Solution: Problem 9P4

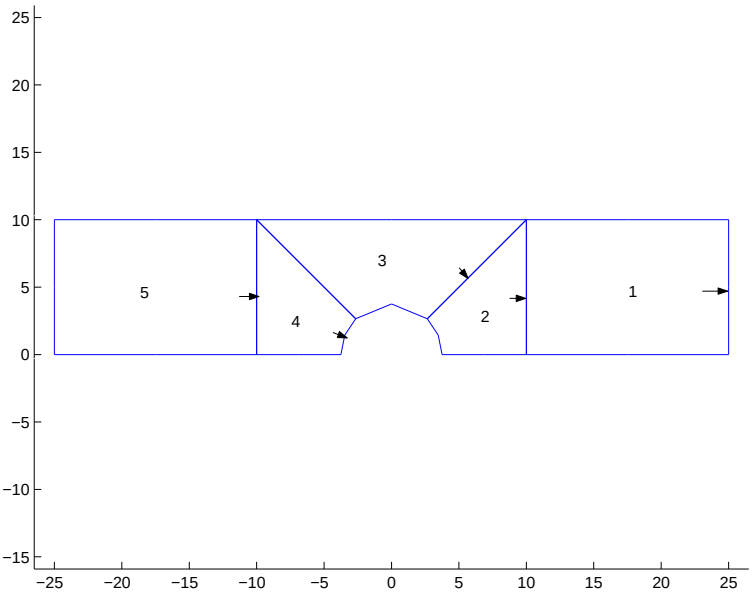
NOTE: This problem allows students to experience the ease with which finite element programs can be used to solve very complex problems. In this case, all that is necessary is that they have confidence in how to use the INCLUDE files. This, in turn, builds confidence in them to edit the main codes to their advantage when working other types of problems. I highly recommend this project as one of the more interesting ones for students to undertake.

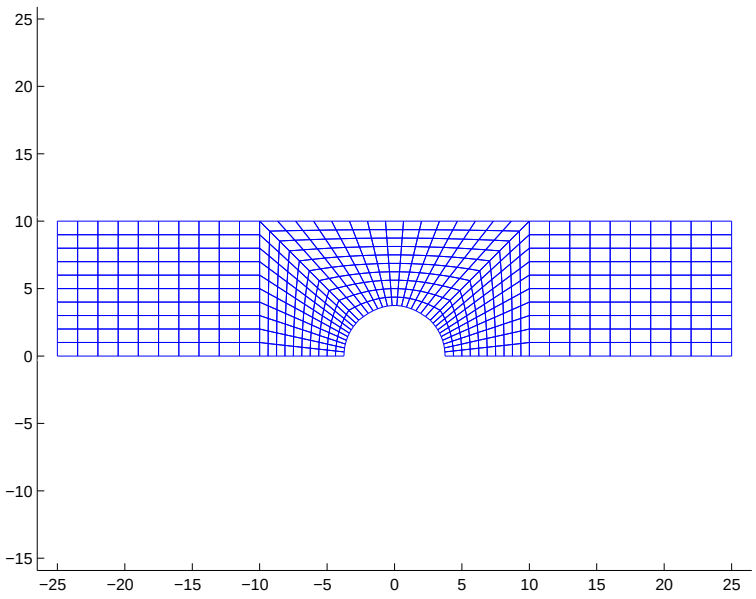
As is stated in the text, two analyses are necessary, (1) the flow of an ideal fluid, and (2) a steady state temperature field. The first analysis is the same as Project 2 in Chapter 7. If this was assigned, students should use some of the work from that project for this one.

The first step will be to create a mesh. The same mesh will be used for both of the analyses - in fact, this is essential.

Creating the MESH

A mesh of four node elements was created using the LOOPS shown. The arrows indicate the LOOP side that was selected as Side 1.






```
%-----------------------
%     NUMLPS    NNPE
%-----------------------
           5       8
%-----------------------
%     NDIV-1    NDIV-2
%-----------------------
          10        10
          10        10
          10        15
          10        10
          10        10
```

```
%    ----------------------
%    The  JOIN array
%    ----------------------
      0 0   0 0   0 0   0 0
      1 3   0 0   0 0   0 0
      2 2   0 0   0 0   0 0
      0 0   3 3   0 0   0 0
      4 3   0 0   0 0   0 0
```

The COORD data file is shown on the next page.

```
%----------------------------------
%     LOOP Coordinates (8 per loop)
%----------------------------------
    25       0
    25       5
    25      10
    17.5    10
    10      10
    10       5
    10       0
    17.5     0
%----------------------------------
    10       0
    10       5
    10      10
    6.3258   6.3258
    2.6516   2.6516
    3.4677   1.4350
    3.75     0
    6.875    0
%----------------------------------
    2.6516   2.6516
    6.3258   6.3258
    10      10
     0      10
   -10      10
   -6.3258   6.3258
   -2.6516   2.6516
     0       3.75
%----------------------------------
   -3.75     0
   -3.4677   1.4350
   -2.6516   2.6516
   -6.3258   6.3258
   -10      10
   -10       5
   -10       0
   -6.875    0
%----------------------------------
   -10       0
   -10       5
   -10      10
   -17.5    10
   -25      10
   -25       5
   -25       0
   -17.5     0
%----------------------------------
```

The user written NPCODE.m INCLUDE code is on the next page.

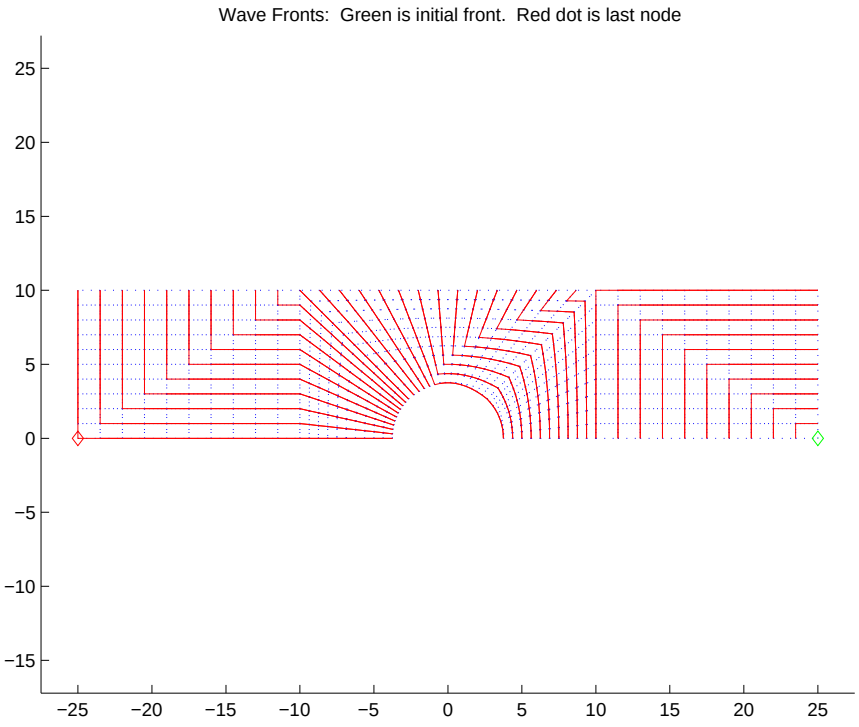
```
%--------------------------------------
%  NPCODE - user written INCLUDE code.
%--------------------------------------

%  ------------------------------------
%  LOOP 1
%  ------------------------------------
   n = NNPS-1;  % NNPS = number of nodes
                %   per side of element

       IEND = n*NDIV(1,1)+1;  % side 1
       for I=1:IEND
          NI=LNP(1,1,I);
          NPcode(NI)=11;
       end

       IEND = n*NDIV(1,2)+1;  % side 2
       for I=1:IEND
          NI=LNP(1,2,I);
          NPcode(NI)=12;
       end

       IEND = n*NDIV(1,4)+1;  % side 4
       for I=1:IEND
          NI=LNP(1,4,I);
          NPcode(NI)=14;
       end

%  ------------------------------------
%  LOOP 2
%  ------------------------------------
       IEND = n*NDIV(2,4)+1;  % side 4
       for I=1:IEND
          NI=LNP(2,4,I);
          NPcode(NI)=24;
       end

       IEND = n*NDIV(2,3)+1;  % side 3
       for I=1:IEND
          NI=LNP(2,3,I);
          NPcode(NI)=23;
       end

%  ------------------------------------
%  LOOP 3
%  ------------------------------------
       IEND = n*NDIV(3,2)+1;  % side 2
       for I=1:IEND
          NI=LNP(3,2,I);
          NPcode(NI)=32;
       end

       IEND = n*NDIV(3,4)+1; % side 4
       for I=1:IEND
          NI=LNP(3,4,I);
          NPcode(NI)=34;
       end
```

```
%  ------------------------------------
%  LOOP 4
%  ------------------------------------
       IEND = n*NDIV(4,4)+1; %side 4
       for I=1:IEND
          NI=LNP(4,4,I);
          NPcode(NI)=44;
       end

       IEND = n*NDIV(4,1)+1; %side 1
       for I=1:IEND
          NI=LNP(4,1,I);
          NPcode(NI)=41;
       end


%  ------------------------------------
%  LOOP 5
%  ------------------------------------
       IEND = n*NDIV(5,2)+1; %side 2
       for I=1:IEND
          NI=LNP(5,2,I);
          NPcode(NI)=52;
       end

       IEND = n*NDIV(5,3)+1; %side 3
       for I=1:IEND
          NI=LNP(5,3,I);
          NPcode(NI)=53;
       end

       IEND = n*NDIV(5,4)+1; %side 4
       for I=1:IEND
          NI=LNP(5,4,I);
          NPcode(NI)=54;
       end
```

In the above, the NPCODE numbers correspond to (loop side); hence, NPCODE = 34 means it is a node on side 4 of LOOP 3. These numbers are sufficient to specify all boundary conditions needed for the two analyses.

newnum numbering

The mesh numbering gives a bandwidth equal to 122; hence, newnum.m should be able to improve that. Several attempts were made using different initial waves, but 33 seems to be about the minimum bandwidth possible. This can be obtained from several starting waves, but the simplest is starting with only one node, and specify that as the original node 1. The waves are then:




```
    ENTER:
    ---------------------------------------
       1 node numbers for first wave
 <---------------------------------------
wave node  1  < 1

  ---------------------------------------
              Bandwidth
  ---------------------------------------
             IB  =    33

   Symmetric:     Bandwidth =  DOF*IB
   NonSymmetric:  Bandwidth =  2*DOF*IB-1

   DOF = degrees of freedom per node
  ---------------------------------------
```

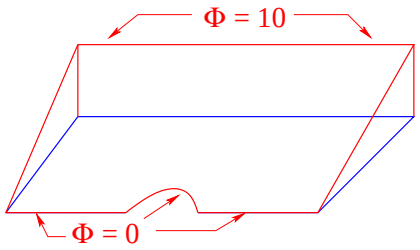
Flow Analysis

To conduct the flow analysis of a perfect fluid, we used the stream function approach so that

$$\frac{\partial \Phi}{\partial y} = u \quad \boxed{\text{velocity in x direction}}$$

$$\frac{\partial \Phi}{\partial x} = v \quad \boxed{\text{velocity in y direction}}$$

The velocity normal to all boundaries are known; hence, the derivative of Φ along these boundaries is known. On the top and lower boundaries, the derivative will be zero; hence, the value of Φ will be constant. On the left (entrance) boundary, and the right (exit) boundary, the derivative will be equal to the entrance and exit velocities, which we specify as unity; hence, the derivative of Φ along these boundaries will be unity. If we specify Φ values on all boundaries other than the pipe, as the y coordinate, and the values around the pipe as zero, then it will meet these boundary conditions. The following figure illustrates these boundary conditions, where the lower boundary is assumed to have a y coordinate of zero (see mesh). The upper boundary has a y coordinate equal to 10.



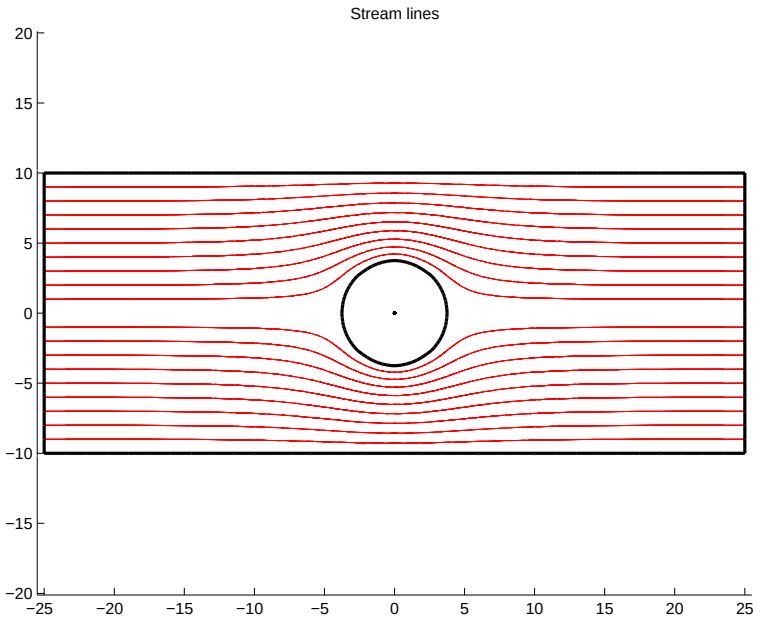
The INCLUDE codes, INITIAL.m and COEF.m are shown on the next page.

```
%---------------------------------
%/////////////////////////////////
%---------------------------------
%  INITIAL.m
%
%  Initialize arrays and specify
%  boundary conditions for program
%  steady.m
%
%  For use for the flow analysis.
%---------------------------------
     for I=1:NUMNP
        if NPcode(I)     == 11
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 12
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 14
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 23
           NPBC(I) = 1;
           PHI(I)=0;
        elseif NPcode(I) == 24
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 32
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 34
           NPBC(I) = 1;
           PHI(I)=0;
        elseif NPcode(I) == 41
           NPBC(I) = 1;
           PHI(I)=0;
        elseif NPcode(I) == 44
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 52
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 53
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        elseif NPcode(I) == 54
           NPBC(I) = 1;
           PHI(I)=YORD(I);
        end
     end
```

```
%--------------------------------------*
%                                      *
% COEF.m                               *
%                                      *
% d    d@  d    d@    d@   d@         *
% --(RX--)+--(RY--)+BX--+BY--+GV+HV=0 *
% dx   dx  dy   dy    dx   dy         *
%                                      *
%--------------------------------------*

    RXJ = 1;
    RYJ = 1;

    BXJ=0;
    BYJ=0;

    GVJ = 0;
    HVJ = 0;
```

After this analysis, you need to save the Φ values in another file which you will load for the temperature analysis. The name of the file I chose was VP.

A plot from topo.m of the stream lines is shown on the next page.



The temperature analysis

Now that we have the streamline function Φ, we can calculate the velocity at any point in our mesh; thus we will be able to perform the steady state temperature analysis.

The boundary conditions for the temperature analysis are known temperature equal to zero at the entrance (left hand boundary) and no heat transfer by conduction at the exit (right hand end). The top boundary is assumed insulated, and there is no heat transfer across the bottom boundary which is a plane of symmetry. The boundary that represents the pipe, is given a uniform flux equal to unity.

Note: when the entrance temperature is set to zero, very small oscillations around zero near the entrance cause program topo.m to plot many zero level contours. For that reason, I chose to 0.0001.

Program INITIAL.m loads the streamline values found from our previous analysis. These values, in the array `VP`, are used in COEF.m to calculate the velocity at each quadrature point. These velocities are used to in the calculation of the coefficients `BX` and `BY` in our equation. The user written codes INITIAL.m and COEF.m are:

```
%---------------------------------
%/////////////////////////////////
%---------------------------------
%  INITIAL.m
%
%  Initialize arrays and specify
%  boundary conditions for program
%  steady.m
%
%  For the temperature analysis.
%---------------------------------
     load VP

     for I=1:NUMNP
        if NPcode(I) == 23
   NPBC(I) = 0;
   QS(I)=1;
        elseif NPcode(I) == 34
   NPBC(I) = 0;
   QS(I)=1;
        elseif NPcode(I) == 41
   NPBC(I) = 0;
   QS(I)=1;
        elseif NPcode(I) == 53
   NPBC(I) = 1;
   PHI(I)=0.0001;
        else
     NPBC(I) = 0;
        end
     end
```

```
%--------------------------------------*
%                                      *
% COEF.m                               *
%                                      *
% d    d@  d    d@    d@   d@          *
% --(RX--)+--(RY--)+BX--+BY--+GV+HV=0  *
% dx   dx  dy   dy    dx   dy          *
%                                      *
%--------------------------------------*


    RXJ = 1;
    RYJ = 1;

    VX=0;
    VY=0;
    for i=1:NNPE
       PHIi=VP(NP(LMENT,i));
       VX=VX+DNDY(i)*PHIi;
       VY=VY-DNDX(i)*PHIi;
    end

    vFac =0.3;
    RhoCp=1.0;
    BXJ=-vFac*RhoCp*VX;
    BYJ=-vFac*RhoCp*VY;

    GVJ = 0;
    HVJ = 0;
```

Note that the streamline values, VC are for a unit velocity entering the system. For any other velocity, their value should be multiplied by the value desired. In the case shown, it was desired to have an entrance velocity of 0.3, hence `vFac` was set equal to this value. The maximum temperatures for each case are:

Velocity	Max. Temp.
0.0	34.778
0.3	6.643
0.6	4.747
1.0	3.888

The color plots of the temperature fields from `topo.m` are shown on the next two pages.

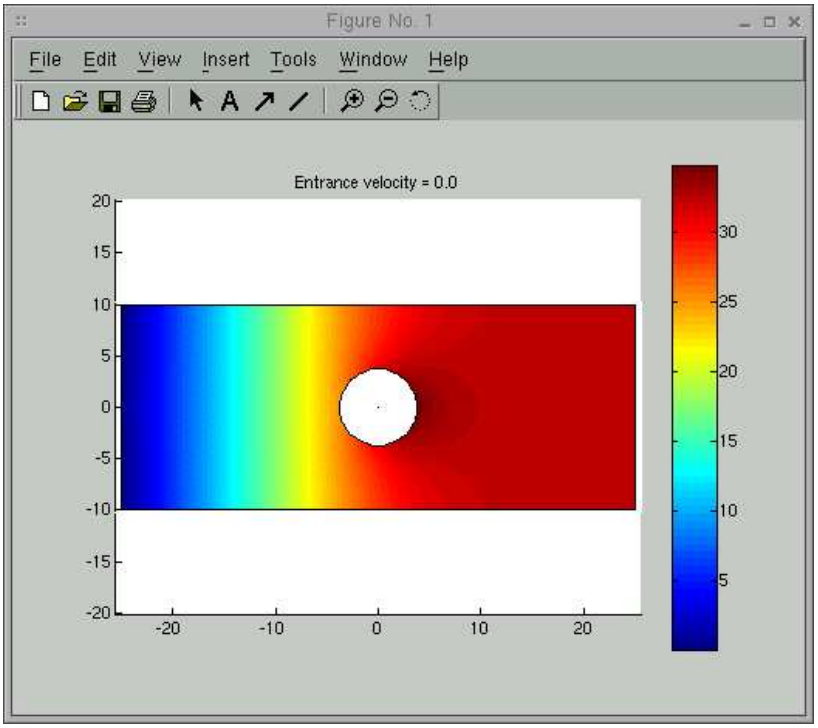


Figure No. 1
File Edit View Insert Tools Window Help
Entrance velocity = 0.0
20
15
10
5
0
-5
-10
-15
-20
-20 -10 0 10 20
30
25
20
15
10
5

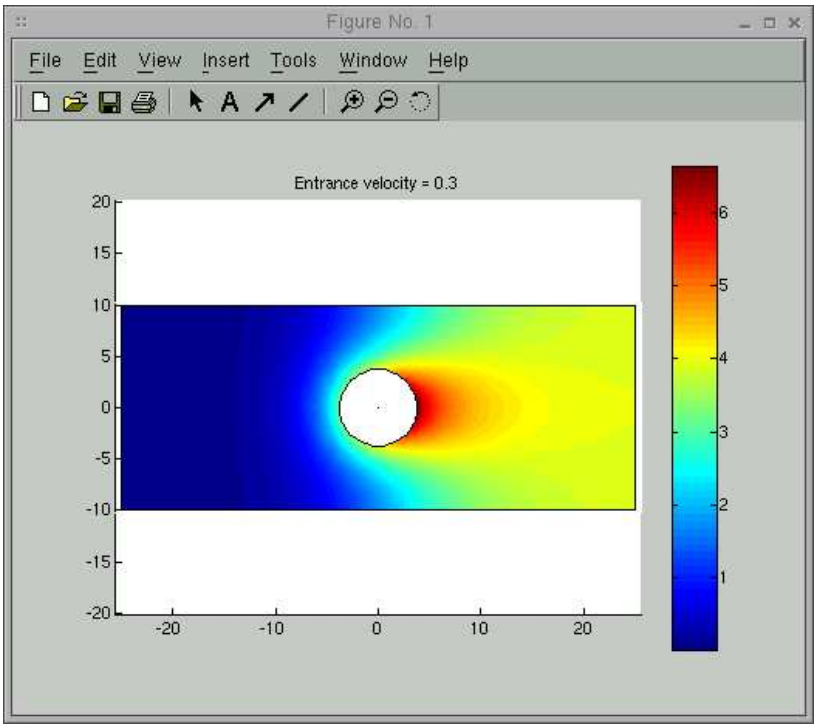


Figure No. 1
File Edit View Insert Tools Window Help
Entrance velocity = 0.3
20
15
10
5
0
-5
-10
-15
-20
-20 -10 0 10 20
6
5
4
3
2
1

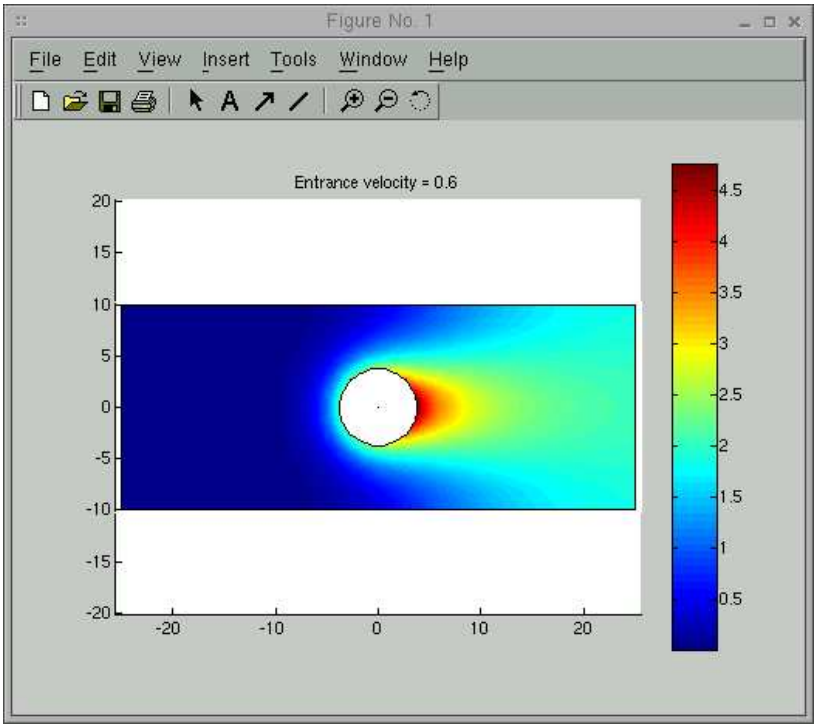


Figure No. 1
File Edit View Insert Tools Window Help
Entrance velocity = 0.6
20
15
10
5
0
-5
-10
-15
-20
-20 -10 0 10 20
4.5
4
3.5
3
2.5
2
1.5
1
0.5

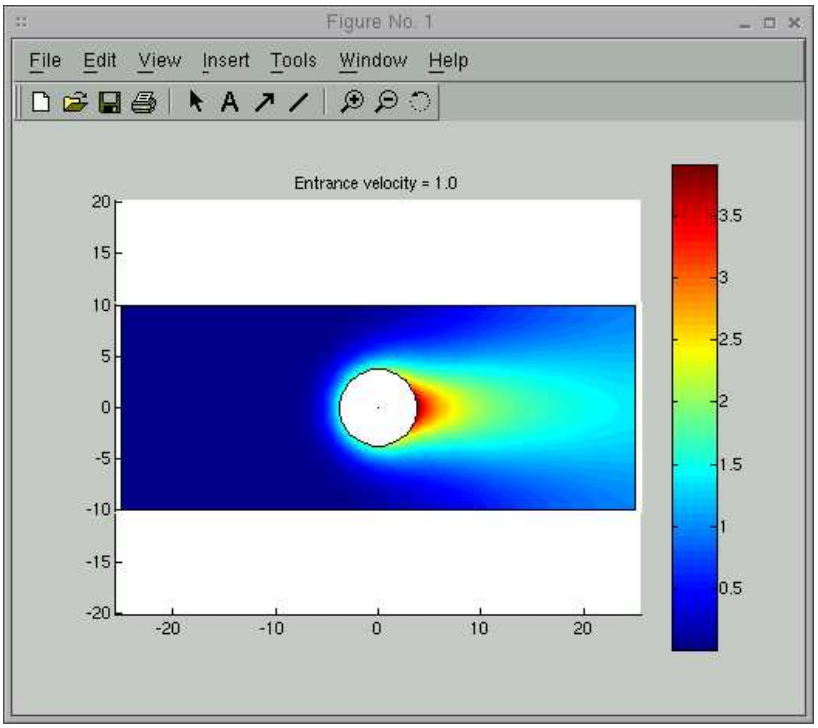


Figure No. 1
File Edit View Insert Tools Window Help
Entrance velocity = 1.0
20
15
10
5
0
-5
-10
-15
-20
-20 -10 0 10 20
3.5
3
2.5
2
1.5
1
0.5