perforatedPlateBoundary and motivation

- A perforated plate is a plate with holes on it.
- Perforated plates split the flow into several jets.
- Perforated plates are usually used at the flow inlets.
- I am currently working on low swirl burners.
- The flow structures evolving from interaction of jets emanating from the perforated plate plays the major role in flame stabilization of these burners.

Boundary types in OpenFOAM

- Boundary conditions that have some geometrical constraints applied to that. These include empty, symmetryPlane, etc.
- Boundary conditions without any physical constraints that are all of type *patch*
- Wall does not force any physical constraints and can be treated as a simple patch with zero pressure gradient
- Wall is given a type of its own to ease the wall treatments, wall functions for example

PP implementation alternatives

- The first alternative is to consider the wall part of the perforated plate as a normal *patch type* but specify different values to the hole/wall parts of the plate. It is important to specify zero pressure gradient condition to the wall part.
- The second alternative is to split the hole/wall part of the plate into real *patch type* and *wall type* boundaries. This necessitate the manipulation of the mesh.

Alternative 1:implementation

- Here we want to compile the boundary condition statically to simpleFoam solver
- Copy the simpleFoam solver to an appropriate directory and change the name to simpleFoamPP
- Start by finding some boundary condition that approximately does what you want, oscillatingFixedValue for example
- Copy all the filesto the working directory and change the names
- Use an editor to replace all the occurances of 'oscillating' with 'perforatedPlate'
- Include the boundary condition in simpleFoam.C header

Alternative 1:implementation

- Open the perforatedPlateFixedValueFvPatchField.H file
- Open the perforatedPlateFixedValueFvPatchField.C file
- Open the pitzDaily2d/0/U file

Alternative 1:validation

- We use the pitzDaily in as a validation case
- Open constant/polyMesh/blockMeshDict file
- The inlet condition is to be changed to a perforated plate. It is defined as a patch, keep it as it is
- Do not change other files in 0/ directory , though boundaries are not set correctly for other fields it is enough for validation
- Have a look at velocity field in paraView

Alternative 1:pros and cons

Advantages:

- It is very easy and straight forward to do it.
- It is done as a high level programming so there exist no risk of interference with other parts of the code. We are at the bottom of the class hierarchy of *OpenFOAM* and we can not possibly affect any other class.
- It can be used for most of the real world problems where a perforated plate is used at the inlet where flow is normal to the plate. This will be discussed more in disadvantages below.

Alternative 1:pros and cons

Disadvantages:

- For any new derived type of *fvPatchFiled* a new boundary condition should be set up. It means that if you want an oscillating value on the boundary you should program *perforatedPlateOscillatingFixedValue* boundary condition.

- It is not possible (or very difficult if possible) to have different basic *fvPatchFiled*s on wall and hole parts of the perforated plate. For instance it is not possible to have *zero Gradient* for the wall part and *fixed Value* on the hole part.

- The whole perforated plate is set to the same *type*(*patch* or *wall*) so when it is needed to really distinguish between a *wall* and a *patch* ,e.g using wall functions, it fails. This may happen when the flow is tangential to the plate.

Alternative 2:background

- Generating the mesh with one single *patch* for the whole perforated plate and then manipulate the mesh by *polyTopoChanger* functions.
- Step into the mesh generating utility *blockMesh* and change the mesh topology directly there.
- The second is implemented here.

Alternative 2:how a mesh is set up

- The *blockMesh* utility uses a dictionary *blockMeshDict* to construct a *polyMesh* class object which is then written to several files and read by applications.
- **points** which contains a list of vectors showing the location of the *vertices*.
- **faces** which contains a list of lists where each list contains the index of *vertices* constructing the *face*.
- **owner** which contains a list of labels which are the index of the owner *cell* of each *face* present in **faces**. The size of this list is equal to the size of faces list.
- **neighbor** which contains a list of labels which are the index to the neighbor cell of the faces present in **faces**. The size of this list is smaller than the size of **faces** as it only keeps the neighbor to the internal faces. The neighbor to the boundary faces is set to -1 by default. Therefore, the size of **neighbor** is equal to the number of internal faces.
- **boundary** which includes all the boundaries as their type, name, start face index in the **faces** files and number of faces.

- **zone** files may also be present if some blocks are specified as zones.

- The important fact here is that the **faces** file is set up in such a way that first all the internal faces are written and then the boundary *patchFields* are written as continuous blocks of faces in the **faces** file. This means that any boundary name corresponds to a single continuous slice of the faces list, it has one start face index and a size.

- In order to split up a perforated plate into two patchFields we have to reorder the face list so that all the faces which together build up the hole part of the plate constitute a continuous slice of the face list, and the same rule applies for the faces building the wall part of the plate.

Alternative 2:get started

- Copy the blockMesh utility to a working directory
- Modify the Make/files
- The main function is in blockMeshApp.C file, open it

Alternative 2:how a mesh is generated

- The blockMeshDict is read
- A *blockMesh* class object names **blocks** is constructed
- This object is not changed
- A *polyPatch* object named **mesh** is constructed some properties of **blocks**

Alternative 2:what is used to construct the mesh

- *blocks.points()*
- *blocks.cells()*
- *blocks.patches()*
- **patchNames**
- **patchTypes**
- **defaultFacesName**
- **defaultFacesType**
- **patchPhysicalTypes**

Alternative 2:modifications

- Open modBlockMesh/blockMeshPP/blockMeshApp.C
- Open modBlockMesh/pitzDaily2d/constant/polyMesh/blockMeshDict
- The information about the perforated plate is under PPInfo
- Open modBlockMesh/pitzDaily2d/0/U
- Follow the oral presentation
- Compile blockMeshPP utility