Final Report: Generative Design System Development for Cooling Systems

General Motors

Yigit Can Cevikol

Sponsor Mentor: Dr. Erik Yen
Faculty Mentor: Professor Gregory Hulbert

Student Team Members: Sanjay Bharati, Yigit Cevikol, William Fielding, Ha Young Kim, Benjamin Liu, Garima Shah

December 16, 2022

INTRODUCTION

A significant challenge in electric vehicle battery lifespan and safety is in keeping the battery at optimal temperatures. To cool the battery, a coolant is pumped through a cooling plate across the battery's surface. The cooling plate geometry determines the path the coolant will follow, which determines how well the coolant regulates the battery temperature as well as the amount of energy required to push the coolant through the plate.

Value to the Sponsor

Traditionally, the cooling plate geometry is designed by applying topology optimization to a human-designed geometry. This limits the geometry to the creativity of human engineers and creates the bottleneck of manually developing the initial geometry. Hence, GM favors generative design, which bypasses the initial design by applying topology optimization to an empty domain.

In the past, GM obtained cooling plate geometries produced by generative design from a third-party generative design software company. GM wants to pioneer their own path in generative design cooling plate geometries, experiment with feeding the generative design engine, and save on the additional cost of purchasing designs from a third-party. To do this, GM wants to build their own generative design engine, and they're asking our team to take the first step by creating each of the sub-engines of the generative design process.

Objectives / Scope

Our sub-engines will be designed to generate an optimized 2D cooling plate geometry for GM's single battery module conditions. Our objectives will be limited to constraining the maximum temperature to under 40 degrees celsius and minimizing temperature deviation and pressure drop. We will use dexcool as the coolant substance, design for a single module, 2D cooling plate, and restrict volumetric flow rate, inlet and ambient temperatures, heat dissipation rate, and dimensions to values specified by our GM sponsor mentor . We will perform thermal-fluid analysis on the geometry and not perform producibility analysis.

Literature review

In addition to gathering literature regarding the generative design and the whole end to end design process, our sponsor has explicitly requested this documentation requirement to serve as a guide for someone to follow to continue from where we have left off and as a reference for explaining our progress and findings.

The generative design process is to give guiding parameters to a design space, then producing novel geometries with no human preconceptions and add onto the world of topology optimization. Generative design makes it easier for designers to reduce the size of an object without compromising its function. It also yields sustainability advantages, reducing material use

and environmental impact while improving performance and lowering costs, which complies to General Motors' company goals.

Deliverables

By December of 2022, GM can expect the MDP Team to have constructed three sub engines from the open source Multiphysics Topology Optimization (MTO) program. The MTO program is a design generation algorithm with a 2D thermal-fluid example problem like the one we described above, but with different battery conditions, constraints, and optimization parameters. The output geometry will be in a CAD file representation. We will not produce a physical prototype. Successful modification of the program to design for the GM single battery module conditions will demonstrate our understanding of the generative design process.

Solution Strategy

The MTO program is an implementation of Finite Element Analysis (FEA). FEA is a method to solve differential equations by representing a system as discrete elements represented by systems of equations which represent the physics properties of the elements and solving these equations at each element to represent the system as a whole. MTO's FEA uses partial differential equations from the Navier-Stokes equations to represent fluid flow in the system and energy conservation equations of thermodynamics to represent convective heat transfer.

To optimize the cooling plate design we designed an objective function that I personal wrote for the program to minimize. The program solves the Navier-Stokes equations to calculate the flow of the coolant through the design space and uses the fluid flow to solve the heat trasfer system of equations. Then it computes derivatives of the fluid flow and thermal systems of equations, which will be used to determine how the design space should be altered. MTO uses a parallel implementation of an algorithm known as Method of Moving Asymptotes (MMA). MMA uses the systems of equations at each element in the system and their first and second-order derivatives to calculate how the density of each element should be altered in order to minimize the objective function defined.

After optimizing the design of the cooling plate the program will check for convergence (if the difference between the current design and previous design is smaller than a given criteria). If the system has converged, the program will stop. Otherwise, it will repeat the cycle and check for convergence again.

Final Status

The team has delivered a design geometry, an example fluid flow analysis problem solved by the example software, and a commercial software optimization problem to GM so that they can

use the different sub-engines mentioned above in their goal of creating their own generative design for their battery cooling plates.

The sponsor monitor from GM has accepted the deliverable and stated that the delivered files were “above satisfactory” based on the purpose of the GM group within the MDP discipline. All of the delivered files will be delivered using the open-source Multiphysics Topology Optimization module and COMSOL Multiphysics.

OUTLINE OF THE QUESTIONS/PROBLEM ADDRESSED

As mentioned in the Introduction, the group was tasked with creating a cooling plate design that would optimally cool the cooling plate battery.

Here were some of the Requirements we were asked to satisfy in regards to this design:

Requirement #1: Single Battery Module Conditions

The Single Battery Module Conditions requirement has three subgroups:

1.1) Maximum battery surface temperature (T_{max})
1.2) Absolute difference between maximum and minimum battery surface temperatures (ΔT)
1.3) Pressure difference of coolant between inlet and outlet ($\triangle P$)

In order to meet the given requirement The team has prepared our validation method by creating a CAD geometry of a single battery module in Autodesk Fusion 360 (shown in **Figure 1**) and performing the thermal and fluid flow analysis using COMSOL Multiphysics on a module with the sample flow path shown in **Figure 2**. This battery module has the same dimensions and battery conditions as the GM battery module.

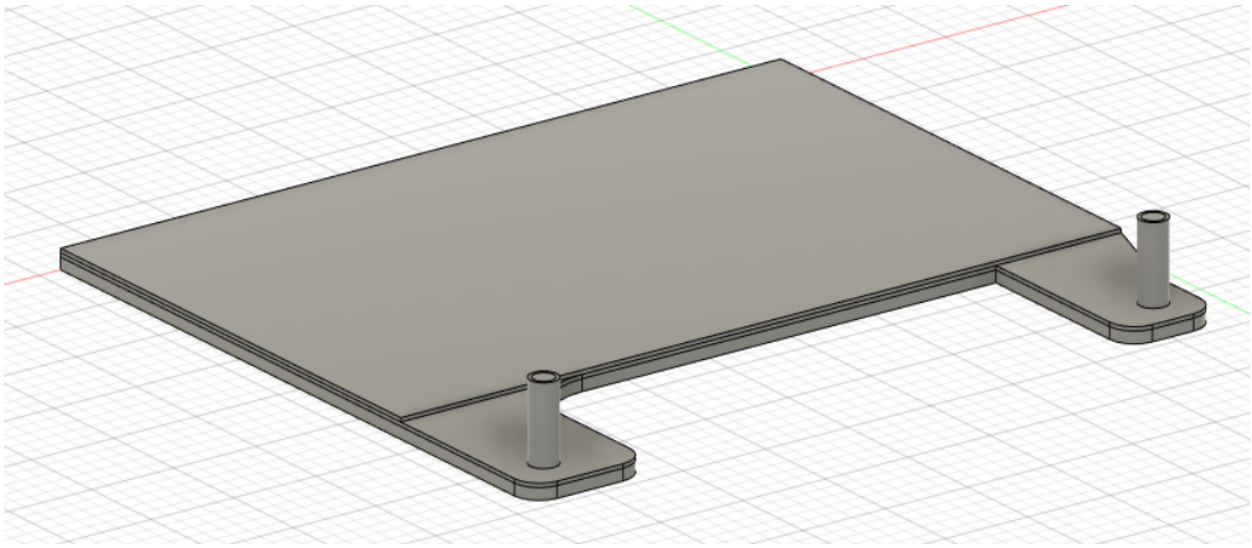


Figure 1: Sample Cooling Plate for One Battery Module

Requirement #2: Progress Documentation

Below is a detailed documentation of the end-to-end process that the GM team intended to create. To explain it further,(**Figure 2**) the flow starts with a literature review, then we move our knowledge into a Primal Solver, which would be representing the design generation phase. Here, we test the design space with distinct example designs to see if we can take advantage of any example reference designs. The adjoint solver is where we would feed this example geometry design into a fluid analysis base to possibly get a fluid flow diagram implemented on the provided geometry. Then, this geometry would be passed to the optimization solver to see if there is a feasible (possibly optimal) solution. In order to optimize the solution even better, the worst-performing parameters would be readjusted in the primal solver until the result converges optimally based on our objective function.

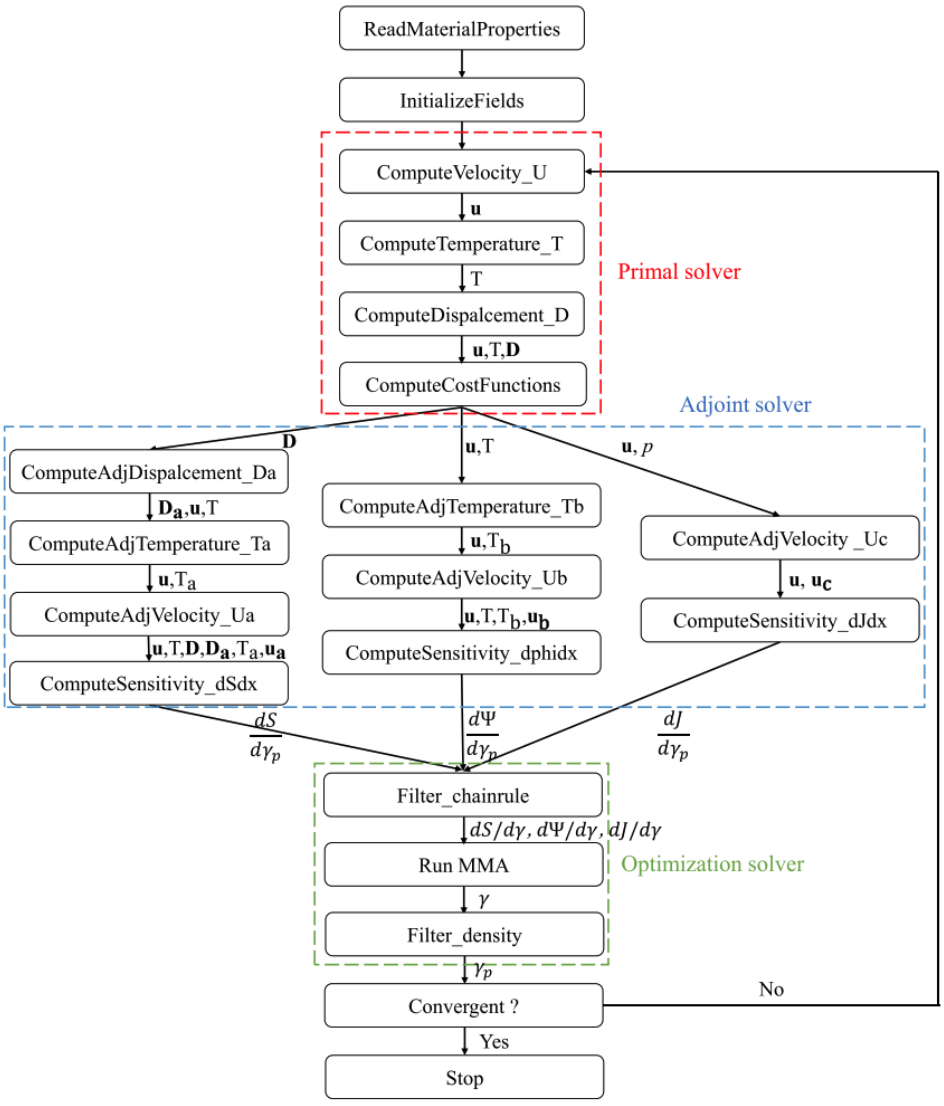


Figure 2. A view of detailed end to end process

METHODS

The main methods used in the project was based on distinct solvers coming out of an opensource software called MTO using multiple solvers, the primal solver specifically computes the state of the current design, for example, the Navier-Stokes equation, which captures fluid motion from properties such as fluid velocity, density, and pressure. The primal solver is mainly composed of code and files from OpenFOAM, and perform steps as following:

1. simpleFoam solving the Navier-Stokes equation
2. Using the velocity obtained from simpleFoam, scalarTransportFoam is used to solve the energy conservation equation, or to find the differences of pressure, velocity, and temperature of the fluid inside the module
3. The obtained field temperature from scalarTransportFoam is then employed to estimate the thermal expansion in the linear elasticity equation, or to find the additional forces within the fluid by thermal expansion, which is computed by solidDisplacementFoam

Each of the following “Foam” extensions from the term indicates that the code or file came from OpenFoam.

Next, the adjoint solver would take variables such as velocity, density, pressure, and density from the primal solver and perform adjoint methods to solve the gradient of the module. As shown in **Figure 2**, the adjoint solver branches off to 3 different equations, they are Navier-Stokes, energy conservation, and linear elasticity equation, which are mentioned in the primal solver. The only difference between the adjoint and the primal solver is that the adjoint solver is an optimized version of the primal solver where the adjoint solver is efficient in computing the model gradient, which shows the differences of fluid pressure and temperature throughout the whole module. It is also worth mentioning that the adjoint solver adopts the adjoint state method, which has the property where the number of computations is independent of the number of parameters of the gradient, hence the adjoint method is more efficient compared to the brute force method where each part of the gradient is repeatedly calculated. For the computing sensitivity block in **Figure 2**, that is to calculate how the velocity field responds to perturbations, which is related to how the gradient would be calculated.

Then, the gradient variables get fed into the optimization solver, where it utilizes the MMA method, and a filter that primarily prevents numerical instabilities for large-scale topology problems. The MMA is where the optimization happens, the MMA method uses upper and lower moving asymptotes to adjust the curvature of the approximate functions, in other words, change the curvatures and solid within the design space within a certain range based on the gradient values from the adjoint solver.

The literature review on both topics explains the majority of how the generative process works and how it can be achieved. The literature provides the basis of the GM MDP team’s work and presents the generative design and the end to end process to our faculty. Without it, the project loses its purpose.

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 Ψ 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
μ and λ = 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
Ψ = psi = objective function

Documented steps of MTO

Another part of our progress documentation is by documenting the steps to set up MTO, the GM MDP team can now reproduce the MTO set up and provide information on how to set the design space for a generative design. This will be specifically beneficial to the company General Motors in harvesting the research result. The functionality of this deliverable is to provide the sponsor, or General Motors, the necessary information to understand the components of MTO; this deliverable will also provide the necessary details to reproduce the set up of MTO, which would allow the sponsor to have access to the team's research algorithm.

Below is a screenshot of the documentation on the MTO Github Structure, where the team detailed information regarding the files and folders. For clarification purposes, “FoamFiles” in **Figure 3** refers to one of the dependencies of MTO, OpenFOAM, which stands for Open-source Field Operation And Manipulation. These “FoamFiles” are used by OpenFOAM to calculate various variables.

MTO Github Structure

1. MMA
    a. MMA.c
    b. MMA.h
2. MTO
    a. 2Dheatsink
        i. app
            1. 0 ← folder contains FoamFiles, all I/O format version 2.0 in ascii with class volScalarField; different dimensions of field and boundary field
            2. constant ← all FoamFiles
                a. thermalProperties
                b. transportProperties
                c. turbulenceProperties
            3. system ← all FoamFiles
                a. blockMeshDict
                b. controlDict
                c. decomposeParDict
                d. fvSchemes
                e. fvSolution
        ii. src_TF
            1. Make
            2. adjointOutletPressureHeat ← FoamFiles
            3. adjoinOutletPressurePower ← FoamFiles
            4. adjointOutletVelocityHeat ← FoamFiles
            5. adjointOutletVelocityPower ← FoamFiles
            6. Heaviside_rho.H

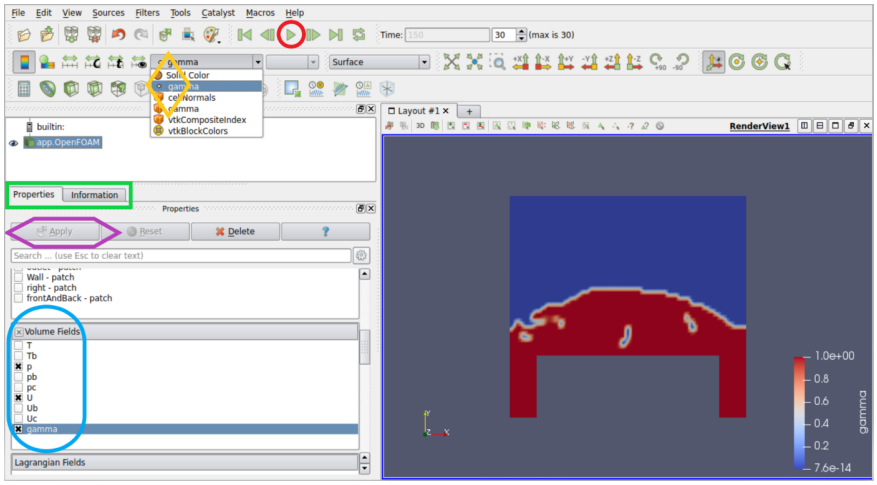
Figure 3. MTO GitHub Structure Document Sample

The challenge for MDP is setting up dependencies and compiling dynamic libraries in order to run MTO. The difficulties arise from lack of online documentation and outdated documentation. For example, most software dependencies of MTO are old. This means that the documentation of that particular software in an older version is difficult to find. In addition to the previous problem, although the team can find supporting documentation for newer versions of dependencies, the team is uncertain whether the source is reliable, meaning following newer online documentation could not work, or potentially hurt our progress by modifying something we shouldn't have. As a result, it is time-consuming to proceed with the setup and configurations, which leads the team to consume a decent amount of time to finish the entire process setup.

The reason why we include our literature review and MTO setup documentation in this requirement is so that the sponsor is able to harvest our learnings and hand them off to others to take over. This requirement is very similar to a "knowledge transfer" in a corporate setting under General Motors' policy. Hence why our sponsor strongly values this requirement.

Displaying the results 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.)



Defining the Cooling Plate Geometry

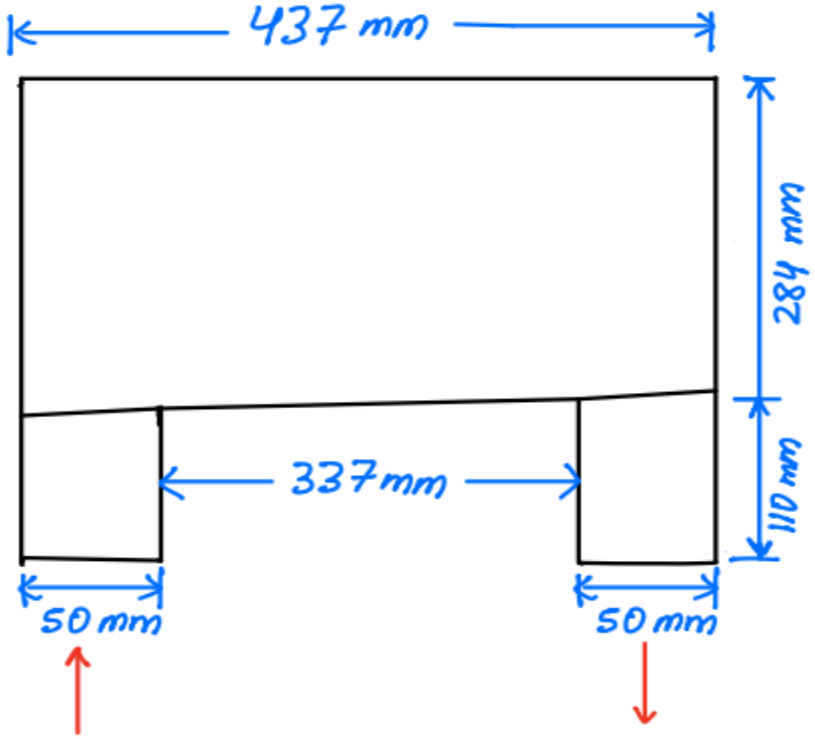


Figure 4

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.

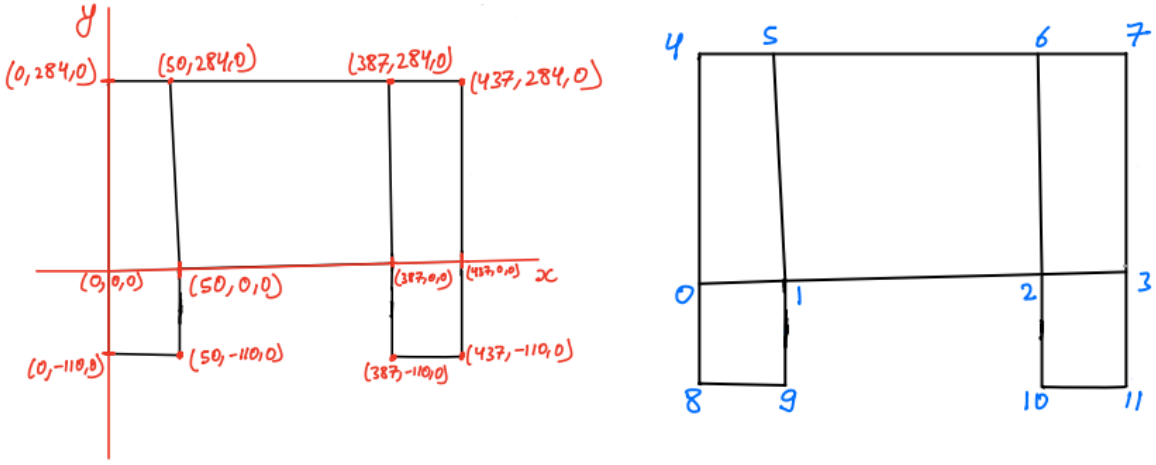


Figure 5

The picture on the right of **Fig.5** 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.

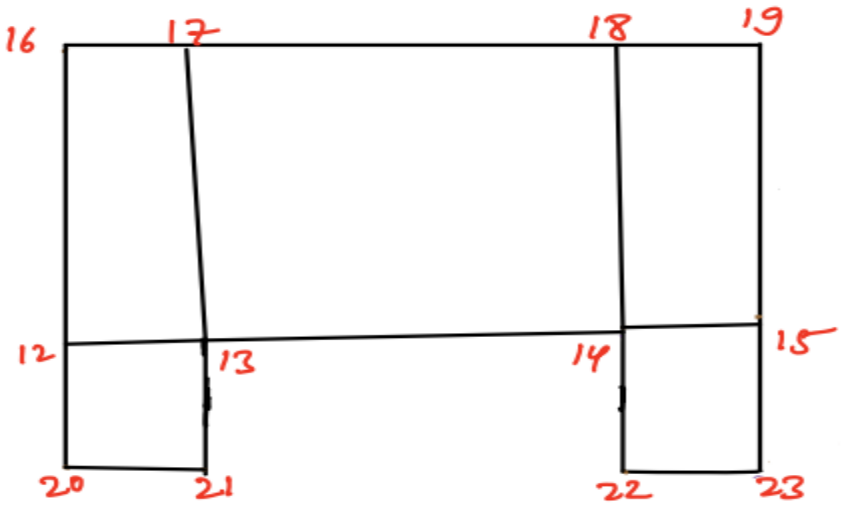


Figure 6

Similarly, the back plane of the cooling plate geometry is constructed, as shown in **Fig.6,** 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.

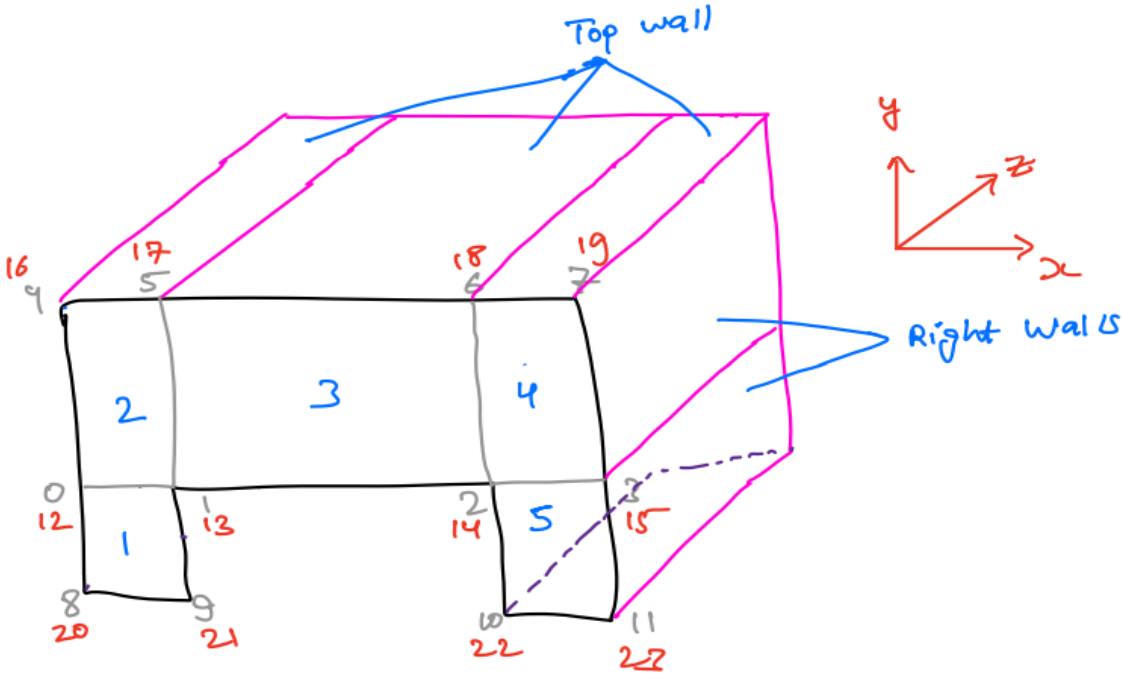


Figure 7

Figure 7 is the cooling plate geometry in 3D.

Steps to Generate Geometry in OpenFOAM

1. Write all twelve vertices of the front plane (**Figure 4)** starting from 0 and in numerical order.
2. Similarly, write all twelve vertices of the back plane (**Figure 5).**
3. Divide the **Figure 6** 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.

RESULTS

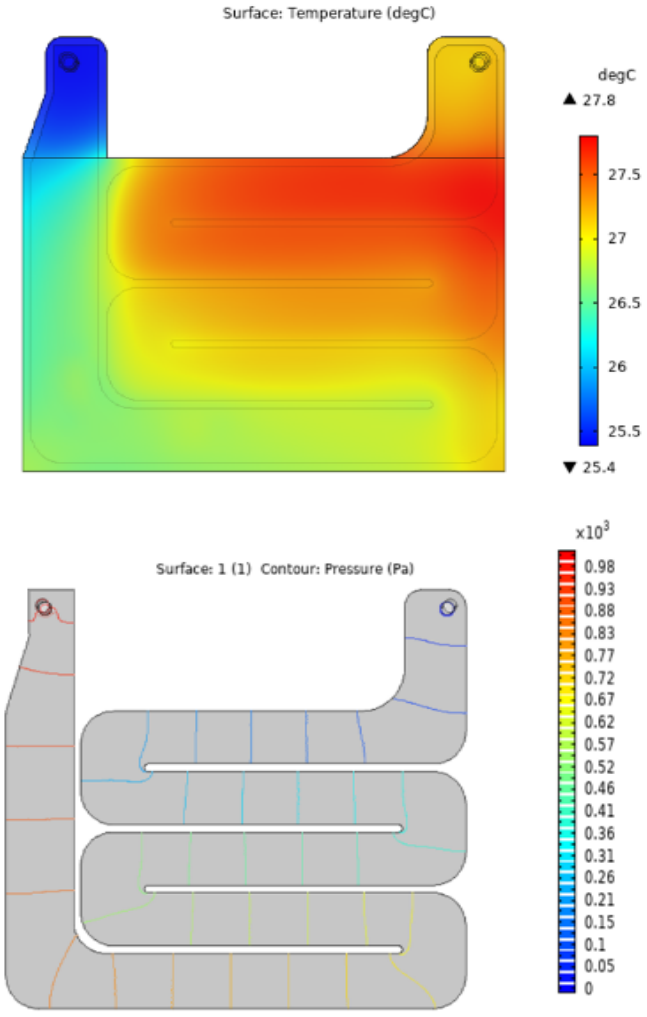


Figure 8. (a) Battery Surface Temperature Distribution; **(b)**Pressure Distribution for Normal mesh

Figure 8 above represents the example we ran using COMSOL multiphysics. Here were some of the findings we got for our example run:

T_{max} **= 27.8˚C ≤ 40˚C**
ΔT **= 27.8 - 25.4 = 2.4˚C ≤ 2.4˚C**
ΔP **= 980 Pa ≤ 1 kPa**

According to these findings, all of our constraints were met from the initial design, however, the error margin on this design was a lot larger than what we intended to have. We ended up with a %11 error rate, which is a lot more than what would be allowable on a vehicle that drives with

living beings inside. In the case of heating, the battery could melt, or even worse, the car could catch fire.

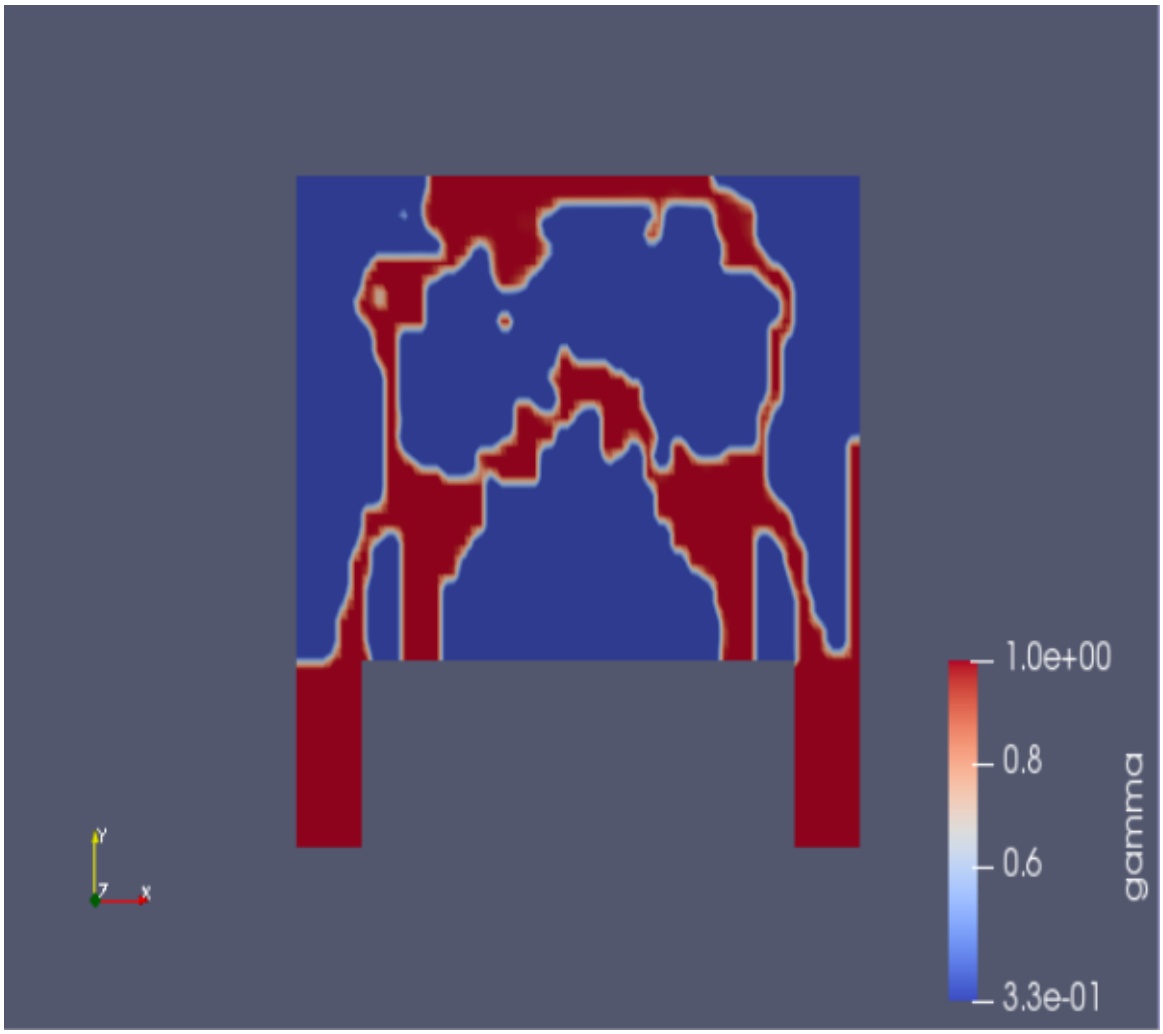


Figure 9: The Final Design

The figure above represents the final design that the GM MDP team was able to produce from the MTO software. In this design, the red regions represent where the fluid would be flowing and the blue regions would represent the solid that is supposed to be in the battery module. The gamma in the legends represents the density of the specific region.

The final design produced was not able to perfectly meet the requirements of General Motors, due to the miscommunications of the sub-engines we created. While the design generation and the fluid analysis were done in the MTO software, we were not able to adjust the optimization module as desired, which was the reason that we used COMSOL Multiphysics to work with external optimization packages.

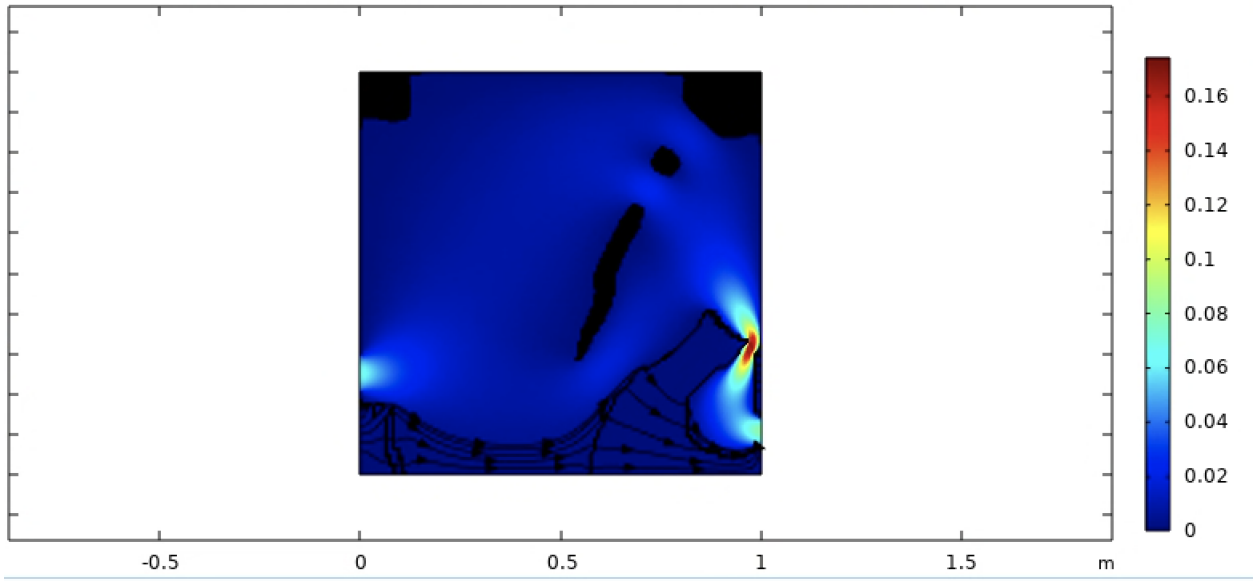


Figure 10. COMSOL Geometry

The figure represents the velocity field of the coolant and how the generative design generated a path for the fluid for the defined inlet and outlet positions. The tightening through the end indicates changes in pressure and coolant velocity within the system. The model needs the correct detection method to calculate the temperature and pressure differences along the base distribution. The goal is to modify the current model so that the optimization solver initially solves for minimizing the average temperature across the surface while minimizing the pressure drop. The next step is to introduce weight parameters x and y to observe how the weights of pressure and temperature drops change the optimal design that would come out of COMSOL. Based on how the results evolve, the values for the weight variables will be adjusted as a constraint. Then the solver should be run for at least 1000 iterations to approach a mostly optimal solution. If the runtime is over 2 hours, the mesh could be reduced to a more straightforward solution.

The final geometry produced in COMSOL was not the geometry we worked with in the MTO software since the team could not extract the geometry file from MTO into a file format that COMSOL could read in. In order to utilize the distinct sub-engines, we have produced two distinct guides to allow GM to implement our sub-engines within their system.

DISCUSSION & CONCLUSIONS

The GM MDP team was able to do a literature review. We analyzed different processes to approach the given problem of creating an optimal cooling plate for the 2027 GM Electric Vehicle Batteries.

The team decided that the optimal approach to the given problem was to use the MTO open-source topology optimization package to create a design geometry, create possible fluid paths, and optimize the problem according to the given design constraints. According to the design requirements we were given, we came up with a sample geometry that would satisfy the requirements and defined an optimization function that accounts for the maximum temperature and pressures while minimizing the change percentages of the pressure and the temperature of the geometry.

We were tasked with creating and connecting three sub-engines to complete the project to an end. After realizing that the entire system would not be completed in time, the sponsoring mentor requested us to document the process in detail so that the GM Corporation could pick up the project from where it was left. Hence, we provided 35 pages of documentation on MTO software and 10-page documentation on COMSOL multiphysics software, documenting our approach and step-by-step explanation of what we did to develop the models shown in the methods section.

As deliverables, we produced a sub-optimal design in the predefined design space. We ran fluid analysis through it multiple times, changing specific parameters to observe the impact of the change and adjusting the model accordingly. As a result, we ended up with more than 200 models as generative design iterations to successfully complete the project.

I learned how to design an end-to-end generative model and process that is self-improving. My personal part in the team was mainly on the Optimization subsection, which allowed me to learn more about advanced optimization disciplines in machinery. By working hand to hand with the design generation team, I was able to improve my skills in coding as well as fluid analysis, which are critical skills of an engineer that is working on an automotive product.

Some of the key takeaways from this project was the importance of principles. It was imperative that even though the times were hard due to the complexity of the project, we did not give up. After countless hours of research and trial and error, the project ended up being a success, satisfying all the requirements.