Advection-Diffusion and Oil Spills

1 Introduction

Fluids move throughout other fluids by two processes: advection and diffusion. For example, when oil is spilled in the ocean, the oil is carried along by ocean currents, and diffuses naturally throughout the water as well. Thus, the contamination levels of beaches near an oil pipeline leak can be modeled using an effective advection-diffusion equation solver.

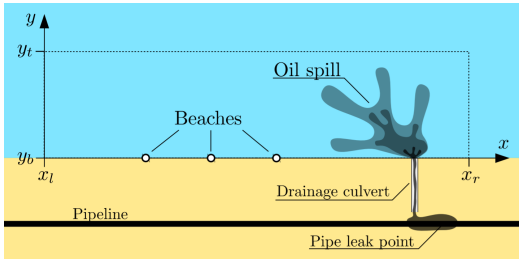


Figure 1: Graphic representation of the 2015 Refugio oil spill.

2 Advection-Diffusion

The dispersion of one primary fluid throughout another moving secondary fluid can be represented by the advection-diffusion equation, given as

$$\frac{\partial c}{\partial t} + \vec{v} \cdot \nabla c = D\Delta c + S, \tag{1}$$

where c is the concentration of the spreading primary fluid, $\vec{v}$ is the velocity vector of the moving body of secondary fluid, D is the diffusion coefficient of the primary fluid, and S is the source of flux of primary fluid into the domain. In the example of the oil spill, $\vec{v}$ represents the ocean currents, which have constant components v_x and v_y, D represents the diffusion rate of the oil into water, and S defines the volume of oil flowing into the ocean. The borders of the domain Ω are sufficiently far from the source of the spill such that the oil concentration at all times t is zero. However, the oil concentration along the coast must satisfy the zero flux condition

$$D\frac{\partial c}{\partial y} - v_y c = 0. \tag{2}$$

3 Algorithm

To evaluate the equation (1) at all the points in the domain Ω, the concentration of oil is represented as a vector rather than a matrix. This is accomplished by means of representing the grid points with some number p rather than their spatial coordinates (x, y), using the formula $p = (j-1)m + i$, where m is the number of discrete points in the x direction. For example, the oil concentration at grid point (3,1), with $m = 40$ and $n = 30$ discrete points along the x and y axes and a domain of $\Omega = [0\ 12]\times[0\ 3]$, would be represented as `cn(370)`

instead of `cn(10,10)`. The concentration of oil at some grid point $cn(p)$ can be evaluated using a combination of explicit and implicit techniques, used to approximate the PDEs for fluid advection and fluid diffusion, respectively.

3.1 Fluid Advection

The differential equation characterizing fluid advection is

$$\frac{\partial c}{\partial t} + \vec{v} \cdot \nabla c = 0, \tag{3}$$

where $\frac{\partial c}{\partial t}$ is the rate of change of the oil concentration at all grid points within Ω with respect to time. The PDE can be redefined using discrete values, such that equation (3) can be represented by

$$\frac{\texttt{cnp1(p) - cn(p)}}{\texttt{dt}} + \frac{\texttt{vx*(cn(p) - cn(p-1))}}{\texttt{dx}} + \frac{\texttt{vy*(cn(p) - cn(p-m))}}{\texttt{dy}} = 0$$

and rearranged to give

$$\texttt{cnp1(p)} = \texttt{cn(p)} - \frac{\texttt{dt}}{\texttt{dx}}\texttt{*vx*(cn(p) - cn(p-1))} - \frac{\texttt{dt}}{\texttt{dy}}\texttt{*vy*(cn(p) - cn(p-m))}. \tag{4}$$

Equation (4) can be evaluated in Matlab with a sufficiently small `dt`, `dx`, and `dy`, and the index `p` can be incremented in a `for` loop to evaluate the solution at all the grid points within Ω for each time step. Since `cnp1` is evaluated by determining the direct sum of other known quantities, this technique is considered **explicit**.

3.2 Fluid Diffusion

The equation that describes general fluid diffusion is

$$\frac{\partial c}{\partial t} = D\Delta c + S, \tag{5}$$

where Δc is the second spatial derivative of the concentration function c. Replacing the elements with discrete quantities gives

$$\frac{\texttt{cnp1(p)-cn(p)}}{\texttt{dt}} = \texttt{D*}\frac{\texttt{cnp1(p-1)-2*cnp1(p)+cnp1(p+1)}}{\texttt{dx*dx}} + \frac{\texttt{cnp1(p-m)-2*cnp1(p)+cnp1(p+m)}}{\texttt{dy*dy}} + \texttt{S(x(i),y(j),t+dt).}$$

Evaluating this expression explicitly, however, takes an extremely small time step to obtain any meaningful degree of accuracy, as the time step must be of the same order of magnitude as `dx*dx` and `dy*dy`. Instead, the diffusion equation must be solved **implicitly** by rearranging the equation to form

```
ac*cnp1(p) + ar*cnp1(p+1) + al*cnp1(p-1) + arr*cnp1(p+m) + all*cnp1(p-m) = RHS(p),
```

where `ac = 1+2*D*dt/dx/dx+2*D*dt/dy/dy`, `ar,al = -D*dt/dx/dx`, `arr,all = -D*dt/dy/dy`, and the equation's right hand side `RHS(p) = cn(p) + dt*S(i,j,t+dt)`. The equation can be solved implicitly using a square matrix `A` of sides `p` and evaluating `cnp1 = A\RHS` for each time step. The matrix `A` is filled using the following loop:

```
for i = 2:m-1
    for j = 2:n-1
        p = (j-1)*m + i;
        A(p,p)   = ac;
        A(p,p+1) = ar;
        A(p,p-1) = al;
        A(p,p+m) = arr;
        A(p,p-m) = all;

        RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt);
    end
end
```

3.3 General Form

The unification of the implicit and explicit methods used to solve the diffusion and advection equations gives a general form similar to the implicit system, but the right hand side term is modified to accomodate the advection term:

```
RHS(p) =
cn(p) + dt*S(x(i),y(j),t) -vx*(dt/dx)*(cn(p+1)-cn(p)) -vy*(dt/dy)*(cn(p+m)-cn(p))
```

The calculation `cnp1 = A\RHS` is performed the exact same way. As the number of discretized points `p` increases with an increase in `m` or `n`, the matrix `A` becomes exponentially larger, such that if `m = 40` and `n = 30`, `A` has over one million elements. To save time in calculating the solution `cnp1`, `A` is converted into a `sparse` matrix, which dramatically reduces the memory `A` occupies by ignoring the zero elements. When `cnp1` is calculated, it is reshaped from a vector of length `p` to a matrix of size `(m, n)` and plotted as a mesh of domain Ω.

3.4 Initial and Boundary Conditions

When time $t = 0$, the concentration across Ω is set equal to the given exact solution $c_{exact} = sin(x)cos(y)e^{-t}$. At the boundaries where i = 1, i = m, and j = n, the concentration c is maintained as the exact solution at all those points:

```
p = (j-1)*m + i;
        A(p,p) = 1;
        RHS(p) = exactSolution(x(i),y(j),t+dt);
```

At the boundary where j = 1, the condition

$$D\frac{\partial c}{\partial y} - v_y c = g. \tag{6}$$

must be fulfilled, where $g = g(x, y, t)$. Deriving g from the exact solution gives $g(x, y, t) = -De^{-t}sin(x)sin(y) - v_y sin(x)cos(y)e^{-t}$. This boundary condition is implemented as follows:

```
j = 1;
      p = (j-1)*m + i;
      A(p,p)   = ac + (2*vy*dt/dy);
      A(p,p+1) = ar;
      A(p,p-1) = al;
      A(p,p+m) = 2*arr;
      RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt) - vx*(dt/dx)*(cn(p+1)-cn(p))...
          - vy*(dt/dy)*(cn(p+m)-cn(p)) - (2*(dt/dy)*g(x(i),y(j),t+dt));
```

The right hand side and some elements of `A` are modified to accomodate the g condition (6).

4 Testing

To test the accuracy of the advection-diffusion solver, the evaluated solution `cn` is compared with the exact solution `solution`, which is a matrix of `p` elements, evaluated according to

```
exactSolution = @(x,y,t) sin(x)*cos(y)*exp(-t);

for k = 1:m
   for l = 1:n
       solution(k,l) = exactSolution(x(k),y(l),t);
   end
end
```

At `t = 1` second, the maximum difference between the approximate solution and the exact solution is calculated for varying resolutions:

Resolution	$\|\|max(cn - solution)\|\|$	Order
20×15	6.49×10^{-2}	–
40×30	3.32×10^{-2}	0.955
80×60	1.67×10^{-2}	0.988
160×120	1.06×10^{-2}	0.576

Table 1: Order of accuracy of the numerical approximation as domain resolution varies.

The order of accuracy is determined by subtracting the error of the previous resolution and the current resolution, and dividing by the error of the current resolution. The result is an indication of how much an error will decrease as a function of the resolution increase. In this case, the grid resolution doubles and the error is cut in half with each iteration, giving an order of accuracy of approximately one. The plots of the solution to equation (1), with several resolutions, can be found in Section A.1 in the Appendix.

5 Oil Spill

Slight modifications are made to the general advection-diffusion code to simulate the 2015 Refugio oil spill. The domain is taken from $[0\ 12]\times[0\ 3]$, with discretization points `m = 160` and `n = 40`. Initially, the concentration of oil throughout the domain is zero, and the boundaries stay at zero, with the exception of the beach. Along the beach, the condition (6) is set to permit zero flux through the boundary, such that $g(x,y,t)=0$. The source term defining oil flow into the ocean is given as a piecewise function

$$\begin{aligned} S(x,y,t) &= \begin{cases} \frac{1}{2}\left(1-\tanh\left(\frac{\sqrt{(x-x_s)^2+y^2}-r_s}{\epsilon}\right)\right), & \text{if } t<0.5, \\ 0, & \text{if } t>0.5, \end{cases} \\ x_s &= 10,\quad r_s = \epsilon = 0.1, \end{aligned} \tag{7}$$

such that the source of oil shuts off after time $t=0.5$. The ocean currents are uniform across the domain, such that `vx = -0.8` and `vy = -0.4`, and the oil has a diffusion coefficient of 0.2. Along certain points on the beach, the oil levels must not exceed $c_{limit}=0.006$, or those locations must be shut down.

6 Results

From time `t = 0` to `t = 0.5`, the oil accumulates at (10,0) in the domain Ω until the source S is turned off, at which point the oil begins to disperse throughout the domain. At time $t=1$, the oil begins to reach the beach closest to the source of the spill (see the beaches in Figure 1). At time $t=1.2$, the beach at point (8,0) becomes unsafe to swim in, as the oil concentration reaches above the critical value of $c_{limit}=0.006$. At $t=4$, the closest beach and the middle beach are unsafe to swim in, and

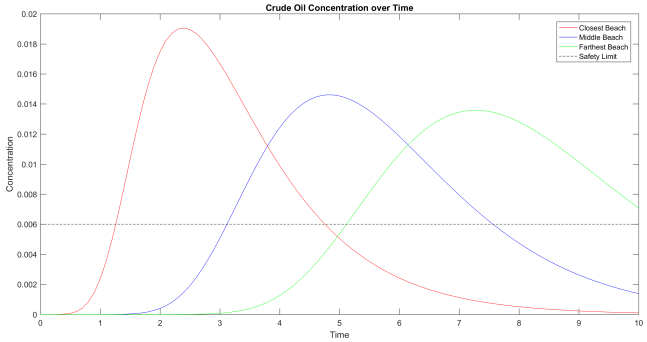
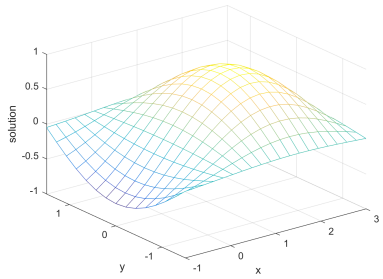


Figure 2: Oil concentration at the beaches located at (8,0), (6,0), and (4,0) within Ω.

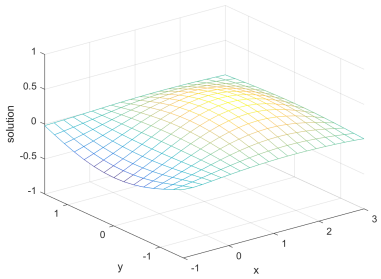
the oil spill has further progressed and diffused along the shoreline. At time $t = 7$, the beaches at (6,0) and (4,0) are unsafe to swim in, although the beach at (8,0) cleared up at $t = 4.8$. The beach at (4,0) is unsafe until after the end of the time interval $t = [0 \;\; 10]$. The full progression of the oil spill can be seen in Section A.2 in the Appendix.

A Figures

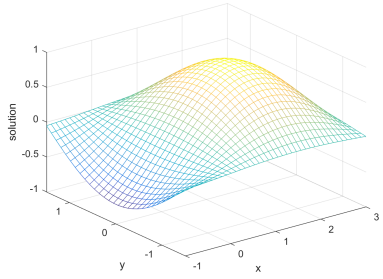
A.1 Advection-Diffusion Resolution Changes



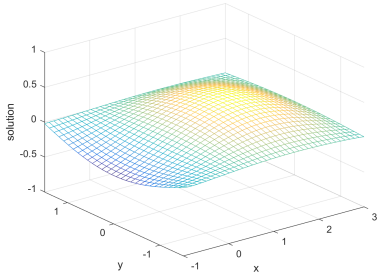
(a) $t = 0$, 20×15



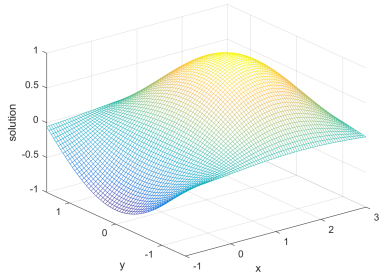
(b) $t = 1$



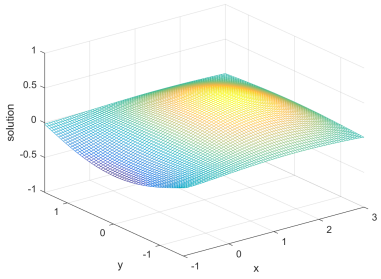
(c) $t = 0$, 40×30



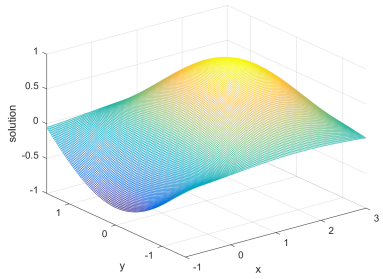
(d) $t = 1$



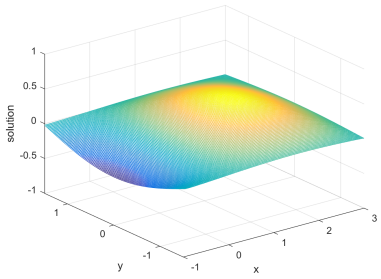
(e) $t = 0$, 80×60



(f) $t = 1$

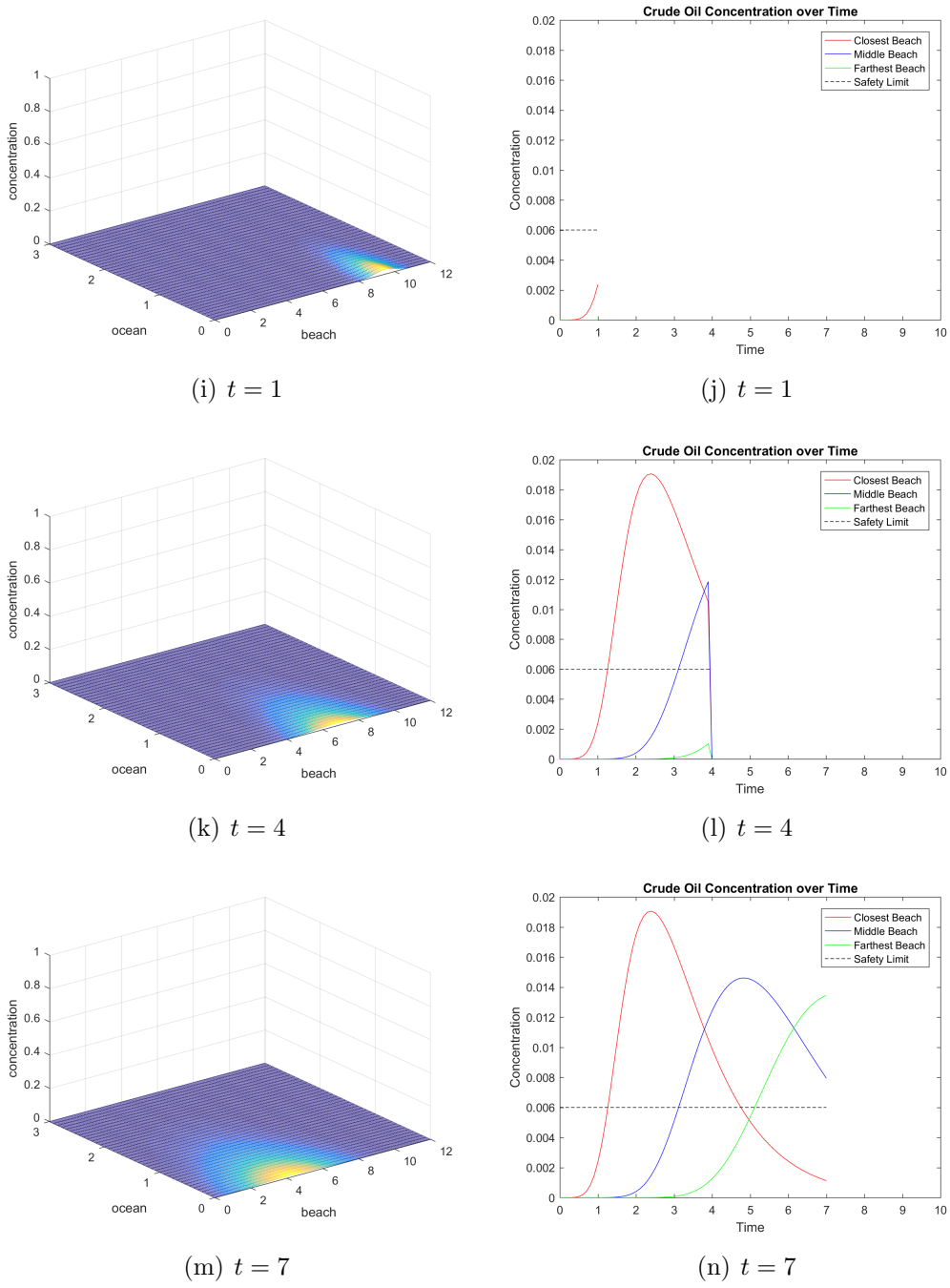


(g) $t = 0$, 160×120



(h) $t = 1$

A.2 Oil Spill



(i) $t = 1$ (j) $t = 1$

(k) $t = 4$ (l) $t = 4$

(m) $t = 7$ (n) $t = 7$

Figure 3: Progression of oil spill and beach concentration levels over time.

B Matlab Implementation

B.1 General Advection-Diffusion

```
%advection_diffusion2D.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%domain [a b] x [c d]:
a = -1; b = 3; c = -1.5; d = 1.5;

%advection field:
vx = -0.8; vy = -0.4;

%resolution, i.e. number of grid points:
m = 20; n = 15;

%initial and final times of simulation:
t = 0; tfinal = 1;

%diffusion coefficient:
D = 0.7;

%source term and bottom boundary condition:
S = @(x,y,t) (2*D*sin(x)*cos(y) + vx*cos(x)*cos(y) - vy*sin(x)*sin(y)...
    - sin(x)*cos(y))*exp(-t);
g = @(x,y,t) -D*exp(-t)*sin(x)*sin(y) - vy*sin(x)*cos(y)*exp(-t);

%define an exact solution to check that the code is correct:
exactSolution = @(x,y,t) sin(x)*cos(y)*exp(-t);
solution = zeros(m,n);

%build the grid:
x = linspace(a,b,m); dx = x(2)-x(1); dt = 0.5*dx;
y = linspace(c,d,n); dy = y(2)-y(1);

%coefficients for evaluating system of equations:
ac  = 1 + 2*D*dt/dx/dx + 2*D*dt/dy/dy;
ar  = -D*dt/dx/dx;
```

```
al  = -D*dt/dx/dx;
arr = -D*dt/dy/dy;
all = -D*dt/dy/dy;

%memory allocation for efficiency:
cn   = zeros(m*n,1);
cnp1 = zeros(m*n,1);
RHS  = zeros(m*n,1);
A    = zeros(m*n,m*n);

%initial condition:
for i = 1:m
    for j = 1:n
        p = (j-1)*m + i;
        cn(p) = exactSolution(x(i),y(j),0);
    end
end

%main loop:
while t < tfinal
   if (t + dt > tfinal)
       dt = tfinal-t;
   end

%implementing boundary conditions:
for i = 2:m-1
      %zero flux boundary:
      j = 1;
      p = (j-1)*m + i;
      A(p,p)   = ac + (2*vy*dt/dy);
      A(p,p+1) = ar;
      A(p,p-1) = al;
      A(p,p+m) = 2*arr;
      RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt) - vx*(dt/dx)*(cn(p+1)-cn(p))...
          - vy*(dt/dy)*(cn(p+m)-cn(p)) - (2*(dt/dy)*g(x(i),y(j),t+dt));
      %top boundary:
      j = n;
      p = (j-1)*m + i;
      A(p,p) = 1;
      RHS(p) = exactSolution(x(i),y(j),t+dt);
end
```

```
%left-right boundary conditions:
for j = 1:n
     i = 1;
        p = (j-1)*m + i;
        A(p,p) = 1;
        RHS(p) = exactSolution(x(i),y(j),t+dt);

     i = m;
        p = (j-1)*m + i;
        A(p,p) = 1;
        RHS(p) = exactSolution(x(i),y(j),t+dt);
end

%remainder of domain points:
for i = 2:m-1
    for j = 2:n-1
        p = (j-1)*m + i;
        A(p,p)   = ac;
        A(p,p+1) = ar;
        A(p,p-1) = al;
        A(p,p+m) = arr;
        A(p,p-m) = all;

        RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt) - vx*(dt/dx)*(cn(p+1)-cn(p))...
            - vy*(dt/dy)*(cn(p+m)-cn(p));
    end
end

%solve for the solution cnp1:
A = sparse(A);
cnp1 = A\RHS;
cn = reshape(cnp1,m,n);

%apply exact solution to grid points:
for k = 1:m
   for l = 1:n
       solution(k,l) = exactSolution(x(k),y(l),t);
   end
end

%prepare for next time step:
t = t+dt;
```

```
%plot results:
figure(1)
mesh(x,y,cn');
xlabel('x');
ylabel('y');
zlabel('solution');
axis([a b c d -1 1]);
pause(0.1)

error = solution - cn;
maximum = max(error(:));
end
```

B.2 Oil Spill Simulation

```
%oil.m

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 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.   %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%domain [a b] x [c d]:
a = 0; b = 12; c = 0; d = 3;

%ocean currents:
vx = -0.8; vy = -0.4;

%resolution, i.e. number of grid points:
m = 160; n = 40;

%time interval and step:
t = 0; tfinal = 10; dt = 0.1; re = 0;
time = linspace(0, tfinal/dt, (tfinal/dt)+1);

%diffusion coefficient and safety limit:
D = 0.2;
limit = 0.006*ones((tfinal/dt)+1,1);

%source term coefficients:
xs = 10; rs = 0.1; epsilon = 0.1;
```

```
%source term and beach flux boundary:
S = @(x,y,t) 0.5*(1-(tanh((sqrt((x-xs)^2 + y^2) - rs)/epsilon)));
g = @(x,y,t) 0;

%build the grid:
x = linspace(a,b,m); dx = x(2)-x(1);
y = linspace(c,d,n); dy = y(2)-y(1);

%coefficients for evaluating system of equations:
ac  = 1 + 2*D*dt/dx/dx + 2*D*dt/dy/dy;
ar  = -D*dt/dx/dx;
al  = -D*dt/dx/dx;
arr = -D*dt/dy/dy;
all = -D*dt/dy/dy;

%memory allocation for efficiency:
cn   = zeros(m*n,1);
cnp1 = zeros(m*n,1);
RHS  = zeros(m*n,1);
A    = zeros(m*n,m*n);
farbeach = zeros(tfinal/dt,1);
midbeach = zeros(tfinal/dt,1);
nearbeach = zeros(tfinal/dt,1);

%initial condition:
for i = 1:m
    for j = 1:n
        p = (j-1)*m + i;
        cn(p) = 0;
    end
end

%main loop:
while t < tfinal
   if (t + dt > tfinal)
       dt = tfinal-t;
   end

   %time of oil shutoff:
   if t > 0.5
       S = @(x,y,t) 0;
```

```
   end

    %implement boundary conditions:
    for i = 2:m-1
      %beach:
      j = 1;
      p = (j-1)*m + i;
      A(p,p)   = ac + (2*vy*dt/dy);
      A(p,p+1) = ar;
      A(p,p-1) = al;
      A(p,p+m) = 2*arr;
      RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt) - vx*(dt/dx)*(cn(p+1)-cn(p))...
          - vy*(dt/dy)*(cn(p+m)-cn(p)) - (2*(dt/dy)*g(x(i),y(j),t+dt));
      %water:
      j = n;
      p = (j-1)*m + i;
      A(p,p) = 1;
      RHS(p) = 0;
    end

    %end-to-end water conditions:
    for j = 1:n
     i = 1;
        p = (j-1)*m + i;
        A(p,p) = 1;
        RHS(p) = 0;

     i = m;
        p = (j-1)*m + i;
        A(p,p) = 1;
        RHS(p) = 0;
    end

%remainder of domain points:
for i = 2:m-1
    for j = 2:n-1
        p = (j-1)*m + i;
        A(p,p)   = ac;
        A(p,p+1) = ar;
        A(p,p-1) = al;
        A(p,p+m) = arr;
        A(p,p-m) = all;
```

```
        RHS(p) = cn(p) + dt*S(x(i),y(j),t+dt) - vx*(dt/dx)*(cn(p+1)-cn(p))...
            - vy*(dt/dy)*(cn(p+m)-cn(p));
    end
end

%solve for the solution cnp1:
farbeach(re+1) = cn(53);
midbeach(re+1) = cn(80);
nearbeach(re+1) = cn(107);
A = sparse(A);
cnp1 = A\RHS;
cn = reshape(cnp1,m,n);

%prepare for next time step:
t = t+dt;
re = re+1;

%plot results:
figure(1)
mesh(x,y,cn');
xlabel('beach');
ylabel('ocean');
zlabel('concentration');
axis([a b c d 0 1]);
pause(0.1)
end

%oil concentration:
figure(2)
plot(time,nearbeach,'r');
hold on
plot(time,midbeach,'b');
hold on
plot(time,farbeach,'g');
hold on
plot(time,limit,'k--');
hold on
title('Crude Oil Concentration over Time');
xlabel('Time');
ylabel('Concentration');
legend('Closest Beach','Middle Beach','Farthest Beach','Safety Limit');
```

```
axis([0 100 0 0.02])
```

C Statement of Acknowledgement

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.