### Nomenclature

|  |  |
| --- | --- |
| **Symbol / character** | **Meaning** |
| *Ψ( i , j )* | Stream function / Potential of the grid point having the location indices *i* and *j*. |
| *a* | Distance between the points (i-1,j) to (i,j). |
| *b* | Distance between the points (i,j) to (i+1,j). |
| *c* | Distance between the points (i,j-1) to (i,j). |
| *d* | Distance between the points (i,j) to (i,j+1). |
|  | |
|  | |
| **Subscript** | **Meaning** |
| *i* | Horizontal location index of any point in the problem domain. |
| *j* | Vertical location index of any point in the problem domain. |
| *( i , j )* | location indices of any point in the problem domain. |

### List of figures

|  |  |  |
| --- | --- | --- |
| **Figure. No** | **Caption** | **Page No.** |
| 1 | Geometry and notation of the problem. |  |
| 2 |  |  |
|  |  |  |

### List of tables

|  |  |  |
| --- | --- | --- |
| Table | Caption | Page No. |
| **Table 1** | Tablulation of various different cases considered for the study. |  |

### List of User Defined Functions

|  |  |
| --- | --- |
| **Function name** | **Description** |
| ***InitialSettings*** | Performs some initial settings like docking the figure, clear all, etc. and initial command line displays. |
|  |  |
| ***InputChannelFlow*** | Function to procure user input interactively. The function asks for the correct values in case of conflicting user inputs (to some extent). |
|  |  |
| ***CreateChannelFlowGeometry*** | This function creates the problem geometry from the user input and displays the problem sketch. |
|  |  |
| ***MakeGrid*** | This functions uses “meshgrid” command to make the finite difference grid for the problem. |
|  |  |
| ***StreamFunction\_Define*** | This function is used to initiate the stream function variable and also to feed the boundary condition values of *psi* into the variable of stream function. |
|  |  |
| ***IdentifyInletPoints*** | For the given user input of inlet lengtj, this function determines the locations of grid points lying on the inlet and returns their indices. |
|  |  |
| ***AllocateFlagValues*** | This function allocates a flag value to all the grid points lying on or inside the obstruction. A flag value of “0” is assigned to all these points, while “1” is assigned all the outer points. |
|  |  |
| ***CalculatePsi*** | This function calculates the stream function value at each grid point in the flow domain. It relies on the above two functions for its operation. |
| ***CalculateVelocity*** | From the calculated value of stream function, this function calculates the value of “*u”* and “*v”* velocities by making use of potential flow theory. |
|  |  |
| ***AllocateFlagValue1*** | This function is used by the above function. It makes possible for “CalculateVelocity” function calculate the “*u”* and “*v”* velocities just around the circular obstruction. |
|  |  |
| ***PlotResults*** | This function can plot the results obtained in various different formats. The required formats can be controlled by making appropriate modifications from inside the problem. |
|  |  |
| ***SetAxis*** | This function is used by above function to set correct axis limits based on the channel dimensions. |
|  |  |
| ***FreeUpMemory*** | Finally this functions clears up unwanted residual variables. |

### List of Variable and brief description

|  |  |
| --- | --- |
| **Function name** | **Description** |
| ***channel\_Dimension*** | This is a array which contains: [ channel\_length ; channel\_breadth ; inlet\_size ]; |
| ***obstruction\_Dimension*** | This variable is defined a 6 element array:   1. The value 1 stores the the type of obstruction.    1. 1 – circle.    2. 2 – ellipse. *(note: work pending)*    3. 3 – rectangle. *(note: work pending)*    4. 4 – no obstruction. 2. The value 2 stores the distance of the obstruction center from the inlet.    1. If value 1 = 4, value 2 = NaN. 3. The value 3 and 4 stores the dimensions of the obstruction.    1. If value 1 = 1, value 2 = value 3 = diameter.    2. If value 1 = 2, value 2 = major diameter, value 3 = minor diameter.    3. If value 1 = 3, value 2 = length, value 3 = breadth.    4. If value 1 = 4, value 2 = value 3 = NaN. 4. The value 5 stores the inclination of the obstruction about x- axis    1. If value 1 = 1, value 4 = 0.    2. If value 1 = 2, value 4 = alpha.    3. If value 1 = 3, value 4 = alpha.    4. If value 1 = 4, value 4 = NaN. 5. The value 6 stores the offset value of the obstruction about x- axis.    1. If value 1 = 4, value 6 = NaN. |
| ***grid\_values*** | This stores the grid lengths along x and y axes. grid\_values = [xgrid ; ygrid]; |
| ***InletVel*** | Value of the stream function at the inlet. |
| ***ChannelWall*** | This stores the defining co-ordinates of the channel wall. |
| ***obst\_coord*** | This stores the x and y coordinate values points obstruction walls in a two row array. |
| ***X,Y*** | X and y values of all the individual grid points. |
| ***StreamFunction*** | This function stores the stream function values of the flow. |
| ***FM*** | Short form for “FlagMatrix”. Stores values of flag values of individual grid points. Please see function “AllocateFlagValues” in the “User Defined function” table. |
| ***u,v*** | u and v velocities of the stream function. |

### Table of Contents

|  |  |  |
| --- | --- | --- |
|  | Abstract | I |
|  | Nomenclature | II |
|  | Table Contents | III |
|  | List of Figures | IV |
|  | List of Tables | V |
| 1 | Problem Stament | 1 |
| 2 | Objectives and requirements |  |
| 3 | Mathematical Formulation of the problem |  |
| 4 | Coding approach to the problem |  |
| *5* | MATLAB function – Variable list and brief explanation of function. |  |
| *6* | Various configurations considered for analysis |  |
| *7* | Results and discussions |  |
| *8* | Validation of the obtained results |  |
| *9* | Conclusion |  |
| *10* | Scope for future work |  |
| *11* | References |  |

### Problem Statement

It is required to estimate the channel flow across a circular cylinder. The flow enters the cannel at a certain velocity and exits at the end of the channel after flowing across the obstruction. Please refer to figure 1 for geometry and notations of the problem.

|  |
| --- |
|  |
| **Figure1**  Geometry and notation of the problem. |

### Objectives and requirements

* The output of the simulation should show that the flow speed on the right side of the cylinder is slower than that on the left.
* Plot the velocity profiles before and after the cylinder for the u and v velocity components along the y axis.
* To find the flow topology near the corners of the external wall? Is this correct? Justify your answer.
* Validate your solution by comparing against the ideal flow theory of your Aerodynamics and Fluid Mechanics courses.

### Mathematical formulation

The flow is modeled by the Laplace’s equation,

(1)

Where, ‘*Ψ*’ is the flow property such as velocity, pressure etc., called as ***stream function.*** Thenumerical approximation of the above equation can be derived as follows:

Consider the Taylor’s series expansion of the function at four different grid points, (i-1,j), (i+1,j), (i,j-1) and (i,j+1), which are immediately adjacent to the grid point (i,j). These are written down in the following forms.

*(2)*

*(3)*

*(4)*

*(5)*

Letting , and adding equations (2) to (5) and neglecting the terms of order 3 and higher, and solving for , we get

*(6)*

Where, (i, j) is the point under consideration, a, b, c and d are respectively, distances of this point from (i-1, j), (i+1, j), (i, j-1) and (i, j+1), *Ψa, Ψb, Ψc and Ψd,* are the stream function values at the points (i-1, j), (i+1, j), (i, j-1) and (i, j+1) respectively.

Now, consider a grid point *(i,j)* located within four unequally spaced grid point neighbours, (*Ψi-1,j* , *Ψi+1,j* , *Ψi,j-1* and *Ψi,j+1*) as shown in figure 3. The equation for , I this case can be wriitn down in the following form:

|  |
| --- |
|  |
| **Figure 2**  Unequal grid spacing and streamfunction indexing |

*(7)*

Where, , , and are the values of the stream function of grid points at distances a, b, c and d respectively from the central point (i,j) with respect to figure 3. It can be shown that when all the individual distances in the above equation become equal to each other, Eqn. 7 reduces back to Eqn. 6.

The vlocites are calculated as follows:

*(8)*

*(9)*

Now, for a grid having indices (*r , c*), with ‘*r’* ranging from bottom to top and ‘*c’* ranging from left to right, the values to the right hand side of Eqn. 8 and Eqn. 9 can be written using the mid-point finite difference approximation as follows:

*(10)*

*(11)*

Then, the value of resultant velocity is found out as

*(12)*

The magnitude is thus given by the formula,

*(13)*

### Algorithms used and developed for coding

### The MATLAB code is addressed in such a way that the channel dimensions,obstruction dimension and location, unequal grid lengths are made controllable through user input values. Hence, a modular approach to the problem is taken since it has many advantages such as ease of addition of other modules and functionalities, easy reading of the program and aids better clarity.

|  |
| --- |
|  |
| **Figure 3:** Basic flowchart outlining the code architecture |

The above flowchart shows the basic architecture of the code. Initially, a few setting are made such as *clearing any residual variables, closing any open figures, docking the figure to main window and some initial user dislay* in the block “Make any initial settings”. Then, the next block acquires the user inputs. Any error in the user inputs are found out and the user is asked to enter the correct value. In the next stage, some tasks such as making the finite difference grid, defining and assignining the boundary conditions and identifying inlet points and allocating flag values to points lying inside, on and outside the cylindrical obstruction are performed. Later, the values of streamfunction are calculated using the finite difference approximation formula (Eqn. 7). Using the values of stream function, the values of the velocity components are calculated.

|  |
| --- |
|  |
| **Figure 4:** Basic flowchart outlining the code architecture |

Acquiring user inputs: A separate function is created to manage the user input phase of the problem. In the code written for this work, this segment comprises of three major parts. The first part acquires user inputs such as channel dimensions and stores them in the 3-element array variable “*channel\_Dimension*”. Then, the second part acquires user inputs of obstruction size, shape and location and stores them in the 6-element array variable “*obstruction\_Dimension*” (Please refer to the “variable list” provided at the beginning of the report). Some extent of corrective measures are included in this part where, the wrong user inputs are not considered and the user is asked again for the value until permissible correct values are input. The third part acquires the user inputs values such as horizontal and vertical grid lengths (“*xgrid and ygrid*”) and stores the in the 2-element variable array “*grid\_Values*”.

Prepare for calculations: This phase makes preperations such as making of rectrangular finite difference grid and also inculcates the boundary conditions at the walls and the inlet into the “*StreamFunction*” variable. The grid points lying inside and on the boundaries of the obstruction are identified by “distance method” and each one of the points in the flow domain is flagged accordingly with values wither 1(for points outside the obstruction boundary) or 0 (for points on and inside obstruction boundary). The below function segment defines the boundary conditions:

|  |
| --- |
| function [StreamFunction] = StreamFunction\_Define(X,Y,channel\_Dimension,InletVel) StreamFunction = zeros(size(X,1),size(X,2)); ***% initialize the variables***  InletPoints = IdentifyInletPoints(X,Y,channel\_Dimension); ***% function to identify inlet points***  StreamFunction(InletPoints(1,:)) = InletVel; % Define inlet % Define inlet B. Condition  % define top wall B. Constion  % define bottom wall B. Constion |

### The below function identifies the inlet grid points of the channel:

|  |
| --- |
| function InletPoints = IdentifyInletPoints(X,Y,channel\_Dimension) track = 1; ***% start count value***  for count = 1:size(Y,1); ***% along the rows***  Dist00 = sqrt(X(count,1)^2 + Y(count,1)^2); ***% calculate the distance from (0,0)***  if Dist00 < channel\_Dimension(3,1)/2;***% make the decision***  InletPoints(track) = count; ***% update inlet point location matrix***  track = track + 1; ***% increment count value***  end  end |

|  |
| --- |
|  |
| **Figure 5:** Flowchart to assign the flag values. |

Figure 5 shows the flowchart for the used to assign flag values to individual grid points. First the Flag Matrix, which contains the flag values for the entire finite difference grid is pre-assigned to ‘ones’ matrix. Then, a search is made to identify the pointsinside the obstruction by using the general distance formula in 2-D Cartesian co-ordinates. The below function shows the algorithm used to assign identify grid point type and allocate flag values to grid points:

|  |
| --- |
| function [FM] = AllocateFlagValue(X,Y,obstruction\_Dimension) h=obstruction\_Dimension(2);k=obstruction\_Dimension(6); ***% (h,k) = obstr. Center coordinates***  D=obstruction\_Dimension(3); ***% Dia of the circular obstruction***  FM = ones(size(X,1),size(X,2)); ***% initialize***  FM(:,1) = NaN; ***% define flag on left walls.***  FM(:,size(X,2)) = NaN; ***% define flag on outlet.***  FM(1,:) = NaN; ***% define flag on bottom walls.***  FM(size(X,1),:) = NaN; ***% define flag on top walls.***  rLimit = size(X,1)-1;cLimit = size(X,2)-1; ***% specify limits***  for r = 2:rLimit  for c = 2:cLimit  T1=X(r,c)-h;T2=Y(r,c)-k;***% these are ((x1-x2) and (y1-y2)) in the general distance formula***  if ((T1\*T1+T2\*T2)^0.5)<=(D/2); ***% if point is inside or on obstr. boundary***  FM(r,c)=0; ***% Set the flag values to zero.***  end  end  end |

### Calculate the required values: This function applies the finite difference formula for calculatinf the stream function values at required grid points as given in Eqn. (7) and iterates either for a predefined no of times or until the error is negligible. Then, the values of *u* and *v* arecalculated.

### Display the results: This function plots the contour, contour, mesh and 2-D line plots of variables “*StreamFunction*”, “*u*” and “*v*” against the flow geometry.

### Predictions from theory, experiments and tool on VITAL

**Predictions from experiments for Flow around cylinder:**

|  |
| --- |
|  |
| **Figure 5a:** Experimental predictions of flow around cylinder. [**John D Anderson]** |

Figure 5 shows the low Re (= 1.54) flow over a circular cylinder. It can be seen that the flow remains attached almost symmetrical. The fluid considered in this experiment is water and the tracking particles used were alluminium powder on the surface of water. In Stokes flow, there exists near balance of pressure forces and friction forces [**REF: Anderson**].

**Predictions from theory for Flow around cylinder:**

|  |
| --- |
|  |
| **Figure 6:**  Theoretical predictions of flow around cylinder.  **[E Krause.**] |
|  |

The flow around a circular cylinder can be obtained by superposinga parallel flow with a dipole. The equation for the stream function in wich case would be written as follows:

*(14)*

Where, M is the dipole moment given by

*(15)*

Where, R is the cylinder radius.

**Predictions from theory for Flow near a corner:**

|  |
| --- |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\CornerFlow_Theory.jpg |
| **Figure 7:**  Potential flow solution of flow near a corner obtained using MATLAB. |

The flow near the walls can be modeled by considering the presence of a stagnation point at the intersection of the walls. The stream function in this case is given by the relation [***E.KRAUSE***]

*(16)*

The below shows the MATLAB code used to obtain Figure 7.

|  |
| --- |
| x = -0.1:0.01:1;y = -1:0.01:0.1;  [X,Y] = meshgrid(x,y);psi = 2\*X.\*Y;  contourf(X,Y,psi,25);colorbar;hold on  figure;plot([min(x) max(x)],[0 0],'k','LineWidth',3)  plot([0 0],[min(y) max(y)],'k','LineWidth',3)  xlabel('X-axis');ylabel('Y-axis')  title('Stream Function \psi near the corners of walls')  axis square;axis tight;print('-djpeg100', 'CornerFlow\_Theory') |

**Predictions from tool on Vital:**

|  |
| --- |
| C:\Documents and Settings\sunil\My Documents\Downloads\contour.jpg |
| **Figure 8:** Snapshot of the result obtained from the tool on VITAL. |

Figure 8 shows the snapshot of the result obtained from the tool available on Vital. The plot shows the contours of flow around the cylinder. The boundary conditions for the streamfunction on the freestream, top and bottom boundaries are set to 1. It can be seen that the fluid moves smoothly across the cylinder with its velocity slowly turning more and more horizontal as it moves farther downstream.

### Results, Observations and Discussions

The flow around a circular cylinder in a channel is analyzed in this work. A finite difference rectangular grid for the cylinder obstruction in channel flow problem is constructed and the boundary conditions are imparted on the stream fuction. Then, using the finite difference approximation of Laplace’s equation, the solution for streamfunction is iteratively calculated at each grid point. Then, using the values of the streamfunction and writing down the finite difference approximation of the velocity values, u, v and U, the velocities are calculated at the grid points. Later the contour plots of streamfunction are plotted for the flow. Then, the values of u, v, and U are plotted against the channel length at various distances from the x-axis.

The below table (Table. 1) shows the various cases considered in the current study.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Case | Channel length | Channel width | Inlet size | Diameter | Distance from inlet | Offset from x-axis | Xgrid, Ygrid | Inlet  Velocity | *Psi* value on top and bottom walls | | **1** | **6m** | **4m** | **1.0m** | **2m** | **3.0m** | **0.0m** | **0.25m, 0.25m** | **1m**/s | **1** | | 2 | 6m | 4m | 1.0m | 2m | 3.0m | 0.0m | 0.10, 0.10 | 1m/s | 1 | | **3** | **6m** | **4m** | **1.0m** | **2m** | **3.0m** | **0.0m** | **0.20, 0.10** | **1m**/s | **1** | | 4 | 6m | 4m | 1.0m | 2m | 3.0m | 0.5m | 0.10, 0.10 | 1m/s | 1 | | 5 | 6m | 4m | 1.0m | 2m | 3.0m | 0.0m | 0.10, 0.10 | 1m/s | 0 | | 6 | 6m | 4m | 1.0m | **No obstruction** | - | - | 0.20, 0.20 | 1m/s | 0 | |
| **Table 1:** Tablulation of various different cases considered for the study. |

The results obtained for the 8 cases mentioned in Table. 1 are given below.

**Case 1** (xgrid = 0.25 m, ygrid = 0.25 m)

|  |  |
| --- | --- |
| M:\Documents\my matlab codes\channel flow across a cylinder\1a_L6W4I1D3d2O0X0.25Y0.25V1TBpsi1_PSICONTOUR.jpeg | M:\Documents\my matlab codes\channel flow across a cylinder\1b_L6W4I1D3d2O0X0.25Y0.25V1TBpsi1_PSICONTOUR.jpeg |
| **Figure 9a: CASE1** Contour plot of the stream function placed on the finite difference grid. | **Figure 9b: CASE1** Stream function plot at the rear of the cylinder. |
| M:\Documents\my matlab codes\channel flow across a cylinder\1c_L6W4I1D3d2O0X0.25Y0.25V1TBpsi1_LineplotU.jpeg | D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\1d_L6W4I1D3d2O0X0.25Y0.25V1TBpsi1_LineplotU.jpeg |
| **Figure 9c: CASE1** Plot of resultant velocity magnitude (m/s) along the channel length in the upper half of the channel. | **Figure 9d: CASE1** Plot of horizontal velocity component (m/s) along the channel length in the upper half of the channel. |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\1e_L6W4I1D3d2O0X0.25Y0.25V1TBpsi1_LineplotU.jpeg  **Figure 9e: CASE1** Plot of vertical velocity component (m/s) along the channel length in the upper half of the channel. | |

It can be seen from comparision of figures Figures. 9a and 9b with Figures. 5a, 6 and 8, that the flow slightly downstream the cylinder if similar. The difference in the plots is due to the change in presence of solid wall (psi = 0) adjacent to the inlet.

It can be seen from the velocity vector plots in Figure. 9d, that the horizontal component of the velocity decreases in the downstream from the cylinder. This shows that the analysis correctly depics the variation in velocity past the cylinder. A small overlapping region in the beginning represents the higher velocity near the inlet.

From Figure. 9e, it can be seen that the vertical velocity component changes sign after the center of the cylinder, which shows that the flow is trying to follow the cylinder profile, in which case is sloping vertically downwards. It can also be noticed that the the value of the vertical velocity tends to become zero near the far end of the channel which suggests that the flow is tending to become horizontal far downstream the cylinder. This phenomenon can be observed to be happening in Figures 5a, 6 and 8, where the flow tends to become horizontal far downstream the cylinder.

Figure 9c shows the variation in the resultant magnitude of the velocity of fluid flow. It can be seen that the velocity in the constricted region of motion i.e. between the cylinder and the channel walls, increases. This suggests that the flow increases in velocity so as o mainitain the continuity theorem in fluid mechanics, since the available are of flow is smaller.

**Case 2** (xgrid = 0.10 m, ygrid = 0.10 m)

|  |  |
| --- | --- |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\2a_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg |  |
| **Figure 10a: CASE2** Contour plot of the stream function placed on the finite difference grid. | **Figure 10b:** **CASE2** Stream function plot at the rear of the cylinder. |
| **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\2c_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg** | **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\2d_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg** |
| **Figure 10c: CASE2** Plot of resultant velocity magnitude along the channel length in the upper half of the channel. | **Figure 10d: CASE2** Plot of horizontal velocity component along the channel length in the upper half of the channel. |
| **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\2e_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg**  **Figure 10e: CASE2** Plot of vertical velocity component along the channel length in the upper half of the channel. | |

This case shows the effect of decreasing the grid spacing on the problem. Figure 10b shows an overlaid line on the streamfunction contour plot, which shows that the flow tends to become horizontal after passing across the cylinder.

From the comparision of Figure 9c (course grid) and Figure 10c (fine grid), two major differences can be seen. One is that, the finer grid places more number of poibts near the inlets and hence removes some part of approximation errors. As a result of this, streamfunction values propagates better during the iterations. The second is that the maximum value of U near the cylinder has increased from 1.15 m/s (inviscid flow !) to 1.4 m/s. Thus the change in the grid lengths has a significant impact on the values of the velocities ths computed.

Even here, it can be seen that the vertical velocity magnitude (Figure 10e) changes sign after passing across the cylinder and then tends towards zero as the fl9ow tries to become horizontal downstream.

**Case 3** (xgrid = 0.20 , ygrid = 0.10)

|  |  |
| --- | --- |
| **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\3a_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg** |  |
| **Figure 11a: CASE3** Contour plot of the stream function placed on the finite difference grid. | **Figure 11b: CASE3** Stream function plot at the rear of the cylinder. |
| **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\3c_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg** | **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\3d_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg** |
| **Figure 11c: CASE3** Plot of resultant velocity magnitude along the channel length in the upper half of the channel. | **Figure 11d: CASE3** Plot of horizontal velocity component along the channel length in the upper half of the channel. |
| **D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\3e_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg**  **Figure 11e: CASE3** Plot of vertical velocity component along the channel length in the upper half of the channel. | |

This case shows the effect of using uequal grid spacing for the computation. Form the comaprision between Figures 9(a to e), 10(a to e) and 11(a to e), it can be seen that the resolution of the values along the x-axis is comparatively lesser than that along the y-axis. This tupe of grid can be used in case of problems, where a higher degree of resolution along one particular dimension is required.

**Case 4** (Cylinder offset from x-axis = 0.5)

|  |  |
| --- | --- |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\4a_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg |  |
| **Figure 12a: CASE4** Contour plot of the stream function placed on the finite difference grid. | **Figure 12b: CASE4** Stream function plot at the rear of the cylinder. |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\4c_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg | D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\4d_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg |
| **Figure 12c: CASE4** Plot of resultant velocity magnitude along the channel length in the upper half of the channel. | **Figure 12d: CASE4** Plot of horizontal velocity component along the channel length in the upper half of the channel. |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\4e_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg  **Figure 12e:** **CASE4** Plot of vertical velocity component along the channel length in the upper half of the channel. | |

This case shows the effect of cylindrical obstructions which are offset from the x-axis on the flow in the channel.

**Case 5** (Ψ on top and bottom walls = 0)

|  |  |
| --- | --- |
| Psi | Psi |
| **Figure 13a: CASE5** Contour plot of the stream function placed on the finite difference grid. | **Figure 13b: CASE5** Stream function plot at the rear of the cylinder. |
|  | D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\CornerFlow_Theory.jpg |
| **Figure 13c: CASE5** Stream function plot near the corner. | **Figure 13d: CASE5** Theoreticalprediction of flow near the corner. Please refer Eqn. 16 and Figure 7. |

**Case 6** (No obstructions)

|  |  |  |
| --- | --- | --- |
| D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\6a_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg | | D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\6b_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1_contour.jpeg |
| **Figure 14a: CASE6** Contour plot of the stream function placed on the finite difference grid. | | **Figure 14b: CASE6** Stream function plot at the rear of the channel. |
|  | | D:\Simulation in Aerospace Engineering\Assessment no 2\channel flow across a cylinder\CornerFlow_Theory.jpg |
| **Figure 14c: CASE6** Stream function plot near the corner. | **Figure 14d:** **CASE6** Theoreticalprediction of flow near the corner. Please refer Eqn. 16 and Figure 7. | |

**Fundamentals of Aerodynamics.. John D Anderson. Mc. Graw Hill publications. ISBN: 007-125408-0**].

[**REF: Fluid Mechanics with problems and solutions. E Krause. Springer. ISBN. 3-540-22981-7**].

### Appendix A: MATLAB code

---------------------------------------------------------------------------------------------------------------------------------------

**InitialSettings**

[channel\_Dimension,obstruction\_Dimension,grid\_values,InletVel] = **INPUT\_ChannelFlow**();

[ChannelWall,obst\_coord] = **CreateChannelFLowGeometry**(channel\_Dimension,obstruction\_Dimension);

[X,Y] = **MakeGrid**(channel\_Dimension,grid\_values);

[StreamFunction] = **StreamFunction\_Define**(X,Y,channel\_Dimension,InletVel);

[FM] = **AllocateFlagValue**(X,Y,obstruction\_Dimension);

[StreamFunction] = **CalculatePsi**(X,Y,StreamFunction,FM,obstruction\_Dimension);

**PLotResults**(X,Y,StreamFunction,channel\_Dimension,obstruction\_Dimension,obst\_coord)

---------------------------------------------------------------------------------------------------------------------------------------

### function [] = InitialSettings()

disp('Welcome to UOL, Dept. of Engg :). Course: SEA')

disp('Channel Flow past an obstruction')

disp('Coded by Sunil Anandatheertha')

disp('Please hit "ENTER" to accept any defaults in this program')

clear all % Clear residual variables

close all % Close all open figures

set(0,'DefaultFigureWindowStyle','docked') % Keep the figure window docked to main window

---------------------------------------------------------------------------------------------------------------------------------------

### function [channel\_Dimension,obstruction\_Dimension,grid\_values,InletVel]=INPUT\_ChannelFlow()

ChW=input('Channel Width>>');ChL=input('Channel Length>>');ISz=input('Inlet Size>>');

channel\_Dimension = [ChL ; ChW ; ISz];disp('Please input :-')

disp('1 for channel flow across cylinder.');disp('2 for channel flow across rectangle.')

disp('3 for unobstructed channel flow.')

for count = 1:1000

value = input('Your input please (def. = 1) >> ');

if isempty(value); value = 1; default = 1; else default = 0; end

if default == 1; break; else

if value == 1 || value == 2 || value == 3; break

else; disp('Please input the correct values "or" use default');

end

end

end;Obstruction = value;clear value

if Obstruction == 1

disp('You chose Cyl.as the obst.'); disp('Enter 1 for circ. cyl. & 2 for ellip cyl.')

disp('Circ. type is default');CylType = input('Your choice of cyl. (def. = 1) >> ');

if isempty(CylType); CylType = 1; end

if CylType == 1

disp('You chose Circ. Cyl. as the obst. ....'); CylDia = input('Cyl. dia. >> ');

if CylDia > (3/4)\*ChW; for count = 1:10000

fprintf('Cyl.Dia must be (<=%d)',(3/4)\*ChW);CylDia = input('Cyl.dia.>>');

if CylDia <= (3/4)\*ChW; break; end; end

end

CylDist = input('Cyl. Dist. from x = 0 >>');

if CylDist <= ISz || CylDist >= (ChL - CylDia); for count = 1:10000

fprintf('Cyl. Dist. must be ( > %d ) & ( < %d ) \n',ISz,ChL - CylDia)

CylDist = input('Cyl. Dist. from x = 0 >>');

if CylDist > ISz && CylDist < (ChL - CylDia); break; end; end

end

fprintf('Cyl. Off. must be ( < %d)\n',1.25\*(ChW - CylDia)/2)

CylOff = input('Your value for Cyl. Off. (def. = 0) >> '); default = 0;

if isempty(CylOff); CylOff = 0; default = 1; end

if default == 0; if CylOff >= 1.25\*(ChW - CylDia)/2;for count = 1:10000

fprintf('Cyl. Off. must be ( < %d)\n',1.25\*(ChW - CylDia)/2)

CylOff = input('Your value for Cyl. Off. (def. = 0) >> ');

if isempty(CylOff); CylOff = 0; end

if CylOff < 1.25\*(ChW - CylDia)/2; break; end; end; end

end; obstruction\_Dimension = [1 ; CylDist ; CylDia ; CylDia ; 0 ; CylOff];

clear CylDia CylOff default count;

elseif CylType == 2

disp('You chose Ellip.Cyl.as the obst.');CylDiaMajor=input('Cyl. maj. dia. >> ');

if CylDiaMajor == 0 || CylDiaMajor > (3/4)\*ChW

for count = 1:10000

fprintf('Cyl. Mj. Dia must be ( > 0 and <= %d)',(3/4)\*ChW )

CylDiaMajor = input('Cyl. dia. >> ');

if CylDiaMajor > 0 && CylDiaMajor <= (3/4)\*ChW; break; end

end

end

CylDiaMinor = input('Cyl. min. dia. (def.: r\_maj = r\_min) >> '); default = 0;

if isempty(CylDiaMinor); CylDiaMinor = CylDiaMajor; default = 1; end

if default == 0; if CylDiaMinor == 0 || CylDiaMinor < 0

for count = 1:10000

disp('Cyl. Min. Dia cannot be zero and -ve ..')

CylDiaMinor = input('Cyl. min. dia. (def.: r\_maj = r\_min) >> ');

if isempty(CylDiaMinor); CylDiaMinor = CylDiaMajor; end

if CylDiaMinor <= CylDiaMajor && CylDiaMinor > 0; break; end

end

elseif CylDiaMinor > CylDiaMajor

for count = 1:10000; disp('Min. dia. must be > 0 and < maj. dia.')

CylDiaMinor = input('Cyl. min. dia. (def.: r\_maj = r\_min) >> ');

if isempty(CylDiaMinor); CylDiaMinor = CylDiaMajor; end

if CylDiaMinor <= CylDiaMajor && CylDiaMinor > 0; break; end

end; end

end; clear default; CylDist = input('Cyl. Dist. from x = 0 >>');

if CylDist<=(ISz+CylDiaMajor/2-CylDiaMinor/2)||CylDist>=(ChL-CylDiaMajor)

for count = 1:10000

fprintf('Cyl.Dist.must be (>%2.4f)&(<%2.4f)\n',ISz+CylDiaMajor/2

- CylDiaMinor/2,ChL - CylDiaMajor)

CylDist = input('Cyl. Dist. from x = 0 >>');

if CylDist>(ISz+CylDiaMajor/2-CylDiaMinor/2)&&CylDist<(ChL-CylDiaMajor)

break; end

end

end

CylAngle=input('Cyl.inclination about horz., in deg.(def.=0)>>');default = 0;

if isempty(CylAngle); CylAngle = 0; default = 1; end

if default == 0

if CylAngle > 360; CylAngle = +mod(CylAngle,360);

elseif CylAngle < -360; CylAngle = -mod(sqrt(CylAngle^2),360);

end

end; clear default

CylOff = input('Cyl. Offset from Ch. axis (def. = 0) >> '); default = 0;

if isempty(CylOff); CylOff = 0; default = 0; end

if default == 0

if CylOff >= 0.75\*(ChW - CylDiaMajor)/2; for count = 1:10000

fprintf('Cyl. Off. must be ( < %2.4f)\n',0.75\*(ChW - CylDiaMajor)/2)

CylOff = input('Your value for Cyl. Off. (def. = 0) >> ');

if isempty(CylOff); CylOff = 0; end

if CylOff < 0.75\*(ChW - CylDiaMajor)/2; break; end; end

end

end

obstruction\_Dimension=[2;CylDist;CylDiaMajor;CylDiaMinor;CylAngle;CylOff];

clear CylDiaMajor CylDiaMinor CylAngle CylOff CylDist default

end; clear CylType

elseif Obstruction == 2

disp('You chose Rect. as the obstr. ....')

RecDist=input('Distance from inlet >> ');

if isempty(RecDist);RecDist=channel\_Dimension(1,1)/2;end

RecBreadth=input('Rect. Breadth >> ');

if isempty(RecBreadth);RecBreadth=channel\_Dimension(2,1)/2;end

RecLength=input('Rect. Length >> ');

if isempty(RecLength);RecLength = RecBreadth; end

RecAngle=input('Rect. inclination with horz. in deg. (def. = 0) >> ');

if isempty(RecAngle);RecAngle = 0;end

RecOff=input('Rect. Offset from Channel axis (def. = 0) >> ');

if isempty(RecOff); RecOff = 0; end

obstruction\_Dimension = [3 ; RecDist ; RecBreadth ; RecLength ; RecAngle ; RecOff];

clear RecBreadth RecLength RecAngle RecOff

elseif Obstruction == 3;obstruction\_Dimension=[4;NaN;NaN;NaN;NaN;NaN];end

Xgrid = input('Grid Spacing along X-axis >>');

fprintf('Maximum Y grid spacing allowed is %2.4f \n',ISz/2)

Ygrid = input('Grid Spacing along Y-axis >>');grid\_values = [Xgrid Ygrid];

InletVel = input('Please input the inlet velocity >>');

---------------------------------------------------------------------------------------------------------------------------------------

### function [ChannelWall,obst\_coord] = CreateChannelFLowGeometry(channel\_Dimension,obstruction\_Dimension)

ChannelWall = [0 +channel\_Dimension(3,1)/2;

0 +channel\_Dimension(2,1)/2;

channel\_Dimension(1,1) +channel\_Dimension(2,1)/2;

0 -channel\_Dimension(3,1)/2;

0 -channel\_Dimension(2,1)/2;

channel\_Dimension(1,1) -channel\_Dimension(2,1)/2];obst\_coord = 0;

hold on,box on,axis square

plot(ChannelWall(1:3,1),ChannelWall(1:3,2),'r','LineWidth',3)

plot(ChannelWall(1:3,1),ChannelWall(1:3,2),'ko','MarkerSize',8,'MarkerFaceColor',[0 0 0])

plot(ChannelWall(4:6,1),ChannelWall(4:6,2),'r','LineWidth',3)

plot(ChannelWall(4:6,1),ChannelWall(4:6,2),'ko','MarkerSize',8,'MarkerFaceColor',[0 0 0])

if obstruction\_Dimension(1,1) == 1

h = obstruction\_Dimension(2,1);k = obstruction\_Dimension(6,1);

X = h + (obstruction\_Dimension(3,1)/2)\*cosd(0:1:360);

Y = k + (obstruction\_Dimension(3,1)/2)\*sind(0:1:360);obst\_coord = [X ; Y];

plot(X,Y,'k','LineWidth',3);fill(X,Y,'g');pause(0.001);axis equal

elseif obstruction\_Dimension(1,1) == 2

X = (obstruction\_Dimension(2,1)/2)\*cosd(0:1:360);

Y = (obstruction\_Dimension(3,1)/2)\*sind(0:1:360);XY = zeros(numel(X),2);

for c = 1 : numel(X)

XY(c,:) = [X(c) Y(c)]\*[+cosd(obstruction\_Dimension(5,1)) sind(obstruction\_Dimension(5,1));

-sind(obstruction\_Dimension(5,1)) cosd(obstruction\_Dimension(5,1))];

end

X = XY(:,1) + obstruction\_Dimension(2,1);Y = XY(:,2) + obstruction\_Dimension(6,1);

obst\_coord = [X ; Y];clear XY;

plot(X,Y,'k','LineWidth',3);fill(X,Y,'g');pause(0.001);axis equal

elseif obstruction\_Dimension(1,1) == 4

end

---------------------------------------------------------------------------------------------------------------------------------------

### function [X,Y] = MakeGrid(channel\_Dimension,grid\_values)

x = 0:grid\_values(1):channel\_Dimension(1,1);

y = (0:grid\_values(2):channel\_Dimension(2,1)) - channel\_Dimension(2,1)/2;

[X,Y] = meshgrid(x,y);

---------------------------------------------------------------------------------------------------------------------------------------

### function [StreamFunction] = StreamFunction\_Define(X,Y,channel\_Dimension,InletVel)

StreamFunction = zeros(size(X,1),size(X,2));

InletPoints = IdentifyInletPoints(X,Y,channel\_Dimension);

StreamFunction(InletPoints(1,:)) = InletVel;

---------------------------------------------------------------------------------------------------------------------------------------

### function InletPoints = IdentifyInletPoints(X,Y,channel\_Dimension)

track = 1; for count = 1:size(Y,1); Dist00 = sqrt(X(count,1)^2 + Y(count,1)^2);

if Dist00 < channel\_Dimension(3,1)/2;

InletPoints(track) = count; track = track + 1;

end

end

---------------------------------------------------------------------------------------------------------------------------------------

### function [FM] = AllocateFlagValue(X,Y,obstruction\_Dimension)

h=obstruction\_Dimension(2);k=obstruction\_Dimension(6);D=obstruction\_Dimension(3);

FM = ones(size(X,1),size(X,2));FM(:,1) = NaN;FM(:,size(X,2)) = NaN;FM(1,:) = NaN;

FM(size(X,1),:) = NaN;rLimit = size(X,1)-1;cLimit = size(X,2)-1;

for r = 2:rLimit

for c = 2:cLimit

T1 = X(r,c)-h; T2 = Y(r,c)-k;if ((T1\*T1+T2\*T2)^0.5)<=(D/2);FM(r,c)=0;end

end

end

---------------------------------------------------------------------------------------------------------------------------------------

### function [StreamFunction] = CalculatePsi(X,Y,StreamFunction,FM,obstruction\_Dimension)

rLimit = size(X,1)-1;cLimit = size(X,2)-1;

for count = 1:10000

for c = 2:cLimit

for r = 2:rLimit

if obstruction\_Dimension(1) == 1

if FM(r,c) == 1

a=abs(X(r,c) - X(r,c-1)); b = abs(X(r,c+1) - X(r,c));

cc=abs(Y(r,c) - Y(r-1,c)); d = abs(Y(r+1,c) - Y(r,c));

t1=StreamFunction(r,c-1)/(a\*(a+b));t2=StreamFunction(r,c+1)/(b\*(a+b));

t3=StreamFunction(r-1,c)/(cc\*(cc+d));t4=StreamFunction(r+1,c)/(d\*(cc+d));

t5=1/(a\*b) + 1/(cc\*d);StreamFunction(r,c)=(t1+t2+t3+t4)/t5;

end

elseif obstruction\_Dimension(1) == 4

a = abs(X(r,c) - X(r,c-1));b = abs(X(r,c+1) - X(r,c));

cc= abs(Y(r,c) - Y(r-1,c)); d = abs(Y(r+1,c) - Y(r,c));

t1 = StreamFunction(r,c-1)/(a\*(a+b)); t2 = StreamFunction(r,c+1)/(b\*(a+b));

t3 = StreamFunction(r-1,c)/(cc\*(cc+d)); t4 = StreamFunction(r+1,c)/(d\*(cc+d));

t5 = 1/(a\*b) + 1/(cc\*d); StreamFunction(r,c) = (t1 + t2 + t3 + t4)/t5;

end

end

end

if obstruction\_Dimension(1) == 1

cEnd = cLimit + 1;

for r = 2:rLimit

a = abs(X(r,cEnd) - X(r,cEnd-1)); cc= abs(Y(r,cEnd) - X(r-1,cEnd));

d = abs(Y(r+1,cEnd) - Y(r,cEnd)); t1 = StreamFunction(r,cEnd-1)/(a\*(a+a));

t3 = StreamFunction(r-1,cEnd)/(cc\*(cc+d)); t4 = StreamFunction(r+1,cEnd)/(d\*(cc+d));

t5 = 1/(a\*a) + 1/(cc\*d); StreamFunction(r,cEnd) = ( 2\*t1 + t3 + t4 ) / t5;

end

elseif obstruction\_Dimension(1) == 4

a = abs(X(r,c) - X(r,c-1)); b = abs(X(r,c+1) - X(r,c));

cc= abs(Y(r,c) - Y(r-1,c)); d = abs(Y(r+1,c) - Y(r,c));

t1 = StreamFunction(r,c-1)/(a\*(a+b)); t2 = StreamFunction(r,c+1)/(b\*(a+b));

t3 = StreamFunction(r-1,c)/(cc\*(cc+d)); t4 = StreamFunction(r+1,c)/(d\*(cc+d));

t5 = 1/(a\*b) + 1/(cc\*d); StreamFunction(r,c) = (t1 + t2 + t3 + t4)/t5;

end

end

---------------------------------------------------------------------------------------------------------------------------------------

### function []=PLotResults(X,Y,StreamFunction,channel\_Dimension,obstruction\_Dimension,obst\_coord)

figure;subplot(2,1,1),surf(X,Y,StreamFunction),grid on,box on,axis equal

xlabel('Channel length --- >');ylabel('Channel width --- >')

title('Stream function, \psi'),pause(0.001)

subplot(2,1,2),surf(X,Y,gradient(StreamFunction)),grid on,box on,axis equal

xlabel('Channel length --- >');ylabel('Channel width --- >')

title('Gradient of Stream Function, \nabla\psi'),pause(0.001);figure

mesh(X,Y,StreamFunction),hold on;plot3(X,Y,StreamFunction,'k.','MarkerSize',12)

grid on,box on,axis equal

if obstruction\_Dimension(1,1) == 4

figure; subplot(2,1,1),contourf(X,Y,StreamFunction,100)

xlabel('Channel length --- >');ylabel('Channel width --- >')

title('Stream function, \psi'),pause(0.001),axis equal;SetAxis(channel\_Dimension);

subplot(2,1,2),contourf(X,Y,gradient(StreamFunction),100);xlabel('Channel length>') ylabel('Channel width --- >');title('Gradient of Stream Function, \nabla\psi')

pause(0.001),axis equal; SetAxis(channel\_Dimension);

elseif obstruction\_Dimension(1,1) == 1

figure; subplot(2,1,1),contourf(X,Y,StreamFunction,100),hold on

plot(obst\_coord(1,:),obst\_coord(2,:),'k-','LineWidth',3),fill(X,Y,'g'),alpha(0.5);

xlabel('Channel length --- >');ylabel('Channel width --- >')

title('Stream function, \psi'),pause(0.001),axis equal; SetAxis(channel\_Dimension);

subplot(2,1,2),contourf(X,Y,gradient(StreamFunction),800),hold on

plot(obst\_coord(1,:),obst\_coord(2,:),'k-','LineWidth',3),fill(X,Y,'g'),alpha(0.5);

xlabel('Channel length --- >');ylabel('Channel width --- >')

title('Gradient of Stream Function, \nabla\psi'),pause(0.001),axis equal

SetAxis(channel\_Dimension);

end

---------------------------------------------------------------------------------------------------------------------------------------

### PlotandPrint.m

**contour(X,Y,StreamFunction,800)**

**axis equal;xlabel('Channel Length');ylabel('Channel Width')**

**title('Value of Stream Function \psi');colorbar**

**print('-djpeg100','6a\_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1\_contour.jpeg')**;**figure**

**start = floor(size(X,2)/1.75);NoOfContours = 100;**

**contour(X(:,start:size(X,2)),Y(:,start:size(X,2)),StreamFunction(:,start:size(X,2)),NoOfContours)**

**axis equal;xlabel('Channel Length');ylabel('Channel Width')**

**title('Value of Stream Function \psi');colorbar**

**print('-djpeg100','6b\_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1\_contour.jpeg')**

**figure;startx = floor(size(X,1)/1.25);starty = 1;**

**endy = floor(size(X,2)/6);NoOfContours = 100;**

**contour(X(startx:size(X,1),1:endy),Y(startx:size(X,1),1:endy),StreamFunction(startx:size(X,1),1:endy),NoOfContours);axis equal;xlabel('Channel Length')**

**ylabel('Channel Width');title('Stream Function \psi around the corner')**

**colorbar;print('-djpeg100','6f\_L6W4I1D3d2O0X0.10Y0.10V1TBpsi1\_contour.jpeg')**

---------------------------------------------------------------------------------------------------------------------------------------

### function SetAxis(channel\_Dimension)

xmin = 0; xmax = channel\_Dimension(1,1);ymin = -channel\_Dimension(2,1)/2;

ymax = channel\_Dimension(2,1)/2;box on,axis equal;axis([xmin xmax ymin ymax])

---------------------------------------------------------------------------------------------------------------------------------------

### function FreeUpMemory()

clear obst\_coord grid\_values h k FM x y

### Appendix B: Sample outputs

|  |  |
| --- | --- |
| **Sample output 1** | **Sample output 2** |
| %%%%%%%%%%%%%%%%%%%%%%%%%%%%  Welcome to UOL, Dept. of Engg :). Course: SEA  Channel Flow past an obstruction  MATLAB code by Sunil Anandatheertha  Please hit "ENTER" to accept any defaults in this program  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Channel Width >> 4  Channel Length >> 6  Inlet Size >> 1  %%%%%%%%%%%%%%%%%%%%%%  Please input :-  1 for channel flow across cylinder.  2 for channel flow across rectangle.  3 for unobstructed channel flow.  Your input please (def. = 1) >>  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  You chose Cyl. as the obst. ....  Enter 1 for circular cyl. and 2 for ellipsoidal cyl.  Circ. type is default  Your choice of cyl. (def. = 1) >>  You chose Circ. Cyl. as the obst. ....  Cyl. dia. >> 2  Cyl. Dist. from x = 0 >>3  Cyl. Off. must be ( < 1.250000e+000)  Your value for Cyl. Off. (def. = 0) >>  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Grid Spacing along X-axis >>.1  Maximum Y grid spacing allowed is 0.5000  Grid Spacing along Y-axis >>.1  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Please input the inlet velocity >>1  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Current iteration no. is >> 250  Current iteration no. is >> 500  Current iteration no. is >> 750  Current iteration no. is >> 1000  %%%%%%%%%%%%%%%%%%%%%% | Welcome to UOL, Dept. of Engg :). Course: SEA  Channel Flow past an obstruction  A MATLAB code by Sunil Anandatheertha  Please hit "ENTER" to accept any defaults in this program  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Channel Width >> 4  Channel Length >> 6  Inlet Size >> 1  %%%%%%%%%%%%%%%%%%%%%%  Please input :-  1 for channel flow across cylinder.  2 for channel flow across rectangle.  3 for unobstructed channel flow.  Your input please (def. = 1) >> 3  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Grid Spacing along X-axis >>.2  Maximum Y grid spacing allowed is 0.5000  Grid Spacing along Y-axis >>.2  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Please input the inlet velocity >>1  %%%%%%%%%%%%%%%%%%%%%%  % %  % %  %%%%%%%%%%%%%%%%%%%%%%  Current iteration no. is >> 250  Current iteration no. is >> 500  Current iteration no. is >> 750  Current iteration no. is >> 1000  %%%%%%%%%%%%%%%%%%%%%% |