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

May 12, 2014

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. Given the model-based discrete velocity data of the bay, obtained through the use of the Regional Ocean Modeling System (ROMS), we implement bilinear and bicubic spatial interpolation methods and a 3^{rd} order lagrangian polynomial to interpolate in time. This interpolation 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. From these trajectories, we apply both a deterministic method and a probabilistic method to identify coherent structures in the flow field. We validate interpolation methods against analytic functions and integration methods against ordinary differential equations with analytic solutions. The deterministic method is validated using the Duffing equation while the probabilistic method is validated using results from [7]. Lastly we apply our methods to a modeled flow field of the Chesapeake Bay and compare the results of the two methods.

Contents

1 Introduction

Studying the dynamics of 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 every living thing 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 said to be part of a coherent set and different coherent sets are separated by what we will call manifolds[2]. Examples of these coherent sets include 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, analogous to observational data, 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 spatial and temporal interpolation methods.

In §2 we discuss the approach and algorithms used to interpolate velocity values both spatially and temporally as well as how we then integrate those velocity values with respect to time to obtain trajectories. It is also in this section that we describe the methods used to identify coherent structures within the Chesapeake Bay. The implementation is briefly discussed in §3. Validation of all of the methods is discussed in §4. This section contains the approach as well as the results of validation. We then apply the validated methods to the Chesapeake Bay data set to obtain the results given in §5. The data set itself is also discussed in section §5. In §6 we conclude the project by comparing and discussing the methods.

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 $\frac{dy}{dt} = v(x, y, t)$. The velocity values u and v are given on a grid at discrete times and therefore any time integration of $\frac{dx}{dt}$ and $\frac{dy}{dt}$ requires that we be able to interpolate u and v, both spatially and temporily at any (x, y, t) within the data set's spatial domain and time interval.

The second part consists of implementing lagrangian analysis methods based on the trajectories calculated in part 1.

2.1 Trajectory computation

We will see in this section that in order to calculate trajectories we need to integrate velocity along them in time. For our data this will also require interpolation in both time and space. We discuss these three components of trajectory calculation in reverse order (*i.e.* spatial interpolation, time interpolation, and integration). This build up allows us to see the connections between all three.

Because Lagrangian analysis requires computation of a large number of trajectories, computational efficiency is as an important factor as accuracy in selecting the method for the trajectory computation.

2.1.1 Spatial Interpolation

Given a velocity field given as a data set that is discrete in space and time we need to be able to interpolate any off-grid velocity in order to properly calculate trajectories. To do this, we implement a bilinear spatial interpolation method, as well as a bicubic spatial interpolation method. The bicubic method is the more accurate of the two but we also see a tradeoff in efficiency, which will be discussed. We also have a time dimension which will be interpolated using a 3^{rd} order Lagrange polynomial in time. The interpolation of u and v 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 (Equation 2.1) to approximate $u(x, y)$ (at a given constant time) within the grid cell created by the 4 nearest points[8][10].

$$u(x, y) = a_0 + a_1 x + a_2 y + a_3 xy \tag{2.1}$$

$$\begin{pmatrix} 1 & x_i & y_j & x_i * y_j \\ 1 & x_{i+1} & y_j & x_{i+1} * y_j \\ 1 & x_i & y_{j+1} & x_i * y_{j+1} \\ 1 & x_{i+1} & y_{j+1} & x_{i+1} * y_{j+1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} u(x_i, y_j) \\ u(x_{i+1}, y_j) \\ u(x_i, y_{j+1}) \\ u(x_{i+1}, y_{j+1}) \end{pmatrix} \tag{2.2}$$

To create that surface, we solve Equation 2.2 for the coefficients (a values) and then evaluate $u(x, y)$ at our point (x, y) on the surface.

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 [10][11]. 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. This interpoltion creates the surface in Equation 2.3 where the 16 b_{ij} coefficients need to be determined.

$$\begin{aligned} u(x, y) = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + b_{02}y^2 + b_{21}x^2y + b_{12}xy^2 + b_{22}x^2y^2 \\ + b_{30}x^3 + b_{03}y^3 + b_{31}x^3y + b_{13}xy^3 + b_{32}x^3y^2 + b_{23}x^2y^3 + b_{33}x^3y^3 \end{aligned} \tag{2.3}$$

To determine these 16 b_{ij} values we need to solve a 16 by 16 system of equations, determined by the velocity values and the 3 derivatives at each of the 4 nearest points, where the derivative and cross derivatives are approximated with second order central difference schemes (Equation 2.4).

$$\begin{aligned} \frac{\partial u}{\partial x} &= \frac{u(x_{i+1}, y_j) - u(x_{i-1}, y_j)}{2\Delta x} \\ \frac{\partial u}{\partial y} &= \frac{u(x_i, y_{j+1}) - u(x_i, y_{j-1})}{2\Delta y} \\ \frac{\partial^2 u}{\partial x \partial y} &= \frac{u(x_{i+1}, y_{j+1}) - u(x_{i+1}, y_{j-1}) - u(x_{i-1}, y_{j+1}) + u(x_{i-1}, y_{j-1})}{4\Delta x \Delta y} \end{aligned} \tag{2.4}$$

Sovling the linear system $A\vec{x} = b$ is $O(n^3)$. $n_1 = 4$ for bilinear interpolation and $n_2 = 16$ for bicubic. This is equivalient to saying that $n_2 = 4n_1$ so we expect bicubic interpolation to take $\frac{O([4n_1]^3)}{O([n_1]^3)}$ or ~ 64 times longer than bilinear interpolation. The comparison of the two methods is given in §3.2.

In the case that our point (x, y) is not within our x, y, and t bounds we cannot interpolate, for either of the two spatial interpolation methods. In this situation our code is to use NaN as the interpolated velocities values. In the case of the bicubic interpolation, we must alter our derivative equations for the boundary of our domain because our centered approximation can no longer be used. In the case of these boundary locations we calculate the derivatives using a 1^{st}-order forward or backward approximation to the derivatives and cross derivatives, based on which boundary edge we are concerned with. For example if we are concerned with the point (x_f, y_j), where $j = 2, 3, ..., f-1$, the first derivative with respect to x would be approximated as $\frac{\partial u}{\partial x} = \frac{u(x_f, y_j) - u(x_{f-1}, y_j)}{\Delta x}$ which is a 1^{st}-order backward approximation.

For the purposes of the analysis of the Chesapeake Bay data we will be using bilinear interpolation. The use of bilinear (instead of bicubic) is preferred due to the computational expense of bicubic. We expect a difference in computational time simply by observing that we need 4 times as many function and function derivative values for bicubic as we do for bilinear.

2.1.2 Time interpolation

Time interpolation is done using 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}.

$$\begin{aligned} u(t) = &\frac{(t-t_i)(t-t_{i+1})(t-t_{i+2})}{(t_{i-1}-t_i)(t_{i-1}-t_{i+1})(t_{i-1}-t_{i+2})} u(t_{i-1}) \\ &+ \frac{(t-t_{i-1})(t-t_{i+1})(t-t_{i+2})}{(t_i-t_{i-1})(t_i-t_{i+1})(t_i-t_{i+2})} u(t_i) \\ &+ \frac{(t-t_{i-1})(t-t_i)(t-t_{i+2})}{(t_{i+1}-t_{i-1})(t_{i+1}-t_i)(t_{i+1}-t_{i+2})} u(t_{i+1}) \\ &+ \frac{(t-t_{i-1})(t-t_i)(t-t_{i+1})}{(t_{i+2}-t_{i-1})(t_{i+2}-t_i)(t_{i+2}-t_{i+1})} u(t_{i+2}) \end{aligned} \tag{2.5}$$

In the event that our time value is between $t_0$and t_1 we use the interpolating polynomial that goes through t_0, t_1, t_2, and t_3. Similarly, if t_f is the final value of time at which we have velocity values, then for some time in the interval $[t_{f-1}, t_f]$ we interpolate using the polynomial that goes through the times $t_{f-3}, t_{f-2}, t_{f-1}$, and t_f.

The integration of any trajectory that goes outside of the domain of our data set will be haulted and that particular point is not considered in the final analysis. Most points stay within the bounds of our domain and so this does not affect our analysis significantly.

2.1.3 Time integration

To calculate the trajectories, we use two methods of the temporal integration. First we use a simple 4^{th}-order Runge Kutta method (RK4) (Equation 2.6) and then secondly the Runge Kutta Fehlberg method (RKF) (Equations 2.7 through 2.9). The RKF is a 5^{th}-order method. [9][12]

For the RK4 method the $\vec{k_i}$ term in Equation 2.6 is a 1 by 2 vector for x and y. $V(1) = u$ and $V(2) = v$ calculated at times in the interval $[t_n, t_{n+1}]$. After being calculated we then use a weighted average of these $\vec{k_i}$ values to determine the final value of (x_{n+1}, y_{n+1}). dt is the fixed time step used. It should be noted that in the code these $\vec{k_i}$ terms are in fact matrices of dimension n by 2 where n is the number of trajectories calculated. This is done to take advantage of MATLAB's efficient matrix and vector arithmetic.

$$\begin{aligned}
\vec{k}_1 &= \vec{V}(t_n, \vec{x}_n) \\
\vec{k}_2 &= \vec{V}(t_n + dt/2, \vec{x}_n + dt\vec{k}_1/2) \\
\vec{k}_3 &= \vec{V}(t_n + dt/2, \vec{x}_n + dt\vec{k}_2/2) \\
\vec{k}_4 &= \vec{V}(t_{n+1}, \vec{x}_n + dt\vec{k}_3) \\
\vec{k} &= \frac{\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4}{6} \\
\vec{x}_{n+1} &= \vec{x}_n + \vec{k}dt
\end{aligned} \tag{2.6}$$

For the RKF 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 2.7. Similar to RK4 we have a set of values that represent weighted function evaluations between t_n and t_{n+1}.

$$
\begin{aligned}
\vec{k}_1 &= u(t_n, \vec{x}_n) \\
\vec{k}_2 &= u(t_n + \frac{dt}{4}, \vec{x}_n + \frac{\vec{k}_1}{4}) \\
\vec{k}_3 &= u(t_n + \frac{3dt}{8}, \vec{x}_n + \frac{3\vec{k}_1}{32} + \frac{9\vec{k}_2}{32}) \\
\vec{k}_4 &= u(t_n + \frac{12dt}{13}, \vec{x}_n + \frac{1932}{2197}\vec{k}_1 - \frac{7200}{2197}\vec{k}_2 + \frac{7296}{2197}\vec{k}_3) \\
\vec{k}_5 &= u(t_n + dt, \vec{x}_n + \frac{439}{216}\vec{k}_1 - 8k_2 + \frac{3680}{513}\vec{k}_3 - \frac{845}{4104}\vec{k}_4) \\
\vec{k}_6 &= u(t_n + \frac{dt}{2}, \vec{x}_n - \frac{8}{27}\vec{k}_1 + 2\vec{k}_2 - \frac{3544}{2565}\vec{k}_3 + \frac{1859}{4104}\vec{k}_4 - \frac{11}{40}\vec{k}_5)
\end{aligned}
\tag{2.7}
$$

The adaptive time step dt_{new} is computed from the previous time step dt_{old} as follows. Let $\vec{x}^{[4]}_{n+1}$ be the solution to the 4^{th}-order solution produced by RKF at time step $n+1$ and $\vec{x}^{[5]}_{n+1}$ be the 5^{th}-order solution of RKF at time step $n+1$. We calculate both solutions in Equation 2.8.

$$
\vec{x}^{[4]}_{n+1} = \vec{x}_n + \left(\frac{25}{216}\vec{k}_1 + \frac{1408}{2565}\vec{k}_3 + \frac{2197}{4104}\vec{k}_4 - \frac{1}{5}\vec{k}_5 \right) \tag{2.8a}
$$

$$
\vec{x}^{[5]}_{n+1} = \vec{x}_n + \left(\frac{16}{135}\vec{k}_1 + \frac{6656}{12825}\vec{k}_3 + \frac{28561}{56430}\vec{k}_4 - \frac{9}{50}\vec{k}_5 + \frac{2}{55}\vec{k}_6 \right) \tag{2.8b}
$$

We then calculate the difference in solutions, $\epsilon \equiv |\vec{x}^{[5]}_{n+1} - \vec{x}^{[4]}_{n+1}|$. If the maximum component of ϵ is greater than some tolerance, tol, we must decrease the time step to dt_{new} and implement Equations 2.7 and 2.8 again to get a more accurate value for (x_{n+1}, y_{n+1}). The method for decreasing the time step is

$$
dt_{new} = dt_{old} \left(\frac{tol}{2\epsilon} \right)^{1/4} \tag{2.9}
$$

The power of $1/4$ is due to the 4^{th} order accuracy of the least accurate of the two solutions (4^{th} order). The factor of 2 in the denominator is usually used to ensure the new time step is small enough. We may also like to be able to increase the time step if both the x and y components of ϵ are 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 5^{th} order Runge Kutta method ($2.4c$ and $2.4d$) and use Equation 2.10 to

calculate (x_{n+2}, y_{n+2}).

$$dt_{new} = 2dt_{old} \tag{2.10}$$

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

2.2 Lagrangian Analysis

In this section, we discuss two methods: one deterministic and the other probabilistic. With the deterministic model we calculate the state of the particle at every time step given its initial condition and the velocity field. It requires much computation to obtain the final state of all of the particles. With the probabilistic model we require the only the initial distribution and the final distribution of particles. Using this fact we determine the probability of a particle moving from one small sub-domain to another sub-domain. It is these probabilities that allow us to analyze the system.

2.2.1 Deterministic method

One way to analyze the lagrangian behavior of the bay is the use the M function described in [1], as shown in Equation 2.12. We integrate this differential equation along with x and y from $t_0 - \tau$ to $t_0 + \tau$. The backward integration (from t_0 to $t_0 - \tau$) yields the unstable manifold while the forward integration (t_0 to $t_0 + \tau$) yields the stable manifold. We must choose τ such that the system has had sufficient time to exibit these manifolds.

$$\frac{d\vec{x}}{dt} = \vec{V}(\vec{x}, t) \tag{2.11}$$

$$\frac{d}{dt} M(\vec{x}, t)_\tau = \left[\sum_{i=1}^{n} \left(\frac{dx_i(t)}{dt} \right)^2 \right]^{1/2} \tag{2.12}$$

To calculate the M value, or the distance each particle has traveled in time= 2τ, we initially set $M = 0$ and then integrate Equation 2.12 along with Equation 2.11 using either

RK4 or RKF.

The idea behind this method 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.

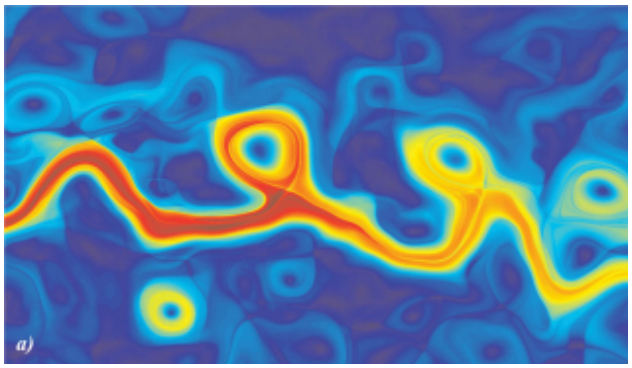


Figure 1: The M Function applied to the Kuroshio Current (May 2, 2003) with $\tau = 15$ days between longitudes $148°E - 168°E$ and latitudes $30°N - 41.5°N$. The color represents the total distance a particle traveled (plotted at the initial condition) with red being the greatest distance and blue the 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) in [1], 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 Probabilistic method

Instead of the method described in §2.2.1 we might want to use a more probabilistic method. One such method would be the method proposed in [3][5] where we have some domain partitioned into different cells and we analyze the probabilities of particles from some cell α

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

moving into cell β over some time interval in the corresponding domain.

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. In our analysis these initial conditions are uniformly distributed, rather than randomly distributed.

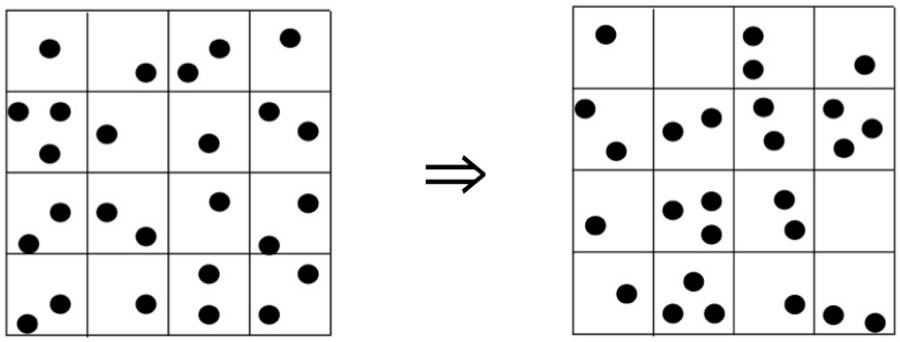


Figure 2: Here we see some initial domain, D_i (left) and the same domain at some later time D_j (right). We can turn these 2D domains into matrices whose values represent the number of particles in the given cell. This Matrix can then be transformed into a 1D vector. In Equation 2.13 we see D_i being multiplied by some transition matrix T whose values represent the probability that a particle in some cell α will end up in cell β, which is equal to the final domain, D_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 2.13. 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 probability that a particle initially in cell α will end up in cell β. Each row of T adds up to 1, making T a stochastic matrix.

$$\begin{pmatrix} a_i & b_i & \cdots & h_i & k_i \end{pmatrix} \begin{pmatrix} T_{a\to a} & T_{a\to b} & \cdots & T_{a\to k} \\ T_{b\to a} & T_{b\to b} & \cdots & T_{b\to k} \\ \vdots & \vdots & \ddots & \vdots \\ T_{k\to a} & T_{k\to b} & \cdots & T_{k\to k} \end{pmatrix} = \begin{pmatrix} a_j & b_j & \cdots & h_j & k_j \end{pmatrix} \tag{2.13}$$

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 eigenvalues and eigenvectors of T. This is done with Matlab's eigs command, which computes the eigenvalues of 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(s) (and the corresponding eigenvector(s)) to identify the dominating dynamics of the Chesapeake Bay. In reality some points in the domain will leave the domain and will therefore not be included in the analysis. This means we may end up with a few rows of zeros and the largest eigenvalue may be less than, and not equal to 1.

3 Implementation

All algorithms are written in MATLAB on a MacBook Pro with a 2.3 GHz Intel Core i5 processor with 4 GB of RAM. All algorithms are designed to run in series. The algorithms for the interpolation and trajectory calculation did not benefit from being modified to run in parallel using MATLABs Parallel Computing Toolbox.

4 Validation

We validate every step in the trajectory calculation as well as both lagrangian analysis methods. We validate the interpolation by taking a system of ODEs for which we have a solution and comparing interpolated velocity values with analytic velocity values. The time integration method is validated on a system of ODEs, of which we have an exact solution.

The deterministic method is validated using the same system of ODEs as is used to validate the time integration, of which we have an analytic expression for the M values. We also validate using the Duffing equation which has known stable and unstable manifolds. The probabilistic method is validated using a system of ODEs for which we have the solution to compare [7].

4.1 Trajectory computation

4.1.1 Spatial Interpolation

To validate the interpolation methods, we apply the interpolation code to the known velocity field shown in Equation 4.1. [4]

$$\frac{dx}{dt} = -\frac{A\pi}{k} cos(\pi y)\left(sin(kx) + \epsilon k cos(\omega t) cos(kx)\right) \tag{4.1a}$$

$$\frac{dy}{dt} = A sin(\pi y)\left(cos(kx) - \epsilon k cos(\omega t) sin(kx)\right) \tag{4.1b}$$

This velocity field, given $A = 0.1$, $k = 1$, $\omega = 0.6$, and $\epsilon = 10$, is regular, though nonlinear in space and time. Trajectories are chaotic as in the Chesapeake Bay data set. By sampling this function on a uniform grid we can verify that the interpolation methods developed in this project do indeed interpolate off grid velocity values. Using these functions we compare the accuracy of the different interpolation methods. This provides a better understanding of the limitations of certain lower order interplation methods (bilinear) as compared to the higher order methods (bicubic) as well as the limitations of the Lagrange polynomial time interpolation.

An example of a interpolated $u(x, y)$ surfaces is shown in Figure 3. The time dependence of the velocity is dealt with by setting $t = 1.0$. This plot shows the velocity ($\frac{dx}{dt}$) given in Equation 4.1a interpolated using the Bilinear interpolation method. The blue grid points are uniformly distributed true values of the velocity where $dx = dy = 0.02$. On the same surface is 10,000 randomly chosen (x, y) pairs in red at which the velocity was interpolated. The light blue (cyan) dots that appear to be on the $u = 0$ surface are the error ($u_{interpolated} - u_{exact}$) values of the interpolated velocities. This provides us with at least a visual confirmation that the function is indeed interpolating the velocites properly.

Figure 4 shows the L_1, L_2, and L_∞ norms of the absolute error for bilinear interpolation. The norms are plotted on a loglog plot to show the relationship between the error and stepsize. The slope of each line corresponds to b in the equation $error = a * (\Delta x)^b$. The slope values for all of the error plots are given in Table 1.

We see that the L_1 norm, with a slope of 2.07, closely follows the $O(\Delta x^2)$ line, indicating that bilinear interpolation is a second order accuracy method. In addition, the L_∞ norm, with a slope of 1.07, follows the $O(\Delta x)$ line, confirming convergence. For this plot $t = \pi$, dx and dy changed together at the same rate (neither was held constant).

Figure 5 shows the error associated with bicubic interpolation, as a function of spatial stepsize dx and dy. The figure shows the the mean error (blue) and maximum error (green) plotted on a loglog plot alongside the Δx^4 line (red). This is the mean and maximum errors of 50,000 randomly chosen points in the unit square $[0, 1] \times [0, 1]$. Both errors exhibit an $O(\Delta x^4)$ behavior, with slopes of 4.02 and 3.96 respectively. This indicates that this is a 4^{th} order method.

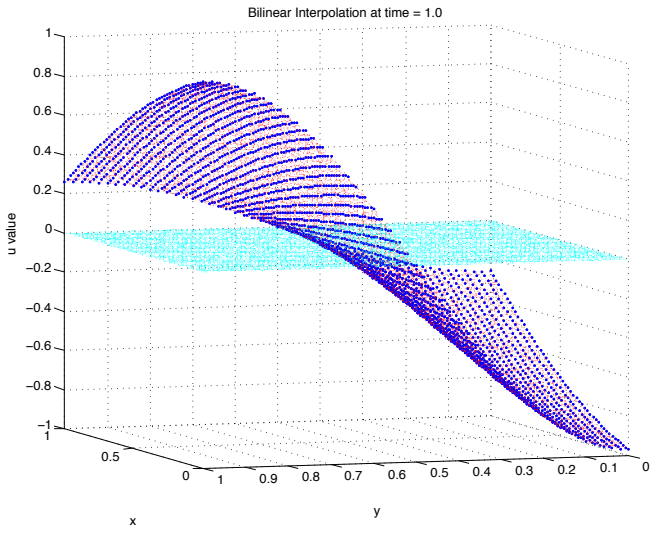


Figure 3: A plot showing the velocity given in Equation 3.1a interpolated by the bilinear method. The blue dots are the uniformly sampled grid (data) and the red dots that follow the same surface are the 10,000 randomly chose (x, y) pairs at which u was interpolated. The light blue (cyan) dots are the error values for each of the red interpolated values. This allows us to visualize the magnitude of the error for such a surface. For this interpolation, $t = 1.0$ and $dx = dy = 0.02$. This plot visually confirms that the method is indeed working properly.

A comparison of the two interpolation methods (bilinear and bicubic) is given in Figure 6, showing the L_1 norm of the error for both methods. This demonstrates that the bicubic method is more accurate than the bilinear method, as we expected.

The timing comparison of the two methods is given in Figure 7. It is in this figure that we see the cost of the increase in accuracy. The top panel shows the timing of both methods as a function of n, the number of interpolated values. We see that to interpolate 100,000 velocities it takes the bicubic method ≈ 27 seconds and the bilinear method ≈ 0.5 seconds. The ratio of bicubic timing to blinear timing is given in the bottom panel. For all n this ratio is approximately 58 to 60. This agrees with our estimate from §2.1.1 which predicted a ratio of approximately 64.

Figure 8 shows the errors associated with the time interpolation using bilinear spatial interpolation. We see in the top panel that the error of the interpolation is a function of dt. dx and dy also change at the same rate as dt in this top panel. In the bottom panel it is only dt that changes. In the bottom panel the error is approximately constant around 0.02 and is associated with $dx = dy = 0.1$. The constant error of 0.0003 is associated with $dx = dy = 0.01$. This shows that for constant spatial step size, the error doesn't change. It is only for the top plot when the spatial step size is changed along with the time step size that we see a decrease in the error. This suggests that the error due to the time interpolation

method (Figure)	norm/error type	slope value
Bilinear	1 - norm	2.0655
(Figure 4)	2 - norm	1.4912
	∞ - norm	1.0659
Bicubic	1 - norm	4.0076
(Figure 5)	∞ - norm	3.9480
Bilinear (Figure 6)	mean error	2.0025
Bicubic (Figure 6)	mean error	4.0060
Time Interpolation	With changing dx,dy	1.9804
(Figure 8)	With constant dx,dy	ranges from -0.1 to 0.1
RK4 (Figure 10)	mean of the maximum error in y	3.9915
RKF5 (Figure 10)	mean of the maximum error in y	5.0013

Table 1: This table gives the slope values of the error plots shown in Figure 4, Figure 5, Figure 6 , Figure 8, and Figure 10. All slopes were calculated using a least squares linear fit to the data.

is smaller than the error that arrises from the use of spatial interpolation.

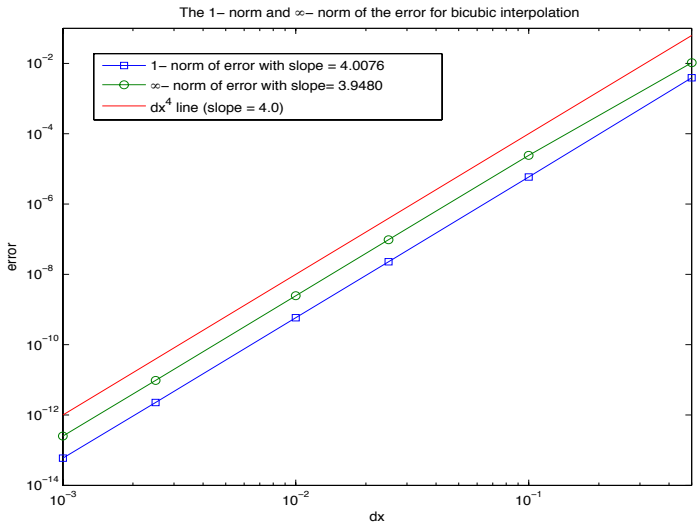


Figure 5: The 1-norm of the absolute error of 50,000 interpolated points (blue squares) with a slope of 4.01 and the ∞-norm error (green circles) with a slope of 3.95 are shown in this figure. For comparison the $y = dx^4$ line is also given (red). This demonstrates that the bicubic interpolation is a 4^{th} order method. Both slopes were calculated using a least squares linear fit.

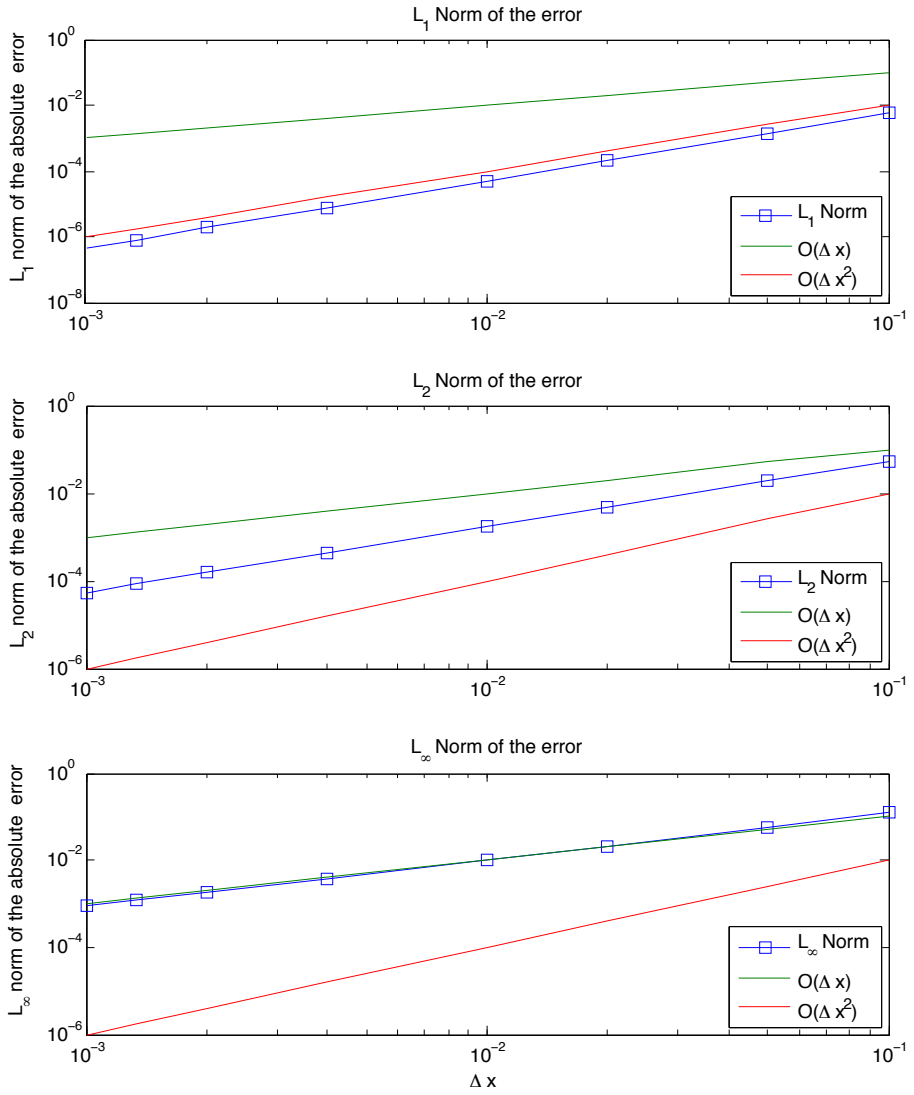


Figure 4: This plot shows dependence of the L_1, L_2 and L_∞ norms of the absolute error on the spatial step size (dx and dy). The top plot shows the L_1 norm to be of order Δx^2. This tells us that this is a second order accuracy method. The L_∞ norm follows the $O(\Delta x)$ line, confirming convergence. For this plot $t = \pi$, dx and dy changed together (neither was held constant). Slopes of the least squares linear fit to each line can be found in Table 1.

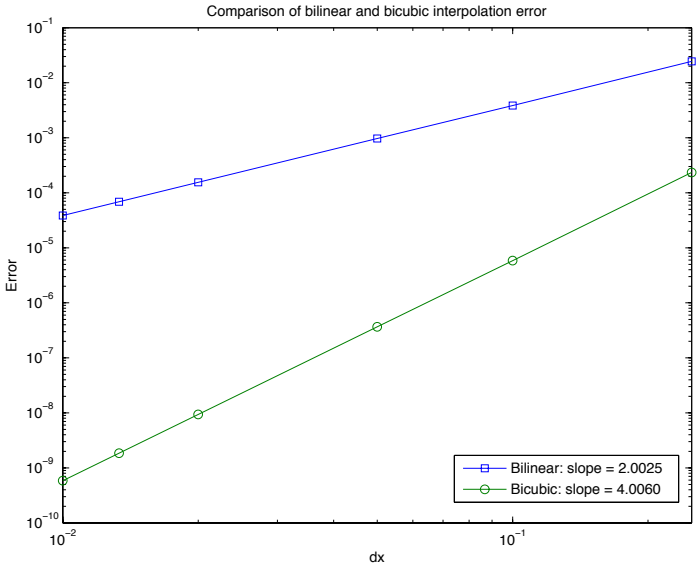


Figure 6: For 50,000 interpolated points, the mean absolute error is shown as a function of spatial step size, dx for both the bilinear method and the bicubic method. Bilinear interpolation is a 2^{nd} order method (slope = 2.0025) and bicubic interpolation is a 4^{th} order method (slope = 4.0060). dy changes with dx.

4.1.2 Time Integration

To verify the time integration methods we calculate the trajectories of the system

$$\begin{aligned} \frac{dx}{dt} &= -y \\ \frac{dy}{dt} &= x \end{aligned} \tag{4.2}$$

where the solution with (x_0, y_0) at $t = 0$ is given by Equation 4.3.

$$\begin{aligned} x(t) &= -y_0 sin(t) + x_0 cos(t) \\ y(t) &= y_0 cos(t) + x_0 sin(t) \end{aligned} \tag{4.3}$$

We do this while using the interpolation methods. Because this system of differential equations is linear, we will have an exact interpolation.

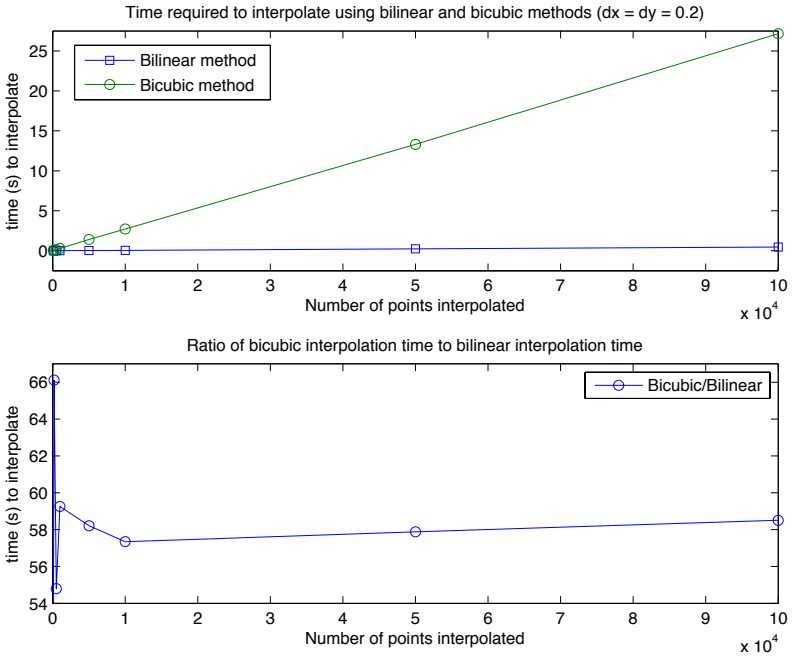


Figure 7: These two plots are a comparison of the time required to interpolate n number of trajectories (x axis) for both the bilinear and bicubic methods in the top panel. In the bottom panel we see the ratio $\frac{\text{time to interpolate with bicubic}}{\text{time to interpolate with bilinear}}$ plotted against the number of trajectories interpolated. For 10^5 trajectories it takes bicubic about 59 times as long as it takes bilinear.

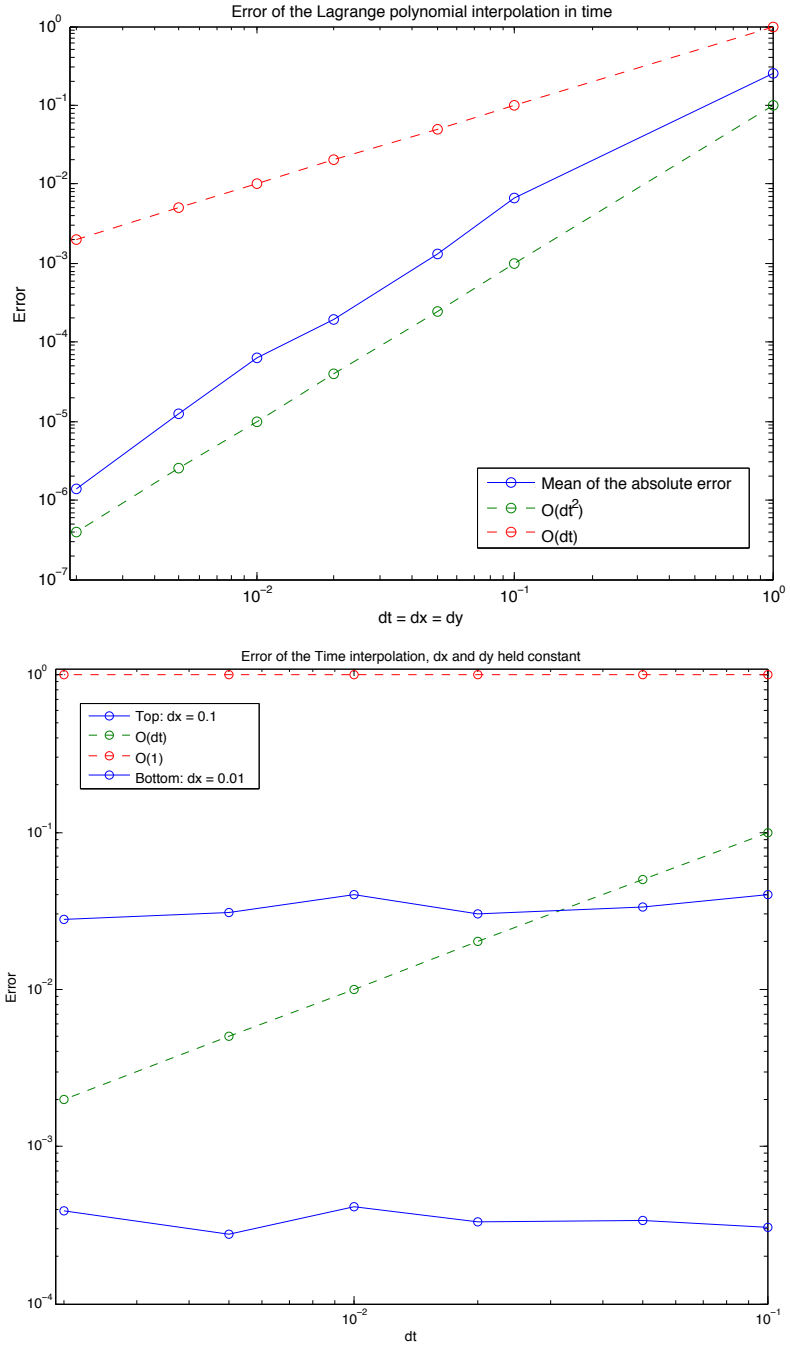


Figure 8: We see in the top panel that the error of the interpolation as a function of dt. dx and dy also change at the same rate as dt in this top panel. In the bottom panel it is only dt that changes. In the bottom panel the error of approximately 0.02 is associated with $dx = dy = 0.1$ while the error of approximately 0.0003 is associated with $dx = dy = 0.01$. This shows that for constant spatial step size, the error doesn't change, indicating that the error from the time interpolation is smaller than that of the bilinear interpolation.

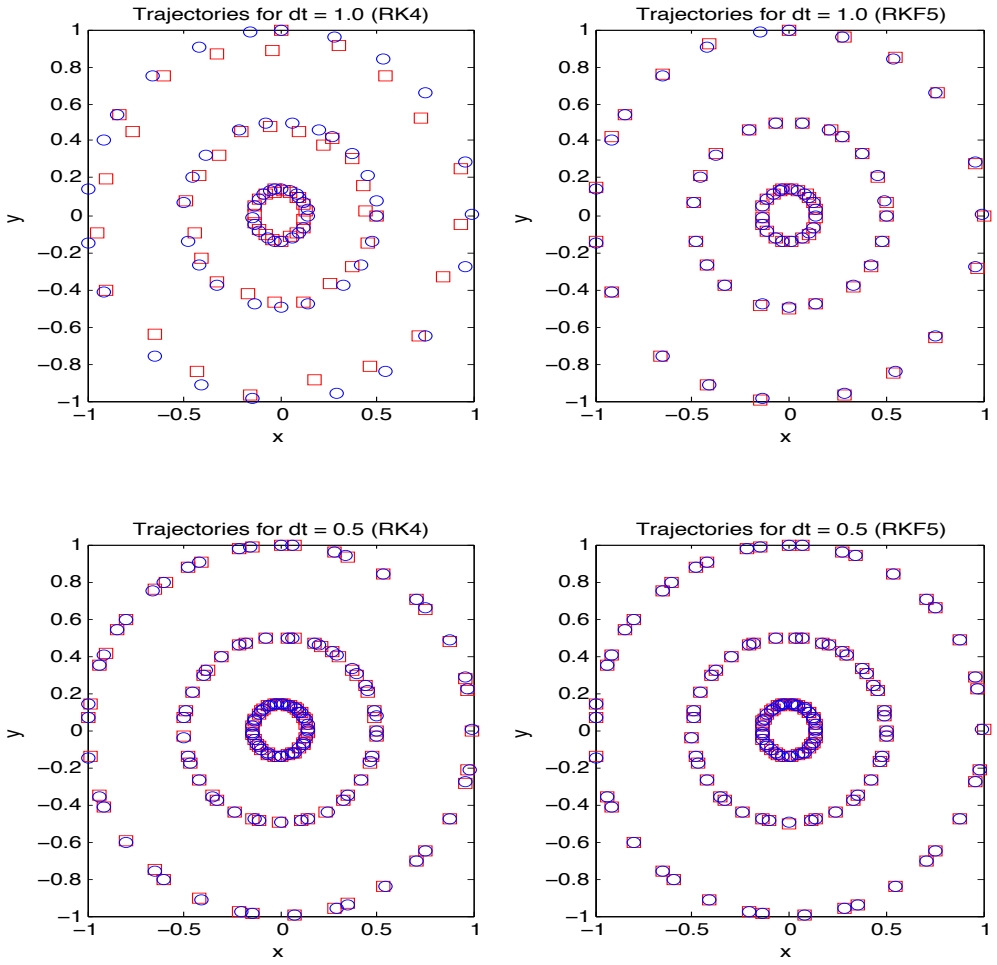


Figure 9: Trajectories calculated for both RK4 (left panels) and the RKF (right panels) methods from $t = 0$ to $t = 20$. $dx = dy = 0.1$ and $dt_{grid} = 0.5$. The true trajectories are marked by blue circles and the numerical trajectory is marked by red squares. The top panels are trajectories where $dt = 1.0$ and the bottom are where $dt = 0.1$. Trajectories for RKF are visibly better than those of RK4 for the same time step size, as is expected.

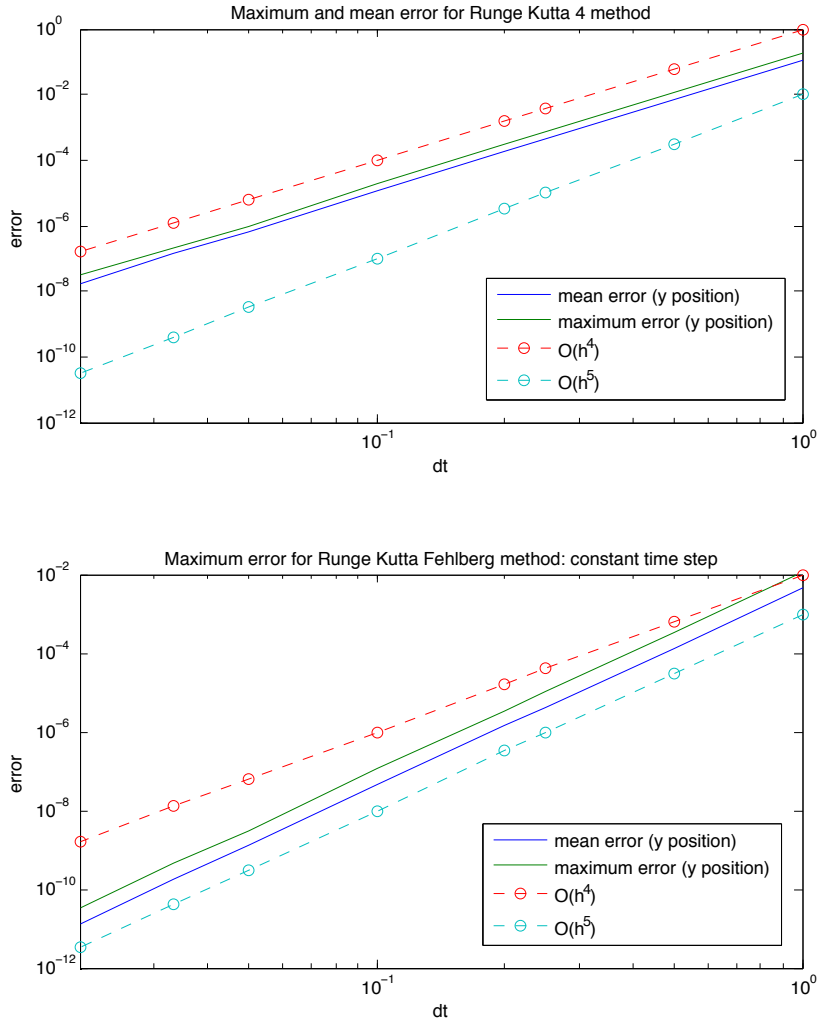


Figure 10: For 20 trajectories, the maximum error of all trajectories is plotting, alongside the average of the maximum of each trajectory. Parameters are the same as those in Figure 9. RKF is not time adaptive for these plots, dt is fixed to allow for a proper comparison. For RK4 (top panel) both lines follow the $O(dt^4)$ while for RKF (bottom panel) follows the $O(dt^5)$ line. This confirms that the RK4 is a forth order approximation while the RKF method is a fifth order approximation. In this plot h refers to the time step of the time integration.

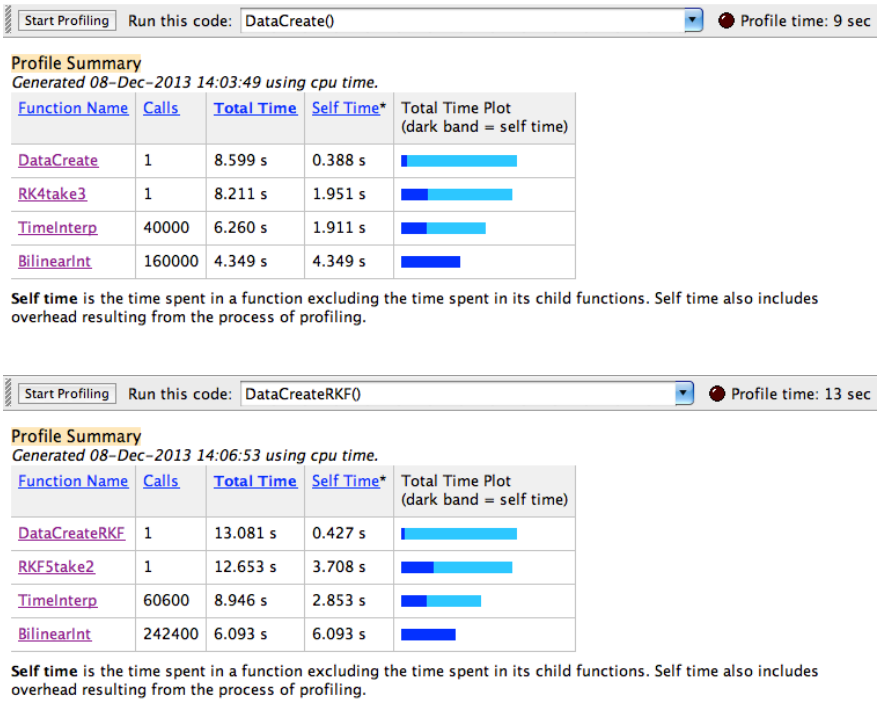


Figure 11: Image of the MATLAB profiler for RK4 (top panel) and RKF (bottom panel). Both runs are done for 50 trajectories. For both integration methods, approximately 75% of the time is spent in the time interpolation function, TimeInterp.m, or the bilinear interpolation function, BilinearInt.m.

Figure 9 shows trajectories calculated for both the 4^{th}-order Runge Kutta method (left panels) and the Runge Kutta Fehlberg method (right panels) from $t = 0$ to $t = 20$. $dx = dy = 0.1$ and $dt_{grid} = 0.5$. The true trajectories are marked by blue circles and the numerical trajectory is marked by red squares. The top panels are trajectories where $dt = 1.0$ and the bottom are where $dt = 0.1$. It is clear that for $dt = 1.0$ (Top left panel) RK4 is not sufficient for computing the solution but the accuracy becomes better for smaller dt (Bottom left panel). For RKF, it is clear that while the solution for $dt = 0.1$ (Top right panel) is better than that of RK4, we also can detect an improvement for $dt = 0.1$ (Bottom right panel).

Figure 10 shows the error of both time integration methods. From these plots it is clear that the RK4 method is a forth order method and RKF is a fifth order method, as was expected. The slopes are given in Table 1.

Figure 11 shows an example of the MATLAB profiler applied to the RK4 method (top panel) and the RKF method (bottom panel). Both runs are done for 50 trajectories. For both integration methods, approximately 75% of the time is spent in the time interpolation

function, TimeInterp.m, or the bilinear interpolation function, BilinearInt.m. For RKF we recall that we calculate a 4^{th} order solution and a 5^{th} order solution using 6 function evaluations (or time interpolations in our work) instead of 4 for the 4^{th} order and 5 for the 5^{th} order for a total of 9 function evaluations. For these 50 trajectories calculated, the code took 13 seconds, 9 of which was spent interpolating. This is a savings of 4.5 seconds because we only need to call the time interpolation function 6 times per iteration of the trajectory calculation.

It should be noted that later versions of the codes are more efficient and faster but the comparison of the methods using the original versions can still provide insight into the methods.

For the purposes of this project we decided to use RK4 as our integration method because our spatial interpolation moethid is $O(\Delta x^2)$ so we won't benefit from the extra accuracy of RKF.

4.2 Lagrangian Analysis

4.2.1 Deterministic method

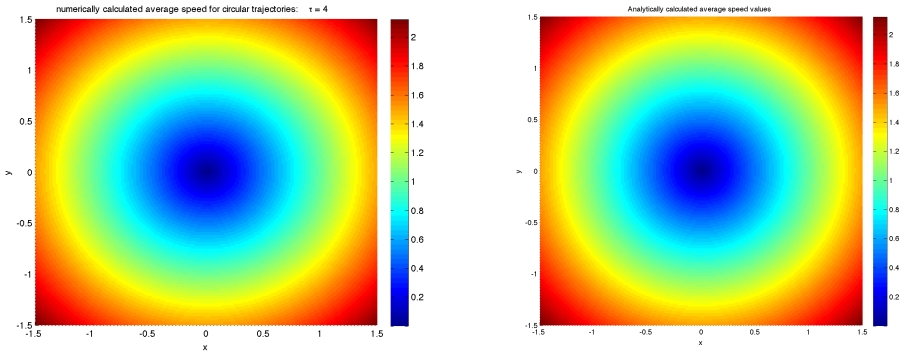
Validation of the deterministic method is done in two ways. The first approach is to apply the method to the circular trajectories used to validate the time integration (Equation 4.2). We have an analytic solution for the value of the M function for the circular trajectories, given in Equation 4.4. This is simply the arclength of the path a particle takes.

$$M(x_0, y_0, \tau) = 2\tau\sqrt{x_0^2 + y_0^2} \tag{4.4}$$

Applying the M function to the system in Equation 4.2, where $\tau = 4$, and then dividing all M values by 2τ yields the top panel of Figure 12. The data, integration and plotting parameters are given in Table 2. The expected average speed (Equation $4.4/2\tau$) is shown in the bottom panel of Figure 12. Both the top panel the bottom panel look identical, confirming that the code is calculating the length of path that a particle follows.

Figure	Database		Initial Condition and Integration			plotting
	dx, dy	dt	dx, dy	dt	τ	dx, dy
12	0.04	$0.03\bar{3}$	0.015	0.1	4.0	0.015
13,14	0.02	0.0400	0.01	0.10	4.0	0.0025
15	0.01	0.0481	0.01	0.05	10.0	0.0025

Table 2: This table gives the test database parameters (dx, dy, dt) as well as spacing for initial conditions, integration timestep size, and τ for several M function figures.



(a) **(b)**

Figure 12: Average speed values for circular trajectories produced by Equation 4.2 using a) the M function method and b) the analytically calculated values from dividing Equation 4.4 by 2τ. These plots show that points closer to the origin (blue) travel less distance than those further away (red). The plots are visibly indistinguishable. In fact, the largest difference between any given numerically calculated average speed and the analytical value was $3.6 * 10^{-6}$.

We then validate our method using the Duffing equation (Equation 4.5) where $\epsilon = 0$ for the autonomous case and $\epsilon = 0.1$ for the non-autonomous case [14]. We replicate the results found in [6] for both the autonomous Duffing equation and the non-autonomous Duffing equation for the purpose of verification. We note that in some plots the average speed is plotted instead of M value. This is done by simply dividing the M values by 2τ. We do this to facilitate easier comparison between different τ values for the same system.

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= x - x^3 + \epsilon * sin(t) \end{aligned} \tag{4.5}$$

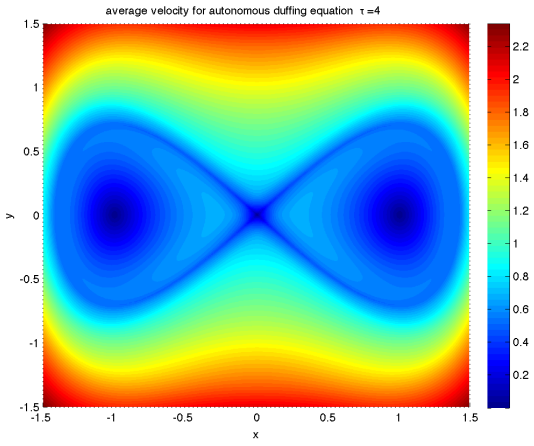


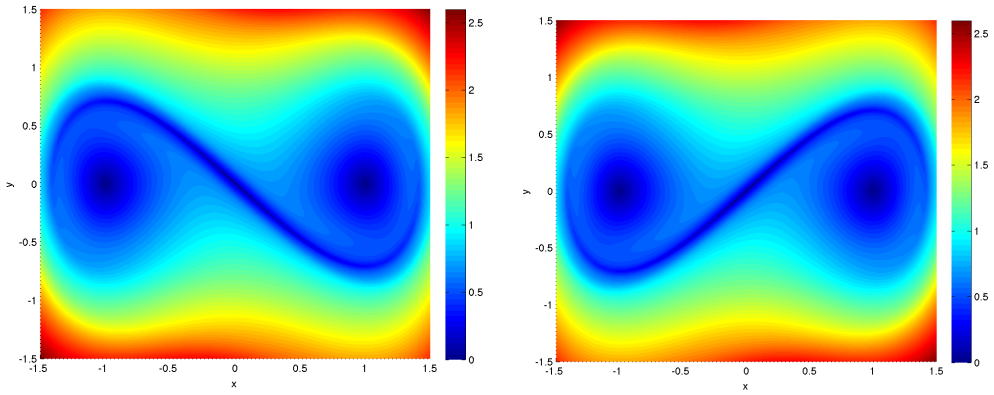
Figure 13: The average speed,calculated using the M function, for the autonomous duffing equation 4.5 ($\epsilon = 0.0$). This shows the stable and unstable manifolds together, as well as the lobed behavior of the system. This figure agrees with the results from [6].

The average speed is given in Figure 13 for the autonomous Duffing equation where $\tau = 4$. This plot agrees with the results given in [6]. While we see both the unstable and unstable directions in this plot, they can be seen separately in Figure 14.

The plot given in Figure 13 demonstrates how we may detect coherent structures with this method. We know that the left and right lobes contain particles that do not mix beyond the manifolds. This is evident in the Figure, as we can see a clear line separating the lobed trajectories from the trajectories outside of the manifold. It is this type of separation that we are looking for in our analysis of the bay.

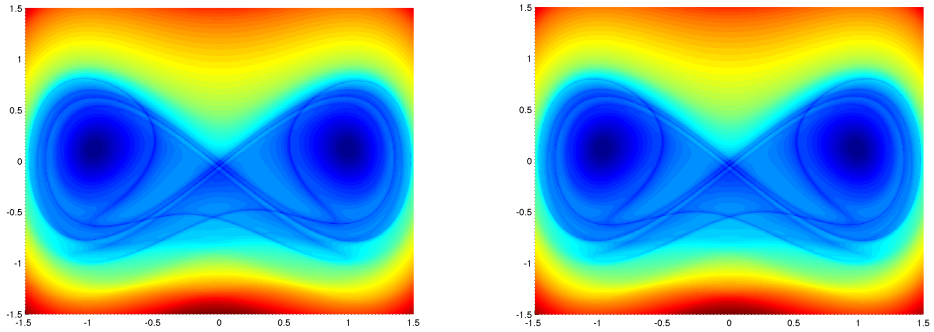
We also chose to verify the M function on the non-autonomous duffing equation, as shown in Figure 15. The left hand panel shows the system for $\tau = 0$ and $\epsilon = 0.1$, plotted using interpolated velocity values.

We verified this plot in two ways: the first by comparing our plot to the same system with exact function values instead of interpolated velocities. This is shown in Figure 15 (right panel). The second way we verified the M function method on this system was to compare our plot (left) with that in Figure 16. This plot is taken from [6] and shows the results the authors obtained (left) as well as a portion of the stable (blue) and unstable (red) manifolds of the system (right).



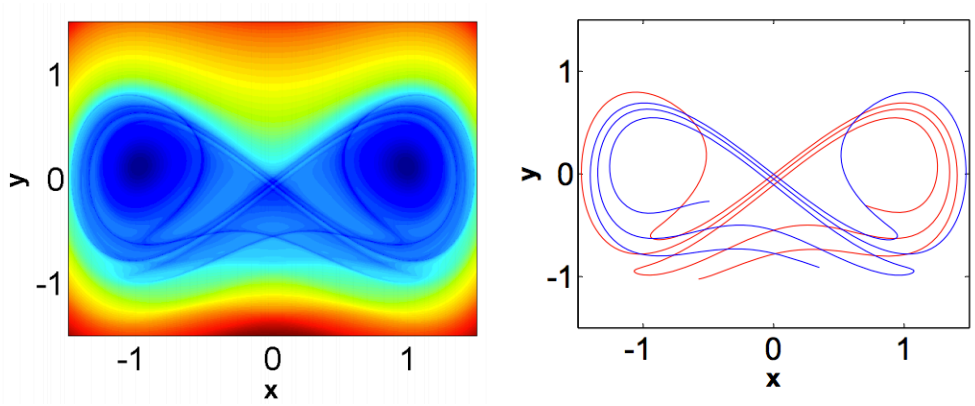
(a) **(b)**

Figure 14: The stable (left) and unstable (right) manifolds after $\tau = 4.0$.



(a) **(b)**

Figure 15: We see the non-autonomous duffing equation given in Equation 4.5 where $\epsilon = 0.1$. On the left is the system using interpolated values for the function and on the right is the plot when we instead use the exact function values.



(a) (b)

Figure 16: The results from [6] for the nonautonomous duffing equation given in Equation 4.5 where $\epsilon = 0.1$ (left). This figure matches well with our results in Figure 15. Segments of the stable and unstable manifolds, for the same system, from [6] are shown (right). This plot also agrees with our results.

4.2.2 Probabilistic method

We will verify this probabilistic method against results provided in [7] for periodically driven double gyre flow. The system used to verify our code is given in Equation 4.6, where $f(x,t) = \delta sin(\omega t)x^2 + (1 - 2\delta sin(\omega t))x$.

$$\begin{aligned} \frac{dx}{dt} &= -\pi A sin(\pi f(x,t)) cos(\pi y) \\ \frac{dy}{dt} &= \pi A cos(\pi f(x,t)) sin(\pi y) \frac{df(x,t)}{dx} \end{aligned} \tag{4.6}$$

The parameters are given as $A = 0.25$, $\delta = 0.25$, and $\omega = 2\pi$.

Using the system of equations provided in Equation 4.6, we calculate the transition matrix (2.13) by partitioning the domain $[0,2] \times [0,1]$ into 2^{15} square boxes with 400 uniformly distributed points per box. This gives a domain cell width of 2^{-7} and $13,107,200$ trajectories calculated from $t_0 = 0$ to $\tau = 1$. We then calculate the second larges eigenvalue, λ_2 along with the corresponding eigenvector, v_2. Our calculated $\lambda_2 = 0.9997$ while the paper with which we compared our results obtained an eigenvalue of $\lambda_2 = 0.9998$. We attribute this small difference to the method used to approximate the eigenvalues. The eigenvector is then reshaped (to reflect the original domain).

Our result is given in Figure 17. This eigenvector was originally normalized. The eigen-

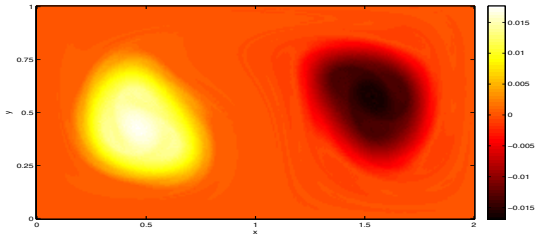
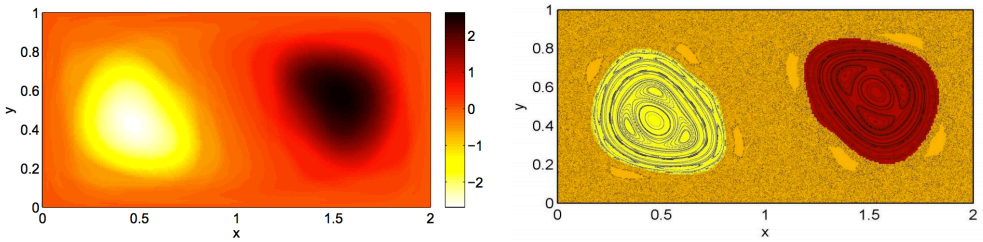


Figure 17: The reshaped (normalized) second eigenvector, v_2, for the system given in Equation 4.6.



(a) **(b)**

Figure 18: a) v_2 and b) almost invariant sets of the double gyre system, taken from [7]. We use panel a to validate our results for v_2 and panel b to demonstrate the structures we may expect to find in v_2.

vector obtained in [7] is given in Figure 18(a). There appear to be a difference in scale between the two plots. While our scale goes from -0.015 to 0.015, the scale in Figure 18(a) goes from -2.0 to 2.0. This is only because we originally normalized our eigenvector, which is not true of the eigenvector shown in Figure 18(a). There is also the problem of the reverse nature of our colorbar. This is due to a factor of -1 in from of our eigenvector. This factor was introduced to provide a better comparison between the two figures. While the scales are off by a factor of -1, the actual plots seem to be well matched. The last difference that we discuss between the two figures is the resolution difference. There appears to be better resolution in our plot, given that we manage to capture some of the structures seen in 18(b) particularly in the right gyre. This resolution difference can likely be attributed to a scaling difference again. If we decrease the magnitude of the bounds on the colorbar, we see a decrease in resolution, resembling that of Figure 18(a).

Our assessment is that the two eigenvalues and eigenvectors are approximately equal and any differences between the two eigenvectors is due to a difference in eigenvalue/eigenvector calculation methods and a difference in plotting preferences.

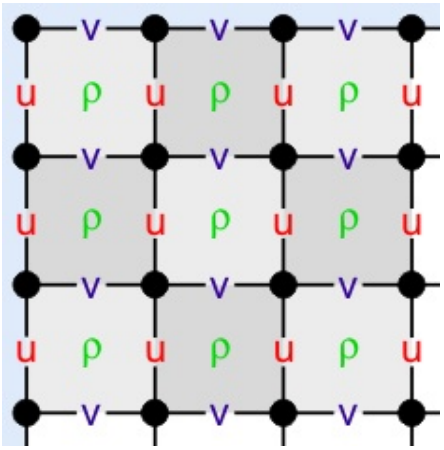


Figure 19: This grid is the Arakawa C-grid used by ROMS. We can see that the x-directed velocities are calculated along the left and right side panels of each grid cell while the y-directed velocities are calculated along the upper and lower faces of the grid cells.[1]

5 Application: Chesapeake Bay

5.1 Data set

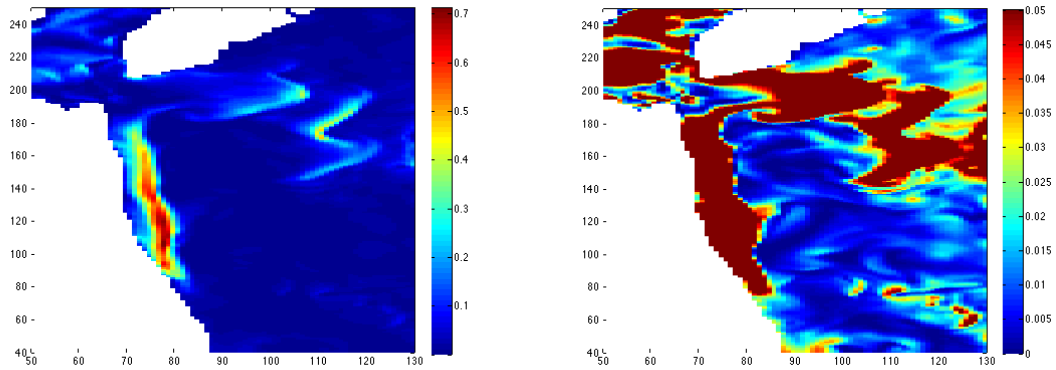
The goal of this project is to be able to analyze the dynamics of the Chesapeake Bay. To do this, we need velocity data corresponding to the surface of the bay. For the purposes of this analysis we will be using the Regional Ocean Modeling System (ROMS) to generate a velocity field for some interval of time.

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 are staggered using an Arakawa C-grid, as shown in Figure 19. 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. ROMS automatically sets all of the velocity values to zero on land[13].

Using the staggered grid (Figure 19) we interpolate u and v 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.

The grid used for the bay is not a regular uniform grid and therefore all of the calculations will be done in index space where we normalize the velocities from (m/s) to (grid spaces/s). This will allow for easier and faster calculations while not changing the general dynamics of the data. This means that the Chesapeake Bay will appear skew in our analysis of the bay.

To make the computations easier we will assume a steady state flow, so that the velocity



(a) **(b)**

Figure 20: The kinetic energy (KE) values for the steady-state Chesapeake Bay system we study. We see the a) original plot of KE as well as b) the KE plotted with a narrower color scale. This narrower color scale allows us to more easily see the low speed regions.

field is the same at every time.

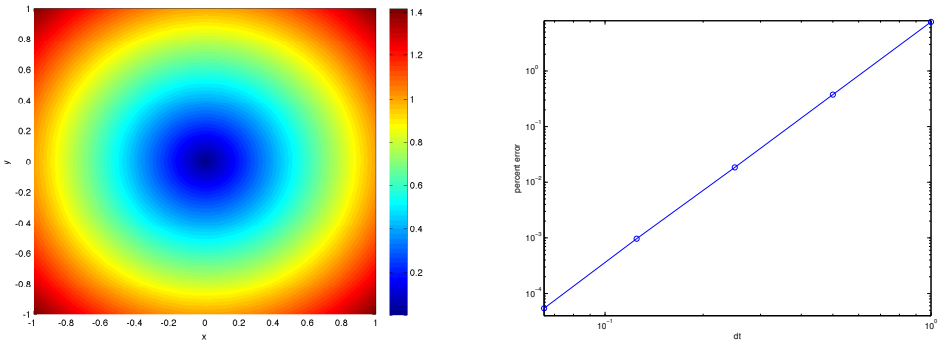
5.2 Validation

We validate our analysis of the ROMS data is two ways. First we applied the same analysis and code to circular trajectories where the velocity values were evaluated on an Arakawa C-grid. The second way was to run our M function on the data and then compare it to results obtained through ROMS. For both methods we rely on not only the absolute error but the point-wise percent error. The percent error is given by Equation 5.1 where M is our M value and M_{V} is the M value obtained through the validation method.

$$\text{Percent Error} = \frac{M - M_{\mathrm{V}}}{M_{\mathrm{V}}} \tag{5.1}$$

For the circular trajectories the M function results were compared to the exact results given b Equation 4.4. The results are given in Figure 21. We see in 21(a) the M values for our circular trajectories. In Figure 21(b) we see the maximum percent error, plotted against dt. The slope of this line is calculated as 4.3, which is consistent with a 4^{th} order time integration scheme.

[1]ROMS Wiki: Numerical Solution Technique. April 2012.



(a) **(b)**

Figure 21: a) M function values for circular trajectories calculated from staggared u and v velocity fields. b) The maximum point-wise percent error of the circular trajectories as a function of time step, dt. The slope of this error line is 4.3 which agrees with the order of the RK4 method.

For the second validation approach, the results are shown in Figure 22. The M value results of our analysis are given in Figure 22(a), measured in meters, while the percent error is given in Figure 22(b). From this error analysis we see some of the highest errors in or on the border of high velocity regions. This is reasonable, as the integration methods used in ROMS is a fourth order Milne predictor with a fourth-order Hamming corrector which is different from our Runge Kutta 4 approach. While the two methods are of the same order, any small difference in trajectories in or near the high velocity regions may result in larger differences in the M values. It should also be noted that as τ is increased the maximum and average percent difference decreases.

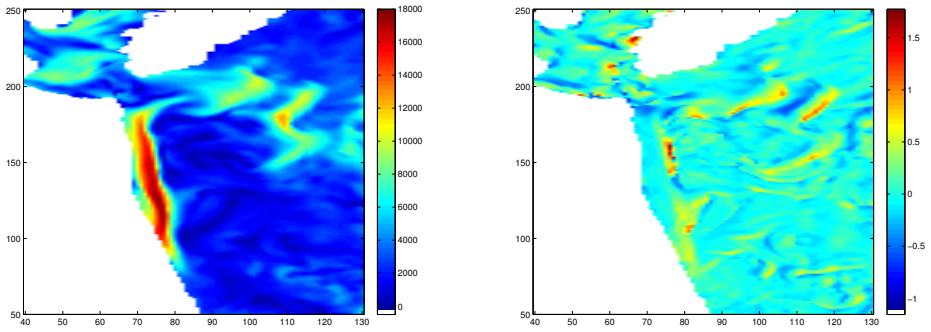
5.3 Results

The following two sections give the results of the Chesapeake Bay analysis.

5.3.1 Deterministic Method

Applying the M function to the ROMS data yields Figures 23, 24, and 24(c). The whole bay is shown in Figure 23 where $t_0 = 0$ and $\tau = 30$ min. The average speed which is shown is calculated using 105,501 trajectories. The red regions correspond to faster moving particles while the dark blue regions correspond to more stable regions of slower moving particles. From this image we can see there is a clear distinction between several regions of fast moving particles and nearby or bordering regions of slow moving particles.

In Figure 24(a) we see the forward half the integration and in Figure 24(b) we see the backward half of the integration for the M function analysis where $t_0 = 0$ and $\tau = 6$ hours



(a) **(b)**

Figure 22: a) Forward integration from $t = 0$ to $\tau = 6$ hr for our analysis to obtain M values, measured in meters. b) The percent error of our analysis. To calculate the percent error we calculate the M values through ROMS. Percent error is calculated using Equation 5.1. We see the maximum percent error is approximately 1.7%.

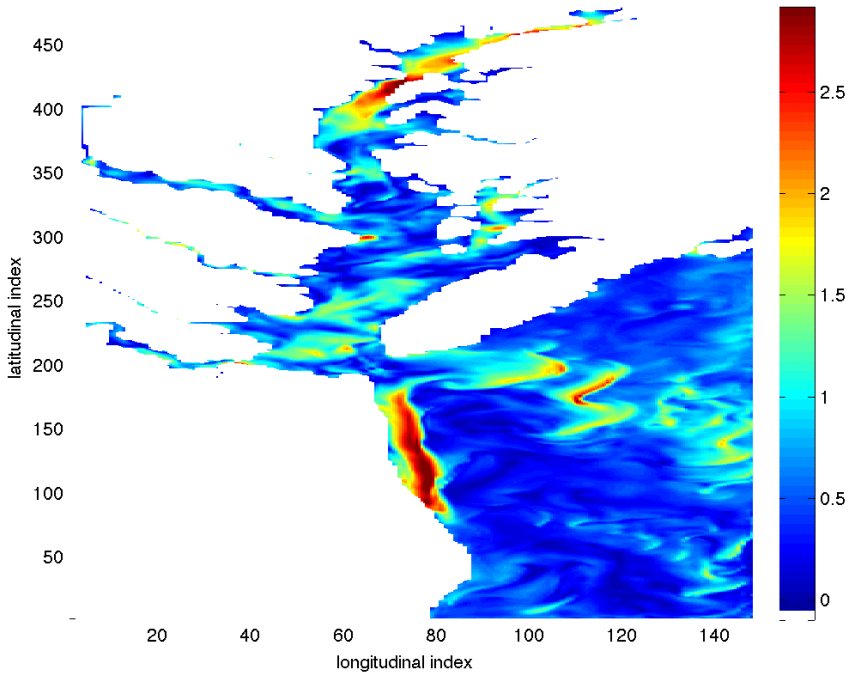


Figure 23: Shown above is the M function plot for the entire bay at $t = 0$ for $\tau = 30$ min, using 105,501 trajectories. (corresponding to a spacing between initial conditions of $dx = 0.5$.) Plotted is the average speed. The higher valued colors (red) correspond to faster moving particles while the lower valued colors (blue) correspond to slower moving particles. It is within the region of these slower moving particles that we may look to find the stable and unstable manifolds.

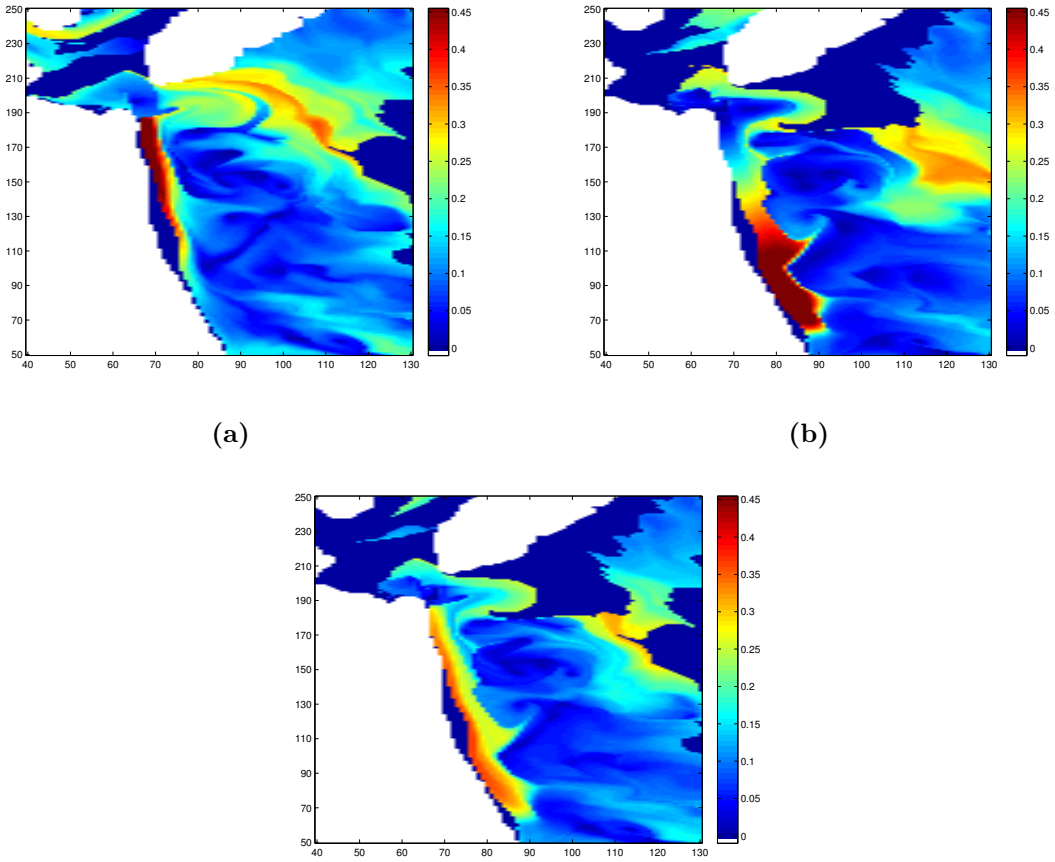


Figure 24: The M function plots for a section of the bay from $t_0 = 0$ to $\tau = 48$ hours. a) The forward integration, b) the backward integration, and c) the two averaged together for the total set of M values. All plots are showing the average speed in meters/second and use the same colorscale. The darkest blue portions are the intial conditions whose trajectories traveled outside of the boundaries of the bay and therefore have no M value. At these initial conditions we set the M function value to zero for plotting purposes.

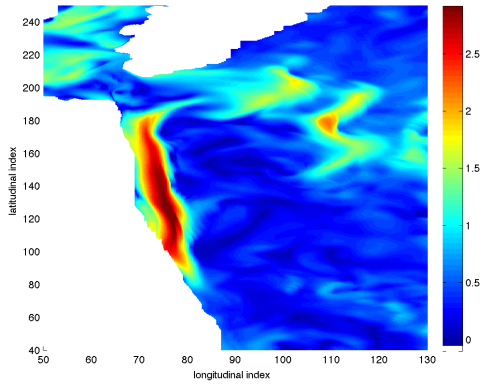


Figure 25: This plot shows the forward integration for $t = 0$ to $\tau = 6$ hr. This plot was created for the purposes of comparison with the plots created using the probabilistic method.

for a section of the bay. Together the total M value is shown in Figure 24(c). These average speed plots are calculated using 12,110 trajectories. This section is taken from the opening of the bay corresponding to the lower right corner of Figure 23. We see the emergence of the stable manifold in (a) and the unstable manifold in (b).

Figure 25 shows a very different picture of the forward integration of the bay. For this plot we used $t_0 = 0$ and $\tau =$hr for 777,720 trajectories. Here we see hints of the lagrangian structures from Figure 24(a). The similarities between the two plots are due to fixed point that are surrounded by rotating trajectories and near zero velocity regions. The trajectories used to create this plot are also used to apply the probabilistic analysis method to the Chesapeake Bay data set.

5.3.2 Probabilistic Method

For this method we used 777,720 trajectories to calculate the transition matrix. Our cell length was 1 grid length. This results in a domain with roughly 16 initial conditions per cell with the exception of the bay coast which may have less than 16 initial conditions. These trajectories correspond to the forward integration of the M function where $t_0 = 0$ and $\tau = 6$ hours.

From this method we present the first five eigenvectors. The first, v_1 is given in Figure 26 while v_2 and v_3 are given in Figure 27 and v_4 and v_5 are given in Figure 28. Also included in this section are plots of trajectories for areas of interest indicated by the eigenvectors.

In v_1 we see three distinct features. The first, in the upper right, is a green ring-like structure. This feature is not seen in the other eigenvectors shown here. Trajectories corre-

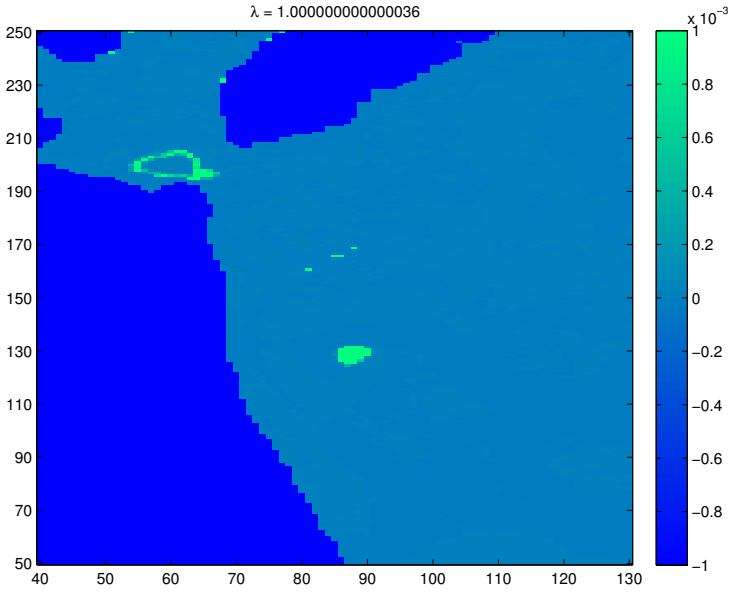
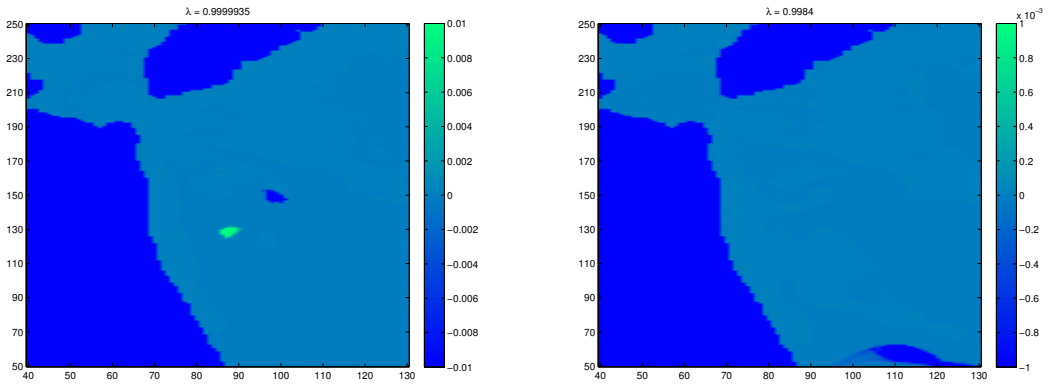
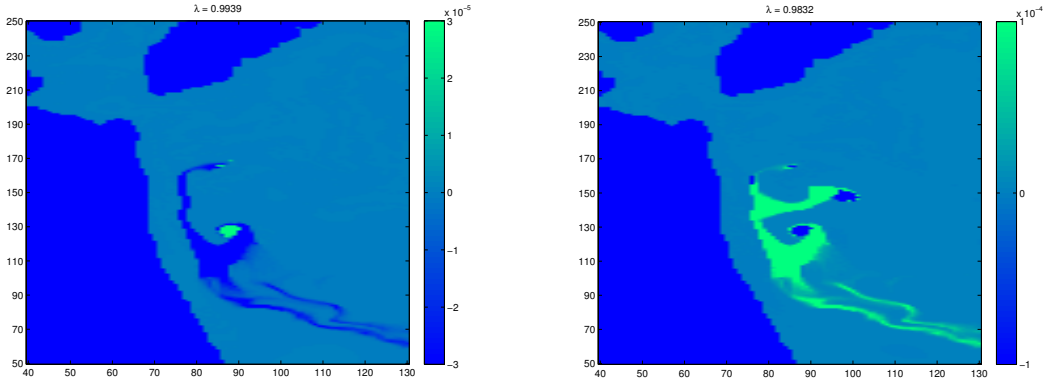


Figure 26: The eigenvector (v_1) associated with the largest eigenvalue ($\lambda_1 = 1.0000$) for 777,720 trajectories. The cell size is equal to 1 grid length. This results in 16 trajectories starting in each cell. This number might not be exact for cells on the edge of the bay. We see 3 features indicated: a ring-like structure in the upper left, several pixels of green in the middle, below which we see a larger oval-shaped feature. The ring-like feature indicates two fixed points and their surrounding trajectories, the second feature is a line of convergence of trajectories and the third feature is a another fixed point.



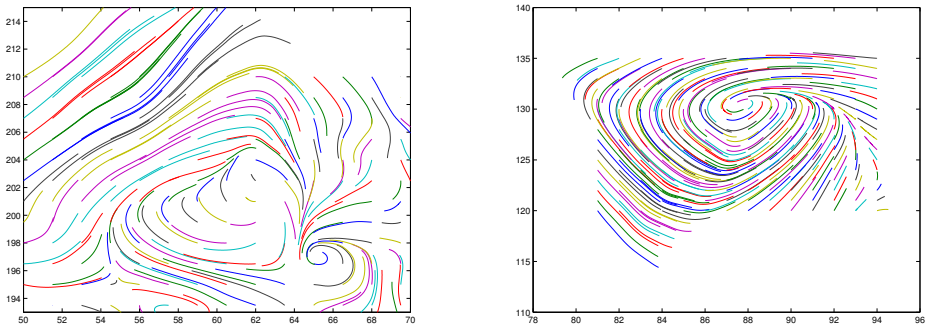
(a) v_2 **(b)** v_3

Figure 27: We see a) the eigenvector, v_2, associated with $\lambda_2 = 0.999994$ and b) the eigenvector, v_3, associated with $\lambda_3 = 0.9984$ for the same trajectories as v_1. In (a) we see in green the same fixed point as was indicated in v_1. We also see another fixed point, indicated by the blue oval in this figure. In (b) the feature indicated is at the bottom of the domain. This curved shaped is associated with yet another fixed point and the surround trajectories.



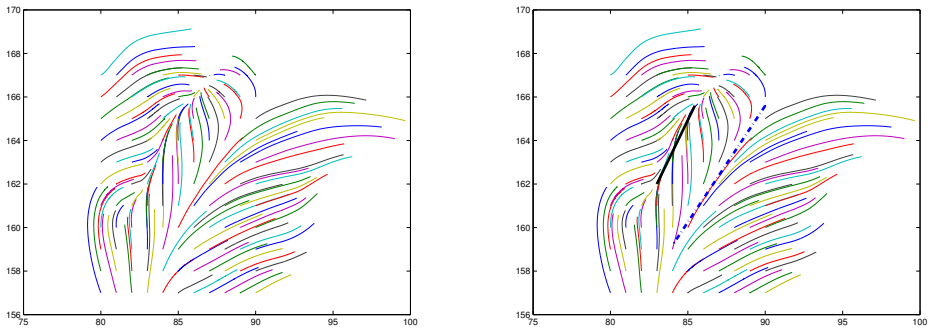
(a) v_4 **(b)** v_5

Figure 28: The next two eigenvectors are given in this figure. a) v_4 is associated with $\lambda_4 = 0.9939$ and b) v_5 is associated with $\lambda_5 = .9832$. In (a) we see the same fixed point from v_1 and v_2 in green. We also see some indication of the line of convergence indicated by v_1. Also seen is a weaker but larger underlying feature in dark blue that connects the green features. In (b) we see a similar underlying feature. This time in green. This green underlying feature connects two fixed points (blue) as well as two other small features that appear to be associated with that line of convergence previously discussed.



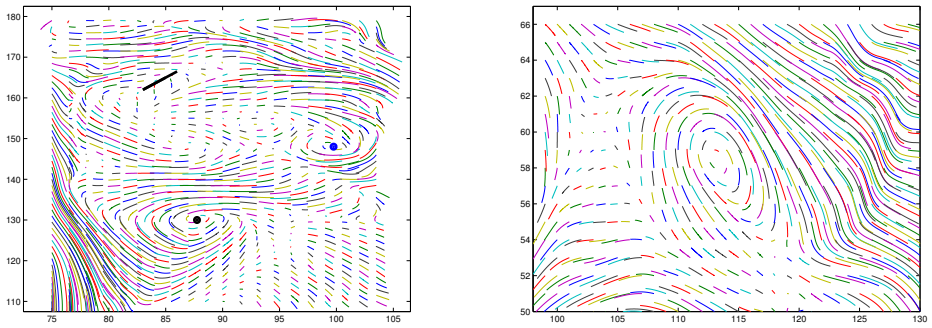
(a) **(b)**

Figure 29: a) Trajectories from the region in which we find the ring-like structure in Figure 26. b) fixed point seen in v_1, v_2, v_4, and v_5 along with the surrounding trajectories. Both sets of trajectories were calculated by integrating over the time interval $t \in [0, 6]$, measured in hours.



(a) **(b)**

Figure 30: These figures show trajectories that were calculated by integrating over the time interval $t \in [0, 24]$ measured in hours. This region is associated with the line of convergence seen in v_1 in Figure 26. b) The same region but with a black line highlighting the convergent direction and a blue dot-dash line that indicates a divergent direction.



(a) **(b)**

Figure 31: a) Here we see trajectories from a larger view of our domain on the bay. The black line indicates the convergent line from Figure 30. The blue dot indicates the fixed point from Figure 29(b). It is this region that can be seen in v_1, v_2, v_4, and v_5. The blue dot indicates another fixed point which can be seen in v_2 and v_5. b) This plot shows trajectories from the lower right corner of our domain. What we see here is a fixed point and the associated rotating trajectories that can be detected in v_3.

sponding to this region are given in Figure 29(a) which shows that this structure is actually a larger set of trajectories rotating about a fixed point with a smaller set trajectories rotating around a fixed point to the right. The larger set composes the ring-like shape seen while the smaller set is the protrusion on the right of the ring. The second feature seen in v_1 is the green oval located around $(87, 130)$.This region is associated with the rotating traectories shown in Figure 29(b). This feature is also indicated in Figure 31(a) by a black dot. The third feature in v_1 is a few pixels of green located between $x \in (80, 90)$ at approximately $y = 165$. This set of pixels follows a line of convergence shown in Figure 30(a). This is in fact a hint of the stable manifold which can only be seen in the M function at much later times.

The next eigenvector ,v_2, indicates two features shown in Figure 27(a). The green oval in this plot is the second feature from v_1. More interestingly we see another oval, but in blue. This oval is associated with another fixed point and its surrounding rotating trajectories. We can see these trajectories in Figure 31(a). The approximate fixed point of these trajectories is indicated by a blue dot.

The third eigenvector, given in Figure 27(b), indicates a feature in the lower right corner of the domain. This feature is explained by the trajectories given in Figure 31(b). This indicated region is actually another region of rotating trajectories.

In v_4, given in Figure 28(a), we see several structures. First, in green, we see the same fixed point as was visible in v_1 and v_2. Secondly, we notice similar green pixels as we noticed

in v_1 which indicated the line of convergence for many trajectories. This line was also highlighted in Figure 30(b). The last thing that we notice is the underlying structure in dark blue. This structure is much weaker than the others and indicates the connection between the other two features in this eigenvector. This structure might also indicate some prefered direction of travel for trajectories that escape convergent regions and regions of periodicity.

Lastly, we discuss v_5, given in Figure 28(b). This figure indicates structure similar to those indicated by v_4. We see the same convergent line indicated, the same fixed point (both from v_1) as well as a similar underlying feature. A difference that we do detect is in the appearance of the fixed point noted in v_2. Altogether this demonstrates that there is some weak connection between these features which must be investigated further.

6 Concluding remarks and further discussion

We have shown that the interpolation method chosen (bilinear interpolation) paired with a 4^{th} order integration scheme (Runge Kutta 4) provide trajectories that are accuarate enough to reveal dynamical features in the many systems, such as the Duffing equation as well the Chesapeake Bay.

Both the deterministic method and the probabilistic method presented in this paper have shown to highlight certain aspects of the dynamics of the systems studied. The M function seems best suited for finding regions of diffusive trajectories as well as highlighting regions of transition between structures particles. The probabilistic method highlights periodic regions and possible connections between these periodic regions. We also saw that the probabalistic method indicated the location of the stable manifold before it was visible in the M function plot.

The deterministic method is superior computationally when it comes to the number of trajectories needed for a given resolution. For the same resolution, the deterministic method requires far fewer particles than the probabilistic method. it should be noted that we do find that this difference in resolution does not significantly take away from the information provided by the probabilistic method. The lower resolution images created using the probabilistic method do provide an insight into the coherent structures within the bay that may have otherwise gone unnoticed if we had been using only the deterministic method.

7 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
- The ROMS data set
- Proposal document and presentation
- Mid-year document and presentation
- Final report and presentation

8 Schedule and Milestones

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

- Develop code for Interpolation *(October - November)*
  - Develop bilinear method (no time interpolation) *(October)* (done on time)
  - Develop bicubic method (no time interpolation) *(October - Early November)* (done in January)
  - Develop the method for Lagrange Polynomial interpolation in time *(November)* (done on time)
  - Validation of Interpolation methods *(October to Late November)* (done on time and January)
  - Time permitting: Parallelization of the interpolation process (done in January)
- Time integration methods *(Late November - December)*
  - 4^{th} oder Runge Kutta *(Late November)* (done on time)
  - 5^{th} order Runge Kutta Fehlberg method *(Late November - Early December)* (done on time)
  - Validation of the Trajectory (time integration) computation *(Late November - Early December)* (done on time)
- M Function analysis *(January - mid February)*
  - Modify time integration methods to incorporate calculation of the distance each particle travels *(January - mid February)* (done on time)

  - Validation of this M Function *((January - mid February)*(done on time)

- Probabalistic method *(Mid February - April)*
  - Set up indexing *(February)* (done on time)
  - Create T Matrix *(Early March)* (done on time)
  - Compute eigenvalues and eigenvectors T *(March)* (done on time)
  - Analyze dominating dynamics *(Late March - Early April)* (done on time)
  - Validate Probabalistic method using test case *(Late March - Early April)* (done on time)
  - Compare Deterministic method and Probabalistic method *(Early April)* (done on time)

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] Stremler M. A., Ross S. D., Grover P., Kumar P. "Topological Chaos and Periodic Braiding of Almost-Cyclic Sets". *Physical Review Letters* 106, 114101 (2011).

[6] Mancho A.M., Wiggins S., Curbelo J., Mendoza C. "Lagrangian Descriptors: A Method for Revealing Phase Space Structures of General Time Dependent Dynamical Systems". *Communications in Nonlinear Science and Numerical Simulation*, 18, (2013) 3530-3557.

[7] Froyland G., Padberg-Gehle K. *Ergodic Theory, Open Dynamics, and Coherent Structures.*, "Chapter: Almost-invariant and finite-time coherent sets: directionality, diration, and diffusion." *Mathematics and Statistics* Vol. 70, Springer (2014) `http://web.maths.unsw.edu.au/~froyland/froyland_padberg_v23.pdf`

[8] 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`

[9] Greg Fasshauer "Chapter 5: Error Control" Illinois Institute of Technology, Chicago, IL. April 24, 2007. `http://www.math.iit.edu/~fass/478578_Chapter_5.pdf`

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

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

[12] Davis, L. M. RKF and ABM. Montana State University, Bozeman, MT. `http://www.math.montana.edu/~davis/Classes/MA442/Sp07/Notes/RKF_ABM.pdf`

[13] ROMS Wiki: Numerical Solution Technique. April 2012. Last visited: Sept. 22 2013 `https://www.myroms.org/wiki/index.php/Numerical\_Solution\_Technique`

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

[15] 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