# Tutorials

DAFOAM的Tutorials很多，可以到网站上选择学习：

<https://dafoam.github.io/mydoc_tutorials_aero_m6.html>

## Aerodynamics

Onera M6 wing

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is an aerodynamic shape optimization case for the Onera M6 wing in transonic conditions.

Case: Wing aerodynamic optimization

Geometry: Onera M6 wing

Objective function: Drag coefficient (CD)

Lift coefficient (CL): 0.27

Design variables: 120 FFD points moving in the y direction, five twists, and one angle of attack.

Constraints: volume, thickness, LE/TE, and lift constraints (total number: 114)

Mach number: 0.839 (285 m/s)

Reynolds number: 11.7 million

Mesh cells: ~37,000

Solver: DARhoSimpleCFoam

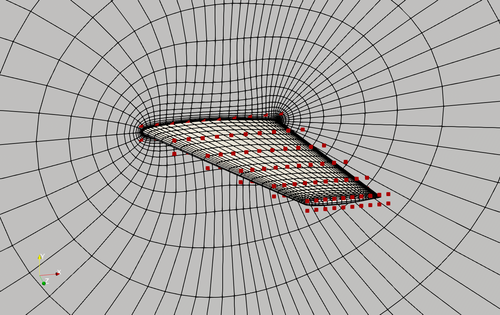


Fig. 1. Mesh and FFD points for the Onera M6 wing.

The “runScript.py” is based on the one used in the NACA0012 [transonic case](https://dafoam.github.io/mydoc_tutorials_naca0012_transonic.html) with the following modifications:

* In “meshOptions”, we set only one symmetry plane, instead of two symmetry planes used in the 2D airfoil case.
* We compute the number of spanwise FFD points and set it to “nTwist” by calling nTwists = DVGeo.addRefAxis(“bodyAxis”, xFraction=0.25, alignIndex=”k”).
* We define a function to change the twist at these spanwise FFD sections. Note that we do NOT change the root twist (we already had angle of attack as design variable), so the first element in the twist design variable is the twist at the 2nd spanwise location.

def twist(val, geo):

for i in range(1, nTwists):

geo.rot\_z["bodyAxis"].coef[i] = val[i - 1]

* We call the following functions to add the twist design variable. Again, we have nTwist-1 twists because we do not change the root twist.

DVGeo.addGeoDVGlobal("twist", np.zeros(nTwists - 1), twist, lower=-10.0, upper=10.0, scale=1.0)

daOptions["designVar"]["twist"] = {"designVarType": "FFD"}

* The leList and teList are oblique to the y axis. They should be close to the leading and trailing edges but completely within the wing surface.

leList = [[0.01, 0.0, 1e-3], [0.7, 0.0, 1.19]]

teList = [[0.79, 0.0, 1e-3], [1.135, 0.0, 1.19]]

* We call the following functions to constrain the leading and trailing edge FFD movements by requiring them to move in the opposite directions. There is no need to manually set up the LE/TE linear constraints, as was done in the 2D airfoil case. Also, there is no need to impose symmetry constraints for k=0 and k=1 since this is a 3D wing case.

# Le/Te constraints

DVCon.addLeTeConstraints(0, "iLow")

DVCon.addLeTeConstraints(0, "iHigh")

To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/Onera\_M6\_Wing and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

The above script will generate a structured hex mesh using pyHyp. Alternatively, you can generate an unstructured snappy hex mesh by calling:

./preProcessing\_snappyHexMesh.sh

Then, use the following command to run the optimization with 8 CPU cores:

mpirun -np 8 python runScript.py 2>&1 | tee logOpt.txt

For the structured hex mesh, the case ran for 50 steps and took about 3 hours using Intel 2.6 GHz CPU with 8 cores on one Skylake node of [Stampede 2](https://portal.xsede.org/tacc-stampede2). According to “logOpt.txt” and “opt\_SLSQP.txt”, the initial drag is 0.016597241 and the optimized drag is 0.013540159 with a drag reduction of **18.4%**.

The evolution of pressure and shape during the optimization is as follows.

Fig. 2. Pressure and shape evolution during the optimization process

## Aerostructural

MACH tutorial wing

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is an aerostructural shape optimization case for the MACH tutorial wing in subsonic conditions. The flow is solved using the DARhoSimpleFoam CFD solver and the structure is solved using an open-source FEM solver [TACS](https://github.com/smdogroup/tacs). The load and displacement transfer is computed using [FUNtoFEM](https://github.com/smdogroup/funtofem). The aerostructural coupling is implemented in the [OpenMDAO/Mphys](https://github.com/OpenMDAO/mphys) framework.

Case: Wing aerostructural optimization

Geometry: MACH tutorial wing

Objective function: Drag coefficient (CD)

Lift coefficient (CL): 0.5

Design variables: 96 FFD points moving in the y direction, seven twists, and one angle of attack.

Constraints: volume, thickness, LE/TE, lift, and stress constraints (total number: 118)

Mach number: ~0.3 (100 m/s)

Reynolds number: ~30 million

Mesh cells: ~38,000

Solver: DARhoSimpleFoam

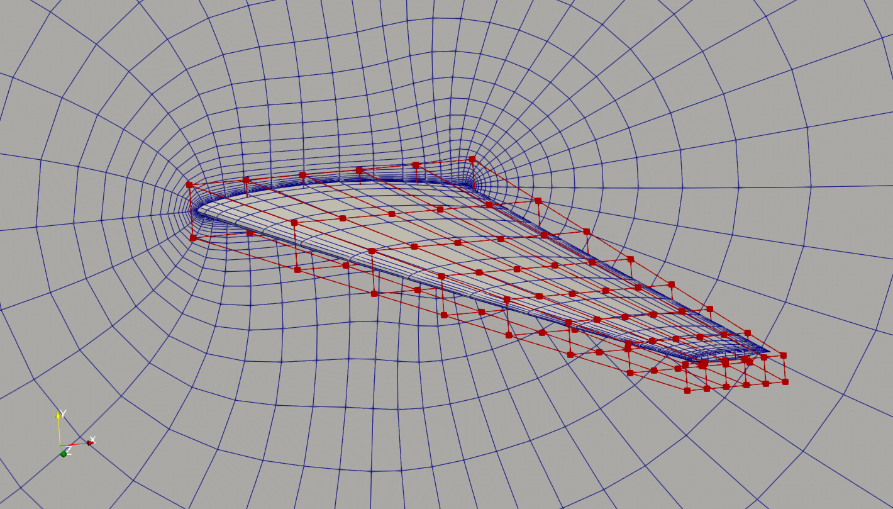


Fig. 1. Mesh and FFD points for the MACH tutorial wing.

To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/MACH\_Tutorial\_Wing and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

The above script will generate a structured hex mesh using pyHyp. Then, use the following command to run the optimization with 8 CPU cores:

mpirun -np 8 python runScript\_AeroStruct.py 2>&1 | tee logOpt.txt

## Heat\_Transfer

U bend channel

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is a heat transfer optimization case for the von Karman U bend channel, representative of a internal cooling section in turbines.

Case: Heat transfer optimization for U bend cooling channels

Geometry: von Karman U bend channel

Objective function: Combined wall heat flux and total pressure loss

Design variables: 114 FFD points moving in the x, y, and z directions

Constraints: Symmetry constraint (total number: 38)

Mach number: 0.02

Reynolds number: 4.2e4

Mesh cells: 4.8 K

Solver: DASimpleTFoam

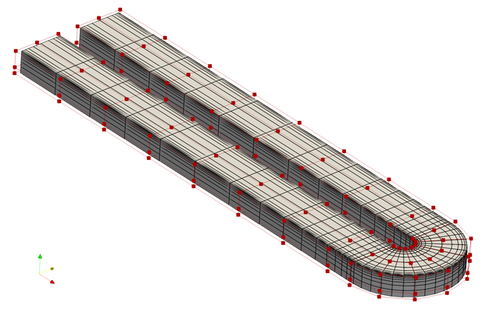


Fig. 1. Mesh and FFD points for the U bend channel case

In this case, we set up a combined objective function of wall heat flux and total pressure loss. This is done by setting three parts for the objFunc. part1 and part2 are the total pressure at the inlet and outlet, respectively. part3 is the wall heat flux. The weights between the heat flux and total pressure loss is 0.5.

"objFunc": {

"obj": {

"part1": {

"type": "totalPressure",

"source": "patchToFace",

"patches": ["inlet"],

"scale": 1.0 / CPL0 \* CPL\_weight,

"addToAdjoint": True,

},

"part2": {

"type": "totalPressure",

"source": "patchToFace",

"patches": ["outlet"],

"scale": -1.0 / CPL0 \* CPL\_weight,

"addToAdjoint": True,

},

"part3": {

"type": "wallHeatFlux",

"source": "patchToFace",

"patches": ["ubendup"],

"scale": 1.0 / HFL0 \* HFL\_weight,

"addToAdjoint": True,

}

},

},

To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/UBend\_Channel and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

Then, use the following command to run the optimization with 4 CPU cores:

mpirun -np 4 python runScript.py 2>&1 | tee logOpt.txt

This case ran for 18 iterations. The heat flux remained the same while the total pressure loss reduced by 53.4%.

Fig. 2. Evolution of wall heat flux and velocity during the optimization

## Solid\_Mechanics

Plate hole

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is a structural optimization case for a plate with a hole in the middle.

Case: Structural optimization for the plate hole configuration

Geometry: Plate hole

Objective function: Weight

Design variables: 24 FFD points moving in the x and y directions

Constraints: Symmetry constraint (total number: 18), max stress constraint

Mesh cells: 4 K

Solver: DASolidDisplacementFoam

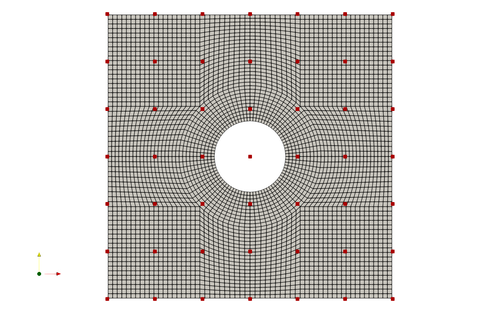


Fig. 1. Mesh and FFD points for the plate hole case

|  |
| --- |
|  |

To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/PlateHole\_Structure and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

Then, use the following command to run the optimization with 1 CPU cores:

python runScript.py 2>&1 | tee logOpt.txt

After conducting the optimization, use ParaView to ensure the simulation has been completed successfully. There should be 10 geometry iterations which minimize mass while preserving the original maximum stress.

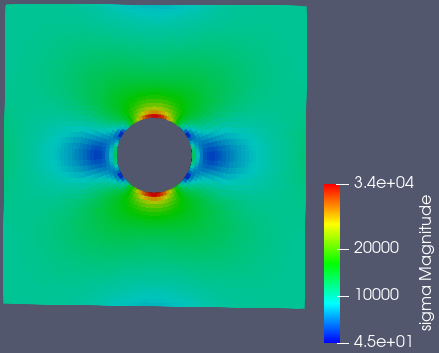


Fig. 2. Stress results and distribution before optimization

Fig. 3. Optimization results and animation

As you can see, the optimization eliminates material to reduce component weight while preserving the maximum stress. The mass is reduced from 11950 to 10701, or a reduction of mass by 10.5 percent. This can be seen in the logOpt.txt output file.

We have gone further to compare the simulation results from OpenFoam to the results from a numerical solution for a plate hole case and the same simulation executed in an established software ANSYS, version 2019 R2. The results are shown below in figure 7 below.



Fig. 4. Stress equation for a plate hole

This equation in Fig 4 assumes the plate has infinite width, which causes some discrepancy near the edge of the plate. The maximum case for this equation occurs at +90 deg and -90 deg.

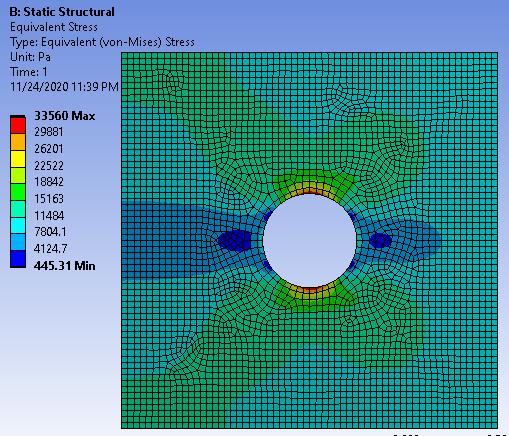


Fig. 5. ANSYS stress results and distribution before optimization with mesh

It was not possible to use the exact constraints as ANSYS would display an error that it was under constrained, so the setup is shown in figure 6. Also, an automated mesh was used which differs from the mesh in OpenFoam. This will result in varying results, but will sufficiently correlate results between OpenFoam and ANSYS solutions.

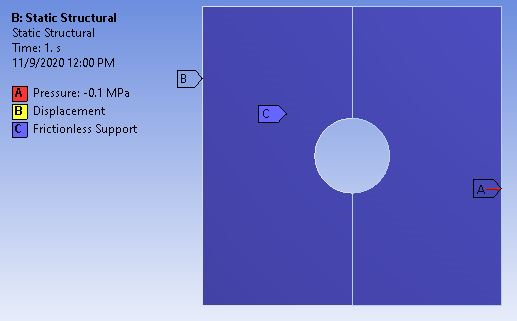


Fig. 6. ANSYS simulation setup

Additionally, by plotting both OpenFoam and ANSYS results against the numerical solution, a comparison can be made between the various methods and tools for stress analysis. The numerical solution is created using the equation shown in Fig 4.

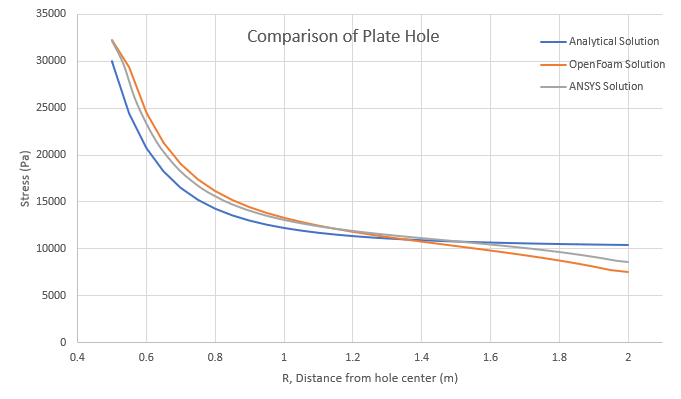


Fig. 7. Plate hole case comparison graph

Both ANSYS and OpenFoam show good relation with the numerical case. OpenFoam shows very good relation to an established tool for finite element structural analysis. Very near the hole, the analysis tools show a higher stress value than the numerical solution. This creates a conservative case for design purposes. On the extremity, the numerical solution has a higher stress than the analysis tools. This is due to the presence of the plate edge. The maximum stress for OpenFoam and ANSYS are 32205 Pa and 32174 Pa, respectively. Compared to the 30000 Pa numerical solution, this creates a percent error of 7.35% for OpenFoam and 7.25% for ANSYS. As for the minimum stress case, OpenFoam and ANSYS result in 7580 Pa and 8580 Pa, respectively. Comparing this again to the numerical solution of 10371 Pa, this creates a percent error of 26.9% for OpenFoam and 17.3% for ANSYS. The hole overestimates stress while the extremity underestimates stress using the simulation tools. Between these two points, the correlation is quite good. The average percent error is 10.7% for OpenFoam and 7.0% for ANSYS.

The validation of OpenFoam’s analysis supports the optimization results. The 10.5% decrease in mass is significant and the plate hole case is common in aerospace with an abundance of fastener holes in many structures.

-By Adam Bodenham

## Hydrodynamics

JBC bulk carrier hull (船舶)

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is a hydrodynamic optimization for the JBC bulk carrier hull.

Case: Ship hydrodynamic optimization with self-propulsion

Geometry: Japan Bulk Carrier (JBC) hull

Objective function: Drag

Design variables: 32 FFD points moving in the y direction

Constraints: Volume, thickness, symmetry, and curvature constraints (total number: 83)

Mach number: less than 0.01

Reynolds number: 7.5 million

Mesh cells: 265 K

Solver: DASimpleFoam

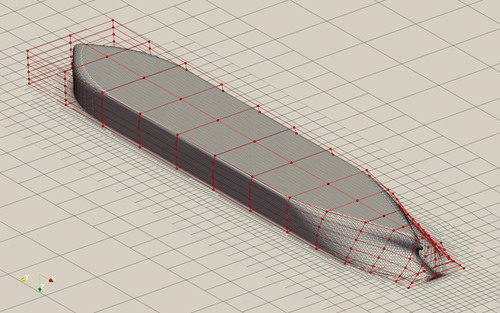


Fig. 1. Mesh and FFD points for the JBC hull

To avoid wave shapes, this case impose a curvature constraint to the hull shape. The aggregated curvature (approximated max-curvature) in the optimized design is constrained to be less than 1.2 times of the curvature in the baseline design. See the following code to add the curvature constraint. Here the “hullCurv.xyz” file is a 2D surface mesh that define which area the curvature constraint is applied to. This surface mesh should be as close as possible to the hull shape. Then, pyGeo will load in this surface mesh in the FFD box and deform it along with the hull. This way, the curvature of this 2D surface mesh is the curvature of the hull.

# Curvature constraints

DVCon.addCurvatureConstraint(

"./FFD/hullCurv.xyz", curvatureType="KSmean", lower=0.0, upper=1.21, addToPyOpt=True, scaled=True

)

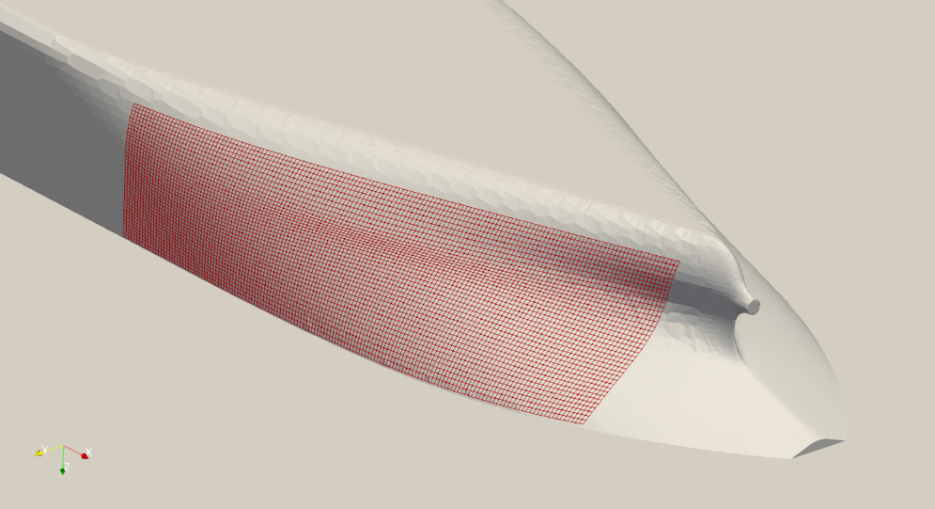


Fig. 2. Example of adding curvature constraint. The red mesh is the 2D surface mesh defined hullCurv.xyz

This case requires the IPOPT optimizer and the AD version of DAFoam. To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/JBC\_Hull and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

We recommend running this case on an HPC system with 30 CPU cores:

mpirun -np 30 python runScript.py 2>&1 | tee logOpt.txt

This case ran for 40 steps. The drag reduces by 2.1%.

\*.gif

Fig. 3. Evolution of hull shape and pressure distribution during the optimization.

## Topology\_Optimization

pitzDaily

**Note:** We recommend going through the tutorial in [Get started](https://dafoam.github.io/mydoc_get_started_download_docker.html) before running this case.

The following is a density-based topology optimization case for the pitzDaily case in OpenFOAM.

Case: pitzDaily topology optimization

Geometry: pitzDaily channel

Objective function: Total pressure loss

Design variables: 12,225 alpha porosity values in the flow field

Constraints: None

Mach number: 0.12 (40 m/s)

Reynolds number: 0.13 million

Mesh cells: ~12,225

Solver: DASimpleFoam

Fig. 1. Velocity and alpha porosity evolution during the optimization

The “runScript.py” for a topology optimization is similar to the one used for shape optimization. In other words, we still need to specify a dummy design surface (although we will not move it)

"designSurfaces": ["upperWall"],

And we need to create a dummy FFD box to cover the dummy design surface and load it to DVGeo,

DVGeo = DVGeometry("./FFD/dummyFFD.xyz")

We also need to specify set up a dummy reference axis to make sure we can use global design variable (alpha porosity field)

DVGeo.addRefAxis("dummyAxis", xFraction=0.25, alignIndex="k")

Other notes:

* This case only works with the **SNOPT** optimizer at this moment.
* We need to specify the total number of cells “nCells”.
* We need to choose the Jacobian free option in daOptions: “adjJacobianOption”: “JacobianFree”. This means that we need to compile the AD version of OpenFOAM and DAFoam (see [here](https://dafoam.github.io/mydoc_installation_source.html#compile-dafoam-with-automatic-differentiation-optional)). If you use the Docker image, they have been compiled so no additional action is needed.
* We need to properly scale the design variable because the alpha porosity field could be very large, e.g., 1e4. So we use “scale=1e-4” in DVGeo.addGeoDVVGlobal(“alphaPorosity”, …).

To run this case, first download [tutorials](https://github.com/DAFoam/tutorials/archive/main.tar.gz) and untar it. Then go to tutorials-main/pitzDaily and run the “preProcessing.sh” script to generate the mesh:

./preProcessing.sh

Then, use the following command to run the optimization with 4 CPU cores:

mpirun -np 4 python runScript.py 2>&1 | tee logOpt.txt

The case ran for 36 steps and took about 2 hours using Intel 3.0 GHz CPU with 4 cores. According to “logOpt.txt” and “opt\_SNOPT\_summary.txt”, the initial pressure loss is 0.28228 and the optimized drag is 0.16579 with a pressure-loss reduction of **41.3%**.

## Field\_Inversion

**Note:** The periodic hill case is known to have an adjoint accuracy issue when running in parallel. So please run this case in serial. This issue affects ONLY cases with periodic boundary conditions. For other cases without periodic boundary conditions, such as the wind turbine and hump, we can run field inversion in parallel.

Overview

The following is a demonstration of how to perform field inversion using DAFoam. We have selected the periodic hill flow as a demonstrative case. In this tutorial we will show how we can augment the Spalart-Allmaras model using velocity field data for “training”. For the purposes of this tutorial, we will be treating the results from the k-epsilon model as the reference data (for simplicity). We have found the k-epsilon results to be closer to high-fidelity data compared to other RANS models.

Reynolds number: 5,600

Mesh cells: 3,500

Adjoint solver: DASimpleFoam

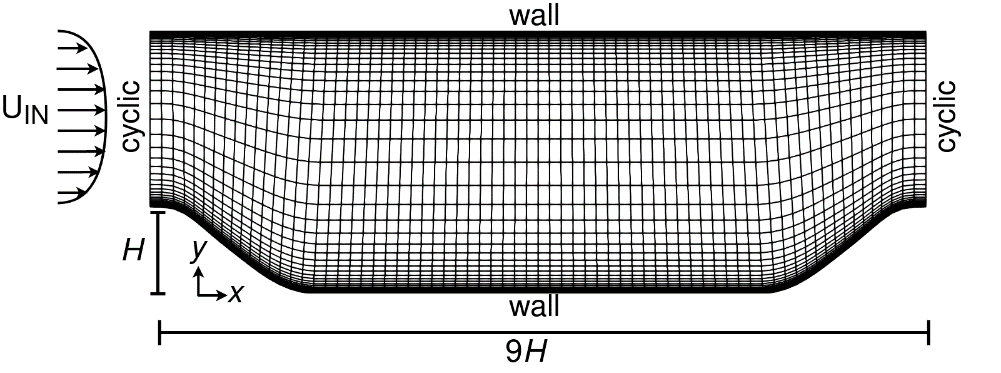


Fig. 1. Periodic hill geometry and mesh

Note

The mesh used in this case is very coarse, in order to allow a relatively fast run time. While appropriate for demonstrative purposes, the users are cautioned that these field inversion results are sub-optimal and that better results are achievable with a higher-quality mesh.

Contact

Please note that the field inversion feature in DAFoam is a work in progress. More tutorials, documentation, and features will be added as the work progresses. In the meantime, if you have questions about the tool or would like to collaborate please get in touch: obidar1@sheffield.ac.uk.