Tutorial: Calculation of Flow Uniformity

Introduction

The purpose of this tutorial is to provide guidelines and recommendations for setting up and solving a flow uniformity problem with the help of a user-defined function (UDF).

This tutorial demonstrates how to do the following:

- Modify the UDF for a specific case.
- Use the UDF to calculate the flow uniformity indexes for steady-state flow.

Prerequisites

This tutorial is written with the assumption that you have completed Tutorial 1 from ANSYS FLUENT 12.0 Tutorial Guide, and that you are familiar with the ANSYS FLUENT navigation pane and menu structure. Some steps in the setup and solution procedure will not be shown explicitly.

For more details about UDFs, see ANSYS FLUENT 12.0 UDF Manual.

Problem Description

The problem considers a 3D flow passage. A schematic of the problem is shown in Figure 1. There are two flow inlets and an outlet. The postprocessing planes (post-01, post-02, post-03, and post-04) have already been created with the mesh.

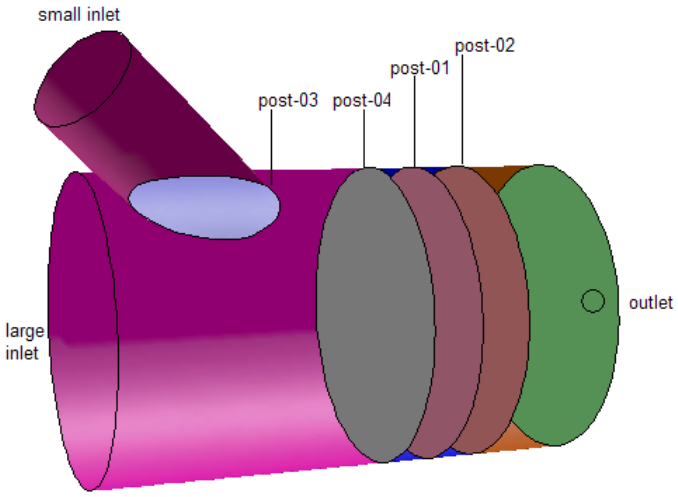


Figure 1: Schematic of the Problem

Setup and Solution

Preparation

1. Copy the files (`flow-uniformity.msh.gz`, `cat-general.c`, and `input.txt`) to the working folder.

2. Use FLUENT Launcher to start the 3D version of ANSYS FLUENT.

   For more information about FLUENT Launcher *see Section* 1.1.2, Starting ANSYS FLUENT Using FLUENT Launcher *in* ANSYS FLUENT *12.0 User's Guide.*

3. Enable Double-Precision in the Options list.

4. Click the UDF Compiler tab and ensure that the Setup Compilation Environment for UDF is enabled.

   The path to the .bat file which is required to compile the UDF will be displayed as soon as you enable Setup Compilation Environment for UDF.

   If the UDF Compiler *tab does not appear in the* FLUENT Launcher *dialog box by default, click the* Show More >> *button to view the additional settings.*

   The Display Options *are enabled by default. Therefore, after you read in the mesh, it will be displayed in the embedded graphics window.*

Step 1: Mesh

1. Read the mesh file (`flow-uniformity.msh.gz`).

   File → Read → Mesh...

   As the mesh file is read, ANSYS FLUENT *will report the progress in the console.*

Step 2: General Settings

1. Retain the default solver settings.

   General

2. Check the mesh (see Figure 2).

   General → Check

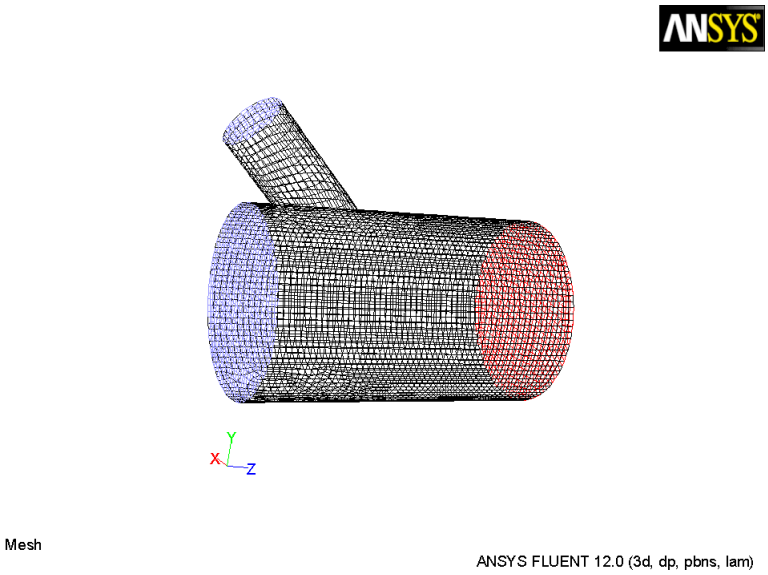


Figure 2: Mesh Display

   ANSYS FLUENT *will perform various checks on the mesh and will report the progress in the console. Make sure the minimum volume reported is a positive number.*

3. Scale the mesh to centimeters.

   General → Scale...

   (a) Select cm from Mesh Was Created In drop-down list.

   (b) Click Scale and close Scale Mesh dialog box.

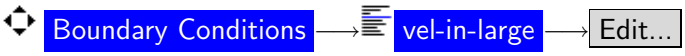
Step 3: Models

Select the k-epsilon turbulence model.

You will use the default fluid properties of air for this problem. Hence, you need not make any changes to the material properties.

Step 4: Boundary Conditions

1. Set the boundary conditions for vel-in-large.

   (a) Select Magnitude, Normal to Boundary from the Velocity Specification Method drop-down list.

   (b) Enter 0.5 m/s for Velocity Magnitude.

   (c) Select Intensity and Hydraulic Diameter from the Specification Method drop-down list.

   (d) Enter 5% for Turbulent Intensity.

   (e) Enter 0.06 m for Hydraulic Diameter.

   (f) Click OK to close the Velocity Inlet dialog box.

2. Set the boundary conditions for velocity-in-small.

   (a) Select Magnitude, Normal to Boundary from the Velocity Specification Method drop-down list.

   (b) Enter 2 m/s for Velocity Magnitude.

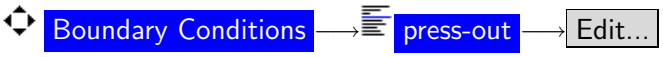
   (c) Select Intensity and Hydraulic Diameter from the Specification Method drop-down list.

   (d) Enter 5% for Turbulent Intensity.

   (e) Enter 0.02 m for Hydraulic Diameter.

   (f) Click OK to close the Velocity Inlet dialog box.

3. Set the boundary conditions for press-out.

   (a) Select Normal to Boundary from the Backflow Direction Specification Method drop-down list.

   (b) Select Intensity and Hydraulic Diameter from the Specification Method drop-down list.

   (c) Enter 5% for Backflow Turbulent Intensity.

   (d) Enter 0.06 m for Backflow Hydraulic Diameter.

   (e) Click OK to close the Pressure Outlet dialog box.

Step 5: Operating Conditions

 Boundary Conditions → Operating Conditions...

1. Retain the default operating conditions.

Step 6: Solution

1. Retain the default solution control parameters.
2. Initialize the flow field from vel-in-large.

    Solution Initialization
3. Start the calculation for 100 iterations.

    Run Calculation

   The solution will converge in approximately 60 iterations.
4. Save the case and data files (`flow-1.cas/dat.gz`).

Step 7: Postprocessing

1. Display filled contours of velocity magnitude.

    Graphics and Animations →  Contours → Set Up...

   (a) Enable Filled in the Options list.

   (b) Select Velocity... and Velocity Magnitude from Contours of drop-down list.

   (c) Select post-01, post-02, post-03, and post-04 from the Surfaces list.

   (d) Enable Draw Mesh in the Options group box.

   The Mesh Display *dialog box will open.*

   i. Disable Edges and enable Faces in the Options group box.

   ii. Deslect all the surfaces from the Surfaces list and click the Outline.

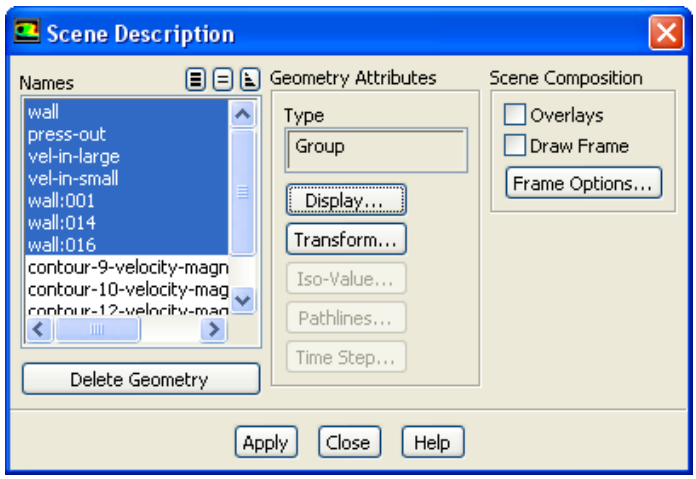
   iii. Click Display and close Mesh Display dialog box.

   (e) Click Display and close Contours dialog box.

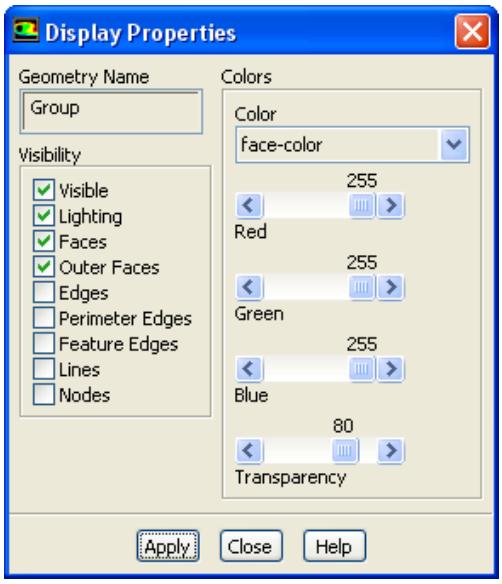
2. Manipulate the display using the Scene Description dialog box.

   Graphics and Animations → Scene...

   (a) Select the surfaces as shown in the Scene Description dialog box.



   (b) Click Display... in the Geometry Attributes group box to open the Display Properties dialog box.



      i. Enable Lighting in the Visibility group box.

      ii. Disable Edges, Lines, and Nodes in the Visibility group box.

      iii. Enable Outer Faces in the Visibility group box.

      iv. Set the sliders for Red, Green, and Blue to 255 in the Colors group box.

      v. Set the slider for Transparency to 80.

      vi. Click Apply and close the Display Properties dialog box (see Figure 3).

   (c) Close Scene Description dialog box.

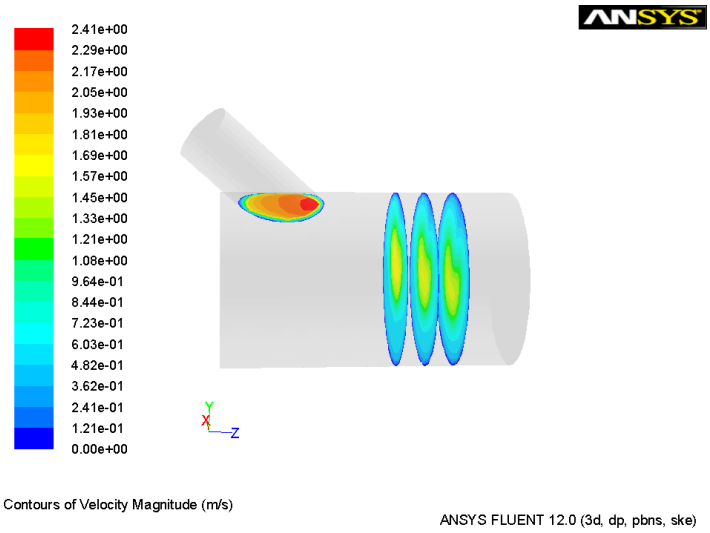


Figure 3: Contours of Velocity Magnitude

3. Display velocity vectors (see Figure 4).

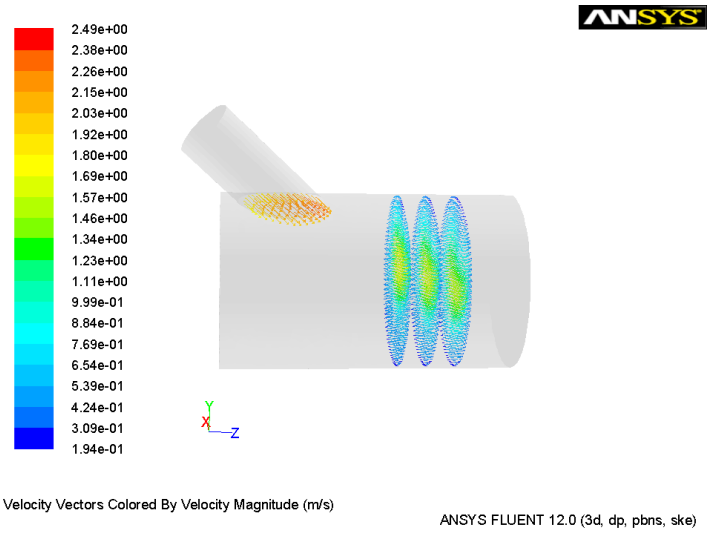


Figure 4: Velocity Vectors Colored by Velocity Magnitude

4. Calculate the flow uniformity.

   (a) Set up the input file (`input.txt`).

   The input file (`input.txt`) has to be processed before executing the UDF. For the details about the input file, see Appendix 1: Details of the Input File.

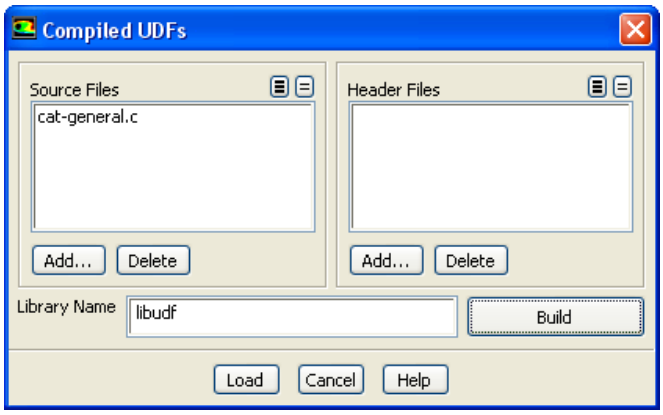
   (b) Set the Number of User-Defined Memory Locations to 4.

   Define ⟶ User-Defined ⟶ Memory...

   (c) Save the case file (`flow.cas.gz`).

   (d) Compile the UDF.

   Define ⟶ User-Defined ⟶ Functions ⟶ Compiled...



   i. Click Add... and select the source file, cat-general.c.

   For details about the UDF file (cat-general.c), refer to Appendix 2: Contents of UDF.

   ii. Click Build to build the library.

   A Warning dialog box opens, asking you to ensure that the UDF source files are in the same folder that contains the case and data files. Click OK.

   iii. Click Load to load the newly created UDF library.

   (e) Execute the UDF.

   Define ⟶ User-Defined ⟶ Execute on Demand...

   i. Select Uniform_steady::libudf from the Execute On Demand drop-down list.

   ii. Click Execute.

   (f) The output on the screen and in the file are shown in Appendix 3: Output of UDF. The UDF reads the input file (`input.txt`) and generates the output file (`output.txt`) automatically in each simulation. The parameters of the output are explained in Appendix 4: General Structure of UDF. If you want to preserve the old results, rename the `output.txt` file. Otherwise the data for the next run will be appended to the old file.

Appendix 1: Details of the Input File

- The first line is the number of postprocessing faces. You can define more than one face zones for flow uniformity calculation.
- Each line after the first line has the parameters of a face-zone. They are defined in the following order:
  - POST_BOUNDARY_ZONE_ID
  - POST_CELL_ZONE_ID_1
  - POST_CELL_ZONE_ID_2
  - L_XX
  - L_YX
  - L_ZX
  - L_XY
  - L_YY
  - L_ZY

  Further detailed description of the parameters is given as follows:

  1. POST_BOUNDARY_ZONE_ID is the ID of the face zone.
  2. POST_CELL_ZONE_ID_1 and POST_CELL_ZONE_ID_2 are the IDs of two adjacent cell zones of the postprocessing face zone. They are the same if the face zone is inside a cell zone.
  3. L_XX, L_YX, L_ZX are the direction cosines of the x-axis of the local coordinate. The UDF uses a local coordinate for each face zone to compute the flow uniformity. The local x-axis will be the major axis, and local y-axis will be the minor axis. You should define the direction of longer length as x-axis, and the other direction in the same plane as y-axis.
  4. L_XY, L_YY, L_ZY are the direction cosines of the y-axis of the local coordinate. The UDF uses a local coordinate for each face zone to compute the flow uniformity. The local x-axis will be the major axis, and local y-axis direction will be the minor axis. You should define the direction of longer length as x-axis, and the other direction in the same plane as y-axis.

- The definition of the direction cosines may affect the eccentricity because it reflects the local flow distribution on the plane. The gamma does not depend on the direction cosines.

Appendix 2: Contents of UDF

The contents of the UDF file are as follows:

```
/************************************************************/
/*                                                          */
/* User-Defined Functions for calculation of flow uniformity  */
/*                                                          */
/************************************************************/
#include "udf.h"

int ncount;
int npost;

/*You need to modify these three lines*/
int POST_BOUNDARY_ZONE_ID[50];
int POST_CELL_ZONE_ID_1[50];
int POST_CELL_ZONE_ID_2[50];

float L_XX[50];
float L_YX[50];
float L_ZX[50];

float L_XY[50];
float L_YY[50];
float L_ZY[50];

float L_XZ[50];
float L_YZ[50];
float L_ZZ[50];


static void read_input()
{
int i=0;
FILE *fpin;

if((fpin=fopen("input.txt","r"))==NULL)
{
printf("Input file does not exist!");
}
fscanf(fpin, "%d",&npost);
for(i=0;i<npost;i++)
{
fscanf(fpin,"%d %d %d %f %f %f %f %f %f", &POST_BOUNDARY_ZONE_ID[i], &POST_CELL_ZONE_ID_1[i],
&POST_CELL_ZONE_ID_2[i], &L_XX[i], &L_YX[i], &L_ZX[i], &L_XY[i], &L_YY[i], &L_ZY[i]);

L_XZ[i]=L_YX[i]*L_ZY[i]-L_ZX[i]*L_YY[i];
L_YZ[i]=L_ZX[i]*L_XY[i]-L_XX[i]*L_ZY[i];
L_ZZ[i]=L_XX[i]*L_YY[i]-L_XY[i]*L_YX[i];
}
fclose(fpin);
}

static void init_udm()
{

   Thread *tc;
   cell_t c;

   Domain *domain;
domain = Get_Domain(1);
thread_loop_c(tc,domain)
```

```
     {
       begin_c_loop(c,tc)

{
   C_UDMI(c,tc,0)=0;
   C_UDMI(c,tc,1)=0;
   C_UDMI(c,tc,2)=0;
   C_UDMI(c,tc,3)=0;
}
       end_c_loop(c,tc)
}
   ncount=0;
}



static void cat_post(int flag)
{

   float xf[ND_ND], x_vmax_global[ND_ND], x_vmax, y_vmax;
   float x_face_local, y_face_local;
   float xn[ND_ND],x_node_local,y_node_local;
   float x_min, x_max, x_mid, y_min, y_max, y_mid;
   float L_major, L_minor;
   float eccent_x, eccent_y, eccent;
   float vmag, vmax, vavg, vratio, vel_space;
   float A_face_vec[ND_ND], A_face_mag, A_tot;
   float a_35, v_35, unif_35, a_65, v_65, unif_65, gamma;
   int k;

   face_t f, f_vmax;
   cell_t c0, c1;
   Thread *tf, *tc0, *tc1;

FILE *fp;

   Domain *domain;
domain = Get_Domain(1);


#define BIGNUM 1e20

if((fp=fopen("output.txt", "r"))==NULL)
{
fp=fopen("output.txt","w+");
fprintf(fp, "post-face    eccent     gamma     unif_35     unif_65     vratio     vmax \n");
fclose(fp);
}

fp=fopen("output.txt", "a");

/*  find local y,z min/max extents based on node values */


   x_min=BIGNUM;
   y_min=BIGNUM;
   x_max=-BIGNUM;
   y_max=-BIGNUM;
   tf=Lookup_Thread(domain, POST_BOUNDARY_ZONE_ID[flag]);
```

```
   begin_f_loop(f,tf)
   {
     f_node_loop(f,tf,k)

     {
       xn[0]=NODE_X(F_NODE(f,tf,k));
       xn[1]=NODE_Y(F_NODE(f,tf,k));
      xn[2]=NODE_Z(F_NODE(f,tf,k));
      x_node_local = NVD_DOT(xn,L_XX[flag],L_YX[flag],L_ZX[flag]);
      y_node_local = NVD_DOT(xn,L_XY[flag],L_YY[flag],L_ZY[flag]);
      x_min=MIN(x_node_local,x_min);
      x_max=MAX(x_node_local,x_max);
      y_min=MIN(y_node_local,y_min);
      y_max=MAX(y_node_local,y_max);
}
   }
   end_f_loop(f,tf)
   x_mid=0.5*(x_min + x_max);
   y_mid=0.5*(y_min + y_max);
   L_major = x_max - x_min;
   L_minor = y_max - y_min;


Message("============================================================\n");
Message("post-%d\n",flag);
Message("============================================================\n");
   Message("Geometry information:\n");
   Message("(x_min, x_mid, x_max) = (%f, %f, %f) [m]\n", x_min, x_mid, x_max);
   Message("(y_min, y_mid, y_max) = (%f, %f, %f) [m]\n", y_min, y_mid, y_max);
   Message("(L_major, L_minor) = (%f, %f) [m]\n\n", L_major, L_minor);

/* velocity index (eccentricity) computation */
   vmax=0.0;
   vavg=0.0;
   A_tot=0.0;
   tf=Lookup_Thread(domain, POST_BOUNDARY_ZONE_ID[flag]);
tc0=Lookup_Thread(domain, POST_CELL_ZONE_ID_1[flag]);
   tc1=Lookup_Thread(domain, POST_CELL_ZONE_ID_2[flag]);
   begin_f_loop(f,tf)
   {
     F_AREA(A_face_vec,f,tf);
     A_face_mag=NV_MAG(A_face_vec);
     A_tot+=A_face_mag;
     c0=F_C0(f,tf);
     c1=F_C1(f,tf);
vmag=0.5*(C_UDMI(c0,tc0,3)+C_UDMI(c1,tc1,3));
     vavg+=vmag*A_face_mag;
     if(vmag > vmax)
     {
       vmax=vmag;
       f_vmax=f;
     }
   }

   end_f_loop(f,tf)
   vavg/=A_tot;
   vratio=vmax/vavg;
   F_CENTROID(x_vmax_global,f_vmax,tf);

   x_vmax= NVD_DOT(x_vmax_global,L_XX[flag],L_YX[flag],L_ZX[flag]);
   y_vmax= NVD_DOT(x_vmax_global,L_XY[flag],L_YY[flag],L_ZY[flag]);

   eccent_x=2.0*(x_vmax-x_mid)/L_major;
   eccent_y=2.0*(y_vmax-y_mid)/L_minor;
eccent=sqrt(eccent_x*eccent_x+eccent_y*eccent_y);
```

```
   Message("Eccentricity information:\n");
   Message("(x_vmax, y_vmax) = (%f, %f) [m]\n", x_vmax, y_vmax);
   Message("eccentricity = (%f, %f) ->  %f \n\n",eccent_x, eccent_y, eccent);
   Message("vmax = %f m/s occurs at face %d\n", vmax, f_vmax);

   Message("vratio = %f\n", vratio);

/* uniformity, gamma calculations */
   v_35=0.35*vmax;
   v_65=0.65*vmax;
   a_35=0.0;
   a_65=0.0;
   gamma=0.0;
   begin_f_loop(f,tf)
   {
     F_AREA(A_face_vec,f,tf);
     A_face_mag=NV_MAG(A_face_vec);
     c0=F_C0(f,tf);
     c1=F_C1(f,tf);
      vmag=0.5*(C_UDMI(c0,tc0,3)+C_UDMI(c1,tc1,3));
     if(vmag >= v_35) a_35+=A_face_mag;
     if(vmag >= v_65) a_65+=A_face_mag;
     gamma+=ABS(vmag-vavg)*A_face_mag;
   }
   end_f_loop(f,tf)
   gamma=1.0-gamma/(2.*vavg*A_tot);

   unif_35=a_35/A_tot*100;
   unif_65=a_65/A_tot*100;

   Message("uniformity index = %f/%f\n",unif_65,unif_35);
   Message("gamma = %f\n",gamma);


fprintf(fp, "post%d    %9.3f  %9.3f  %9.3f   %9.3f  %9.3f  %9.3f \n", flag, eccent, gamma,
 unif_35, unif_65,  vratio, vmax);
fclose(fp);

}
static void set_udm(int flag)

{
/* NOTE: THIS ROUTINE REQUIRES ALLOCATION OF */
/* 4 USER DEFINED MEMORY LOCATIONS IN FLUENT */

#define X_PRIME 0
#define Y_PRIME 1
#define Z_PRIME 2
#define V_MAG 3

   float xc[ND_ND];
   Thread *tc1,*tc2;
   cell_t c;

   Domain *domain;
domain = Get_Domain(1);


   tc1=Lookup_Thread(domain, POST_CELL_ZONE_ID_1[flag]);
   tc2=Lookup_Thread(domain, POST_CELL_ZONE_ID_2[flag]);
```

```
begin_c_loop(c,tc1)
   {
     C_CENTROID(xc,c,tc1);
     C_UDMI(c,tc1,X_PRIME)=NVD_DOT(xc,L_XX[flag],L_YX[flag],L_ZX[flag]);
     C_UDMI(c,tc1,Y_PRIME)=NVD_DOT(xc,L_XY[flag],L_YY[flag],L_ZY[flag]);
     C_UDMI(c,tc1,Z_PRIME)=NVD_DOT(xc,L_XZ[flag],L_YZ[flag],L_ZZ[flag]);
     C_UDMI(c,tc1,V_MAG)=sqrt(C_VMAG2(c,tc1));
   }
   end_c_loop(c,tc1)

   if(POST_CELL_ZONE_ID_1[flag] != POST_CELL_ZONE_ID_2[flag])
   {
     begin_c_loop(c,tc2)
     {
     C_CENTROID(xc,c,tc2);
     C_UDMI(c,tc2,X_PRIME)=NVD_DOT(xc,L_XX[flag],L_YX[flag],L_ZX[flag]);
     C_UDMI(c,tc2,Y_PRIME)=NVD_DOT(xc,L_XY[flag],L_YY[flag],L_ZY[flag]);
     C_UDMI(c,tc2,Z_PRIME)=NVD_DOT(xc,L_XZ[flag],L_YZ[flag],L_ZZ[flag]);
     C_UDMI(c,tc2,V_MAG)=sqrt(C_VMAG2(c,tc2));
     }
   end_c_loop(c,tc2)
   }
}


DEFINE_ON_DEMAND(Uniform_steady)
{
int i=0;

read_input();

init_udm();
for(i=0; i<npost;i++)
{
set_udm(i);
cat_post(i);
}
}
```

Notes:

- The postprocessing face must be a real face zone and should be created in preprocessing.
- Only Uniform-steady can be used as a Execute On Demand function.

  You can use the ***EXECUTE_ON_DEMAND*** *macro for calculations which do not participate in the solution iterations, such as postprocessing.*

Appendix 3: Output of UDF

Output on the screen:

```
================================================================
post-0
================================================================
Geometry information:
(x _min, x _mid, x _max) = (-0.030000, 0.000000, 0.030000) [m]
(y _min, y _mid, y _max) = (-0.030000, 0.000000, 0.030000) [m]
(L _major, L _minor) = (0.060000, 0.060000) [m]

Eccentricity information:
(x _vmax, y _vmax) = (0.002929, 0.001147) [m]
eccentricity = (0.097642, 0.038245) ->  0.104865

vmax = 1.576765 m/s occurs at face 306
vratio = 2.062665
uniformity index = 23.614954/66.099402
gamma = 0.807598
================================================================
post-1
================================================================
Geometry information:
(x _min, x _mid, x _max) = (-0.030000, 0.000000, 0.030000) [m]
(y _min, y _mid, y _max) = (-0.030000, 0.000000, 0.030000) [m]
(L _major, L _minor) = (0.060000, 0.060000) [m]

Eccentricity information:
(x _vmax, y _vmax) = (0.004870, -0.003402) [m]
eccentricity = (0.162350, -0.113398) ->  0.198032

vmax = 1.495621 m/s occurs at face 278
vratio = 1.961932
uniformity index = 27.457386/72.582035
gamma = 0.813344
================================================================
post-2
================================================================
Geometry information:
(x _min, x _mid, x _max) = (-0.044035, -0.029980, -0.015925) [m]
(y _min, y _mid, y _max) = (-0.010000, -0.000000, 0.009999) [m]
(L _major, L _minor) = (0.028110, 0.019999) [m]
```

```
Eccentricity information:
(x _vmax, y _vmax) = (-0.018510, -0.001128) [m]
eccentricity = (0.816052, -0.112719) ->  0.823800

vmax = 2.378441 m/s occurs at face 53
vratio = 1.149897
uniformity index = 99.176368/100.000000
gamma = 0.956343
================================================================
post-3
================================================================
Geometry information:
(x _min, x _mid, x _max) = (-0.030000, 0.000000, 0.030000) [m]
(y _min, y _mid, y _max) = (-0.030000, 0.000000, 0.030000) [m]
(L _major, L _minor) = (0.060000, 0.060000) [m]

Eccentricity information:
(x _vmax, y _vmax) = (0.000981, 0.005763) [m]
eccentricity = (0.032705, 0.192102) ->  0.194866

vmax = 1.677601 m/s occurs at face 334
vratio = 2.189714
uniformity index = 21.332038/55.492872
gamma = 0.802111
```

Output in a file:

post-face	eccent	gamma	unif _35	unif _65	vratio	vmax
post0	0.105	0.807	65.308	23.475	2.071	1.582
post1	0.198	0.812	72.158	27.038	1.969	1.501
post2	0.819	0.956	100.000	98.426	1.146	2.371
post3	0.195	0.802	55.005	21.096	2.197	1.682

Appendix 4: General Structure of UDF

The UDF can be used for uniform steady-state analysis. It is used as a compiled UDF. There are three subroutines, init_udm, set_udm, and cat_post in the UDF. You need to set four user-defined memory locations, in which udm-0, udm-1, and udm-2 store the coordinates, and udm-3 stores velocity. The functions of the subroutines are as follows:

- init_udm: It is used to initialize the user-defined memory.
- set_udm: It computes the local coordinate in the selected cell zones based on the directional cosines provided by users. It stores the local coordinates in udm-0, udm-1, and udm-2, and velocity magnitude of the selected cell zones in udm-3.
- Cat_post: Compute flow parameter for the postprocessing faces.

Calculation of Flow Parameters:

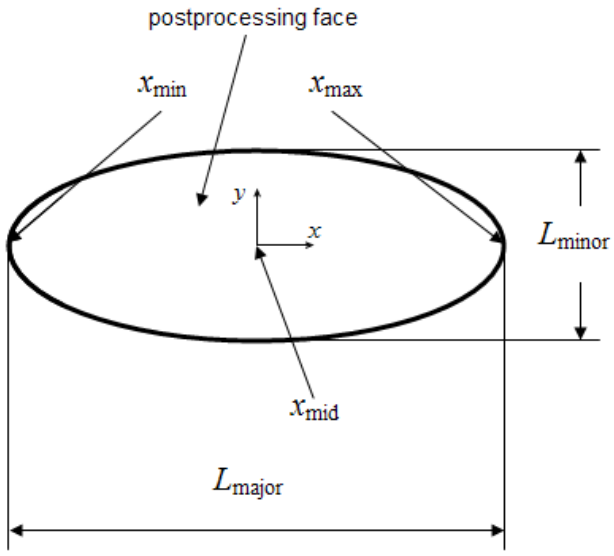


Figure 5: Postprocessing Face

1. Calculation of geometric parameters of the postprocessing face.
   (a) Length of major and minor axes:

$$L\_\text{major} = x\_\text{max} - x\_\text{min} \tag{1}$$

$$L\_\text{minor} = y\_\text{max} - y\_\text{min} \tag{2}$$

   where,

   L_major = length of major axis
   L_minor = length of minor axis
   x_max = maximum x coordinate
   y_max = maximum y coordinate
   x_min = minimum x coordinate
   y_min = minimum y coordinate

   (b) Center of the zone:

$$x\_\text{mid} = \frac{x\_\text{max} - x\_\text{min}}{2} \tag{3}$$

$$y\_\text{mid} = \frac{y\_\text{max} - y\_\text{min}}{2} \tag{4}$$

2. Calculation of velocity index (eccentricity).

   Compute the location (x and y coordinates) of the maximum velocity and the distance between this location and the center of the face. Find the maximum velocity and velocity ratio.

   Eccentricity (ε) is defined as:

$$\varepsilon\_\text{x} = \frac{2\,(x\_\text{vmax} - x\_\text{mid})}{L\_\text{major}} \tag{5}$$

$$\varepsilon\_\text{y} = \frac{2\,(y\_\text{vmax} - y\_\text{mid})}{L\_\text{minor}} \tag{6}$$

$$\varepsilon = \sqrt{\varepsilon\_\text{x}^2 + \varepsilon\_\text{y}^2} \tag{7}$$

   Velocity ratio (ν_ratio) can be defined as the ratio of the maximum velocity to the average velocity.

$$\nu\_\text{ratio} = \frac{\nu\_\text{max}}{\nu\_\text{avg}} \tag{8}$$

3. Calculation of flow uniformity.

   Compute the uniformity index (area based), gamma, and vel_space. The uniformity index is based on these reference velocities: ν_65 (65% of the maximum velocity) and ν_35 (35% of the maximum velocity).

The uniformity is defined as:

$$\varphi\_65 = \frac{A\_65}{100 A\_\text{total}} \tag{9}$$

$$\varphi\_35 = \frac{A\_35}{100 A\_\text{total}} \tag{10}$$

where,

A_65 = the area of which velocity is higher than ν_65
A_35 = the area of which velocity is higher than ν_35

Gamma is defined as,

$$\gamma = 1 - \frac{\sum (|\nu\_\text{i} - \nu\_\text{avg}| A\_\text{i})}{2 \nu\_\text{avg} A\_\text{total}} \tag{11}$$

4. Output parameters.

   (a) On the screen:

   The parameters of the output on the screen can be explained as follows:

   - (x_max, y_max) = Location of the maximum velocity
   - eccentricity = (ε_x, ε_y) $\Rightarrow \varepsilon$
   - vmax = maximum velocity
   - vratio = ν_ratio
   - uniformity index = φ_65/φ_35
   - gamma = γ

   (b) In the output file:

   The parameters in the output file can be explained as follows:

   - time = flow time
   - eccent = ε
   - gamma = γ
   - unif_35 = φ_35
   - unif_65 = φ_65
   - vmax = maximum velocity

Summary

In this tutorial, a flow uniformity problem was set up and solved for a steady state flow with the help of UDF.