Brian Knisely
ME523 Assignment 8

Problem 1

```
%%% solve for temperature %%%

function calct

% solve for temperature

% global variables needed by this function
  global nij nim njm nj
  global urf ien
  global gfacx gfacy
  global xf yf xc yc
  global li
  global visc prr densit dtr
  global f1 f2
  global gds
  global ltime
  global t0

% global variables set or modified by this function
  global su ap awe aso ano aea
  global t

%%% initialize su and ap to zero
  su(:) = 0;
  ap(:) = 0;

%%% compute convection and diffusion contributions for east and west faces
% set contributions to aea, awe, and su
% use CDS for diffusion
% use CDS/UDS blend for convection
% keep UDS convection on left-hand side of the linear system (implicit terms),
%   and put the rest of the convection contribution on the right-hand side
%   as a deferred correction; see calcuv for examples

  for i=2:nim-1
    fxe  = gfacx(i); % cell-face geometric interpolation factors
    fxp  = 1. - fxe; % 1 - (cell-face geometric interpolation factors)
    dx = xc(i) - xc(i-1); % current dx value

    for j=2:njm
      ij  = li(i) + j; % current index translated to 1D array index
      ije = ij + nj; % next index to the east translated to 1D array index

      dy = yf(j) - yf(j-1);

      % diffusive flux coefficient
      d = visc*prr*dy/dx;
      ce = min(f1(ij), 0.);
      cp = max(f1(ij), 0.);

      aea(ij) = ce - d;
      awe(ije) = -cp - d;

      fuds = cp*t(ij) + ce*t(ije);
      fcds = f1(ij)*(t(ije)*fxe + t(ij)*fxp);

      su(ij)  = su(ij) + gds(ien) * (fuds - fcds);
      su(ije) = su(ije) - gds(ien) * (fuds - fcds);

    end
  end

%%% compute convection and diffusion contributions for north and south faces
% set contributions to ano, aso, and su
% use CDS for diffusion
% use CDS/UDS blend for convection
% keep UDS convection on left-hand side of the linear system (implicit terms),
%   and put the rest of the convection contribution on the right-hand side
%   as a deferred correction; see calcuv for examples
  for j=2:njm-1
    fyn  = gfacy(i); % cell-face geometric interpolation factors
    fyp  = 1. - fyn; % 1 - (cell-face geometric interpolation factors)
    dy = yc(j) - yc(j-1); % current dy value

    for i=2:nim
      ij  = li(i) + j; % current index translated to 1D array index
      ijn = ij + 1; % next index to the east translated to 1D array index

      dx = xf(i) - xf(i-1);

      % diffusive flux coefficient
      d = visc*prr*dx/dy;

      % convective fluxes
      cn = min(f2(ij), 0.);
      cp = max(f2(ij), 0.);

      ano(ij) = cn - d;
      aso(ijn) = -cp - d;

      fuds = cp*t(ij) + cn*t(ijn);
      fcds = f2(ij) * (t(ijn)*fyn + t(ij)*fyp);

      su(ij)  = su(ij) + gds(ien) * (fuds - fcds);
      su(ijn) = su(ijn) - gds(ien) * (fuds - fcds);

    end
  end

%%% compute volume integrals
% here the only contribution is from the unsteady term
```

```
  for i = 2:nim;
    dx = xf(i) - xf(i-1);

    for j = 2:njm;
      ij = li(i) + j;
      dy = yf(j) - yf(j-1);
      vol = dx*dy;

      if ltime == 1;
        apt = densit*vol*dtr;
        su(ij) = su(ij) + apt*t0(ij);
        ap(ij) = ap(ij) + apt;
      end
    end
  end
  %%% apply bcs
  bct;

  %%% apply under-relaxation to ap and su
  % see calcuv for examples
  for i = 2:nim;
    for ij = li(i) + 2: li(i) + nj
        ap(ij) = (ap(ij) - awe(ij) - aea(ij) - ano(ij) - aso(ij))/urf(ien);
        su(ij) = su(ij) + (1. - urf(ien))*ap(ij)*t(ij);
    end
  end

  %%% solve for temperature
  sipsol(ien)

endfunction

%%% end of calct %%%
```

Problem 2

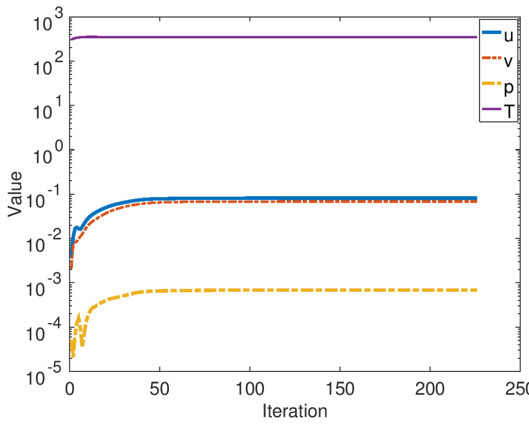

Figure 1: Convergence of velocities, pressure, and temperature

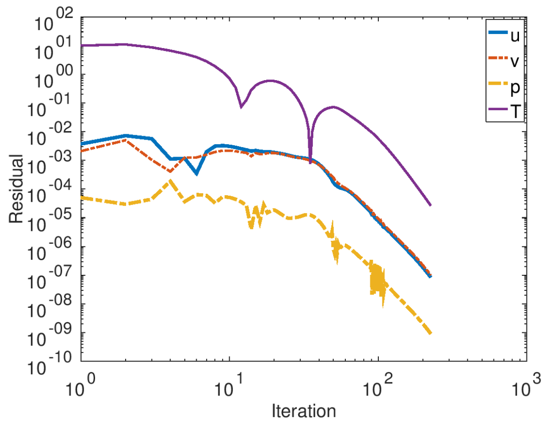

Figure 2: Residuals of velocities, pressure, and temperature

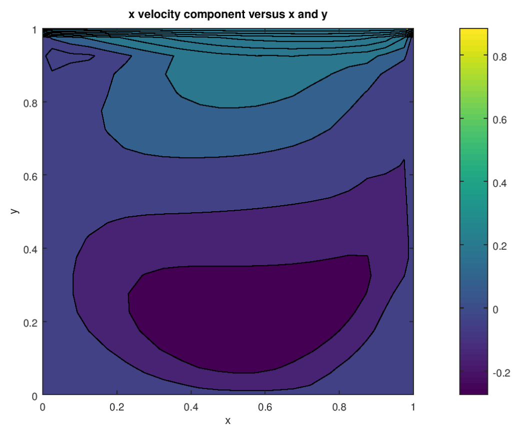

Figure 3: x velocity component versus x and y

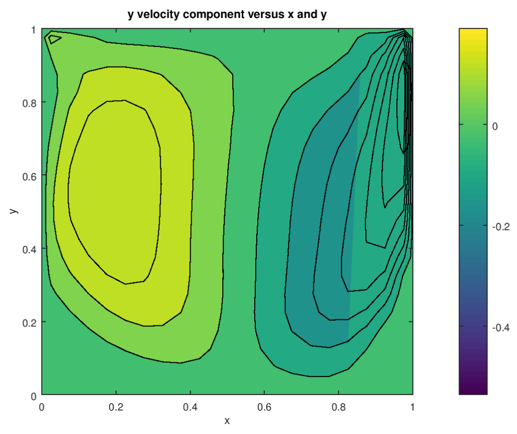

Figure 4: y velocity component versus x and y

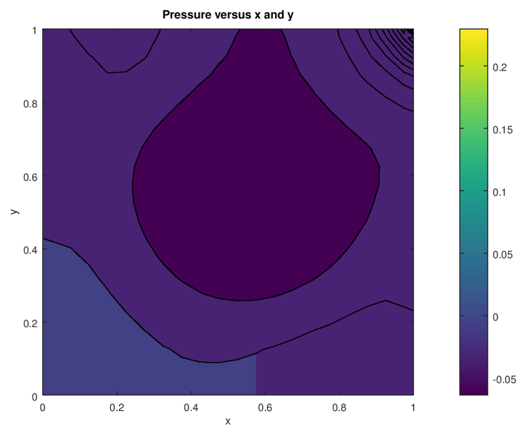

Figure 5: Pressure versus x and y

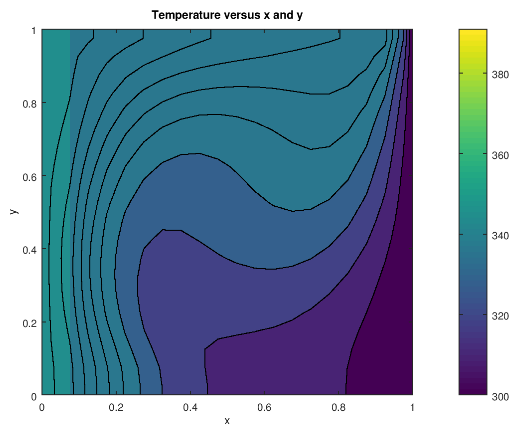

Figure 6: Temperature versus x and y

Note: the convergence and residual plot are displaying the absolute values to accommodate log scales on the y-axis of each.

From the contours, it is evident that the flow is swirling in a clockwise direction. With the "lid" of the box constantly moving to the right, the fluid is being forced to move to the right at the top of the box (as shown in the x-velocity plot). In the y-velocity plot, the fluid has a negative (downward) velocity near the right side. In the pressure plot, the Bernoulli effect is demonstrated, with the lowest pressure at the center of the box (where the velocity is the lowest). The highest pressure is found at the area with the highest velocity, in the top right corner.

Problem 3

• Approximately 1000 time steps and 1 hour of CPU time were needed with dt=0.1 for the unsteady solution to converge on the steady solution within two significant figures. The temperature value converged the most quickly, then velocities, and finally pressure required the most time to converge.

• The number of outer iterations required for each time step decreases as the solution approaches steady state. Using the previous time-step results as an estimate for the next time step becomes a better approximation with time. At steady state, the values at both the current and previous time step are equal.

- The unsteady code was tested with dt = 1, dt = 10, dt = 100, dt = 1000, dt = 10000, and dt = 100000. With the implicit Euler method, there is theoretically no limit to the timestep that can be taken. The implicit Euler method is supposed to be unconditionally stable for a simple 1D problem. With two dimensions and other numerical issues possible, the method is not necessarily unconditionally stable. That being said, the solution converged for all tested values of dt. The maximum value of dt for which the solution converges, if there is one, is greater than 100000 seconds.

-

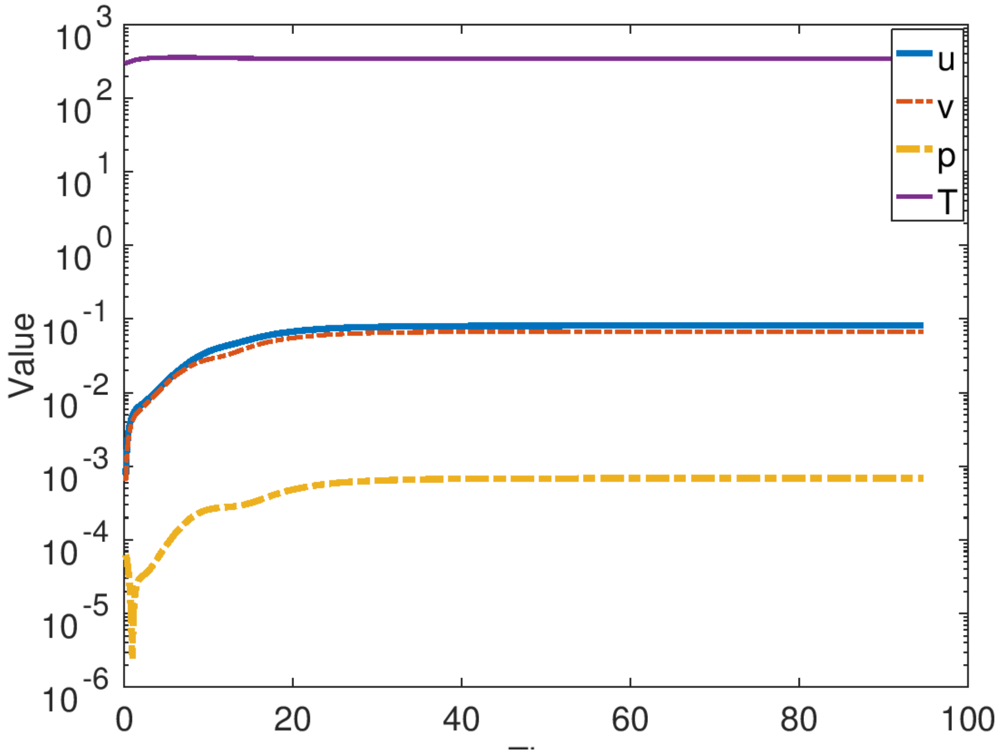

Figure 7: Parameters in the unsteady problem as a function of time (note: the x-axis should be labeled ”Time” but the plot would not save without cutting off the bottom)