```
% David O. Lignell
% 12/27/03
% Constant density, 2-D, laminar flow solver
% Constant viscosity, constant x and y grid
% Time accurate, explicit solver.  (or at least time advance to steady state)
% Explicit Euler is used
% Pressure solve, not pressure correction
% Allow for wall, inlet, outlet BC
% See make grid for setup

clc;
clear;

make_grid;

Re = 500;
mu = 1.0e-3;
nu = 1.0e-6;
rho = mu/nu;
v_in = Re*nu/W;
u_in = 0.0;
mass_in = (v_in^2+u_in^2)^0.5 * 4;       % grid specific
gx = 0.0;
gy = 0.0;

%Simulation control                      (stolen from Jeremy's soln)
ttime = 0.1*W^2/nu;                      %total time iterations
CFL = 1.0;                               %CFL constant
dtc = CFL*0.5*dx/v_in;                   %time step, convective
dtd = CFL*0.5*dx^2/nu;                   %time step, diffusive
dt = min(dtc,dtd)/1.0;                   %choose a dt
rtol = 1.0e-6;

ttime = ttime / 2;
% Create and initialize field variables

u_vel     = zeros(nx,ny);
v_vel     = u_vel;
pressure  = u_vel;
u_cc      = zeros(nx,ny);
v_cc      = u_cc;
v_vel(nx/2-1:nx/2+2,1:2) = v_in;

u_hat = zeros(nx,ny);
v_hat = zeros(nx,ny);

Ap_init = 1;                                % Flag to calc Ap in get_Ap once only
time = 0.0;
titer = 0;

while time < ttime
    time = time + dt;
    titer = titer + 1;

    fprintf('-------------------------------------\n')
    fprintf('       Time = ')
    fprintf('%6.2f\n',time)
    fprintf('Iteration # = ')
    fprintf('%6.0i\n',titer)

    get_uv_hat2;
    get_Ap;                                 % Get pressure equation Ap * P = Bp m
atricies
    P = Ap\Bp;
    P = reshape(P,nx-2,ny-2);               % P size here is nx-2, ny-2
    %P = P - sum(sum(P))/((nx-2)*(ny-2));   % Uncomment if Ap, Bp don't force a r
eference P
                                            % Code works with uncommented and Ap,
 Bp general, not sure why (walls?)
    % get u and v velocities at n+1

    for i=3:nx-1
        u_vel(i,2:ny-1) = u_hat(i,2:ny-1) + dt/rho/dx*( P(i-1-1,1:ny-2) - P(i-1,1:ny
```

```
-2) );
    end
    for j=3:ny-1
        v_vel(2:nx-1,j) = v_hat(2:nx-1,j) + dt/rho/dy*( P(1:nx-2,j-1-1) - P(1:nx-2,j
-1) );
    end

    get_max_divergence;                         % Test continuity and print to screen

    for i=2:nx-1
        u_cc(i,2:ny-1) = 0.5*( u_vel(i,2:ny-1)+u_vel(i+1,2:ny-1) );
    end
    for j=2:ny-1
        v_cc(2:nx-1,j) = 0.5*( v_vel(2:nx-1,j)+v_vel(2:nx-1,j+1) );
    end
    KE(titer,1) = sum(sum( (u_cc.*u_cc + v_cc.*v_cc).^0.5));   % total KE as converg
ence indicator
    fprintf('KE = ');
    fprintf('%6.2e\n', KE(titer));

    %plot the stuff:            ( this part stolen from Jeremy's soln)
    if (mod(titer,30) == 0)
        close all
        subplot(221), quiver(x',y',u_cc(2:nx-1,2:ny-1),v_cc(2:nx-1,2:ny-1));title('v
el vectors')
        axis image
        subplot(222), plot(KE); title('Kinetic Energy');
        subplot(223), streamslice(u_cc',v_cc');title('stream lines')
        axis image
        subplot(224), contourf(v_cc',10);title('u-vel contour');
        axis fill
        pause(0.5)

    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

% get_Ap.m

% David O. Lignell
% Routine returns the Coefficient matrix for the pressure equation
%    As well as the rhs vector Bp
% e.g. Ap * P = Bp,  or P = Ap/Bp
% Ap should not change, so we only have to do this once
% Note for diag -1, the last element of the diagonal is trimmed, while for
% diag 1 the first element is trimmed (if the diagonals are the size of the mat)
% Note we fix a value of P=0 at about the middle cell.  This can be commented out
%   but be sure to uncomment the P shift for the soln in ns.m (e.g. the shift around
 the average).

nxv = nx-2;                                 % number of p cells x-dir
nyv = ny-2;
nvars = nxv*nyv;
Bp = zeros(nx,ny);   % not nxv, nyv

if Ap_init == 1

    ds = repmat(dt*dx/rho/dy, nvars,1);              % south neighbor diagonal
    dn = ds;
    de = repmat(dt*dy/rho/dx, nvars,1);
    dw = de;
    dp = repmat(-2*dt/rho*(dx/dy+dy/dx),nvars,1);

    de([1:nxv:nvars])  = 0.0;     % correct right wall
    dw([nxv:nxv:nvars]) = 0.0;    % correct left wall
    % correct top and bottom
    dp([1:nxv nvars-nxv+1:nvars]) = dp([1:nxv nvars-nxv+1:nvars]) + dt/rho*dx/dy;
    % correct left wall
    dp(1:nxv:nvars) = dp(1:nxv:nvars) + dt/rho*dy/dx;
    % correct right wall
    dp(nxv:nxv:nvars) = dp(nxv:nxv:nvars) + dt/rho*dy/dx;
```

```
    Ap = spdiags([ds dw dp de dn], [-nxv -1 0 1 nxv], nvars, nvars);

    Ap(floor(nvars/2+nxv/2),:) = 0;                    % specify pressure at one poi
nt
    Ap(floor(nvars/2+nxv/2),floor(nvars/2+nxv/2)) = 1.0;

    % get the right hand side here
    Ap_init = 0;
end

for i=2:nx-1
    for j=2:ny-1
        Bp(i,j) = dy*(u_hat(i+1,j)-u_hat(i,j)) + dx*(v_hat(i,j+1)-v_hat(i,j));
    end
end
Bp(:,2)    = Bp(:,2) - dx*v_vel(:,2);              % fix bottom
Bp(:,ny-1) = Bp(:,ny-1) + dx*v_vel(:,ny);          % fix top
Bp(2,:)    = Bp(2,:) - dy*u_vel(2,:);              % fix left
Bp(nx-1,:) = Bp(nx-1,:) + dy*u_vel(nx,:);          % fix right

Bp = reshape(Bp(2:nx-1,2:ny-1),nvars,1);
Bp(floor(nvars/2+nxv/2),1) = 0.0;                   % specifying pressure at one p
oint


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

% get_max_divergence.m

% David O. Lignell
% Compute maximum cell divergence to test satisfaction of the continuity equation

for i=2:nx-1
    for j=2:ny-1
        div(i,j) = (u_vel(i+1,j)-u_vel(i,j))*dy + dx*(v_vel(i,j+1)-v_vel(i,j));
    end
end

maxdiv = max(max(abs(div)));
[imd,jmd] = find(abs(div) == maxdiv);
maxv   = max([max(abs(v_vel)) max(abs(u_vel))]);
fprintf('maxdiv = ')
fprintf('%6.2e\n', maxdiv);
fprintf('maxu = ')
fprintf('%6.2e\n', maxv);
%imd
%jmd

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

% get_uv_hat2.m

% David O. Lignell 12/29/03
% u hat is basically the rhs (all but unsteady term) of the mom eqn, without pressur
e,
%   and with the old velocity
% e.g. u(n+1) = uhat(n) + dt/rho/dx*(Pw-Pe)
% So, the algorithm gets uhat, then plugs the above into continuity and solves P,
%    then solves the above for u(n+1)
%
% This technique is very simple and straight forward.
% Every cell in FLOW domain has the relevant face velocities and derivatives compute
d
% For boundaries, just make sure the face value of vel and dv/dx are correct
% The outlet is treated by assuming u is 0, and v is the adjacent value (upstream) +
 a velocity
%    Correction to force mass conseration.  Note that this is done immediately at ti
me = 0.
% Alternately, we could extrapolate the velocity some way, then apply a correction.
% Note that this is not the same as assuming the gradient of velocity is zero
```

```
v_vel(2:nx-1,ny) = v_vel(2:nx-1,ny-1);               % Set the v-vel at the outlet
u_vel(3:nx-1,ny) = 0.0;
mass_out = sum(v_vel(2:nx-1,ny));
v_corr = (mass_in-mass_out)/(nx-2);
v_vel(2:nx-1,ny) = v_vel(2:nx-1,ny)+v_corr;
mass_out = sum(v_vel(2:nx-1,ny));
if abs((mass_in-mass_out)/mass_in) > 1.0e-6;
    fprintf('error in outlet velocity');
end

for i=3:nx-1                                  % For u cells
    for j=2:ny-1

        ue    = 0.5*(u_vel(i+1,j)+u_vel(i,j));
        uw    = 0.5*(u_vel(i-1,j)+u_vel(i,j));
        un    = 0.5*(u_vel(i,j+1)+u_vel(i,j));
        us    = 0.5*(u_vel(i,j-1)+u_vel(i,j));
        vn    = 0.5*(v_vel(i,j+1)+v_vel(i-1,j+1));
        vs    = 0.5*(v_vel(i-1,j)+v_vel(i,j));
        dudxe = (u_vel(i+1,j)-u_vel(i,j))/dx;
        dudxw = (u_vel(i,j)-u_vel(i-1,j))/dx;
        dudyn = (u_vel(i,j+1)-u_vel(i,j))/dy;
        dudys = (u_vel(i,j)-u_vel(i,j-1))/dy;
        dvdxn = (v_vel(i,j+1)-v_vel(i-1,j+1))/dx;
        dvdxs = (v_vel(i,j)-v_vel(i-1,j))/dx;

        if j == ny-1
            un    = u_vel(i,j+1);
            dudyn = dudyn * 2.0;
        elseif j == 2
            us    = u_vel(i,j-1);
            dudys = dudys*2.0;
        end


        u_hat(i,j) = ( rho*dy*(uw*uw-ue*ue) + rho*dx*(us*vs-un*vn) + ...
                       2.0*mu*dy*(dudxe-dudxw) + mu*dx*(dudyn-dudys) + ...
                       mu*dx*(dvdxn-dvdxs) + ...
                       rho*gx*dx*dy ) * dt/(rho*dx*dy) + u_vel(i,j);

    end
end

for i=2:nx-1                                      % v cells
    for j=3:ny-1

        vn    = 0.5*(v_vel(i,j+1)+v_vel(i,j));
        vs    = 0.5*(v_vel(i,j-1)+v_vel(i,j));
        ve    = 0.5*(v_vel(i+1,j)+v_vel(i,j));
        vw    = 0.5*(v_vel(i-1,j)+v_vel(i,j));
        ue    = 0.5*(u_vel(i+1,j)+u_vel(i+1,j-1));
        uw    = 0.5*(u_vel(i,j)+u_vel(i,j-1));
        dvdyn = (v_vel(i,j+1)-v_vel(i,j))/dy;            % is zero at j=ny-1
        dvdys = (v_vel(i,j)-v_vel(i,j-1))/dy;
        dvdxe = (v_vel(i+1,j)-v_vel(i,j))/dx;
        dvdxw = (v_vel(i,j)-v_vel(i-1,j))/dx;
        dudye = (u_vel(i+1,j)-u_vel(i+1,j-1))/dy;
        dudyw = (u_vel(i,j)-u_vel(i,j-1))/dy;

        if i == 2
            vw    = v_vel(i-1,j);
            dvdxw = dvdxw * 2.0;
        elseif i == nx-1
            ve    = v_vel(i+1,j);
            dvdxe = dvdxe * 2.0;
        end

        v_hat(i,j) = v_vel(i,j) + dt/(rho*dx*dy) * ...
            ( rho*dx*(vs*vs-vn*vn) + rho*dy*(vw*uw-ve*ue) + ...
            2.0*mu*dx*(dvdyn-dvdys) + mu*dy*(dvdxe-dvdxw) + ...
            mu*dy*(dudye-dudyw) + ...
            rho*gy*dx*dy );
```

```
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%

% make_grid.m

% David O. Lignell
% 12/27/03
% make_grid.m
% called by ns.m
% specify grid dimensions, cell types
% Code assumes constant delta_x, and delta_y
% Obviously there should be a whole number of cells in each dir so make L/ny whole,
etc.
% This grid is for the slot jet problem of phils cfd class
% Inlet area is 0.2*W, and is 4 cells across
%
%      |    outlet BC   |
%      |                |
%      |                |
%      |                |
%      |    Fluid       |
%      |                | Walls
%      |                |
%      |       ^        |            ^ y , j
%      |       |        |            |
%      |____|in|_____|        x,i    |---->        Lower right corner = (0,0)

L   = 4.0;                      % y-dir grid width
W   = 1.0;                      % x-dir grid length

% Set base flow domain

nx = 80;
ny = nx*L/W;
dx = W/nx;
dy = L/ny;

[x y]  = meshgrid(dx:dx:W, dy:dy:L);
% Add ghost cells (for simple grids this is the boundary cells)

nx = nx + 2;
ny = ny + 2;

% Define four valid cell types (flow, wall, inlet, outlet)
% Flow is 0, inlet is 1, outlet is 2, wall is 3
% Assume flow only in interior cells, inlet and outlet only on boundary (ghost cells
)
% and wall cells anywhere

cell_types = zeros(nx,ny);  % not used in this code

cell_types(1,:)        = 3;
cell_types(end,:)      = 3;
cell_types(2:nx-1,end) = 2;

win = W*0.2;
cell_types(nx/2-1:nx/2+2,1) = 1;
```