TURBOSTREAM RANS Simulation and Analysis

MATLAB Script to run TURBOSTREAM

The script 'Run_Datum.m' takes the Autogrid and Pointwise .hdf5 mesh files as inputs, patches them together, applies boundary conditions and then runs them on TURBOSTREAM on a gpu.

Loading meshes:

- The two separate meshes for the blades and ducts are loaded into the script as g and g_inlet respectively
- Datum_22_input.hdf5 is from Autogrid and input.hdf5 is from Pointwise
- g and g_inlet are then patched together and reassigned to g

Block and patch IDs:

- The first figure that appears when running the scripts shows which parts of the duct are stationary and rotating. To find out the block id (bid) of each line look at the z coordinate
    - The block is accessed by g{bid+1} for g, otherwise it is just the block id number
- If new mesh is uploaded, script should work but **need to update block ids**

Applying boundary conditions:

- To apply a boundary condition to a face or 'patch' of a block you must first define the location of the face/patch area you are applying the boundary condition to. This is done by defining the correct block using p.bid = (block id) and then defining the area using the start and end coordinate in each axis direction. For j axis p.jst= (j direction start) p.jen= (j direction end) and then do the same for k and i axes.
- p.kind = (number) defines the boundary condition for the patch. The number chosen corresponds to a boundary conditions. These are defined in the file ts_tstream_patch_kind.py, containing all the types of patches and their corresponding number for p.kind. The file location on the Darwin computer is:

  cd rds/project/hpc/rds-hpc-pullan/ts3/turbostream_3.6.2/scripts/ts/

  Example:
    - g{1} accesses block 0 which has 6 patches and provides the information about it
    - ts_get_pids(g,0) gets the patch ids for block 0

Running on TURBOSTREAM:

- Once boundary conditions have been applied the following is then run in order:
    - First run on Poisson solver
    - Then mixing length model
    - If mixing length model works, can then run SA (Spell Art) model which is less stable and needs more accurate initial guess
    - Lots of information on Google about different turbulence models etc.

Post TURBOSTREAM analysis in MATLAB:

- To load the correct file first load eVTOL then:

  g = ts_read_hdf5([dr.ts 'name.hdf5']);

- To plot convergence:

  ts_plot_conv([dr.ts 'log_ducted_fan_1_fullML.txt']

- Cuts by taken at different points in the block domain to extract flow properties at that location. There are two types of cuts:

1. **Unstructured cut:** these are random cuts anywhere in the domain

   c = ts_structunstrct_cut(g, xr, bids, IJ, something else) = (g, [0.5 0 ; 0.5 10], [array of bids that cut goes through or just leave as empty array], 2 , 1)

   - The xr are the coordinates the cut goes from and to
   - if axial cut IJ = 2 ^
   - if constant r cut IJ = 1 >
   - set last entry to 1

2. **Structured cut:** these are simpler cuts along walls and the edges of blocks, these are often easier to define

   c = ts_structured_cut(g,9,5,5,1,'en',1,'en')

- Extract mass average flow properties from 2d cuts

  [Po, mass, A] = ts_mass_average(c_stator, 'Po', 3)

- the last entry is 1,2,3 depending on whether you want it in spanwise, pitchwise or averaged over the whole area
- use these values for thrust calculations – write code to do so automatically

Post TURBOSTREAM analysis in Paraview:

Streamlines of the flow can be viewed in Paraview using:

- Exports the ts file in a different format so that you can view streamlines on paraview
  - Ts_export_paraview(g,[dr.ts 'what you want to save it as .hdf5'], 'HighSpeed')

- In paraview do the following filters:
  - Extractsubset with z=0
  - Surfacevectors input vectors = V and constraint mode = parallel
  - Mask points 2/20/0 and random ticked
  - Stream tracer with customer source V/BOTH/Runge-kutta/cell length/0.2/0.010.5/100
    Input = surfacevectors
    Source = maskpoints
  - Tube mwall/normals/6/capping ticked/1e-4/off/10

Evaluating NaNs

When running meshes in TURBOSTREAM, often they will fail and in the output .txt file there will be lots of NaNs, often indicating that the mesh needs refining further. There are many reasons why a run might fail for example right handed blocks, poor mesh quality, poor boundary conditions but some methods of working out the reason are outlined below:

- Always check in this file that pressure is pulling air through the fan and isn't negative
- Steps to check error:
  - Look which block has the largest residual
  - Can also plot graphs of residuals in MATLAB, if graph has already been plotted by running datum use the 'close all' command and then run 'ts_plot_conv([dr.ts 'log_ducted_fan_1.txt'}, 1, 1) to accurately identify the block
  - Can also open the block in paraview using filters>extract block and see if the meshes line up well and whether they need refining to match better
- Can check patches are correct by ts_check_patches(g,2) and then exporting with ts_export_paraview(g,[dr.ts 'mesh_test.hdf5'],[])
  - In paraview open mesh_test.xdmf
  - Apply extractsubset2 to ducted_fan_poisson_input.xdmf to see where I direction of block is
- Can run for less steps line 211 g=ts_create_initial(g,10000,10000,I.guess) the second variable in the brackets is the step number