

Markov-Chain Models of Ocean Dispersion

Kate Maurer-Song

September 19, 2018

Abstract

The ocean is one of Earth's major transport systems. This transport by currents contributes to heat, CO₂, salt, and debris movement both on the ocean's surface and below. This project explores how this transport can be modelled by a discrete Markov chain and how the advection-diffusion equation can be derived from the corresponding transition matrix.

1 Introduction

Ocean transport is the result of surface and underwater motion. Wind stress and the Earth's rotation contribute to the surface currents. Because of the Coriolis force (caused by Earth's rotation), water at the surface moves at 45° of the wind's direction. Below the surface, the wind-driven velocity decreases with depth and rotates further away from the wind direction, as a result of the stress induced by the fluid layers above. The resulting depth-integrated transport, known as Ekman transport, is perpendicular to the wind.

Ocean transport does not just move water. The transport of heat and carbon dioxide is particularly important to Earth's climate. Heat transported through jets and gyres regulates Earth's temperature [3]. For example, the Gulf Stream carries heat from the Gulf of Mexico to the Arctic; this causes Northern Europe's relatively warm climate for its latitude. Carbon dioxide transport on the other hand plays a big role in climate change. The ocean can only absorb so much CO₂ without consequence. Excess CO₂ that is not absorbed by the atmosphere or land is absorbed and decomposed by the ocean. This excess leads to ocean acidification and decreased CO₂ removal from the atmosphere [13]. Hence, to better understand the Earth's climate and changes, it is crucial to first understand ocean transport and diffusion.

To study ocean transport, we will use discrete particle trajectory data from a simulation of a highly simplified model of the North Atlantic Ocean. The discrete trajectories will be formulated into Markov Chain Models which will allow us to determine long-term particle distribution, transport velocity, and diffusivity.

2 Transport Markov Matrix

2.1 Model and Overview

We use simulation results from a 3-layer quasigeostrophic model that crudely represents a region in the North Atlantic; the model is driven by wind leading to a circulation that features a jet (starting in the West) separating the two main gyres and an active field of mesoscale eddies. The partial differential equations of the model are solved using a finite-difference discretisation as described in section 3.1 of [9]. 2500 particles are then advected using a piecewise linear representation of the quasigeostrophic streamfunction [9]. The locations of these 2500 particles are recorded every day for 1828 days (5 years). On day 1, the particles are evenly dispersed between $x \in [4.4416 \text{ m}, 3839300 \text{ m}]$ and $y \in [25.2820 \text{ m}, 3839900 \text{ m}]$.

We will use the trajectory data to estimate two different transition matrices. The first approximates particle movement over 5 years. The second approximates particle movement over a decided time increment, *inc.* Because the data models an incompressible flow, we do not expect to see any long term particle convergence for either transition matrix.

2.2 Markov Transport Matrices

Particle movement can be represented by Markov chains. First, we divide in the ocean surface into numbered cells in order to obtain a numbered finite state space. We then track the discrete trajectories by recording each particle's daily box number [6].

In practice, we use a square $m \times m$ cell grid (where $m \times m = n$). This allows for $\{B_1, B_2, \dots, B_n\}$ boxes. When using $m = 16$, we get $\{B_1, B_2, \dots, B_{256}\}$, where B_1 is in the lower left-hand corner, B_2 is directly above B_1 ; B_{17} is directly to the right of B_1 ; B_{256} is in the upper right-hand corner.

Each box has the same dimension. Let the North-South length be $dy = y_{large}/m$ and the East-West width be $dx = x_{large}/m$, where y_{large} is the maximal y coordinate and x_{large} is the maximal x coordinate. That is, the y and x domain sizes are divided by the number of boxes per row. Each particle's location for every timestep is then converted into the box location and recorded into matrix \mathbf{B} . \mathbf{B} is 1828×2500 and designed so that $\mathbf{B}(i, j) = k$ if particle j is in box k on day i . For computational efficiency, we form matrix \mathbf{B}_r , a reduced form of \mathbf{B} . We define an increment value (in days) so that $\{inc \mid 1 \leq inc \leq 1828; inc \in \mathbb{N}\}$ and a starting day, s . Then, \mathbf{B}_r takes the rows from \mathbf{B} every inc days, starting from day s .

2.2.1 Model A

Method A forms a long term Markov Transport Matrix by using short trajectories. To cover as much of the 5 year period as possible, we examine each trajectory every inc steps as opposed to just comparing day 1 and day 1828. Using short trajectories as opposed to long should make the model more accurate, as long trajectories compound more error [5].

Given the initial particle distribution and one-step transition matrix, we can completely characterize the Discrete Markov Chain. That is, we can determine particle distribution for all timesteps. Mathematically, $p_k^t = \sum_j \mathbf{T}_{j \rightarrow k} p_j^{(t-1)}$, where p_k^t is the probability vector of hitting box k from box j at time t , $\mathbf{T}_{j \rightarrow k}$ is the one-step transition matrix from j to k , and $p_j^{(t-1)}$ is the probability vector of being in box j at time $t - 1$. If we can determine \mathbf{T} we can calculate the particle distributions for all timesteps.

Specifically for Model A, we will be forming the transition matrix \mathbf{T}_A via the Chapman-Kolmogorov method. We let $\mathbf{T}_A = \mathbf{T}_{c1} \times \mathbf{T}_{c2} \times \mathbf{T}_{c3} \dots$ (until as many timesteps are covered as possible), where the \mathbf{T}_{c_i} 's are " inc -step" transport matrices. In this case, moving one-step is equivalent to moving forward inc days until as much of the 5-year span is covered as possible.

The entries of \mathbf{T}_{c_i} are simple ratios:

$$\mathbf{T}_{c_i}(k, q) = \frac{\#\{x \in \{\text{Box } q \text{ at time } 1 + (inc \times i) \mid \text{Box } k \text{ at time } (inc \times (i - 1))\}\}}{\#\{x \in \text{Box } k \text{ at time } 1 + (inc \times (i - 1))\}},$$

where x is a particle. As each \mathbf{T}_{c_i} is formed every inc steps starting at day 1, we will end up with $matx = \lfloor \frac{1828}{1+inc} \rfloor$ intermediate stochastic matrices. Consequently, $\mathbf{T}_A = \prod_{i=1}^{matx} \mathbf{T}_{c_i}$.

2.2.2 Model B

Model B instead forms a single Markov Transport Matrix to study particle movement over inc days only. As in Model A, we use \mathbf{B}_r , the reduced form of \mathbf{B} . However, we directly form the discrete Transport Matrix, \mathbf{T}_B . This time, we are interested in change after inc days: $p_k^{inc+1} = \sum_j \mathbf{T}_{j \rightarrow k} p_j^1$.

Instead of creating $matx$ intermediate matrices, \mathbf{T}_B is defined:

$$\mathbf{T}_B(k, q) = \frac{\bar{x} \in \text{travelling from Box } k \text{ to Box } q \text{ after } inc \text{ days}}{\bar{x} \in \text{starting in Box } k \text{ at day } 1},$$

where \bar{x} is the mean number of particles. To accomplish this we count how many times all particles travel from Box k to Box q after every inc days and form an average value for each start and end location pair.

2.3 Eigenvectors and Eigenvalues

As stated earlier, $p_k^t = \sum_j \mathbf{T}_{j \rightarrow k} p_j^{(t-1)}$. If we let p be a row vector, then $p^{(t)} = p^{(t-1)} \mathbf{T} = p^{(t-2)} \mathbf{T}^2 = \dots = p^{(0)} \mathbf{T}^t$. Intuitively, this makes sense; given an initial stationary distribution, $p^{(0)}$,

to determine the distribution at time t we multiply $p^{(0)}$ by \mathbf{T}^t , the one-step matrix to the t^{th} power.

By the properties of stochastic matrices, we know that there are n eigenvalues (λ_i) and corresponding left eigenvectors (\vec{v}_i) associated with transport matrix \mathbf{T} [10]. Then for constants $A_i \in \mathbb{C}$ and $i \in [1, n]$:

$$\begin{aligned} p^{(t)} &= p^{(0)} \mathbf{T}^t \\ &= A_1 \vec{v}_1 (\lambda_1)^t + A_2 \vec{v}_2 (\lambda_2)^t + \dots + A_n \vec{v}_n (\lambda_n)^t \\ &= \sum_{k=1}^n (\lambda_k)^t \vec{v}_k A_k. \end{aligned}$$

Therefore, for all $k \in [1, n]$, p_k^n can be expressed as a linear combination of the eigenvectors of \mathbf{T} .

Additionally, by Perron-Frobenius Theorem, we know that there is a leading eigenvalue $\lambda_1 = 1$ [8]. Obviously, the corresponding eigenvector yields $\vec{v} = \vec{v} \mathbf{T}$. Contextually, this means that there is a set of probability densities that stay the same after an application of the transport matrix. That is, $p_k^t \rightarrow p_k^\infty$ as $t \rightarrow \infty$. Therefore, the left eigenvectors of the transport matrix correspond to forward-time dynamics. As stated earlier, this model shows an incompressible flow. This means that p_k^∞ will converge to uniformity which eliminates any high particle density regions.

The n eigenvalues also yield n right eigenvectors. If given \mathbf{T}^T , the right eigenvectors now follow the rules of left multiplication, thus satisfying the same properties as above. However, the similarities end here, as the right eigenvectors represent backwards time dynamics. Clearly, the right eigenvector corresponding to $\lambda_1 = 1$ is uniform, as \mathbf{T} is row stochastic. Particularly we are interested in the right eigenvector corresponding to λ_2 . The entries of this eigenvector indicate the distance each corresponding box is from the equilibrium position, where distance corresponds to length of time. For example, \vec{r}_i estimates the time until trajectories within box i reach uniformity [14].

Both the right and left eigenvalues are the same. The first eigenvalue, as already made clear, is always $\lambda_1 = 1$. The second eigenvalue, λ_2 gives the rate of convergence of the probability p_k^T towards the (uniform) invariant probability p_k^∞ . The closer λ_2 is to 1, the slower that rate is [7].

For \mathbf{T}_A and \mathbf{T}_B we compute both the left and right eigenvectors using the in-built Matlab commands. Some of the eigenvalues and corresponding vectors will include imaginary components but we disregard these during analysis.

Both sets of eigenvectors are plotted using the `pcolor` command. This allocates each entry (1 through n) of the eigenvector to the corresponding box. The eigenvector is then realised by colour gradient.

2.4 Statistics

For transport matrix \mathbf{T}_B , we estimate the probability of a particle moving from box k to box q in inc days. A binomial random variable measures the number of successes over N trials. If we think of this move between boxes as a success, we can create a binomial random variable, T_x , with success probability $\mathbf{T}(k, q)$ and number of trials \vec{N} , the vector of total number of visits to box i .

To determine the reliability of the model we compute the standard deviation of the binomial random variable, $T_x(k, q) = \frac{\mathbf{T}(k, q)}{N^{(k)}}$. We compute the standard deviation of T_x .

$$\text{Var}(T_x) = \text{Var}\left(\frac{T}{N}\right) = \frac{1}{N^2} \text{Var}(T) = \frac{p(1-p)}{N} \Rightarrow \text{SD}(T_x) = \sqrt{\frac{p(1-p)}{N}}.$$

That is, computing the standard deviation gives an $n \times n$ matrix \mathbf{T}_x , where $\mathbf{T}_x(k, q) = \sqrt{\frac{\mathbf{T}(k, q)(1-\mathbf{T}(k, q))}{N^{(q)}}}$. These values are then plotted against the values from matrix \mathbf{T} .

We aim to keep the error no worse than 20% which can be achieved by using the appropriate m value.

2.5 Diffusion

For Model B we can calculate diffusion-advection coefficients for the chosen inc value. Since the transition matrix varies smoothly, it is possible to make time and space continuous in order to

approximate the relation $p^t = p^{t-1}\mathbf{T}$ by an advection-diffusion equation. To find $\frac{dP_i}{dt}$ we adapt the continuous time random walk diffusion-advection equation for this discrete time application. First, derive the formula for the one-dimensional case.

Let P_j^n be a vector of probabilities at time n of being in box j and $\mathbf{T}_{i \rightarrow j}$ be entry (i, j) of the discrete row stochastic Markov matrix. By Total Probability Rule and Chapman-Kolmogorov Theorem,

$$P_j^{n+1} = \sum_i \mathbf{T}_{i \rightarrow j} P_i^n. \quad (1)$$

We estimate the first time derivative of P_j^n by

$$\partial_t P_j^n = \frac{\sum_i \mathbf{T}_{i \rightarrow j} P_i^n - P_j^n}{\Delta t}. \quad (2)$$

Next, rename some variables:

$$P_i^n = p(t, i\Delta x) = p(t, (i-j+j)\Delta x); \quad (3)$$

$$P_j^n = p(t, j\Delta x) = p(t, x). \quad (4)$$

Taylor expanding (3) around $\Delta x(i-j) = 0$ to the second degree, we find that

$$p(t, i\Delta x) = p(t, j\Delta x) + \partial_x \left(\Delta x(i-j)p(t, j\Delta x) \right) + \frac{1}{2} \partial_{xx} \left((\Delta x(i-j))^2 p(t, j\Delta x) \right). \quad (5)$$

Introducing (5) into (2) and noting that $\sum_j \mathbf{T}_{i \rightarrow j} = 1$ gives

$$\partial_t p(t, x) = \partial_x \left(p(t, x) \frac{\Delta x}{\Delta t} \sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right) + \frac{1}{2} \partial_{xx} \left(p(t, x) \frac{\Delta x^2}{\Delta t} \sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} \right). \quad (6)$$

To find $\frac{dP_j}{dt}$ we must also derive $\partial_{tt} P_j$ (to the second order). This will allow us to determine (via Taylor Series) $\frac{dP_j}{dt} = \partial_t P_j - \frac{\Delta t}{2} \partial_{tt} P_j$ in order to form the final diffusion-advection discrete equation.

$$\partial_{tt} p(t, x) = \partial_x \left(\partial_t \left(\frac{\Delta x}{\Delta t} p(t, x) \sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right) \right) + \frac{\Delta x^2}{2} \partial_{xx} \left(\partial_t \left(\frac{1}{\Delta t} p(t, x) \sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} \right) \right) \quad (7)$$

$$= \partial_{xx} \left(\frac{\Delta x^2}{\Delta t^2} p(t, x) \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right). \quad (8)$$

Introducing (8) into (6) and isolating $\partial_t P_i$ gives

$$\partial_t p(t, x) = \partial_x \left(\frac{\Delta x}{\Delta t} p(t, x) \sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right) + \frac{1}{2} \partial_{xx} \left(\frac{\Delta x^2}{\Delta t} p(t, x) \left[\sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} - \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right] \right). \quad (9)$$

For the chosen time increment inc , let $u = \frac{\Delta x}{\Delta t} \sum_i (j-i) \mathbf{T}_{i \rightarrow j}$ be mean velocity and $\kappa = \frac{\Delta x^2}{2\Delta t} \left[\sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} - \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right]$ be diffusivity. Then,

$$\partial_t p(t, x) = -\partial_x (up(t, x)) + \partial_{xx} (\kappa p(t, x)) \quad (10)$$

For (10) to be of use for Model B, we must account for movement in both the y direction (North-South) and x (East-West) direction.

Define the mean velocity vector:

$$\vec{u} = \begin{bmatrix} \frac{1}{\Delta t} \sum_j (j-i) \mathbf{T}_{i \rightarrow j} \\ \frac{1}{\Delta t} \sum_j (j-i) \mathbf{T}_{i \rightarrow j} \end{bmatrix},$$

and the diffusivity matrix:

$$\kappa = \begin{bmatrix} \kappa_{1,1}(j) & \kappa_{1,2}(j) \\ \kappa_{2,1}(j) & \kappa_{2,2}(j) \end{bmatrix}.$$

Each element from κ leads to a piecewise constant field $\kappa(x, y)$ via the map $j \mapsto (x(j), y(j))$, where $x(j)$ and $y(j)$ are the coordinates of the centre of box j and $i \mapsto (x(i), y(i))$ are the coordinates of the centre of box i . Hence,

$$\begin{aligned}\kappa_{1,1} &= \frac{1}{2\Delta t} \left[\sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} - \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right] \\ \kappa_{1,2} = \kappa_{2,1} &= \frac{1}{2\Delta t} \left[\sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} - \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right] \\ \kappa_{2,2} &= \frac{1}{2\Delta t} \left[\sum_i (i-j)^2 \mathbf{T}_{i \rightarrow j} - \left(\sum_i (i-j) \mathbf{T}_{i \rightarrow j} \right)^2 \right].\end{aligned}$$

Then, $\partial_t p = -\partial_x(u_1 p) - \partial_y(u_2 p) + \partial_{xx}(\kappa_{1,1} p) + 2\partial_{xy}(\kappa_{1,2} p) + \partial_{yy}(\kappa_{2,2} p)$.

3 Results

Before calculating \mathbf{T}_A or \mathbf{T}_B we plot some trajectories of the particles in a so-called spaghetti diagram [11]. For clarity Figure 1 only examines 10 particle paths. These diagrams give a good visualization of movement through the 3 layers. In the first layer, the particles move throughout a larger area of the ocean surface. As the depth increases, the particles stay closer to their initial locations. This is due to Ekman Transport: as the depth increases, wind has less influence on the particle motion causing sluggish movement [12].

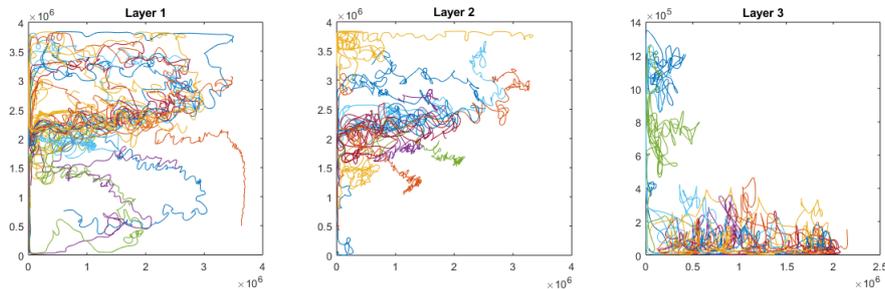


Figure 1: The spaghetti diagrams of 10 particles for the 3 layers.

3.1 Preliminaries

(For the detailed code, see Appendix). To implement the algorithm we choose some variables.

- We divide the ocean surface into 16×16 boxes; let $m = 16$ and $n = 256$. This means that $dx = 239,994$ and $dy = 240,000$. \mathbf{T}_A and \mathbf{T}_B are then 256×256 Markov matrices.
- Let $inc = 50$. For Model A, each of the 35 contributing transport matrices are calculated every 50 days. For Model B, average particle movement is investigated over 50 days.
- While we will compute all 256 eigenvalues and corresponding right and left eigenvectors, we will only plot the first 2 real eigenvectors of each.

3.2 The Matrix

Model A and B both form row stochastic matrices. Model A covers as much of 5 years as possible in 50 day increments while Model B studies movement over 50 days. In comparison to \mathbf{T}_A , \mathbf{T}_B is

much sparser. (For the given parameters, \mathbf{T}_B has 8677 zero entries and \mathbf{T}_A has no zero entries). This is because particles cannot move over many boxes in only 50 days. Often times, the particles only move one box (in any direction) in 50 days. This is also why the maximum value for each row ($r \in [1, 256]$) of \mathbf{T}_B tends to be within column $[r - 16, r + 16]$.

3.2.1 Eigenvalues and Eigenvectors

We isolate the real eigenvalues and corresponding left and right eigenvectors for both models.

3.2.2 Model A

All 3 layers of the ocean yield row stochastic transport matrices. As expected by the Perron-Frobenius Theorem, the leading eigenvalue is $\lambda_1 = 1$ [8]. The corresponding right eigenvector is $(0.625, 0.625, \dots, 0.625)^T$ (meaning that the expected dispersion is uniform) and corresponding left eigenvector indicates the 5 year stationary distribution[2]. For all 3 layers, the stationary distribution is not significantly far from uniform. This means that p_k^∞ corresponds to uniform mixing. Therefore, in this model, there is no particle patch formation. This uniformity is clearly seen in the first panels of Figures 2 and 3.

There is also movement associated with the other eigenvalues. The eigenpicture corresponding to $\lambda_2 = 0.19674$ in Figure 2 shows high particle concentration in the lower left-hand corner. This means that if particles are placed in the southern hemisphere, in a relatively short time (359 days), the particles will converge to uniform.

The right eigenvectors in panels 2 of Figure 3 indicate that the model approaches uniformity relatively fast, thus agreeing with the left eigenpictures.

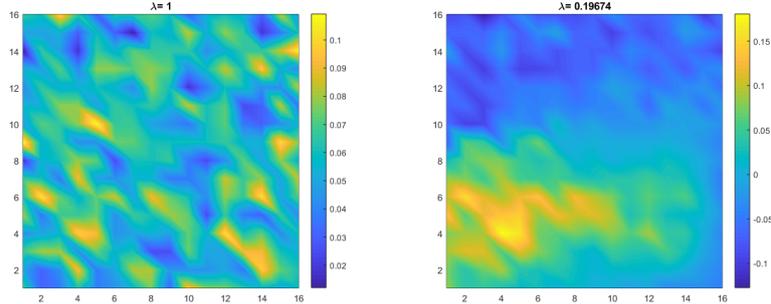


Figure 2: Layer 1, left eigenvectors for $m = 16$, $inc = 50$.

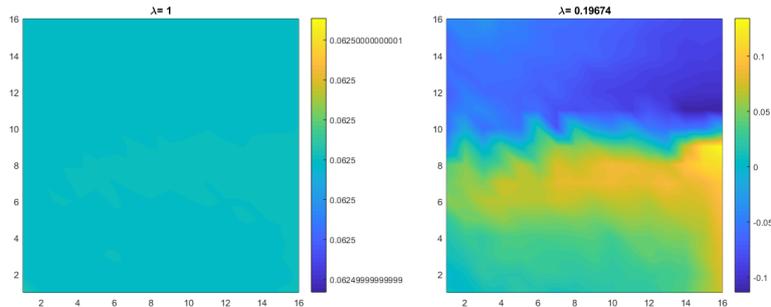


Figure 3: Layer 1, right eigenvectors for $m = 16$, $inc = 50$

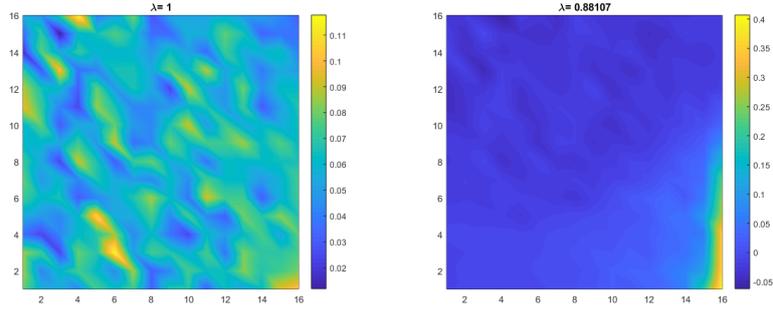


Figure 4: Layer 2, left eigenvectors for $m = 16$, $inc = 50$.

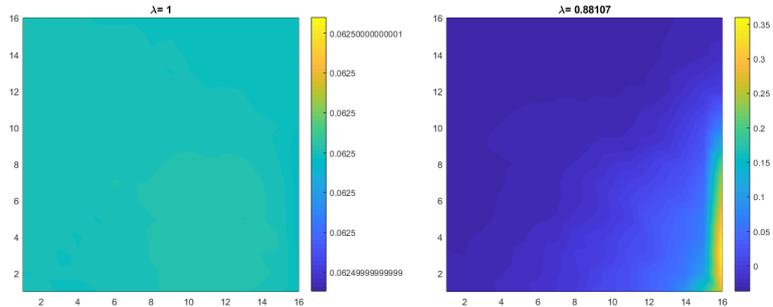


Figure 5: Layer 2, right eigenvectors for $m = 16$, $inc = 50$.

As discussed earlier, the ocean's surface is subject to wind stress and the Coriolis force in order to transport water (and particles) towards the right. As layers descend, there is less wind influence which in turn decreases the magnitude of the transport velocity. However, the force from the above layer pushes causes the sub-layers to again travel to the right (perpetuating the Ekman Transport, just in deeper layers). The change of magnitude and direction of water transport through the layers causes an Ekman spiral. This motion traps sub-surface water to the centre of gyres [12]. Again, the motion in deeper layers is slower, as reflected in the eigenvalues. Figures 5 and 4 exhibit different eigenpictures due to Ekman Transport and Spiraling.

3.2.3 Model B

Model B also creates a row stochastic matrix with leading eigenvalue $\lambda_1 = 1$. By design, this model determines particle movement over $inc = 50$ days for $m = 16$. As seen in Figures 7 and 6 the final uniformity still holds. However, the second eigenvalue is much closer to 1. This means that movement towards uniformity is slower (47 days when particle concentration is in the Southern hemisphere).

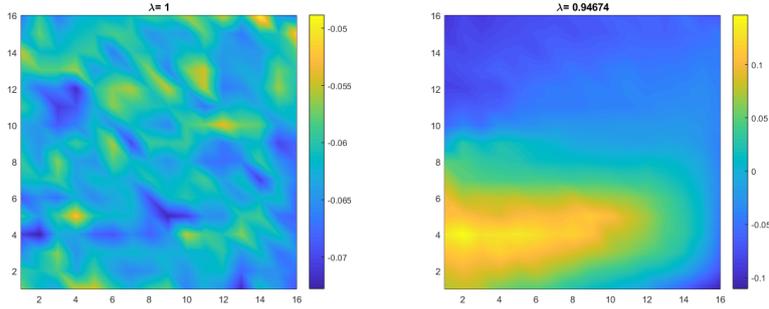


Figure 6: Layer 1, left eigenvectors for $m = 16$, $inc = 50$.

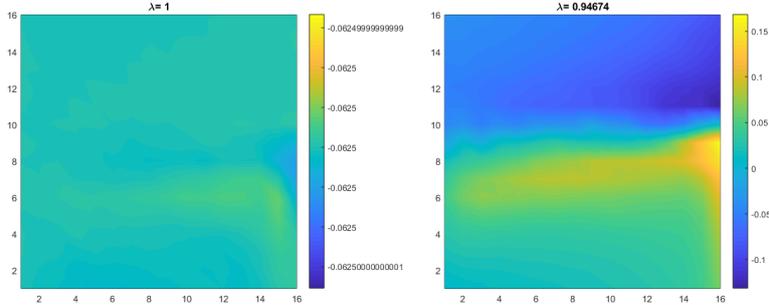


Figure 7: Layer 1, right eigenvectors for $m = 16$, $inc = 50$.

3.3 Connecting Models

Markov chains follow the Chapman-Kolmogorov equations: $(\mathbf{T}_{inc})^x = \mathbf{T}_{inc*x}$. That is, taking the x power of a transport matrix correlating to inc days is the same as computing the transport matrix for movement after $inc*x$ days. Because Model B determines the average inc day dynamics, Chapman-Kolmogorov will not strictly hold. However, it is expected that $\mathbf{T}_A \approx (\mathbf{T}_B)^{matx}$.

To determine the differences between the models, we compute the mean square difference between \mathbf{T}_A and $(\mathbf{T}_B)^{matx}$. As number of boxes increases, the mean square difference decreases. This makes sense; as box diameter approaches 0, the approximate invariant measures generally converge to the true values [4]. Varying inc is more interesting. As inc increases between 1 and 913, the mean square difference grows as well. However, when inc is between 914 and 1828, the mean square difference is 0. This is simply due to the value of $matx$. Once inc goes over 913, $matx = 1$ and \mathbf{T}_A and \mathbf{T}_B are equal. For the earlier inc values, $matx \geq 2$. Since there is less particle exchange between boxes for lower inc values, each intermediate matrix \mathbf{T}_c is more similar to \mathbf{T}_B which results in greater equality between \mathbf{T}_A and $(\mathbf{T}_B)^{matx}$.

3.4 Statistics

While we can determine the statistics for both models, we are more interested in the statistics for Model B for inc values where $matx \geq 2$. Changing the number of boxes (by changing m) will change the statistics of the model. For $m = 10$, the mean percent error is 8.16%. To calculate this, we sum the standard deviation matrix across each row and determine the mean percent error.

While this error is low, it makes sense to increase the number of boxes. This gives more detail to the particle movement but in turn, it will increase the error.

Increase m so that $m = 16$. The matrix values of \mathbf{T}_B against standard deviation is shown in Figure 8. The new mean percent error is 17.49%, which we deem acceptable for our purposes.

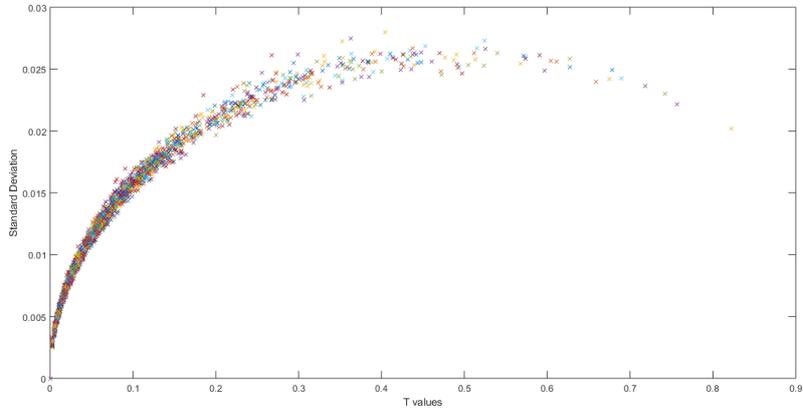


Figure 8: Transport Matrix values against the standard deviation for $m = 16$, $inc = 50$

3.5 Diffusion

Diffusion is the movement of the particles from high density to lower density regions. We compute the mean velocity and diffusivity coefficients as derived in 2.5. Using the velocity coefficients, we create a plot of velocity vectors corresponding to the box locations in Figure 9. In this plot, the centralised jet is clearly visible.

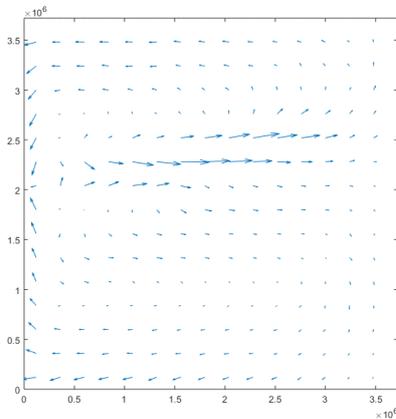


Figure 9: This shows the velocity vectors for Layer 1 over 50 days for $m = 16$. The largest vector magnitude is 0.1959 m/s.

In the advection-diffusion equation, diffusivity has three different components, $\kappa_{1,1}$, $\kappa_{1,2}$, and $\kappa_{2,2}$ which relate to diffusivity in the x , x and y , and y directions respectively. To plot the diffusivity, we first create the aforementioned matrix κ . κ is a 16×16 matrix, where the $(i, j)^{\text{th}}$ entry is the 2×2 matrix,

$$\begin{bmatrix} \kappa_{1,1}(i, j) & \kappa_{1,2}(i, j) \\ \kappa_{2,1}(i, j) & \kappa_{2,2}(i, j) \end{bmatrix}.$$

We then plot the log-10 norm (or log-10 of the largest eigenvalue) for each of these 2×2 matrices. This yields the magnitude of the overall diffusion values from layer 1 as seen in Figure 10, 11, and 12. As the amount of time studied (inc) decreases, so does the region of greatest diffusivity. This is because particles have less time to diffuse across the ocean.

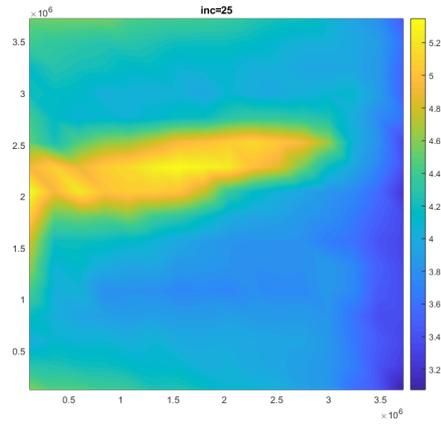


Figure 10: Visualisation of the diffusivity in log-10 scale for a 25 day period in layer 1.

By comparing Figures 9 and 11, we see that the jet (region of strong velocities) coincides with regions of stronger diffusivity which intuitively makes sense.

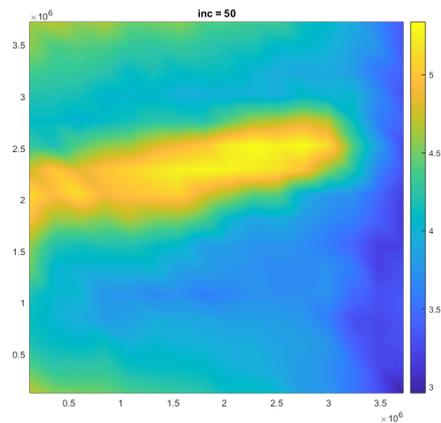


Figure 11: Visualisation of the diffusivity in log-10 scale for a 50 day period in layer 1.

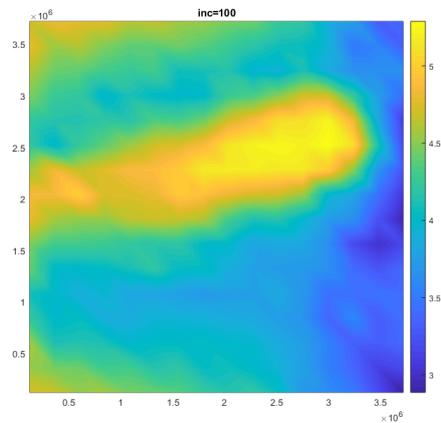


Figure 12: Visualisation of the diffusivity in log-10 scale for a 100 day period in layer 1.

However, the diffusivity computation breaks down when *inc* grows too large. Figure 13 illustrates this breakdown. While we do expect the region of high diffusivity to grow the centralised

jet should remain. In Figure 13, the central jet is subject to too much computational noise, thus invalidating the diffusivity at $inc = 200$. This timescale now allows for dynamics outside of the jet to interfere with the results. Therefore, we do not obtain valid results when inc is too large. In practice, it is best to keep $inc \leq 100$; we choose $inc = 200$ as the breakdown is clearly visible.

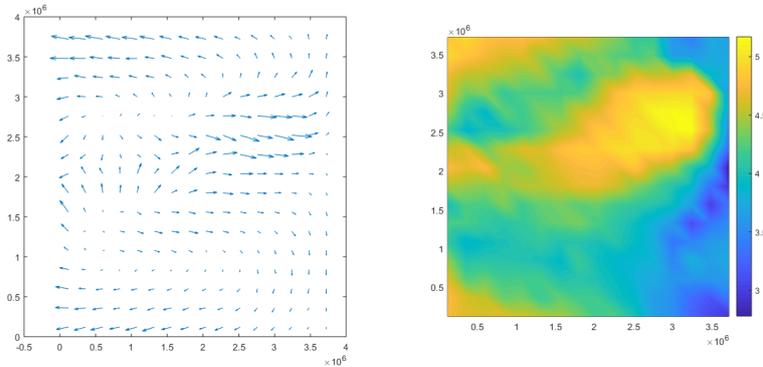


Figure 13: Visualisation of the velocity vectors (maximal vector magnitude = 0.1026 m/s) and diffusivity in log-10 scale for $inc = 200$ and $m = 16$. When inc is too high, the computation breaks down.

4 Acknowledgements

This project would not have been possible without the time and instruction of my supervisor Prof. Jacques Vanneste. I am very grateful to him for helping me pursue my interest in oceanography. Thanks are also due to Tony Ying and Dr. James Maddison for providing the trajectory data.

A Code: Model A

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Preliminaries%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The 3-layer ocean model has 2500 particles over 1828 days.          %
% This code will find the right and left eigenvectors of the Markov   %
% Transport Matrix for a single layer of this model. The right      %
% eigenvectors correspond to backward time dynamics and the left    %
% eigenvectors correspond to forward time dynamics. To calculate the %
% matrix, the ocean surface is subdivided into boxes that particles  %
% move within. This code can be run for any of the 3 ocean layers.  %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all;
load('traj_Npart2500_TempIntv1827_h1_nReal1_L1.mat'); %Load the trajectory data.
                                                    %Ln corresponds to layer n.

x = x/100; y = y/100; %Change the unit from centimeters to meters
[a, b] = size(x); %a=number of time steps (1828 days = 5 years); b=number of particles (2500)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Boxes%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The surface of the ocean model is divided into n=m*m boxes. Each  %
% box is the same size (determined by particle coordinates and number%
% of boxes chosen. To create the transport matrix, we only need to  %
% consider the particle's box locations at every chosen timestep.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

m = 16; %Determine how many boxes the model has. There will be n=m*m boxes.

```

```

n = m*m; %The trasport matrix will be size n by n.
dy = max(max(y))/m; %dy is the north-south length of the box.
dx = max(max(x))/m; %dx is the east-west length of the
                    %box. Both are determined by
                    %finding the maximum coordinate
                    %location and dividing by the
                    %number of boxes per row.
B = zeros(a,b); %B indicates the box location of each particle for every time step.
                %Boxes are numbered so that Box 1 is the lowerleft corner.
                %Box 2 is directly above Box 1. Box m+1 is directly right
                %of Box 1 and Box n is in the upper righthand corner.
for j=1:b
    for i=1:a
        B(i,j) = 1+(floor(y(i,j)/dy)) + (floor(x(i,j)/dx))*m;
    end
end

inc = 50; %Determine the day increments.
s=1; %Determine the start date of the calculations.
matx = floor(a/(inc+s)); %By choosing inc and s, matx determines how many transport matrices
                        % will be calculated. This can also be thought of as how many time
                        % steps will be considered based off an inc and s.
Br = B(s:inc:end,:); %Form the reduced box matrix, Br
                    %by taking the locations every inc days starting from day s.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Transport Matrix %%%%%%%%%
% Each transport matrix has entries: %
% Tc(k,q) = [# of particles in k that end in q]/[# of particles in k] %
% Tc is taken between every inc days. There will be matx of this matrices.%
% The final transport matrix, T, is the product of the matx Tc matrices. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Tc = cell(1,matx);

for k = 1:n
    for q=1:n
        start = (Br == k); %For every particle over the matx timesteps,
                          %return 1 if particle starts in in Box k and 0 otherwise.
        traj = (Br == q); %For every particle over the matx timesteps,
                          %return 1 if particle ends in Box q and 0 otherwise.
        mult = start(1:end-1,:).*traj(2:end,:); %When a particle travels from Box k to Box q
                                                %mult(k,q) = 1.
        top = sum(mult' == 1); %Sums up the number of particles that travel from k to q.
        tot = sum(start(1:end-1,:)'); %Sums up the particles that start in box k.
        P = top./tot; %The ratio between top and tot.
        for i=1:matx %This loop fills in the matx stochastic matrices.
            Tc{i}(k,q) = P(i); %Each of the matx stochastic matrices finds
                              %the probabilities over timestep inc.
        end
    end
end
end

%To form the final stochastic matrix T, the matx Tc matrices are multiplied
%together in a loop.
T = Tc{1,1};
for z = 2:matx

```

```

    T = T*Tc{1,z};
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Eigenvalues/vectors %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% From T, we compute the right and left eigenvectors and %
% corresponding eigenvalues. From there, the real eigenvalues and %
% corresponding vectors are isolated and the values form the %
% vectors are plotted using the image command. The right eigenvectors%
% represent backward time dynamics and the left eigenvectors %
% represent forward time dynamics. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[R,D,L] = eig(T); %In built Matlab command to find the
                %[right eigenvector, eigenvalues, left eigenvectors].
[evals, ind]= sort(diag(D), 'descend'); %Make a vector of eigenvalues for ease of use.
R = R(:,ind);
L = L(:,ind);

%Before plotting the eigenvectors, we isolate the real eigenvalues and
%corresponding right and left eigenvectors.
RvR = zeros(n,n); %First we will return an nxn matrix of eigenvectors.
                %This is similiar to R and L, except that
RvL = zeros(n,n); %any eigenvectors with imaginary parts will be replaced
                %with vectors of 0.

for i=1:n
    if imag(evals(i)) == 0 %When the eigenvalue has no imaginary part,
        RvR(:,i) = R(:,i); %the corresponding eigenvector is the i-th column of RvR,
        RvL(:,i) = L(:,i); %the i-th column on RvL,
        reEvals(i) = evals(i); %and the eigenvalue is the i-th entry of vector reEvals.
    else %When the eigenvalue has an imaginary part,
        RvR(:,i) = 0; %the i-th columns of RvR and RvL are all 0s
        RvL(:,i) = 0;
        reEvals(i) = 0; %and the corresponding reEvals entry is set to 0.
    end
end
end

reEvals = nonzeros(reEvals); %To create a vector of only real eigenvalues,
                %only take the nonzero elements.
RvR = reshape(nonzeros(RvR), [n length(reEvals)]); %The corresponding right eigenvectors
                %are reshaped into the correct
                %format to exclude any full 0
                %vectors.
RvL = reshape(nonzeros(RvL), [n length(reEvals)]); %The same is done for the left eigenvectors.

p = 2; %Choose p; plot the first p eigenvectors.
figure(1) %Figure 1 plots the first p right eigenvectors in one subplot.

for i=1:p
    subplot(1,p,i)

    pcolor(reshape(RvR(:,i), [m m])); shading interp %each entry of the right eigenvector is
                %plotted in boxes 1 through n= m*m
                %by the reshape and image command.

    axis square
    txt = ['\lambda= ' num2str(reEvals(i))]; %label each subplot with the corresponding eigenvalue.
end

```

```

title(txt,'Interpreter','tex')
colorbar %add a color legend to each plot.
end

figure(2) %Figure 2 plots the first p left eigenvectors in one subplot
        %by the same technique as in figure 1.
for i=1:p
subplot(1,p,i)
pcolor(reshape(RvL(:,i), [m m])); shading interp
axis square
txt = ['\lambda= ' num2str(reEvals(i))];
title(txt,'Interpreter','tex')
colorbar

end

\section{Code: Model B}
\begin{verbatim}

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Preliminaries%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The 3-layer ocean model has 2500 particles over 1828 days.          %
% This code will find the right and left eigenvectors of the Markov %
% Transport Matrix for a single layer of this model for a decided %
% increment. The right eigenvectors correspond to backward time %
% dynamics and the left eigenvectors correspond to forward time %
% dynamics. To calculate the matrix, the ocean surface is subdivided %
% into boxes that particles move within. This code can be run for %
% any of the 3 ocean layers. The preliminaries and box construction %
% are the same as model A.                                           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; close all;
load('traj_Npart2500_TempIntv1827_h1_nReal1_L1.mat');
x = x/100; y = y/100;
[a, b] = size(x);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Boxes %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The surface of the ocean model is divided into n=m*m boxes. Each %
% box is the same size (determined by particle coordinates and number%
% of boxes chosen. To create the transport matrix, we only need to %
% consider the particle's box locations at every chosen timestep. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
m = 16;
n = m*m;
dy = max(max(y))/m;
dx = max(max(x))/m;
B = zeros(a,b);
for j=1:b
    for i=1:a
        B(i,j) = 1+(floor(y(i,j)/dy)) + (floor(x(i,j)/dx))*m;
    end
end

inc = 50; %Determine the day increment to find transition probabilities for.

```

```

s=1;
Br = B(s:inc:end,:);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Transport Matrix %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% The transport matrix has entries: %
% T(k,q) = [# of particles in k that end in q]/[# of particles in k]. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for k = 1:n
    for q=1:n
        start = (Br == k); %For every particle over the matx timesteps,
            %return 1 if particle starts in in Box k and 0 otherwise.
        traj = (Br == q); %For every particle over the matx timesteps,
            %return 1 if particle ends in Box q and 0 otherwise.
        mult = start(1:end-1,:).*traj(2:end,:); %When a particle travels from Box k to Box q
            %mult(k,q) = 1.
        top = sum(sum(mult' == 1)); %Sums up the number of particles that travel from
            %k to q for everytime step and particle.
        tot = sum(sum(start(1:end-1,:)')); %Sums up the particles that start in box k
            %for everytime step and particle.
        T(k,q) = top./tot; %The ratio between top and tot.
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Eigenvalues/vectors %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% From T, we compute the right and left eigenvectors and %
% corresponding eigenvalues. From there, the real eigenvalues and %
% corresponding vectors are isolated and the values form the %
% vectors are plotted using the image command. The right eigenvectors%
% represent backward time dynamics and the left eigenvectors %
% represent forward time dynamics. The code is the same as Model A %
% but the time interpretation is different. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[R,D,L] = eig(T);
[evals, ind]= sort(diag(D), 'descend');
R = R(:,ind);
L = L(:,ind);
RvR = zeros(n,n);
RvL = zeros(n,n);

for i=1:n
    if imag(evalv(i)) == 0
        RvR(:,i) = R(:,i);
        RvL(:,i) = L(:,i);
        reEvals(i) = evals(i);
    else
        RvR(:,i) = 0;
        RvL(:,i) = 0;
        reEvals(i) = 0;
    end
end

reEvals = nonzeros(reEvals);
RvR = reshape(nonzeros(RvR), [n length(reEvals)]);
RvL = reshape(nonzeros(RvL), [n length(reEvals)]);

```

```

p = 2;
figure;

for i=1:p
subplot(1,p,i)

pcolor(reshape(RvR(:,i), [m m])); shading interp
axis square
txt = ['\lambda= ' num2str(reEvals(i))];
title(txt,'Interpreter','tex')
colorbar
end

figure;
for i=1:p
subplot(1,p,i)
pcolor(reshape(RvL(:,i), [m m])); shading interp
txt = ['\lambda= ' num2str(reEvals(i))];
title(txt,'Interpreter','tex')
colorbar
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Statistics %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% To help determine the reliability of the model, we compute and plot%
% the standard deviation of Tx = T/N, where Tx is a random variable, %
% T is the transport matrix, and N is the number of particles in %
% Box i. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:n
N(i) = sum(sum(Br == i)); %N(i) gives the total number of times particles enter
                        %Box i over all timesteps.
end
SDv = sqrt(T.*(1-T)./N); %SDv gives the standard deviation for Tx,
                        %where Tx is binomially distributed.

figure;
plot(T,SDv,'x') %Plot the standard deviation against the values from transport matrix T.
ylabel('Standard Deviation');
xlabel('T values');
Pe = mean(sum(SDv)); %returns the mean percent error

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% Diffusion %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% We determine the coefficients for the discrete advection-diffusion %
% equation. First, we must create vectors xc and yc with the centre %
% coordinates for each box. (For example, (xc(1),yc(1)) gives the xy %
% location of box 1). After this, we calculate the coefficients in a %
% single loop. We then reshape all the coefficients into an m x m %
% matrix so that we can plot the coefficients over the coordinates. %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i=1:m
xc(1+m*(i-1):i*m) = (2*i-1)*dx/2; %The x coordinates repeat every m iterations,
                                %increasing by dx every time.
ycp(i) = (2*i-1)*dy/2; %the y coordinates increase by dy every m iterations.

```

```

end
yc = repmat(ycp, [1 m]); %Repeat the size m yc vector m times.

deltinv = 1/(inc*86400); %Our unit, s-1. Note: inc is in days, there are
                        %86400 seconds per day.

%ux, uy, kxx, kyy, and kxy are derived in the report. ux and uy are the
%mean velocities in the x and y direction respectively. kxx, kyy, kxy are
%the diffusivities.
for i=1:n
    ux(i) = deltinu * (xc(i) - xc(:))'*T(:,i);
    uy(i) = deltinu * (yc(i) - yc(:))'*T(:,i);
    kxx(i) = deltinu/2*( ((xc(i)-xc(:)).^2)'*T(:,i)-(ux(i)/-deltinu).^2 );
    kyy(i) = deltinu/2*( (yc(i)-yc(:))'.^2*T(:,i)-(uy(i)/-deltinu).^2 );
    kxy(i) = deltinu/2*( (xc(i)-xc(:))'.^2*T(:,i)-(uy(i)/-deltinu).^2 );
end

%We reshape the coefficients and xc and yc for illustrative purposes.
ux = reshape(ux, [m m]);
uy = reshape(uy, [m m]);
kxx = reshape(kxx, [m m]);
kxy = reshape(kxy, [m m]);
kyy = reshape(kyy, [m m]);
xc = reshape(xc, [m m]);
yc = reshape(yc, [m m]);

%To plot the diffusivities we compute the norm
%(largest eigenvalue) for each 2x2 entry of k.
for i=1:m
    for j=1:m
        kn(i,j) = norm([kxx(i,j), kxy(i,j); kxy(i,j), kyy(i,j)]);
    end
end

%Plot the log-10 of the norm to see the diffusivity
figure;
pcolor(xc,yc,log10(kn)); shading interp; colorbar
axis square

%This plot shows the jet stream by plotting velocity vectors.
figure;
quiver(xc,yc,ux,uy);
axis square

```

References

- [1] J. N. Darroch and E. Seneta. *On Quasi-Stationary Distributions in Absorbing Discrete-Time Finite Markov Chains*. Journal of Applied Probability, 2, 1965.
- [2] Peter Deuffhard, Wilhelm Huisinga, Alexander Fischer, Christof Schütte. *Identification of almost invariant aggregates in reversible nearly uncoupled Markov chains*. Linear Algebra and its Applications, 315, 2000.
- [3] Raffaele Ferrari, David Ferreira. *What processes drive the ocean heat transport?* Ocean Modelling, 38, 2011.
- [4] Gary Froyland. *Statistically optimal almost-invariant sets*. Physica D: Nonlinear Phenomena, 200, 2005.
- [5] Gary Froyland, Oliver Junge, Peter Koltai. *Estimating Long Term Behavior of Flows Without Trajectory Integration: The Infinitesimal Generator Approach*. SIAM Journal on Numerical Analysis, 51, 2013.
- [6] Gary Froyland, Robyn M. Stuart, Erik van Sebille. *How well-connected is the surface of the global ocean?* Chaos, 24, 2014.
- [7] Steven T. Garren, Richard L. Smith. *Estimating the second largest eigenvalue of a Markov transition matrix*. Bernoulli, 6, 2000.
- [8] Masaaki Kijima. *Markov Processes for Stochastic Modeling*. Chapman & Hall, 1997.
- [9] J.R. Maddison, D.P. Marshall, J. Shipton. *On the dynamical influence of ocean eddy potential vorticity fluxes*. Ocean Modelling, 92, 2015.
- [10] J.R. Norris. *Markov Chains*. Cambridge University Press, 1997.
- [11] Paul R. Pinet. *Essential Invitation to Oceanography*. Jones & Bartlett, 2012.
- [12] James F. Price, Robert A. Weller, Rebecca R. Schudlich. *Wind-Driven Ocean Currents and Ekman Transport*. Science, 11 December 1987.
- [13] Christopher L. Sabine, Richard A. Feely, Nicolas Gruber, Robert M. Key, Kitack Lee, John L. Bullister, Rik Wanninkhof, C. S. Wong, Douglas W. R. Wallace, Bronte Tilbrook, Frank J. Millero, Tsung-Hung Peng, Alexander Kozyr, Tsueno Ono, Aida F. Rios. *The Oceanic Sink for Anthropogenic CO₂*. Science, 305, 2004.
- [14] William J. Stewart. *Introduction to the numerical solution of Markov chains*. Princeton University Press, 1994.