Modeling Contaminant Flow in the Puget Sound

Jordan Trinka
Advisor: Eric Sullivan Ph.D.

June 1, 2017

Abstract

In this paper, we mathematically model contaminant flow in a two-dimensional domain of the Puget Sound using a finite element numerical solution to the advection-diffusion equation coupled with a finite difference numerical solution to the Navier-Stokes equations. We offer two models of contaminant flow in this domain, the first uses a Gaussian point source model of contaminant flow. The second model utilizes a Gaussian point source and a constant boundary source. Figure 1 shows the result of our second model after 560 seconds of run time.

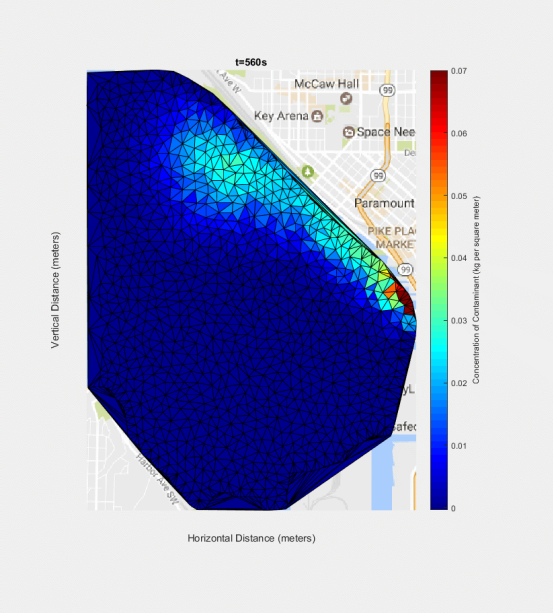

Figure 1: Constant Source Numerical Solution at $t = 560$ seconds

In addition to graphically showing our results, we also provide convergence testing of both our numerical results at the final time-step of both our models and find that the numerical results of our models are converging to an analytic solution at that time-step. We also perform a graphical and numerical sensitivity analysis of our models and find that our numerical solutions to both our models are insensitive to small changes in the diffusivity parameter.

Contents

List of Figures

List of Tables

1 Introduction

The Deepwater Horizon spill of April, 2010 was a massive environmental catastrophe. This spill prompted an intense clean up effort by volunteers, government environmental agencies, and the United States Navy. Areas of the Gulf of Mexico that were damaged by the spill needed to be located and cleaned up. Performing a predictive analysis of contaminant spread in the Gulf before a spill like this would help clean-up crews pinpoint what areas would require the most attention in the event that a spill occurs. Using this analysis, clean-up time would be shortened and clean-up cost would be lowered as crews would not have to spend time locating contaminant that has spread throughout the region.

In this paper, we mathematically model the advection and diffusion of a contaminant on the surface of a portion of the Puget Sound by a finite element numerical solution to the advection-diffusion equation in order to predict what areas of an environment would be most damaged by a spill. The advection-diffusion equation requires us to know the velocity vector field of the water, so, we use a finite difference scheme to solve the Navier-Stokes equations on our domain to provide the velocity field. We use these approximate solutions to create two models of a contaminant spill in the Puget Sound when the tide is going out. One model consists of a point source model of a sinking ship and the other model utilizes a point source and a constant source spill into the domain from a broken pipeline.

We start with an introduction into the finite element method in Section 2 by numerically solving a second order, non-homogeneous, ordinary differential equation. We then show our Puget Sound domain in which we model contaminant flow and offer our treatment of boundary conditions for the two models in Section 3. After this, we discuss our methods used to solve the advection-diffusion equation numerically and we discuss our numerical solution to the Navier-Stokes equations to build our velocity vector field in Sections 4 and 5 respectively. We then offer our results from our two models in Section 6 and perform a sensitivity analysis on both models in Section 7. We finish with a discussion of future work in Section 8 and a conclusion in Section 9.

2 Overview of the Finite Element Method

The purpose of using the finite element method is to approximate the solution to boundary value problems. In particular, the finite element method is often employed to numerically solve partial differential equations on realistic domains where an "analytic solution is not readily available" [3]. We summarize a problem that is solved in [2] in order to introduce the reader to the finite element method. All work in the rest of this section is drawn from [2] and [1].

Consider

$$-u''(x) = f(x) \tag{1}$$

where $u \in H^1(\Omega)$, $f \in L^2(\Omega)$, $u(0) = u(1) = 0$ for the domain, $\Omega = \{x : 0 < x < 1\}$. The space, H^1, is a Sobolev space whose functions and their first derivatives are square-integrable. The L^2 space is the space of all square-integrable functions [3]. By integrating twice, we find there is an analytic solution $u(x)$, however, for purposes of this quick overview, we solve (1) numerically by the finite element method.

We start by getting the weak form of (1) by multiplying both sides by a test function

$v \in H^1(\Omega)$ and integrating both sides to get

$$\int_\Omega -u''(x)\, v(x)\, dx = \int_\Omega f(x)\, v(x)\, dx. \tag{2}$$

We integrate the left-hand side of (2) by parts and use the fact that $u(0) = u(1) = 0$ to get

$$\int_\Omega u'(x)\, v'(x)\, dx = \int_\Omega f(x)\, v(x)\, dx. \tag{3}$$

Next we create a finite set of nodes i along Ω which we use to interpolate $u(x)$, $v(x)$, and $f(x)$ at each node i. In doing this, we also construct the following set of linear, piecewise-continuous, basis functions where each basis function is centered at a node i

$$\eta_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } [x_{i-1}, x_i] \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } [x_i, x_{i+1}] \end{cases}$$

with $i = 1, ..., M$ where M is the number of nodes which we specify along Ω. We see that $\eta_i(x_i) = 1$ and $\eta_i(x_{i-1}) = \eta_i(x_{i+1}) = 0$. From here, we express $u(x)$ and $v(x)$ as a finite dimensional linear combination of the basis functions η_i [3] as shown below

$$\begin{aligned} v(x) &= \sum_{i=1}^{M} \beta_i \eta_i(x) \quad \text{for } i = 1, ..., M \\ u(x) &= \sum_{j=1}^{M} \xi_j \eta_j(x) \quad \text{for } j = 1, ..., M \end{aligned} \tag{4}$$

where $x \in [0, 1]$, $\beta_i = v(x_i)$, and $\xi_j = u(x_j)$. Using these interpolations, we see that (3) becomes

$$\sum_{i=1}^{M} \sum_{j=1}^{M} \xi_j \int_\Omega \eta_j' \eta_i' dx = \sum_{i=1}^{M} \int_\Omega f(x_i)\, \eta_i dx \tag{5}$$

This is a linear system of the form

$$\mathbf{A}\underline{\xi} = \underline{b}. \tag{6}$$

where we define the stiffness matrix, $\mathbf{A}$, as $\mathbf{A}_{i,j} = \int_\Omega \eta_i' \eta_j' dx$ and right-hand side, $\underline{b}$, as $\underline{b}_i = \int_\Omega f(x_i)\, \eta_i dx$ for $i, j = 1, ..., M$.

We compute the elements $\mathbf{A}$ by first noticing that $\eta_i' \eta_j' = 0$ if $|i - j| > 1 \; \forall \; x \in [0, 1]$. This fact causes $\mathbf{A}$ to be tri-diagonal. For this particular example, we partition our domain uniformly with $h_j = \frac{1}{M+1}$. To calculate the main diagonal of $\mathbf{A}$ we calculate

$$\int_\Omega \eta_j' \eta_j' dx = \int_{x_{j-1}}^{x_j} \frac{1}{h_j^2} dx + \int_{x_j}^{x_{j+1}} \frac{1}{h_{j+1}^2} dx = \frac{1}{h_j} + \frac{1}{h_{j+1}} \tag{7}$$

for $j = 1, ..., M$. For the other diagonals, we have

$$\int_\Omega \eta_j' \eta_{j-1}' dx = \int_\Omega \eta_{j-1}' \eta_j' dx = -\int_{x_{j-1}}^{x_j} \frac{1}{h_j^2} dx = -\frac{1}{h_j} \tag{8}$$

for $j = 2, ..., M$. So now (6) becomes

$$\frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & \dots & -1 & 2 \end{bmatrix} \begin{bmatrix} \xi_1 \\ \vdots \\ \vdots \\ \vdots \\ \xi_M \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ \vdots \\ b_M \end{bmatrix} \tag{9}$$

We can now augment stiffness matrix $\mathbf{A}$ with right-hand side $\underline{b}$ and use row reduction to solve for $\underline{\xi}$.

Now that we have an idea of how the finite element method works, we continue into our particular problem where we use the finite element method to numerically solve the advection-diffusion equation in a two-dimensional domain of the Puget Sound. We couple this solution with a finite difference numerical solution to the Navier-Stokes equations which gives us the velocity field when the tide is going out.

3 Puget Sound Domain, Boundary, and Initial Conditions

We start with our Puget Sound domain $\Omega \subset \mathbb{R}^2$ where Ω is a bounded Lipschitz domain that also has a polygonal boundary Γ [1]. Our domain is shown in Figure 2 outlined in black and red.

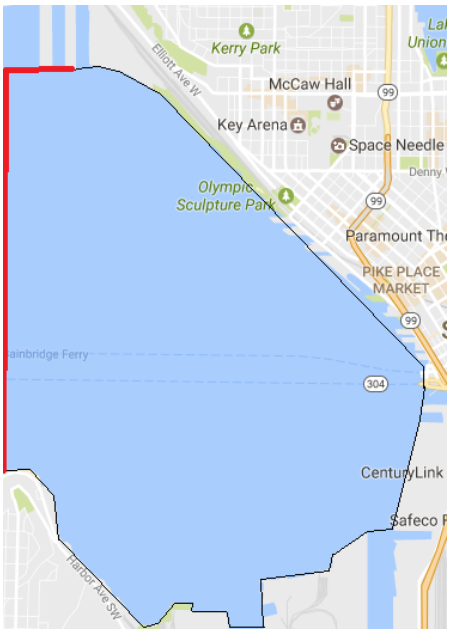

Figure 2: Puget Sound Domain

We apply Neumann boundary conditions, Γ_N, along the boundary outlined in black while the red outlined boundary is Dirichlet, Γ_D, such that $\Gamma = \Gamma_N \cup \Gamma_D$. We create two models of

contaminant flow within the domain. The first model utilizes an arbitrarily chosen point source Gaussian initial condition of contaminant with the function value along the Dirichlet boundary being set to 0. We set the function value along the Dirichlet boundary to be 0 as we are not concerned with modeling contaminant flow into the greater Puget Sound. To ensure contaminant cannot flow onto land, we set the flow normal to the Neumann boundary to be 0. The second model utilizes two sources of contaminant. The first source being a point source Gaussian initial condition of contaminant and the second being a constant source of contaminant along part of the shoreline in order to model a constant spill into Ω. In order to model the spill on the shoreline, we choose to make an arbitrarily chosen portion of the Neumann boundary found in Figure 2, Dirichlet, with a function value that is nonzero. We choose the function value along this portion of the Dirichlet boundary on the shoreline to be close in value to that of the peak of the Gaussian initial condition. The function value along the remaining Dirichlet boundary and the remaining Neumann boundary is set to 0. We start this Gaussian initial condition so that the peak of the Gaussian is close to the shoreline boundary where the constant spill occurs. This new boundary condition is outlined in magenta in Figure 3

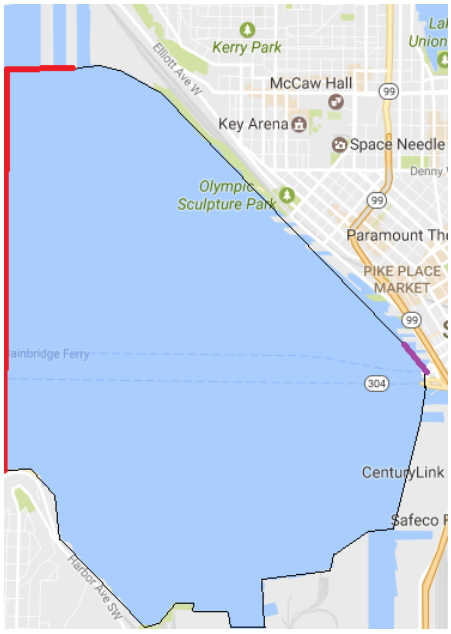

Figure 3: Puget Sound Domain with Non-Homogenous Dirichlet Conditions

4 The Advection-Diffusion Equation

Consider the advection-diffusion equation

$$\frac{\partial u}{\partial t} = D\Delta u + \underline{v} \cdot \underline{\nabla} u + f \tag{10}$$

where $u \in H^1(\Omega)$, the components of $\underline{v}$ are in $H^1(\Omega)$, $f \in L^2(\Omega)$, Δ is the Laplace operator, and D is the coefficient of diffusivity [3]. We see that the left-hand side of (10) is the change in concentration of contaminant with respect to time, Δu models the diffusion of contaminant, $\underline{v} \cdot \underline{\nabla} u$ models the advection of contaminant, and f is a forcing term.

For purposes of this project, we take u to be concentration of contaminant measured in kilograms per square meter, t is time measured in seconds, D is the diffusivity constant measured in square meters per second, $\underline{v}$ is the velocity vector field whose components are measured in meters per second, and $f = 0$. From here we begin our discretization in order to allow us to apply the finite element method to (10).

4.1 Implicit in Time Euler Discretization

Building off of ideas in [1], we discretize (10), in time, with an implicit Euler step to get

$$\frac{U^n - U^{n-1}}{dt} = D\Delta U^n + \underline{v} \cdot \underline{\nabla} U^n \tag{11}$$

where dt is our time-step and U^n is the numerical solution to u at time-step n. Notice we do not include $\underline{v}$ in our discretization since we instead numerically solve for the steady state velocity field before we numerically solve the advection-diffusion equation. We now rearrange (11) to obtain

$$U^n - dtD\Delta U^n - dt\underline{v} \cdot \underline{\nabla} U^n = U^{n-1}. \tag{12}$$

Using (12), we now build the weak form for the finite element method.

4.2 Weak Form

We find the weak form of (12) by multiplying both sides by a test function $W \in H^1(\Omega)$ and integrating both sides to get

$$\int_\Omega U^n W dx - dtD \int_\Omega \Delta U^n W dx - dt \int_\Omega (\underline{v} \cdot \underline{\nabla} U^n) W dx = \int_\Omega U^{n-1} W dx \tag{13}$$

where dx denotes the element of area in $\mathbb{R}^2$. We integrate $\int_\Omega \Delta U^n W$ by parts and we define $g = \frac{\partial u}{\partial m} = 0$ along the Neumann boundary where m is normal to Γ_{N} to obtain

$$\int_\Omega U^n W dx - dtD \left(\int_{\Gamma_{\mathrm{N}}} g^n W ds - \int_\Omega \underline{\nabla} W \cdot \underline{\nabla} U^n dx \right) - dt \int_\Omega (\underline{v} \cdot \underline{\nabla} U^n) W dx = \int_\Omega U^{n-1} W dx \tag{14}$$

where $ds = \underline{m} dx$. We rearrange (14) to get

$$\int_\Omega U^n W dx + dtD \int_\Omega \underline{\nabla} W \cdot \underline{\nabla} U^n dx - dt \int_\Omega (\underline{v} \cdot \underline{\nabla} U^n) W dx = dtD \int_{\Gamma_{\mathrm{N}}} g^n W ds + \int_\Omega U^{n-1} W dx. \tag{15}$$

We now apply the finite element method to approximate U^n and W as linear combinations of basis functions.

4.3 Applying the Finite Element Method

We start by creating a triangularization of Ω, T_Ω, by dividing the domain up into a finite set of triangular elements, T. We create this triangularization such that no edge of each $T \in T_\Omega$

crosses the edge of another triangular element and the elements are made up of three nodes in our domain. We then extend ideas covered in section 2 and we create a finite set of two variable, piecewise-continuous, linear, basis functions centered at each node k such that

$$\eta_k\left(x_i, y_i\right) = \begin{cases} 1 \ \texttt{if} \ \ i = k \\ 0 \ \texttt{if} \ \ i \neq k \text{ for } i, k = 1, ..., N \end{cases}$$

where N is the number of nodes that we specify in our domain Ω.

We now continue our extension of ideas covered in 2 and express W and U^n as a linear combination of basis functions multiplied by a finite approximation of W and U^n at some node $k = 1, ..., N$ to obtain the following

$$\begin{aligned} W\left(x, y\right) &= \sum_{i=1}^{N} \beta_i \eta_i\left(x, y\right), \ \ x, y \in \Omega, \ \text{for } i = 1, ..., N \\ U^n\left(x, y\right) &= \sum_{j=1}^{N} \xi_j^n \eta_j\left(x, y\right), \ \ x, y \in \Omega, \ \text{for } j = 1, ..., N \end{aligned} \tag{16}$$

where $\beta_i = W\left(x_i, y_i\right)$ and $\xi_j^n = U^n\left(x_j, y_j\right)$.

Consider the term $\int_\Omega \underline{\nabla} W \cdot \underline{\nabla} U^n dx$ from (15). If we substitute in our linear combinations, we obtain the following

$$\int_\Omega \underline{\nabla} W \cdot \underline{\nabla} U^n dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \int_\Omega \underline{\nabla}\eta_j \xi_j^n \cdot \underline{\nabla}\eta_i \beta_i dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \xi_j^n \beta_i \int_\Omega \underline{\nabla}\eta_j \cdot \underline{\nabla}\eta_i dx. \tag{17}$$

In order to compute $\int_\Omega \underline{\nabla}\eta_j \cdot \underline{\nabla}\eta_i dx$ we sum the contributions from the different triangular elements that make up each basis function. When we do this we obtain

$$\xi_j^n \beta_i \sum_{T \in T_\Omega} \int_T \underline{\nabla}\eta_j \cdot \underline{\nabla}\eta_i dx, \ \ i, j = 1, ..., N. \tag{18}$$

Therefore, we create a stiffness matrix $\mathbf{A}$ that is as follows

$$\mathbf{A}_{i,j} = \sum_{T \in T_\Omega} \int_T \underline{\nabla}\eta_j \cdot \underline{\nabla}\eta_i dx, \ \ i, j = 1, ..., N. \tag{19}$$

We also express $\int_\Omega U^n W dx$ and $\int_\Omega \left(\underline{v} \cdot \underline{\nabla} U^n\right) W dx$ from (15) as linear combinations of basis functions and their respective weights. We do this by substituting in the discretizations of U^n and W found in (16) for U^n and W in the terms described in the previous sentence. We sum the contributions from the different triangular elements that make up each basis function to get the following expressions

$$\int_\Omega U^n W dx = \sum_{i=1}^{N}\sum_{j=1}^{N} \int_\Omega \eta_j \xi_j^n \eta_i \beta_i dx = \xi_j^n \beta_i \sum_{T \in T_\Omega} \int_T \eta_j \eta_i dx, \ \ i, j = 1, ..., N \tag{20}$$

$$\int_\Omega \left(\underline{v}\cdot\underline{\nabla}U^n\right) W dx = \sum_{i=1}^{N}\sum_{j=1}^{N}\int_\Omega \left(\underline{v}\cdot\underline{\nabla}\eta_j\xi_j^n\right)\eta_i\beta_i dx = \xi_j^n\beta_i \sum_{T\in T_\Omega}\int_T \left(\underline{v}\cdot\underline{\nabla}\eta_j\right)\eta_i dx,\ \ i,j=1,...,N. \tag{21}$$

Therefore, we obtain the following mass matrices $\mathbf{B}$ and $\mathbf{C}$

$$\mathbf{B}_{i,j} = \sum_{T\in T_\Omega}\int_T \eta_j\eta_i dx,\ \ i,j=1,...,N \tag{22}$$

$$\mathbf{C}_{i,j} = \sum_{T\in T_\Omega}\int_T \left(\underline{v}\cdot\underline{\nabla}\eta_j\right)\eta_i dx,\ \ i,j=1,...,N. \tag{23}$$

For the right-hand side of (15), we begin with

$$Ddt\int_{\Gamma_N} g^n W ds + \int_\Omega U^{n-1} W dx. \tag{24}$$

Applying (16), we get

$$Ddt\sum_{i=1}^{N}\sum_{j=1}^{N}\int_{\Gamma_N} g^n\beta_i\eta_i ds + \int_\Omega \xi_j^{n-1}\eta_j\beta_i\eta_i dx \tag{25}$$

where $\xi_j^{n-1} = U^{n-1}(x_j, y_j)$. We sum over the boundary contributions of each triangular element boundary on Γ_N and we sum the contributions of each triangular element for our non-boundary term to get

$$\left(\sum_{E\in\Gamma_N} Ddt\int_E g^n\eta_i ds + \xi_j^{n-1}\sum_{T\in T_\Omega}\int_T \eta_j\eta_i dx\right)\beta_i,\ \ i,j=1,...,N \tag{26}$$

where E is an edge of Γ_N. Now we define

$$\underline{\psi}_i^n = \sum_{E\in\Gamma_N} Ddt\int_E g^n\eta_i ds,\ \ i=1,...,N. \tag{27}$$

We substitute (27), as well as $\mathbf{B}$ for $\sum_{T\in T_\Omega}\int_T \eta_j\eta_i dx,\ \ i,j=1,...,N$, into (26) to obtain

$$\left(\underline{\psi}^n + \mathbf{B}\underline{\xi}^{n-1}\right)\underline{\beta}. \tag{28}$$

Now we define the right-hand side, $\underline{b}^n$, as

$$\underline{b}^n = \underline{\psi}^n + \mathbf{B}\underline{\xi}^{n-1}. \tag{29}$$

We now use our stiffness matrix, mass matrices, and right hand side to rewrite (15) in our discretized, finite element form as follows

$$\left(\underline{\xi}^n\underline{\beta}\mathbf{B} + \underline{\xi}^n\underline{\beta}dtD\mathbf{A} - \underline{\xi}^n\underline{\beta}dt\mathbf{C}\right) = \underline{b}^n\underline{\beta}. \tag{30}$$

This simplifies to

$$\left(\mathbf{B} + dtD\mathbf{A} - dt\mathbf{C}\right)\underline{\xi}^n = \underline{b}^n. \tag{31}$$

The mathematical and coding techniques used to construct $\mathbf{A}$, $\mathbf{B}$, and $\underline{b}$ are covered in [1]. We see that [1] takes care of the diffusive term and offers a treatment of the Neumann and Dirichlet boundary conditions. We now derive our own computation techniques for $\mathbf{C}$ as inspired by [1] which is used to compute the advection term.

We start with (23). Since each basis function is piecewise-continuous and linear, then $\underline{\nabla}\eta_i(x,y)$ is a constant gradient vector of the form

$$\underline{\nabla}\eta_i = \left(C_1\hat{i} + C_2\hat{j}\right), \ i = 1, ..., N \tag{32}$$

where C_1 and C_2 are constants. Since we are using a steady state solution of the Navier-Stokes equations at each node, we state that

$$\underline{v}_i = \left(v_{i_1}\hat{i} + v_{i_2}\hat{j}\right), \ i = 1, ..., N \tag{33}$$

where v_{i_1} is the velocity in the $\hat{i}$ direction at node i and v_{i_2} is the velocity in the $\hat{j}$ direction at node i. We use a tolerance scheme that ensures that each node i in our finite element mesh will have some $\underline{v}_i$ associated with it. So, define R as the set of nodes in our finite element mesh and S as the set of nodes in our finite difference code. Define i_x, κ_x, i_y, and κ_y as being the x,y coordinate pairs for nodes $i \in R$ and $\kappa \in S$ respectively. Next we create an ϵ tolerance scheme such that if $i \in R$ and $\kappa \in S$ are close in x and y value, then we assign i the velocity of κ. Mathematically, this means we choose an ϵ such that if $|i_x - \kappa_x| \leq \epsilon$ and $|i_y - \kappa_y| \leq \epsilon$, then $\underline{v}_i = \underline{v}_\kappa$. In order to ensure that each node $i \in R$ has some sort of velocity associated with it, we utilize a very fine finite difference mesh. This allows us to choose ϵ as small so that the velocity assigned to each node $i \in R$ is close to the velocity of the nodes $\kappa \in S$ that would surround i if i was present in S. Using this technique, we rewrite (23) as

$$\mathbf{C}_{i,j} = \sum_{T \in T_\Omega} \underline{v} \cdot \underline{\nabla}\eta_j \int_T \eta_i dx, \ i, j = 1, ..., N. \tag{34}$$

The integration of each basis function in (34) can be particularly cumbersome to code so we devise a barycentric approximation of the basis functions which allows us to avoid the integrations. Realize that for very fine triangular meshes where R has a sufficiently large number of nodes, the barycentric average of each element that makes up each basis function is close in x and y value to the x and y values of the nodes that make up each triangular element. So let $x_{\text{bc}}, y_{\text{bc}}$ denote the barycentric average of a triangular element that makes up each basis function. We create a fine mesh in our finite element code and evaluate each basis function element at the barycenter and use this as our approximation of the basis function in order to get

$$\eta_i(x,y) \approx \eta_i(x_{\text{bc}}, y_{\text{bc}}) = \frac{1}{3}, \ i = 1, ..., N. \tag{35}$$

Using our barycentric approximation of the basis functions, (34) becomes

$$\mathbf{C}_{i,j} = \sum_{T \in T_\Omega} \frac{\underline{v} \cdot \underline{\nabla}\eta_j}{3} \int_T dx, \ \ i, j = 1, ..., N. \tag{36}$$

From here $\int_T dx$ is just the area formed by each triangular region, A, which by [1] is given as

$$A = \frac{1}{2}\det\left(\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}\right). \tag{37}$$

Therefore, we can now rewrite (23) as being approximated by the following expression

$$\mathbf{C}_{i,j} = \sum_{T \in T_\Omega} \int_T (\underline{v} \cdot \underline{\nabla}\eta_j)\, \eta_i dx \approx \sum_{T \in T_\Omega} \frac{\underline{v} \cdot \underline{\nabla}\eta_j}{6} \det\left(\begin{bmatrix} x_2 - x_1 & x_3 - x_1 \\ y_2 - y_1 & y_3 - y_1 \end{bmatrix}\right), \quad i, j = 1, ..., N. \tag{38}$$

5 Navier-Stokes Velocity Vector Field

To build our velocity vector field, we utilize a finite difference numerical solution to the incompressible Navier-Stokes equations without internal or external forces

$$\frac{\partial \underline{v}}{\partial t} + (\underline{v} \cdot \nabla)\, \underline{v} - \nu \Delta \underline{v} = \underline{0}. \tag{39}$$

where $\frac{\partial \underline{v}}{\partial t}$ is the change in velocity with respect to time, $(\underline{v} \cdot \nabla)\, \underline{v}$ is the advective term, and $\nu \Delta \underline{v}$ is the viscous term which models diffusion of velocity through the fluid. For this project, the components of $\underline{v}$ are measured in meters per second, t is measured in seconds, and ν is the kinematic velocity measured in square meters per second. We now discuss our numerical solution to (39).

5.1 Initial Conditions and Treatment of Boundaries

To begin, we create the initial condition for our velocity vector field as seen in Figure 4 which shows velocity vector field arrows pointing out into the greater Puget Sound. This initial condition is used to build the steady-state velocity field of the Sound when the tide is going out.

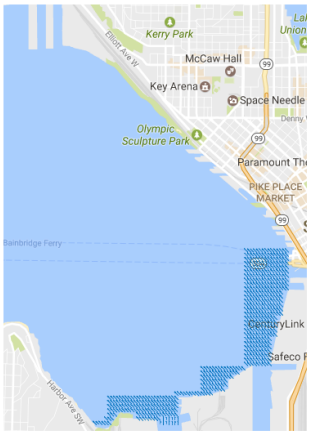

Figure 4: Initial Condition for the Navier-Stokes Equations

The magnitude of each velocity vector at each node κ that was chosen in our initial condition shown in Figure 4 is 3 meters per second which was chosen arbitrarily. We should also note that the vectors in Figure 4 point at 45 degree angles as measured from the horizontal axis.

Since water doesn't travel over land, we implement homogeneous Dirichlet conditions along the black boundary shown in Figure 2 for our finite difference numerical solution to the Navier-Stokes equations. We do this by connecting the boundaries formed in our finite element code to our finite difference code. So, we create a tolerance scheme where if the nodes of our finite element mesh that are on our Neumann boundary are close in x and y value to the x and y value of the nodes in our finite difference mesh, then we enforce Dirichlet boundary conditions along those finite difference code nodes. Mathematically, we begin by defining $i_{\mathrm{N}} \in R$ to be the nodes that connect the edges that form Γ_{N} in our finite element code. Now define i_{N_x}, κ_x, i_{N_y}, κ_y as the x and y coordinates of nodes $i_{\mathrm{N}} \in R$ and $\kappa \in S$ respectively. If $|i_{\mathrm{N}_x} - \kappa_x| < \epsilon$ and $\left|i_{\mathrm{N}_y} - \kappa_y\right| < \epsilon$ for some chosen ϵ, then the velocity vector at node κ is set to $\underline{0}$. We choose ϵ arbitrarily such that we don't zero out too many nodes of our finite difference code but also so that we form boundaries along our shoreline. We now develop a finite difference method to approximate the solution to (39) at our interior nodes.

5.2 MacCormack Discretization

We use a MacCormack discretization to find the numerical solution to (39). This method uses implicit and explicit Euler steps and takes the average. The MacCormack discretization is a second order accurate method and helps resolve stability issues that might arise with numerical solutions.

We begin with the following explicit Euler approximation of the Navier-Stokes equations

$$\begin{aligned}
\frac{\partial v^f_{1_{i,j}}}{\partial t} &\approx -v^n_{1_{i,j}} \frac{\left(v^n_{1_{i+1,j}} - v^n_{1_{i,j}}\right)}{dx_{FD}} - v^n_{2_{i,j}} \frac{\left(v^n_{1_{i,j+1}} - v^n_{1_{i,j}}\right)}{dy_{FD}} + \nu \Delta v^n_{1_{CD}} \\
\frac{\partial v^f_{2_{i,j}}}{\partial t} &\approx -v^n_{1_{i,j}} \frac{\left(v^n_{2_{i+1,j}} - v^n_{2_{i,j}}\right)}{dx_{FD}} - v^n_{2_{i,j}} \frac{\left(v^n_{2_{i,j+1}} - v^n_{2_{i,j}}\right)}{dy_{FD}} + \nu \Delta v^n_{2_{CD}}
\end{aligned} \tag{40}$$

where $v^f_{k_{i,j}}$ denotes the first approximation of the kth component of $\underline{v}$ with $k = \{1, 2\}$ at spatial nodes i, j. We also have that $v^n_{k_{i,j}}$ is the velocity of the kth component of $\underline{v}$ at time-step n, $\Delta v^n_{k_{CD}}$ is the centered-difference approximation to the Laplacian of the kth component of $\underline{v}$ at time step n, and dx_{FD}, dy_{FD} are the spatial steps of our finite difference method in the x and y directions respectively. We now calculate our predictive step, $v^p_{k_{i,j}}$, with the following

$$v^p_{k_{i,j}} = v^n_{k_{i,j}} + \frac{\partial v^f_{k_{i,j}}}{\partial t} dt_{FD} \tag{41}$$

where dt_{FD} is the time-step of our finite difference method. Using this predictive step, we now calculate our second approximations to (39), $\frac{\partial v^s_{k_{i,j}}}{\partial t}$, with the following

$$\frac{\partial v^s_{1_{i,j}}}{\partial t} \approx \left(-v^f_{1_{i,j}} \frac{\left(v^f_{1_{i,j}} - v^f_{1_{i-1,j}}\right)}{dx_{FD}} - v^f_{2_{i,j}} \frac{\left(v^f_{1_{i,j}} - v^f_{1_{i,j-1}}\right)}{dy_{FD}} + \nu \Delta v^f_{1_{CD}} \right) \tag{42}$$
$$\frac{\partial v^s_{2_{i,j}}}{\partial t} \approx \left(-v^f_{1_{i,j}} \frac{\left(v^f_{2_{i,j}} - v^f_{2_{i-1,j}}\right)}{dx_{FD}} - v^f_{2_{i,j}} \frac{\left(v^n_{2_{i,j}} - v^f_{2_{i,j-1}}\right)}{dy_{FD}} + \nu \Delta v^f_{2_{CD}} \right).$$

Using this second approximation to (39), we now calculate our next time-step approximations, $v^{n+1}_{k_{i,j}}$, with

$$v^{n+1}_{k_{i,j}} = v^n_{k_{i,j}} + \left(\frac{\partial v^f_{k_{i,j}}}{\partial t} + \frac{\partial v^s_{k_{i,j}}}{\partial t} \right) \frac{dt_{FD}}{2}. \tag{43}$$

Using this MacCormack method, we run our code until we obtain the steady state of $\underline{v}$ which can be found in Figure 5 which shows how the contaminant travels within our domain while the tide is going out.

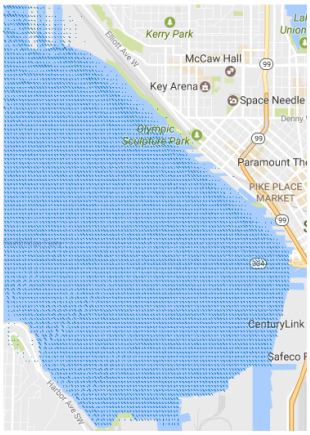

Figure 5: Steady State Solution to the Navier-Stokes Equations

Figure 5 shows velocity field vectors traveling out towards the greater Puget Sound. We see that our tolerance scheme has failed in the upper left-hand corner of Figure 5 which is indicated by velocity field arrows breaking through that boundary due to the lack of Neumann boundary nodes in our finite element code. We also see velocity field arrows breaking through our shoreline boundary in the lower left-hand corner of Figure 5. We would fix this issue by either slightly increasing ϵ or increasing the number of Neumann boundary nodes in our finite element code along that portion of our domain.

6 Results

Using our finite element and finite difference methods, we build our solutions. We multiply each term in our velocity vector field by $\frac{1}{10}$ for stability reasons with $\frac{1}{10}$ being chosen as an arbitrary.

We also choose a diffusion coefficient of 1 for stability reasons with 1 also being chosen arbitrarily.

6.1 Point Source Model

We begin with modeling point source contaminant flow in our domain shown in Figure 2 which models a sinking ship. The following series of snap-shots show the progression of contaminant flow at different times t

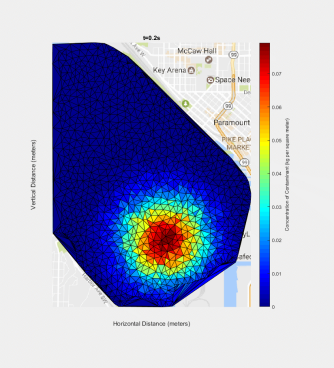

Figure 6: Point Source Numerical Solution at time $t = 0.2$ seconds

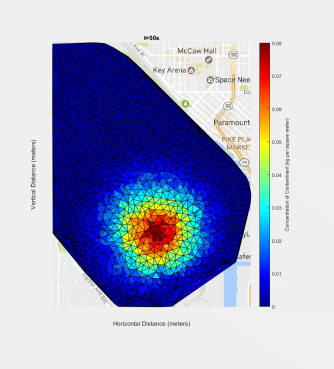

Figure 7: Point Source Numerical Solution at time $t = 50$ seconds

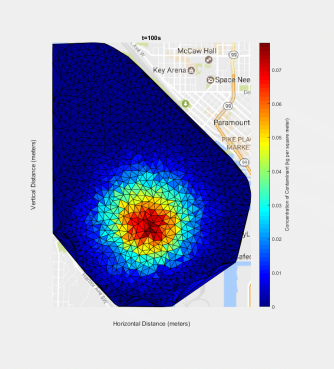

Figure 8: Point Source Numerical Solution at time $t = 100$ seconds

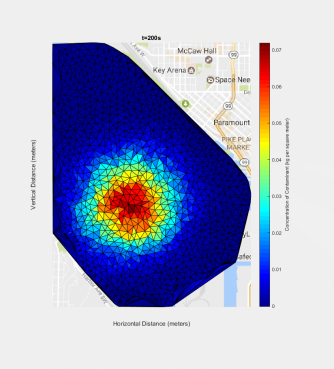

Figure 9: Point Source Numerical Solution at time $t = 200$ seconds

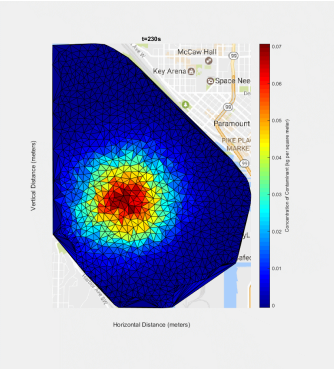

Figure 10: Point Source Numerical Solution at time $t = 230$ seconds

In order to confirm our numerical results we test for the convergence of our numerical solution as well as conservation of concentration since we are not adding or subtracting any contaminant in this first model.

6.2 Convergence Testing for Point Source Model

We perform convergence testing to see if our numerical solution will eventually converge to the analytic solution as we decrease dt. So, we run our point source model code for a 230 seconds of simulated time and then save the resulting final numerical solution into a vector. We do this starting with $dt = 0.1$ and then we decrease dt by a factor of 10 each time we run the code and save the final result. We then measure the Euclidean distance between final result vectors as expressed by

$$\|\underline{U}_{dt_1}^{end} - \underline{U}_{dt_2}^{end}\| \tag{44}$$

where $\underline{U}_{dt_1}^{end}$ is the numerical solution at the final time step of $t = 230$ seconds found using a time-step of dt_1. We also note that $\underline{U}_{dt_2}^{end}$ uses a time-step of dt_2 which is 10 times smaller than dt_1. Our solution converges, if each time we decrease dt by a factor of 10, then the Euclidean distance between numerical solutions at the final time-step should decrease by a factor of 10 as well. When we test this, we obtain Table 1.

Table 1: Convergence Testing for Point Source Model

Time-Steps	Euclidean Distance Between Numerical Solutions
0.1 and 0.01	3.3623×10^{-4}
0.01 and 0.001	3.3656×10^{-5}
0.001 and 0.0001	3.3660×10^{-6}

As we see in Table 1, our numerical solution is converging as we decrease dt by a factor of 10. If we had infinite computing power and could make dt infinitely small, we would see the numerical solution converge to the analytic solution at the final time-step. We also want to test conservation of concentration of contaminant.

6.3 Conservation of Concentration for Point Source Model

Since we are not adding or removing any concentration of contaminant in our point source model, contaminant should be conserved. Therefore, we sum the total concentration of contaminant at each time-step in our numerical solution and plot it to see if there's any significant change in concentration of contaminant. When we do this, we obtain Figure 11.

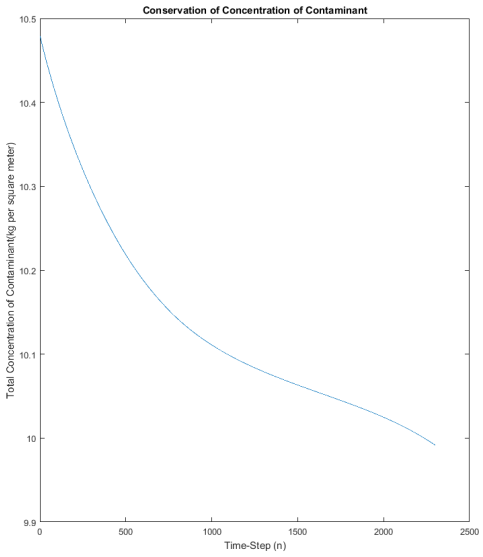

Figure 11: Conservation of Concentration of Contaminant

As we see in Figure 11, the concentration of contaminant is not fully conserved. We now check the rate as to how fast we are losing contaminant as time carries on by employing a second order accurate, centered-difference numerical differentiation scheme in order to find the derivative of the vector shown in Figure 11. When we do this, we obtain Figure 12 which shows the rate of change of contaminant in Ω.

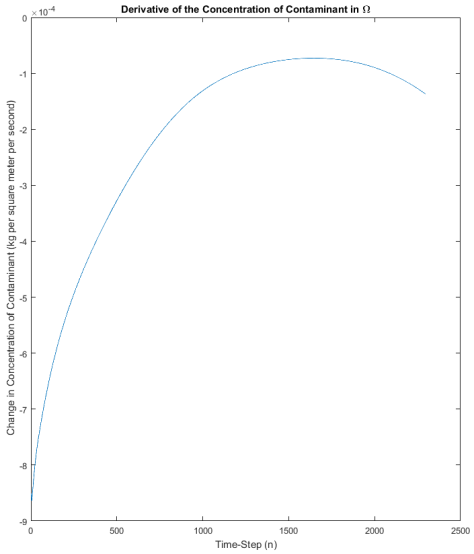

Figure 12: Rate of Change of Concentration of Contaminant in Ω

As we see in Figure 12, the rate of decrease of contaminant is on the order of 10^{-4} for the 230 seconds of simulation time that we ran our program for. For small amounts of run time, this is not an issue. A reason for this decrease in contaminant could be the fact that we use a barycentric approximation of each triangular element that forms each basis function. We would remedy this problem by increasing our number of nodes N and improving the approximations used to derive each mass matrix and the right-hand side. We could also employ a finite volume method to fix this issue.

6.4 Constant Source Model

Using the same velocity vector field and diffusivity of 1 square meter per second that we used in our point source solution, we now model contaminant flow from a constant source along our boundary as shown in magenta in Figure 3. We add a constant amount of concentration of contaminant into our domain each time-step in addition to introducing a Gaussian point source concentration of contaminant at the first time-step close to the magenta boundary in Figure 3. As stated in section 3, we choose the function value of the Gaussian initial condition at the peak to be close in value to the function value along the Dirichlet boundary highlighted in magenta shown in Figure 3.

Long term, we expect a linear growth in the concentration of contaminant as time increases since the rate of growth of concentration of contaminant into our domain is constant. The following images show the contaminant flow for this situation when we run our program for 560 seconds of simulation time. We find that the solution goes heavily unstable as it runs but then as time continues on, the diffusive term takes over and smooths out the solution resulting in stability.

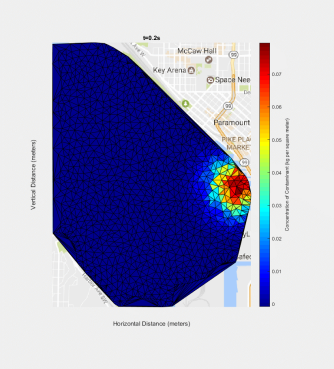

Figure 13: Constant Source Numerical Solution at $t = 0.2$ seconds

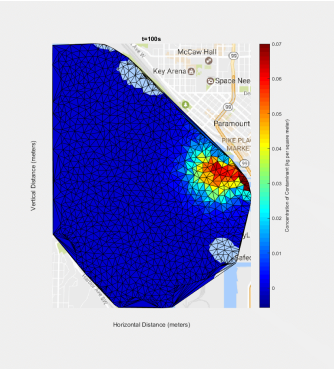

Figure 14: Constant Source Numerical Solution at $t = 100$ seconds

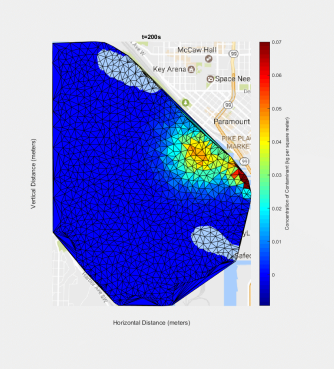

Figure 15: Constant Source Numerical Solution at $t = 200$ seconds

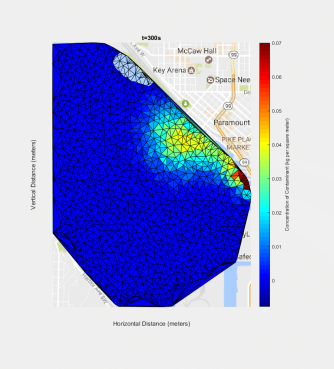

Figure 16: Constant Source Numerical Solution at $t = 300$ seconds

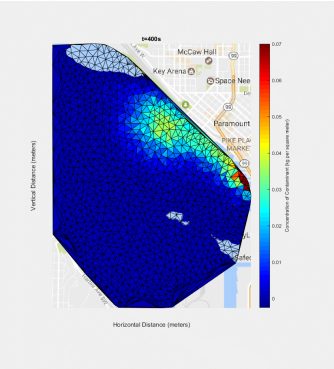

Figure 17: Constant Source Numerical Solution at $t = 400$ seconds

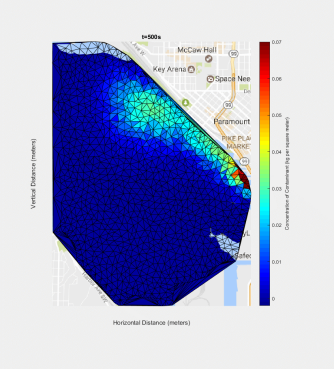

Figure 18: Constant Source Numerical Solution at $t = 500$ seconds

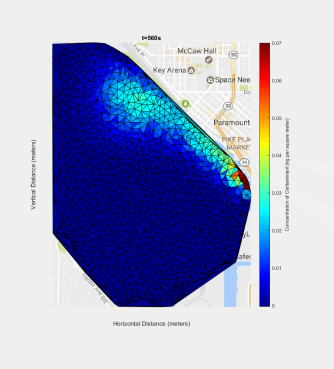

Figure 19: Constant Source Numerical Solution at $t = 560$ seconds

Even though we have stability issues as the simulation continues, we find that we eventually reach stable results. As was the case with our point source model, we test this model for convergence and instead of conservation (since in this case we are adding in contaminant to our domain) we test for a long term linear increase of concentration of contaminant.

6.5 Convergence Testing Constant Source Model

We use the same method for testing convergence in this case as we did when we tested for convergence in our point source model to obtain Table 2.

Table 2: Convergence Testing for Constant Source Model

Time-Steps	Euclidean Distance Between Numerical Solutions
0.1 and 0.01	2.8178×10^{-4}
0.01 and 0.001	2.8262×10^{-5}
0.001 and 0.0001	2.8271×10^{-6}

As we see in Table 2, our numerical solution is converging. We now test for addition of concentration of contaminant into our domain.

6.6 Conservation of Concentration for Constant Source Model

We test for addition of concentration of contaminant into our domain using the same methods as described before with our point source model. In this case, we expect to see a long term linear growth in the concentration of contaminant since the rate at which we add contaminant into our domain is constant. When we test this, we obtain Figure 20.

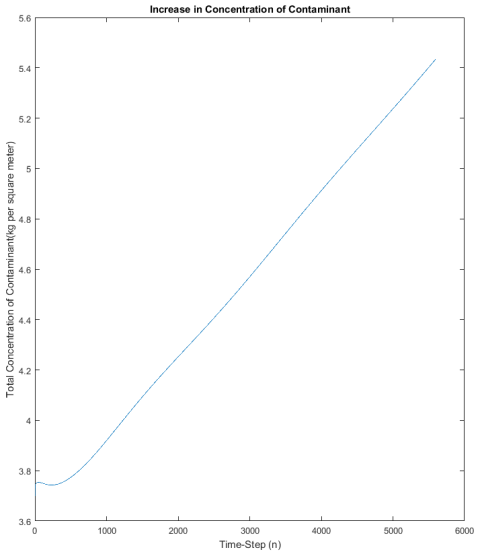

Figure 20: Addition of Concentration of Contaminant in Ω

We see in Figure 20 that as time increases, the total concentration of contaminant in Ω linearly increases. There's some slight instability issues that arise early on that are remedied by the diffusive term as time carries on which leads to a stable increase in concentration of contaminant. We now test the sensitivity of our numerical solutions to the diffusivity parameter, D.

7 Sensitivity Analyses

We test the sensitivity of our numerical results to D by first selecting a set of nodes $i \in R$ such that $145 < i_x < 155$ where i_x is the x coordinate of node i. Each node that is a part of this region is circled in Figure 21.

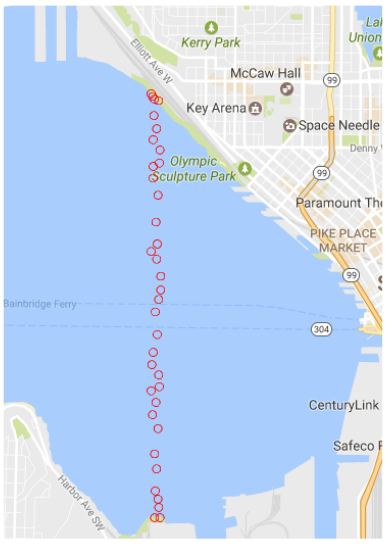

Figure 21: Region Used to Measure Sensitivity of Solutions

We start by measuring the total concentration of contaminant that flows through this region at each time-step with $D = 1$. We then measure the total concentration of contaminant that flows through the region shown in Figure 21 by varying D by $\pm 10\%$ of 1 and compare these results against the numerical solution when $D = 1$.

7.1 Sensitivity Analyses of Point Source Model

We begin with a sensitivity analysis of our point source model. We use the above mentioned method to obtain a graphical sensitivity of our numerical solution to our point source model to obtain Figure 22.

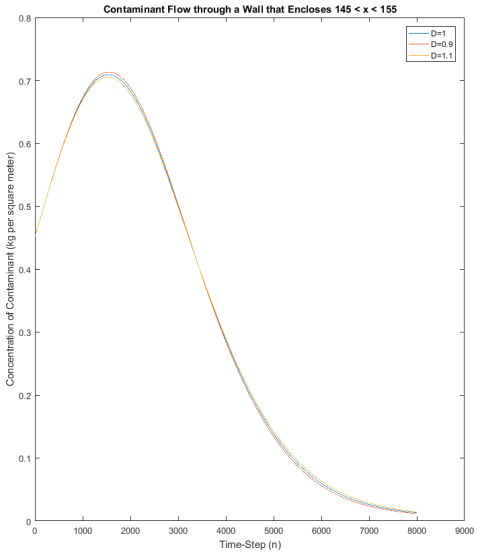

Figure 22: Difference in Numerical Solutions at Different Diffusivities

As we see, there is very little graphical difference between solutions by varying D by $\pm 10\%$.

We now perform a numerical sensitivity analysis to measure the relative change in our numerical solution to our point source model at a time-step n to relative changes in D. So, we utilize the following

$$\mathrm{S} = \frac{\Delta U_{tot}^{n} D}{U_{tot}^{n} \Delta D} \tag{45}$$

where ΔU_{tot}^{n} is the change in the total concentration of contaminant passing through the region shown in Figure 21 at time-step n, between numerical solutions when $D = 1$ and $D = 1.1$ or $D = 0.9$. We also see that ΔD is the change in diffusivity, S is sensitivity (which has no units), and U_{tot}^{n} is the total concentration of contaminant passing through the region shown in Figure 21 at time-step n when $D = 1.1$ or $D = 0.9$. We measure the sensitivity of our numerical solution at the time-step where roughly most of the concentration of contaminant is present because that is the time when most environmental damage would occur to the area shown in Figure 21. According to Figure 22, this time-step appears to be roughly when n is 1900 so we measure the sensitivity of our model to $\pm 10\%$ of the diffusivity coefficient at time-step $n = 1900$. When we do this we obtain the Table 3

Table 3: Sensitivity Point Source Model

Change in Diffusivity	S
$+10\%$	-0.0773
-10%	0.0646

As we see, our point source model in the region shown in Figure 21 is insensitive to small changes in D at the time-step of $n = 1900$.

7.2 Sensitivity Analyses Constant Source Model

We now repeat the same process mentioned in the previous subsection for our constant source model and we first obtain the graphical sensitivity of our numerical solution to our constant source model as shown in Figure 23.

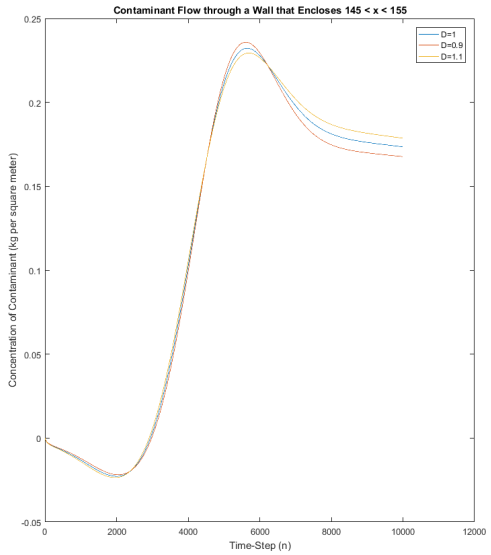

Figure 23: Difference in Numerical Solutions at Different Diffusivities

Graphically, our numerical solution to the constant source model appears to be more sensitive than our numerical solution to the point source model. This being said, we do see that the numerical solution to the constant source model graphically appears to be insensitive to small changes in the diffusivity parameter D.

We repeat our numerical sensitivity analysis for our constant source model at time-step $n = 5900$ (the time-step that appears to have the highest concentration of contaminant flowing through the region shown in Figure 21 as according to Figure 23) and we obtain Table 4.

Table 4: Sensitivity Constant Source Model

Change in Diffusivity	S
$+10\%$	-0.0893
-10%	0.0870

As we see, our constant source model is slightly more sensitive to changes in D than our point source model however, our numerical solution to the constant source model is still insensitive to changes in D.

8 Future Work

The first issue that needs to be addressed is the stability problem. In order to help fix this problem we would implement optimized meshing software to maximize the number of nodes we use in our finite element code. The next way we could improve on the stability issue is to find a way to avoid using a barycentric approximation of the basis functions when we build our stiffness matrix $\mathbf{A}$, mass matrices $\mathbf{B}$, $\mathbf{C}$, and right-hand side $\underline{b}$. After fixing the stability issue, we would then use a realistic diffusivity and magnitude for our velocity vector field. Finally, we want to optimize our tolerance scheme which we use to connect our finite element and finite difference code.

9 Conclusion

Contaminant spills in aquatic ecosystems cause immense damage to the environment. The goal of this paper is to expand on research that has already been done in order to mathematically predict how to most efficiently clean up a spill prior to an accident occurring. In this paper, we were able to give an overview of the finite element method, numerically solve the advection-diffusion equation and Navier-Stokes equations on a Puget Sound domain, offer results, and provide sensitivity analyses of our results. Using the ideas presented in this paper and in Section 8, we can extend on the this work to better model contaminant flow in bodies of water.

10 Works Cited

References

[1] *Remarks around 50 lines of Matlab: short finite element implementation*, Jochen Alberty, Carsten Carstensen and Stefan A. Funken, *Numerical Algorithms 20* (1999), pp. 117-137

[2] *Numerical Solution of Partial Differential Equations by the Finite Element Method*, Claes Johnson, Dover Publications, INC. Mineola, New York (2009) pp. 14-20

[3] Dr. Eric Sullivan, Assistant Professor of Mathematics, Carroll College (2017)