AAE 694: Computational Aeroacoustics

Project 2

Michael Crawley

3/3/2011

Introduction

In this project, the two dimensional linearized Euler equations are solved numerically given initial conditions and a mean flow Mach number. This numerical solution is accomplished using an explicit finite difference method; the coefficients for the spatial derivatives derived from Tam's fourth order Dispersion-Relation-Preserving scheme, and the coefficients for the temporal derivatives are derived from a third order optimized scheme. The nondimensionalized governing equations for this study are therefore

$$\frac{\partial \rho}{\partial t} = -\frac{\partial}{\partial x}(M_x\rho + u) - \frac{\partial}{\partial y}(M_y\rho + v)$$
$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(M_x u + p) - \frac{\partial}{\partial y}(M_y u)$$
$$\frac{\partial v}{\partial t} = -\frac{\partial}{\partial x}(M_x v) - \frac{\partial}{\partial y}(M_y v + p)$$
$$\frac{\partial p}{\partial t} = -\frac{\partial}{\partial x}(M_x p + u) - \frac{\partial}{\partial y}(M_y p + v)$$

These equations are subsequently discretized as

$$\rho_{l,m}^{n+1} = \rho_{l,m}^{n} + \Delta t \sum_{j=0}^{3} b_j \left\{ -\sum_{j=-3}^{3} a_j\left(M_x \rho_{l+j,m}^{n-j} + u_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y \rho_{l,m+j}^{n-j} + v_{l,m+j}^{n-j}\right)\right\}$$
$$u_{l,m}^{n+1} = u_{l,m}^{n} + \Delta t \sum_{j=0}^{3} b_j \left\{ -\sum_{j=-3}^{3} a_j\left(M_x u_{l+j,m}^{n-j} + p_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y u_{l,m+j}^{n-j}\right)\right\}$$
$$v_{l,m}^{n+1} = v_{l,m}^{n} + \Delta t \sum_{j=0}^{3} b_j \left\{ -\sum_{j=-3}^{3} a_j\left(M_x v_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y v_{l,m+j}^{n-j} + p_{l,m+k}^{n-j}\right)\right\}$$
$$p_{l,m}^{n+1} = p_{l,m}^{n} + \Delta t \sum_{j=0}^{3} b_j \left\{ -\sum_{j=-3}^{3} a_j\left(M_x p_{l+j,m}^{n-j} + u_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y p_{l,m+j}^{n-j} + v_{l,m+k}^{n-j}\right)\right\}$$

Initial Conditions

The initial conditions for the study are shown in Figure 1, where the nondimensionalized pressure, density, x velocity, and y velocity contours are plotted. The initial conditions take the form of two perturbations: one centered at the origin with relatively strong amplitude, and one centered at (67,0) with relatively weak amplitude.

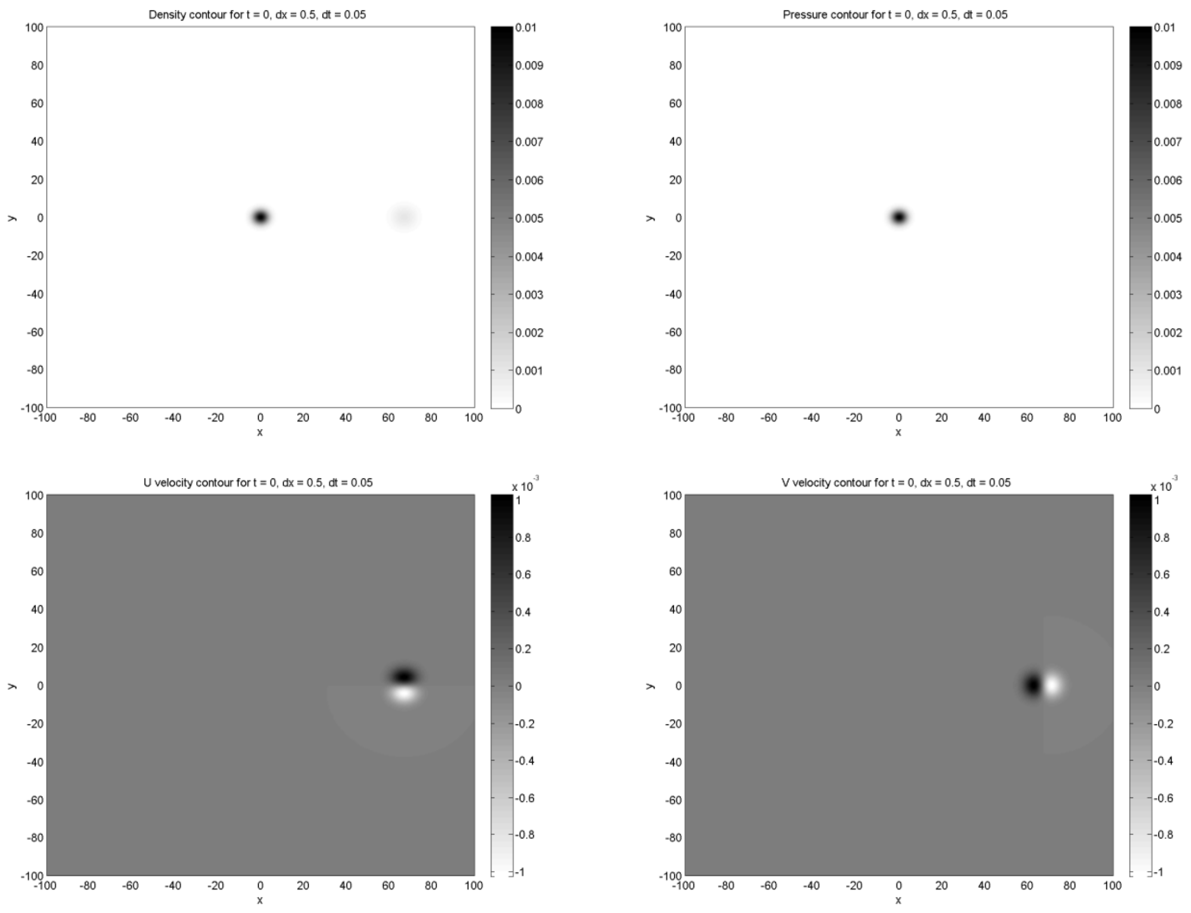

Figure 1: Density, pressure, and velocity contours for the initial conditions.

Boundary Conditions

Special care must be taken at the boundaries of the domain to ensure that the disturbances propagate out of the domain without producing reflections at the boundaries. In order to accomplish this, radiation (for acoustic only waves) and outflow boundary (for acoustic, entropy and vorticity waves) conditions are imposed at the edges of the computational domain.

Radiation Boundaries

In the derivation of the radiation boundary conditions, it is assumed that the boundaries of the computational domain are sufficiently far from the acoustic source so that they may be considered to be in the source's far field. The radiation boundary conditions can be found from the asymptotic solution to the linearized Euler equations, and in nondimensionalized form are therefore

$$\frac{\partial \phi}{\partial t} = -A\frac{\partial \phi}{\partial x} - B\frac{\partial \phi}{\partial y} - C\phi$$

$$\phi = \{\rho; u; v; p\}$$

$$A = \cos\theta\left[M_x\cos\theta + M_y\sin\theta + \sqrt{1-\left(M_x\sin\theta - M_y\cos\theta\right)^2}\right]$$

$$B = \sin\theta\left[M_x\cos\theta + M_y\sin\theta + \sqrt{1-\left(M_x\sin\theta - M_y\cos\theta\right)^2}\right]$$

$$C = \frac{1}{2}(x^2+y^2)^{-1/2}\left[M_x\cos\theta + M_y\sin\theta + \sqrt{1-\left(M_x\sin\theta - M_y\cos\theta\right)^2}\right]$$

These boundary conditions can therefore be discretized as

$$\phi_{l,m}^{n+1} = \phi_{l,m}^{n} + \Delta t\sum_{j=0}^{3} b_j\left\{-A_{l,m}\sum_{j=-3}^{3} a_j\phi_{l+j,m}^{n-j} - B_{l,m}\sum_{j=-3}^{3} a_j\phi_{l,m+j}^{n-j} - C_{l,m}\phi_{l,m}\right\}$$

Outflow Boundaries

Unlike the far field region, the outflow region contains entropy and vorticity waves in addition to the acoustic waves. The outflow boundary conditions can therefore be found from the asymptotic solutions to the acoustic, entropy and vorticity waves. In nondimensionalized form, they are

$$\frac{\partial\rho}{\partial t} = \frac{\partial}{\partial x}(p - M_x\rho) + \frac{\partial}{\partial y}(p - M_y\rho) - A\frac{\partial p}{\partial x} - B\frac{\partial p}{\partial y} - Cp$$

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(M_x u + p) - \frac{\partial}{\partial y}(M_y u)$$

$$\frac{\partial v}{\partial t} = -\frac{\partial}{\partial x}(M_x v) - \frac{\partial}{\partial y}(M_y v + p)$$

$$\frac{\partial p}{\partial t} = -A\frac{\partial p}{\partial x} - B\frac{\partial p}{\partial y} - Cp$$

These equations are then discretized as

$$\rho_{l,m}^{n+1} = \rho_{l,m}^{n} + \Delta t\sum_{j=0}^{3} b_j\left\{\sum_{j=-3}^{3} a_j\left(p_{l+j,m}^{n-j} - M_x\rho_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(p_{l,m+j}^{n-j} - M_y\rho_{l,m+j}^{n-j}\right)\right.$$
$$\left. - A_{l,m}\sum_{j=-3}^{3} a_j p_{l+j,m}^{n-j} - B_{l,m}\sum_{j=-3}^{3} a_j p_{l,m+j}^{n-j} - C_{l,m}p_{l,m}\right\}$$

$$u_{l,m}^{n+1} = u_{l,m}^{n} + \Delta t\sum_{j=0}^{3} b_j\left\{-\sum_{j=-3}^{3} a_j\left(M_x u_{l+j,m}^{n-j} + p_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y u_{l,m+j}^{n-j}\right)\right\}$$

$$v_{l,m}^{n+1} = v_{l,m}^{n} + \Delta t\sum_{j=0}^{3} b_j\left\{-\sum_{j=-3}^{3} a_j\left(M_x v_{l+j,m}^{n-j}\right) - \sum_{j=-3}^{3} a_j\left(M_y v_{l,m+j}^{n-j} + p_{l,m+k}^{n-j}\right)\right\}$$

$$p_{l,m}^{n+1} = p_{l,m}^{n} + \Delta t\sum_{j=0}^{3} b_j\left\{-A_{l,m}\sum_{j=-3}^{3} a_j p_{l+j,m}^{n-j} - B_{l,m}\sum_{j=-3}^{3} a_j p_{l,m+j}^{n-j} - C_{l,m}p_{l,m}\right\}$$

Grid Resolution

The effects of the grid resolution are shown in Figure 2, where the density contours for two time periods and two different spatial/temporal grid resolutions are shown. Clearly, using the coarse grid results in negligible changes in the disturbances' amplitude and locations. It is only after 600 seconds, when the disturbances waves have propagated out of the computational domain, do the effects of the grid resolution become apparent. As one would expect, the finer grid results in a smoother contour than the coarse grid, which exhibits numerous additional oscillations. For the remainder of this study, the fine grid will be used.

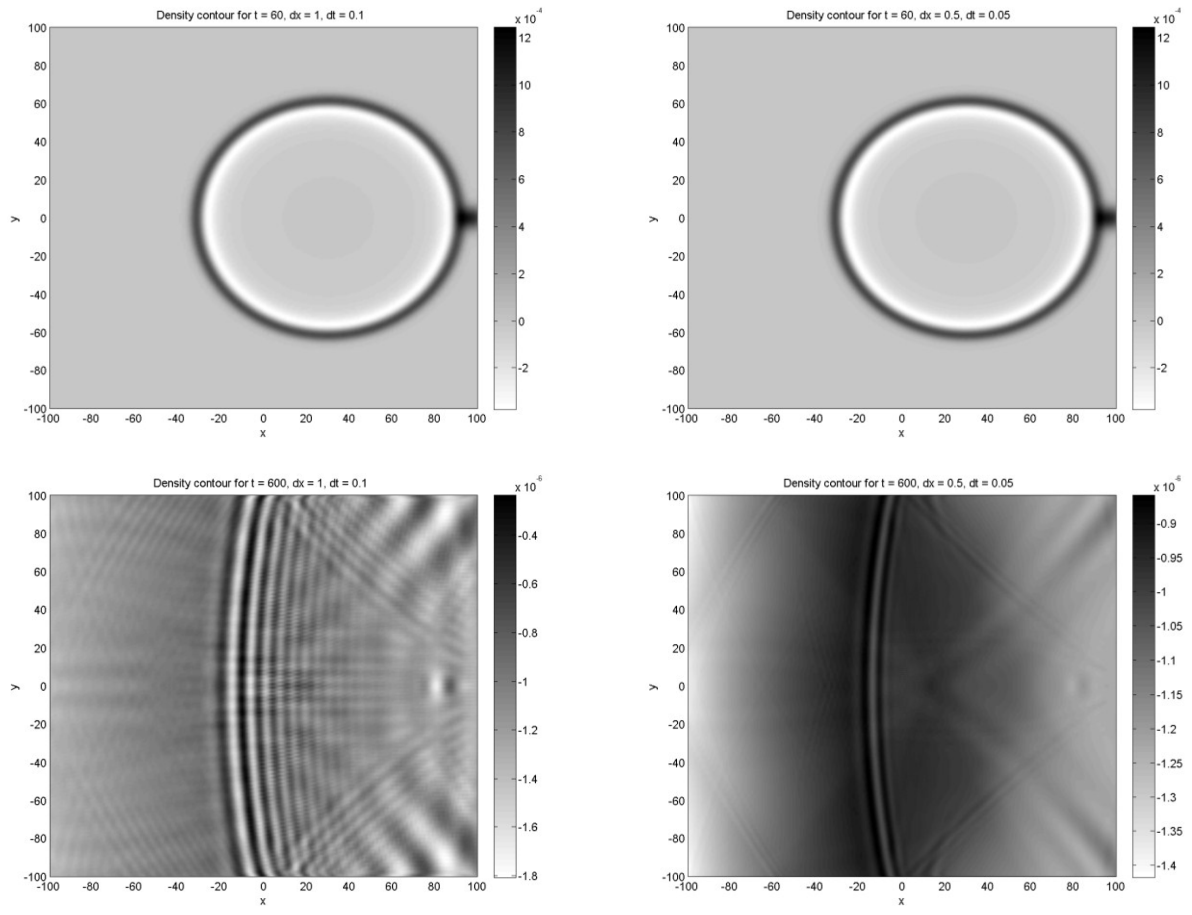

Figure 2: Density contours at 60 seconds and 600 seconds for two different spatial-temporal grid resolutions.

Results

Case 1

For the first case, the mean flow Mach number is set as 0.5 and the flow direction is along the x-axis only. Figure 3 shows the density contour at selected time periods. From the figure, it is clear that the evolution of the perturbations is being properly captured by the numerical scheme. The high amplitude perturbation evolves into an expanding wave front with decreasing amplitude, which expands at sonic speed, while the center of the wave front moves with the mean flow. Meanwhile, the low amplitude perturbation evolves much more slowly, and is advected out of the computational domain before it can expand appreciably. By 600 seconds, the primary waves have propagated out of the domain entirely, leaving only very low amplitude oscillations in the contour.

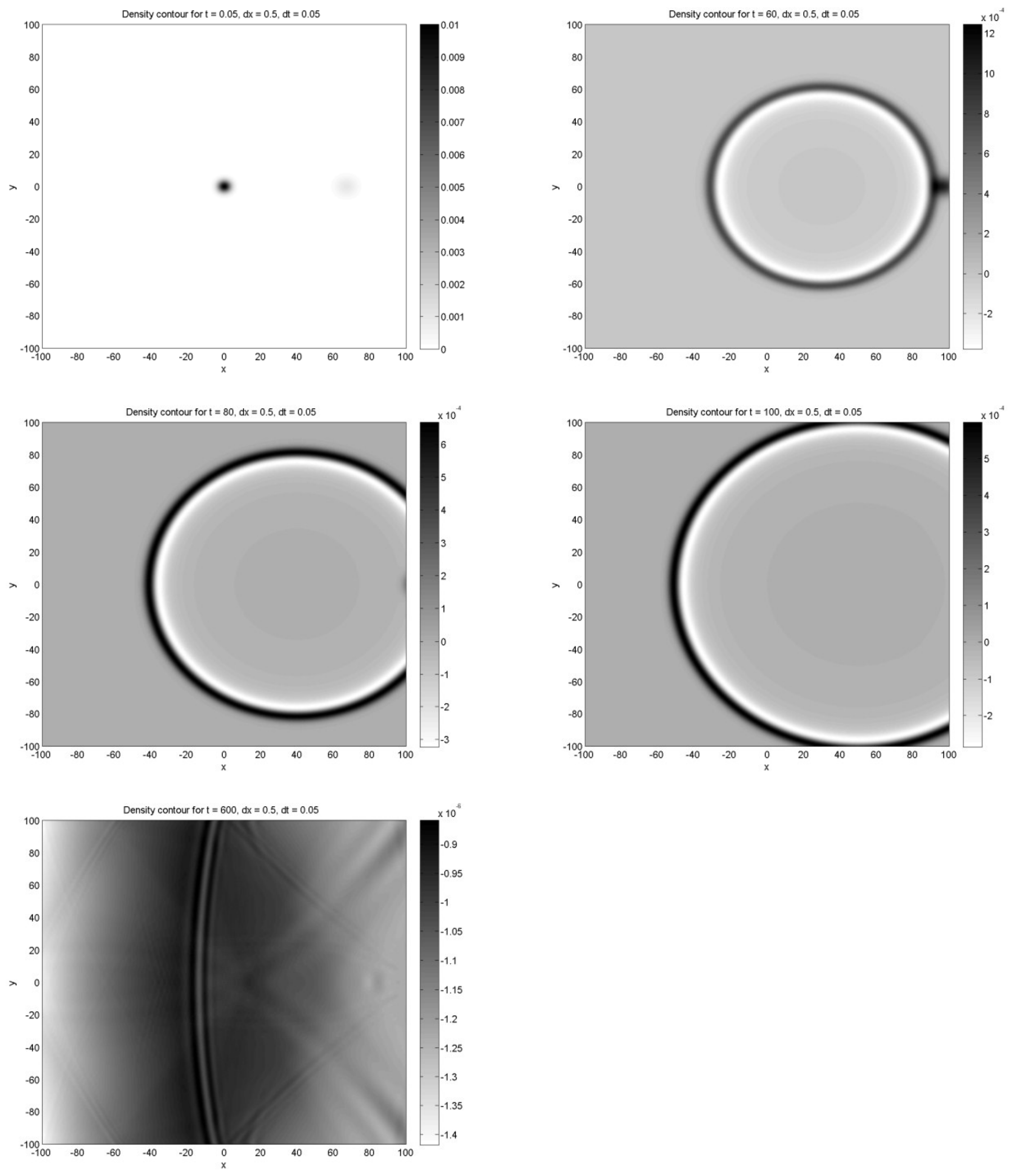

Figure 3: Density contour at 0.05, 60, 80, 100, and 600 seconds.

Figure 4 shows the pressure contour for selected time periods. Again, it is clear that the numerical scheme is properly capturing the evolution of the pressure disturbance. The pressure contours are nearly identical to the density contours, with the exception that only one perturbation exists in the pressure domain.

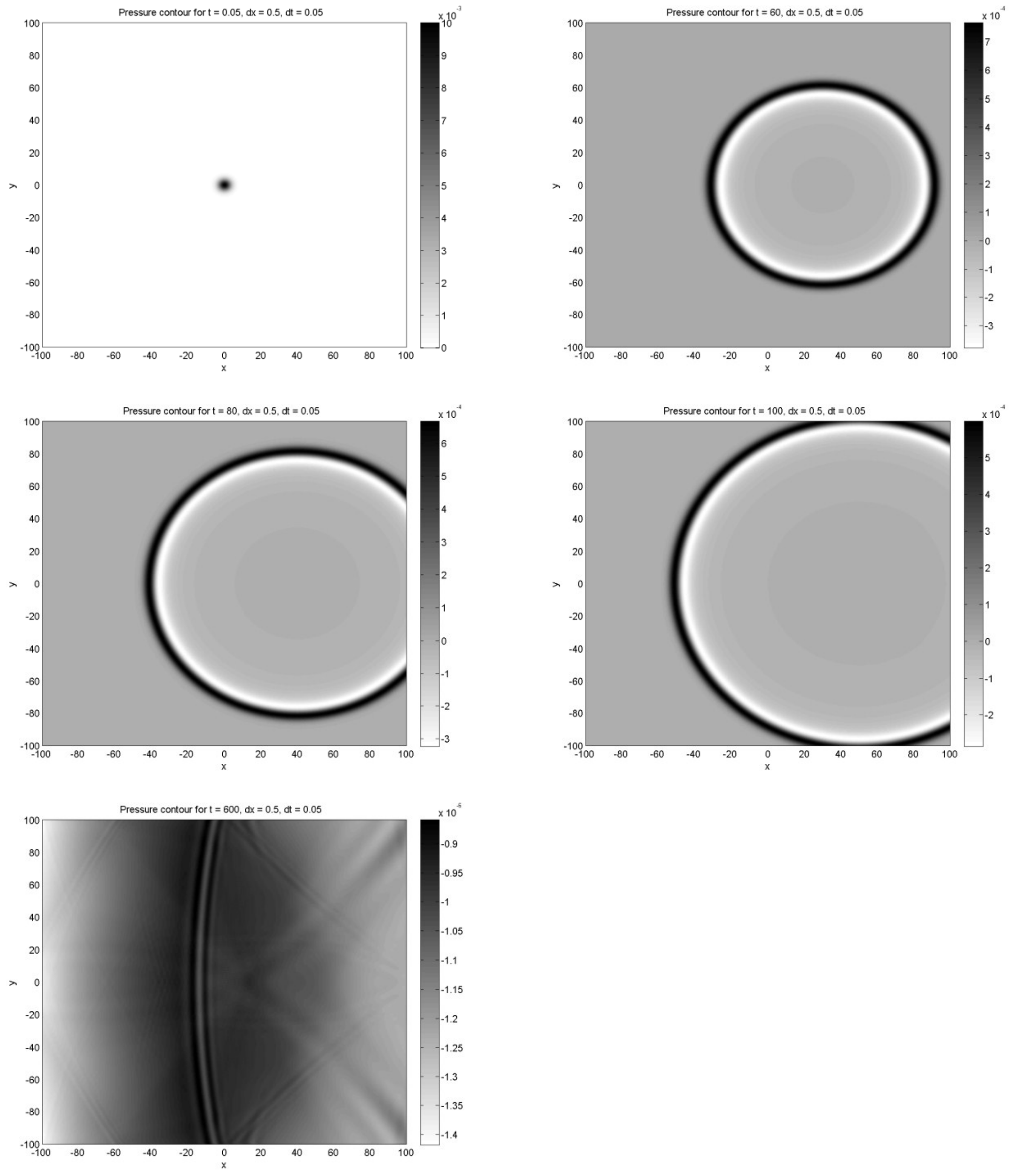

Figure 4: Pressure contour at 0.05, 60, 80, 100, and 600 seconds.

Figure 5 and Figure 6 show the x-direction and y-direction velocity contours, respectively. Initially, only the weak perturbation centered at (67,0) existed in these domains. However, the coupling between density, pressure and velocity results in the generation of a perturbation at the origin, which quickly overtakes the initial perturbation.

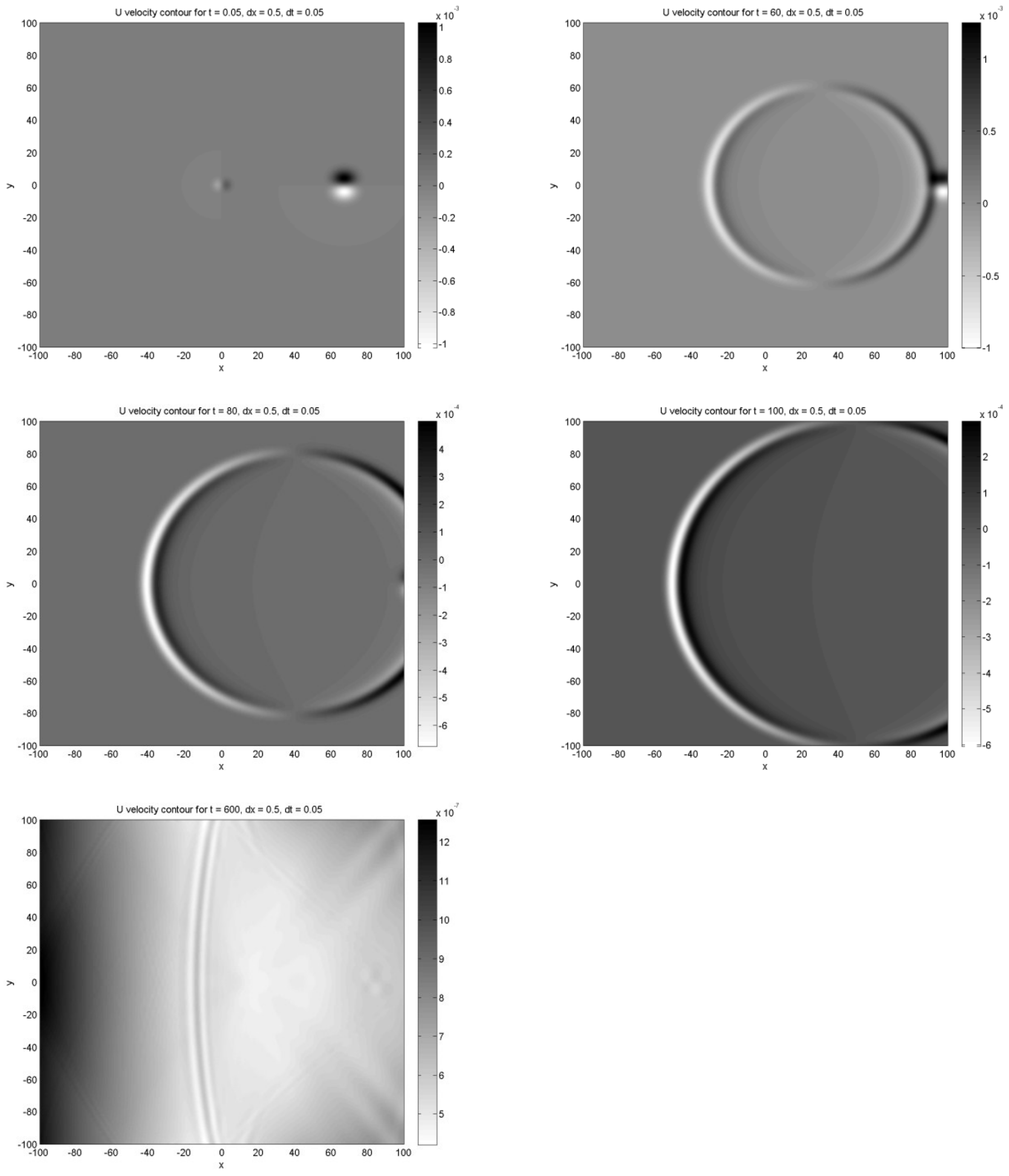

Figure 5: u velocity contour at 0.05, 60, 80, 100, and 600 seconds.

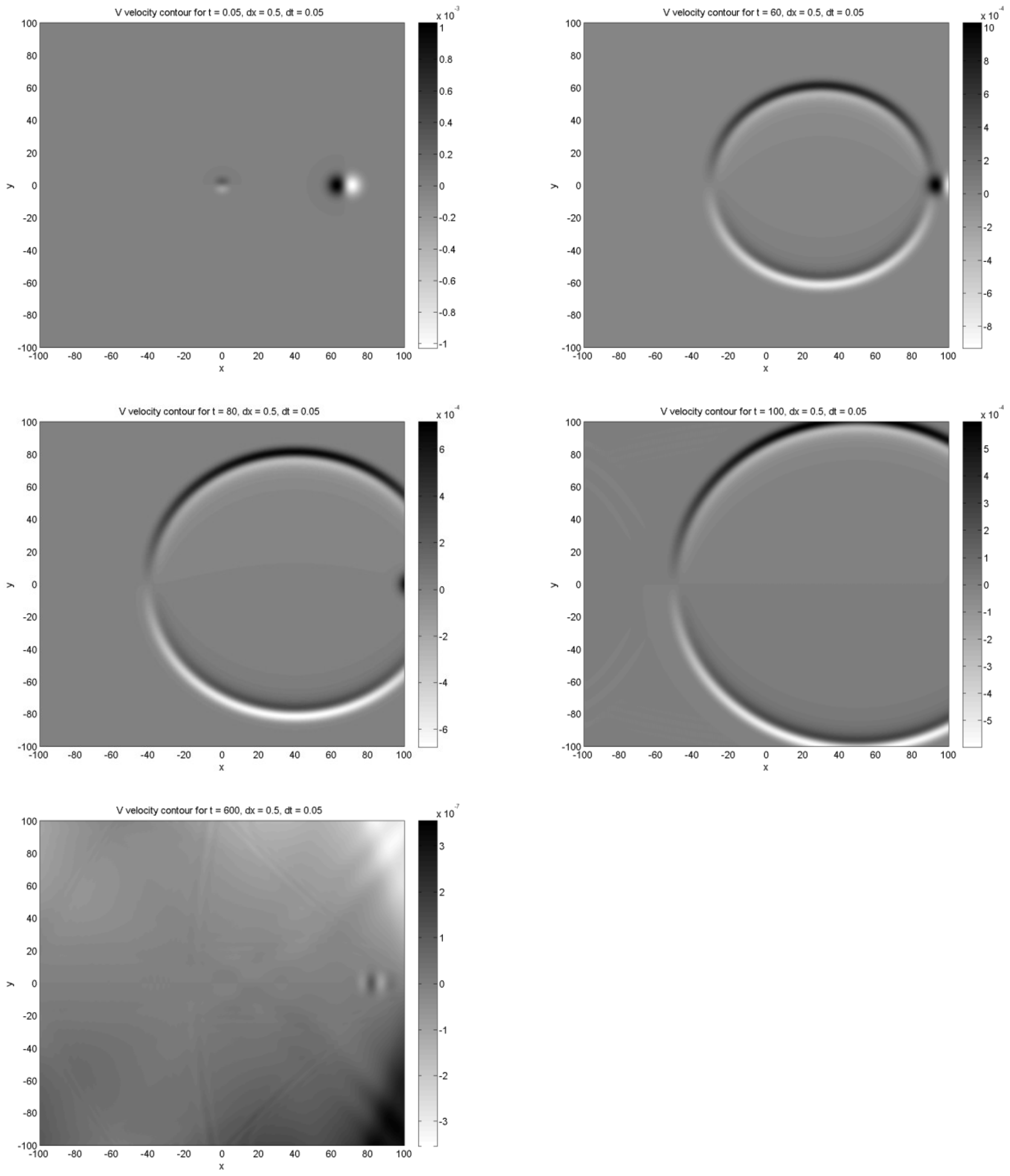

Figure 6: v velocity contour at 0.05, 60, 80, 100, and 600 seconds.

Case 2

For the second case, identical initial conditions are use and the mean flow Mach number has the same magnitude. However, in this case the mean flow is directed in the $\theta = \frac{\pi}{4}$ direction and hence, both the right and the top boundaries are now outflow boundaries. The density, pressure, x-direction velocity and v-direction velocity contours for selected time periods are shown in Figure 7, Figure 8, Figure 9, and Figure 10, respectively. As with the first case, the high amplitude perturbation produces an expanding wave front which propagates outwards and is undeformed by the mean flow. The weak perturbation is advected out of the computational domain before it is able to undergo significant evolution. By the end of the simulation, the perturbation waves have propagated out of the domain entirely, and only low amplitude oscillations remain.

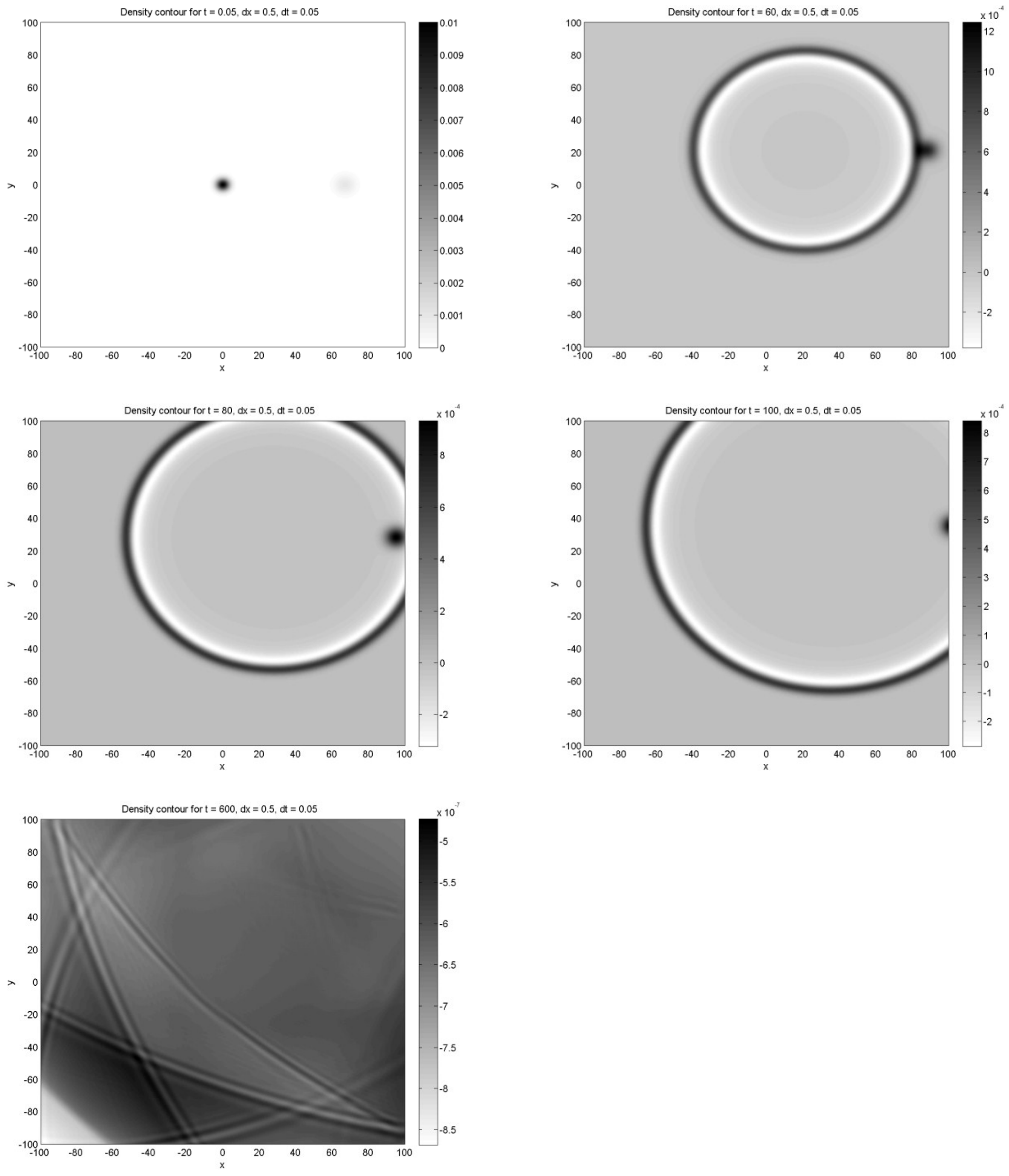

Figure 7: Density contour at 0.05, 60, 80, 100, and 600 seconds.

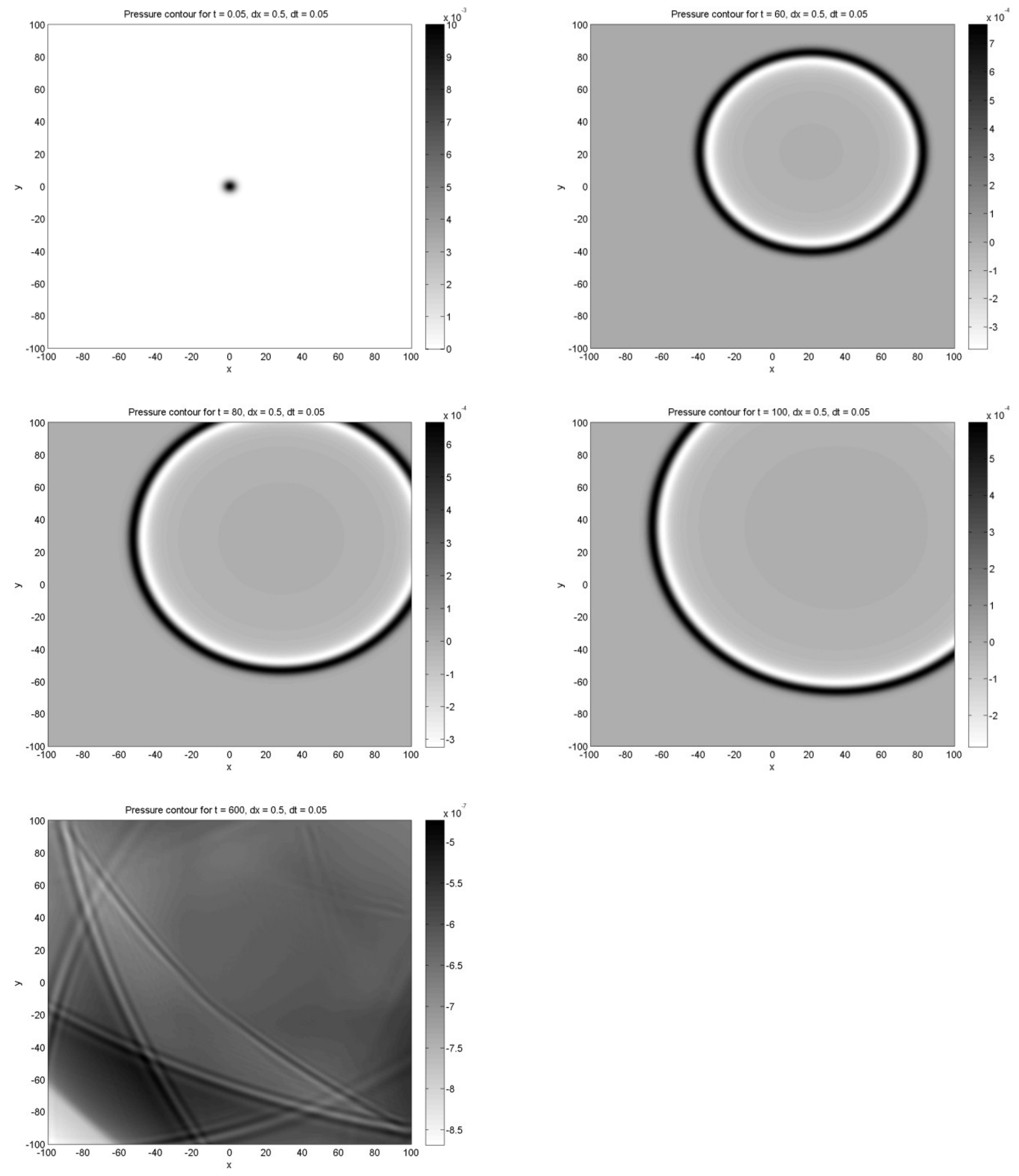

Figure 8: Pressure contour at 0.05, 60, 80, 100, and 600 seconds.

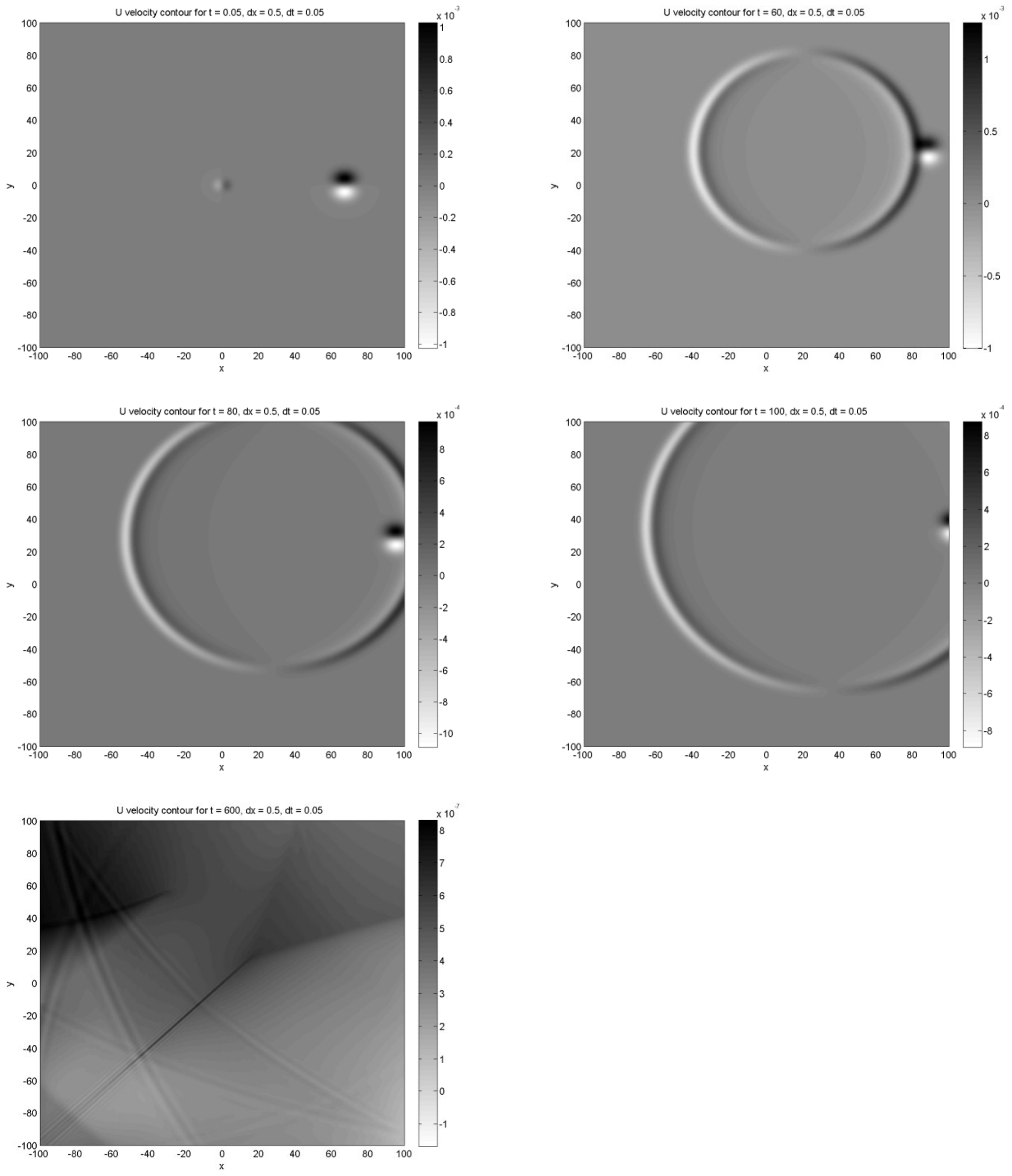

Figure 9: u velocity contour at 0.05, 60, 80, 100, and 600 seconds.

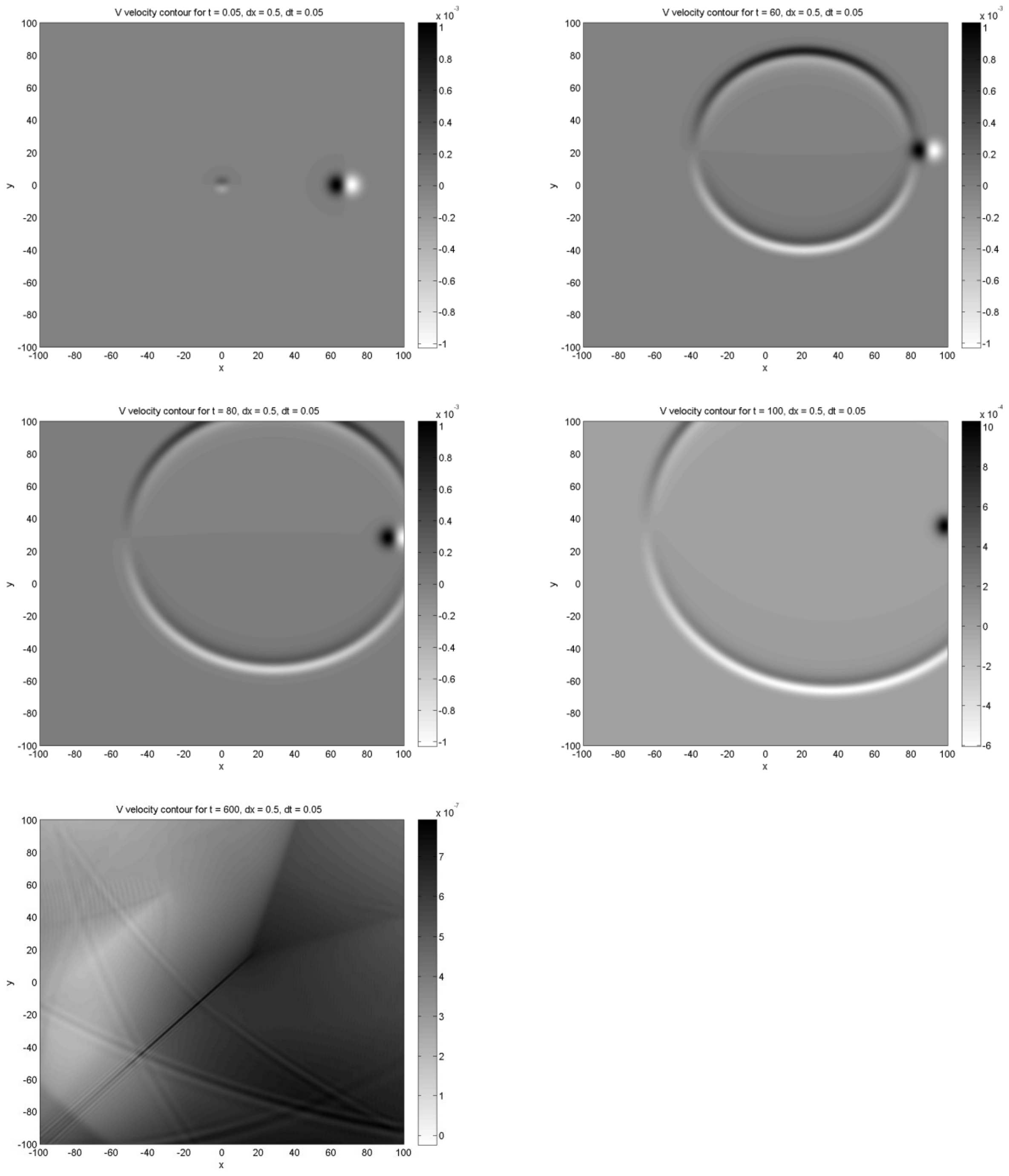

Figure 10: v velocity contour at 0.05, 60, 80, 100, and 600 seconds.

Appendix

Appendix A: Main Program

```
function [] = Project2(dx,dt,Mx,My)
    %Euler equation solver, completed for AAE 694 Project 2 by
    %Michael Crawley
    %Inputs:
    %       dx: spatial step size
    %       dt: temporal step size
    %       Mx: mean flow in x-direction
    %       My: mean flow in y-direction
    tic;
    dy = dx;

    %Set up grid
    X = -100:dx:100;
    Y = -100:dy:100;
    [x, y] = meshgrid(X,Y);
    foldername = [num2str(length(X)),' x ',num2str(length(Y)), ' Mx',
strrep(num2str(Mx),'.','_'), ' My',strrep(num2str(My),'.','_')];
    shift = [0 0 -1];

    %Set maximum time for simulation to run
    time_itr_max = 0.05*12000;
    plot_times = 0.05*[100 200 300 400 500 600 800 1000 1200 1600 2000 4000
12000];

    %Boundary condition coefficients
    r = sqrt(x.^2+y.^2);
    V = Mx*(x./r)+My*(y./r)+sqrt(1-(Mx*(y./r)-My*(x./r)).^2);
    A = V.*(x./r);
    B = V.*(y./r);
    C = V./(2*r);

    %3rd order optimized time discretization coefficients (b0, b1, b2, b3)
    b = [2.3025580883830 -2.4910075998482 1.5743409331815 -0.38589142217162];

    %initialize variables
    rho = zeros(length(y),length(x),4);
    u = rho; v = rho; p = rho;
    Drho = rho;
    Du = u;
    Dv = v;
    Dp = p;

    %initial conditions
    rho(:,:,4) = 0.01*exp(-log(2)*(x.^2+y.^2)/9)+0.001*exp(-log(2)*((x-
67).^2+y.^2)/25);
    u(:,:,4) = 0.0004*y.*exp(-log(2)*((x-67).^2+y.^2)/25);
    v(:,:,4) = -0.0004*(x-67).*exp(-log(2)*((x-67).^2+y.^2)/25);
    p(:,:,4) = 0.01*exp(-log(2)*(x.^2+y.^2)/9);

    if exist(foldername,'file') ~= 7
        mkdir(pwd,foldername);
    end
    cd(foldername);
    h = waitbar(0,'Calculating...');
    imax = length(0:dt:time_itr_max)-1;
    for i = 1:imax %time step loop
        waitbar(i/imax,h,['i = ',num2str(i)]);

        %time shift spatial derivative matrices
        Drho = circshift(Drho,shift);
        Du = circshift(Du,shift);
        Dv = circshift(Dv,shift);
        Dp = circshift(Dp,shift);

        %interior nodes
        Drhoi = NumericalDerivative(1,6,dx,-(Mx*rho(:,:,4)+u(:,:,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*rho(:,:,4)+v(:,:,4))','-DRP')';
        Dui = NumericalDerivative(1,6,dx,-(Mx*u(:,:,4)+p(:,:,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*u(:,:,4))','-DRP')';
        Dvi = NumericalDerivative(1,6,dx,-(Mx*v(:,:,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*v(:,:,4)+p(:,:,4))','-DRP')';
        Dpi = NumericalDerivative(1,6,dx,-(Mx*p(:,:,4)+u(:,:,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*p(:,:,4)+v(:,:,4))','-DRP')';

        %left boundary (radiation)
        Drhol = A(:,1:7).*NumericalDerivative(1,6,dx,-(rho(:,1:7,4)),'-
DRP')+B(:,1:7).*NumericalDerivative(1,6,dy,-(rho(:,1:7,4))','-DRP')'-
C(:,1:7).*rho(:,1:7,4);
        Dul = A(:,1:7).*NumericalDerivative(1,6,dx,-(u(:,1:7,4)),'-
DRP')+B(:,1:7).*NumericalDerivative(1,6,dy,-(u(:,1:7,4))','-DRP')'-
C(:,1:7).*u(:,1:7,4);
        Dvl = A(:,1:7).*NumericalDerivative(1,6,dx,-(v(:,1:7,4)),'-
DRP')+B(:,1:7).*NumericalDerivative(1,6,dy,-(v(:,1:7,4))','-DRP')'-
C(:,1:7).*v(:,1:7,4);
        Dpl = A(:,1:7).*NumericalDerivative(1,6,dx,-(p(:,1:7,4)),'-
DRP')+B(:,1:7).*NumericalDerivative(1,6,dy,-(p(:,1:7,4))','-DRP')'-
C(:,1:7).*p(:,1:7,4);

        %right boundary
        if Mx == 0 %radiation boundary conditions
            Drhor = A(:,end-6:end).*NumericalDerivative(1,6,dx,-(rho(:,end-
6:end,4)),'-DRP')+B(:,end-6:end).*NumericalDerivative(1,6,dy,-(rho(:,end-
6:end,4))','-DRP')'-C(:,end-6:end).*rho(:,end-6:end,4);
            Dur = A(:,end-6:end).*NumericalDerivative(1,6,dx,-(u(:,end-
6:end,4)),'-DRP')+B(:,end-6:end).*NumericalDerivative(1,6,dy,-(u(:,end-
6:end,4))','-DRP')'-C(:,end-6:end).*u(:,end-6:end,4);
            Dvr = A(:,end-6:end).*NumericalDerivative(1,6,dx,-(v(:,end-
6:end,4)),'-DRP')+B(:,end-6:end).*NumericalDerivative(1,6,dy,-(v(:,end-
6:end,4))','-DRP')'-C(:,end-6:end).*v(:,end-6:end,4);
            Dpr = A(:,end-6:end).*NumericalDerivative(1,6,dx,-(p(:,end-
6:end,4)),'-DRP')+B(:,end-6:end).*NumericalDerivative(1,6,dy,-(p(:,end-
6:end,4))','-DRP')'-C(:,end-6:end).*p(:,end-6:end,4);
        else %outflow boundary conditions
            Drhor = NumericalDerivative(1,6,dx,-Mx*rho(:,end-
6:end,4)+Mx*p(:,end-6:end,4),'-DRP')+A(:,end-
6:end).*NumericalDerivative(1,6,dx,-p(:,end-6:end,4),'-
DRP')+NumericalDerivative(1,6,dy,(-My*rho(:,end-6:end,4)+My*p(:,end-
6:end,4))','-DRP')'+B(:,end-6:end).*NumericalDerivative(1,6,dy,-p(:,end-
6:end,4)','-DRP')'-C(:,end-6:end).*p(:,end-6:end,4);
            Dur = NumericalDerivative(1,6,dx,-(Mx*u(:,end-6:end,4)+p(:,end-
6:end,4)),'-DRP')+NumericalDerivative(1,6,dy,-(My*u(:,end-6:end,4))','-
DRP')';
            Dvr = NumericalDerivative(1,6,dx,-(Mx*v(:,end-6:end,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*v(:,end-6:end,4)+p(:,end-6:end,4))','-
DRP')';
            Dpr = A(:,end-6:end).*NumericalDerivative(1,6,dx,-(p(:,end-
6:end,4)),'-DRP')+B(:,end-6:end).*NumericalDerivative(1,6,dy,-(p(:,end-
6:end,4))','-DRP')'-C(:,end-6:end).*p(:,end-6:end,4);
        end

        %bottom boundary (radiation)
        Drhob = A(1:7,:).*NumericalDerivative(1,6,dx,-(rho(1:7,:,4)),'-
DRP')+B(1:7,:).*NumericalDerivative(1,6,dy,-(rho(1:7,:,4))','-DRP')'-
C(1:7,:).*rho(1:7,:,4);
        Dub = A(1:7,:).*NumericalDerivative(1,6,dx,-(u(1:7,:,4)),'-
DRP')+B(1:7,:).*NumericalDerivative(1,6,dy,-(u(1:7,:,4))','-DRP')'-
C(1:7,:).*u(1:7,:,4);
        Dvb = A(1:7,:).*NumericalDerivative(1,6,dx,-(v(1:7,:,4)),'-
DRP')+B(1:7,:).*NumericalDerivative(1,6,dy,-(v(1:7,:,4))','-DRP')'-
C(1:7,:).*v(1:7,:,4);
        Dpb = A(1:7,:).*NumericalDerivative(1,6,dx,-(p(1:7,:,4)),'-
DRP')+B(1:7,:).*NumericalDerivative(1,6,dy,-(p(1:7,:,4))','-DRP')'-
C(1:7,:).*p(1:7,:,4);

        %top boundary
        if My == 0 %radiation boundary conditions
            Drhot = A(end-6:end,:).*NumericalDerivative(1,6,dx,-(rho(end-
6:end,:,4)),'-DRP')+B(end-6:end,:).*NumericalDerivative(1,6,dy,-(rho(end-
6:end,:,4))','-DRP')'-C(end-6:end,:).*rho(end-6:end,:,4);
            Dut = A(end-6:end,:).*NumericalDerivative(1,6,dx,-(u(end-
6:end,:,4)),'-DRP')+B(end-6:end,:).*NumericalDerivative(1,6,dy,-(u(end-
6:end,:,4))','-DRP')'-C(end-6:end,:).*u(end-6:end,:,4);
            Dvt = A(end-6:end,:).*NumericalDerivative(1,6,dx,-(v(end-
6:end,:,4)),'-DRP')+B(end-6:end,:).*NumericalDerivative(1,6,dy,-(v(end-
6:end,:,4))','-DRP')'-C(end-6:end,:).*v(end-6:end,:,4);
            Dpt = A(end-6:end,:).*NumericalDerivative(1,6,dx,-(p(end-
6:end,:,4)),'-DRP')+B(end-6:end,:).*NumericalDerivative(1,6,dy,-(p(end-
6:end,:,4))','-DRP')'-C(end-6:end,:).*p(end-6:end,:,4);
        else %outflow boundary conditions
            Drhot = NumericalDerivative(1,6,dx,-Mx*rho(end-
6:end,:,4)+Mx*p(end-6:end,:,4),'-DRP')+A(end-
6:end,:).*NumericalDerivative(1,6,dx,-p(end-6:end,:,4),'-
DRP')+NumericalDerivative(1,6,dy,(-My*rho(end-6:end,:,4)+My*p(end-
6:end,:,4))','-DRP')'+B(end-6:end,:).*NumericalDerivative(1,6,dy,-p(end-
6:end,:,4)','-DRP')'-C(end-6:end,:).*p(end-6:end,:,4);
            Dut = NumericalDerivative(1,6,dx,-(Mx*u(end-6:end,:,4)+p(end-
6:end,:,4)),'-DRP')+NumericalDerivative(1,6,dy,-(My*u(end-6:end,:,4))','-
DRP')';
            Dvt = NumericalDerivative(1,6,dx,-(Mx*v(end-6:end,:,4)),'-
DRP')+NumericalDerivative(1,6,dy,-(My*v(end-6:end,:,4)+p(end-6:end,:,4))','-
DRP')';
```

```
            Dpt = A(end-6:end,:).*NumericalDerivative(1,6,dx,-(p(end-
6:end,:,4)),'-DRP')+B(end-6:end,:).*NumericalDerivative(1,6,dy,-(p(end-
6:end,:,4))','-DRP')'-C(end-6:end,:).*p(end-6:end,:,4);
        end

        %stitch together interior and boundary nodes
        Drho(:,:,4) = [Drhol(:,1:3) [Drhob(1:3,4:end-3); Drhoi(4:end-3,4:end-
3); Drhot(5:7,4:end-3)] Drhor(:,5:end)];
        Du(:,:,4) = [Dul(:,1:3) [Dub(1:3,4:end-3); Dui(4:end-3,4:end-3);
Dut(5:7,4:end-3)] Dur(:,5:end)];
        Dv(:,:,4) = [Dvl(:,1:3) [Dvb(1:3,4:end-3); Dvi(4:end-3,4:end-3);
Dvt(5:7,4:end-3)] Dvr(:,5:end)];
        Dp(:,:,4) = [Dpl(:,1:3) [Dpb(1:3,4:end-3); Dpi(4:end-3,4:end-3);
Dpt(5:7,4:end-3)] Dpr(:,5:end)];

        %calculate physical variables at new time period
        rhoswitch =
rho(:,:,4)+dt*(b(1)*Drho(:,:,4)+b(2)*Drho(:,:,3)+b(3)*Drho(:,:,2)+b(4)*Drho(:
,:,1));
        uswitch =
u(:,:,4)+dt*(b(1)*Du(:,:,4)+b(2)*Du(:,:,3)+b(3)*Du(:,:,2)+b(4)*Du(:,:,1));
        vswitch =
v(:,:,4)+dt*(b(1)*Dv(:,:,4)+b(2)*Dv(:,:,3)+b(3)*Dv(:,:,2)+b(4)*Dv(:,:,1));
        pswitch =
p(:,:,4)+dt*(b(1)*Dp(:,:,4)+b(2)*Dp(:,:,3)+b(3)*Dp(:,:,2)+b(4)*Dp(:,:,1));


        %%shift physical variable matrices
        rho = circshift(rho,shift);
        u = circshift(u,shift);
        v = circshift(v,shift);
        p = circshift(p,shift);

        %update physical variables
        rho(:,:,4) = rhoswitch;
        u(:,:,4) = uswitch;
        v(:,:,4) = vswitch;
        p(:,:,4) = pswitch;

        %plot variables at specified times
        if any(abs(dt*i - plot_times) <= eps) || i ==1
            pcolor(X,Y,rho(:,:,end)); colorbar; shading interp;
colormap(flipud(gray)); xlabel('x'); ylabel('y'); title(['Density contour for
t = ',num2str(dt*i),', dx = ', num2str(dx),', dt = ', num2str(dt)]);
            saveas(gcf,['density_i',num2str(i)],'fig');

            pcolor(X,Y,u(:,:,end)); colorbar; shading interp;
colormap(flipud(gray)); xlabel('x'); ylabel('y'); title(['U velocity contour
for t = ',num2str(dt*i),', dx = ', num2str(dx),', dt = ', num2str(dt)]);
            saveas(gcf,['u_i',num2str(i)],'fig');

            pcolor(X,Y,v(:,:,end)); colorbar; shading interp;
colormap(flipud(gray)); xlabel('x'); ylabel('y'); title(['V velocity contour
for t = ',num2str(dt*i),', dx = ', num2str(dx),', dt = ', num2str(dt)]);
            saveas(gcf,['v_i',num2str(i)],'fig');
            pcolor(X,Y,p(:,:,end)); colorbar; shading interp;
colormap(flipud(gray)); xlabel('x'); ylabel('y'); title(['Pressure contour
for t = ',num2str(i),', dx = ', num2str(dx),', dt = ', num2str(dt)]);
            saveas(gcf,['p_i',num2str(i)],'fig');
            close all;
        end
    end
    cd ..
    close(h);
    compute_time = toc
end
```

Appendix B: Numerical Derivative Routine

```
function [deriv] = NumericalDerivative(Dorder, Horder, h, phi,varargin)
    %Numerically derivate matrix given constant step size
    %Code Version: 3.2 @ 2011-02-23
    %This is not complete; DRP scheme is only available for central
    %differencing
    %Inputs:
    %   Dorder: Derivative Order
    %   Horder: Number of terms to use
    %   h: step size
    %   phi: function to derivate
    %   options:    '-center' for central difference only
    %               '-upstream' for backward difference only
    %               '-downstream' for forward difference only
    %               '-DRP' for 4th order DRP scheme
    %Outputs:
    %   deriv: Dorder derivative of phi with Horder accuracy

    [~, N] = size(phi);

    %set default scheme (if not given by user)
    if any(strcmp(varargin,'-DRP'))
        if Dorder ~= 1 || Horder ~= 6
            warning('DRP scheme only available for 1st derivative, 4th order.
Switching to standard Taylor series expansion mode.'); %#ok<WNTAG>
            varargin(strcmp(varargin,'-DRP')) = '';
        end
        if length(varargin) ==1
            varargin{2} = '-center';
        end
    end

    if isempty(varargin)
       varargin{1} = '-center';
    end

    %calculate coefficients for differencing operations
    if any(strcmp(varargin,'-center'))
        if mod(Horder,2)
            error('Given accuracy order cannot be accomplished with central
finite difference method; please provide even accuracy order');
        end
        %central differencing coefficients
        n = Horder/2+floor((Dorder-1)/2);
        if any(strcmp(varargin,'-DRP'))
            coefs = DRPlookup(3,3,h);
            coefm = spdiags(repmat(coefs',N,1),-3:3,N,N);
        else
            coefs = TSE(n,n,h,Dorder);
            coefm = spdiags(repmat(coefs',N,1),-n:n,N,N);
        end

        %left bound coefficients
        for i = 1:Horder/2+floor((Dorder-1)/2)
            if any(strcmp(varargin,'-DRP'))
```

```
                coefm(i,1:7) = DRPlookup(i-1,7-i,h);
            else
                coefm(i,1:2*n+Dorder) = TSE(i-1,Horder+Dorder-i,h,Dorder)';
            end
        end

        %right bound coefficients
        for i = N-Horder/2+floor((Dorder-1)/2)+1:N
            if any(strcmp(varargin,'-DRP'))
                coefm(i,end-6:end) = DRPlookup(-N+6+i,N-i,h);
            else
                coefm(i,end+1-(2*n+Dorder):end) = TSE(-N+Horder+Dorder-1+i,N-
i,h,Dorder)';
            end
        end

    elseif any(strcmp(varargin,'-upstream'));
        %Upstream differencing coefficients
        coefs = TSE(Horder+Dorder-1,0,h,Dorder);
        coefm = spdiags(repmat(coefs',N,1),-(Horder+Dorder-1):0,N,N);

        %Left bound coefficients
        for i = 1:Horder+Dorder-1
           coefm(i,1:Horder+Dorder) = TSE(i-1,Horder+Dorder-i,h,Dorder)';
        end

    elseif any(strcmp(varargin,'-downstream'));
        %Downstream differencing coefficients
        coefs = TSE(0,Horder+Dorder-1,h,Dorder);
        coefm = spdiags(repmat(coefs',N,1),0:Horder+Dorder-1,N,N);

        %Right bound coefficients
        for i = N-Horder-Dorder+1:N
           coefm(i,end-(Horder+Dorder-1):end) = TSE(-N+Horder+Dorder-1+i,N-
i,h,Dorder)';
        end
    end
    deriv = phi*(coefm'); %perform finite differencing
end

function [coefs] = DRPlookup(n,m,h)
%Looks up coefficients for 4th Order, 1st Derivative DRP scheme
    a{33} = [-0.02084314277031176 ...
                0.166705904414580469 ...
                -0.77088238051822552 ...
                0 ...
                0.77088238051822552 ...
                -0.166705904414580469 ...
                0.02084314277031176]';

    a{42} = [0.02636943100 ...
                -0.16613853300 ...
                0.518484526d0 ...
                -1.27327473700 ...
                0.47476091400 ...
                0.46884035700 ...
                -0.049041958d0]';

    a{51} = [-0.048230454 ...
                0.281814650 ...
                -0.768949766 ...
                1.388928322 ...
                -2.147776050 ...
                1.084875676 ...
                0.209337622]';

    a{60} = [0.203876371 ...
                -1.128328861 ...
                2.833498741 ...
                -4.461567104 ...
                5.108851915 ...
                -4.748611401 ...
                2.192280339]';

    a{24} = -flipud(a{42});
    a{15} = -flipud(a{51});
    a{06} = -flipud(a{60});

    coefs = a{str2double(strcat([num2str(n),num2str(m)]))}/h;
end
```