Oil Spill Simulation

Note: I confirm that I did not use codes from the web or from past years' assignments and that the work I submit is my own and my own only.

1 Introduction

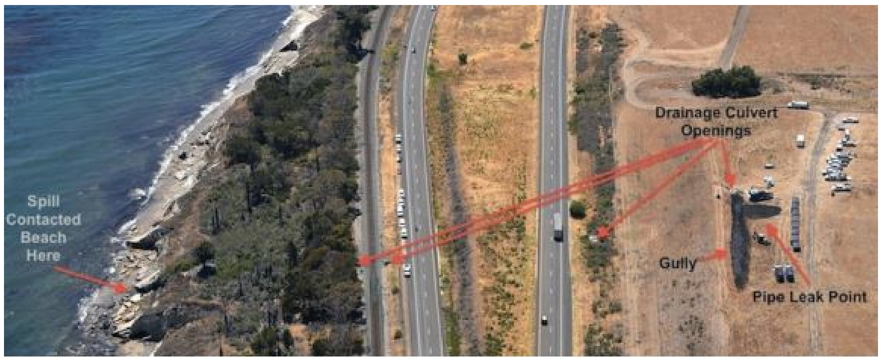


Figure 1: Aerial Photo of Refugio Oil Spill

On May 20, 2015, a broken onshore pipeline near Santa Barbara spewed oil down a storm drain and into the ocean for several hours before it was shut off. Currents and natural diffusion of contaminants are two effects that account for how the oil spreads throughout the ocean. According to health officials, a beach concentration of oil greater than $c_{limit} = 0.006$ is deemed unsafe. When the oil concentration becomes greater than c_{limit}, the city of Santa Barbara will close the beach to keep the public safe.

You can model this situation by a 2D advection-diffusion equation. Consider the domain $\Omega = [x_l; x_r] \times [y_b; y_t]$, a given velocity vector $\vec{v} = (v_x, v_y)$ which represents the ocean's currents, a source term f = f(x,y,t) representing the amount of oil spilling into the ocean, and a concentration c = c(x,y,t) representing the concentration of oil in the ocean. The following is a representation of the 2D advection-diffusion equation:

$$\frac{\partial c}{\partial t} + \vec{v} \cdot \nabla c = D\Delta c + f, \quad \text{for all } (x, y) \in \Omega \tag{1}$$

where D is the rate of diffusion of oil in water, with the initial conditions

$$c(t_{start}, x, y) = 0$$

We are to assume the left, right, and top boundaries of the domain Ω are far enough, so that the oil concentration stays zero at those boundaries during the course of the simulation. We can impose the following Dirichlet boundary conditions:

$$c(t, x, y) = 0, \quad \text{if x} = x_l \text{ or x} = x_r \text{ or y} = y_t$$

At the bottoms boundary of the domain Ω (i.e. the shoreline), the oil concentration has to satisfy the no-flux boundary condition. We can impose the following Neumann boundary condition:

$$D\frac{\partial c}{\partial y} - v_y c = 0, \quad \text{if y} = y_b$$

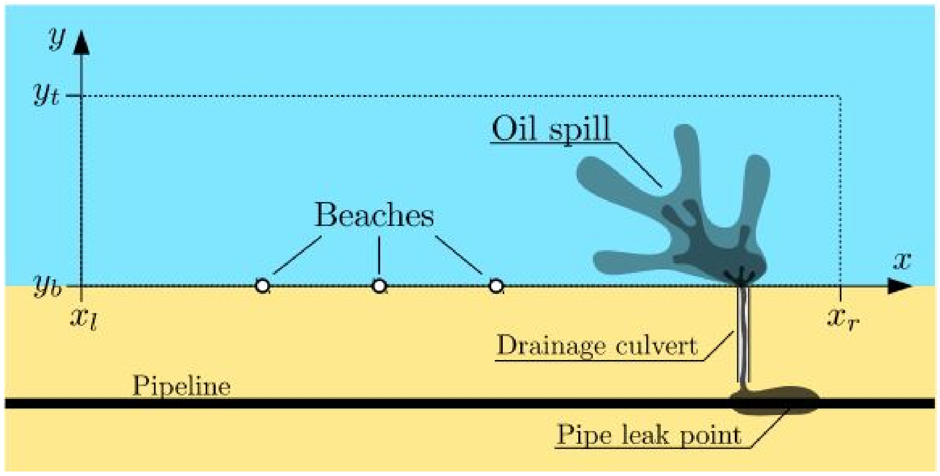


Figure 2: Schematic Representation of the Refugio Oil Spill

The mathematical representation of the problem to be solved is described as:

$$\begin{cases} \textbf{PDE}: \quad \frac{\partial c}{\partial t} + \vec{v} \cdot \nabla c = D\Delta c + f, \quad (x,y) \in \Omega = [x_l; x_r] \times [y_b; y_t] \\ \textbf{BC}: \quad c(t,x,y) = c_{bc}(t,x,y), \quad \text{if x} = x_l, \text{ x} = x_r, \text{ or y} = y_t \\ \qquad\quad D\frac{\partial c}{\partial y} - v_y c = g(t,x,y), \quad \text{if y} = y_b \\ \textbf{IC}: \quad c(t_{start}, x, y) = c_{start}(x,y), \quad (x,y) \in \Omega \end{cases}$$

Where f = f(t,x,y), c_{bc} = c_{bc}(t,x,y), g = g(t,x,y), and c_{start} = c_{start}(x,y) are given functions describing the source term, the boundary conditions, and the inital condition.

The goal of this exercise is to write a MATLAB code to solve this problem. The discretized numerical solution will be compared to the exact solution. Then, the code will be used as a simulation tool to determine when the beaches are safe or unsafe.

2 Discretization of Exact Solution

2.1 Problem 2a.

Write down an approximation for:

$$\frac{\partial c}{\partial t} + \vec{v} \cdot \nabla c = D\Delta c + f$$

using methods learned in class.

2.2 Algorithm

In order to write an approximation for the PDE above, we need to discretize both the advection and diffusion terms in the PDE.

For advection, we need to apply the upwind scheme. The upwind scheme characterizes a class of numerical discretization methods for solving hyperbolic PDEs by using differencing biased in the direction determined by the sign of the characteristic speeds. For stability, the Courant Friedrichs Lewy condition (CFL) should be satisfied:

$$|\frac{c\Delta t}{\Delta x}| \leq 1$$

In order to solve the advection part of this equation, we utilize tools from the Euler Step method which is a numerical method to solve first order first degree differential equation with a given initial value. It is the most basic explicit method for numerical integration of ordinary differential equations and is the simplest Runge Kutta method.

Lets consider the 2D advection case, where there are two different velocity fields v_1 and v_2:

$$\vec{v} \cdot \nabla c = v_x \frac{\partial c}{\partial x} + v_y \frac{\partial c}{\partial y}$$

For the velocity fields, the discretization of $\frac{\partial c}{\partial x}$ and $\frac{\partial c}{\partial y}$ depends on the sign of v_x and v_y. There will be four cases to look at for the velocity fields, which depends on if v_x and v_y are positive or negative. We will use the upwind scheme to approximate the 2D advection, which yields the following equations:

$$v_x \begin{cases} \frac{c^n_{i+1,j} - c^n_{i,j}}{\Delta x} & \text{if } v_x < 0 \\ \frac{c^n_{i,j} - c^n_{i-1,j}}{\Delta x} & \text{if } v_x \geq 0 \end{cases} \qquad v_y \begin{cases} \frac{c^n_{i,j+1} - c^n_{i,j}}{\Delta y} & \text{if } v_y < 0 \\ \frac{c^n_{i,j} - c^n_{i,j-1}}{\Delta y} & \text{if } v_y \geq 0 \end{cases}$$

Now, we can solve for the advection part of equation **(1)** in terms of the first order upwind scheme. Recalling the different cases of velocity signs, we can numerically solve for the term $\vec{v} \cdot \Delta c$:

$$\begin{aligned}
&1.\ v_x \geq 0, v_y \geq 0: \quad && v_x \frac{c^n_{i,j} - c^n_{i-1,j}}{\Delta x} + v_y \frac{c^n_{i,j} - c^n_{i,j-1}}{\Delta y} \\
&2.\ v_x \geq 0, v_y < 0: \quad && v_x \frac{c^n_{i,j} - c^n_{i-1,j}}{\Delta x} + v_y \frac{c^n_{i,j+1} - c^n_{i,j}}{\Delta y} \\
&3.\ v_x < 0, v_y \geq 0: \quad && v_x \frac{c^n_{i+1,j} - c^n_{i,j}}{\Delta x} + v_y \frac{c^n_{i,j} - c^n_{i,j-1}}{\Delta y} \\
&4.\ v_x < 0, v_y < 0: \quad && v_x \frac{c^n_{i+1,j} - c^n_{i,j}}{\Delta x} + v_y \frac{c^n_{i,j+1} - c^n_{i,j}}{\Delta y}
\end{aligned} \tag{2}$$

For diffusion, the terms to be analyzed are the oil concentration diffusion and source term. In multi-variable notation, our terms will look like the following $D\Delta c + f$. When this notation is expanded in the 2D case, the diffusion and source term look like the following:

$$D(\frac{\partial^2 c}{\partial x^2} + \frac{\partial^2 c}{\partial y^2}) + f$$

Where D is the diffusion constant, c is the concentration, and f is the source term. A numerical approximation can be made for this differential equation setup, which looks like the following:

$$D\Delta c + f = D(\frac{c^n_{i+1,j} - 2c^n_{i,j} + c^n_{i-1,j}}{\Delta x^2} + \frac{c^n_{i,j+1} - 2c^n_{i,j} + c^n_{i,j-1}}{\Delta y^2}) + f_{i,j} \tag{3}$$

The last term to be discretized is the partial derivative of oil concentration with respect to time. This term is used when iterating our numerical approximation to find the next index value of our solution.

$$\frac{\partial c}{\partial t} = \frac{c^{n+1}_{i,j} - c^n_{i,j}}{\Delta t} \tag{4}$$

Now that we know how to discretize all the terms of our equation, its time to actually discretize the full equation for the oil spill. Using our knowledge of numerical approximation, we can find a discretized advection-diffusion solution for equation **(1)** below:

$$\frac{c^{n+1}_{i,j} - c^n_{i,j}}{\Delta t} + v_x\frac{c^n_{i+1,j} - c^n_{i,j}}{\Delta x} + v_y\frac{c^n_{i,j+1} - c^n_{i,j}}{\Delta y} = D(\frac{c^n_{i+1,j} - 2c^n_{i,j} + c^n_{i-1,j}}{\Delta x^2} + \frac{c^n_{i,j+1} - 2c^n_{i,j} + c^n_{i,j-1}}{\Delta y^2}) + f_{i,j}$$

Now we isolate $c^{n+1}_{i,j}$, to find the numerical approximation for the next point (n + 1) when iterating our numerical solution:

$$c^{n+1}_{i,j} = c^n_{i,j} + D\Delta t(\frac{c^n_{i+1,j} - 2c^n_{i,j} + c^n_{i-1,j}}{\Delta x^2} + \frac{c^n_{i,j+1} - 2c^n_{i,j} + c^n_{i,j-1}}{\Delta y^2}) + f_{i,j}\Delta t - v_x\Delta t\frac{c^n_{i+1,j} - c^n_{i,j}}{\Delta x} - v_y\Delta t\frac{c^n_{i,j+1} - c^n_{i,j}}{\Delta y}$$

Note: For advection, this numerical approximation assumes both velocity values are negative. This relation was chose because the next two parts of the assignment have negative velocity field values.

3 Maximum Error of Numerical and Exact Solution

3.1 Problem 2b.

Implement the numerical scheme and test your code using the following example:

$$\begin{cases} \Omega = [-1;3] \times [-1.5;1.5] \\ D = 0.7 \\ (v_x, v_y) = (-0.8, -0.4) \\ c_{exact} = sin(x)cos(y)exp(-t) \end{cases}$$

Where Ω is the domain, D is the diffusivity, (v_x, v_y) is the velocity field, and c_{exact} is the exact solution. The initial condition c_{start}(x,y), the boundary conditions c_{bc}(t,x,y) and g(t,x,y), and the source term f(t,x,y) should be deduced from the given exact solution c_{exact}.

Solve the advection-diffusion equation from $t_{start} = 0$ to $t_{final} = 0.2$ using the grid resolutions (N_x, N_y) = (20,15), (40,30), and (80,60). Use the time-step $\Delta t = 0.2\Delta x^2$.

Calculate the maximum error between the numerical solution and the exact solution at t = t_{final}. Then, check that the error gets roughly divided by 2 for each grid refinement.

3.2 Derivation of Equations

In order to solve for the maximum error of the concentration, we need to solve for the inital solution $c_{start}(x, y)$, the boundary condition c_{bc}(t,x,y) and g(t,x,y), and the source term f(t,x,y). All of these equations should be deduced from the exact solution $c_{exact} = sin(x)cos(y)exp(-t)$.

The initial solution is given by the following relation $c_{start}(t_{start}, x, y)$. This equation evaluates the exact solution at time t_{start}, which gives the following equation:

$$c_{start} = (t_{start}, x, y) = sin(x)cos(y)exp(-t_{start}) = sin(x)cos(y)$$

The boundary condition for the left, right, and top walls are given by the exact solution c_{exact}. For this, we impose the Dirichlet boundary condition to obtain the following equation:

$$c_{bc}(t, x, y) = c_{exact}(t, x, y) = sin(x)cos(y)exp(-t)$$

The boundary condition for the bottom wall is given by the no-flux boundary condition. For this we apply the Neumann boundary condition to the balanced flux equation and we get a relation for the term g(t,x,y):

$$\begin{aligned} g(t,x,y) &= D\frac{\partial c}{\partial y} - v_y c \\ &= -Dsin(x)sin(y)exp(-t) - v_y c_{exact}(t,x,y) \end{aligned}$$

The source term equation is found by applying the exact solution to equation **(1)** and solving for the f(t,x,y) term. After taking all partial derivatives and isolating f(t,x,y), you get the following relation for the source term:

$$\begin{aligned} f(t,x,y) &= \frac{\partial c}{\partial t} - D\Delta c + \vec{v}\cdot\nabla c \\ &= (-sin(x)cos(y) + v_x cos(x)cos(y) - v_y sin(x)sin(y) + 2Dsin(x)cos(y))exp(-t) \end{aligned}$$

3.3 Algorithm

In order to solve for the approximated numerical solution of the oil spill concentration, you need to use the discretized version formulated above in *section 2* which is shown below:

$$c_{i,j}^{n+1} = c_{i,j}^n + D\Delta t(\frac{c_{i+1,j}^n - 2c_{i,j}^n + c_{i-1,j}^n}{\Delta x^2} + \frac{c_{i,j+1}^n - 2c_{i,j}^n + c_{i,j-1}^n}{\Delta y^2}) + f_{i,j}\Delta t - v_x\Delta t\frac{c_{i+1,j}^n - c_{i,j}^n}{\Delta x} - v_y\Delta t\frac{c_{i,j+1}^n - c_{i,j}^n}{\Delta y}$$

The next important step is to correctly apply the boundary conditions to the problem's domain. For the top, left, and right wall we apply the Dirichlet boundary condition. This corresponds to $c_{bc} = c_{exact}$, which can be shown implemented in the MATLAB code shown below. For the bottom wall, we apply the Robin boundary condition. Since the oil concentration needs to satisfy conservation law, this means whatever comes in, must come out! Using this relation, you will be able to find a "ghost point" for our discretized equation at indice (i,j-1). Then, plug this ghost point into our discretized equation to find our solution for the bottom wall.

Once all the boundary conditions are satisfied, you can solve for the maximum error of the concentration at t_{final}. The error is found by taking the absolute value of the difference between the numerical and exact solution. This will yield a matrix of values corresponding to the error. Now, to obtain **one** max value, you need to use the MATLAB function max() twice on your error matrix. This will output the maximum error value from the entire matrix of error values.

After calculating the error for one of the grid resolutions, you can find the other grid resolution max errors easily by changing the (N_x, N_y) values. In this case, the grid resolution will be doubled with each refinement.

3.4 Results

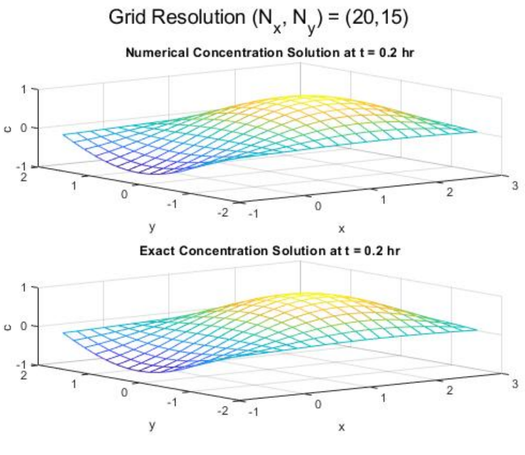
Grid Resolution (N_x, N_y)	Maximum Error
(20,15)	0.019654
(40,30)	0.0096302
(80,60)	0.0047608

The table above shows the maximum error of between eh numerical and the exact solution at t_{final}. The error associated with each case is calculated by taking the absolute value of the difference between the numerical and exact diffusion solution. To obtain the maximum error, you need to apply the max() MATLAB function twice in order to get one maximum error since this is a 2D case. The error is roughly divided by 2 for each grid refinement. This can be seen below for the two cases in this calculation..

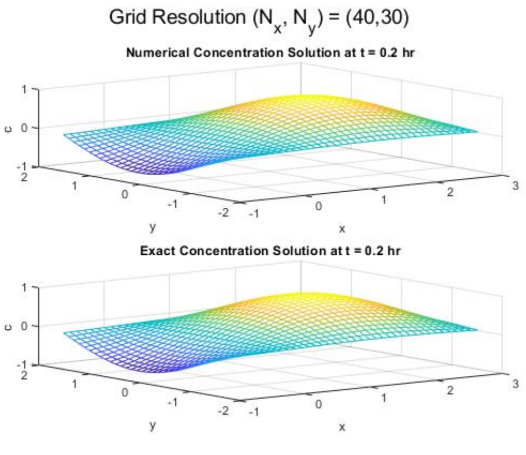
For $(N_x, N_y) = (20, 15)$ to (40,30) and $(N_x, N_y) = (40, 30)$ to (80,60):

$$\begin{aligned}(20,15) \Rightarrow (40,30) \quad \text{error ratio :} \quad \frac{0.019654}{0.0096302} &= 2.0409\ (\sim 2)\\ (40,30) \Rightarrow (80,60) \quad \text{error ratio :} \quad \frac{0.0096302}{0.0047608} &= 2.0228\ (\sim 2)\end{aligned}$$

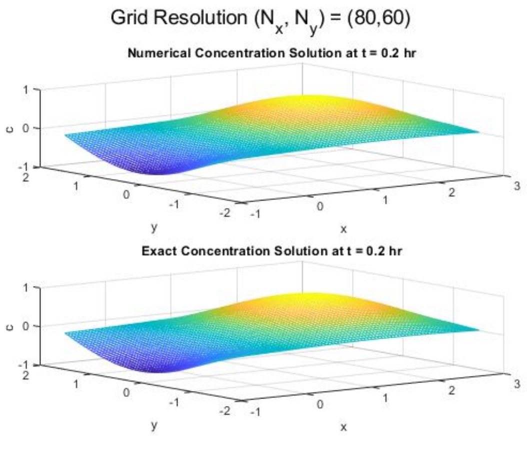
As you can see, this confirms the statement that the error should be roughly one-half the previous grid refinement when increasing the grid resolution by two.



(a) Grid Resolution (20,15)



(b) Grid Resolution (40,30)



(c) Grid Resolution (80,60)

Figure 3: Mesh of Grid Resolution (N_x, N_y) at $t_{final} = 0.2$ s

3.5 MATLAB

```
%ME 17 Final P2b - Alex Nguyen

%I confirm that I did not use codes from the web or fram past years
%assignments and that the work I submit is my own and my own only

clc; clear; close all;

%% Initial Data:
% Domain:
xl = -1; xr = 3; Nx = 20;
yb = -1.5; yt = 1.5; Ny = 15;

% Discretize the and and y axis:
x = linspace(xl, xr, Nx); dx = x(2) - x(1);
y = linspace(yb, yt, Ny); dy = y(2) - y(1);

% Diffusion coefficient:
D = 0.7;

% Velocity Field
vx = -0.8; %x-velocity
vy = -0.4; %y-velocity

% Initial and final time:
t = 0; tfinal=0.2;

% Exact solution:
Exact = @(t,x,y) sin(x)*cos(y)*exp(-t);

% Boundary Conditions:
Cbc = @(t,x,y) Exact(t,x,y);
g = @(t,x,y) -D*sin(x)*sin(y)*exp(-t) - vy*Exact(t,x,y);

% Initial Condition
Cstart = @(x,y) Exact(0,x,y);

% Source Term
```

```
f = @(t,x,y) -sin(x)*cos(y)*exp(-t) + vx*cos(x)*cos(y)*exp(-t) - ...
    vy*sin(x)*sin(y)*exp(-t) + 2*D*sin(x)*cos(y)*exp(-t);

% Initial solution:
cn = zeros(Nx,Ny);
for i = 1:Nx
    for j = 1:Ny
        cn(i,j) = Cstart(x(i),y(j));
    end
end

%% Computation:
% Define the time step:
dt = 0.2*dx*dx;

% Preallocation:
cnp1 = zeros(Nx,Ny);
ce = zeros(Nx,Ny);

% March in time
while t < tfinal
    if t + dt > tfinal
        dt = tfinal - t; %ensures tfinal is reached
    end

    % Use update rule:
    for i = 2:Nx-1
        for j = 2:Ny-1

            cnp1(i,j) = cn(i,j) + D*dt*(cn(i+1,j) - 2*cn(i,j) + cn(i-1,j))/dx/dx ...
                + D*dt*(cn(i,j+1) - 2*cn(i,j) + cn(i,j-1))/dy/dy ...
                + dt*f(t,x(i),y(j)) - dt*vx*(cn(i+1,j) - cn(i,j))/dx - ...
                dt*vy*(cn(i,j+1)-cn(i,j))/dy;

        end
    end

    % Apply Boundary Conditions
    for j = 1:Ny
```

```
        cnp1(1,j)  = Cbc(t+dt,xl,y(j)); %Left Boundary
        cnp1(Nx,j) = Cbc(t+dt,xr,y(j)); %Right Boundary

    end


    for i = 1:Nx

        cnp1(i,Ny) = Cbc(t+dt,x(i),yt); %Top Boundary

    end

    for i = 2:Nx-1

        cnyo = cn(i,2) - 2*dy*(g(t,x(i),yb) + vy*cn(i,1))/D; %Ghost Value

        cnp1(i,1) = cn(i,1) + D*dt*(cn(i+1,1) - 2*cn(i,1) + cn(i-1,1))/dx/dx ...
            + D*dt*(cn(i,2) - 2*cn(i,1) + cnyo)/dy/dy ...
            + dt*f(t,x(i),yb) - dt*vx*(cn(i+1,1) - cn(i,1))/dx - ...
            dt*vy*(cn(i,2) - cn(i,1))/dy; %Bottom Boundary
    end

    % Update time and prepare for next iterate:
    t = t + dt;
    cn = cnp1;

    % Exact solution:
    for i = 1:Nx
        for j = 1:Ny
            ce(i,j) = Exact(t,x(i),y(j));
        end
    end

end

% Plots
subplot(2,1,1)
mesh(x,y,cn')
```

```
title('Numerical Concentration Solution at t = 0.2 hr')
xlabel('x'); ylabel('y'); zlabel('c');
subplot(2,1,2);
mesh(x,y,ce');
title('Exact Concentration Solution at t = 0.2 hr')
xlabel('x'); ylabel('y'); zlabel('c');
z = sprintf('Grid Resolution (N_x,N_y) = (%d, %d)',Nx,Ny);
suptitle(z)

% Error
error = max(max(abs(cn - ce)));
disp(['Maximum Error: ' num2str(error)])
```

4 Concentration of Oil Spill Over Time

4.1 Problem 2c.

Use your code to simulate the spreading of the oil in the ocean using the following parameters:

$$\begin{cases} \Omega = [0;12] \times [0;3] \\ D = 0.2 \\ (v_x, v_y) = (-0.8, -0.4) \\ c_{start}(t,x,y) = 0 \\ c_{bc}(t,x,y) = 0 \\ g(t,x,y) = 0 \\ f(t,x,y) = \begin{cases} \frac{1}{2}(1 - \tanh(\frac{\sqrt{(x-x_s)^2+y^2}-r_x}{\epsilon})), & \text{if t} < 0.5 \\ 0, & \text{if t} > 0.5 \end{cases} \end{cases}$$

Ω is the domain, D is the diffusivity, and (v_x, v_y) is the velocity field. The initial condition is c_{start}(t,x,y), and the boundary conditions are c_{bc}(t,x,y) and g(t,x,y). The source term f(t,x,y) is given in this problem, but is subject to change depending on the time value present.

Solve the advection-diffusion equation from $t_{start} = 0$ to $t_{final} = 10$ using $N_x = 80$ grid points in the x-direction and $N_y = 20$ grid points in the y-direction. Let the time step be $\Delta t = 0.2\Delta x^2$. Where values $x_s = 10$, $r_s = 0.1$, and $\epsilon = 0.1$.

Plot the oil concentration vs time at the location of the three beaches, i.e. at (i,j) = (20,1), (40,1), and (60,1). Determine the time periods at which each of the three beaches should be closed. Take a snapshot of the oil concentration at t = 1, 4, and 7 hours (use MATLAB commands *mesh* and *contourf*).

4.2 Algorithm

In order to simulate the spreading of the oil in the ocean, you must known your parameters first. The inital and boundary conditions are all zero for all values of t, x, and y. The only nonzero equation is the source term, but only for a small period of time after t_{start}. Then, the source term goes to zero eventually.

As done previously, in order to numerically approximate the oil spill concentration, you need to use the discretized version of the oil spill equation formulated in *section 2* which is shown below as reference again.

$$c_{i,j}^{n+1} = c_{i,j}^{n} + D\Delta t\left(\frac{c_{i+1,j}^{n} - 2c_{i,j}^{n} + c_{i-1,j}^{n}}{\Delta x^2} + \frac{c_{i,j+1}^{n} - 2c_{i,j}^{n} + c_{i,j-1}^{n}}{\Delta y^2}\right) + f_{i,j}\Delta t - v_x\Delta t\frac{c_{i+1,j}^{n} - c_{i,j}^{n}}{\Delta x} - v_y\Delta t\frac{c_{i,j+1}^{n} - c_{i,j}^{n}}{\Delta y}$$

The next important step is to apply the boundary conditions for our simulation. For the top, left, and right wall the problem states the boundaries of the domain Ω are far enough, so that the oil concentration stays zero throughout the course of the simulation.

$$c_{bc}(t,x,y) = 0 \quad \text{for x} = x_l,\ \text{x} = x_r,\ \text{or y} = y_t$$

For the bottom wall of the domain Ω, the oil spill concentration needs to satisfy the no-flux condition boundary condition which looks like the following:

$$D\frac{\partial c}{\partial y} - v_y c = 0 \quad \text{for y} = y_b$$

The last step for applying boundary conditions is figuring out what the values to set your corners in the computational domain. Luckily, in this case, all the corners are equal to zero due as stated in the problem statement. This makes life easy, and we can apply the Dirichlet boundary conditions at each of the four corners which yield a concentration of zero.

The next step is to plot the oil concentration vs time at the location of the three beaches, i.e. (i,j) = (20,1), (40,1), and (60,1). This is done by preallocating variables which to be able to store the numerical value of the concentration at each value (i,j) throughout each iteration. It is also important to make an array for the time values of these concentration arrays. Now, we are able to plot each of individual beach's oil concentration with respect to time.

To determine the time periods at which each of the three beaches should be closed, you need to use logic statements or look at the array of data after the simulation is completed. The logic statement method is shown in my MATLAB code, but I also looked at my oil concentration and time data to ensure the correctness of data. When creating logic statements to display the availability of the beach, you need to set your $c_{limit} = 0.006$. You can then proceed to write if statements for when the concentration of the beach is less than c_{limit} at index = i **and** for when the concentration of the beach is greater than or equal to c_{limit} at a index = i + 1. With these two if-statements met, you will then want to output the time value of the while loop to know when the beaches should be closed.

The same process can be done for deciding when a beaches should open, except for this case you would want to look at the opposite case. You can then proceed to write if statements for when the concentration of the beach is greater than or equal to c_{limit} at index = i **and** for when the concentration of the beach is less than than c_{limit} at a index = i + 1. With these two if-statements met, you will then want to output the time value of the while loop to know when the beaches can open again.

Also, you could observe how the oil spill concentration advects and diffuses in the ocean by using the mesh and contourf functions in MATLAB. In order to see the progression over time, you would put these commands in the while loop with the pause(dt) command. For sake of time, I left the mesh and contourf commands outside my while loop to obtain the snapshots of the oil concentration at t = 1, 4, and 7.

4.3 Results

	Beach 1	Beach 2	Beach 3
(i,j)	(20,1)	(40,1)	(60,1)
closed	t = 6.3174 hr	t = 3.2810 hr	t = 0.5030 hr
open	t = N/A	t = 8.9201 hr	t = 3.7563 hr

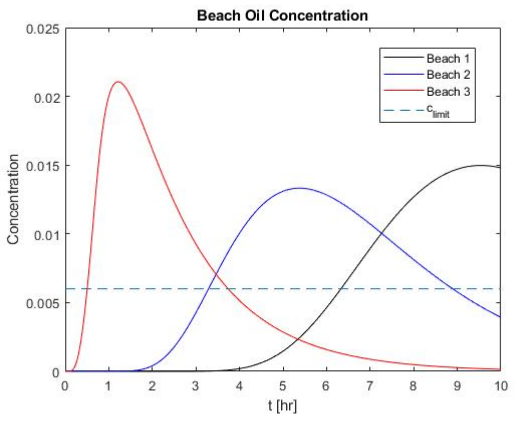
The calculated time values for when the beaches should be closed or opened are shown above. The unsafe oil concentration limit chosen for when a beach should close is given by $c_{limit} = 0.006$. Once the concentration value goes above this value, city health officials say the beach should be closed to the public. The reverse is also true, once the beach concentration value goes below 0.006 the beach should be safe to be opened again.

Using logic statements in MATLAB, you can find the time values for which the beaches will need to be closed or when they can be opened again. Also, this can be done by looking at the array of concentration values for beach 1, 2, and 2. The calculated time values are rough estimates for the opening and closing the beach values. You'll notice one beach will not be able to open within the time span of t_{start} to t_{final}. This is because its concentration value is still above 0.006 at 10 hrs when the simulation stops. If t_{final} were to be increased from 10 hours to some greater value, at some point beach 1 will reopen.

You notice that beach 3 closes first, then beach 2, and then beach 1. This is due to how I define the beach values corresponding to the indices. The closure times make sense because the first beach to be closed should be the farthest right in the computation domain, which is beach 3 (60,1). Then the next beach to be closed should be 2 then beach 1. Likewise, the opening of the beaches make sense. Since beach 3 was the first to close, it should be the first to open. Then, beach 2 should open. Beach 1 didn't have enough time to meet the safety standard to be opened, but at some later point in time beach 1 will be safe to be reopened.

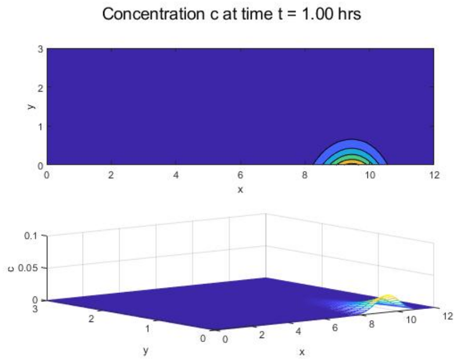
Also, the snapshots of the oil concentration at the times 1, 4, and 7 hours can be seen below. You can see that as time progresses the oil concentration will have a wider width and spread to the left boundary of the water grid. This makes sense intuitively for me, because as time progresses I expect the fluid to diffuse throughout the ocean. This is due to the advection-diffusion process of the oil in the ocean, which causes the oil to spread out according to this model. The mesh and contourf plot output is due to the source term f(t,x,y), since the inital and boundary conditions are both zero. So the only term affecting the plots is the source term.

Note: This solution is for the specified grid points of (N_x, N_y) = (80,20). If the grid points were to change, then accuracy of the solution will change as well. In general, a more accurate solution arises when you create a finer grid point resolution in the x-direction and y-direction. In this problem, the grid points stay constant and no maximum error calculation is required.

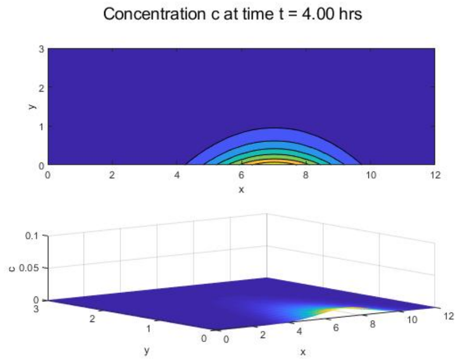


(a) Beach Concentration

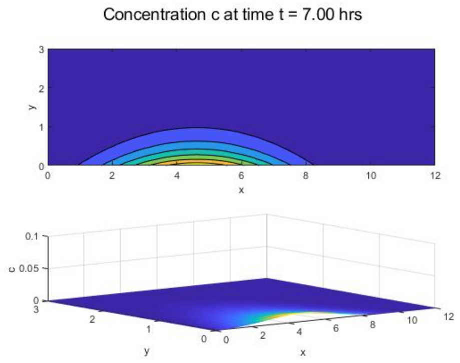
Figure 4: Oil Concentration vs Time at the Location of the Three Beaches



(a) Oil Concentration at t = 1 hrs



(b) Oil Concentration at t = 4 hrs



(c) Oil Concentration at t = 7 hrs

Figure 5: Oil Concentration at Different Times

4.4 MATLAB

```
%ME 17 Final P2c - Alex Nguyen

%I confirm that I did not use codes from the web or from past years
%assignments and that the work I submit is my own and my own only

clc; clear; close all;

%% Initial Data:
% Domain:
xl = 0; xr = 12; Nx = 80;
yb = 0; yt = 3; Ny = 20;

% Discretize the and and y axis:
x = linspace(xl, xr, Nx); dx = x(2) - x(1);
y = linspace(yb, yt, Ny); dy = y(2) - y(1);

% Diffusion coefficient:
D = 0.2;

% Velocity Field
vx = -0.8; %x-velocity
vy = -0.4; %y-velocity

% Initial and final time:
t = 0; tfinal = 10;

% Exact solution:
Exact = @(t,x,y) sin(x)*cos(y)*exp(-t);

% Boundary Conditions:
Cbc = @(t,x,y) 0;
g = @(t,x,y) 0;

% Initial Condition
Cstart = @(t,x,y) 0;

% Source Term Constants
```

```
xs = 10; rs = 0.1; ep = 0.1;

% Initial solution:
cn = zeros(Nx,Ny);
for i = 1:Nx
    for j = 1:Ny
        cn(i,j) = Cstart(t,x(i),y(j));
    end
end

%% Computation:
% Define the time step:
dt = 0.2*dx*dx;

% Preallocation:
cnp1 = zeros(Nx,Ny);
ce = zeros(Nx,Ny);
f = zeros(Nx,Ny);
cn1 = zeros(length(0:dt:tfinal),1);
cn2 = zeros(length(0:dt:tfinal),1);
cn3 = zeros(length(0:dt:tfinal),1);
tn = zeros(length(0:dt:tfinal),1);

n = 1; %initial counter value

% March in time
while t < tfinal
    if t + dt > tfinal
        dt = tfinal - t; %ensures tfinal is reached
    end

    % Source Term
    if t < 0.5
        f = @(t,x,y) 0.5*(1 - tanh((sqrt((x - xs)^2 + y^2) - rs)/ep));
    elseif t > 0.5
        f = @(t,x,y) 0;
    end

    % Use update rule:
```

```
for i = 2:Nx-1
    for j = 2:Ny-1

        cnp1(i,j) = cn(i,j) + D*dt*(cn(i+1,j) - 2*cn(i,j) + cn(i-1,j))/dx/dx ...
            + D*dt*(cn(i,j+1) - 2*cn(i,j) + cn(i,j-1))/dy/dy ...
            + dt*f(t,x(i),y(j)) - dt*vx*(cn(i+1,j) - cn(i,j))/dx - ...
            dt*vy*(cn(i,j+1)-cn(i,j))/dy;

    end
end

% Left and Right Boundary
for j = 1:Ny

    cnp1(1,j)  = Cbc(t+dt,xl,y(j)); %Left
    cnp1(Nx,j) = Cbc(t+dt,xr,y(j)); %Right

end

% Top Boundary
for i = 1:Nx

    cnp1(i,Ny) = Cbc(t+dt,x(i),yt); %Top

end

% Bottom Boundary
for i = 2:Nx-1

    cnyo = cn(i,2) - 2*dy*(g(t,x(i),yb) + vy*cn(i,1))/D; %Ghost Value

    cnp1(i,1) = cn(i,1) + D*dt*(cn(i+1,1) - 2*cn(i,1) + cn(i-1,1))/dx/dx ...
        + D*dt*(cn(i,2) - 2*cn(i,1) + cnyo)/dy/dy ...
        + dt*f(t,x(i),yb) - dt*vx*(cn(i+1,1) - cn(i,1))/dx - ...
        dt*vy*(cn(i,2) - cn(i,1))/dy; %Bottom
end

% Bottom Left and Bottom Right Corner
for j = 1:Ny
```

```
        cnp1(1,1)  = Cbc(t+dt,xl,yb); %Bottom Left
        cnp1(Nx,Ny) = Cbc(t+dt,xr,yb); %Bottom Right

    end

    % Beach Concentrations and Time
    cn1(n) = cn(20,1); %Beach 1
    cn2(n) = cn(40,1); %Beach 2
    cn3(n) = cn(60,1); %Beach 3
    tn(n) = t;

    % Update values and prepare for next iterate:
    n = n + 1; %Counter
    t = t + dt; %Time
    cn = cnp1; %Concentration

end

% Determine When the Beaches Should Be Closed!
for i = 1:length(tn)-1

    if cn1(i) < 0.006 && cn1(i+1) >= 0.006
        disp(['Beach 1 at (20,0) should be closed at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

for i = 1:length(tn)-1

    if cn2(i) < 0.006 && cn2(i+1) >= 0.006
        disp(['Beach 2 at (40,0) should be closed at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

for i = 1:length(tn)-1
```

```
    if cn3(i) < 0.006 && cn3(i+1) >= 0.006
        disp(['Beach 3 at (60,0) should be closed at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

% Determine When the Beaches Should Be Opened!
for i = 1:length(tn)-1

    if cn1(i) >= 0.006 && cn1(i+1) < 0.006
        disp(['Beach 1 at (20,0) should be opened at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

for i = 2:length(tn)

    if cn2(i) >= 0.006 && cn2(i+1) < 0.006
        disp(['Beach 2 at (40,0) should be opened at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

for i = 2:length(tn)

    if cn3(i) >= 0.006 && cn3(i+1) < 0.006
        disp(['Beach 3 at (60,0) should be closed at time ' num2str(tn(i)) ' hrs']);
        break
    end

end

% Concentration Plot Over Time
figure(1);
subplot(2,1,1)
```

```
contourf(x,y,cn')
xlabel('x'); ylabel('y'); zlabel('c');
subplot(2,1,2)
mesh(x,y,cn')
axis([xl xr yb yt 0 0.1])
xlabel('x'); ylabel('y'); zlabel('c');
z = sprintf('Concentration c at time t = %4.2f hrs',t);
suptitle(z)

% Oil Concentration vs Time at the Three Beaches
figure(2)
plot(tn,cn1,'k',tn,cn2,'b',tn,cn3,'r');
hold on;
plot([0 tfinal],[0.006 0.006],'--')
title('Beach Oil Concentration') %(i,j) = (60,0)
ylabel('Concentration')
xlabel('t [hr]')
legend('Beach 1','Beach 2','Beach 3','c_{limit}','location','best')
```