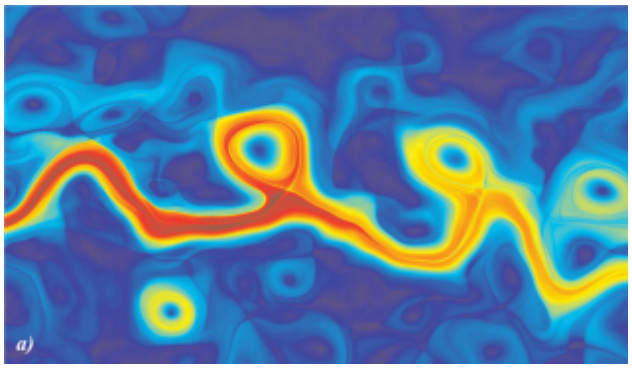


Figure 1: The M Function applied to the Kuroshio Current (May 2, 2003) with $\tau = 15$ days between longitudes $148°E - 168°E$ and latitudes $30°N - 41.5°N$. The color represents the total distance a particle traveled (plotted at the initial condition) with red being the greatest distance and blue the shortest distance. Same colored regions indicate regions of particles traveling approximately the same distance. The contrast between blue and red regions indicate a difference between two different coherent structures. It is the regions with sharp changes in color that we are interested in, as these are the regions we expect to find a manifold separating two dynamically different regions. [1]

Figure 1 shows the M Function being computed for the Kuroshio Current ($\tau = 15$ days), as an example of what we might expect to see from our own analysis. Relative to one another the red regions are the set of particles that traveled the farther and the particles in the blue regions traveled the shortest distance. The sharp change in color between a blue region and a red region indicates that there must be some manifold inbetween the two regions of particles.

### 2.2.2 Probabalistic method: Coherent set

Instead of the method described in §3.2.1 we might want to use a more quantitative method. One such method would be the probabalistic method proposed in [3] where we have some domain partitioned into different cells and we analyze the probabilities of particles from some cell $\alpha$ moving into cell $\beta$ over some time interval.

We first partition the domain at time $i$ and at time $j$, as in Figure 2. This image represents the initial domain $D_i$ (left) with a set of distributed initial conditions and the same domain $D_j$ (right) at some later time $j$.

We then take the 2D domain and transform it into matrix whose values correspond to the number of particles in each cell. Each cell initially contains approximately 100 particles. The Matrix then is transformed into a vector. This new vector is shown in Equation 2.9. This equation represents $D_iT = D_j$ where $D_i$ and $D_j$ are the domain at the initial time and the final time, respectively. T is a Transition matrix, whose elements $T_{\alpha\to\beta}$ represent the

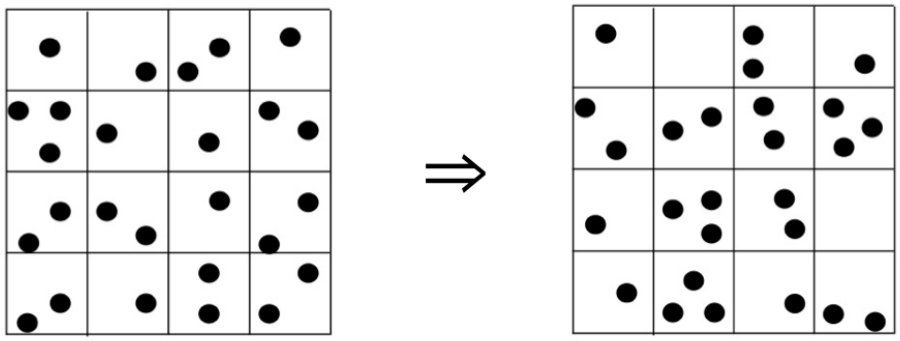

Figure 2: Here we see some initial domain, $D_i$ (left) and the same domain at some later time $D_j$ (right). We can turn these 2D domains into matrices whose values represent the number of particles in the given cell. This Matrix can then be transformed into a 1D vector. In Equation 2.9 we see $D_i$ being multiplied by some transition matrix T whose values represent the probability that a particle in some cell $\alpha$ will end up in cell $\beta$, which is equal to the final domain, $D_j$.

probability that a particle initially in cell $\alpha$ will end up in cell $\beta$.

$$\begin{pmatrix} a_i & b_i & \cdots & h_i & k_i \end{pmatrix} \begin{pmatrix} T_{a\to a} & T_{a\to b} & \cdots & T_{a\to k} \\ T_{b\to a} & T_{b\to b} & \cdots & T_{b\to k} \\ \vdots & \vdots & \ddots & \vdots \\ T_{k\to a} & T_{k\to b} & \cdots & T_{k\to k} \end{pmatrix} = \begin{pmatrix} a_j & b_j & \cdots & h_j & k_j \end{pmatrix} \tag{2.9}$$

We can calculate the transition matrix, $T$ relatively easily, as we have the trajectories of each initial condition and therefore we know the initial and final cell locations of each particle we initialized.

From this transition matrix, we want to compute the single value decomposition of the matrix to obtain the eigenvalues and eigenvectors of $T$. This will be done with Matlab's SVD command, which will compute the eigenvalues T. We know from the Perron-Frobenius Theorem that our transition matrix $T$ will have a largest eigenvalue of 1, where all other eigenvalues are less than 1. We can exploit this by using the largest eigenvalue (and its corresponding eigenvector) to reconstruct the dominating dynamics of the Chesapeake Bay (as is done for image reconstruction).

# 3 Validation

## 3.1 Part 1: Trajectory computation

### 3.1.1 Spatial Interpolation

To validate the interpolation methods, I will be applying my interpolation code to the known velocity field shown in Equation 3.1. [4]

$$\frac{dx}{dt} = -\frac{A\pi}{k} cos(\pi y)\left(sin(kx) + \epsilon k cos(\omega t) cos(kx)\right) \tag{3.1a}$$

$$\frac{dy}{dt} = A sin(\pi y)\left(cos(kx) - \epsilon k cos(\omega t) sin(kx)\right) \tag{3.1b}$$

This velocity field, given $A = 0.1$, $k = 1$, $\omega = 0.6$, and $\epsilon = 10$, exhibits chaotic behaviors, similar to what we might expect in the Bay. By sampling this function on an Arakawa C-grid we can verify that the interpolation methods developed in this project do indeed interpolate off grid velocity values. Using these functions we can also compare the accuracy of the different interpolation methods. This will provide a better understanding of the limitations of certain lower order interplation methods (bilinear) as compared to the higher order methods (bicubic).

### 3.1.2 Time Integration

To verify the trajectory calculations the results can be compared to those of well-known methods such as Matlab's built in ODE solver, ODE45. The ROMS database also calculates trajectories, which can be compared to the calculation of the trajectories from this project.

## 3.2 Part 2: Lagrangian Analysis

### 3.2.1 Deterministic method: Lagrangian descriptor

Validation of the deterministic method will be done by applying the method to a well-studied systems with known unstable and stable manifolds. One such system is the double-well Duffing equation, seen in Equation 3.2. [10]

$$\frac{d^2x}{dt^2} - x - x^3 = 0 \tag{3.2}$$

There is not much concern with debugging this piece of the code because we are just adding another variable into our time integration code. Once the time integration method

is working, we should not encounter much trouble with extending the code to suit our Deterministic Lagrangian descriptor method.

### 3.2.2 Probabalistic method: Coherent set

As mentioned in §3.2.1 we can verify this probabalistic method against a well studied method (possible the Duffing equation (Equation 3.2)). This system should lend itself to the verification of the probabalistic method because the eigenvectors for the fixed point are known.

We can also compare both methods to one another to verify that the methods agree.

# 4 Testing: Application to the Chesapeake Bay

## 4.1 Data set

The end goal of this project is the analyze the dynamics of the Chesapeake Bay. To do this, we need velocity data corresponding to the bay. For the purposes of this analysis we will be using the Regional Ocean Modeling System (ROMS) to generate a velocity field for some interval of time. This will give us a 3 dimensional set of discrete points. We will have two length dimensions ($x$ and $y$) as well as a third dimension in time ($t$).

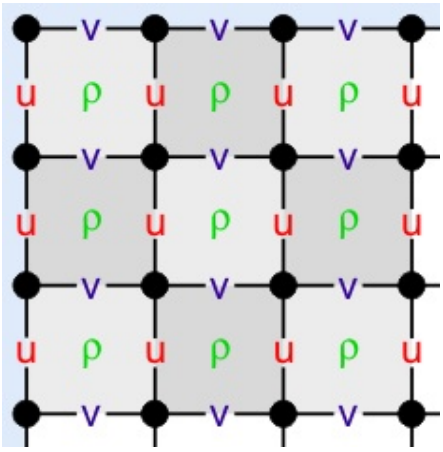


Figure 3: This grid is the Arakawa C-grid used by ROMS. We can see that the x-directed velocities are calculated along the left and right side panels of each grid cell while the y-directed velocities are calculated along the upper and lower faces of the grid cells.[9]

ROMS is a terrain-following primitive equations model for the ocean that will model the dynamics of the Bay well enough for the purposes of this analysis. The velocities produced

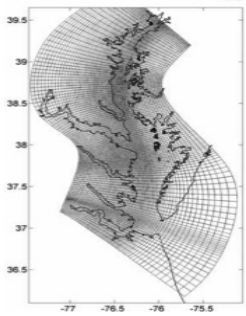


Figure 4: This is the ROMS grid shown on the Chesapeake Bay. This is the grid we will be working with for this project. *Image courtesy of the UMD ROMS group.*

are staggered using an Arakawa C-grid, as shown in Figure 3. Using the Arakawa C-grid is a more natural way to express the state variables in the context of solving the fluid flow equations within ROMS. The grid placed over the bay can be seen in Figure 4. [9]. *Note:* ROMS automatically sets all of the velocity values to zero on land.

## 4.2 Approach

Using the staggered grid (Figure 3) we will interpolate $u$ and then $v$, which are found separately, at each step of the time integration. The $u$ and $v$ grid regions must be chosen so that both $u$ and $v$ regions overlap where the point at which we wish to interpolate is in their intersection. Figure 5 demonstrates this overlap.

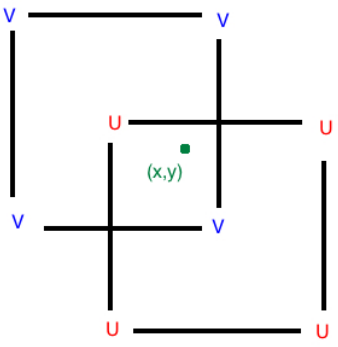


Figure 5: This grid is the Arakawa C-grid used by ROMS. If we want to interpolate the velocity field at some point $(x, y)$ then we must interpolate using the overlapping grid cells from $v$ and $u$.

# 5 Implementation

All algorithms will written in MATLAB on a MacBook Pro with a 2.3 GHz Intel Core i5 processor with 4 GB of RAM. All algorithms will initially designed to run in series. Time permitting, the algorithms for the interpolation and trajectory calculation will later be modified to run in parallel using MATLABs Parallel Computing Toolbox.

# 6 Deliverables

- Code that:
  - interpolates and calculates the trajectories of the ROMS data set
  - calculates and plots the M values from the M Function [1]
  - calculates the transition matrix and its eigenvectors and eigenvalues
- A comparison of the M Function analysis and the Probabalistic analysis
- The ROMS data set
- Proposal document and presentation
- Mid-year document and presentation
- Final report and presentation

# 7 Milestones

Below is the tentative schedule for the approach part of the project.

## 7.1 Part 1

- Develop code for Interpolation *(October - November)*
  - Develop bilinear method (no time interpolation) *(October)*
  - Develop bicubic method (no time interpolation) *(October - Early November)*
  - Develop the method for Lagrange Polynomial interpolation in time *(November)*
  - Validation of Interpolation methods *(October to Late November)*
  - Time permitting: Parallelization of the interpolation process
- Time integration methods *(Late November - December)*

  - $4^{th}$ oder Runge Kutta *(Late November)*
  - $5^{th}$ order Runge Kutta Fehlberg method *(Late November - Early December)*
  - Validation of the Trajectory (time integration) computation *(Late November - Early December)*

## 7.2 Part 2

- M Function analysis *(January - mid February)*
  - Modify time integration methods to incorporate calculation of the distance each particle travels *(January - mid February)*
  - Validation of this M Function *((January - mid February)*
- Probabalistic method *(Mid February - April)*
  - Set up indexing *(February)*
  - Solve system $D_iT = D_j$ for $T_{ij}$ Matrix *(Early March)*
  - Compute SVD of $T$ *(March)*
  - Reconstruct image of dominating dynamics *(Late March - Early April)*
  - Validate Probabalistic method using Duffing equation *(Late March - Early April)*
  - Compare Deterministic method and Probabalistic method *(Early April)*
  - Time permitting: Create my own SVD code

# References


[1] Mancho A. M., Mendoza C. Hidden Geometry of Ocean Flows, *Physical Review Letters*, 105(3) (2010).

[2] Shadden, S. C., Lekien F., Marsden J. E. "Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two- dimensional aperiodic flows". *Physica D: Nonlinear Phenomena*, 212, (2005) (34), 271304

[3] Froyland G., et al. Coherent sets for nonautonomous dynamical systems. *Physica D*, 239 (2010) 1527 1541.

[4] Mancho A. M., Small D., Wiggins S. A comparison of methods for interpolating chaotic flows from discrete velocity data. *Computers & Fluids*, 35 (2006), 416-428.

[5] Fessler F. A., "Chapter in 2D Interpolation", The University of Michigan, Ann Arbor, MI. <http://web.eecs.umich.edu/ fessler/course/556/l/n-07-interp.pdf>

[6] Lancaster D., "A Review of Some Image Pixel Interpolation Algorithms". Synergetics, Thatcher, AZ. 2007. <http://www.tinaja.com/glib/pixintpl.pdf>

[7] Shu X.. Bicubic Interpolation McMaster University, Canada. March 25th 2013. <http://www.ece.mcmaster.ca/ xwu/3sk3/interpolation.pdf>

[8] <http://www.math.montana.edu/ davis/Classes/MA442/Sp07/Notes/RKF_ABM.pdf>

[9] ROMS Wiki: Numerical Solution Technique. April 2012. Last visited: Sept. 22 2013 <https://www.myroms.org/wiki/index.php/Numerical_Solution_Technique>

[10] Alligood, K. T., Sauer T. D., and Yorke J. A. *Chaos.* Springer New York, 1996. (pp 406-407)

[11] Hermans, D."Runge-Kutta-Fehlberg Method" University of Birmingham, UK. Jan. 10th, 2002. <http://web.mat.bham.ac.uk/D.F.M.Hermans/msmxg6/ln/lnotes176.html>