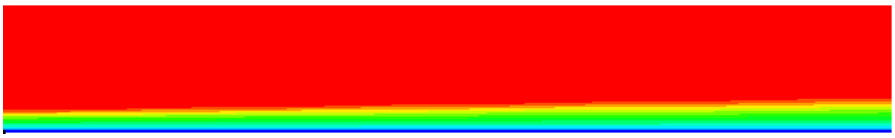
GENERATING A TURBULENT BOUNDARY LAYER INFLOW PROFILE FOR JOE

MICHAEL EMORY

Contents

Date: December 15, 2009.

1. Introduction



Many of the simulations being performed using JOE require turbulent inflow profiles. The particular boundary condition file will need to include different quantities of interest depending on which turbulence model is used (k, ϵ, ω, ν_{SA}, etc). For simulations where the inflow condition is a turbulent boundary layer this document describes a procedure by which you can generate an appropriate boundary condition file for JOE. The method can be explained in a few simple steps:

(1) generate laminar profile using Olaf's code for desired conditions (Mach, T, μ_∞, etc)
(2) build flat plate grid (JOE will read .msh, .cas, etc), making sure same distribution of Y values is maintained for all X values
(3) run JOE using laminar profile and flat plate grid until a converged solution is reached
(4) use provided Matlab script to evaluate and extract desired turbulent boundary layer profile

This essentially starts the flat plate at the beginning of the domain, and then transition occurs to a turbulent boundary layer, after which a turbulent boundary layer will then grow. Based on your desired criteria at the inflow ($\delta_{99\%}$, RE_θ, etc) the profile is cut and a 2-D inflow profile is generated. Sample code is provided for reading and using the generated profile in your joe.cpp file.

2. Generate Laminar Profile

This section will explain how to use Olaf Marxen's laminar boundary layer code to generate an inlet profile for the flat plate simulation. (help from Rene, Simon, or Olaf?)

3. Flat Plate Simulation

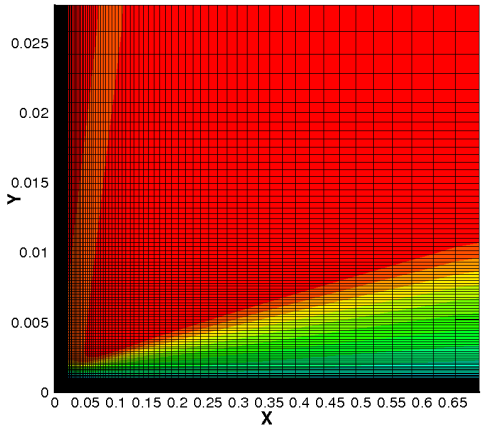


Figure 1. X and Y are not the same scale

3.1. **Generate a Grid.** Using your grid generation technique of choice, a rectangular grid must be generated. Keeping in mind that the laminar BL must transition then grow into the correct size of turbulent BL, the grid must be approx. 3 δ in height and 10 δ in length. It is highly recommended to use the Y coordinates from the mesh of your end-goal simulation as the Y values of the plate. The X values should be reasonable in size (100-200) points along the plate, having a smaller spacing near the beginning of the domain where transition will occur, as in Fig. 1. Do not worry about the details of the size of the mesh, there are commands within JOE to scale the Y and X coordinates if the original grid is not optimal.

NOTE:
make sure all Y value distributions are the same for each X location

3.2. **Running JOE for the Flat Plate.** The joe.cpp file found in this folder can be used to run the flat plate for different turbulence models, the one we will be using as our example is the $k-\omega$ SST of Menter (1994). Looking at line 547 (Fig. 3) you will see that in order to run JOE with the correct turbulence model you must use an argument declared after the execution call, in our case we will run joe using:

```
$ mpirun -np 4 ./joe 2
```

The current file should be set up to read in laminar profile's exported by Olaf's code, with one simple manipulation. The first line of the laminar inlet file must be changed from variable listings to n=# d=#, where n is the total number of points, and d is the number of variables in the profile. The variables typically found in the laminar profile are: X, Y, Z, P, ρ, u, v, and w. These values are read in to JOE in the initial Hook() routine shown in Fig. 4. If you have different variables this portion of code must be edited. We typically assign the following boundary conditions for the flow, which are specified in the Joe.in file:

outlet: the back and top of the domain are set to NEUMANN

sides: the spanwise side walls are set as SYMMETRY planes

inlet: this is set to a HOOK condition, where HOOK refers to the code in boundaryHook()

wall: the bottom surface is set to WALL, with either an adiabatic or isothermal condition

3.3. **Initial Values for Turbulent Quantities.** In the Joe.in file the user must specify initial values for certain turbulence parameters (depending on the turbulence model being used). The methods described at the CFD-online Wiki[1] are currently used in our simulations, it is up to the user to determine their own values based on the simulation being run and available information. The user defined variables of turbulence intensity (I) and viscosity ratio (ν_t/ν_{lam}) must be specified, we are using $I = 0.001$ and $\nu_{ratio} = 1$.

4. Using the Matlab Script

The file extractProfile.m is a Matlab script which reads the data output from the JOE simulation. The file KOMSST_profile.txt is generated by JOE and outputs certain variables for every cell in the domain. Since JOE is an unstructured solver, the data is not written in an ij ordered format. In order to sort and create structured data from this file, the Matlab script sorts first by X location, then Y location to build an ij data structure for each variable. For this reason it is important to have the same Y values for each X value, otherwise the sorting algorithm may not work properly.

Running the script, the user can select what critical value to check in order to cut the profile (a desired RE_θ, δ, etc). The turbulence model must also be selected. The KOMSST_profile.txt is then read in, and various measures of the boundary layer thickness are computed. After data has been generated for each X location, the user is prompted for the cutoff value for the critical variable. If the user specifies a value which is not in the current range of data, the user is shown the range of values and is prompted again. If the value is within the range, the profile is exported in 'word'_turb_inlet.txt where 'word' describes the turbulence model used.

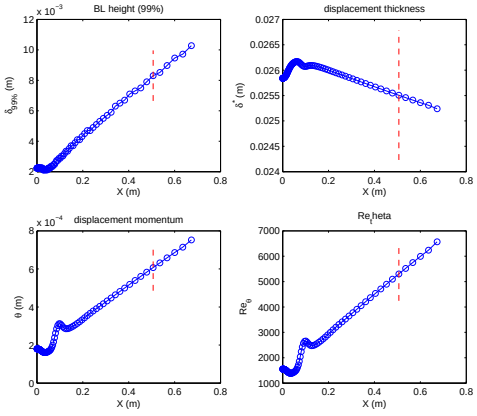


Figure 2. sample graphical output from matlab script

4.1. **Graphs Provided.** It is important to make sure the slice you have extracted is from the fully developed turbulent

[1] http://www.cfd-online.com/Wiki/Turbulence_free-stream_boundary_conditions

boundary layer, and not from the transition region. For this reason there are four graphs exported after selecting a slice, which show you the range of values and your slice location (in dashed red lines) for several measures of BL thickness. If your slice is in the constant slope section of the plot, it means you are in the developed zone, see Fig. 2.

5. Troubleshooting

5.1. **Profile not Correct Size.** This is either a problem with the laminar boundary layer size or length of domain issue. There are flags built in to the Joe.in file which can help in both of these situations. The first is a scaling of the laminar boundary layer, which will allow the user to increase/decrease the size of the laminar inlet profile. This is done using the variable SCALE_BL. One must take caution not to scale the inflow too much, since the boundary layer will be represented by fewer and fewer grid points. If you are scaling the boundary layer below the resolution of the mesh you have, you can scale the Y coordinates of the grid using SCALE_MESH_Y. If the domain is too short and all you want to do is extend the domain so that the BL can grow to the correct size, use the variable SCALE_MESH_X, which allows you to stretch the X locations. The user can customize the mesh modification by editing the transformMesh() routine (Fig. 5) or use the builtin scaling variables discussed above and specifying values for these in Joe.in. Keep in mind that in order to do mesh transformation via Joe.in the flag TRANSFORM_MESH must be present in the file.

5.2. **Matlab Script not reading KOMSST_profile.txt Correctly.** This is most likely due to problems writing the data file when using multiple processors. If running on one processor the file should be generated correctly. As a workaround for multiprocessor simulations, run the simulation again for 0 iterations, using the last restart.out as the restart file 'RESTART=./restart.out', making sure to only run on one processor. This should fix the KOMSST_profile.txt file.

other problems might arise from total number of variables being exported, and the number of header lines specified by the Matlab command importData().

6. Reference Code Chunks

```
switch (run)
    {
    case 0:   joe = new PlateLaminar(inputFileName);    break;
    case 1:   joe = new PlateTurbSA(inputFileName);     break;
    case 2:   joe = new PlateTurbSST(inputFileName);    break;
    case 3:   joe = new PlateTurbWXKOM(inputFileName);    break;
```

Figure 3. arguments to specify turbulence model

```
fscanf(fp, "n=%d\td=%d", &npos, &nval);
    getMem2D(&boundVal1, 0, npos-1, 0, nval-1, "boundVal1");

    // file has values
    // x, y, z,press, rho, u, v, w
    for (int i=0; i<npos; i++)
    {
      for (int v=0; v<nval; v++)
      {
        fscanf(fp, "%lf", &boundVal1[i][v]);
        if (v==1)
      boundVal1[i][v] *= fakt_bl;

    printf("%.6le\t", boundVal1[i][v]);
      }
      printf("\n");
    }
```

Figure 4. initialHook() code which reads laminar inlet profile

```
void transformMeshHook()
 {
   if (!checkParam("TRANSFORM_MESH")) return;
   double scale_mesh_X = getDoubleParam("SCALE_MESH_X", "1.0");
   double scale_mesh_Y = getDoubleParam("SCALE_MESH_Y", "1.0");
   for (int ino = 0; ino < getNno(); ++ino)
   {
       x_no[ino][0] *= scale_mesh_X;
       x_no[ino][1] *= scale_mesh_Y;
   }


   // clearFlag for wall distance to recompute the wallDist when mesh deformed
   DoubleScalar *wallD = getScalarData("wallDist");
   wallD->clearFlag();
 }
```

FIGURE 5. transformMesh() routine to scale mesh