Politecnico di Torino

Master course in Aerospace Engineering

Politecnico di Torino

Master's Degree thesis

Modelling of heat transfer through TPMS, turbulent and laminar flow performance evaluation and code to code benchmark

Supervisors:
Laura Savoldi – Politecnico di Torino
Luca Marocco – Politecnico di Milano
Eleonora Gajetti – Politecnico di Torino

Candidate:
Marco Luca Carbotta

Acadamic year 2022/2023

Contents

List of Figures

List of Tables

Introduction

1.1 Additive Manufacturing

Additive manufacturing, also known as 3D printing is a relatively new type of manufacturing technology that uses a layer-by-layer addition method to produce complex objects. This is a revolutionary approach to manufacturing that made possible the production of complex shapes and revolutionary geometries, allowing the general public to manufacture products at home with commercial 3D printers and companies to manufacture complex parts that couldn't be done via traditional methods, like casting or extrusion. Additive manufacturing has allowed the design and production of topologically optimized components, which are designed to minimize stresses while utilizing as less material as possible, keeping a low weight. While this manufacturing method can still be considered expensive, the advantages are unique, making it the future of engineering manufacturing in many fields that want to achieve high strength and low weights in their products, such as the aerospace and automotive industries. This type of manufacturing can be achieved by different methods [13] such as:

- **Extrusion**, which is the most common method for commercial 3D printers sold to the general public, this generally uses a plastic material, such as PLA, that is melted and extruded by a tractor-feed system through a nozzle. The print head and/or the bed is moved to the correct X/Y/Z positions for placing the material.
- **Powder bed fusion**, which is one of the first commercialized methods for additive manufacturing. This process involves a metallic powder bed that is targeted by a thermal source, usually a laser, in specific parts of the bed so that in only certain regions the powder melts and fuses.
- **Direct energy deposition**, these type of processes enable the creation of parts by melting the material as it is being deposited, this method is also used primarily for metal powders. This method doesn't use a powder bed, instead it melts with a laser the material that is being deposited by the nozzles.

As said before, in the latest years advancements in 3D printing have been made such that complex geometries can be designed and manufactured [22] and this does not only matter for the aforementioned industries that want to keep a low weight. This has also enabled the manufacturing of porous structures for heat transfer, which yield notable performances compared to their traditional counterparts [10].

1.2 Porous heat exchangers

N. Delalic et al. [9] found that porous media in compact heat exchangers had many advantages, finding high power density of the system, extremely good thermal conductivity and high thermal

capacity. If this type of heat exchangers were to be used in everyday home applications, such as boilers or central heating systems would reduce CO_2 emissions and almost nullify NO_x emissions, but still their cost remains the limiting factor. This type of heat exchangers have superior turbulence induction and heat dispersion capabilities which allows them to have better efficiencies in turbulent flows. [6]. The heat transfer capacity of these type of 3d printed porous structures has been studied by F. Alawwa et al, [34] finding that at high flow rates periodic surfaces outperform the traditional fin design. Advancements in the additive manufacturing have allowed the production of complex pure copper elements by laser-based powder bed fusion, wire arc and selective laser melting methods [23, 27, 16], this enables the production of porous structures with the metal.
These kind of heat exchanger use forced convection to achieve higher cooling performance through enhanced flow mixing due to the tortuosity of the passways, and due to metal foams have been defined that can have a surface area to ratio around $1000-3000 m^2/m^3$ [12] and can be manufactured with powder bed methods.

1.2.1 Triply Periodic Minimal Surfaces

A peculiar type of porous structure is a triply periodic minimal surface, a type of structure that is periodic in the 3 dimensions of space. These structures already exist in nature, such as in butterfly wings [18] and given their high surface to area ratio they have a high thermal and mechanical performance [21] [29]. Since TPMS have these characteristics they are currently being studied in many applications, from bone tissue scaffolding [36] to impact structures in the aerospace field [37], finding that depending on the material and type of unit varies the mechanical properties of the geometry: it has been found that for uniform structures a gyroid-type unit can withstand more load and absorb more energy than the primitive type. Different type of TPMS units will be shown in chapter 2.

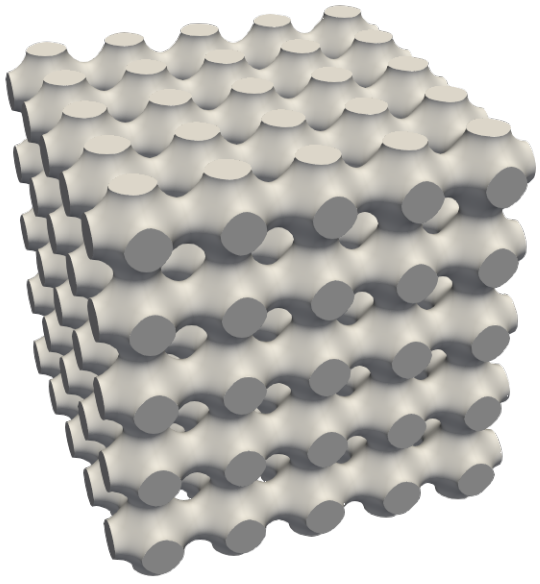


Figure 1.1: Schwartz unit TPMS

The heat transfer performance of TPMS lattices has been studied for some time now, ideally the minimal surface-to-area ratio of the geometries should grant optimal heat transfer capabilities. They have been studied as heatsinks for electronic components [28] and heat exchangers [17] using air and fluid flows, finding that the use of TPMS can reduce their size by up to 60% [29]. Z. Cheng et al. [28] found that under free convection conditions a TPMS-based heat exchanger dissipates 48-61% more energy than a traditional pin-fin heat sink. New TPMS based heat exchangers have been found to have higher overall heat transfer coefficient in relation to traditional plate exchangers, even being two times more efficient with a diamond-type unit [24]. This type of heatsink could have a practical use in nuclear fusion generators as a new generation of cooling divertor tile in the W7-X Stellarator.

1.3 Fusion reactors

Nuclear energy can be achieved through fission, with the splitting of an atom, or through fusion, with the merging of two atoms. The latter process usually generates energy through the merging of a deuterium and a tritium atom, both hydrogen isotopes, the product of the fusion has a smaller mass than the reactant, thus converting that mass difference into energy. Since this process is achieved through hydrogen isotopes there is a practically limitless supply of fuel and no risk of radioactive waste or incidents with fusion reactors given that the reaction is not self-sustaining. The last advantage listed is also the main problem with the process, sustaining a fusion reaction is an extremely complex endeavor given the temperatures involved and the need to control the plasma inside the reactor. Tokamak-type reactors utilize a magnetic confinement fusion [11], where the charged particles are confined with a magnetic cage. These reactors have a relatively simple toroidal shape that achieves a high confinement time for energy and charged particles, but this shape has its limits, causing instabilities in the plasma current, this limits the operation of tokamaks to being achieved only in pulse conditions.
Another type of reactor is the Stellarator, which uses a complex shape to confine the plasma on an external magnetic field coil. The concept was devised in the 1950s [2] by Lyman Spitzer. This type of confinement makes it so that the operation of the reactor can be continuous, and while Tokamaks are still the most common type of reactor, Stellarator operation has been made possible by advancements in computing power that have allowed for the optimization of magnetic configuration [26].

1.3.1 Wendelstein 7-X Stellarator

The Wendelstein 7-X is the world's largest stellarator-type nuclear fusion reactor built by the Max-Planck-Institut für Plasmaphysik in Greifswald in 2014. The W7-X uses magnetic fields to contain the hot plasma needed to sustain a controlled nuclear fusion reaction. This magnetic field is produced by 50 non-planar superconducting magnetic coils that are cooled with liquid helium to superconduction temperatures.

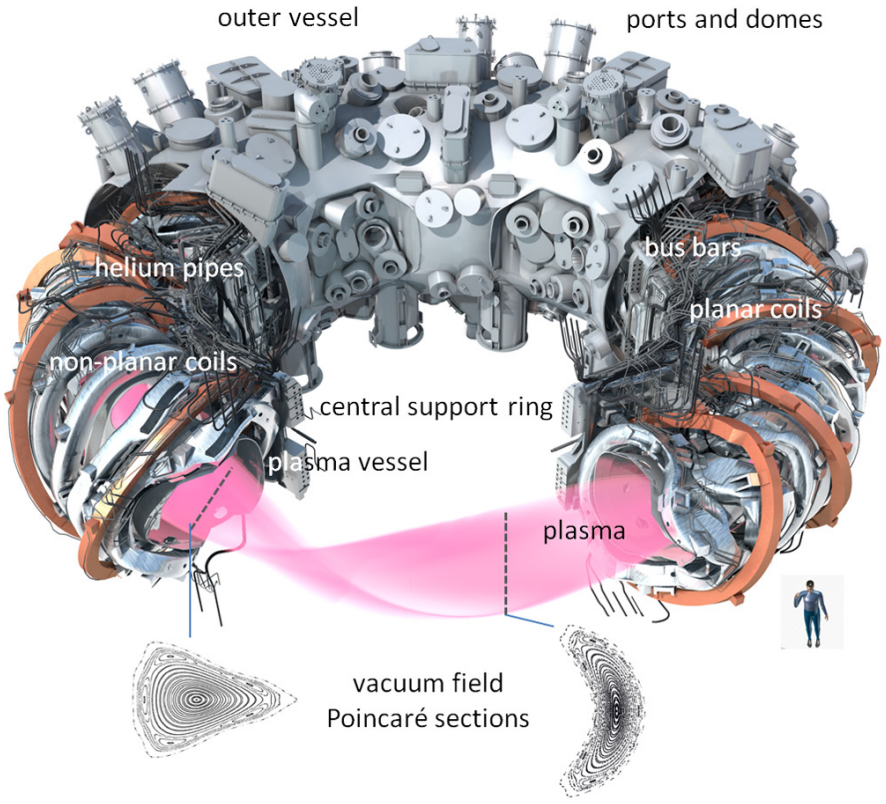


Figure 1.2: Schematic diagram of Wendlestein 7-X Stellarator [15]

The divertor is a component of a nuclear reactor that carries out different critical functions, one of which is to remove the heat produced by particle bombardment, radiation and volumetric nuclear heating [33].

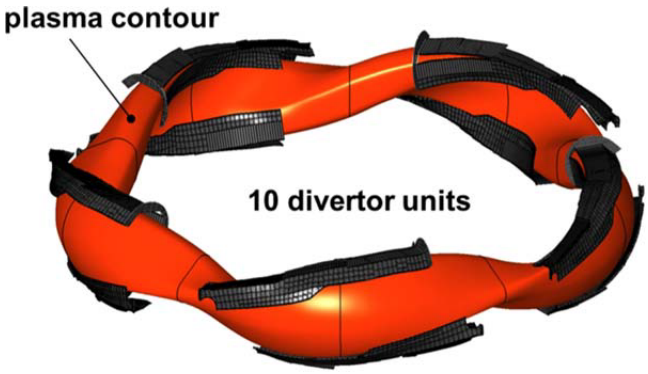


Figure 1.3: W7-X Island divertors [19]

This component requires water-cooled tile elements to sustain the high thermal loads given by the application.

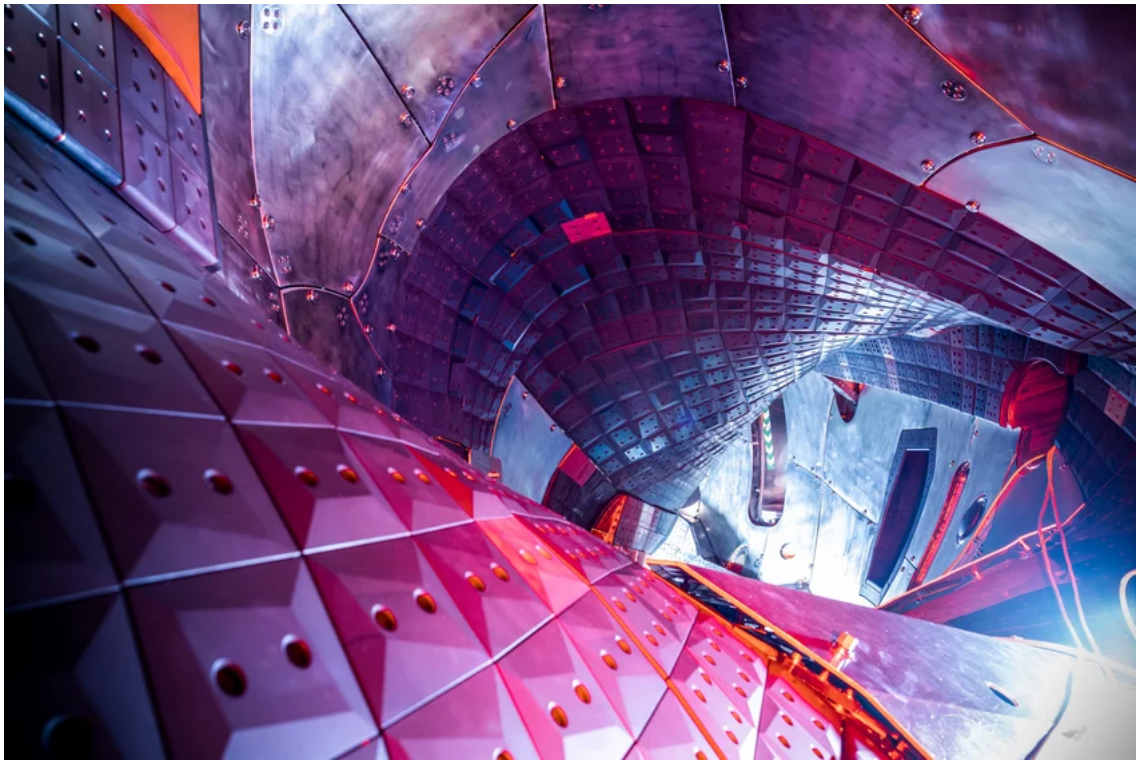


Figure 1.4: Graphite tiles inside the divertor [39]

Currently produced tiles are being designed to remove stationary heat fluxes of up to $10 MW/m^2$ [25] and to allow pulse lengths of up to 30 minutes. New, smaller size elements are also being studied, using copper for the tile main body and tungsten for the cooling surface [35].

1.4 Aim of the thesis

The listed characteristics of TPMS based heat dissipation elements make them a good candidate as a divertor tile element. So a new TPMS geometry has been devised, with a water cooled copper TPMS lattice and a Plasma-facing tungsten tile. A performance study has been conducted with OpenCFD's open source software: OpenFOAM. This software has many advantages, with the main one being that it is an open-source codebase that allows us to better understand its inner workings and it also makes it interesting for future development of a code pipeline using other open-source softwares such as MaSMaker [32] to build TPMS geometries and integrate the two codes togheter. For the thesis a commercial software, nTopology, has been used to generate the TPMS lattice geometry. Another aim of the thesis is to evaluate the performance of OpenFOAM compared to more streamlined commercial meshers. A code to code benchmark has been included comparing the latter with Siemens STAR-CCM+ and Ansys Fluent.

Triply Periodic Minimal Surfaces

Triply periodic minimal surfaces are geometries that, as the name suggests, are periodic in the three main directions in space. They were first devised in the 1800s by Schwarz and the concept was later expanded on by his student Neovious [1]
These surfaces are periodic and implicit with zero mean curvature, so that they can be expressed with mathematical equations and just by modifying parameters within these equations we can modify these geometries.

Since TPMS are implicit surfaces their geometry can be expressed with algebraic equations where

$$f(x, y, z) = C$$

There are two methods to express TPMS:

- Enneper-Weierstrass parametrical representation:

$$\begin{cases} x = \Re\left(e^{i\theta}\int_{\omega_0}^{\omega}(1-\tau^2)R(\tau)d\tau\right) \\ y = \Re\left(e^{i\theta}\int_{\omega_0}^{\omega}i(1+\tau^2)R(\tau)d\tau\right) \\ z = \Re\left(e^{i\theta}\int_{\omega_0}^{\omega}2\tau R(\tau)d\tau\right) \end{cases}$$

  Where $i^2 = -1$, τ is a complex variable, θ is the Bonnet angle and $\Re$ is the real part of a complex variable. $R(\tau)$ is the Weirestrass function for different kinds of TPMS units, for the Gyroid, Primitive and Diamond surface its value is:

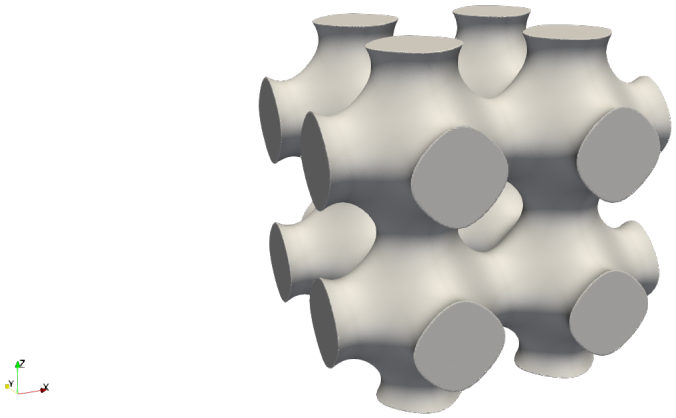
$$R(\tau) = \frac{1}{\sqrt{\tau^8 - 14\tau^4 + 1}}$$

- TPMS can also be generated by the following equation:

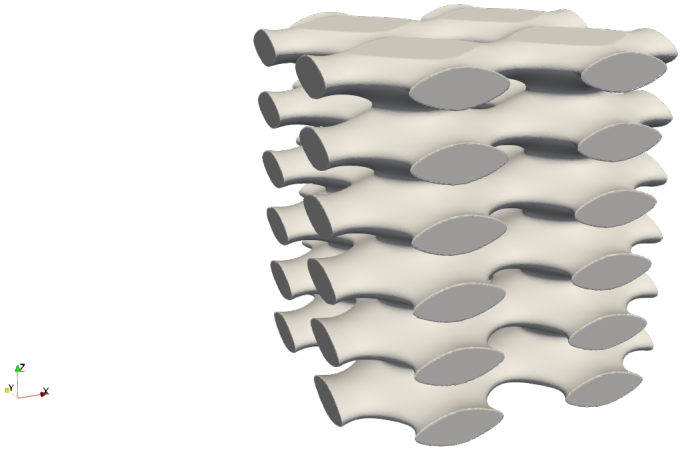
$$\phi(r) = \sum_{k=1}^{K} A_k \cos\left[\frac{2\pi\left(h_k \cdot r\right)}{\lambda_k} + P_k\right] = C$$

  Where A_k is the amplitude, λ_k is the period factor and P_k is the function phase.

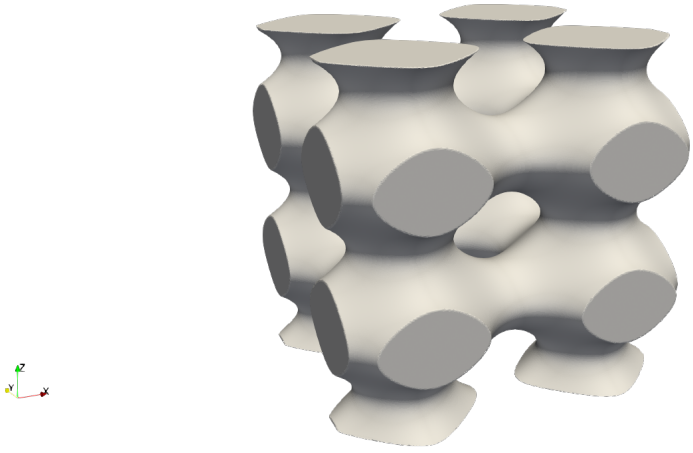
From these equations we can extrapolate that ω influences the TPMS period and C influences the curvature:



(a) $\omega_x = 1 \qquad \omega_y = 1 \qquad \omega_z = 1$



(b) $\omega_x = 1 \qquad \omega_y = 1 \qquad \omega_z = 0.4$



(c) $\omega_x = 1.3 \qquad \omega_y = 1.2 \qquad \omega_z = 0.92$

Figure 2.1: Schwarz-P TPMS with different values of ω

Different equations can express different types of TPMS:

- Gyroid, devised by Alan Schoen in 1970[3]

$$f(x,y,z) = \sin(\omega_x x) + \cos(\omega_y y) + \sin(\omega_z z)\cos(\omega_x x) + \sin(\omega_y y)\sin(\omega_z z) = C$$

  Bonnet angle = 38.0147°

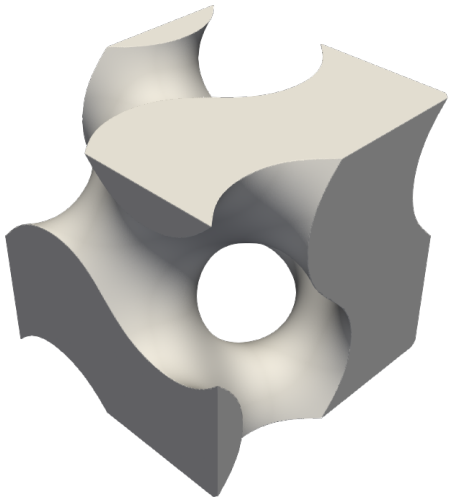


Figure 2.2: 10x10x10 mm Gyroid Cell

- Primitive

$$f(x,y,z) = \cos(\omega_x x) + \cos(\omega_y y) + \cos(\omega_z z) = C$$

  Bonnet angle = 90°

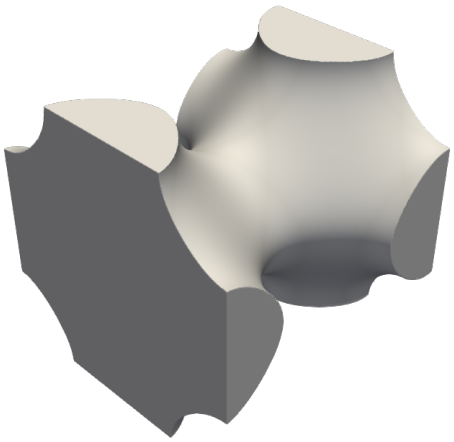


Figure 2.3: 10x10x10 mm Primitive Cell

- Diamond

$$f(x, y, z) = \cos(\omega_x x)\cos(\omega_y y)\cos(\omega_z z) - \sin(\omega_x x)\sin(\omega_y y)\sin(\omega_z z)$$

Bonnet angle = 0°

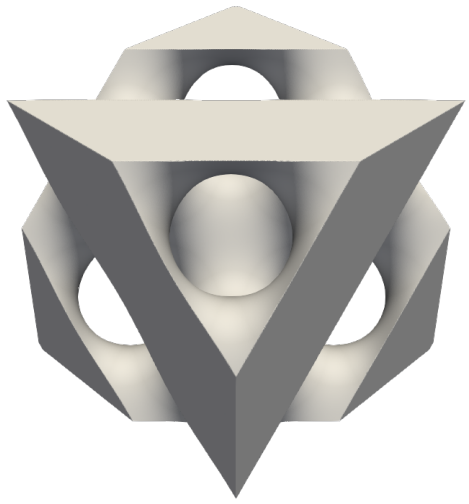


Figure 2.4: 10x10x10 mm Diamond Cell

- Neovious

$$3 \cdot (cos(\omega_x x) + cos(\omega_y y) + cos(\omega_z z)) + 4 \cdot cos(\omega_x x) \cdot cos(\omega_y y) \cdot cos(\omega_z z)$$

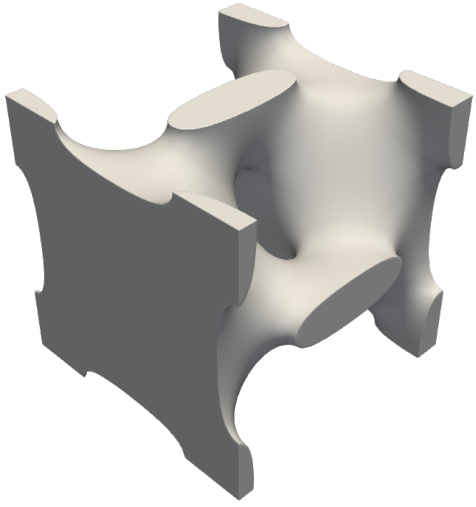
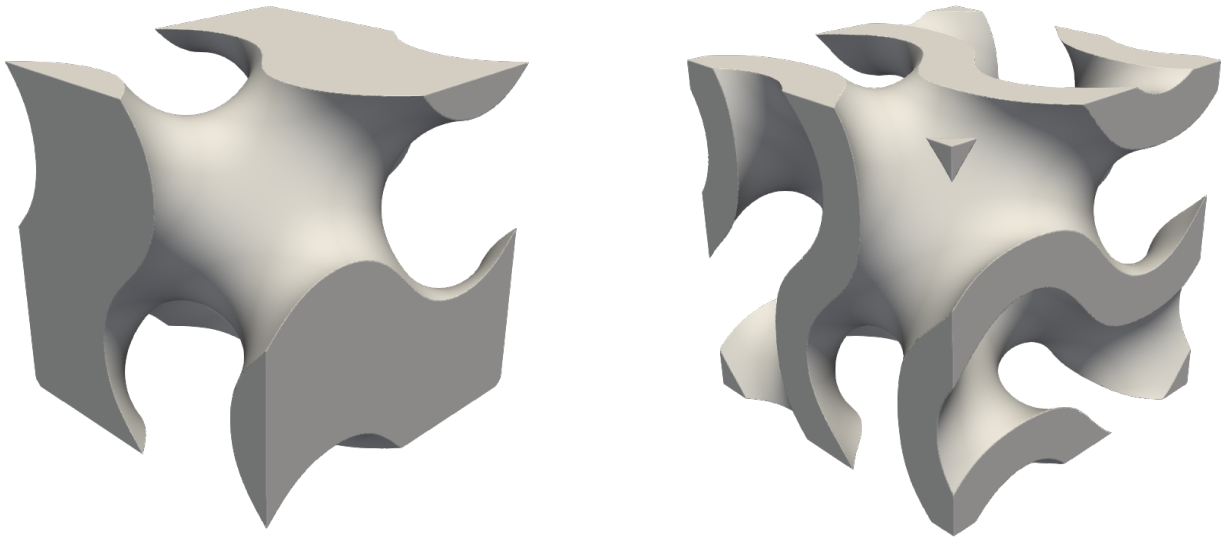


Figure 2.5: 10x10x10 mm Neovious Cell

2.1 TPMS Geometry generation with nTopology

For the generation of our TPMS Geometry we will be using nTopology, which contains the necessary utilities to generate the TPMS lattice. The software can generate both walled and solid TPMS:



(a) Gyroid Solid

(b) Gyroid Sheet

Figure 2.6: Gyroid Solid and Gyroid Sheet Images

The sheet configuration allows for two different fluid channels to exist, meanwhile the solid configuration allows for only one flow channel. For our geometry we are going to be using the solid geometry shown in figure 2.6a which has a 10x10x10 mm cell dimension and no surface offset, so that we can achieve a porosity of 50% meaning that the empty volume is equal to the lattice volume.

OpenFOAM

OpenFOAM is an open-source CFD software developed by released by OpenCFD in 2004. It has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to solid dynamics and electromagnetics.

The models are implemented using an equation syntax that closely follow the mathematical notation, builidng from the operators for:

- Time rate of change: $\frac{\partial}{\partial t}(\phi)$
- Gradient: $\nabla\phi$
- Divegence: $\nabla \cdot \phi$
- Laplacian: $\nabla^2\phi$
- Linearised source: $s\phi$

Complex equations can also be written in a human readable form, the above examples use finite volume calculus. Implicit terms are represented using the finite volume method using "fvm::" form, for example we can render the transport equation to evolve the P-1 radiation model:

$$\nabla \cdot (\Gamma \nabla G) - aG = -4\varepsilon\sigma T^4 - E$$

is represented by the code

```
solve
(
    fvm::laplacian(gamma, G_)
  - fvm::Sp(a_, G_)
 ==
  - 4.0*(e_*physicoChemical::sigma*pow4(T_))-E_
);
```

3.1 Meshing

3.1.1 BlockMesh

The blockMesh utility generates parametric meshes with grading and curved edges, as the name suggests, the mesh is generated by defining multiple blocks defined by 8 vertices, one at each corner of a hexahedron. In our case we will be defining a single block which will be snapped to the geometry with the SnappyHexMesh mesher.

3.1.2 SnappyHexMesh

The snappyHexMesh utility generates 3D meshes containing hexahedra and split-hexahedra from surface geometries in STL format. To start we will be needing a background mesh generated with the blockmesh utility. The meshing process is divided in steps:

- Cell splitting
  Cell splitting is performed is done accordingly to the edge features specified by the user near the edges of the STL surface.

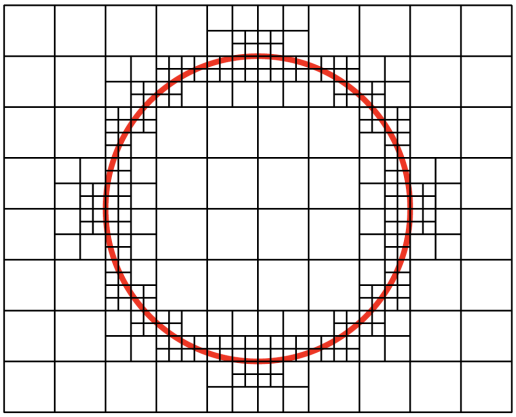


Figure 3.1: Cell splitting around a circle surface

- Cell removal
  Once the splitting is complete, also according to user input, the cells outside the domain of interest are removed.

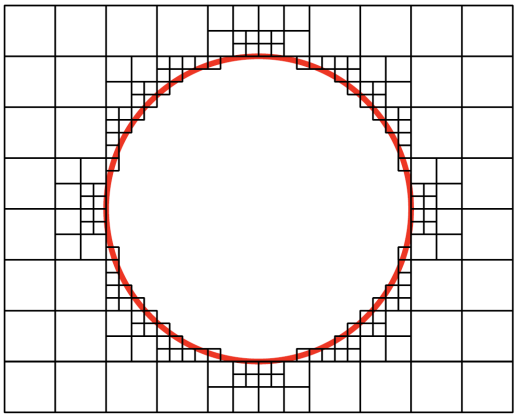


Figure 3.2: Cell removal around a circle surface

- Region refinement
  If we want a finer mesh in a certain regions, assuming a fluid flow from the left in figure 3.3 we can refine the wake region behind the circle:

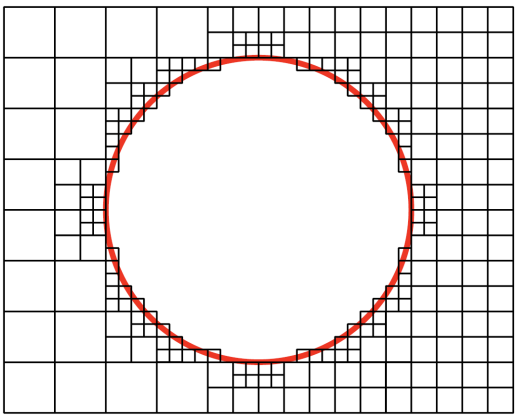


Figure 3.3: Region refinement around a circle surface

- Snapping
  The next stage is snapping to the surface, the software does so with the following steps:

  1. Displace the vertices in the castellated boundary onto the STL surface
  2. Solver for relaxation of the internal mesh with the latest displaced boundary vertices
  3. find the vertices that cause mesh quality parameters to be violated
  4. Reduce the displacement of those vertices from their initial value and repeat from 2 until mesh quality is satisfied.

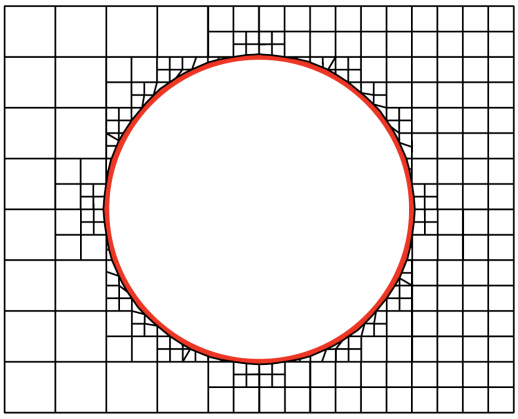


Figure 3.4: Cell snapping around a circle surface

3.2 Solvers

OpenFOAM has many different solvers to be used for different cases, in our we're going to use the "chtMultiRegionSimpleFoam" solver which uses the SIMPLE algorithm to generate a steady state solution with conjugate heat transfer.

3.2.1 The SIMPLE algorithm

The Semi Implicit Method for Pressure Linked Equations (SIMPLE) combines the $\mathbf{u}$,p momentum and mass conservation equations in a sequential solution. [5] The equations are coupled with an algorithm which uses the following notation to describe the terms in the momentum equation:

$$A\mathbf{u} - \mathbf{H}(\mathbf{u}) \equiv \frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) - \frac{\rho(T,p)}{\rho_0}\mathbf{g}$$

where $A\mathbf{u}$ is a linear term in $\mathbf{u}$ and $-\mathbf{H}(\mathbf{u})$ is a function of $\mathbf{u}$ and other sources. The SIMPLE algorithm is used in CFD for flows that reach a steady state, when the flow variables have reached a constant state and are not changing. This state makes it so that the time derivative terms can be ignored. The algorithm uses an iterative sequence which is the following:

1. Construct a matrix equation for energy, with an under relaxation factor α_t.
2. Solve the energy equation for T
3. Update $\rho(T,p)$ according to an equation of state
4. Construct a matrix equation from the momentum equation excluding $\nabla \cdot p$, with under-relaxation factor α_u:

$$\mathbf{A} \cdot \mathbf{u} - \mathbf{b} \equiv \nabla \cdot (\mathbf{u}\mathbf{u}) - \nabla \cdot (\nu \nabla \mathbf{u}) - \frac{\rho(T,p)}{\rho_0}\mathbf{g}$$

5. The second term is equated with $-\nabla \cdot p$ and the matrix equation is solved for $\mathbf{u}$
6. The first term is used to evaluate $A\mathbf{u}$ and $\mathbf{H}(\mathbf{u})$
7. Solve the pressure equation with the evaluated terms

$$\nabla \cdot \frac{1}{A}\nabla p = \nabla \cdot \left[\frac{\mathbf{H}(\mathbf{u})}{A}\right]$$

8. Correct the flux φ_f so that it obeys mass conservation using also a

$$\phi_\mathrm{f} := \mathrm{S}_\mathrm{f} \cdot \left(\frac{\mathbf{H}(\mathbf{u})}{A}\right)_\mathrm{f} - \left(\frac{|\mathbf{S}_\mathrm{f}|}{A}\right)_\mathrm{f} \nabla_\mathrm{n} p_\mathrm{f}$$

   This flux corrector equation interpolates $\mathbf{u}$ to cell faces and evaluates $\phi_\mathrm{f} = \mathbf{S}_\mathrm{f} \cdot \mathbf{u}_\mathrm{f}$.

9. Correct u using the momentum corrector equation and an under-relaxation factor α_p:

$$\mathbf{u} := \frac{\mathbf{H}(\mathbf{u})}{A} - \frac{1}{A}\nabla p$$

   Which updates $\mathbf{u}$ based on the latest values of $\mathbf{u}$ and p.

Convergence

Since a steady-state solver doesn't use time derivatives the matrix equation convergence is reduced because of the absence of a diagonal dominance. That is the main reason as to why we apply the discussed under relaxation factors to the algorithm. The T and $\mathbf{u}$ fields commonly use values for α of 0.7, sometimes reaching 0.3 for compressible flow cases. The p field is different, the flux corrector requires that p is not under-relaxed to ensure that the flux ϕ_f satisfies the mass conservation. To find the optimal α_p for the momentum corrector a series of consideration has been made [31] which finds that $\alpha_p = 1 - \alpha_u$.

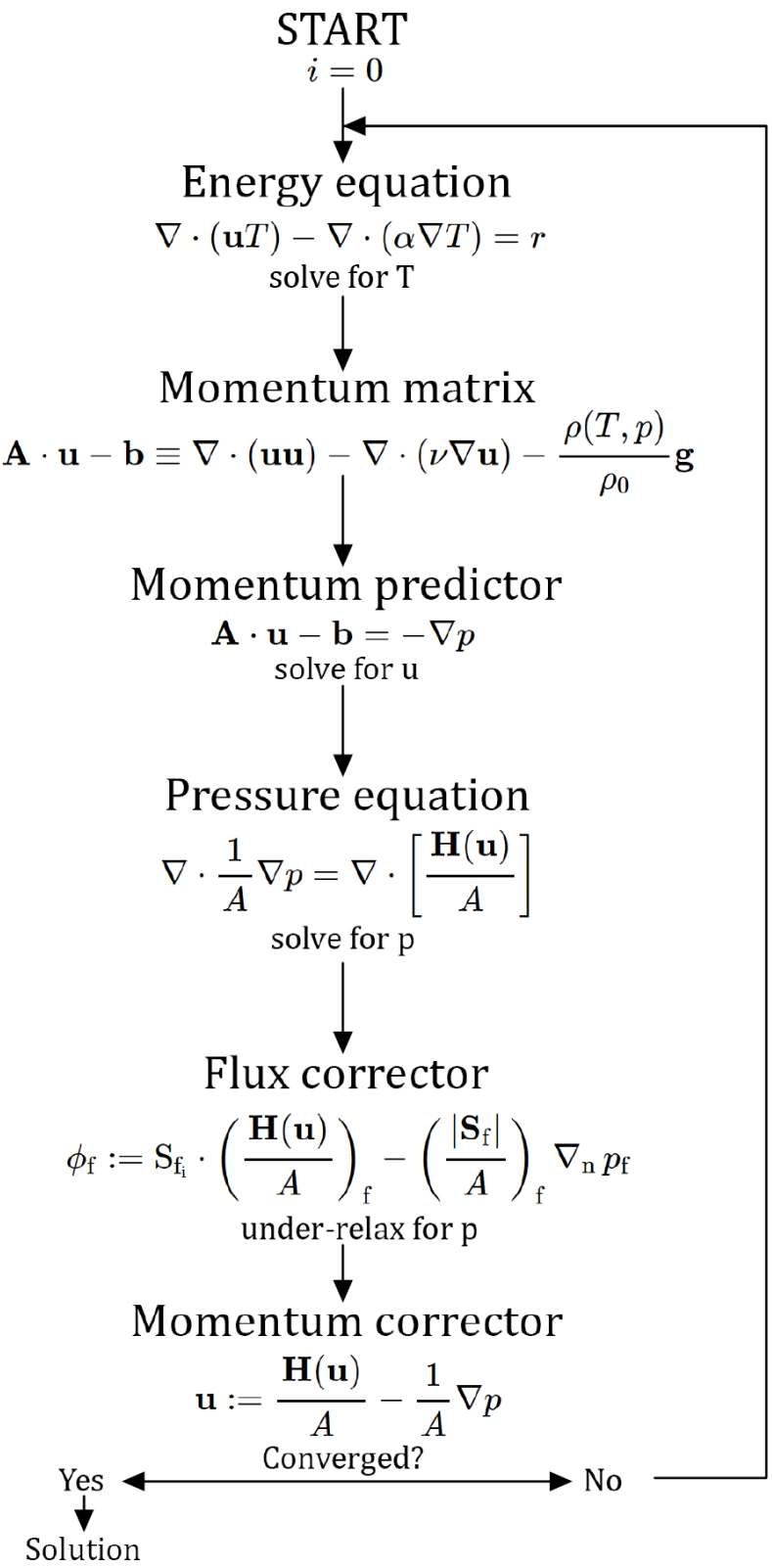


Figure 3.5: SIMPLE Algorithm

3.2.2 chtMultiRegionSimpleFoam structure

The solver calls for a chtMultiRegionSimpleFoam.C file which includes the following lines:

Line	Function
fvCFD.H	Loads finite volume method
rhoThermo.H	Loads thermophysical properties of the fluid regions
turbulentFluidThermoModel.H	Loads user specified turbulent or laminar model
regionProperties.H	Specifies solid and fluid regions
solidThermo.H	Loads the thermophysical properties of the solid region
radiationModel.H	Radiation modelling
fvOptions.H	User specified solver options, e.g. temperature limits
createTime.H	Initiates time variables based on the controlDict file
createMeshes.H	Loads the fluid and solid meshes
createFields.H	Loads the fields for fluid and solid regions

Table 3.1: chtMultiRegionSimpleFoam structure

We can see now the solver loop code in detail:

```
while (runTime.loop())
{
    Info<< "Time = " << runTime.timeName() << nl << endl;

    forAll(fluidRegions, i)
    {
        Info<< "\nSolving for fluid region "
            << fluidRegions[i].name() << endl;
        #include "setRegionFluidFields.H"
        #include "readFluidMultiRegionSIMPLEControls.H"
        #include "solveFluid.H"
    }

    forAll(solidRegions, i)
    {
        #include "setRegionSolidFields.H"
        #include "readSolidMultiRegionSIMPLEControls.H"
        #include "solveSolid.H"
    }
}
```

As we can see the code takes the run time and solves for each fluid and solid regions setting the fields from the last timestep, then readFluidMultiRegionsSIMPLEControls checks the SIMPLE algorithm and with the equations are solved, solveFluid is structured so that it includes:

```
//  Pressure-velocity SIMPLE corrector

{
    if (frozenFlow)
    {
        #include "EEqn.H"
    }
    else
    {
        p_rgh.storePrevIter();
        rho.storePrevIter();

        #include "UEqn.H"
        #include "EEqn.H"
        if (!coupled)
        {
            #include "pEqn.H"
            turb.correct();
        }
    }
}
```

The frozenFlow utility use is specified in the fvSolution dictionary for our case, it can be used once the flow in our case is converged but temperature is not, so that the solvers only solves the energy equation as we can see in the code.

Otherwise we proceed normally solving for the transport equation with UEqn.h:

```
UEqn =
(
    fvm::div(phi, U)
  + MRF.DDt(rho, U)
  + turb.divDevRhoReff(U)
 ==
    fvOptions(rho, U)
);
```

Which solves the general transport equation for $\mathbf{u}$:

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\varphi\mathbf{u}) - \nabla \cdot (\rho U) = S_\varphi$$

The first term is the rate of change of property φ, the second term is responsible for the advection of property φ by the fluid flow and the third term shows the diffusion of property φ meanwhile S_ϕ is the source term.

For the energy equation we are going to solve with the Eeqn.H:

```
{
volScalarField& he = thermo.he();
```

```
fvScalarMatrix EEqn
(
    fvm::div(phi, he)
  + (
        he.name() == "e"
      ? fvc::div(phi, volScalarField("Ekp", 0.5*magSqr(U) + p/rho))
      : fvc::div(phi, volScalarField("K", 0.5*magSqr(U)))
    )
  - fvm::laplacian(turb.alphaEff(), he)
 ==
    rho*(U&g)
  + rad.Sh(thermo, he)
  + fvOptions(rho, he)
);
}
```

here the he term represents either h or e switching the 5th term of the equation depending on the chosen variable.

$$\frac{\partial \rho e}{\partial t} + \nabla \cdot (\rho \mathbf{U} e) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho U K) - \nabla \cdot (\mathbf{U} p) = -\nabla \cdot q + \nabla \cdot (\tau \cdot U) + \rho r + \rho \mathbf{g} \cdot \mathbf{U}$$

$$\frac{\partial \rho h}{\partial t} + \nabla \cdot (\rho \mathbf{U} h) + \frac{\partial \rho K}{\partial t} + \nabla \cdot (\rho U K) - \frac{\partial p}{\partial t} = -\nabla \cdot q + \nabla \cdot (\tau \cdot U) + \rho r + \rho \mathbf{g} \cdot \mathbf{U}$$

3.3 Numerical Schemes

OpenFOAM utilizes collocated variable arrangements that mostly focus on the Finite Volume Method for which the conservative form of the general scalar transport equation of the generic property φ takes the form:

$$\frac{\partial}{\partial t}(\rho \phi) + \nabla \cdot (\rho \phi \mathbf{u}) = \nabla \cdot (\Gamma \nabla \varphi) + S_\varphi$$

The first term is the unsteady term, the second is convection that are equal to the diffusion term plus a source term.

This is integrated over the volume such that:

$$\int_V \frac{\partial}{\partial t}(\rho \phi) dV + \int_V \nabla \cdot (\rho \phi \mathbf{u})\, dV = \int_V \nabla \cdot (\Gamma \nabla \phi)\, dV + \int_V S_\phi dV$$

And this equation is discretised to produce a system of algebraic equations of the form:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ 1 \end{bmatrix}$$

That we can express as:

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

Where we define $\mathbf{A}$ as the coefficient matrix, $\mathbf{x}$ as the vector of unknowns and $\mathbf{b}$ as the source vector. The discretization process employs user defined schemes to build the $\mathbf{A}$ and $\mathbf{b}$ vectors which are selected in the fvSchemes dictionary.

Temporal schemes

The time schemes define how a property is integrated as a function of time, we can define this function as

$$\frac{\partial}{\partial t}(\varphi)$$

We have different options to use, in the case of a steady state time scheme we can use the steadyState option which sets the temporal derivative contributions to zero:

$$\frac{\partial}{\partial t}(\varphi) = 0$$

Spatial Schemes

Spatial schemes use interpolation schemes and the Gauss theorem to convert cell based quantiteties to cell faces, and volume integrals to surface integrals. There are different types:

- Gradient schemes
  The gradient of the generic property φ is:

  $$\nabla\phi = \mathbf{e}_1\frac{\partial}{\partial x_1}\phi + \mathbf{e}_2\frac{\partial}{\partial x_2}\phi + \mathbf{e}_3\frac{\partial}{\partial x_3}\phi$$

  where the $\mathbf{e}$ vectors are the unit vectors. We use the Gauss scheme which uses the Gauss theorem to calculate the cell gradient:

  $$\int_V (\nabla\cdot\mathbf{u})\, dV = \oint_S (\mathbf{n}\cdot\mathbf{u})\, dS$$

  And we use a cell-based linear interpolation scheme. We can also use limiters to limit the gradient and bound the face value of a cell to the neighboring cell values. We are going to mainly use cell-limited gradient schemes where the cell gradient is limited so that the cell faces using the gradient are bounded by neighboring cells minimum and maximum limits. For example, we can consider a limiter between cells P and N that bounds the property φ in P to cell N as in figure 3.6:

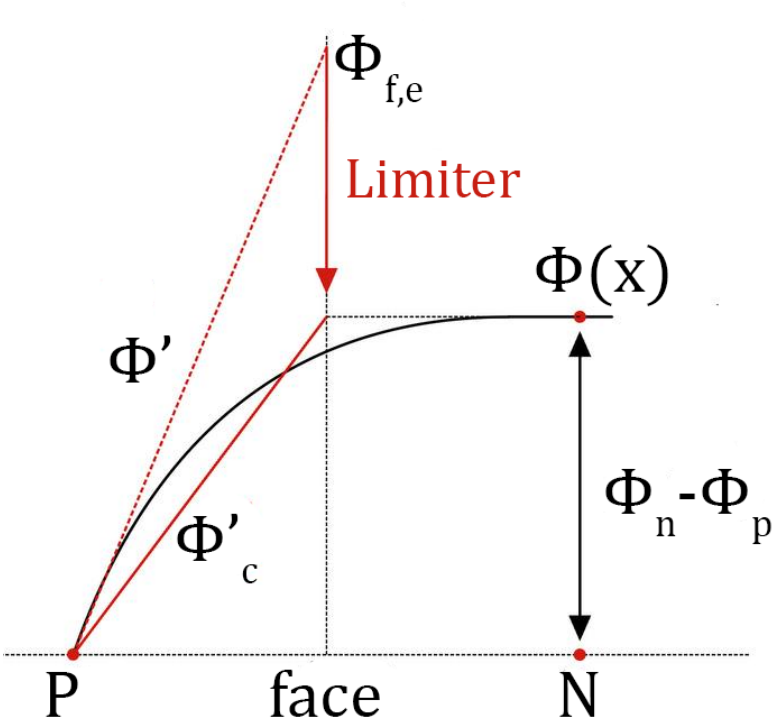


Figure 3.6: Cell based limiter

This helps to solve grids with relatively high non-orthogonality and we are going to use these type of limiters.

- Divergence schemes
  Divergence schemes handle how the divergence of property φ is handled:

$$\nabla \cdot \varphi$$

$$\nabla \cdot \varphi = \frac{\partial \varphi_x}{\partial x} \frac{\partial \varphi_y}{\partial y} \frac{\partial \varphi_z}{\partial z}$$

. We are going to use mainly first order upwind schemes and linear/ linear-upwind second order schemes so that our solution is stable since we will be dealing with a complex mesh for our geometry. Upwind is a first order bounded scheme, which sets the face value according to the upstream value, it assumes that the cell values are isotropic with a value that represents the average value.

$$\varphi_f = \varphi_c$$

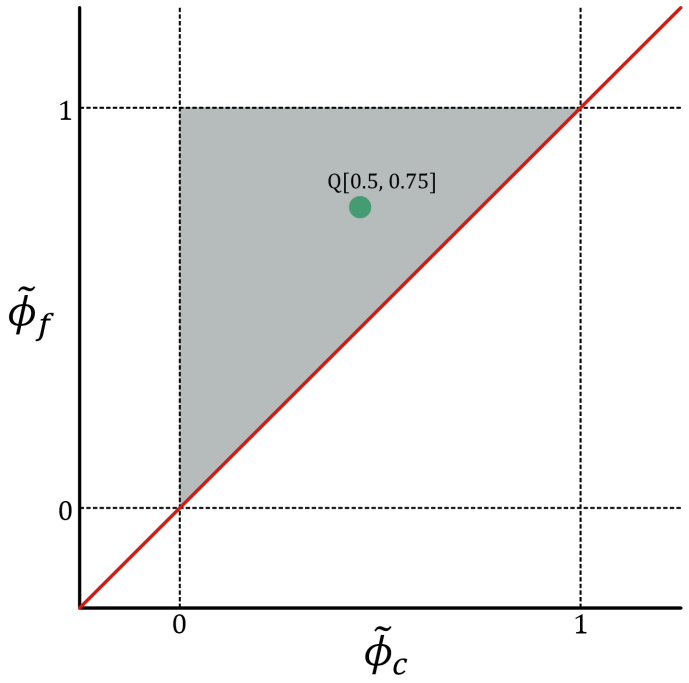


Figure 3.7: Normalized variable scheme for first order upwind scheme

Otherwise linear upwind uses upwind interpolation weights, but uses a correction based on the local cell gradient, it is a second order unbounded scheme.

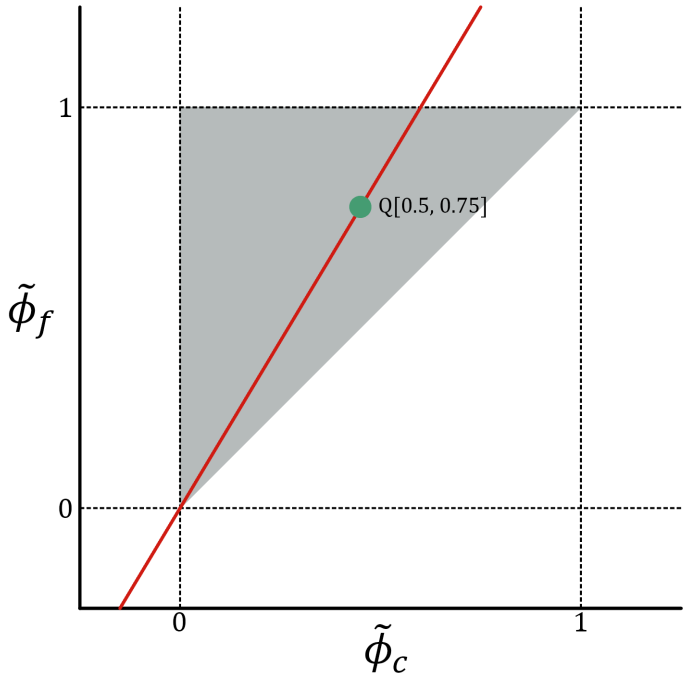


Figure 3.8: Normalized variable scheme for second order upwind scheme

- Laplacian schemes Laplacian schemes handle how the Laplacian of the generic property φ is solved:

$$\nabla^2\varphi = \frac{\partial^2}{\partial x_1^2}\varphi\frac{\partial^2}{\partial x_2^2}\varphi\frac{\partial^2}{\partial x_3^2}\varphi$$

  which can also be expressed as a combination of divergence and gradient operators using a diffusion coefficient Γ:

$$\nabla \cdot (\Gamma\nabla\varphi)$$

  These Laplacian schemes are all based on the application of the Gauss theorem and the use the specified interpolation scheme to transform coefficients from cell values to the faces.

- Surface-normal gradient schemes The full gradient of property Q can be interpolated from the cell based gradient. The surface normal contribution is represented by:

$$\nabla_f^{\perp} Q = \mathbf{n} \cdot (\nabla Q)_f$$

  where n is the face unit normal.

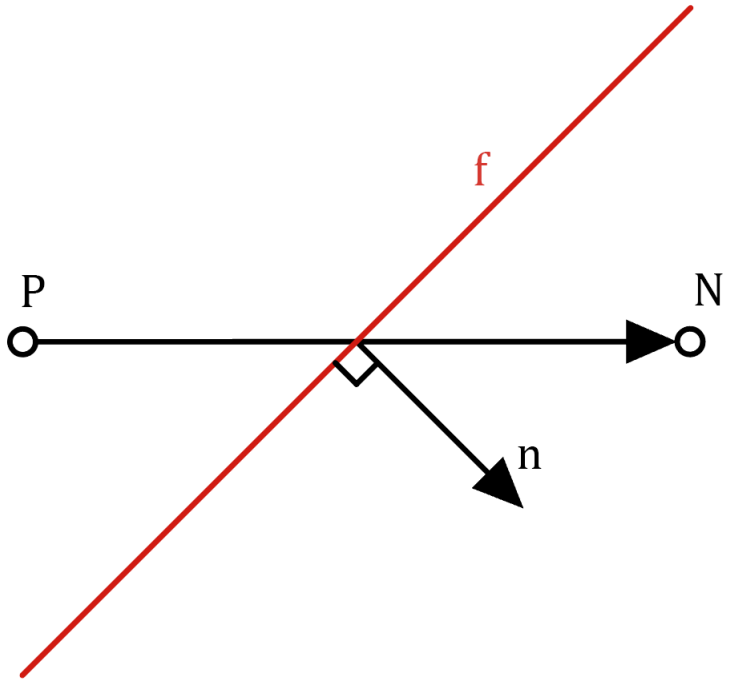


Figure 3.9: Gradient calculation

From 3.9 we can see that there is that the degree of non-orthogonality is given by the angle between **d** and **n** vectors, we can choose our scheme based on that degree. Non-orthogonality is a very important factor in solution stability and accuracy, the diffusion term needs an orthogonal mesh, otherwise the boundedness of the solution cannot be guaranteed. To stabilize the solution non orthogonal corrections are applied, using limited or corrected schemes.

These schemes are governed by a coefficient ψ that determines how heavily corrected is the scheme, a more stable scheme yields less accurate results.

$$\psi = \begin{cases} 0 & \text{corresponds to uncorrected,} \\ 0.333 & \text{non} - \text{orthogonal correction} \leq 0.5 \times \text{orthogonal part,} \\ 0.5 & \text{non} - \text{orthogonal correction} \leq \text{orthogonal part,} \\ 1 & \text{corresponds to correction} \end{cases}$$

3.4 Turbulence Model

For the turbulence model there are two main models that we can utilize which are both two equation models.

3.4.1 $k - \varepsilon$ model

The $k - \varepsilon$ model utilizes the transport equations of the turbulent kinetic energy k and of the turbulent dissipation rate ε. The conservation equation for turbulent kinetic energy is:

$$\rho \frac{Dk}{Dt} = \nabla \cdot (\rho D_k \nabla k) + \rho G - \frac{2}{3}\rho k(\nabla \cdot \mathbf{u}) - \rho\varepsilon$$

And the transport equation for ε is

$$\rho \frac{D\varepsilon}{Dt} = \nabla \cdot (\rho D_\varepsilon \nabla \varepsilon) + c_1 \rho G \frac{\varepsilon}{k} - \frac{2}{3} c_1 \rho \varepsilon (\nabla \cdot \mathbf{u}) - c_2 \rho \frac{\varepsilon^2}{k}$$

G is the turbulence generation:

$$G = 2\nu_\mathrm{t} \operatorname{dev} \mathbf{D} : \nabla \mathbf{u}$$

We assume constant ρ and replace the material derivatives in conservative form, for the steady state solution the time derivatives will be equal to zero so the equations will become:

$$\nabla \cdot (\mathbf{u}k) - \nabla \cdot (D_k \nabla k) = G - \frac{2}{3} k(\nabla \cdot \mathbf{u}) - \varepsilon$$

$$\nabla \cdot (\mathbf{u}\varepsilon) - \nabla \cdot (D_\varepsilon \nabla \varepsilon) = c_1 G \frac{\varepsilon}{k} - \frac{2}{3} c_1 \varepsilon (\nabla \cdot \mathbf{u}) - c_2 \frac{\varepsilon^2}{k}$$

We are going to utilize the standard model values which have been chosen from a large series of simulations of free shear flows[4]:

- $D_k = \nu + \nu_t/\sigma_k$ this term represent the diffusion generated by the molecular and turbulent motions, adjusted with the coefficient σ_k.
- $D_\varepsilon = \nu + \nu_t/\sigma_\varepsilon$ this term represent the diffusion generated by the molecular and turbulent motions, adjusted with the coefficient σ_ε
- $c_\mu = 0.09$
- $c_1 = 1.44$
- $c_2 = 1.92$
- $\sigma_k = 1.0$
- $\sigma_\varepsilon = 1.3$

3.4.2 Realizable k-ε

The realizable $k-\varepsilon$ turbulence model is a model that satisfies certain mathematical constraints on the Reynolds stresses consistent with the physics of turbulent flows. [28] The model combines the Boussinesq relationship and the eddy viscosity definition to obtain the normal Reynolds stress in an incompressible strained mean flow

$$\overline{u^2} = \frac{2}{3}k - 2\,\nu_\ell \frac{\partial U}{\partial x}$$

.

The model becomes non realizable when the normal stress $\overline{u^2}$, which should be positive by definition becomes negative, this corresponds to the following:

$$\frac{k}{\varepsilon}\frac{\partial U}{\partial x} > \frac{1}{3C_\mu} \approx 3.7$$

So to achieve realizability C_μ becomes a variable depending on mean flow and turbulence i.e. this becomes 0.05 for a strong homogeneous shear flow and 0.09 for the inertial sublayer of equilibrium in boundary layers.

The realizable $k-\varepsilon$ model predicts the spreading rate of planar and round jets more accurately with respect to the standard model, also having superior performance with flows involving rotation, boundary layers under strong adverse pressure gradients, separation and recirculation.

The turbulence kinetic energy equation is given by:

$$\frac{D}{Dt}(\rho k) = \nabla \cdot (\rho D_k \nabla k) + \rho G - \frac{2}{3}\rho\,(\nabla \cdot \mathbf{u})\,k - \rho\varepsilon + S_k$$

and the dissipation rate is:

$$\frac{D}{Dt}(\rho\varepsilon) = \nabla \cdot (\rho D_\varepsilon \nabla \varepsilon) + C_1\rho|\mathbf{S}|\varepsilon - C_2\rho\frac{\varepsilon^2}{k + (\nu\varepsilon)^{0.5}} + S_\varepsilon$$

The turbulent viscosity is calculated with:

$$\nu_t = C_\mu \frac{k^2}{\varepsilon}$$

Where C_μ is:

$$C_\mu = \frac{1}{A_0 + A_s U^* \frac{k}{\varepsilon}}$$

3.4.3 Wall functions

At the walls the tangential flow speed increases rapidly across a thin boundary layer, we can plot a dimensionless velocity and distance to the wall:

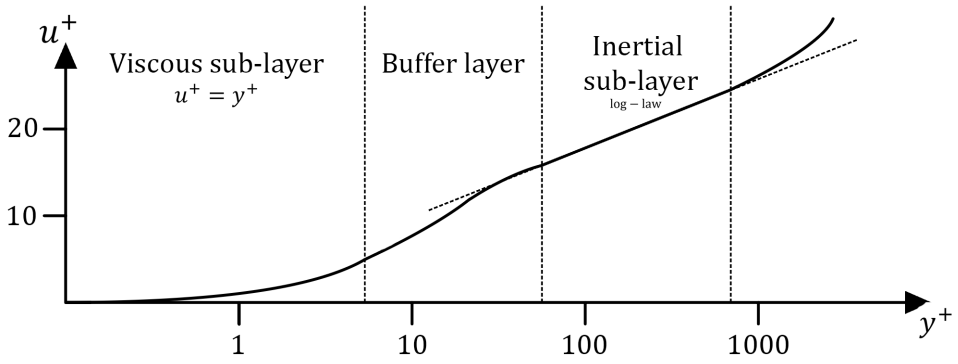


Figure 3.10: Dimensionless velocity profile

$$u^+ = \frac{u_x}{u_\tau} \qquad y^+ = \frac{u_\tau y}{\nu}$$

Where u_τ is a friction velocity which is correlated to the wall shear stress τ_w by

$$\tau_w = \rho u_\tau^2$$

The u^+ profile is defined by the value of y^+:

- $u^+ = y^+$ for $y^+ < 5$, known as the viscous sub-layer
- $l_\mathrm{m} = \kappa y \left[1 - \exp(-y^+/26)\right]$ describes the mixing length in the buffer layer, for $5 < y^+ < 40$
- $u^+ = \frac{1}{\kappa} \ln(y^+) + B$ is the logarithmic law of the wall that is used in the inertial sub-layer, where the flow is turbulent, for $y^+ > 40$

$\kappa = 0.41$ and $B = 5.0 - 5.5$ are constants

To alleviate the computational cost of calculating τ_w with sufficient accuracy, wall functions are used to provide a reasonable prediction of τ_w from a relatively inaccurate $\frac{\partial u_x}{\partial y}$ calculation at the wall.

The main problem is that with turbulent boundary layers the $\frac{\partial u_x}{\partial y}$ requires cells with very small lengths normal to the wall to be accurate, resulting inevitably in a large mesh. [31]

Wall functions use the distance y_p to the wall from the center of each cell so that we have

$$y_p^+ = \frac{u_\tau y_p}{\nu}$$

The calculated $\frac{\partial u_x}{\partial y}$ is significantly lower than its real value so wall function models compensate for the resulting error in prediction of τ_w by increasing the viscosity at the wall, making it so that $\nu_t \neq 0$.

The turbulence generation G influences the distribution of turbulence fields near a wall.

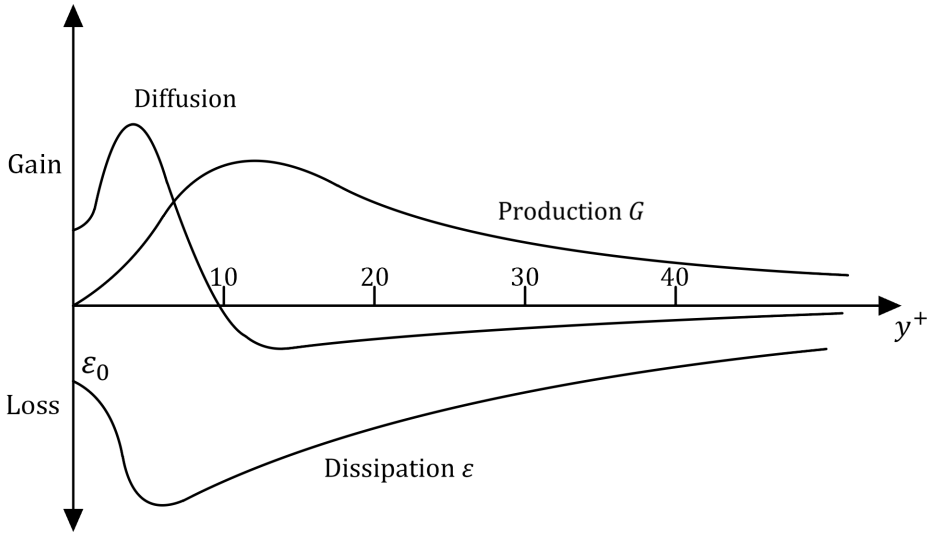


Figure 3.11: Properties evolution with dimensionless distance

As we can estimate τ_w from its wall function, when using a turbulence model we have to specify the boundary conditions at the wall for the turbulent values, i.e. k and ε. The distribution of dimensionless $k^+ = k/\mu_\tau^2$ is represented in figure 3.12.

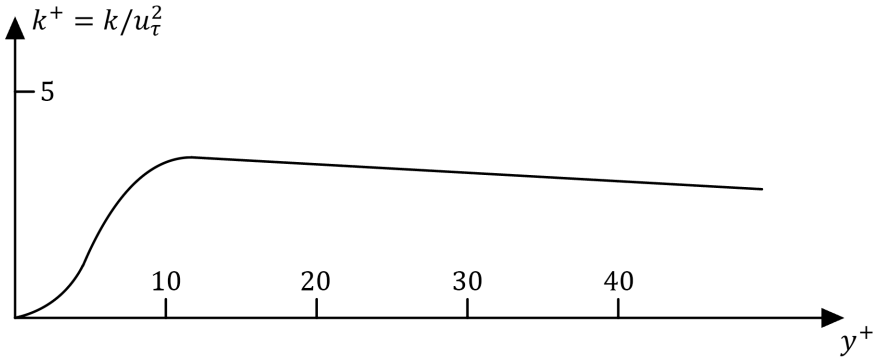


Figure 3.12: k evolution with dimensionless distance

Epsilon wall function

The epsilon wall function is expressed as "epsilonWallFunction" in the boundary conditions. This condition provides a wall constraint on the turbulent kinetic energy dissipation rate, and the turbulent kinetic energy production for low and high Reynolds number turbulence models.

The model expressions are the following:

$$\varepsilon_{vis} = 2wk\frac{\nu_w}{y^2}$$

$$\varepsilon_{log} = wC_\mu\frac{k^{3/2}}{\nu_{t_w}y}$$

$$G = w(\nu_{t_v} + \nu_w)|\mathbf{n} \cdot (\nabla \mathbf{u})_f| C_\mu^{1/4} \frac{\sqrt{k}}{\kappa y} \qquad if \quad y^+ \geq y^+_{tam}$$

Where ε_{vis} and ε_{log} is the turbulent kinetic energy dissipation rate for the viscous and inertial sub-layers respectively. w are the cell corner weights. The epsilon prediction for the viscous and inertial sublayers can be blended with four different methods:

- **Stepwise switch**
  The different estimations are switched at $y^+ = y_{lam}$

  $$\epsilon = \epsilon_{vis} \qquad if \quad y^+ < y^+_{lam}$$

  $$\epsilon = \epsilon_{log} \qquad if \quad y^+ \geq = y^+_{lam}$$

  where $y^+_l am$ is the estimated intersection of the viscous and inertial sublayers in wall units.

- **Maximum value switch**

  $$\varepsilon = \max(\varepsilon_{vis}, \varepsilon_{log})$$

- **Binomial blending**

  $$\epsilon = ((\epsilon_{vis})^n + (\epsilon_{log})^n)^{1/n}$$

  where n is the binomial blending exponent.

- **Exponential blending**

  $$\epsilon = \epsilon_{vis} \exp[-\Gamma] + \epsilon_{log} \exp[-1/\Gamma]$$

  The epsilon estimation is blended between the sublayers using an exponential function. Γ is a blending expression: $\Gamma = 0.001(y^+)^4/(1.0 + y^+)$

G predictions are always blended with the stepwise method, G below y^+_{lam} is assumed to be zero.

Choice of turbulence model

The $k-\varepsilon$ model has been chosen for the case study due to its general applicability, it is rather simple and widely used in many application and given the high number of cells in our mesh, the relatively low computational cost of the model is another advantage. Its realizable variant eliminates many of its disadvantages and it is more appropriate than the standard $k-\varepsilon$ model which struggles with severe curvatures in the flow field [14] that we find in the TPMS. The $k-\omega$ SST model was also considered initially but ultimately, while yielding similar results, the Realizable $k-\varepsilon$ model was chosen because of better stability in the other softwares used in the code-to-code benchmark.

Case study

4.1 Case and geometry

Our geometry is comprised by a box of 100x100 mm with a height of 10 mm, inside this box there is a TPMS lattice with gyroid units of 10x10 mm:

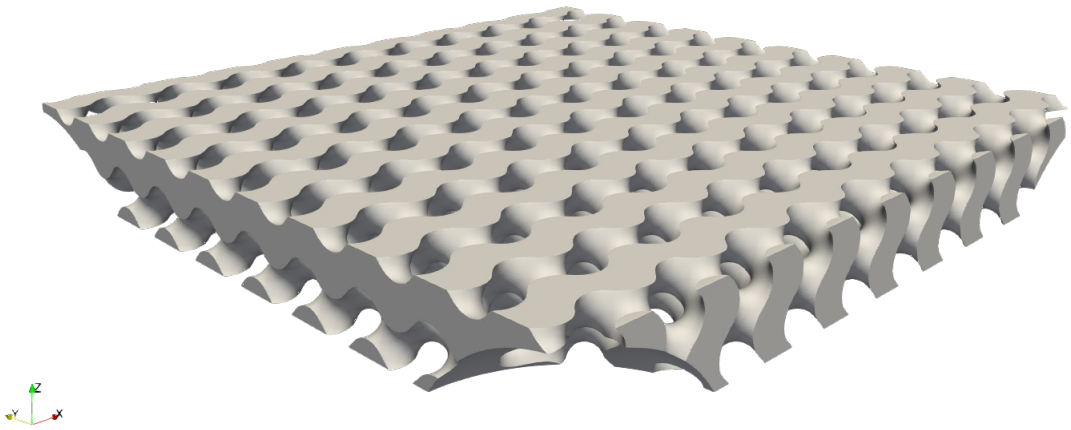


Figure 4.1: Lattice inside the geometry

At the manifold entrance the geometry is cut with an ellipsoid to avoid problems with the interaction of the flux directly with the TPMS geometry.

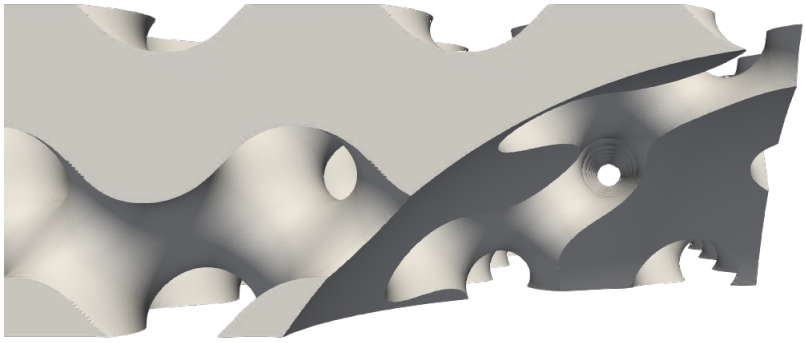


Figure 4.2: Manifold area

Connected to the box there are four manifolds where fluid enters and exits:

Figure 4.3: Solid region

4.2 Meshing

For meshing the snappyHexMesh utility described in section 3.1.2 is utilized, an initial 100x100x25 mm blockMesh is generated from which the mesher will snap on the stl file:

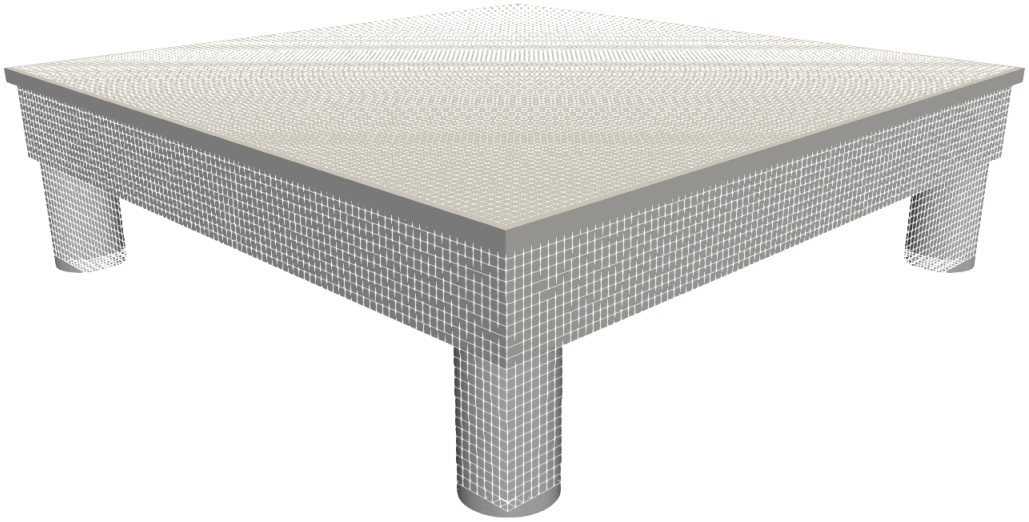


Figure 4.4: Background blockmesh compared with the stl

While the blockmesh in figure 4.4 has a 1 mm background mesh size the actual size of the elements generated in the background mesh utilized is of 0.36 mm. After generating the background mesh we setup the snappyHexMesh through the snappyHexMeshDict file. We set a level of refinement in the fluid region of 2, meaning the tetrahedral mesh gets split in 2 levels:

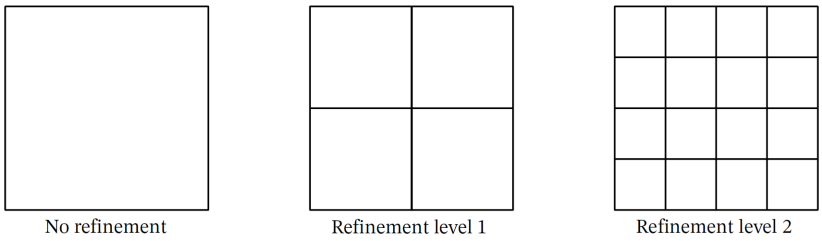


Figure 4.5: Refinement levels

Our interest is in having a finer mesh at the surface, where turbulent phenomena and interactions between the boundary layers is more complex.

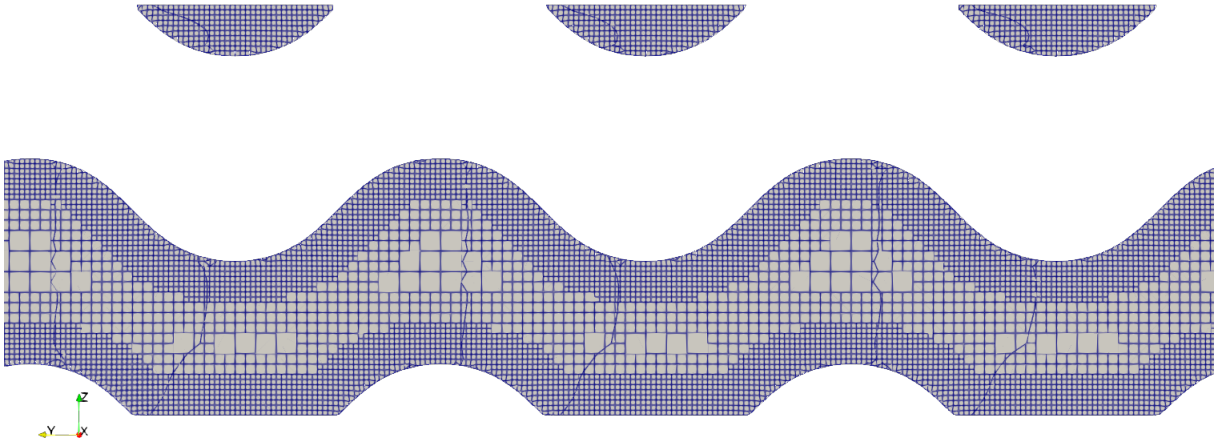


Figure 4.6: Mesh generated with SnappyHexMesh

4.3 Boundary Conditions

In our case we will be evaluating the cooling performance of these TPMS geometries. we will have a mass flow of water at the different manifolds, we will be studying two cases, one with laminar flow and one with turbulent flow with the following mass flow and heat flux:

	Laminar	**Turbulent**
$\dot{m}\,[kg/s]$	0.0004	0.833
$\dot{Q}\,[kW/m^2]$	10	2500

Table 4.1: Mass flow and heat flux boundary conditions

The heat flux is applied directly on a 50x50 mm tungsten plate placed on top of the solid geometry.

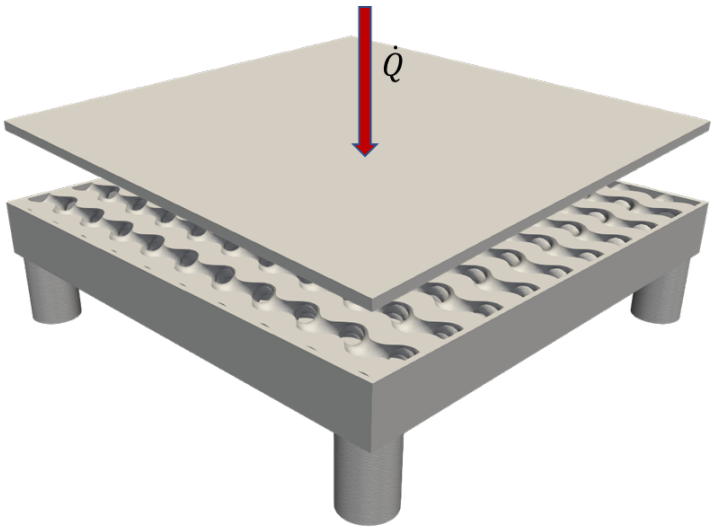


Figure 4.7: Heat Load on the tungsten plate

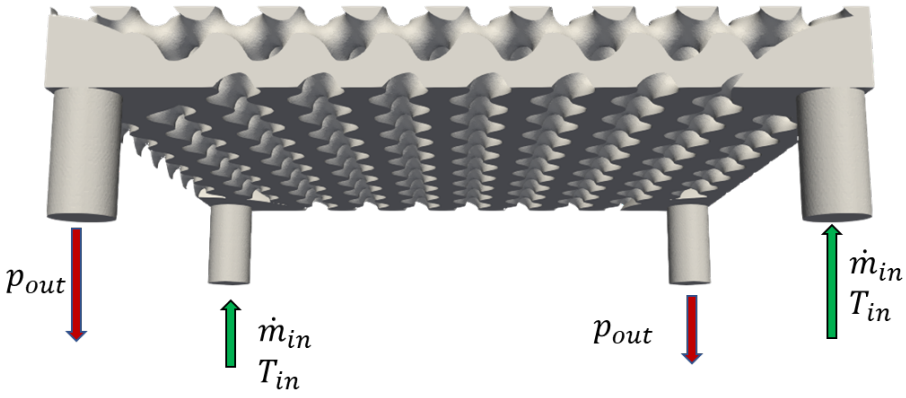


Figure 4.8: Inlet-outlet disposition in the fluid region

Let's see now how to set the proper boundary conditions in OpenFOAM:
Boundary conditions are setup in the "0" folder, where the problem is defined at the starting time, in this folder there are multiple files, where the boundary conditions are defined:

- Velocity
  Velocity boundary conditions are defined in the "U" file, for our case we are setting up a mass flow at the inlet, and the "zeroGradient" type at the outlets which imposes

$$\frac{\partial U}{\partial n} = 0$$

  at the outlet patch so that we avoid backflow. At the walls we use the "noSlip" type to set $u_p = 0$ at the walls.

- Pressure
  Pressure boundary conditions are specified in the "p" and "p_rgh", but the solver actually

solves for p_rgh, which is defined as p' in:

$$p' = p - \rho(g \cdot h)$$

and the momentum equation is solved using this term. So for the "p" we set the boundary conditions to "calculated" across the whole domain. Meanwhile the p_rgh boundary conditions are set to "fixedValue" at the outlets where the value is 7.5 bar and "fixedFluxPressure" at the inlets, where the pressure is specified by the velocity boundary condition.

- Temperature
  The temperature boundary conditions are specified in the "T" file, where we set a "fixedValue" condition at the inlets with a temperature of 300 K, so that the fluid enters at the inlet with a constant temperature of 300 K. For the heat transfer at the walls we are going to use the "turbulentTemperatureRadCoupledMixed" boundary condition, which combines conduction and radiation.

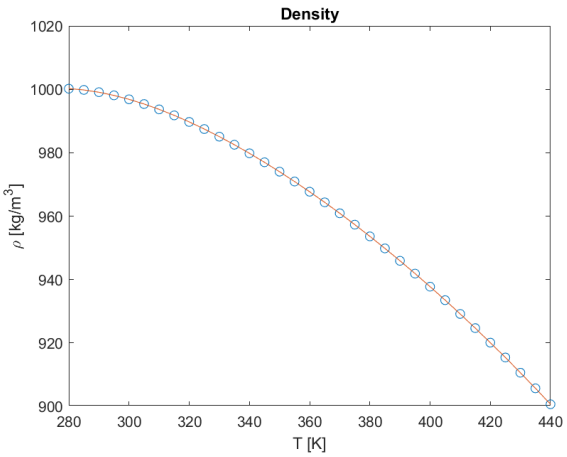
4.4 Material Properties

The solid region material is copper, while the top plate that receives the heat load is tungsten, the properties are the following:

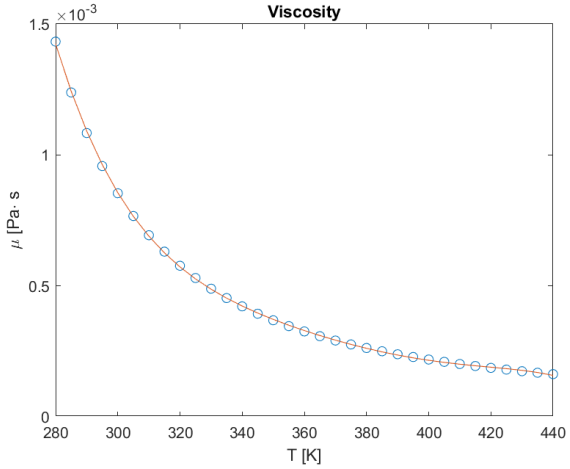
Properties	**Copper**	**Tungsten**
Density (kg/m^3)	8940.0	19300.0
Specific heat ($J/kg \cdot K$)	386.0	134.0
Conductivity ($W/m \cdot K$)	398.0	163.2

Table 4.2: Copper and tungsten properties

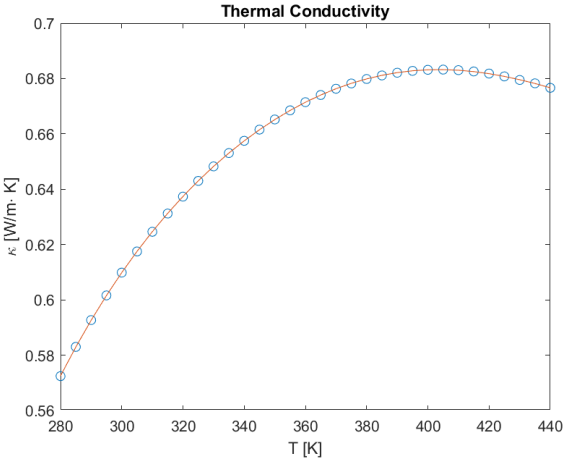
The fluid is water, with variable properties extrapolated with a polynomial fit of data from the U.S. National Institute of Standards and Technologies[40] :



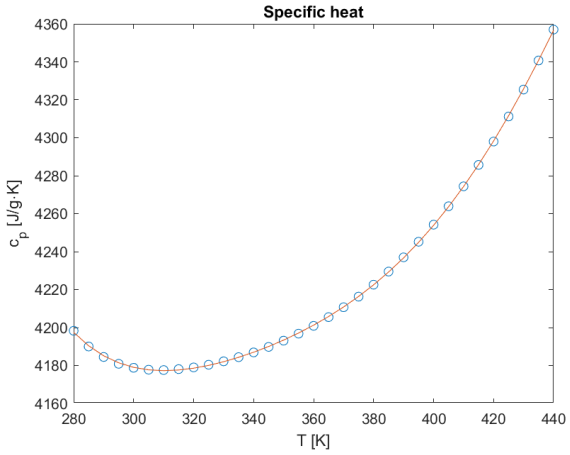
(a) Density



(b) Viscosity



(c) Thermal Conductivity



(d) Specific Heat

Figure 4.9: Variable properties of water

4.5 Grid independence study

Before presenting our results we need to establish the numerical error on our result, but what generates the numerical error? In our case we have to do solution verification, only estimating the error since we don't know our exact solution.

Solution Verification

To estimate this error we're going to do a grid-refinement study. There is a difference between an error estimate and an uncertainty estimate: if we have a value f and an error estimate ε for example we are going to get an improved value $f-\varepsilon$ that is closer to the exact value f_{true}. Meanwhile if we're using an uncertainty estimate $U_{x\%}$ we're going to have an interval $f \pm U_{x\%}$ where f_true probably falls. Our problem is quantifying that probability $x\%$. To determine the probability of 95% we are going to use the Grid Convergence Index (GCI) that utilizes a factor of safety Fs that converts an error estimate, that we calculate with a Richardson Extrapolation (RE), into an uncertainty estimate.

Least Squares GCI

For convergence rates that aren't optimal a least squares method that provides better uncertainty estimation has been developed. [8] that limits the maximum p used in the GCI to the theoretical p. The one term expansion of the discretization error is assumed as:

$$f_i - f_\infty \approx \alpha \Delta_i^p$$

. Then we minimize the function:

$$S\left(f_\infty \alpha, p\right) = \sqrt{\sum_{i=1}^{Ng} \left[f_i - \left(f_\infty + \alpha \Delta_i^p\right)\right]^2}$$

The number of grids Ng must be ¿3 and f_∞ suggests the limit of fine resolution. If the derivatives of S are zero we find:

$$f_\infty = \frac{1}{Ng}\left\{\sum_{i=1}^{Ng} f_i - \alpha \sum_{i=1}^{Ng} \Delta_i^p\right\}$$

$$\alpha = \frac{Ng \sum\limits_{i=1}^{Ng} f_i \Delta_i^p - \left(\sum\limits_{i=1}^{Ng} f_i\right)\left(\sum\limits_{i=1}^{Ng} \Delta_i^p\right)}{Ng \sum\limits_{i=1}^{Ng} \Delta_i^{2p} - \left(\sum\limits_{i=1}^{Ng} \Delta_i^p\right)\left(\sum\limits_{i=1}^{Ng} \Delta_i^p\right)}$$

$$\sum_{i=1}^{Ng} f_i \Delta_i^p \log(\Delta_i) - f_\infty \sum_{i=1}^{Ng} \Delta_i^p \log(\Delta_i) - \alpha \sum_{i=1}^{Ng} \Delta_i^{2p} \log(\Delta_i)$$

This last equation is nonlinear and solved iteratively by a false position method for observed p.

Uncertainty Estimation

The RE assumes that discrete solutions, f, have a power series representation in the grid spacing, h, which for our nonstructured grid is equal to

$$h = \left[\left(\sum_{i=1}^{N} \Delta V_i \right) / N \right]^{1/3}$$

where N is the total number of cells used for the computations and ΔV_i is the volume of the i^{th} cell. Our four grids have the following data:

Grid	**1**	**2**	**3**	**4**
Number of Cells	42055640	18780149	8399334	3837821
$\mathbf{\Delta V_i [mm^3]}$	52898.703	52888.904	52875.738	52867.138
h [m]	0.1079	0.1412	0.1846	0.2397

Table 4.3: Grids' parameters

Since we are going to do a least squares approach we are going to use the minimum of four grids, the desirable refinement factor is

$$r = \frac{h_{coarse}}{h_{fine}} > 1.3$$

to keep an acceptable grid size we're going to stay around the 1.3 value:

Grids	**1/2**	**2/3**	**3/4**
r	1.2984	1.3075	1.3082

Table 4.4: r values for different grids

This value is based on experience and not a formal derivation. In any case the refinement of the grid should be systematical, if we decrease cell size, we should do that in all directions and not just one, even if the grid is unstructured. We can calculate the relative and absolute error against the grid:

$$e_r^{21} = \left| \frac{\varphi_1 - \varphi_2}{\varphi_1} \right|$$

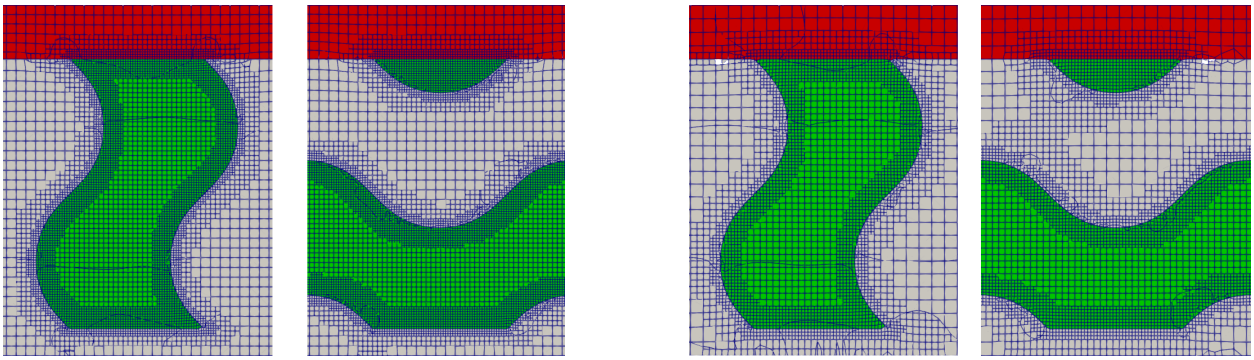
$$e_a^{21} = |\varphi_1 - \varphi_2|$$

As said before p and the extrapolated values φ_{ext} is determined with a least squares method and used to calculated the extrapolated error:

$$e_{ext}^{21} = \left| \frac{\varphi_{ext}^{21} - \varphi_1}{\varphi_{ext}^{21}} \right|$$

At this point we can find the fine Grid Convergence Index using the factor of safety of $Fs = 1.25$ for obtaining a GCI with a 95% confidence interval for structured refinement:

$$GCI_{fine}^{21} = \frac{Fs \cdot e_a^{21}}{r_2 1^p - 1}$$



(a) Refinement level 1

(b) Refinement level 2

(c) Refinement level 3

(d) Refinement level 4

Figure 4.10: Mesh refinement levels

Finally we can find the u_{num} using an expansion factor $k = 1.15$

$$u_{num} = \frac{GCI}{k}$$

Let's take for example the laminar case, for our four meshes we find the following data:

	Δp [Pa]	**ΔT_{Solid}** [K]	**ΔT_{Fluid}** [K]
Mesh 1	3.58617372e-01	57.67067	57.58696
Mesh 2	3.57239478e-01	57.83928	57.74929
Mesh 3	3.55650062e-01	57.98901	57.90084
Mesh 4	3.53467269e-01	58.08218	58.00067

Table 4.5: Values found for laminar case

These values can then be used to find the GCI of the various results and the relative numerical error:

	GCI	u_{num}
Δp	0.0337	0.0293
ΔT_{Solid}	0.0546	0.0475
ΔT_{Fluid}	0.0551	0.0479

Table 4.6: Grid convergence index and relative numerical error

4.6 Results

4.6.1 Turbulent Flow with heat transfer

For our simulations we will be using a combination of first and second order schemes. At the start we will be using 1st order upwind schemes for both U and the turbulent variables k and ε, this yields low residuals and stable results:

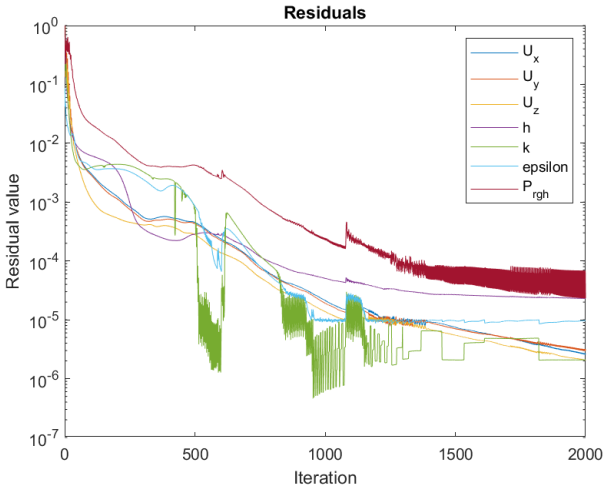
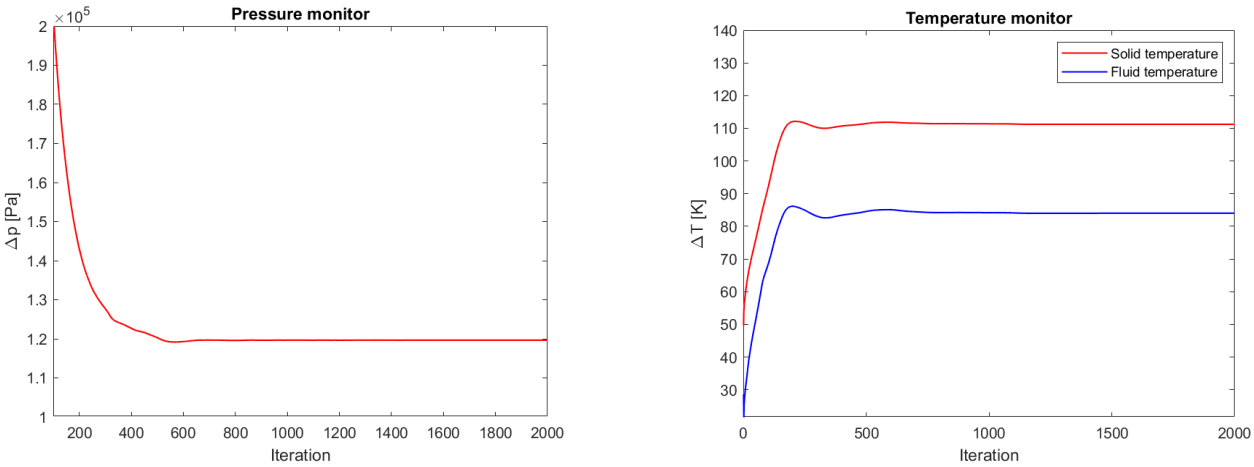


Figure 4.11: Residuals for first order upwind discretization schemes

If we monitor the temperature in a point in the solid region and the pressure difference between the inlet and the outlet we can see the results for in figure 4.12



(a) Δp [Pa] (b) T_W [K]

Figure 4.12: First order upwind discretization schemes results

Although these results are stable first order discretization for the velocity field is generally not acceptable, so we will switch to second order linear-upwind discretization schemes which yield the pressure results in figure 4.13

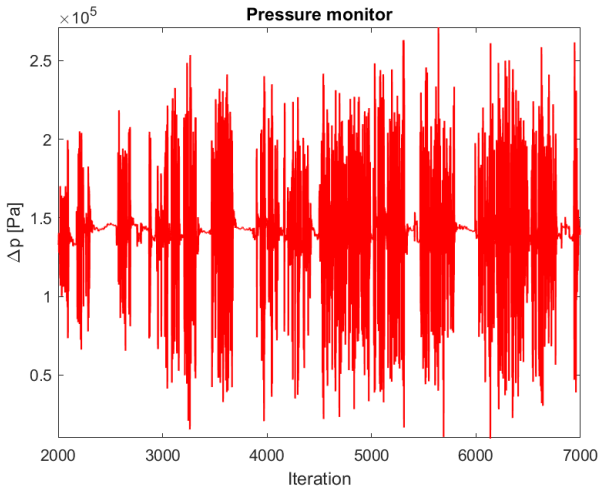
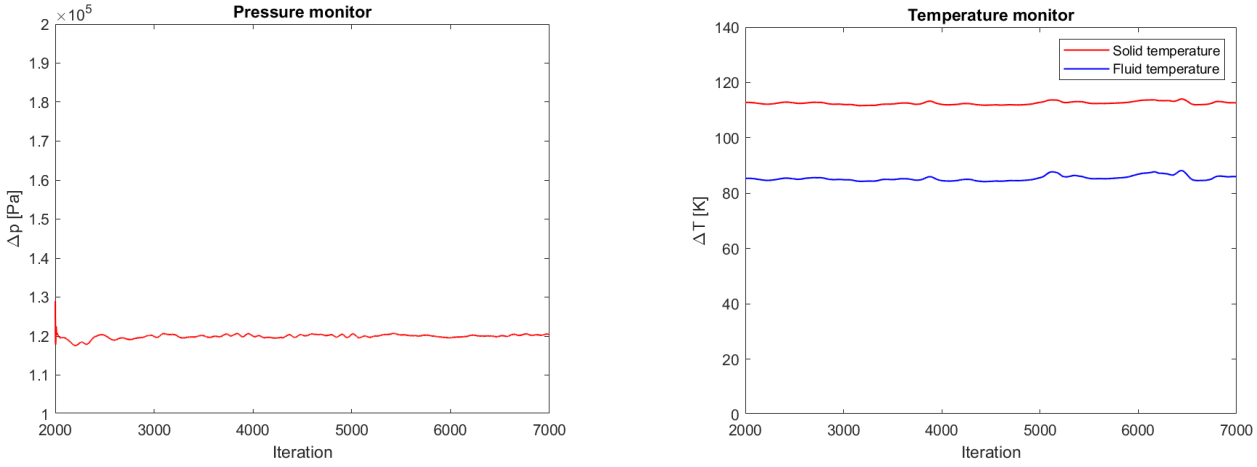


Figure 4.13: Δp [Pa] for 2nd order schemes

As we can see we have high residuals for p introduced by the second order discretization and also a behavior with very high pressure oscillations, to reintroduce stability in our solution we can use a bounded linear-upwind discretization scheme. This type of scheme is related to how the material time derivative is solved, for a generic field e:

$$\frac{De}{Dt} = \frac{\partial e}{\partial t} + \mathbf{U} \cdot \nabla e = \frac{\partial e}{\partial t} + \nabla \cdot (\mathbf{U}e) - (\nabla \cdot \mathbf{U})e$$

The numerical solution gives $\nabla \cdot U = 0$ at convergence, but before the convergence is reached $\nabla \cdot U \neq 0$. For steady state simulations including the third term within a numerical solution gives better convergence and maintains the boundedness of the solution variable. [30] This is done by the bounded variant of the Gauss scheme by automatically including the discretization of the third term with the advection term.



(a) Δp [Pa]

(b) T_W [K]

Figure 4.14: 2nd order linear-upwind discretization schemes results

As we can see from figure 4.23 while we still have oscillations for our results, but they are much less pronounced than the unbounded discretization scheme. Note that the under-relaxation factors

for this case and the following ones are the following:

$$URF_{p_{rgh}} = 0.3 \qquad URF_U = 0.7 \qquad URF_k = 0.7 \qquad URF_\varepsilon = 0.7$$

With the following y^+ values being found on the mesh:

$$y^+_{min} = 0.016 \qquad y^+_{max} = 58.51 \qquad y^+_{avg} = 10.34$$

Heat distribution map

In figure 5.7 we can observe the heat distribution on the tungsten top, as we can see there is a clear heat distribution pattern given by the TPMS geometry. This shows that the convection given by the turbulent flow prevails on conduction and there is a hotspot of around $\approx 425K$.

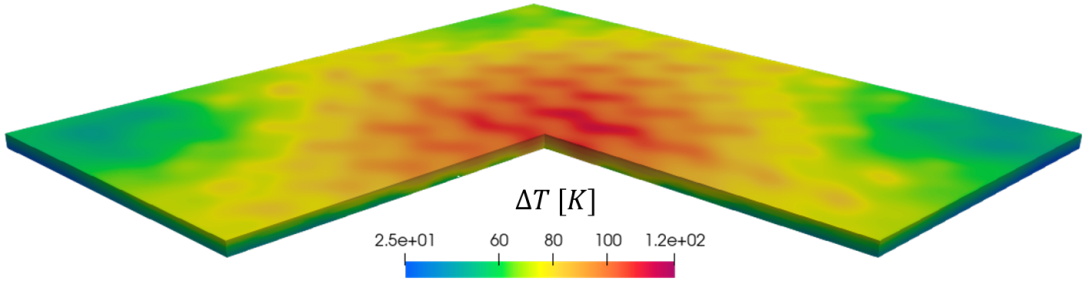


Figure 4.15: Heat distribution in the tungsten top

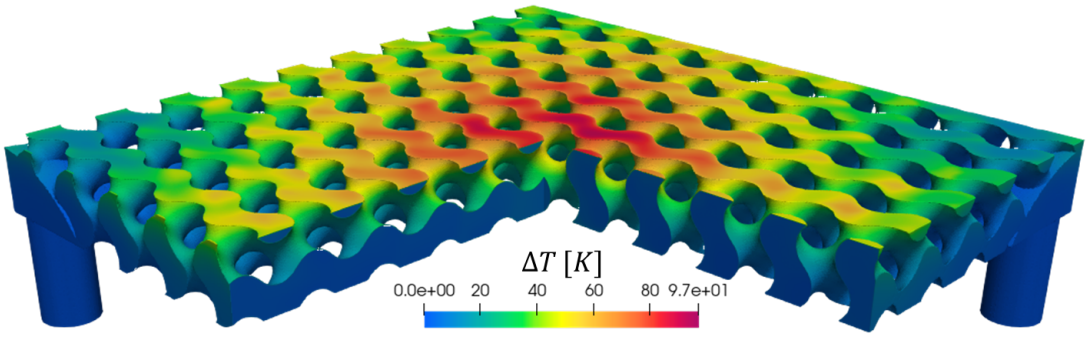


Figure 4.16: Heat distribution in the fluid

The numerical results are the following:

$\mathbf{\Delta T_{max_W}}$	116.258 ± 12.498 K
$\mathbf{\Delta T_{max_f}}$	94.817 ± 11.293 K
$\mathbf{\Delta p}$	123146.523 ± 7167.1 Pa

Table 4.7: Conjugate heat transfer turbulent regime results

In figure 4.26 we can see the flow profile with a linear integral convolution: we can see that in the turbulent case the temperature distribution inside the flow is concentrated mostly on the center

portion, directly below the hot spot on the tungsten plate. The ΔT generally much lower than the one in the solid, this happens because the flow is flowing through the TPMS at an average speed of approximately $0.83 m/s$ in the center section. We can observe the flow behavior in the highlighted sections:

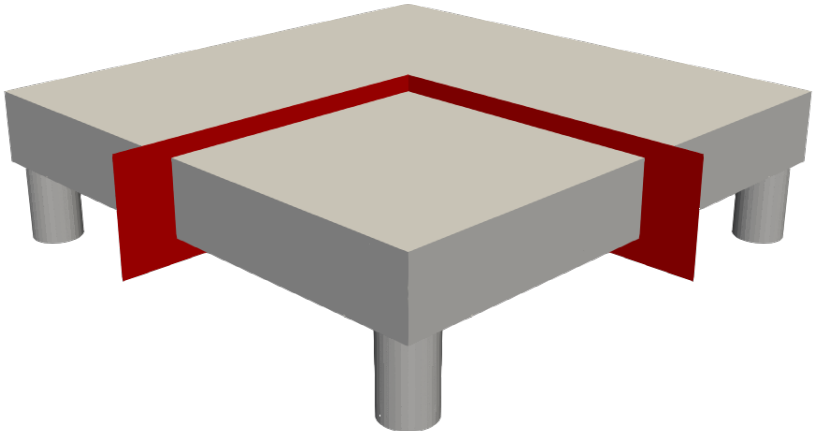


Figure 4.17: Indication of center section

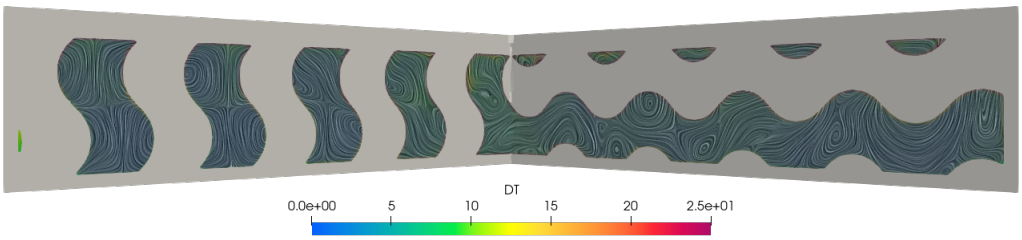


Figure 4.18: Center section LIC of the geometry with temperature distribution

This behavior can also be seen if we take a slice that goes through an inlet-outlet connecting plane:

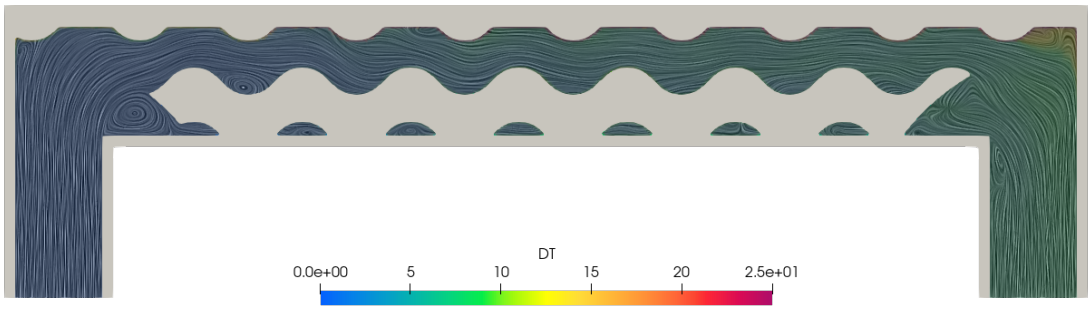


Figure 4.19: Inlet-Outlet LIC of the geometry with temperature distribution

If we compare the temperature profile with the velocities inside the section we can see a direct correlation:

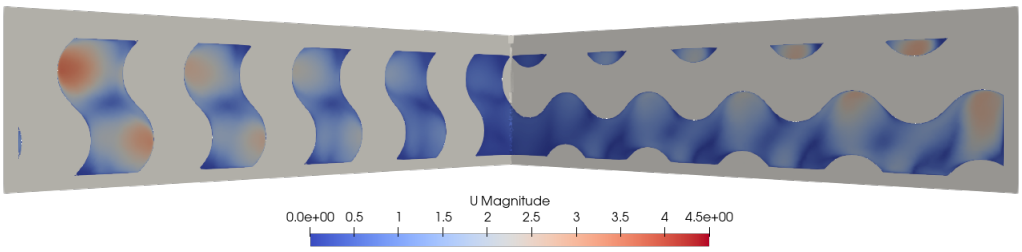
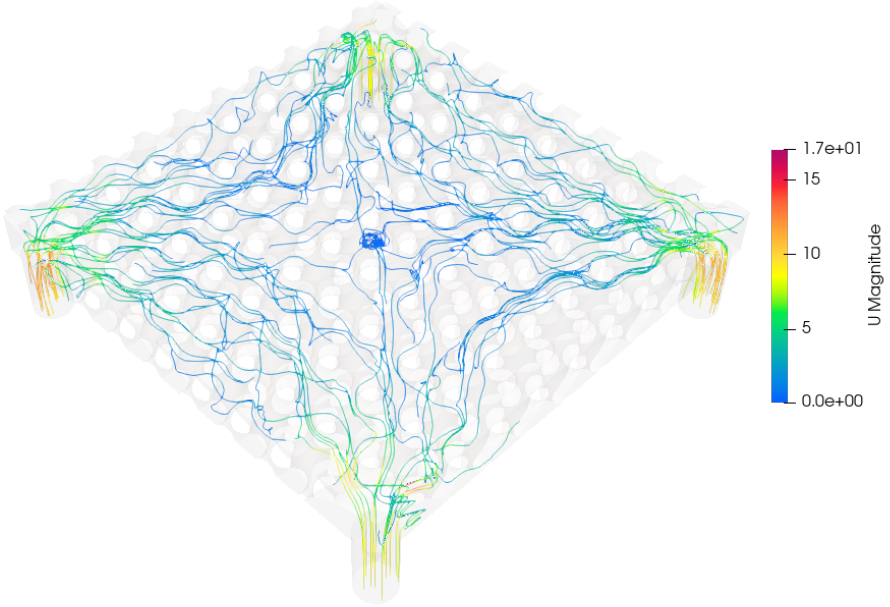
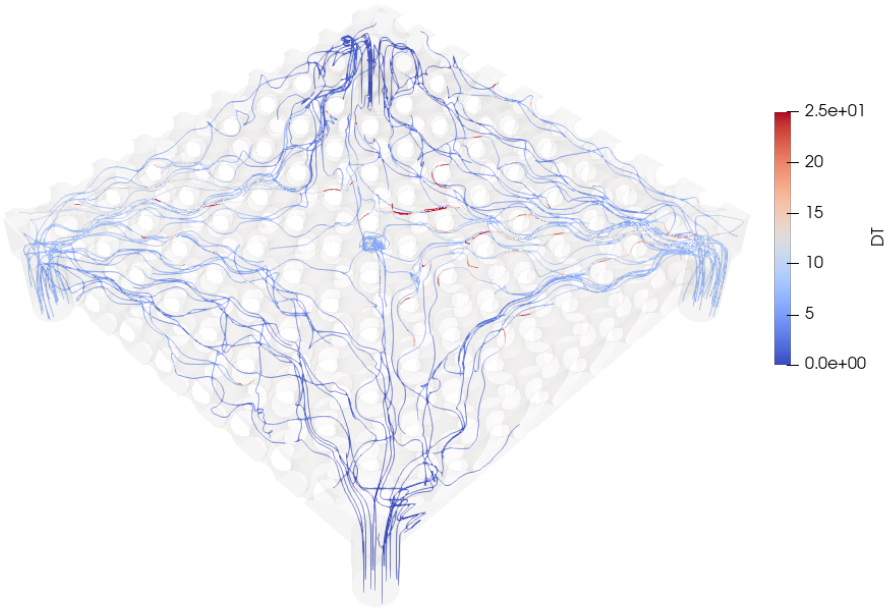


Figure 4.20: Velocity distribution inside the TPMS

As we can see the velocity of the fluid is lower in the center of the TPMS and faster on the side, generating the hotspot we see in figure 5.7.
We can better observe the velocity profile in the TPMS with the temperature in the fluid using the streamlines represented in figure 4.21, these type of representation takes different points and follows the vectors that pass through them. Figure 4.19 once again shows the same concept, as the fluid passes through the geometry the ΔT in the fluid remains relatively low due to the high speeds, we can compare this to the laminar flow in the next section.



(a) Velocity scaled streamlines



(b) Temperature scaled streamlines

Figure 4.21: Streamlines in the fluid region

4.6.2 Laminar Flow with heat transfer

As for the case with laminar flow, since there is no turbulence and the velocities involved are much lower we get lower residuals overall and no oscillation on the solution:

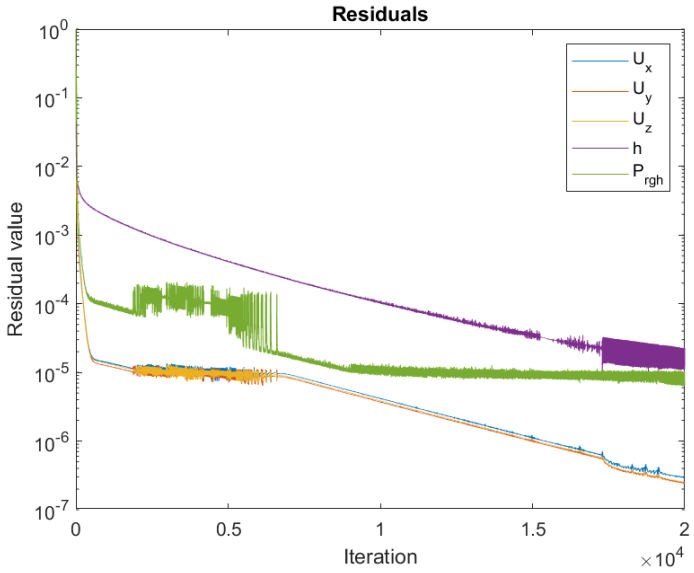
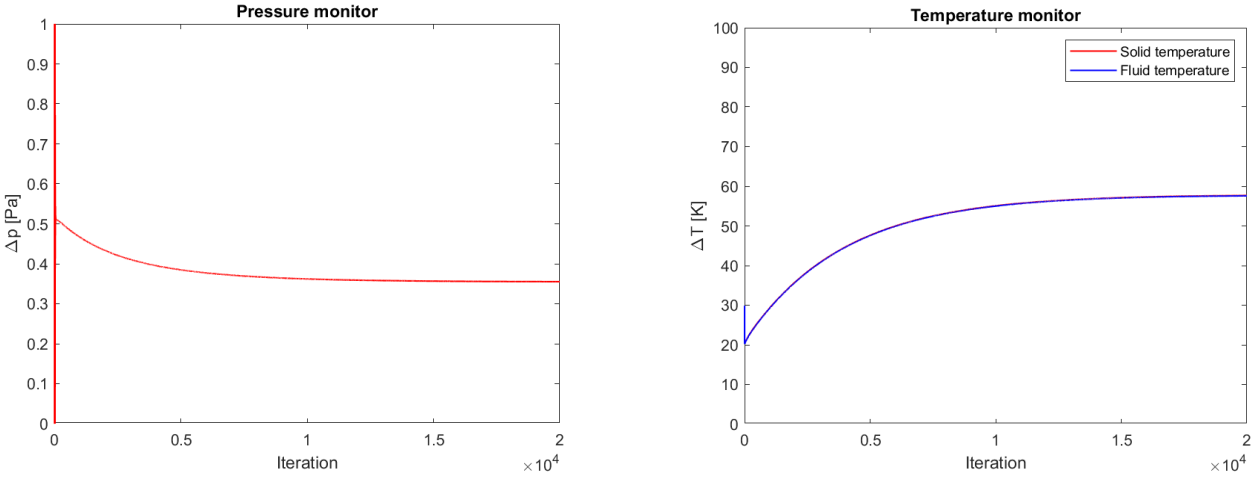


Figure 4.22: Residuals for bounded 2nd order linear-upwind discretization schemes



(a) Δp [Pa]

(b) T_W [K]

Figure 4.23: 2nd order linear-upwind discretization schemes results

Heat distribution map

For the laminar case velocities are much lower, this diminishes the convection heat transfer and conduction dominates, making the heat profile much more evenly spread compared to the turbulent case.

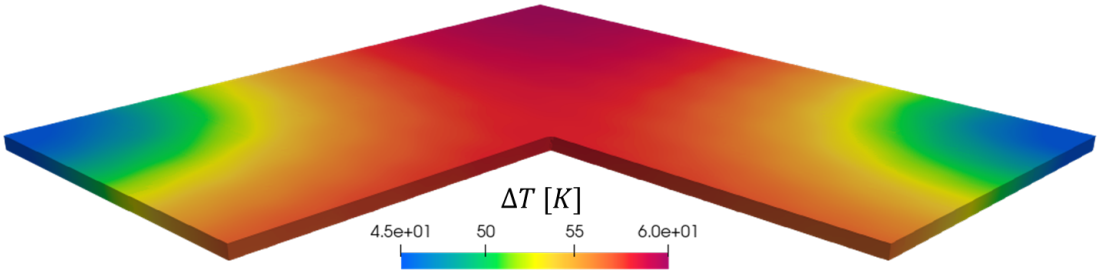


Figure 4.24: Heat distribution in the Tungsten top

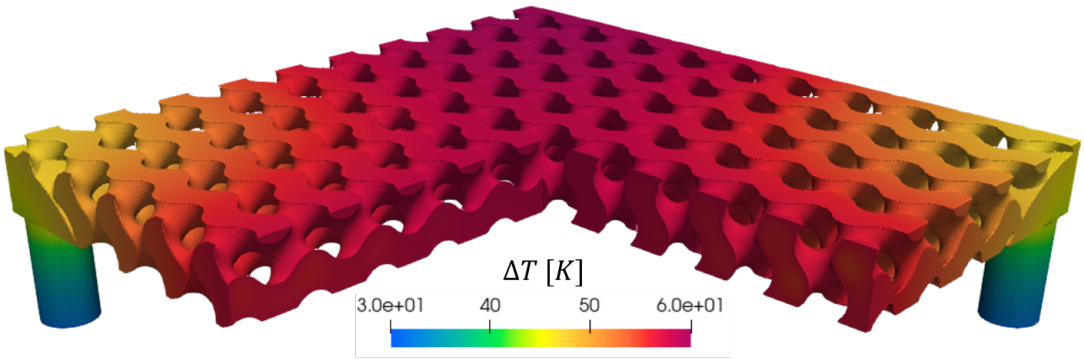


Figure 4.25: Heat distribution in the fluid region

The numerical results are the following:

$\mathbf{\Delta T_{max_W}}$	59.73 ± 2.7583 K
$\mathbf{\Delta T_{max_f}}$	59.66 ± 2.7791 K
$\mathbf{\Delta p}$	0.3553 ± 0.0581 Pa

Table 4.8: Conjugate heat transfer laminar regime results

As we can see, given the much lower heat load applied temperatures are generally lower and in general the results have a lower numerical error. This is given by the fact that the solver ignores the turbulent variables equations and also the speed of the flow is lower and this yields more stable results.
We can observe the internal flow pattern and temperature as done in the turbulent case:

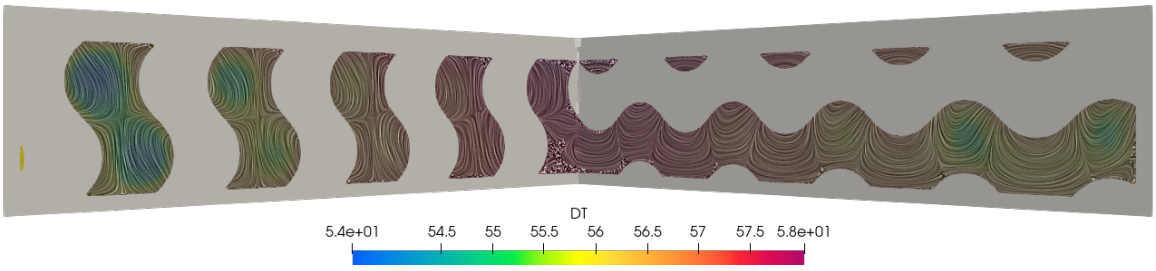


Figure 4.26: Center section LIC of the geometry with temperature distribution

Once again we see direct correlation between temperature and velocity

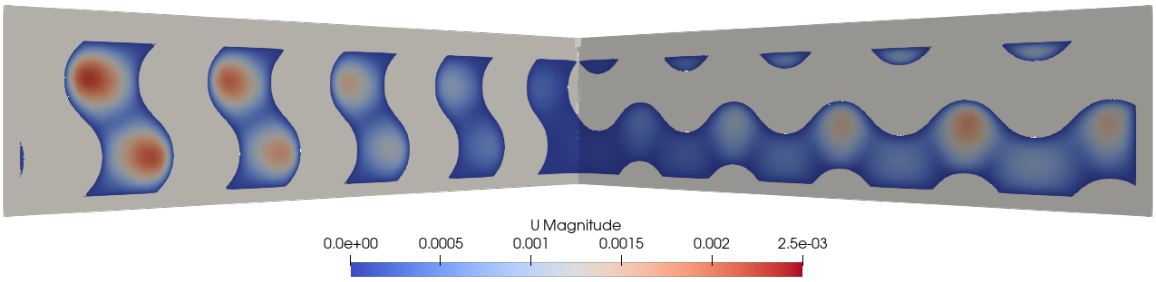


Figure 4.27: Velocity in the center section

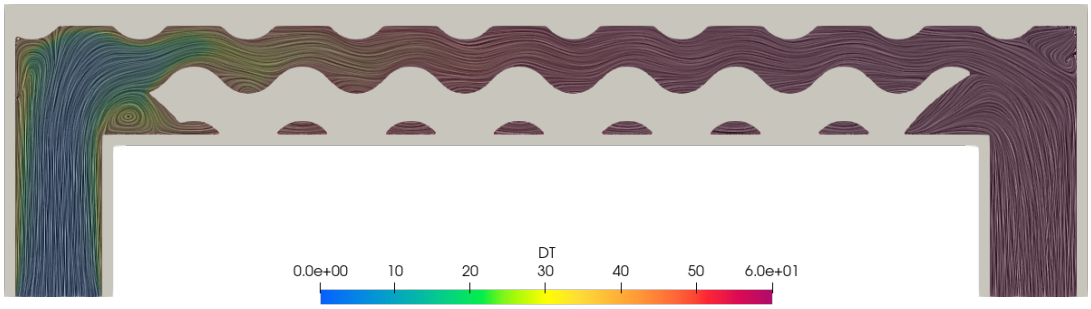


Figure 4.28: Inlet-Outlet LIC of the geometry with temperature distribution

4.6.3 Incompressible flow

To use the same mesh doing the code to code benchmark the simpleFoam incompressible solver is used, this follows the same SIMPLE algorithm described in section 3.2.1. This is done because OpenFOAM doesn't handle polyhedral meshes well and heat transfer makes the problem unstable,

so a simpler, incompressible flow solver is used. For the incompressible case using the STAR-CCM+ mesh we have higher residuals overall and even for bounded second order schemes we have high oscillations in the pressure result. The schemes being used involve the cell and face limiters discussed in section 3.3, with high blending factors specified by the following entries in the scheme dictionary:

```
gradSchemes
{
    default cellLimited Gauss linear 0.5;
    grad(U) faceLimited Gauss linear 1.0;
}
```

Diffusive terms also use a blending factor and limited scheme:

```
laplacianSchemes
{
    default  Gauss linear limited 0.333;
}
snGradSchemes
{
   default limited 0.333;
}
```

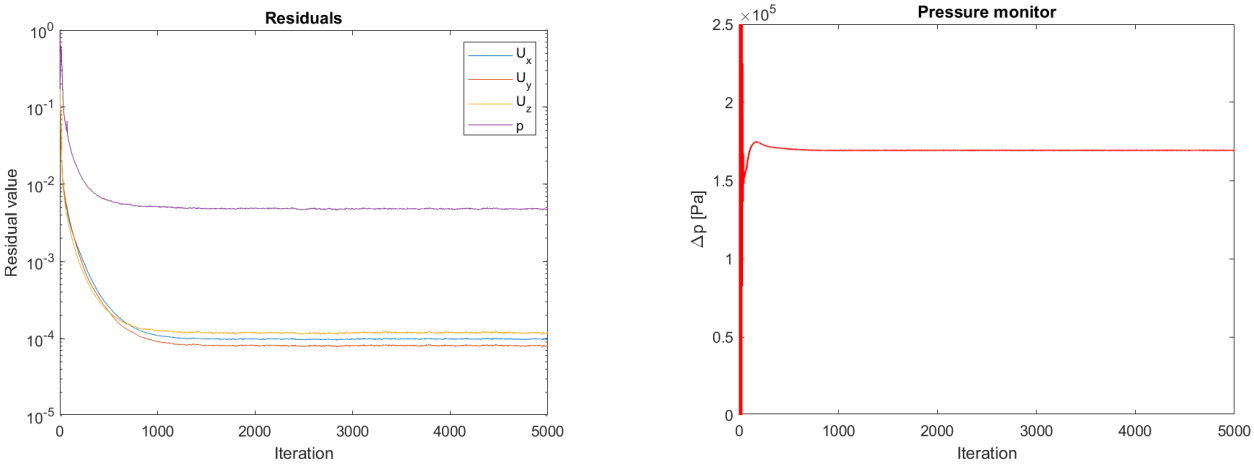
The pressure variable for the incompressible solver "simpleFoam" is intended as kinematic pressure [41], ρ for water at 300K and 7.5 bar is $\rho = 996.85 kg/m^3$ so at the outlet the kinematic pressure is set as:

$$p_k = \frac{p_s}{\rho} = \frac{750000\,[Pa]}{996.85\,[kg/m^3]} = 752.37[m^2/s^2]$$

So the interpret the results from the incompressible solver they should be multiplied for the constant density.
In this section only the stability of the solver will be discussed, the results will be shown in a later section along with the other codes.

Turbulent case

In the turbulent case residuals are high and the pressure difference oscillates, so we use bounded schemes once again.



(a) Residuals (b) $\Delta p[m^2/s^2]$

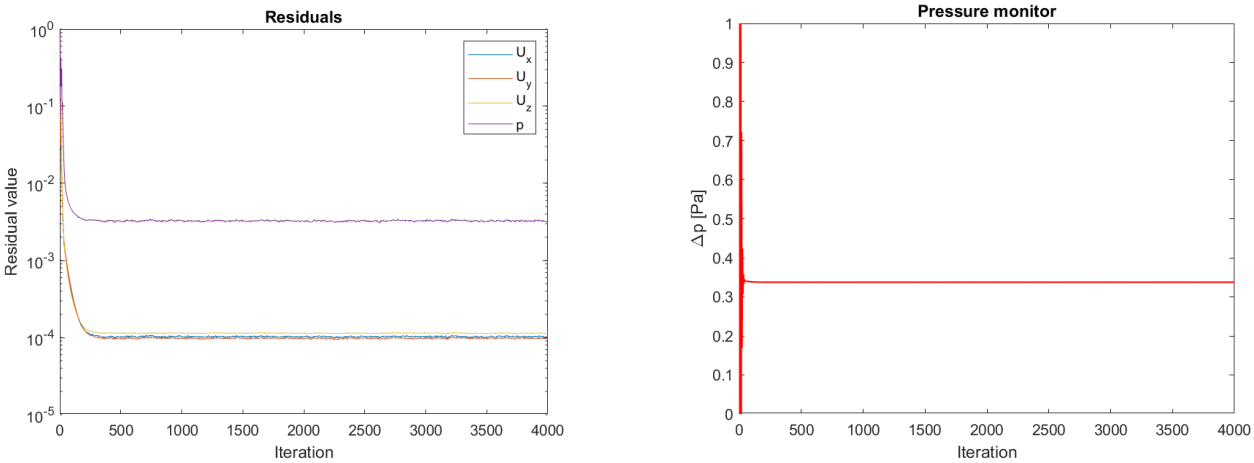
Figure 4.29: Results for turbulent flow using second order bounded schemes

Since the STAR-CCM+ generated mesh has boundary layers the average y^+ value is lower:

$$y^+_{min} = 5 \cdot 10^{-5} \qquad y^+_{max} = 57.48 \qquad y^+_{avg} = 1.29$$

Laminar case

For the laminar flow once again we see that the results have better convergence and are less oscillatory:



(a) Residuals (b) Δp

Figure 4.30: Results for laminar flow using second order bounded schemes

Code to code benchmark

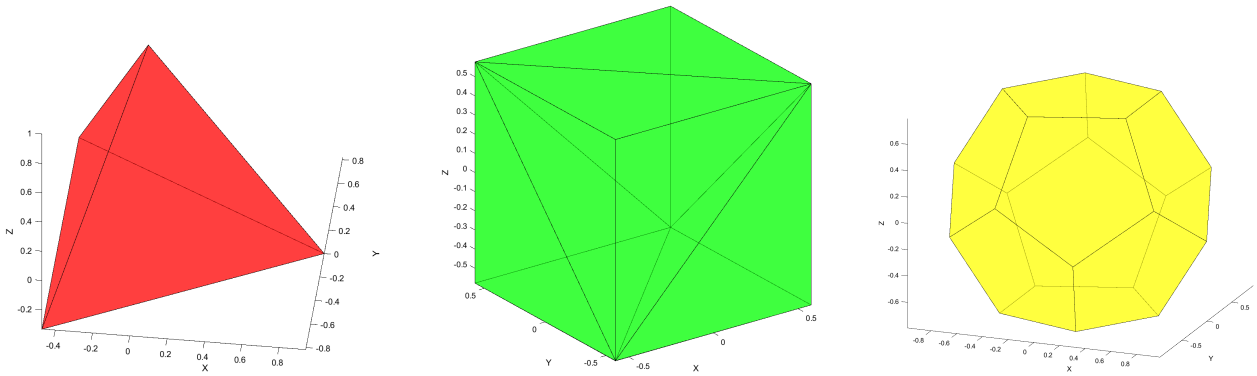
In the following chapter we will compare the results of different software on the same and on their native grid. In our case we will be pitting OpenFOAM against Siemens Star-CCM+ and Ansys Fluent, both commercial softwares.

5.1 Star-CCM+

Star-CCM+ is a commercial CFD software developed by Siemens, the whole workflow, from stl management to meshing and solving, can be handled on Star-CCM+. We are going to use the software to generate polyhedral meshes of our geometry which we're going to use in the other softwares to do a code-to-code benchmark.

5.1.1 Polyhedral meshes

A polyhedral mesh is a type of mesh that uses polyhedra instead of the more traditional hexahedral and tetrahedral meshes. Hexahedral type meshes achieve results with lower diffusion and also a lower discretization error but can generate problems and actually increase the error when the flow isn't perpendicular to the cell faces. [20] Tetrahedral meshes solve these problems, with their main advantage being that the meshing process can be mostly automated, the problem in this case is the high number of elements given the shape of the cell:



(a) Tetrahedral cell (b) Hexahedral cell (c) Polyhedral cell

Figure 5.1: Different types of cell elements

So to combine the best of both worlds polyhedral meshes have been devised. In these type of meshes the cells have many neighbors, this allows them to estimate gradients more accurately. Unfortunately these type of meshes also have high non-orthogonality which is well managed by commercial softwares which use high diffusivity schemes, but is not handled well by OpenFOAM,

causing high numerical errors and simulation divergence, so we are going to run a incompressible simulation and compare the pressure difference to simplify the problem.

5.1.2 Mesh generation

The mesh we are going to be using for the code to code benchmark is a polyhedral mesh generated by STAR-CCM+, with the following characteristics:

- Number of cells: 16.5 Million
- Core flow average cell size: 0.6 mm
- Prism layers: 10
- First layer thickness: 0.01 mm

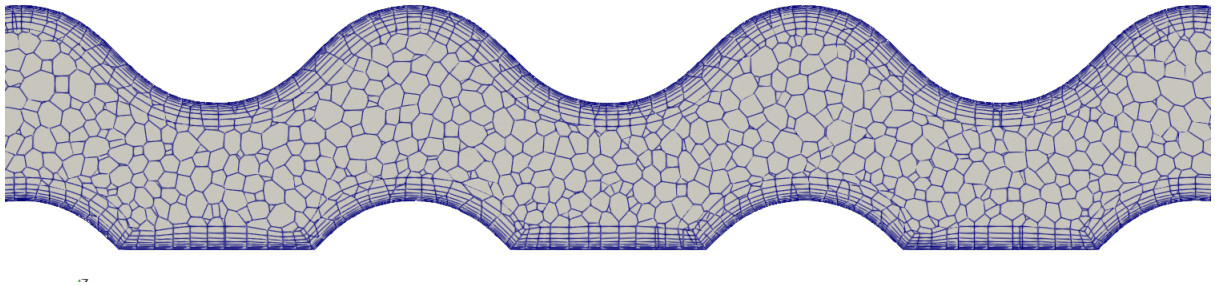


Figure 5.2: Polyhedral mesh generated by Star-CCM+

5.1.3 Solvers and models

The Star-CCM+ algorithm for solving the conjugate heat transfer has two different model references that solve the energy equation:

$$\frac{\partial}{\partial t}\int \rho E dV + \oint_A \rho H \mathbf{v}\cdot d\mathbf{a} = \underbrace{-\oint_A \dot{\mathbf{q}}''\cdot d\mathbf{a}}_{\text{Conduction}} + \underbrace{\oint_A \mathbf{T}\cdot\mathbf{v} d\mathbf{a}}_{\text{Viscous Work}} + \int_V \mathbf{f}_b\cdot\mathbf{v}dV + \oint_A \sum_i h_i J_i d\mathbf{a} + \int_V S_u dV$$

where:

- E is the total energy
- H is the total enthalpy
- $\dot{q}''$ is the heat flux vector
- T is the viscous stress tensor
- v is the velocity vector
- f_b is the body force vector representing the combined body forces.
- h_i is the enthalpy of component i
- J_i is the diffusive flux of component i
- S_u is the source term

Segregated energy model

STAR-CCM+ has three different segregated fluid energy models which are companions to the segregated flow model:

- Segregated fluid enthalpy Solves the total energy equation

$$E = H - \frac{p}{\rho}$$

  where $H = h + |\mathbf{v}|^2/2$ and h is the static enthalpy. The equation is solved for the chemical thermal enthalpy and the the temperature is computed from enthalpy according to the equation of state. Ideal for cases that involve combustion.

- Segregated fluid Temperature This solves again the total energy equation but the temperature is the solved variable which is then used to compute the enthalpy with the equation of state.

- Segregated fluid isothermal This model keeps the temperature constant, ideal for small temperature variations.

The solver is a segregated energy solver that controls the solution update for the chosen segregated model.

Coupled energy model reference

The coupled energy model uses a coupled solver for steadt state simulations. If we write the governing equations in integral form for a volume V and a differential surface area $d\mathbf{a}$ we get:

$$\frac{\partial}{\partial t}\int_V \mathbf{W}dV + \oint [\mathbf{F} - \mathbf{G}] \cdot d\mathbf{a} = \int_V \mathbf{H}dV$$

Where the vectors are:

$$\mathbf{W} = \begin{bmatrix} \rho \\ \rho\mathbf{V} \\ \rho E \end{bmatrix}$$

$$\mathbf{F} = \begin{bmatrix} \rho\mathbf{v} \\ \rho\mathbf{v}\mathbf{v} + p\mathbf{I} \\ \rho\mathbf{v}H + p\mathbf{v} \end{bmatrix}$$

$$\mathbf{G} = \begin{bmatrix} 0 \\ \mathbf{T} \\ \mathbf{T}\cdot\mathbf{v} + \dot{\mathbf{q}} \end{bmatrix}$$

$$\mathbf{H} = \begin{bmatrix} S_u \\ \mathbf{f}_r + \mathbf{f}_g + \mathbf{f}_p + \mathbf{f}_u + \mathbf{f}_\omega + \mathbf{f}_L \\ S_u \end{bmatrix}$$

STAR-CCM+ introduces a preconditioning matrix $\mathbf{\Gamma}$ that multiplies the transient term:

$$\Gamma \frac{\partial}{\partial t} \int_V \mathbf{W} dV + \oint [\mathbf{F} - \mathbf{G}] \cdot d\mathbf{a} = \int_V \mathbf{H} dV$$

Which is:

$$\Gamma = \begin{bmatrix} \theta & 0 & \rho_T \\ \theta \mathbf{v} & \rho \mathbf{I} & \rho_T \mathbf{v} \\ \theta H - \delta & \rho \mathbf{v} & \rho_T H + \rho C_p \end{bmatrix}$$

Which uses $\rho_T = \frac{\partial \rho}{\partial T}\Big|_p$ and $\theta = \frac{1}{U_r^2} - \frac{\rho_T}{\rho C_p}$ where U_r is a reference velocity that is chosen such that the eigenvalues of the system remain well conditioned. If we apply the governing equation to a cell centered control volume the following discretized system is obtained:

$$V_0 \, \Gamma_0 \frac{\partial \mathbf{Q}_0}{\partial t} + \sum_f (\mathbf{f}_f - \mathbf{g}_f) \cdot \mathbf{a} = \mathbf{h} V_0$$

where the sum is over the total number of faces of the cell, $\mathbf{f}_f$ is the convective flux and $\mathbf{g}_f$ is the diffuse flux through the face. V_0 is the cell volume and $\mathbf{\Gamma}_0$ is the preconditioning matrix for the cell. Then for steady state simulation the coupled system of equations is discretized in time and time-stepping is performed until a quasi steady-state solution is obtained.

5.1.4 Turbulence model

For the turbulence model we will be using the realizable k-epsilon turbulence model for consistency with the other softwares. Star-CCM+ uses a two layer approach, devised by Rodi [7] that enables the application of the K-Epsilon model to the viscous-affected layer. The two layers used in the method are the following:

- Layer next to the wall in which the turbulent viscosity μ_t and the turbulent dissipation rate ε are specified as functions of wall distance.
- Layer far from the wall in which the equation for the turbulent dissipation rate is solved.

The values of ε specified in the near wall layer are blended smoothly with the values computed from solving the transport equation far from the wall. The equation for the turbulent kinetic energy k is solved across the whole domain.
The realizable two layer approach combines the realizable K-epsilon model, which contains a new transport equation for the turbulent dissipation rate ε and a variable damping function, and the two layer approach.

5.1.5 Results

For this simulation we are going to use the same boundary conditions as the OpenFOAM case, that means a laminar and a turbulent case with different mass flow and heat flux:

	Laminar	**Turbulent**
$\dot{m}\,[kg/s]$	0.0004	0.833
$\dot{Q}\,[kW/m^2]$	10	2500

Table 5.1: Mass flow and heat flux boundary conditions

Turbulent case with heat transfer

For STAR-CCM+ second order upwind discretization schemes for convection and a first order central scheme for the diffusion scheme are being used. The realizable two-layer $k-\varepsilon$ model will be used for turbulence modeling. In the following heat maps we can see the differences between the turbulent and laminar case:

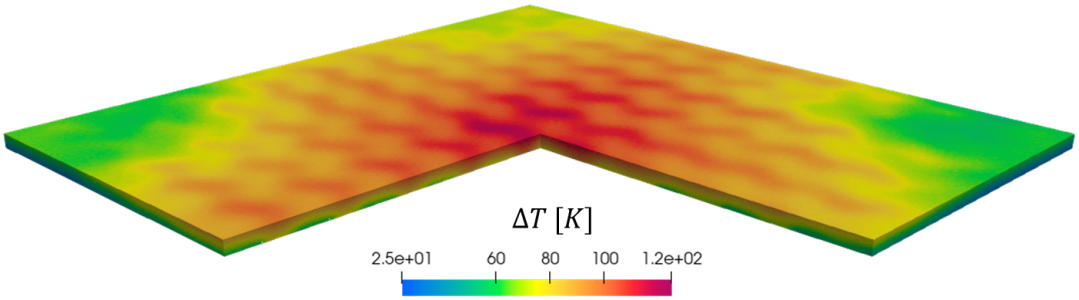


Figure 5.3: Heat distribution in the Tungsten top

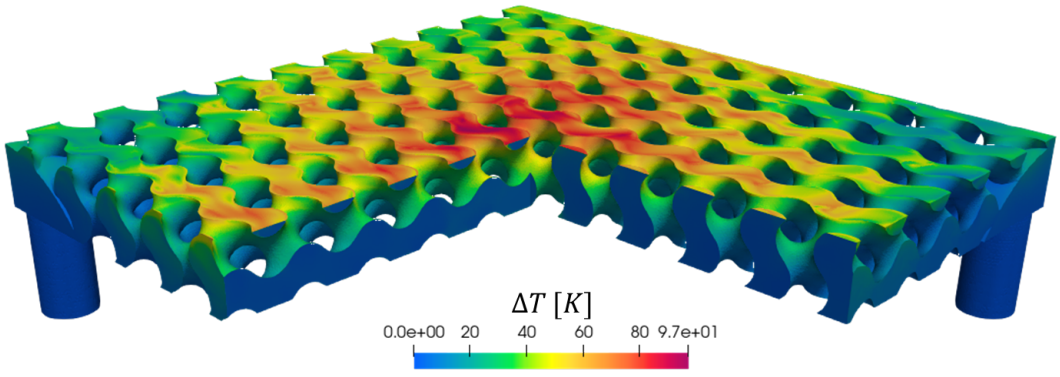


Figure 5.4: Heat distribution in the fluid

The results are the following:

$\mathbf{\Delta T_{max_W}}$	125.163 ± 2.7583 K
$\mathbf{\Delta T_{max_f}}$	102.994 ± 15.799 K
$\mathbf{\Delta p}$	144622 ± 4078.3 Pa

Table 5.2: Conjugate heat transfer turbulent regime results

Generally residuals and stability are much better than OpenFOAM, and we can see that the results differ, this is mainly given by the difference in mesh type, as said before the polyhedral mesh yields more diffusive results. Just as the OpenFOAM simulation we see the same behavior, with temperatures being more uniform in the laminar case. We can also observe the behavior in the center section:

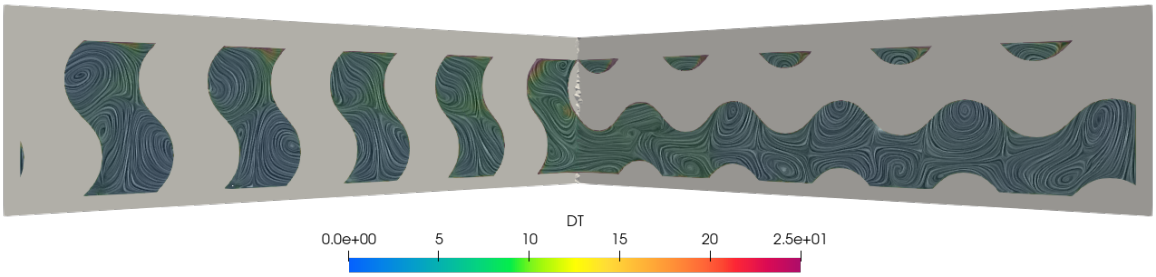


Figure 5.5: Center section LIC of the geometry with temperature distribution

Once again, the high flow speed inside the TPMS allows the temperature in the fluid to be lower overall.

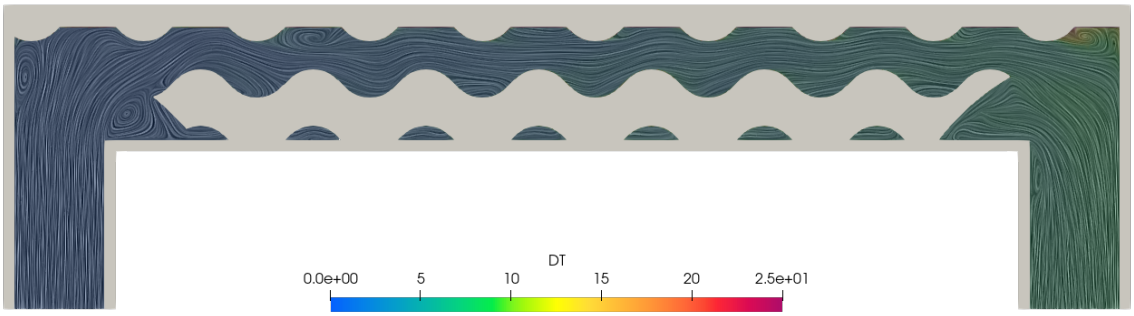


Figure 5.6: Inlet-Outlet LIC of the geometry with temperature distribution

Laminar case with heat transfer

For low speed flows the laminar solver has been used, ignoring the turbulent variables. The lower speeds make a bigger number of iterations to reach the steady-state solution.

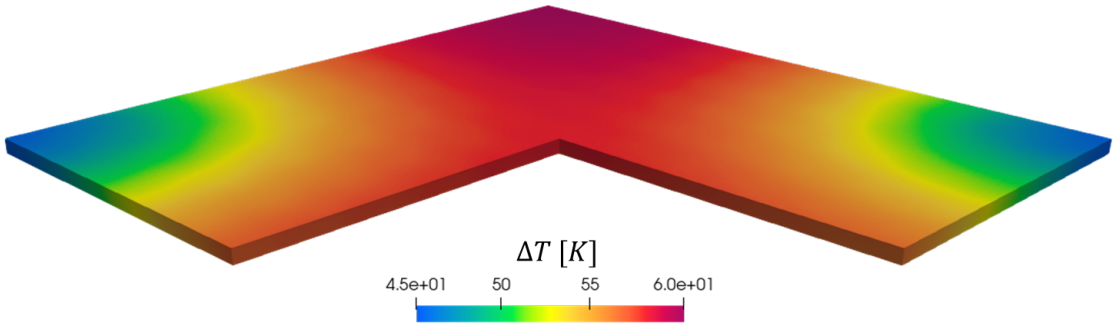


Figure 5.7: Heat distribution in the Tungsten top

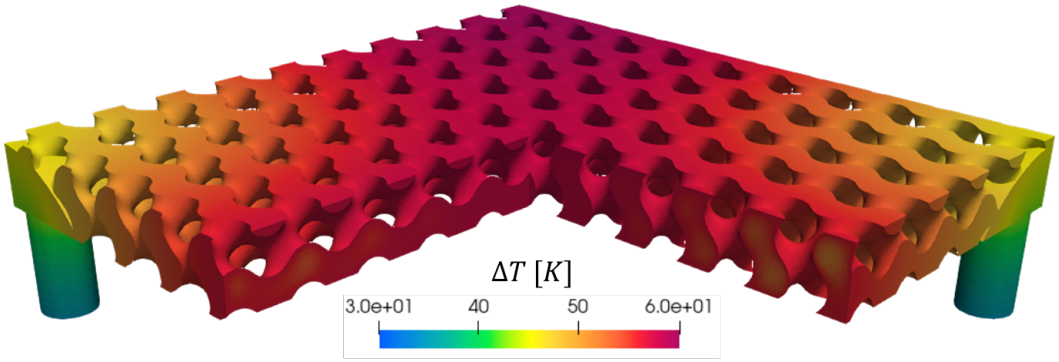


Figure 5.8: Heat distribution in the fluid

And in figure 5.9 and 5.10 we can appreciate the same behavior seen in OpenFOAM also for the laminar flow.

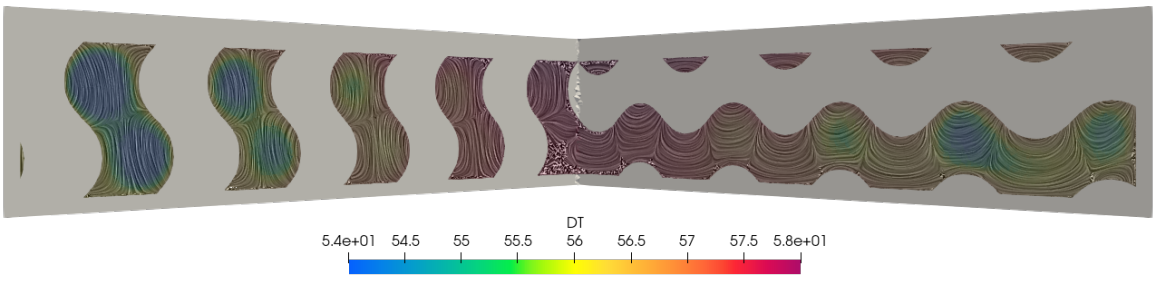


Figure 5.9: Center section LIC of the geometry with temperature distribution

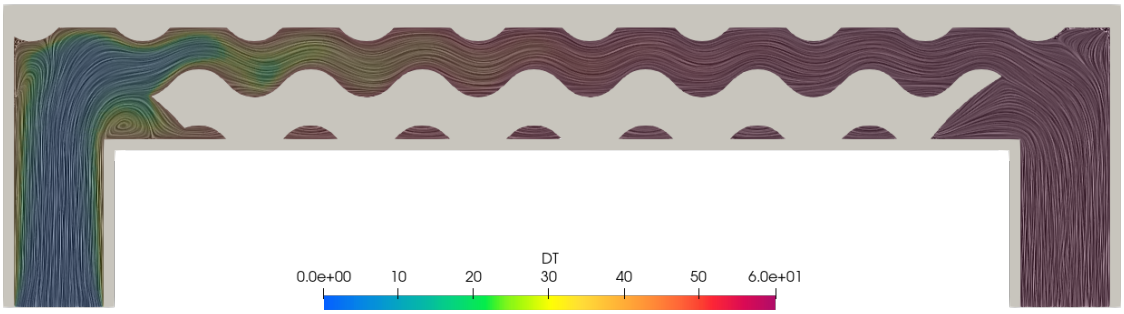


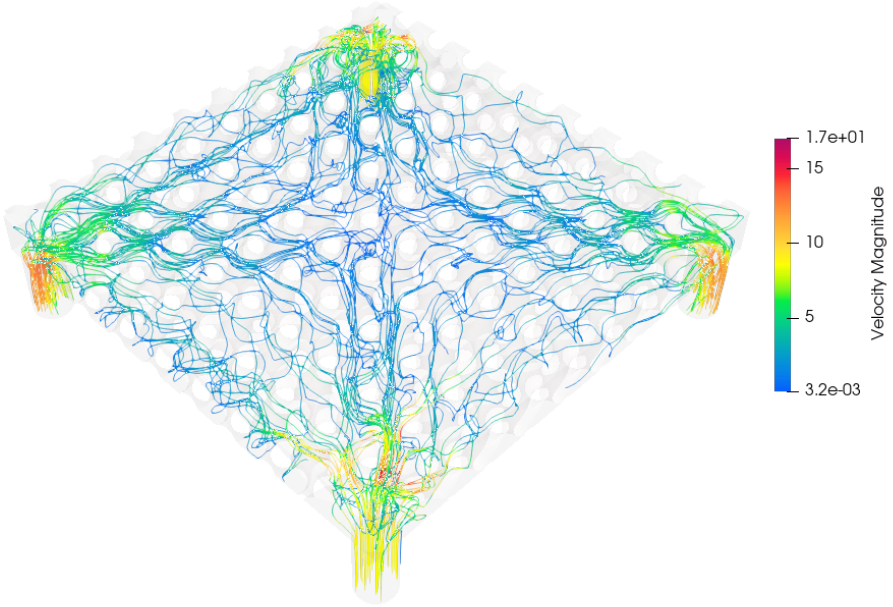
Figure 5.10: Inlet-Outlet LIC of the geometry with temperature distribution

Here the maximum ΔT between the solid and the fluid is similar:

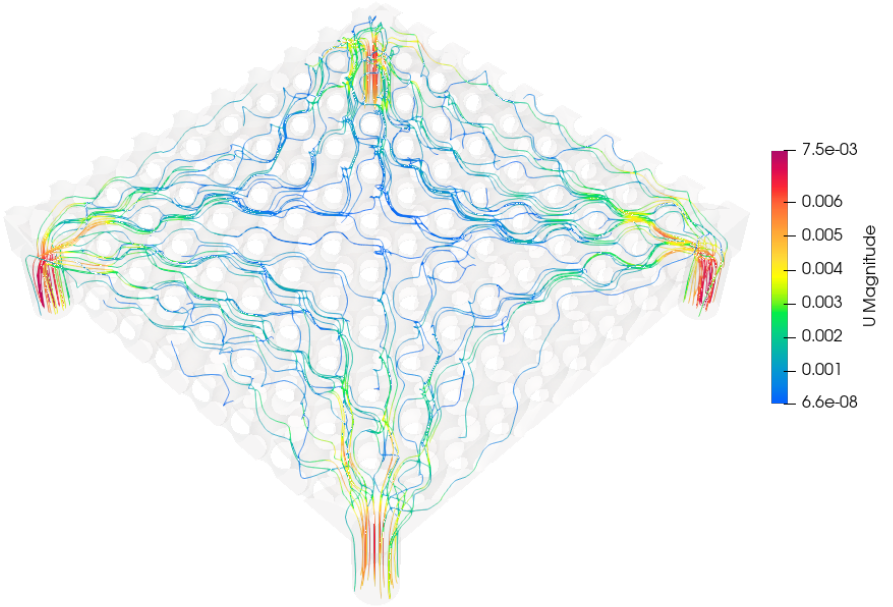
$\mathbf{\Delta T_{max_W}}$	60.234 ± 0.1747 K
$\mathbf{\Delta T_{max_f}}$	60.177 ± 0.0602 K
$\mathbf{\Delta p}$	0.3717 ± 0.0118 Pa

Table 5.3: Conjugate heat transfer turbulent regime results

As we can see we have very small numerical errors at this mesh size, given by the fact that the simulation is more stable. We can compare the streamlines between the laminar and compressible case in figure 5.11 to see how the velocity profile differs. We can see the how in both cases the current permeates the whole TPMS geometry, with lower velocities toward the center of the geometry in both cases. This happens when the two inlet flows collide at the center and get diverted at the outlets.



(a) Streamlines for the turbulent flow



(b) Streamlines for the laminar flow

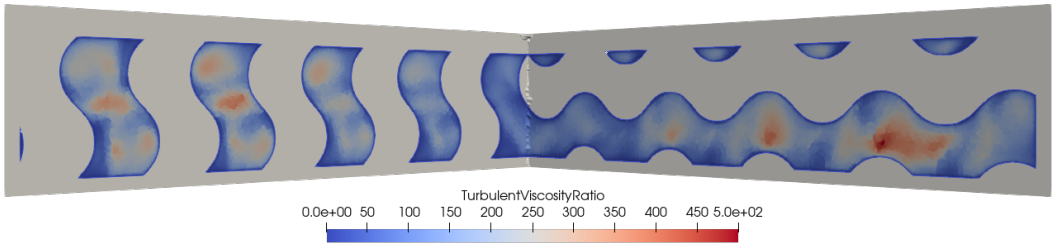
Figure 5.11: Streamlines in the fluid region

5.1.6 Incompressible flow

To achieve a better comparison an incompressible flow model without heat transfer has been implemented, this is necessary because the polyhedral meshes generated by Star-CCM+ cause simulation divergence in other codes and thus a simpler, more stable simulation is included.

5.1.7 Turbulent case

In the turbulent case we can observe the results for velocity and viscosity ratio:



(a) Turbulent viscosity ratio in center section

(b) Velocity in center section

Figure 5.12: STAR-CCM+ Turbulent case results

The average velocity in the center section is around $0.88m/s$, which is very similar to the one found for the compressible case. The viscosity ratio given by ν_t/ν is relatively low, with values under 500, this indicates that the flow has a low turbulence.

5.1.8 Laminar case

In STAR-CCM+ the average velocity in the center section for the laminar case is $1.81565 \cdot 10^{-4} m/s$, which is generally lower than the averages found in the other softwares.

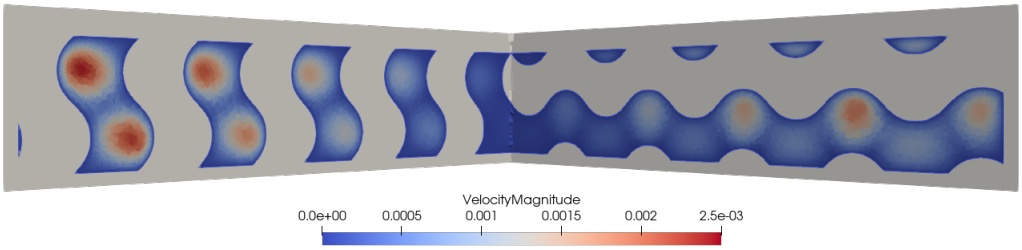


Figure 5.13: Velocity profile in laminar flow

5.2 Fluent

Fluent is a commercial CFD software developed in the 1980s and acquired by Ansys in 2006, it was the first CFD software with a GUI.

5.2.1 Solvers and models

Fluent can use a density based solver or pressure based solver. For the incompressible case we are going to use the pressure based solver. This solver is based on a control volume where the general scalar transport equation is converted to an algebraic equation that can be solved numerically. For the arbitrary control volume V the following transport equation for the generic scalar quantity ϕ is written in integral form:

$$\int_V \frac{\partial \rho\phi}{\partial t}\, dV + \oint \rho\phi\, \vec{v} \cdot d\vec{A} = \oint \Gamma_\phi\, \nabla\phi \cdot d\vec{A} + \int_V S_\phi\, dV$$

Which is applied to every control cell in the computational domain. Discretization on a cell gives the following equation:

$$\frac{\partial \rho\phi}{\partial t}\, V + \sum_f^{N_{\text{theos}}} \rho_f \vec{v}_f \phi_f \cdot \vec{A}_f = \sum_f^{N_{\text{theos}}} \Gamma_\phi\, \nabla\phi_f \cdot \vec{A}_f + S_\phi\, V$$

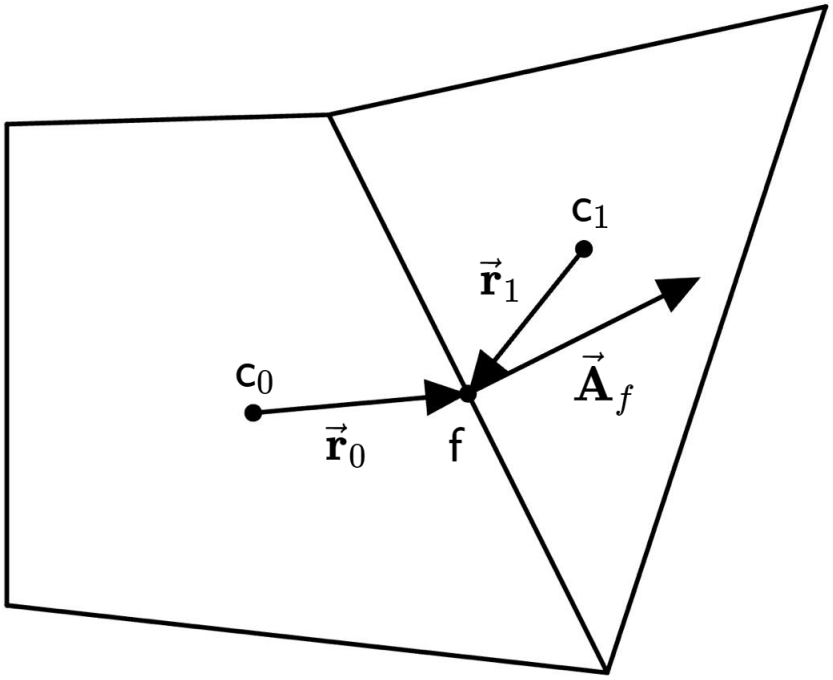


Figure 5.14: Control volume in which the discretized transport equation is applied

An under-relaxation factor *alpha* is used to control the change of the variables. This factor alters the computed change in the generic variable ϕ as follows:

$$\phi = \phi_{old} + \alpha \Delta \phi$$

The transport equations are also under relaxed introducing selective amounts of ϕ in the system of discretized equations:

$$\frac{a_p \phi}{\alpha} = \sum_{nb} a_{nb} \phi_{nb} + b + \frac{1-\alpha}{\alpha} a_p \phi_{\text{old}}$$

Pressure-based Segregated algorithm

This solver uses a sequential algorithm where the equations are solved one after the other. The governing equations are non linear and coupled so the solution loop must be carried out iteratively. In figure 5.15 the segregated algorithm is shown, and we can see that we solve the $U_{vel}, V_{vel}, W_{vel}$ momentum equations one after the other, then we can solve for the pressure correction equation using the velocity field and the mass flux.

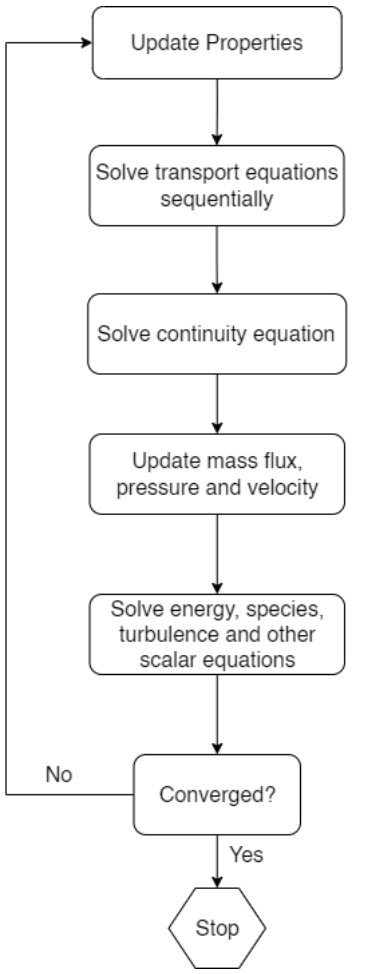


Figure 5.15: Pressure based segregated algorithm

Pressure-based Coupled Algorithm

There is also a coupled algorithm that solves the momentum and continuity equations at the same time and then proceeds like the segregated algorithm. The convergence is better than the segregated method but the memory requirement increases by 1.5-2 times since the discrete system of all momentum and pressure based equations needs to be stored in the memory when solving the pressure and velocity field. [38]

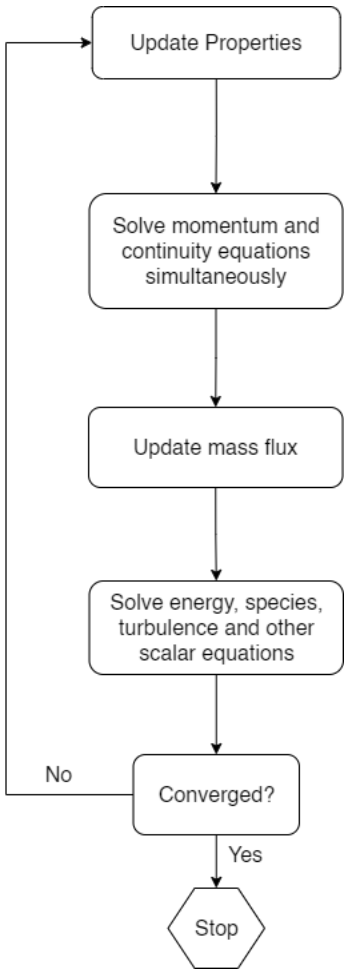


Figure 5.16: Pressure based coupled algorithm

Turbulence model and wall functions

Ansys is once again set to utilize the realizable $k-\varepsilon$. Standard wall functions are used, with the following structure:

$$u^* = y^* \quad for \quad y^* < 11.225$$

$$u^* = \frac{1}{\kappa}\ln(Ey^*) \quad for \quad y^* \geq 11.225$$

Where

$$u^* \equiv \frac{U_P C_\mu^{1/4} k_P^{1/2}}{\tau_w/\rho}$$

$$y^* \equiv \frac{\rho C_\mu^{1/4} k_P^{1/2} y_P}{\mu}$$

For the turbulent variables the k equation is solved in the whole domain, with the boundary condition for k at the wall being:

$$\frac{\partial k}{\partial n} = 0$$

Meanwhile the production of kinetic energy G_k and the turbulent dissipation rate ε are computed with the local equilibrium hypothesis which states that the production of k and its dissipation rate are assumed to be equal in wall-adjacent control volume.
The production of k is based on the log law:

$$G_k = \tau_w \frac{\partial U}{\partial y} = \tau_w \frac{\tau_w}{\kappa \rho k_P^{1/2} y_P}$$

ε is computed in the wall-adjacent cells with the following equation:

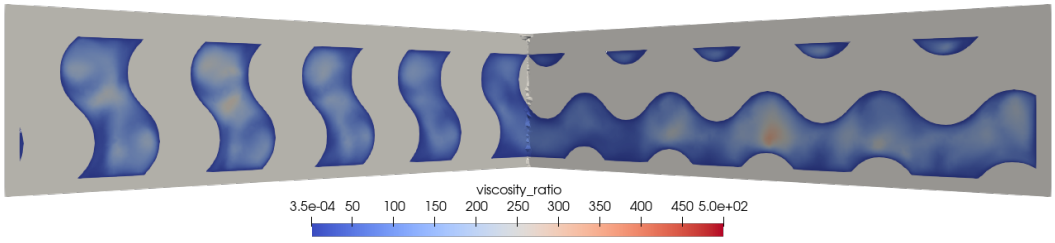
$$\varepsilon_P = \frac{C_\mu^{3/4} k_P^{3/2}}{\kappa y_P}$$

5.3 Results

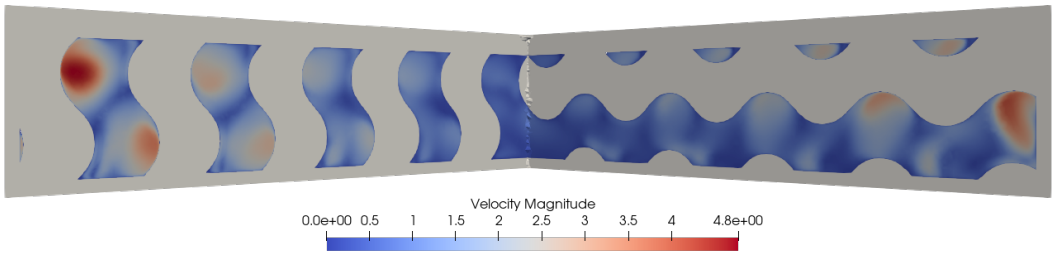
In the following section we will see the results computed by Fluent for the two different cases

5.3.1 Turbulent case

In the turbulent case we can observe the results for velocity and viscosity ratio:



(a) Turbulent viscosity ratio in center section



(b) Velocity in center section

Figure 5.17: Fluent Turbulent case results

The average velocity in the center section is around 0.85, in line with the results found in previous cases. As we can see once again the turbulent viscosity ratio is relatively low and follows

the ones found in the other cases. We can also see the streamlines inside the fluid field in figure 5.18

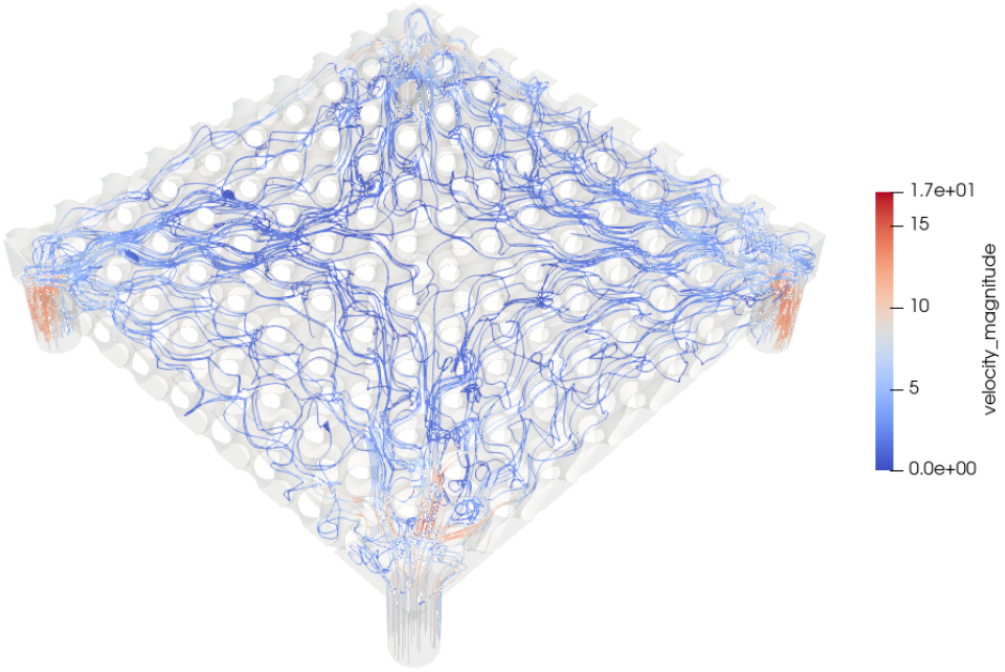


Figure 5.18: Streamlines in the fluid region

5.3.2 Laminar case

For the laminar section we are going to have lower velocities, in Fluent the average velocity in the center section for the laminar case is around $2.2 \cdot 10^{-4} m/s$.

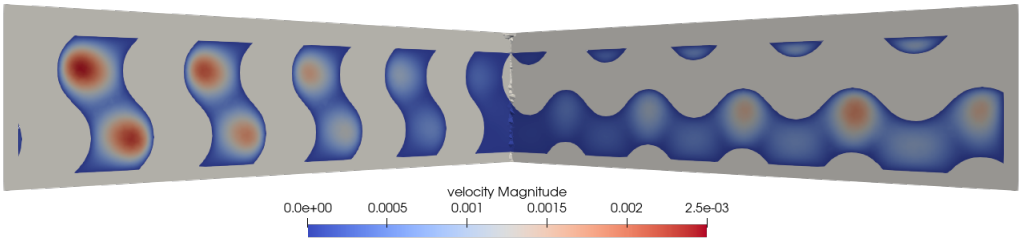


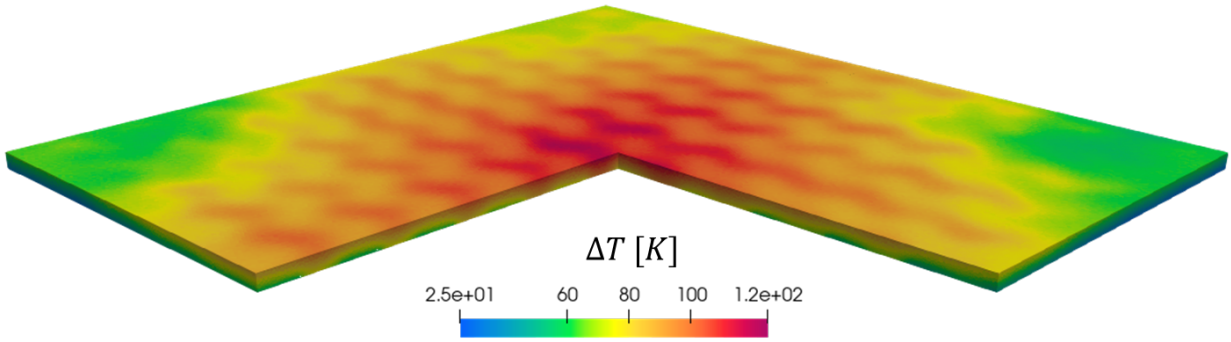
Figure 5.19: Velocity profile in laminar flow

5.4 Comparison

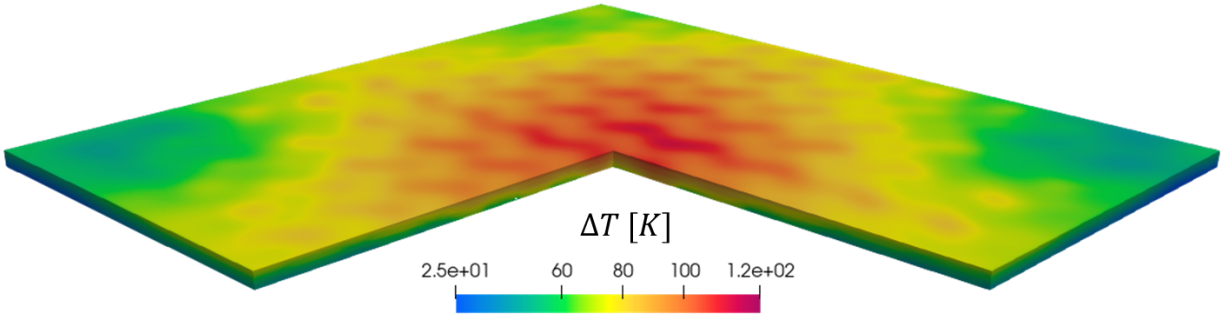
In this final section we will compare the various results that we found with different solvers, every result has a numerical error found with the process explained in section 4.5.

5.4.1 Conjugate Heat Transfer Turbulent

As we can see in the CHT cases in the comparison between Star-CCM+ and OpenFOAM the hotspots are in the same location if not slightly shifted, the temperatures and pressure in OpenFOAM are generally lower. Generally the STAR-CCM+ results are much more stable, having lower residuals and a lower numerical error. These results will be discussed in the final conclusion but we can anticipate that the mesh is the key difference here and the way these softwares handle meshing and solving is very different.



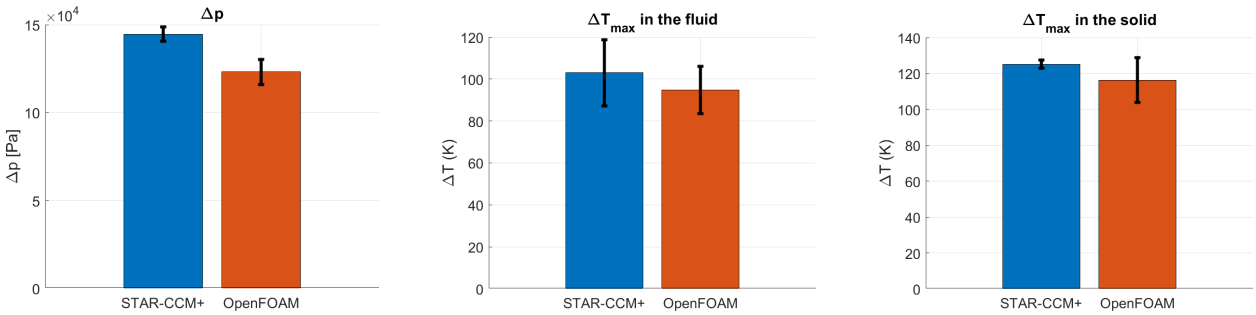
(a) STAR CCM+ Result



(b) OpenFOAM Result

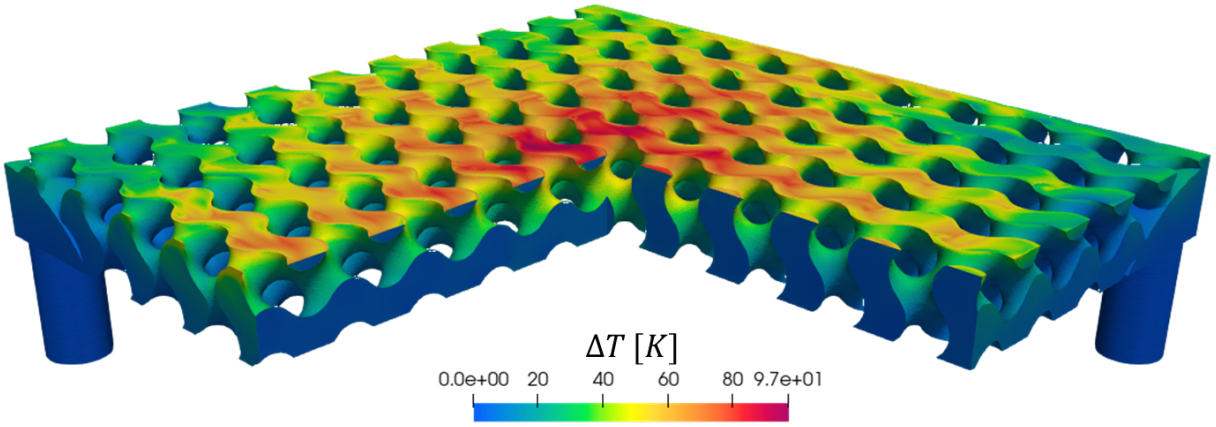
Figure 5.20: Temperature Profile in the tungsten plates

We can also observe the slightly lower temperatures in proximity of the inlets and outlets for the OpenFOAM case, as said before, the main difference here is given by the meshes, the commercial software utilizes a polyhedral mesh which generally requires more diffusion, that means a solution with more numerical dissipation and generally a more uniform heat distribution. For the results presented in the following page we can see that the numerical error is greater for OpenFOAM but that the results generally align with each other.

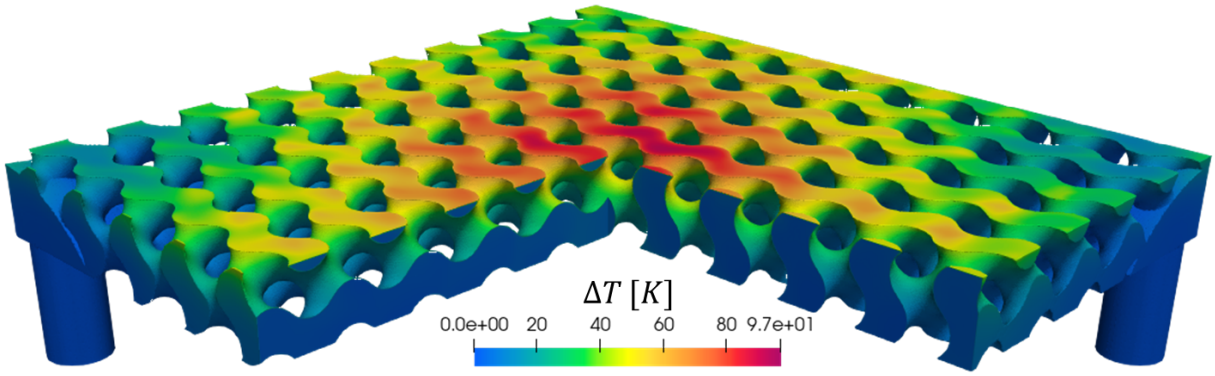


(a) Δp in the fluid (b) ΔT in the fluid region (c) ΔT in the solid region

Figure 5.21: Results for the turbulent heat transfer case



(a) STAR CCM+ Result

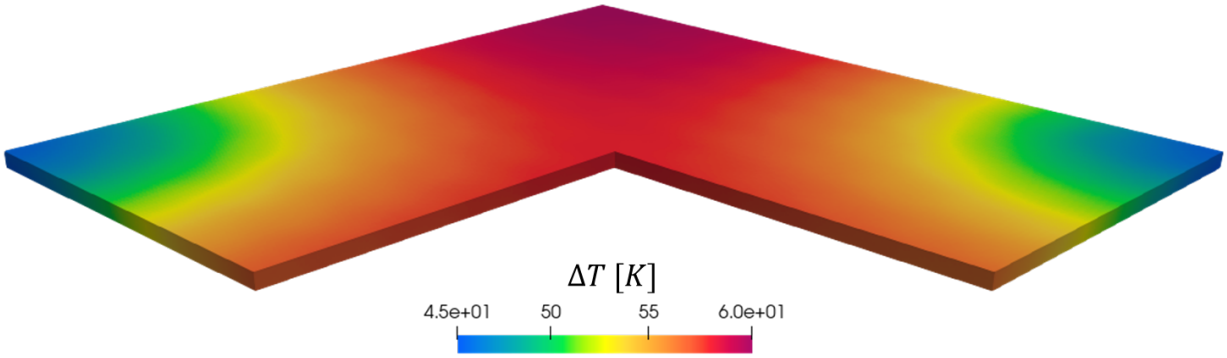


(b) OpenFOAM Result

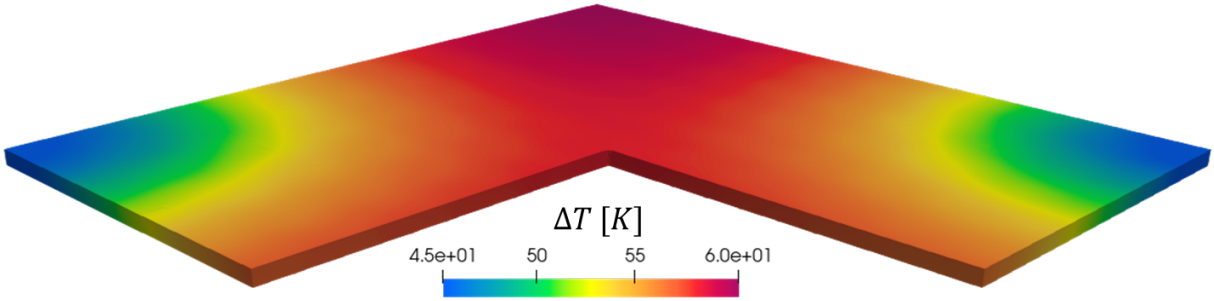
Figure 5.22: Temperature Profile in the fluid

5.4.2 Conjugate Heat Transfer Laminar

In the laminar case the results are much more similar, this is because the laminar case is less complex. We have less variables to solve for and lower velocities and gradients, the temperature is more uniform since conduction heat transfer prevails on convection. These present low residuals and numerical errors in both cases.



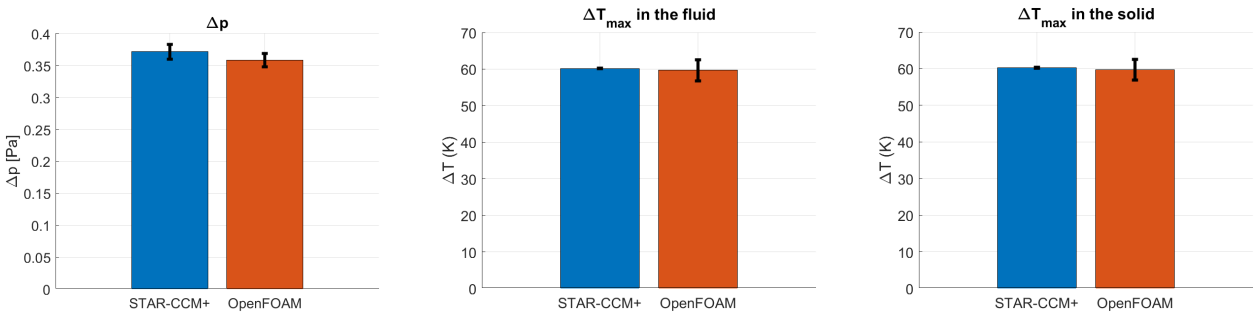
(a) STAR CCM+ Result



(b) OpenFOAM Result

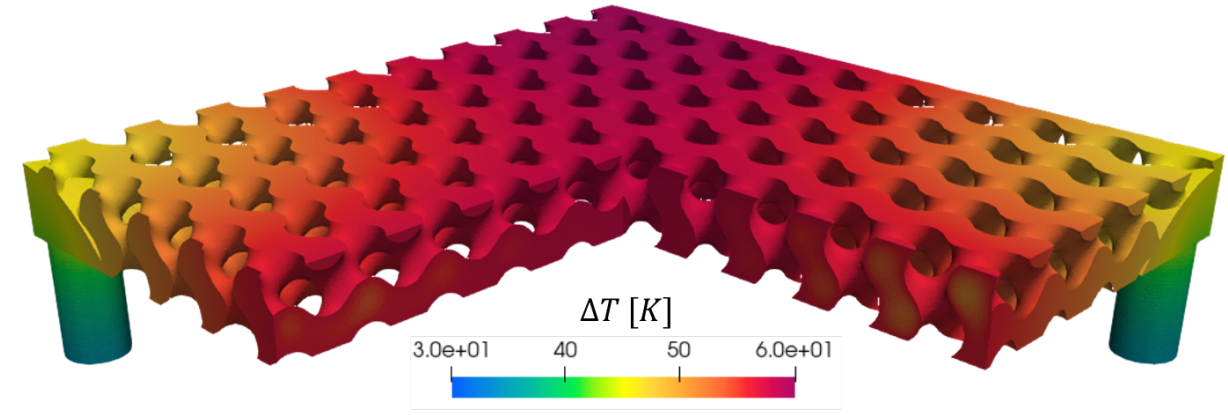
Figure 5.23: Temperature Profile in the tungsten plates

As shown the numerical results are very similar with a much lower numerical error and GCI in respect to the turbulent case. Another difference is the slower convergence, in OpenFOAM the number of iterations to reach convergence in the turbulent case is around 2000 iterations, meanwhile in the laminar we reach more than 20000 iterations. Also the temperature under relaxation factor needs to be 1 in both softwares otherwise the convergence requires a much larger number of iterations, this behavior can once again be connected to the low velocity inside the TPMS. The heat transfer is mainly conducted through conduction, so the whole fluid volume has to reach a certain temperature and doesn't manage to exit the TPMS before heating up to the solid temperature.

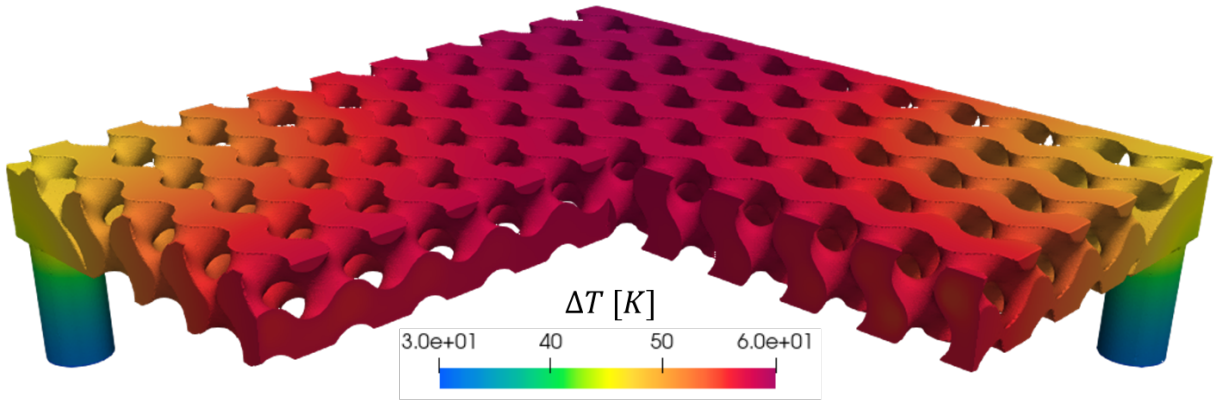


(a) Δp in the fluid (b) ΔT in the fluid region (c) ΔT in the solid region

Figure 5.24: Results for the laminar heat transfer case



(a) STAR CCM+ Result

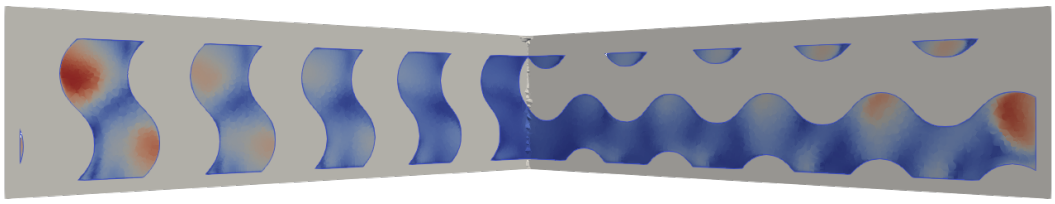


(b) OpenFOAM Result

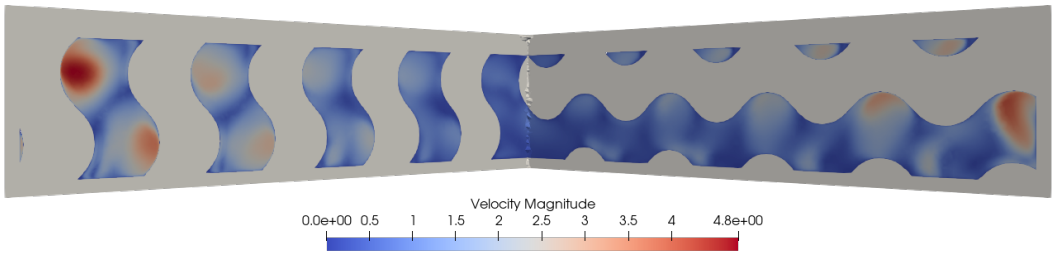
Figure 5.25: Temperature Profile in the fluid

5.4.3 Incompressible Turbulent

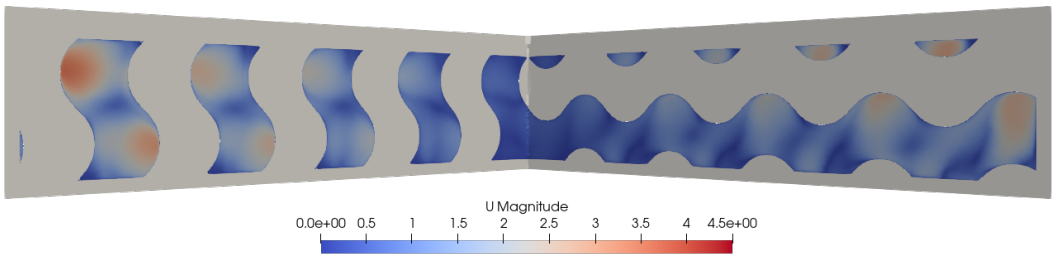
For the hydraulic turbulent case we see that while the velocity profile is similar, there are still some difference, mainly the flow swirl on the rightmost unit in figure 5.26 and lower velocities in OpenFOAM. The viscosity ratio is also lower in OpenFOAM, in accordance with these lower speeds. Meanwhile the high numerical errors in averaged values in the section could be given by the numerical nature of the average in the section, that gives quite different results for calculating the GCI.



(a) Velocity in STAR-CCM+ Section

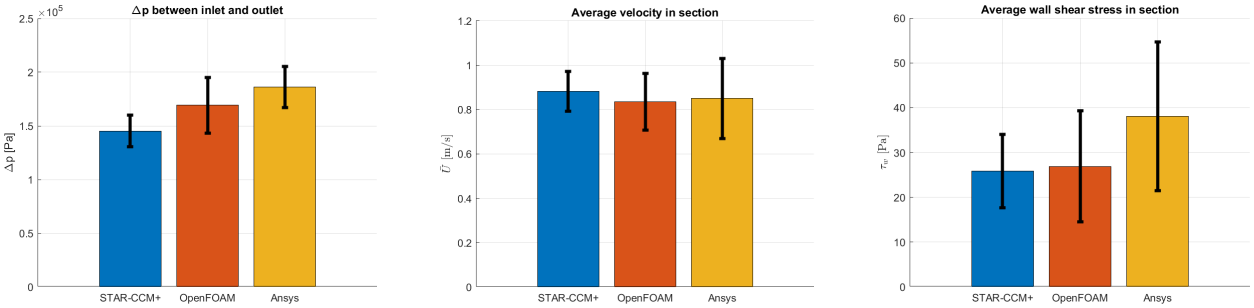


(b) Velocity in Fluent Section



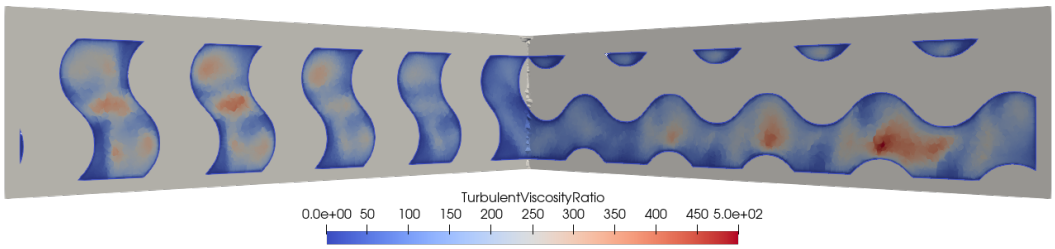
(c) Velocity in OpenFOAM Section

Figure 5.26: Turbulent case velocity comparison

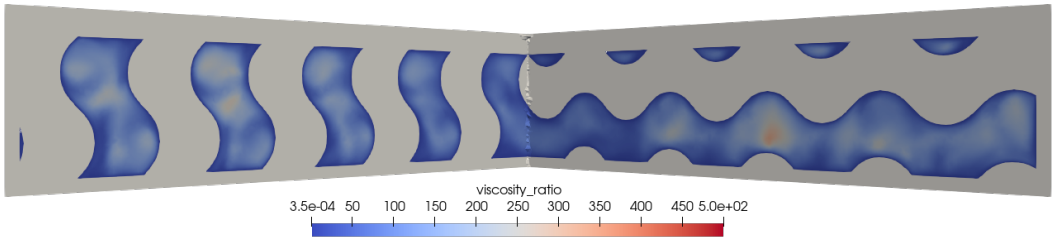


(a) Δp in the fluid (b) Average Velocity (c) Average Wall shear stress

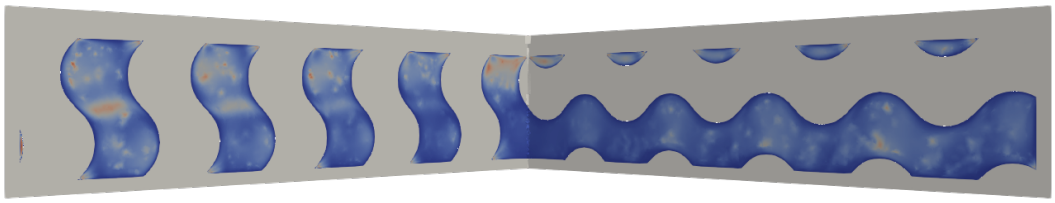
Figure 5.27: Results for the turbulent incompressible case



(a) Viscosity ratio in STAR-CCM+ Section



(b) Viscosity ratio in Fluent Section

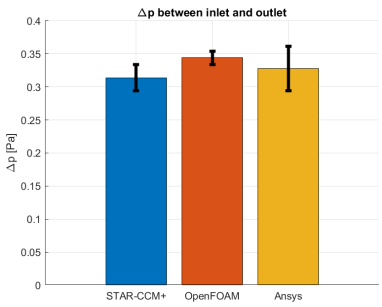


(c) Viscosity ratio in OpenFOAM Section

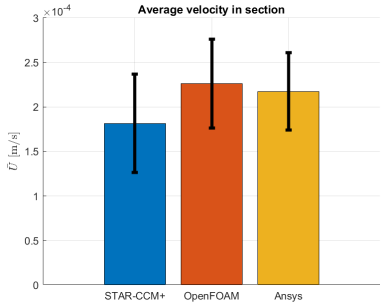
Figure 5.28: Turbulent case viscosity ratio comparison

5.4.4 Incompressible Laminar

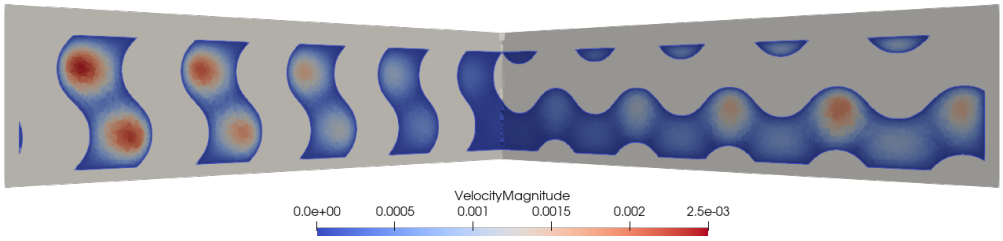
For the laminar case results are much more similar, this is another confirmation of the much better behavior of these flow in simulations, as we can see while we still have high numerical error in the section averages the flow in the section is qualitatively the same.



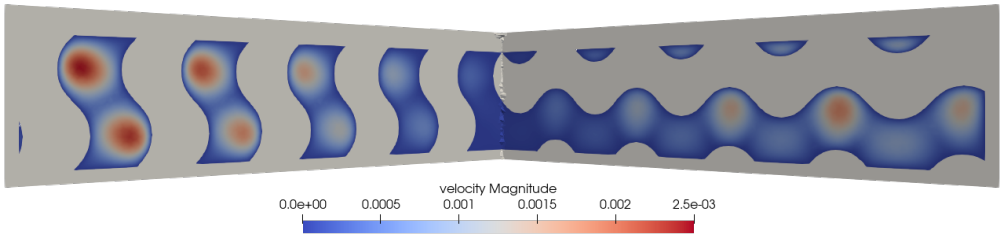
(a) Δp in the fluid



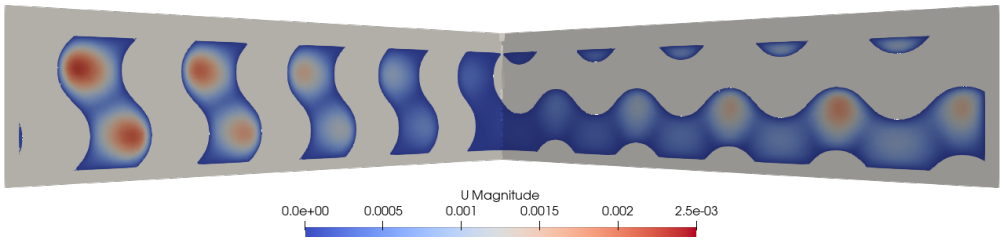
(b) Average Velocity



(a) Average Velocity in STAR-CCM+ Section



(b) Velocity in Fluent Section



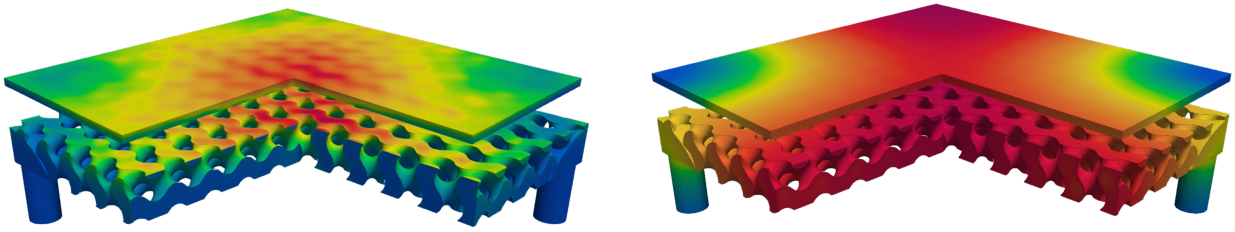
(c) Velocity in OpenFOAM Section

Figure 5.30: Laminar case velocity comparison

Conclusions

6.1 Heat transfer performance

The main difference in the heat we can see is the difference between a lower flow velocity and a higher one. In the first case conduction prevails, the fluid is evenly heated reaching the temperatures of the solid this gives slower convergence but the absence of turbulence also gives a more accurate and more stable solution. In the turbulent case the heat profile of the solid tungsten plate follows the TPMS geometry, with the spots where the copper TPMS touches the plate being 10-20 K hotter than the ones over water. This happens because convection prevails over conduction, given the high velocity of the flow the water mixes and swirls, passing through the entire TPMS in little time, without having the opportunity of heating up to solid temperatures.



(a) Turbulent heat distribution

(b) Laminar heat distribution

Figure 6.1: Turbulent and laminar heat distribution comparison

While we're still far from the 10 MW/m^2 applied by the Stellarator the concept that can be studied are still many, a taller or shorter TPMS could be implemented, sheet gyroid units could be used and different mass fluxes could be applied. With full heat load on the plasma-facing tungsten tile, the solid part can sustain higher temperatures [35] but, given that for 7.5 bar of pressure at around 440 K water changes phase to vapor, further studies with higher heat loads should consider this aspect.

6.2 Software performance

The different results given by OpenFOAM and Star-CCM+ are tied to the type of mesh and this aspect is the most important on the comparison between softwares. The hexahedral mesh is simpler to use and has a smaller degree of non-orthogonality and mesh skewness, which is absolutely necessary for OpenFOAM solvers. These solvers don't produce acceptable results around a non orthogonality degree of 85 making even first-order simulations unstable. Commercial softwares can solve for high non orthogonality and skewness with high stability, having residuals that are 1 to 2 orders of magnitude lower than OpenFOAM.

Mesher	STAR-CCM+	SnappyHexMesh
Max non-Orthogonality	85.24	64
Max skewness	16.93	3.45

Table 6.1: Mesh quality characteristics

A boundary layer generation approach has also been tried but yielded unsatisfactory results since the snappyhexmesh utility generated collapsing layers on the complex geometry and high degrees of non-orthogonality. Many different meshing approaches have been tried in OpenFOAM, with different levels of refinement or base mesh sizes, while the temperature distribution is substantially the same, temperatures change, with some meshes generating higher peak ΔT results, some reaching even 10-15 K more.

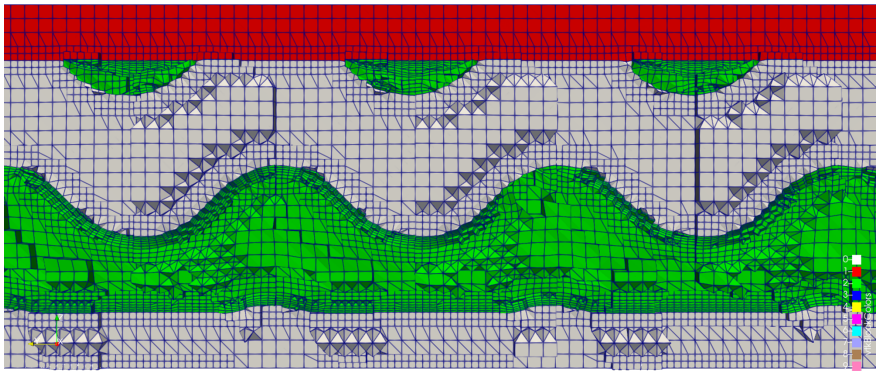
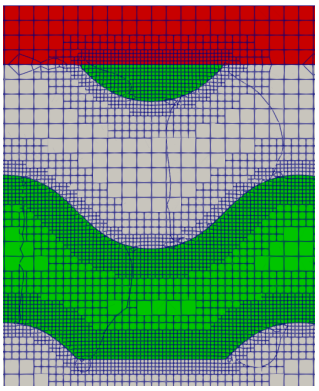
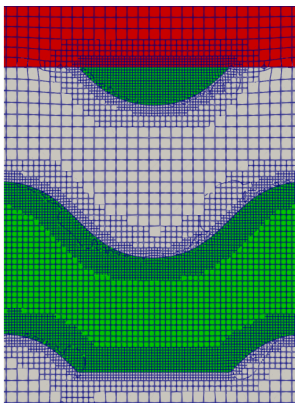


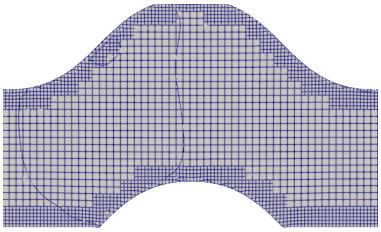
Figure 6.2: Unutilized mesh with boundary layers



(a) Unutilized mesh with 3 refinement levels



(b) Utilized mesh with 2 refinment levels



(c) Unutilized mesh with shorter refinement distance from the edge

Figure 6.3: Mesh comparison

While SnappyHexMesh is a powerful tool, it lacks consistency, and for complex geometries the tool has to be fine tuned using a trial and error process to allow the generation of a high quality mesh. The lack of boundary layers in this case forced the use of a higher refinement near the edges of the

geometries, increasing the number of cells.
This is all shown to emphasize mesh structure and its importance for OpenFOAM. Commercial softwares like STAR-CCM+ or Fluent are built for polyhedral meshes with high non-orthogonality and can be utilized to find stable solutions to these type of meshes. These type of meshes can be run in OpenFOAM but, as seen in the previous pages, they require limiters and under-relaxation to avoid solution divergence.
As we can see we have comparable results for the different softwares, but OpenFOAM generally has a higher numerical error, and STAR-CCM+, which is the software that generated the mesh is the one with the lowest. Still, the results are comparable, indicating that OpenFOAM can give appropriate results for this problem.

Bibliography

[1] Hermann Schwarz. *Gesammelte Mathematische Abhandlungen. vol. 1*. Julius Springer, 1890.

[2] "The Stellarator Concept". In: *The Physics of Fluids* 1.4 (1958). ISSN: 253-264.

[3] Alan H. Schoen. "Infinite periodic minimal surfaces without self-intersections". In: *NASA-TN-D-5541* (1970).

[4] Wolfgang Rodi Brian Launder Alan Morse and Brian Spalding. "Prediction of free shear flows — A comparison of the performance of six turbulence models". In: *NASA Conference on Free Shear Flows* (1972).

[5] S.V Patankar and D.B Spalding. "A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows". In: *International Journal of Heat and Mass Transfer* 15.10 (1972), pp. 1787–1806. ISSN: 0017-9310. DOI: https://doi.org/10.1016/0017-9310(72)90054-3. URL: https://www.sciencedirect.com/science/article/pii/0017931072900543.

[6] M.L. Hunt and C.L. Tien. "Effects of thermal dispersion on forced convection in fibrous media". In: *International Journal of Heat and Mass Transfer* 31.2 (1988), pp. 301–309. ISSN: 0017-9310. DOI: https://doi.org/10.1016/0017-9310(88)90013-0. URL: https://www.sciencedirect.com/science/article/pii/0017931088900130.

[7] *Experience with two-layer models combining the k-epsilon model with a one-equation model near the wall*. Vol. 29. Jan. 1991.

[8] L. Eça and M. Hoekstra. "Verification Procedures for Computational Fluid Dynamics On Trial". In: *IST Report D72-14* (2002).

[9] N. Delalic, Dz. Mulahasanovic, and E.N. Ganic. "Porous media compact heat exchanger unit—experiment and analysis". In: *Experimental Thermal and Fluid Science* 28.2 (2004). The International Symposium on Compact Heat Exchangers, pp. 185–192. ISSN: 0894-1777. DOI: https://doi.org/10.1016/S0894-1777(03)00038-4. URL: https://www.sciencedirect.com/science/article/pii/S0894177703000384.

[10] T. Kim et al. "Convective heat dissipation with lattice-frame materials". In: *Mechanics of Materials* 36.8 (2004). Mechanics of Cellular and Porous Materials, pp. 767–780. ISSN: 0167-6636. DOI: https://doi.org/10.1016/j.mechmat.2003.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0167663603001406.

[11] J. Wesson and D.J. Campbell. *Tokamaks*. International Series of Monographs on Physics. OUP Oxford, 2011. ISBN: 9780199592234. URL: https://books.google.it/books?id=XJssMXjHUr0C.

[12] C.Y. Zhao. "Review on thermal transport in high porosity cellular metal foams with open cells". In: *International Journal of Heat and Mass Transfer* 55.13 (2012), pp. 3618–3632. ISSN: 0017-9310. DOI: https://doi.org/10.1016/j.ijheatmasstransfer.2012.03.017. URL: https://www.sciencedirect.com/science/article/pii/S0017931012001664.

[13] Ian Gibson, David Rosen, and Brent Stucker. *Additive manufacturing technologies: 3D printing, rapid prototyping, and direct digital manufacturing*. New York, NY: Springer New York, 2015. ISBN: 9781493921126.

[14] Jun Fu et al. "Four kinds of the two-equation turbulence model's research on flow field simulation performance of DPF's porous media and swirl-type regeneration burner". In: *Applied Thermal Engineering* 93 (2016), pp. 397–404. ISSN: 1359-4311. DOI: https://doi.org/10.1016/j.applthermaleng.2015.09.116. URL: https://www.sciencedirect.com/science/article/pii/S1359431115010455.

[15] T Klinger et al. "Performance and properties of the first plasmas of Wendelstein 7-X". In: *Plasma Physics and Controlled Fusion* 59.1 (Oct. 2016), p. 014018. DOI: 10.1088/0741-3335/59/1/014018. URL: https://dx.doi.org/10.1088/0741-3335/59/1/014018.

[16] P.A. Lykov, E.V. Safonov, and A.M. Akhmedianov. "Selective Laser Melting of Copper". In: *Materials Engineering and Technologies for Production and Processing*. Vol. 843. Materials Science Forum. Trans Tech Publications Ltd, Mar. 2016, pp. 284–288. DOI: 10.4028/www.scientific.net/MSF.843.284.

[17] S. Akar, S. Rashidi, and J. Abolfazli Esfahani. "Appropriate position of porous insert in a heat exchanger by thermohydraulic analysis". In: *Heat Transfer—Asian Research* 46.8 (2017), pp. 1363–1379. DOI: https://doi.org/10.1002/htj.21279. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/htj.21279. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/htj.21279.

[18] Robert W. Corkery and Eric C. Tyrode. "On the colour of wing scales in butterflies: iridescence and preferred orientation of single gyroid photonic crystals". In: *Interface Focus* 7.4 (2017), p. 20160154. DOI: 10.1098/rsfs.2016.0154. eprint: https://royalsocietypublishing.org/doi/pdf/10.1098/rsfs.2016.0154. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsfs.2016.0154.

[19] Thomas Sunn Pedersen et al. "First results from divertor operation in Wendelstein 7-X". In: *Plasma Physics and Controlled Fusion* 61.1 (Nov. 2018), p. 014035. DOI: 10.1088/1361-6587/aaec25. URL: https://dx.doi.org/10.1088/1361-6587/aaec25.

[20] Marcin Sosnowski et al. "Polyhedral meshing in numerical analysis of conjugate heat transfer". In: *EPJ Web of Conferences*. Vol. 180. EDP Sciences. 2018, p. 02096.

[21] Lei Zhang et al. "Energy absorption characteristics of metallic triply periodic minimal surface sheet structures under compressive loading". In: *Additive Manufacturing* 23 (2018), pp. 505–515. ISSN: 2214-8604. DOI: https://doi.org/10.1016/j.addma.2018.08.007. URL: https://www.sciencedirect.com/science/article/pii/S2214860418304688.

[22] Olaf Diegel, Axel Nordin, and Damien Motte. *A practical guide to design for additive manufacturing*. Springer, 2019.

[23] Justin Baby and Murugaiyan Amirthalingam. "Microstructural development during wire arc additive manufacturing of copper-based components". In: *Welding in the World* 64.2 (2020), pp. 395–405.

[24] Jiho Kim and Dong-Jin Yoo. "3D printed compact heat exchangers with mathematically defined core structures". In: *Journal of Computational Design and Engineering* 7.4 (Apr. 2020), pp. 527–550. ISSN: 2288-5048. DOI: 10.1093/jcde/qwaa032. eprint: https://academic.oup.com/jcde/article-pdf/7/4/527/33646892/qwaa032.pdf. URL: https://doi.org/10.1093/jcde/qwaa032.

[25] J. Boscary et al. "Completion of the production of the W7-X divertor target modules". In: *Fusion Engineering and Design* 166 (2021), p. 112293. ISSN: 0920-3796. DOI: https://doi.org/10.1016/j.fusengdes.2021.112293. URL: https://www.sciencedirect.com/science/article/pii/S0920379621000697.

[26] "Fusion Energy". In: *IAEA BULLETIN* 62.2 (2021).

[27] Suraj Dinkar Jadhav et al. "Laser-based powder bed fusion additive manufacturing of pure copper". In: *Additive Manufacturing* 42 (2021), p. 101990. ISSN: 2214-8604. DOI: https://doi.org/10.1016/j.addma.2021.101990. URL: https://www.sciencedirect.com/science/article/pii/S221486042100155X.

[28] Nada Baobaid et al. "Fluid flow and heat transfer of porous TPMS architected heat sinks in free convection environment". In: *Case Studies in Thermal Engineering* 33 (2022), p. 101944. ISSN: 2214-157X. DOI: https://doi.org/10.1016/j.csite.2022.101944. URL: https://www.sciencedirect.com/science/article/pii/S2214157X22001903.

[29] Krzysztof Dutkowski, Marcin Kruzel, and Krzysztof Rokosz. "Review of the State-of-the-Art Uses of Minimal Surfaces in Heat Transfer". In: *Energies* 15.21 (2022). ISSN: 1996-1073. DOI: 10.3390/en15217994. URL: https://www.mdpi.com/1996-1073/15/21/7994.

[30] Christopher Greenshields. *OpenFOAM v10 User Guide*. London, UK: The OpenFOAM Foundation, 2022. URL: https://doc.cfd.direct/openfoam/user-guide-v10.

[31] Christopher Greenshields and Henry Weller. *Notes on Computational Fluid Dynamics: General Principles*. Reading, UK: CFD Direct Ltd, 2022.

[32] Mauricio Ivan Tenorio-Suárez et al. "MaSMaker: An open-source, portable software to create and integrate maze-like surfaces into arbitrary geometries". In: *SoftwareX* 19 (2022), p. 101203. ISSN: 2352-7110. DOI: https://doi.org/10.1016/j.softx.2022.101203. URL: https://www.sciencedirect.com/science/article/pii/S2352711022001212.

[33] J.H. You et al. "Divertor of the European DEMO: Engineering and technologies for power exhaust". In: *Fusion Engineering and Design* 175 (2022), p. 113010. ISSN: 0920-3796. DOI: https://doi.org/10.1016/j.fusengdes.2022.113010. URL: https://www.sciencedirect.com/science/article/pii/S0920379622000102.

[34] Fares Alawwa et al. "Thermohydraulic performance comparison of 3D printed circuit heatsinks with conventional integral fin heatsinks". In: *Applied Thermal Engineering* 226 (2023), p. 120356. ISSN: 1359-4311. DOI: https://doi.org/10.1016/j.applthermaleng.2023.120356. URL: https://www.sciencedirect.com/science/article/pii/S135943112300385X.

[35] J. Boscary et al. "Conceptual design of the next generation of W7-X divertor W-target elements". In: *Fusion Engineering and Design* 192 (2023), p. 113629. ISSN: 0920-3796. DOI: https://doi.org/10.1016/j.fusengdes.2023.113629. URL: https://www.sciencedirect.com/science/article/pii/S0920379623002132.

[36] Yaxin Cao et al. "Design and mechanical evaluation of additively-manufactured graded TPMS lattices with biodegradable polymer composites". In: *Journal of Materials Research and Technology* 23 (2023), pp. 2868–2880. ISSN: 2238-7854. DOI: https://doi.org/10.1016/j.jmrt.2023.01.221. URL: https://www.sciencedirect.com/science/article/pii/S2238785423002235.

[37] Rafael Santiago et al. "Modelling and optimisation of TPMS-based lattices subjected to high strain-rate impact loadings". In: *International Journal of Impact Engineering* 177 (2023), p. 104592. ISSN: 0734-743X. DOI: https://doi.org/10.1016/j.ijimpeng.2023.104592. URL: https://www.sciencedirect.com/science/article/pii/S0734743X23001033.

[38] Ansys. *Ansys Fluent Documentation*. URL: https://www.afs.enea.it/project/neptunius/docs/fluent/index.htm.

[39] Max Planck Gesellschaft. *Brennpunkte der Kernfusion*. URL: https://www.mpg.de/19734973/brennpunkte-der-kernfusion.

[40] NIST. *Thermophysical properties of fluid systems*. URL: https://webbook.nist.gov/chemistry/fluid/.

[41] OpenCFD. *OpenFOAM User Guide*. URL: https://www.openfoam.com/documentation/guides/latest/doc/index.html.