# Multidisciplinary Design Project
# 2022 General Motors Knowledge Transfer

**Sponsor mentor**: Dr. Erik Yen
**Faculty mentor**: Professor Gregory Hulbert
**Student team**: Sanjay Bharati, Yigit Can Cevikol, William Fielding,
Ha Young Kim, Benjamin Liu, Garima Shah

## Table of Contents

## Overview of the MTO Solver

The Multiphysics Topology Optimization (MTO) program is a parallel solver for multi-physics topology optimization on structured grids created by Minghao Yu, Shilun Ruan, Junfeng Gu, Mengke Ren, Zheng Li, Xinyu Wang, and Changyu Shen in 2020. The MTO repository contains solvers for various design generation problems. The solver that pertains most closely to the thermal-fluid analysis problem the 2022 GM MDP team was studying is the 2Dheatsink solver.

The 2Dheatsink solver has an objective function that measures cooling capacity by mean temperature as primary concern.

From the paper, we observe this objective function as

$$\Psi = \frac{1}{|\Omega|}\int_{\Omega} T d\Omega$$

"where $\Psi$ denotes the objective function, and $|\Omega|$ denotes the volume of the computational domain."

MTO Github: https://github.com/MTopOpt/MTO

Yu, M., Ruan, S., Gu, J. et al. Three-dimensional topology optimization of thermal-fluid-structural problems for cooling system design. Struct Multidisc Optim (2020). https://doi.org/10.1007/s00158-020-02731-z

### Parameters in MTO

Ω = domain (fluid or solid)
Γ = Gamma = boundary
γ = gamma = design variable bounds = fluid/solid
β = the volume fraction occupied by the fluid
$_D$ = Dirichlet portion
$_N$ = Neumann portion
$^F$ = fluid
$^T$ = thermal
$^S$ = solid
ρ = rho = fluid mass density
u = velocity
p = pressure
η = eta = dynamic viscosity
$F_b$ = body force of the fluid
g = stress
T = temperature
n = unitary outward normal
c = specific heat capacity
k = thermal conductivity

$Q$ = heat source
$D$ = solid displacement
$\mu$ and $\lambda$ = mu and lambda = Lamé parameters for the solid
$T_{ref}$ = reference temperature (set to 0)
$a_T$ = thermal expansion coefficient
$I$ = second order unit tensor
$\Psi$ = psi = objective function

## Overview of MTO Setup

To run the MTO program, you will need to have the following software:

1. Ubuntu 18.04. This version is required to run the appropriate OpenFOAM version.
2. OpenFOAM 5.0
3. Paraview 5.4
4. swak4Foam development version
5. PETSc 3.12
6. openMPI 4.1.2

## Setting up the MTO Dependencies

1. Setup Ubuntu 18.04
   a. Setting up Ubuntu 18.04 can be done through WSL for Windows, a virtual machine, or installing it directly to disk.
      i. Tutorial for installing Ubuntu 18.04 using WSL: https://www.public-health.uiowa.edu/it/support/kb48549/
      ii. Tutorial for setting up Ubuntu 18.04 on virtual machine: https://codebots.com/docs/ubuntu-18-04-virtual-machine-setup
      iii. Tutorial for installing Ubuntu 18.04 to disk: https://ubuntu.com/tutorials/install-ubuntu-desktop-1804#1-overview
2. Install OpenFoam5
   a. Description: C++ toolbox for the development of customized numerical solvers, and pre-/post-processing utilities for the solution of continuum mechanics problems, most prominently including computational fluid dynamics (CFD)
   b. Run the following commands in a  Ubuntu terminal to allow apt to recognize OpenFoam
      i. `$ sudo sh -c "wget -O - http://dl.openfoam.org/gpg.key | apt-key add -"`
      ii. `$ sudo add-apt-repository http://dl.openfoam.org/ubuntu`
      iii. `$ sudo apt-get update`
   c. Enter the following command into terminal to download OpenFoam 5 in the current working directory
      i. `$ sudo apt-get -y install openfoam5`
   d. Once OpenFoam5 is installed edit your .bashrc file in for your system to recognize openfoam commands
      i. Open .bashrc with text editor such as vim or gedit (eg. '`$ vim ~/.bashrc`')
      ii. Add '`source /opt/openfoam5/etc/bashrc`' to the end of the .bashrc file
   e. Create a new project directory with the following command
      i. `mkdir -p $FOAM_RUN`
   f. CHECK POINT

        i. You should be able to run the following commands. If there are problems with installation, please refer to the installation problems section
        ii. `cd $FOAM_RUN`
        iii. `cp -r $FOAM_TUTORIALS/incompressible/simpleFoam/pitzDaily .`
        iv. `cd pitzDaily`
        v. `blockMesh`
        vi. `simpleFoam`
        vii. `paraFoam`
    g. OpenFoam user guide
3. Install ParaView 5.4
    a. Description: ParaView is an open-source multiple-platform application for interactive, scientific visualization. It has a client–server architecture to facilitate remote visualization of datasets, and generates level of detail models to maintain interactive frame rates for large datasets.
    b. The full ParaView suite can be found at https://www.paraview.org/download/. Select version 5.4 and download the version that is compatible with your system
    c. ParaFoam, the openfoam adapted version of pareview is included with the openFoam5 installation
4. Install swak4foam development version
    a. Description: GroovyBC to specify arbitrary boundary conditions based on expressions.
    b. Create a folder inside the OpenFoam working directory to store swak4foam and other dependencies such as petsc. (eg. ‘`mkdir dependencies`’)
        i. `sudo apt install git`
        ii. `sudo apt install make`
    c. swak4Foam dependencies
        i. M4
            1. `wget ftp://ftp.gnu.org/gnu/m4/m4-1.4.10.tar.gz`
            2. `tar -xf m4-1.4.10.tar.gz`
            3. `cd m4-1.4.10`
            4. `./configure --prefix=/usr/local/m4`
            5. `make`
            6. `sudo make install`
            7. `cd ..`
        ii. Bison
            1. `wget "http://ftp.gnu.org/gnu/bison/bison-3.3.2.tar.xz"`
            2. `tar -xf bison-3.3.2.tar.xz`
            3. `cd bison-3.2.2`
            4. `PATH=$PATH:/usr/local/m4/bin/`

5. `./configure --prefix=/usr/local/bison --with-libiconv-prefix=/usr/local/libiconv/`
6. `make`
7. `sudo make install`
8. `cd ..`

d. In the dependencies directory enter the following command to download swak4foam
   i. `$ git clone https://github.com/Unofficial-Extend-Project-Mirror/openfoam-extend-swak4Foam-dev.git swak4Foam`

e. The following command will create a new folder titled ‘swak4foam’. In this directory run the following script
   i. `$ PATH=$PATH:/usr/local/bison/bin`
   ii. `$ ./Allwmake`

5. Download PETSc with wget (version 3.12)
   a. Descirption: A suite of data structures and routines developed by Argonne National Laboratory for the scalable solution of scientific applications modeled by partial differential equations.
   b. `$ wget "https://ftp.mcs.anl.gov/pub/petsc/release-snapshots/petsc-3.12.1.tar.gz"`
   c. Make sure to run configuration steps
      i. Download openmpi version 4.1.2
      ii. Run configuration steps for openmpi before configuring petsc
         1. `$ tar -xf openmpi-4.1.2.tar.gz`
         2. `$ cd openmpi-4.1.2`
         3. `$ ./configure --prefix=/usr/local`
         4. <...lots of output...>
         5. `$ make all install`
      iii. Configure with petsc
         1. `$ ./configure --with-cc=gcc --with-cxx=g++ --with-fc=gfortran --download-openmpi={directory to openmpi tarball}/openmpi-4.1.2.tar.gz --download-fblaslapack`
   d. You may need to install gfortran with `$ sudo apt install gfortran`
   e. If issues installing gfortran, sudo apt-get updates and run the above command again
   f. Run $ make all check
6. To build PETSc libraries with (gnu make build), run the following commands. Replace the "..." with the appropriate path to the petsc directory.
   a. `$ make PETSC_DIR=/home/…/petsc/ PETSC_ARCH=arch-linux-c-debug`

b. `$ export PETSC_DIR=/home/…/petsc/ PETSC_ARCH=arch-linux-c-debug`
c. Add the export line to ~/.bashrc (can be accessed with `$ gedit ~/.bashrc`)
    i. Go to the bottom of the bashrc and enter this on a newline: `export PETSC_DIR=/home/…/petsc/ PETSC_ARCH=arch-linux-c-debug`. This is to avoid exporting the same paths again after restarting the terminal.

## Setting up the MTO Repository

Note: These steps do not include the modifications to the MTO code that are needed to design for the GM battery module. The steps to design for the GM battery module are under the "Code Modifications" section.

1. Clone the MTO repository https://github.com/MTopOpt/MTO into the $FOAM_RUN folder. We have also provided our modified repository that designs for a domain with D_normalization, velocity of the coolant at the inlet, and the heat source that fit the GM battery module conditions provided by our sponsor mentor.
    a. `$ cd $FOAM_RUN`
    b. `$ git clone https://github.com/MTopOpt/MTO`
2. Compile the C++ script (MMA.C MMA.H in the MMA folder) into a dynamic library named libMMA_yu.so
    a. `$ g++ *.c -c -fPIC -I $PETSC_DIR/include -I $PETSC_DIR/$PETSC_ARCH/include`
    b. `$ g++ *.o -shared -o libMMA_yu.so`
    c. `$ export LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH`
    d. You may need to connect with petsc.h and petscconf.h
        i. Use -I flag with path to include petsc.h (ex: -I../petsc/include)
        ii. Use another -I flag to reach petscconf.h in arch-linux-c-debug/include
3. Put the compiled C++ script that you just created (libMMA_yu.so) in $FOAM_USER_LIBBIN
    a. `$ mv libMMA_yu.so $FOAM_USER_LIBBIN`
    b. If the above doesn't try, try replacing '$FOAM_USER_LIBBIN' with the actual path to $FOAM_USER_LIBBIN. To find the path to $FOAM_USER_LIBBIN, run `$ echo $FOAM_USER_LIBBIN | tr ':' '\n'`
4. CHECK POINT: your path variables should be as follow:
    a. `PETSC_DIR=/home/…/petsc/`
    b. `PETSC_ARCH=arch-linux-c-debug`
    c. `LD_LIBRARY_PATH=$PWD:$LD_LIBRARY_PATH`
    d. *Optional to add these lines to ~/.bashrc to avoid manually importing every terminal session*

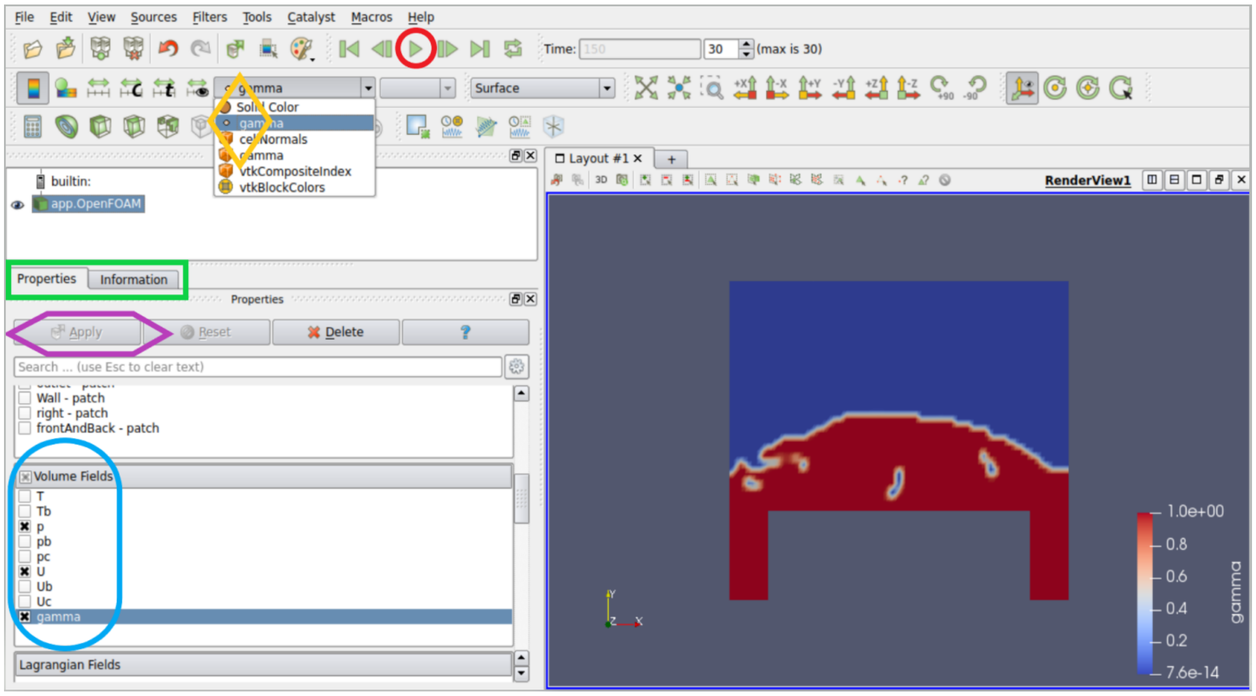
## Running the 2Dheatsink Solver

1. Run `$ wmake` in the 2Dheatsink/src_TF folder. Generally, this should be performed after updating files in the 2Dheatsink/src_TF folder. You may get some warnings of use of old-style cast or unused variables. Make sure that there are no Error messages. If there are, the setup was unsuccessful. Resolve the errors before continuing.
    a. If there are errors regarding missing files, please use the `find` command to find the missing files and add the directory to your `PATH` variable

b. For a compile error in petscmath.h pow(a,b) is overloaded. To fix this, replaced the line (*#define PetscPowReal(a,b)   pow(a,b)*) that was causing the issue in petscmath.h with this:
    i. *#ifdef __cplusplus*
    ii. *#define PetscPowReal(a,b) std::pow(a,b)*
    iii. *#else*
    iv. *#define PetscPowReal(a,b)   pow(a,b)*
    v. *#endif*
2. Run `$ blockMesh` in the 2Dheatsink/app folder. Generally, this should be performed after editing files in the 2Dheatsink/app folder.
3. Run `$ MTO_TF` in the 2Dheatsink/app folder.

## Displaying the Resulting Geometry in ParaView

1. Run `$ paraFoam` in the 2Dheatsink/app folder. This will open paraView
2. Under the "Properties" tab (see green rectangle in the image below), scroll down to the "Volume Fields" box. Select "gamma." (See blue oval in the image below.)
3. Click the "Apply" button at the top of the "Properties" tab. This button should turn from a gray to a green color after you've selected "gamma" in the previous step. (See purple hexagon in the image below.)
4. Select "gamma" under the drop down. (See yellow diamond in image below.) If "gamma" is not an option, hit the green Play button on the top control bar and try again. (See red circle in the image below.)



## Code Modifications

The following are instructions for setting the boundary conditions and constant values D_normalization (1.23053e-7), velocity of the coolant at the inlet (0.147), and the heat source Q (419500.0) to design for the GM battery module.

Under the 2Dheatsink directory, there are two main folders, i.e., app and src_TF. ‘app’ has three subfolders: ‘0’, ‘constant,’ and ‘system.’

There are several modifications done in the files of the app folder. In the src_TF folder, there is a change in only the readThermalProperties.H file.

### Defining the Cooling Plate Geometry

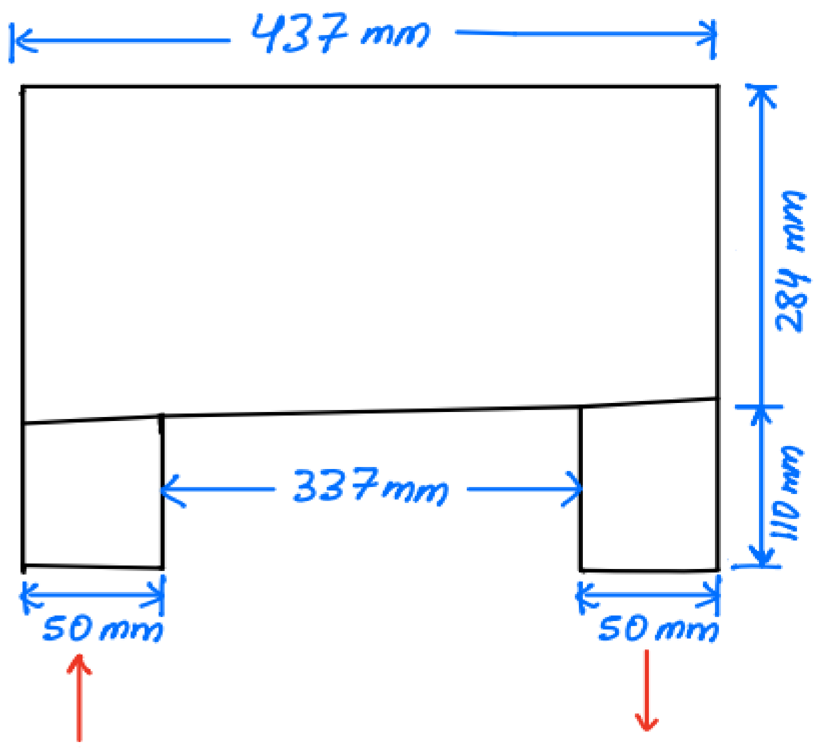


**Figure 1**

The cooling plate geometry has an inlet on the left and an outlet on the right (as indicted by the red arrows). The inlet and outlet have dimensions of 50 mm x 110 mm in the XY plane. The overall dimensions of the cooling plate is 437 mm x 394 mm.

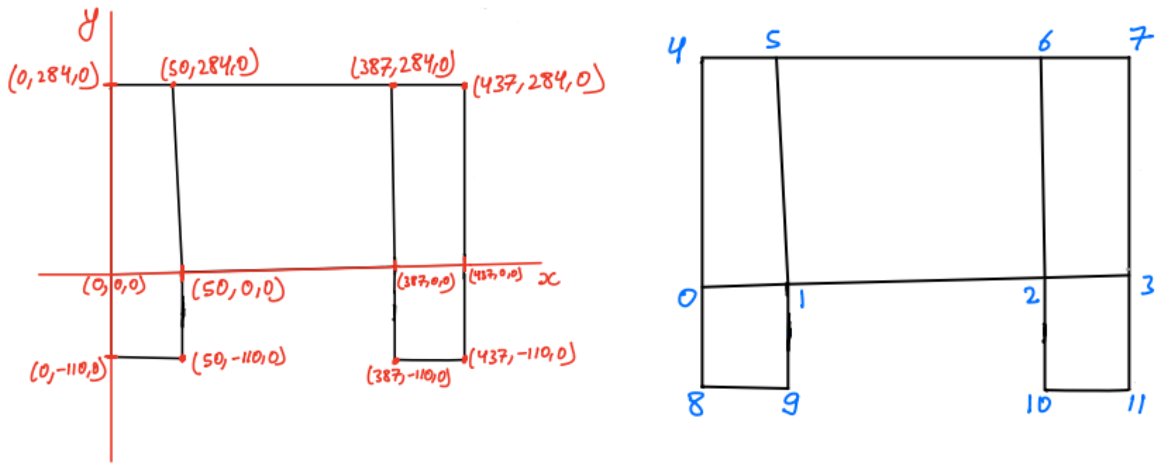


**Figure 2**

The picture on the right of **Fig.2** represents the front plane of the cooling plate in the XYZ plane constructed with 12 vertices. These vertices are numbered starting from the origin as 0, as shown in the left picture.

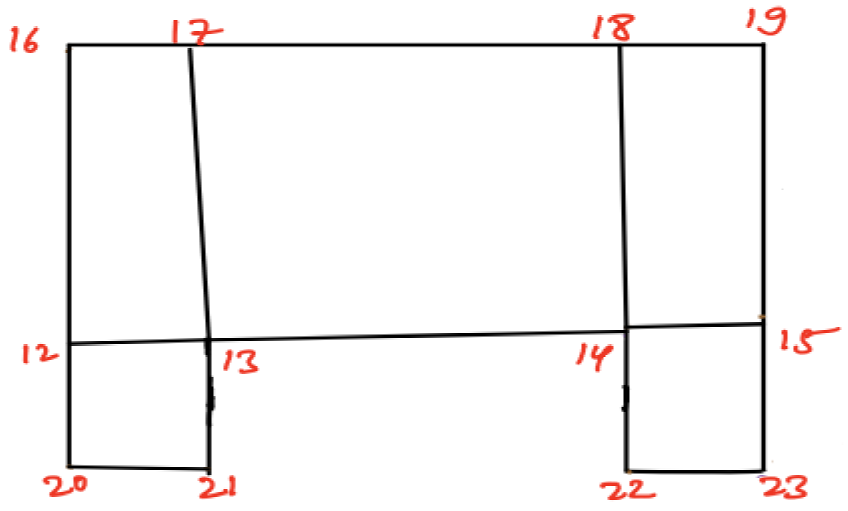


**Figure 3**

Similarly, the back plane of the cooling plate geometry is constructed, as shown in **Fig.3,** by replacing the Z value of the vertices of the front side with 1. This means that the back plane is 1 unit away from the front plane in Z direction.

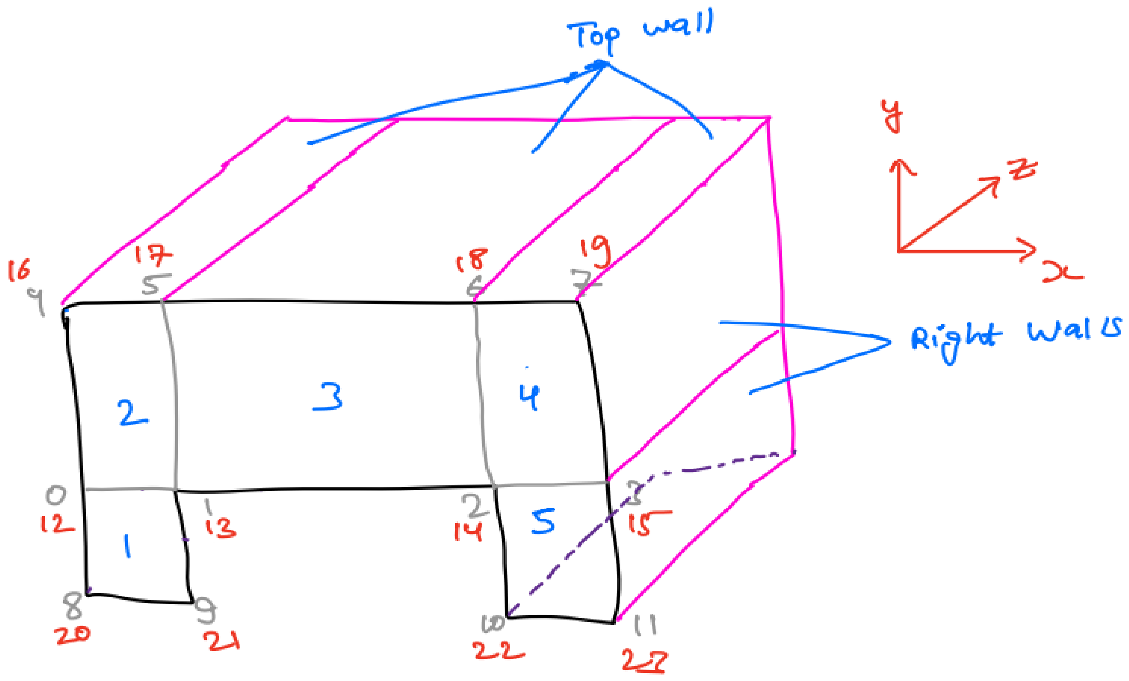


**Figure 4**

**Figure 4** is the cooling plate geometry in 3D.

## Steps to Generate Geometry in OpenFOAM with blockMeshDict File

Location of file: app/system/blockMeshDict

1. Write all twelve vertices of the front plane (**Figure 2)** starting from 0 and in numerical order. (lines 20 to 33 of the file)
2. Similarly, write all twelve vertices of the back plane (**Figure 3).** (lines 36 to 49)
3. Divide the **Figure 4** geometry into five blocks (1,2,3,4,5)
    - Block 1 - inlet (fluid zone)
    - Block 5 - outlet (fluid zone)
    - Block 2,3,4 - Design Space

★ All blocks are hexagonal. So use the function hex().
★ We need front and back plane vertices to represent the block.
★ Start with lower left vertices and make a counterclockwise rotation of the front side. You have 4 vertices now. Do the same with the backside, and you will have another 4 vertices.

- hex() (a b c). (a b c) divides the hexagonal block into *a* parts in X, *b* parts in Y, and *c* parts in Z direction. For 2D, keep c = 1. This step is for creating mesh.

4. Write boundary (inlet, outlet, wall, right)
    - Write boundary type
        - ★ Boundary type patch means that contains no geometric information about the mesh.
        - ★ Boundary-type wall is used for wall functions.
        - ★ Boundary-type symmetry is used for a symmetry plane (We are using symmetry on the right side because, in the actual cooling plate geometry, we have two inlets on the side and one outlet in the middle. We cut the geometry from the middle of the outlet into two symmetrical parts. We are taking the left part.)
        - ★ Boundary type empty is used for the front and back plane to instruct the OpenFOAM to solve in 2D. This is necessary because OpenFOAM always generates geometry in 3D.
    - Write vertices of the faces following counterclockwise rotation and starting with the origin.

## Modifications in the app folder

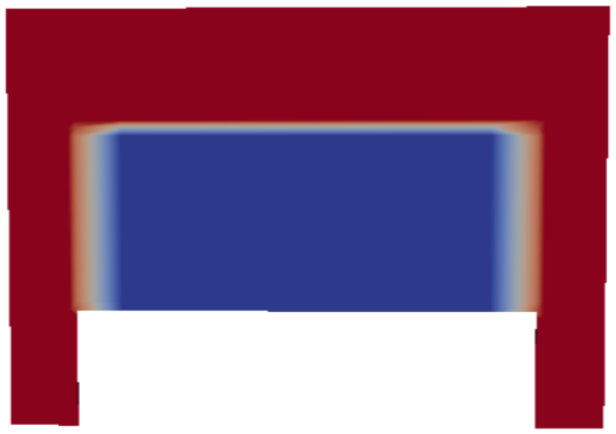
1. Temperature
    a. File location: app/0/T
    b. In line 26 of the file, change the temperature of the inlet coolant value to the desired temperature. Our sponsor mentor provided the value of 25°C (≈ 298 K) for the coolant at the inlet.
        - ★ Units are specified in line 17 of the file
        - ★ In function dimensions [ ], write the power of SI unit of the seven fundamental quantities in the order as Length (m), Mass (kg), Time (s), Temperature (K), Amount of Substance (mol), Electric Current (A), Luminous Intensity (cd).
        - ★ OpenFOAM reads the temperature in Kelvin only
    c. For file Tb (location: app/0/Tb): follow the same procedure mentioned in 1(b)
2. Velocity
    a. File location: app/0/U
    b. In line 26 of the file, change the velocity of the inlet coolant value to the desired velocity.
        - ★ Velocity has three components in the XYZ plane.
        - ★ Coolant is entering in the y direction. So put the x and z component as zero and change the y component to the desired value.
    c. Velocity calculation
        i. Volumetric Flow Rate (Q) = 1 L/min = 1.667e-3 m^3/s (GM provided value for 10 battery modules is 10 L/min. So we reduced it to 1 L/min for 1 battery module)
        ii. Cross section area of the inlet (A) = $\frac{\pi}{4}(diameter\ of\ inlet\ pipe)^2$ (GM cad models have a 12 mm inlet pipe diameter. We are using the same area

for our geometry as well, although we have a rectangular inlet because of the limitations of representing geometries with OpenFoam blockMeshDict).

iii. Flow velocity (v) = $\frac{Q}{A}$ = 0.147 m/s

d. For files Ub (location:app/0/Ub) and Uc (location: app/0/Uc), use the same process and value as for file U.

3. Pressure
   - ★ We assume the pressure in the outlet to be zero in order to find the inlet pressure.
   - ★ No change required for pressure
4. Thermal Properties
   a. File location: app/constant/thermalProperties
   b. In lines 17 and 18 of the file, kf and ks represent the thermal conductivity of the fluid and solid, respectively. The solid material is Aluminium, and the fluid is a 50/50 Ethylene Glycol Water Mixture. Change kf to 0.42 W/(mK) and ks to 237 W/(mK).
   c. In line 19, rhoc represents the values obtained from multiplying density (rho) with the specific heat capacity of fluid (c). For a 50/50 Ethylene Glycol Water Mixture, rho is 1065 kg/m^3, and c is 3488 J/(kgK), which gives rhoc = 3714720 J/(K.m^3).
5. Transport Properties
   a. File location: app/constant/ transportProperties
   b. In line 20, nu = $\frac{viscocity}{rho} = \frac{6.9e-3}{1065}$ = 6.47e-6 m^2/s
   c. Calculation for D_normalization value
      i. New geometry is created with the fluid zone, as shown below in figure



      ii. Power dissipation for the above cooling plate geometry is calculated using the MTO program
      iii. The value obtained is 1.23053e-7 kg.m^2/s^3
      iv. Change the D_normalization value to 1.23053e-7 in line 37 of the file.
6. Turbulence Properties
   a. Our flow is laminar. No change is needed.

## Modifications in the src_TF folder

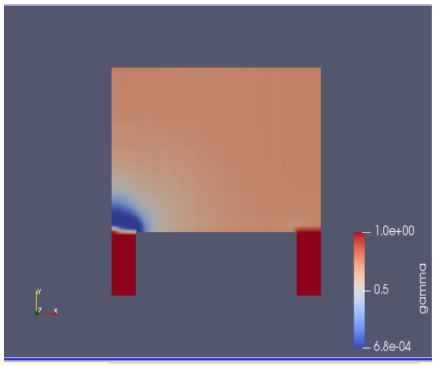
1. Heat Source (Q)
   a. Used in: primal_heat.H and adjoint_heat_T.H
   b. Q = 104.17 W / (437 mm * 284 mm * 2mm) = 419,500 W/m^3
   c. In line 74 of the file readThermalProperties.H, write Q/rhoc = 419500 / 3714720
   d. We set the value to 419500.0. This is specified on the line as follows:
      ```
      dimensionedScalar("Q",
      dimensionSet(0,0,-1,1,0,0,0),419500.0/3714720)
      ```

## Experimenting with Values

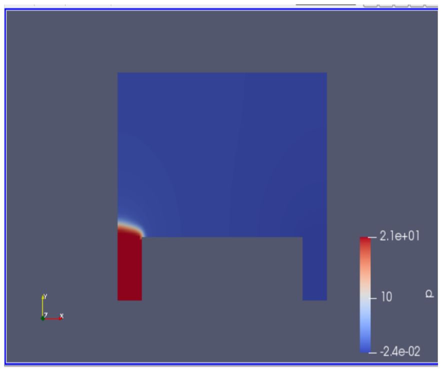
### Heat Source (Q)

**Location**:

We ran 5 MTO simulations using the cooling plate design space and boundary conditions specified with varying values of Q. The values of Q used were 1e9, 1e8, 1e7, 1e6, and 419,500 (the value that our team determined based on our conversions of values specified by the sponsor). None of the simulations ran to completion. All of the iterations crashed after 42 iterations due to a floating point error in the adj flow calculation. All of these crashes appeared to be due to the inlet getting clogged and pressure building up.



**Gamma plot of final iteration of simulation with Q=419,500 before crashing**



**P plot of final iteration of simulation with Q=419,500 before crashing**

All of the designs generated appeared similar as the design space was mostly uniform density between solid and empty and the inlet was blocked preventing the flow of the fluid. Since this appears in all of the simulations run it seems that the issue of the inlet becoming clogged is due to other errors in the simulation. Q did not seem to have an effect on the designs generated but this is hard to assess since all of the simulations were cut off very early. Q clearly had an effect on meanT values as seen below.

| **Q** | 1e9 | 1e8 | 1e7 | 1e6 | 419,500 |
|---|---|---|---|---|---|
| **Final MeanT** | 3670 | 371 | 332 | 301 | 299 |

| **Q** | 1e7 | 7.5e6 | 5e6 | 2.5e6 | 1e6 |
|---|---|---|---|---|---|
| **Final MeanT** | 332 | 324 | 315 | 307 | 301 |

## Voluse

**Location**: app/constant/transportProperties

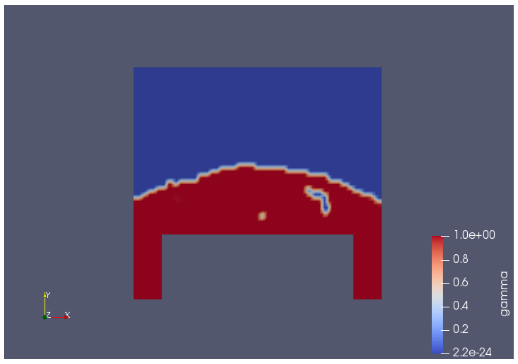
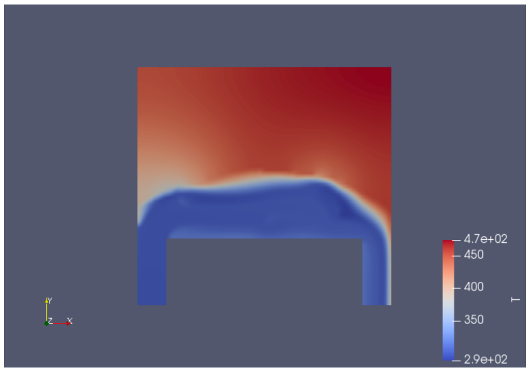
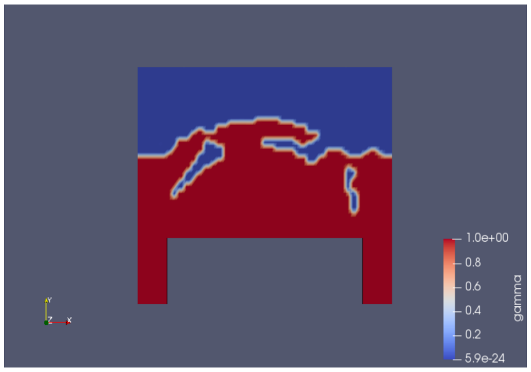
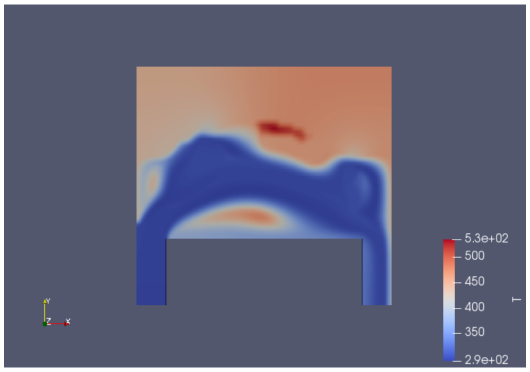
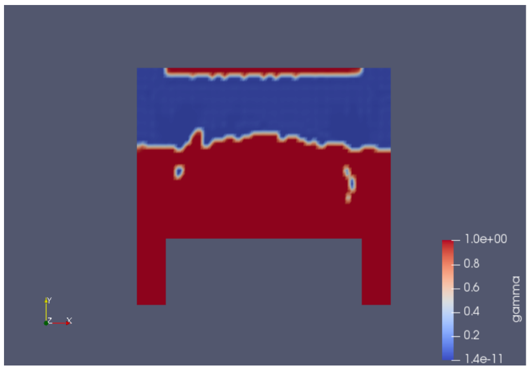
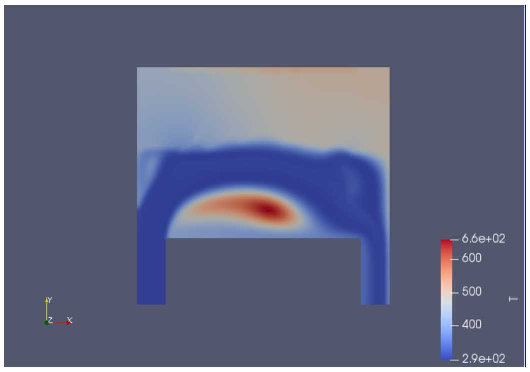
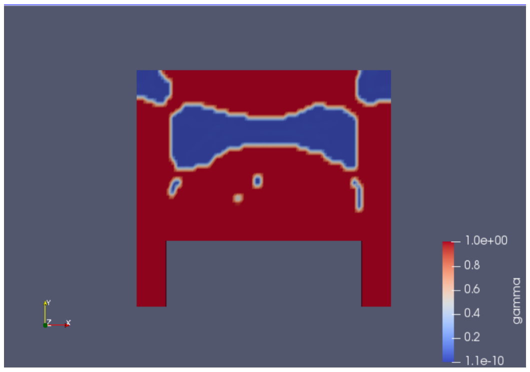
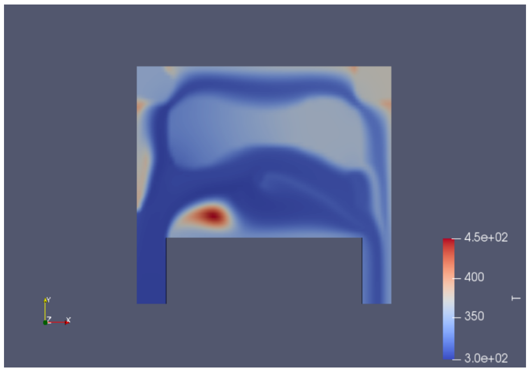
Voluse is the proportion of the domain that is set to be fluid. For example, a voluse of 0.4 will produce a geometry where 40% of the domain is fluid and 60% is solid. The following table shows the final geometry and temperature field that resulted from using different values for voluse.

The voluse did not run to completion when velocity of the coolant at the inlet was set to 0.147 m/s. The solver was run with voluse set to 0.4, 0.6. 0.7, and 0.8. The most iterations achieved was 22. As a result, the solver was run with the velocity set to 0.0147 m/s to better observe the impact of voluse on the resulting geometry.

| voluse | number of iterations achieved before Floating Point Exception error in PETSC |
|---|---|
| 0.4 | 37 |
| 0.6 | 41 |
| 0.7 | 41 |
| 0.8 | 41 |

Note: γ = gamma = design variable bounds = fluid/solid

(The following results were run with U, Ub, and Uc = 0.0147 to better see the impact of changing voluse.)

| Voluse | gamma | T |
|---|---|---|
| 0.4 |  |  |
| 0.6 |  |  |
| 0.7 |  |  |
| 0.8 |  |  |

## D1 (or J^bar in Yu. et al 2019)

**Location**: app/constant/transportProperties

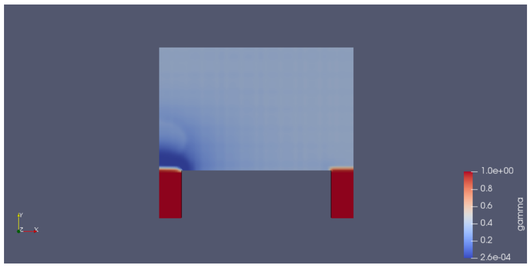
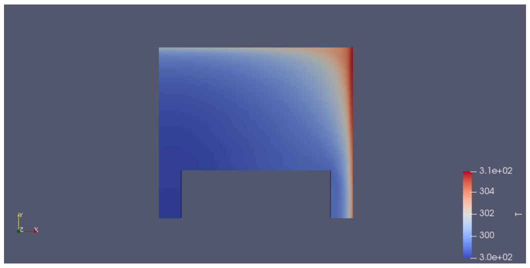
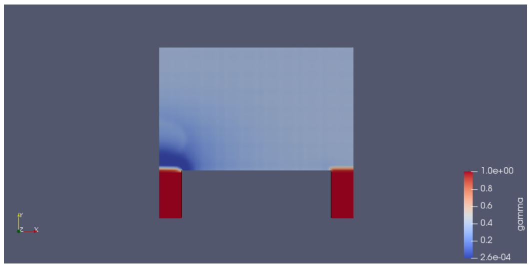
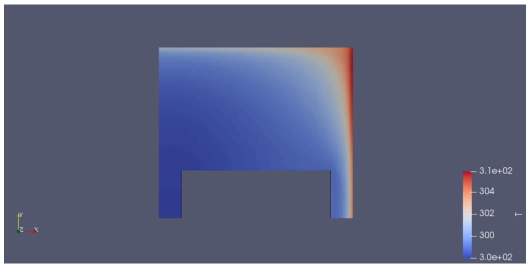
J in this case refers to the power dissipated by the fluid device, and the optimization algorithm is subject to J < J^bar, where J^bar is the upper bound of power dissipation. Changing the J^bar value lets us determine the maximum power dissipated in the fluid module.

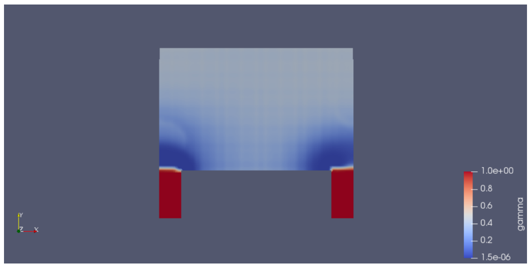
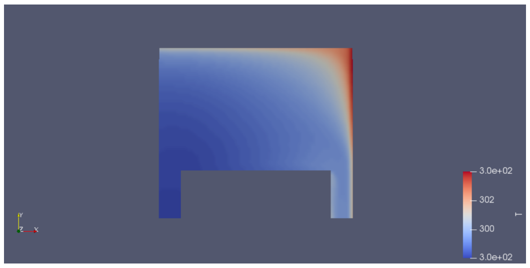
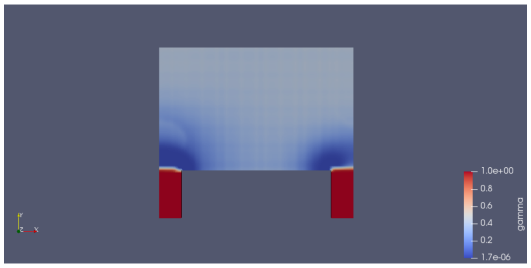
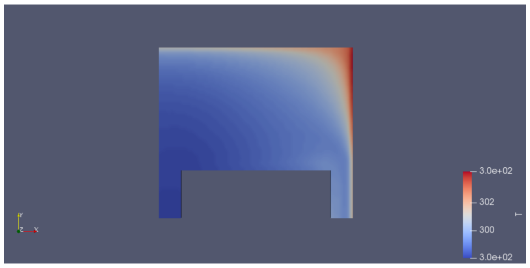
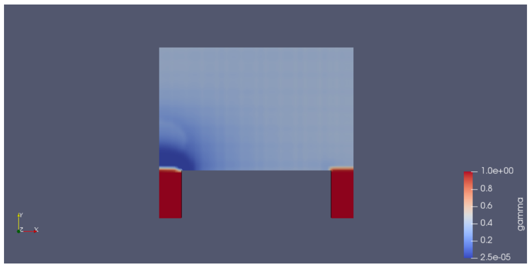
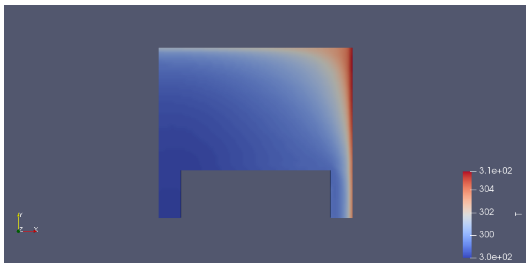
J in this case refers to the power dissipated by the fluid device, and the optimization algorithm is subject to J < J^bar, where J^bar is the upper bound of power dissipation. Changing the J^bar value lets us determine the maximum power dissipated in the fluid module.

For D1 = 10, 40, MTO_TF crashed on the 38th iteration due to a floating point error in the adj flow calculation, presumably by heat and pressure build up. However, for D1 = 80, 120, 200, MTO_TF crashed on the 42nd iteration, presumably for the same reason mentioned for D1 = 10, 40. The reason for the difference between the crash 5 iterations between different D1 values is unknown. The following table shows the final geometry and temperature field that resulted from using different values for D1. From D = 10, D = 40, and so on, the final meanT goes from 299.46 K to 299.24 K.

Note: γ = gamma = design variable bounds = fluid/solid

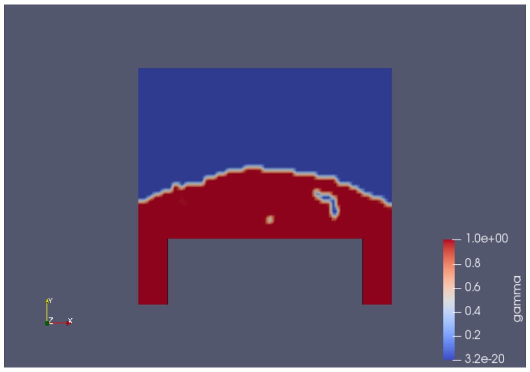
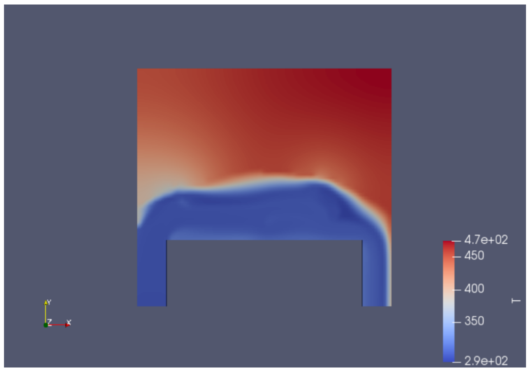
(The following results were run with U, Ub, and Uc = 0.147)

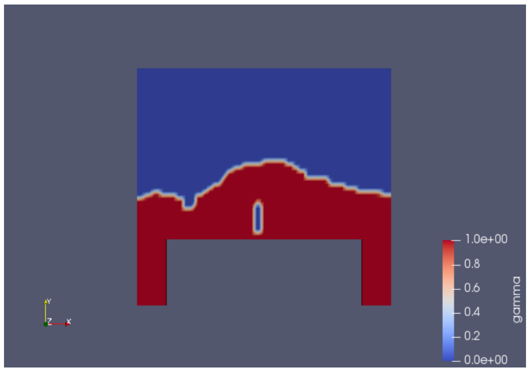
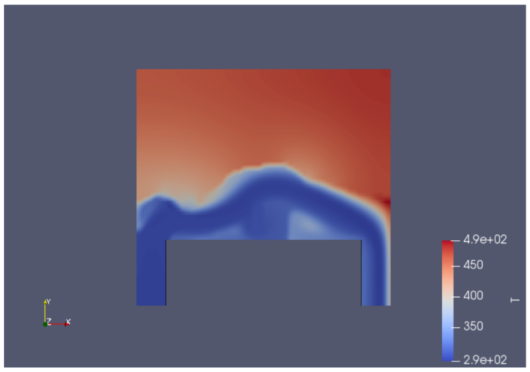
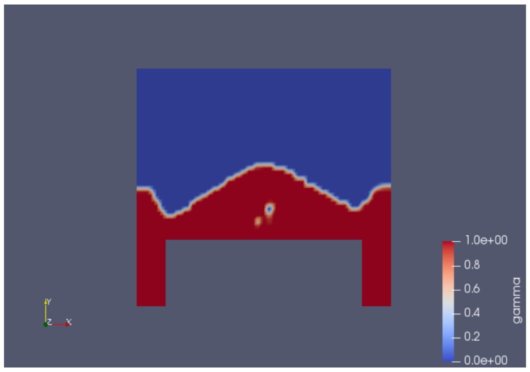
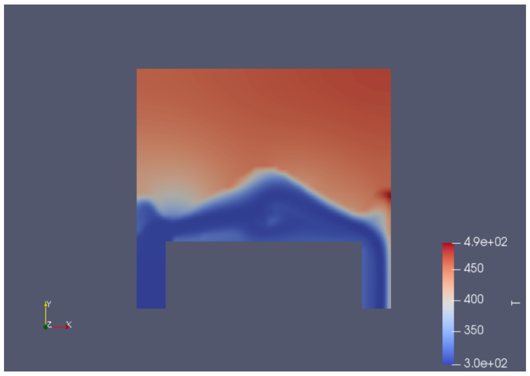
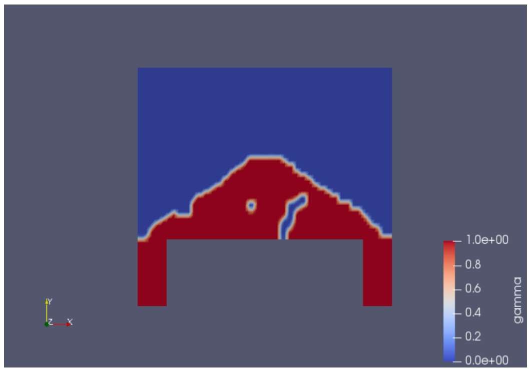
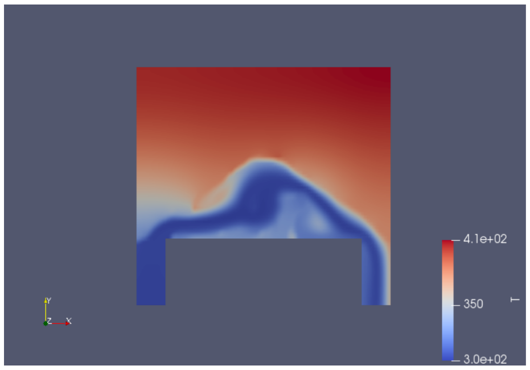
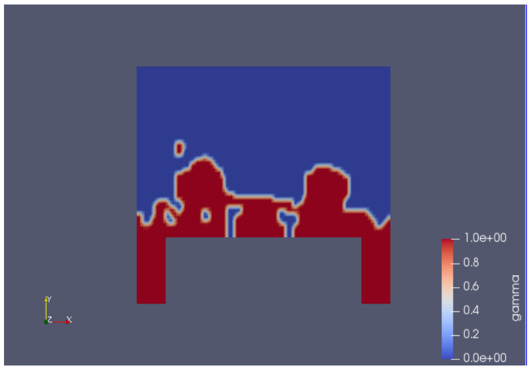
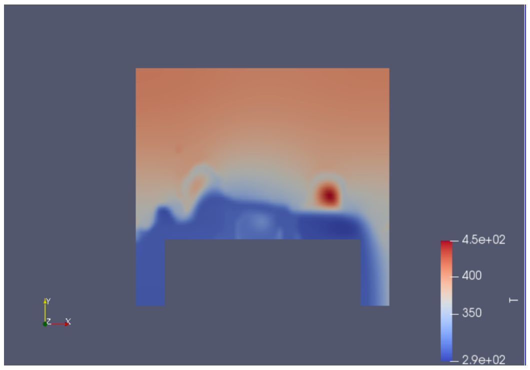
| D1 | gamma | T |
|---|---|---|
| 10 |  |  |
| 40 |  |  |

| D1 | gamma | T |
|---|---|---|
| 80 |  |  |
| 120 |  |  |
| 200 |  |  |

Note: γ = gamma = design variable bounds = fluid/solid

(The following results were run with U, Ub, and Uc = 0.0147 to better see the impact of changing D1.)

| D1 | gamma | T |
|---|---|---|
| 10 |  |  |

| D1 | gamma | T |
|---|---|---|
| 40 |  |  |
| 80 |  |  |
| 120 |  |  |
| 200 |  |  |