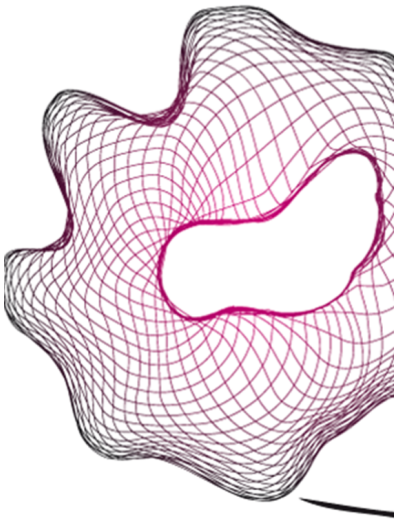
UNIVERSITY OF TWENTE.

Faculty of Engineering Technology



THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

Validation of Eilmer4 by a 3D boundary layer problem

J.G. Spee
Internship Centre for Hypersonics
17 September 2018 - 20 December 2019

Supervisors:
Dr P.A. Jacobs
Dr. R.J. Gollan
Prof. dr. ir. H.W.M. Hoeijmakers

Internship (30 EC)
Master Mechanical Engineering

Egineering Fluid Dynamics Group
Faculty of Engineering Technology
University of Twente
P.O. Box 217
7500 AE Enschede
The Netherlands

ACKNOWLEDGEMENTS

I would like to thank Prof. Dr. Ir. H.W.M. Hoeijmakers for making this amazing experience possible. I always had the wish to go abroad and especially to go to Brisbane Australia. Without him I probably would not have been there. Secondly I would like to thank Dr. P.A. Jacobs. He invested a lot of time to show me around and to supervise me. He was a great help as a colleague as well as a friend giving me tips to explore the beautiful country. Lastly I would like to thank Dr. R. J. Gollan. He helped me a lot to get started with the project, but also with all the Linux related problems.

ABSTRACT

The program Eilmer4 is in constant development. A lot of simulations in two dimensions have been performed with success. Simulations in three dimensions however are a relatively unexplored area. To help further develop the program a comparison study has been performed. The study of Sandy Tirtey *Characterization of a Transitional Hypersonic Boundary-Layer in Wind Tunnel and Flight conditions* [10] has been compared with the results of a simulation done by Eilmer4. The setup of this system is made in a Lua script. In order to minimize errors first a robust system to describe the geometry and boundary conditions had to be generated. To get insights for the $3D$ simulation a $2D$ simulation has been done previously. The final simulation has been compared with the experiments of Tirtey. The flow pattern produced by Eilmer4 matches the results of the experiments. The calculated heat flux however is 2 orders higher in the results of Eilmer4. The calculated modified Stanton number corresponds better although the results still do not correspond completely. This study has improved the program Eilmer4, but to better compare the results more simulations with an even finer grid are suggested.

Keywords: Eilmer, Hypersonic flow, Boundary-layer transition

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

An import part of the Master Mechanical Engineering at the University of Twente is conducting an internship. The student gets the opportunity to apply his obtained knowledge at a company or university and enrich his knowledge in a different way than taught at the university. The student is completely free (till a certain level) to choose where to stay. With big interest I followed the Computational Fluid Dynamics (CFD) course at the university. I also always had the wish to go abroad for a time to experience other cultures. To do an internship at the Centre for Hypersonics at the University of Queensland (UQ) was a great chance I had to grab.

The University of Queensland was founded in 1909 and belongs to the world's top universities according to several independent rankings [13][12]. Its main campus is situated in St. Lucia. A beautiful location in one of the meanders of the Brisbane river.

The Centre for Hypersonics group (CfH), led by professor Richard Morgan, is part of the School of Mechanical & Mining Engineering. The group has many areas of expertise, but the most important ones are shock wave generation, SCRAMjet propulsion and rocket-launched flight testing [3]. It is widely recognized as one of the leading research groups in the field of hypersonics which is done both experimentally as by computational fluid dynamics. Their X-labs (as they call their laboratory) contains multiple expansion tubes whereof the X3 expansion tube is the biggest. With these they are capable of creating hypersonic flow conditions for a short amount of time. Figure 1.1 shows an example of a test performed in their X3 expansion tube.

Dr. Peter Jacobs and Dr. Rowan Gollan play a key role in the CFD research of the group. They developed the open source Eilmer program. It began as a tool to help with the design of shock tunnels and expansion tubes, but has become a full CFD program for compressible flows [1]. The program is now in its fourth version called Eilmer4 and is still updated daily with all sorts of expansions and fixes to continuously improve the program.

The constant development of Eilmer4 asks for continuous validation of the program. The program gets minor updates and bug fixes daily to add new features or improve the computational time or storage space. Eilmer4 is already used for many two dimensional simulations, but simulations in three dimensions are still a fairly unexplored area. This can be seen from

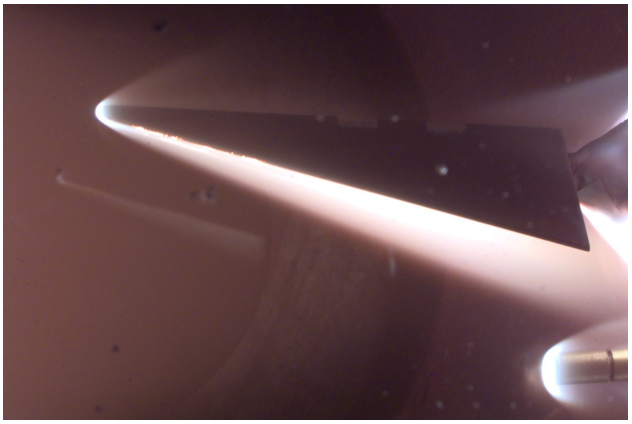


Figure 1.1: Example of a test performed in the X3 expansion tube (Blunt waverider (Ne-H2 at $9.25km/s$)) [2]

the number of examples which are available as you download the program.

Sandy Tirtey researched the effects of a small roughness element on a plate subjected to hypersonic speed[10]. These are simple geometries with complex flow phenomena. He investigated this with experiments as well as with a CFD simulation. These experiments are perfect to reproduce with Eilmer4. It can give good insights in modeling more complex 3D geometries in Eilmer4, but also verification of the results produced by the program helps further develop it. Therefore my task is to create the model of Tirteys experiments in Eilmer4 and compare the results in order to further improve the program Eilmer4.

Chapter 2

PROBLEM SPECIFICATION

Earth re-entry has been and will be a big topic for science. Countless variables make it hard for researchers to correctly simulate the re-entry of space vehicles. Because of the uncertainties huge safety margins are taken into account when designing such vehicles. These vehicles enter the earth atmosphere with hypersonic speeds. Hypersonic speeds are characterized as speeds with a Mach number higher than 5 [11]. Sandy Tirtey did a lot of research into this topic. In his paper *Characterization of a Transitional Hypersonic Boundary Layer in Wind Tunnel and Flight conditions* [10] he conducted several experiments about the effects of a small roughness element on a plate which is subjected to hypersonic speeds. This element creates a boundary layer and Tirtey wanted to improve the understanding of the flow topology and the physical process behind this phenomenon. This contributes to designing better thermal protections systems because similar phenomena appear when for example the heat shield has been damaged.

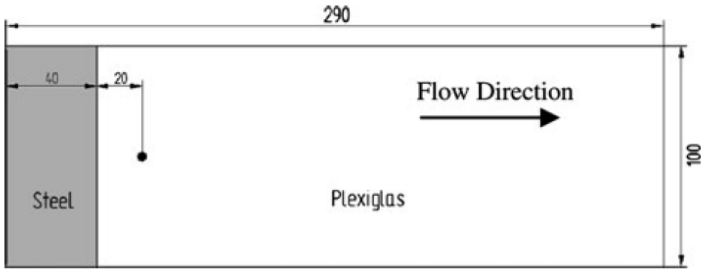


Figure 2.1: Top-view of the experimental setup used by Tirtey [10] (Dimensions are in millimeters)

He performed real experiments as well as CFD simulations using the finite-volume based LORE code. The experimental setup is shown in figure 2.1. Tirtey used different elements for his study which can be seen in figure 2.2. These elements are placed $60mm$ from the leading edge of a plate. The plate has a leading edge radius of $0.5mm$. Tirtey performed several experiments with different total pressures, but with a constant total temperature of $500K$ leading to low, medium and high Reynolds conditions. The wall temperature started at $T_w = 292K$ with an error margin of $2K$ and rose during the tests $4K$ for laminar flow and up to $12K$ for turbulent flow. For its CFD simulations he used an isothermal wall conditions with a constant temperature of $T_w = 300K$.

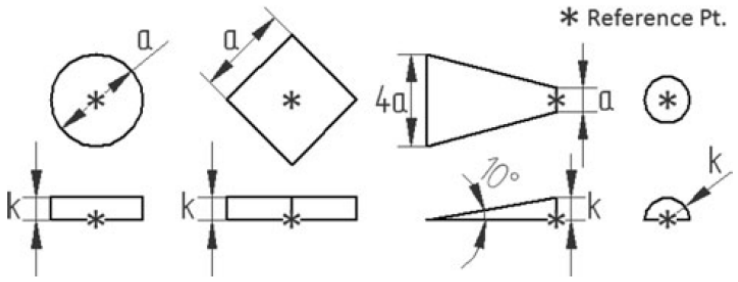


Figure 2.2: Elements used by Tirtey [10]

2.1 Eilmer4

Eilmer4 is a program for the numerical simulation of transient, compressible gas flows. This is done by using the finite volume method. The program can be used for 2D and 3D problems. The user has to define the spatial domain, the initial gas state and boundary conditions where after the program does the calculations according to the laws of gas dynamics. The input script uses the Lua language. This is a powerful and fast programming language which is easy understandable [8]. Eilmer itself is mainly written in the D language. Its predecessor was written in the C++ language, but the aim of the developers P.A. Jacobs and R.J Gollan was to build a program which was easily understandable by advanced gas dynamics students or an undergraduate student of engineering. Most of the time these students do not possess highly developed programming skills and therefore with Eilmer4 they made the switch to the powerful D language with the Lua scripting language [7]. The program is in constant development. Jacobs and Gollan (but also others) provide nearly daily updates for the program with bug fixes, performance improvements or user comfort optimization's. The $2D$ simulations of Eilmer4 are already quite good and fast. Three dimensional simulations are however a rather unexplored area.

2.2 Objective

The objective of this study is to help further improve the program Eilmer4 by carrying out one of the simulations done by Tirtey. The results will be compared to see if the program gives the correct results, but also the experience of the user to set up a big, geometrically more challenging, three dimensional simulation will be evaluated.

The experiments of Tirtey [10] are a good test to see how the program performs with a big $3D$ simulation. The interest does not only lie in the results of the simulation, but also ease of use and computational time. To do so only one of Tirteys experiments will be simulated. The rectangle element is chosen since this is the easiest element to model. Also only a high Reynolds number simulation has been performed. The initial conditions of the simulation

are listed in table 2.1. One has to note that the Reynolds number of $7.5 * 10^6$ is calculated with the whole length of the plate ($290mm$). As can be read in chapter 3 the length of the plate is only $115mm$ where after the domain is cut off for numerous reasons. The Reynolds number is kept the same.

P_0	$3.1 * 10^6 Pa$
T_0	$500K$
T_w	$300K$
M	6
Re	$7.5 * 10^6$

Table 2.1: Initial conditions

2.3 Robust modeling

Grids in Eilmer4 can be self made or imported from an external program such as GridPro [4]. For this study the choice has been made to use a self made grid. The reason is that in this way not only the $3D$ simulations of Eilmer4 can be improved, but also tips can be given to create a better user experience for this kind of simulations.

Eilmer4 is a very versatile program and has countless possibilities. This has the advantage that the user can optimize it to his needs. The downside is that the user needs to define every single element or option because otherwise the program will give errors. To create a mesh for the simple cube shown in figure 2.3 one first needs to define all eight points where after the volume can be defined. This can then be meshed. The program will mesh the volume structured (i.e. evenly spaced). For most problems the user wants to refine the mesh at critical points which can be done with Robertsfunctions

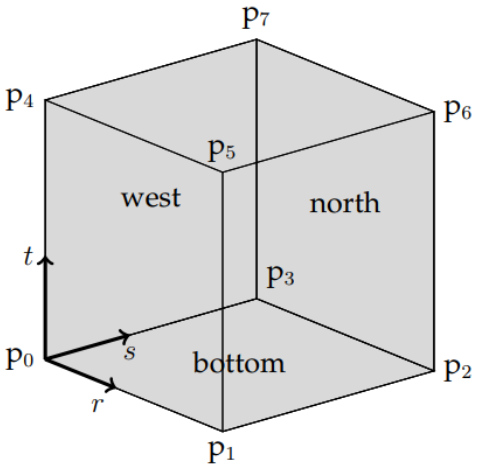


Figure 2.3: Simple cube

Robertsfunctions need to be defined for a line segment. One can specify to which end (or both) refinement is needed and the rate of refinement. A volume has 12 line segments, so 12 Robertsfunctions can be defined. For two volumes next to each other, this already will be 24 and since the two volumes share one face, one must make sure that the 4 shared line segments have the same Robertsfunctions, otherwise misaligned cells will give errors. For a complex, or even a simple geometry, with multiple volumes one can imagine that precautions must be taken in order to prevent hours of debugging. Therefore a robust system has to be developed such that it is clear how all entities affect each other.

Chapter 3

CREATING THE GRID

3.1 Topology

With the problem defined in chapter 2 first the domain of the simulation had to be determined. By doing a $2D$ simulation one gets an idea about the domain and the corresponding computational time. Therefore the domain above the plate has a length of $150mm$, a width of $20mm$ and a height of $20mm$. It is obvious to use an symmetry axis. The xz-plane is chosen as the axis of symmetry. This makes the effective width $40mm$. The original plate was $290mm$ in length and $100mm$ in width, but since our main interest is the behavior of the flow around and shortly after the element, this domain would suffice and shortens the computational time. Figure 3.1 shows to topology of the problem for the xy-plane and figure 3.2 shows the topology for the yz-plane. The origin is in point $bA1$. This is on top of the flat plate on the symmetry plane. To minimalize the chance of errors a clear notation system is used. Every axis has a different notation.

1. Points along the x-direction have small letters (a, b c, etc)
2. Points along the y-direction have capital letters (A, B C, etc)
3. Points along the z-direction have numbers (0, 1, 2, etc)

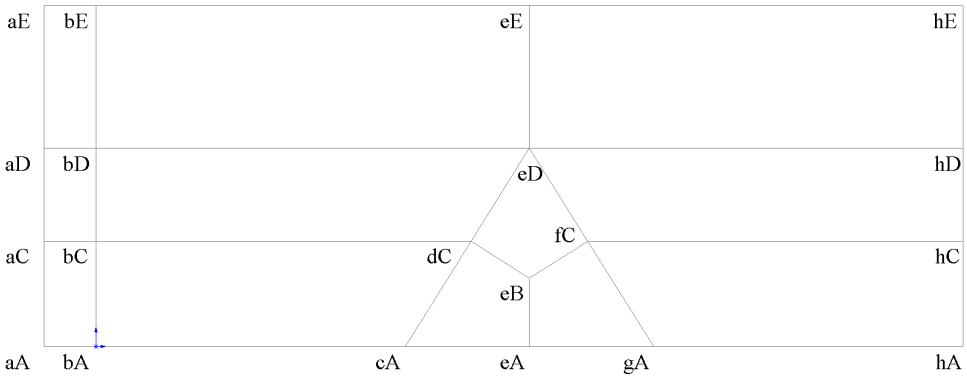


Figure 3.1: Topology of the problem in the xy-plane.

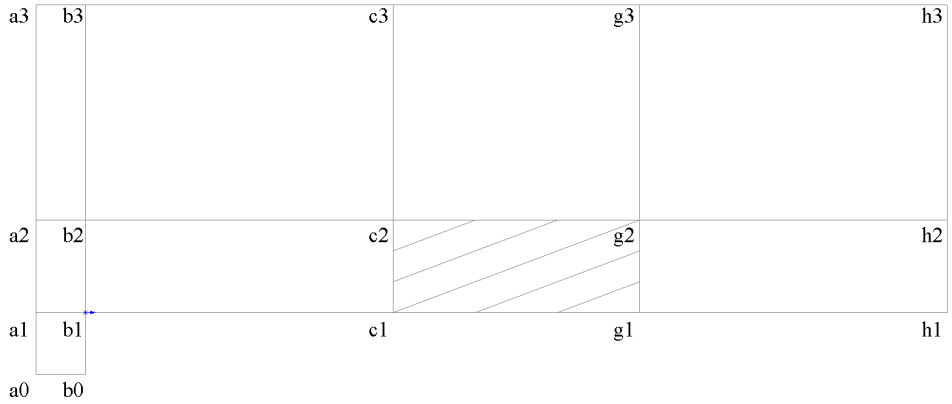


Figure 3.2: Topology of the problem in the yz-plane.

As a point normally first is denoted with its x-value followed by its y- and z-value, this is no exception, so a point is indicated with a small letter followed by a capital letter and a number. For example: $cA1$, $eE3$. In positive direction the letters or numbers are ascending. The numbering of the points is reasonably straightforward for the yz-plane. The xy-plane is a little bit more difficult around the element. There is only one point on the B line: eB. During the experiments one can decide to move this point to create different shaped elements in order to improve accuracy. One has to make sure that the y-position of point eB has a smaller value than the points on the C line otherwise the system is not correct anymore. This should not be a problem since it gives very strange shaped elements if point eB moves across the C line.

3.2 Line segments definition

For line segments a similar construction is used. Line segments are denoted with $xxYY00$ where the outer points on an axis are used. If it is on a axis line then only one of the indices has been used. For example line $bdC1$ is between points $bC1$ and $dC1$ and line $cdAC3$ is between points $cA3$ and $dC3$.

3.3 Volume segments definition

The same way of defining line segments applies for volumes. Listing 3.1 shows an example of how a volume is defined. Always the outer two points of volume on a particular axis are stated in the name of the volume. This is because points $dC2$ and $fC2$ form the outer two points on the x-axis, points $eB2$ and $eD2$ form two outer two points on the y-axis and 23 because those are the boundaries on the z-axis.

```
244 vol_dfBD23 = TFIVolume:new{vertices={dC2, eB2, fC2, eD2, dC3, eB3, fC3,
        eD3}}
```

Listing 3.1: Volume definition

3.4 Grids

A grid is defined as shown in listing 3.2. To define a grid a volume, a cf list and numbers of cells are needed. The definition of grids itself is the same as with volumes.

```
306 grid_abAC01 = StructuredGrid:new{pvolume=vol_abAC01,
307                                  cfList=cf_abAE02,
308                                  niv=nab+1, njv=nAC+1, nkv=n01+1}
```

Listing 3.2: Grid definition

Number of cells

The number of cells use the same definition as the line segment definition: "n" followed by the outer points of the line segment which is divided into cells.

Robertsfunction definition

As said in the beginning of this chapter, Robertsfunctions are used to refine a mesh around critical points. An example of a Robertsfunction is:

```
1 RobertsFunction:new{end0=true, end1=false, beta=1.1}
```

Listing 3.3: Robertsfunction definition

The user needs to specify which end needs refinement (or both) and what the rate of refinement is (β). The closer the β gets to 1.0 the more clustered the line segment will be. end0 will always be closer to zero in the coordinate system then end1. So if one applies the Robertsfunction in listing 3.3 on the line between p_4 and p_5 in figure 2.3, cells will get smaller towards point p_4. Of course every line can have its own Robertsfunction which will probably give the best result, but it will also be a tedious work to specify every Robertsfunction and make sure that it is applied correctly on neighbouring grids. Therefore it is easier to use a Robertsfunction for multiple lines, but again a system is needed to minimize mistakes. This is easiest explained by an example. Listing 3.4 shows line 302 in the code. This Robertsfunction is applied over the whole z-axis and it is used to refine around the edge of the plate. The name starts with "rcf" followed by ab which is the line segment the Robertsfunction applies to. $AE03$ is the domain in which all line segments ab will be given this Robertsfunction.

```
302 rcfabAE03 = RobertsFunction:new{end0=false, end1=true, beta=1.05}
```

Listing 3.4: Robertsfunction example

Cf list

The cf list combines all Robertsfunctions for that particular volume. Again for every volume a different cf list can be made, but this is unnecessary. For example: there is no interesting difference between volume vol_abAC01 which is at the leading edge of the plate and neighbouring volume vol_abCD01. Therefore the same Robertsfunctions can be applied to the same edges of the volume and the same cf list can be used. The same naming system for the Robertsfunctions is used for the cf lists. Listing 3.5 shows an example. These cflist can be applied on the domain $abAE02$ which is the whole domain in front of the leading edge except for the upper volumes (see figure 2.3 on which edges of the cube the Robertsfunctions are applied).

```
304 cf_abAE02 = {edge01=rcfabAE03, edge32=rcfabAE03, edge45=rcfabAE03,
    edge76=rcfabAE03}
```

Listing 3.5: cf list example

Element

The main difficulty of this problem is the element placed on the plate. The symmetry axis splits the rectangular element in a triangular element. This triangular element has to be described with rectangular cells and preferably not strange shaped cells. Figure 3.3 shows a close up of the topology of the element. The faces as defined in Eilmer and shown in figure 2.3 have been added. Normally the south face of one cell will match the north face of the neighbouring cell. The same applies for west and east. Because the triangle has to be divided into rectangles, there is a misalignment in this concept and one can see in figure 3.3 that on segment $efBC$ an east north couple arose and on segment $efCD$ a north west couple developed. The first does not give problems, but the latter on the other hand does.

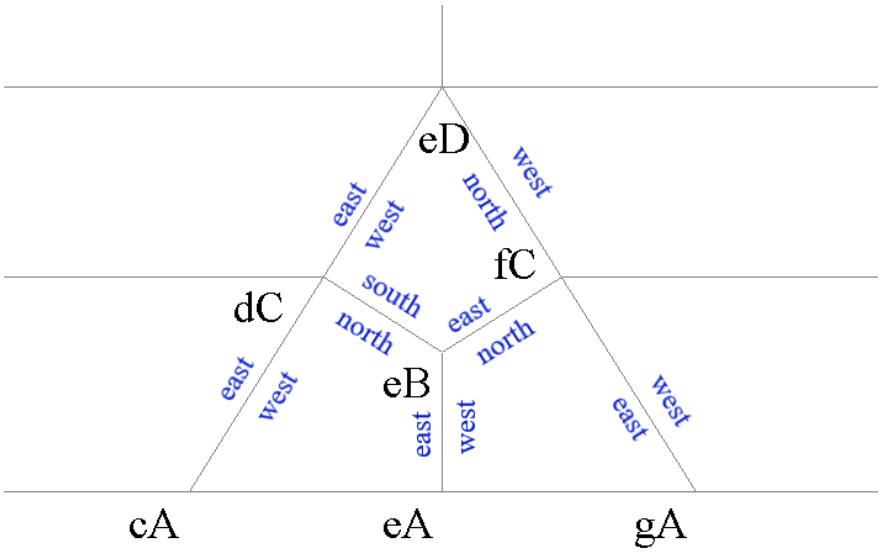


Figure 3.3: Close up of topology around the element

The problem lies with the definition of the Robertsfunction. The corresponding Robertsfunction is shown on listing 3.6. $end0 = false$ and $end1 = true$ so the Robertsfunction will cluster towards $end1$. $end0$ will always be closest to the origin. So if it is applied on a west line segment in figure 2.3, p_0 will be $end0$ and p_3 will be $end1$. Applied on a north line segment in the same figure, p_3 will be $end0$ and p_2 will be $end1$.

```
346 rcfCD12 = RobertsFunction:new{end0=false, end1=true, beta=beta_CD12}
```

Listing 3.6: Robertsfunction of line $efCD$

The Robertsfunction in listing 3.6 applied on the north face of quadrilateral $efBD$ will result in $end0$ being eD and $end1$ being fC. The same Robertsfunction applied to the west face of quadrilateral $ehCD$ will result in $end0$ being fC and $end1$ being eD. This will give clustering around different points and cell misalignment will occur. To solve this a mirror Robertsfunction has been made with a new variable for the β so if one wants to make adjustments only this new variable has to be changed.

Flow blocks

The final step to define the model is making flow blocks from the grids. The same denomination as for the grids and volumes is used. An example of a flow block can be found in line 453 (see listing 3.7). In the flow blocks are the boundary and flow conditions specified. The above example is at the leading edge of the plate. Therefore an inflow initial state is used and the west face also has a inflow boundary condition. The east face is the leading edge

and therefore a wall boundary condition is applied. This is with a no slip condition. The program can identify blocks next to each other, so not all faces have to be specified. If a face is not next to another block and it is not specified automatically a wall boundary condition with slip is used. The inflow initial state is best used on all blocks. However, at the trailing edge of the element ($eD2$ until $gA2$ this gives a problem. The program thinks there will be inflow from the west boundary, but since the wall of the element is there, there can not be inflow and there also is no information of the blocks above the element at the first step. This gives errors and therefore a noflow state is used which is the same as the inflow state except that the velocity is set to zero.

```
453 blk_abAC01 = FluidBlockArray{grid=grid_abAC01, initialState=inflow, nib
    =1, njb=1,
454                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow},
455                                             east=WallBC_NoSlip_FixedT:new{
                                                Twall=T_w}}} --b0000
```

Listing 3.7: Flow block example

Chapter 4

RESULTS AND DISCUSSION

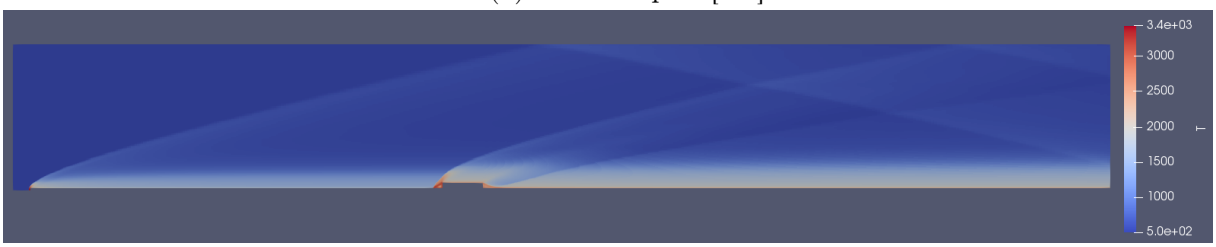
4.1 The model

The system described in chapter 3 worked very good. After writing the whole code (around 500 lines by the time of the first simulation) only three minor errors were present. Due to the consistent system these errors could easily be found and fixed. Some other errors were found after a first successful simulation. These were related to the problem described in section 3.4. After these were repaired everything worked smoothly. A lot of grid refinements have taken place between the first and the final test. The number of cells on a line segment could easily be adjusted without giving errors. Also changing the point of refinement (with use of the Robertsfunctions) was simple and again without errors.
The system developed worked perfectly, but a remark has to be made that this may differ per person. Everybody has his own likes and dislikes and therefore this system can work for one person, but another person may dislike it.

4.2 2D analysis

Simulating is a process of trial and error and it is almost impossible to make a very detailed $3D$ model without any errors. First a $2D$ simulation has performed in order to see what critical points are and how big domain one has to choose.

(a) Pressure plot $[Pa]$



(b) Temperature plot $[K]$

Figure 4.1: Results of $2D$ simulation with 105600 cells

The results of the last $2D$ simulation are shown in figure 4.1. A fairly fine grid has been used with 105600 cells. The pressure plot (see figure 4.1a) clearly shows the two boundary layers. The pressure at the leading edge of the plate is roughly 20 times as high as the surrounding pressure of $6.1MPa$ while the pressure at the leading edge of the element is only 10 times as high. Due to the wall with slip boundary condition at the topside of the domain the boundary layer reflected. A close up of the pressure in the leading edge is shown in figure 4.2. Although a really fine mesh is used, one can still see that around this critical point the mesh is fairly coarse and information is lost due to the averaging of neighbouring cells. This "stairs pattern" is created because the program takes the average of all neighbouring cells and assumes that the outcome is valid for the entire cell. Unfortunately there is not much that can be done except for refining.

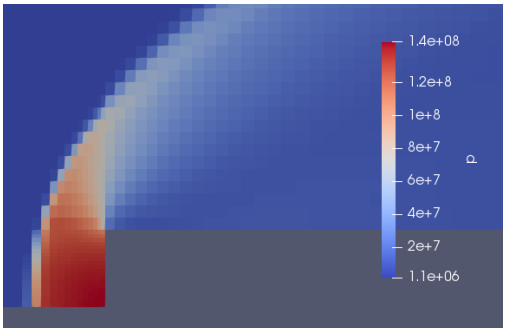


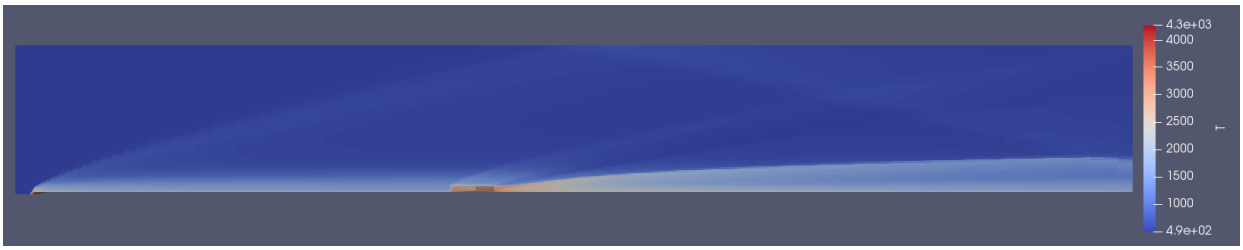
Figure 4.2: Close up of the pressure $[Pa]$ at the leading edge of the plate

The temperature plot (figure 4.1b) also shows the two boundary layers, but it also shows a third boundary layer at the trailing edge of the element. This corresponds to the low pressure zone behind the element.

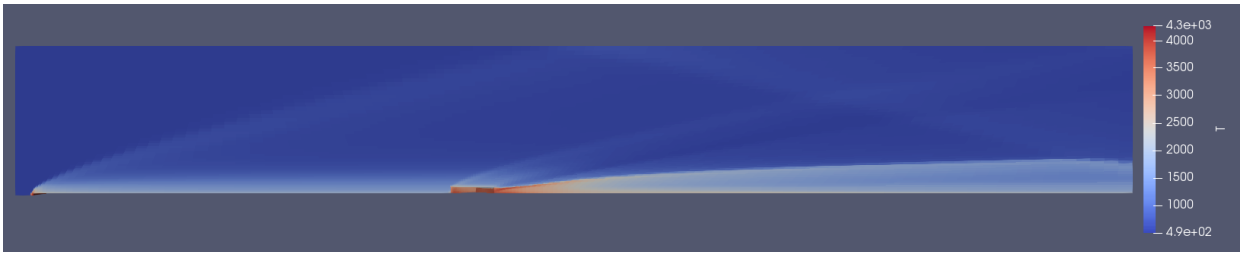
4.3 3D analysis

Perfect gas and Ideal gas assumption

As told in section 4.2 the simulation process is mainly trail and error and the researcher always has to make decisions between getting the most accurate results and computational time. The simulations always have been performed with the perfect gas assumption because Tirtey used the same in his experiments. At the fourth simulation with only around 700.000 cells the computational time was enormous. Therefore a second simulation with the ideal gas assumption has been done. Figure 4.3 shows the results of the temperature distribution of the perfect gas (figure 4.3a) and ideal gas (figure 4.3b). One can see that there is almost no difference except for the temperature behind the element.



(a) Temperature plot of a simulation using perfect gas



(b) Temperature plot of a simulation using ideal gas

Figure 4.3: Comparison of perfect gas and ideal gas assumptions

A close up of this area can be found in figure 4.4. Here the left half of the image is with the ideal gas assumption and the right half is with the perfect gas assumption. The difference of $700K$ is substantial, but the drastically reduced computational time was more important and therefore further simulation have been performed with an ideal gas assumption.

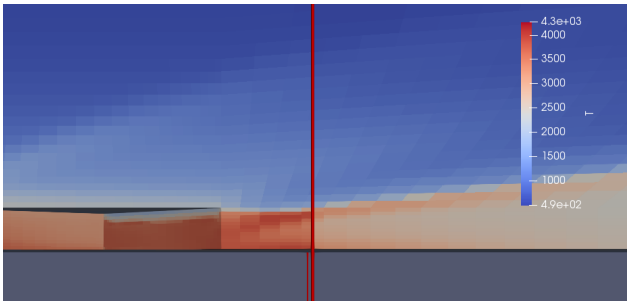


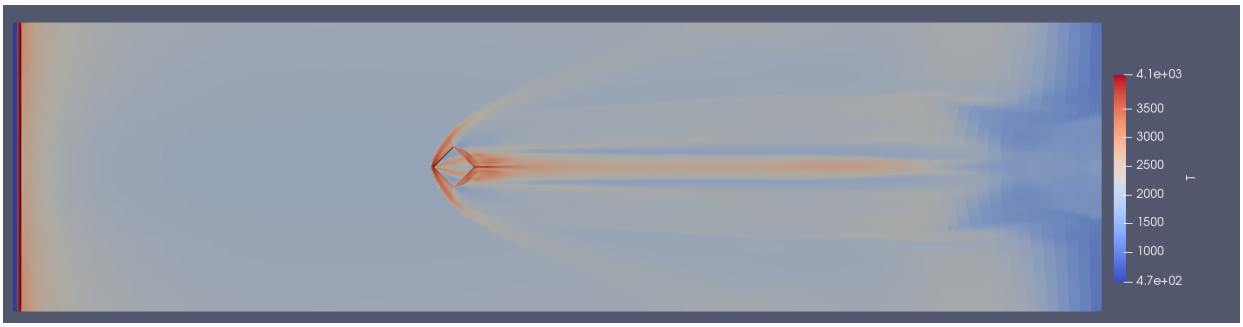
Figure 4.4: Detail of temperature difference after element with left ideal gas and right perfect gas state

4.4 Final simulation

The final simulation is done with a grid of 1424500 cells and took 39 hours and 41 minutes on the Goliath EAIT Faculty Cluster HPC (High Performance Computer). Figures 4.5a and 4.5b show the pressure and temperature distribution with a bottom view on the problem. Because of the high pressure at the leading edge of the plate, the difference in pressure around the element is hard to tell. Figure 4.6 shows the temperature distribution along the symmetry axis. This image looks similar to figure 4.1b. A small difference is noticeable

before and after the element due to the difference in $2D$ and $3D$ since in $3D$ the flow can go over and around the element whereas in $2D$ the flow only can go over the element. Figure 4.7 shows a $3D$ picture made out of slices of the simulation to give an idea how the temperature distributes in three dimensions across the plane. Clearly visible is the high temperature in the boundary layer developed around the element. Going further from the element, the temperature rapidly decreases. High temperatures can also be seen just behind the north and south edges of the element. The flow goes over as well as around the element. The flows going around the element (north and south in the x-y plane) meet at the eastern edge of the element creating very high temperatures. Vortices are created at the southern and northern edge and these create the characteristic profile. This can also be seen in the CFD simulations done by Tirtey [10].

(a) Bottom view of the pressure distribution



(b) Bottom view of the temperature distribution

Figure 4.5: Results of the final simulation with 1424500 cells

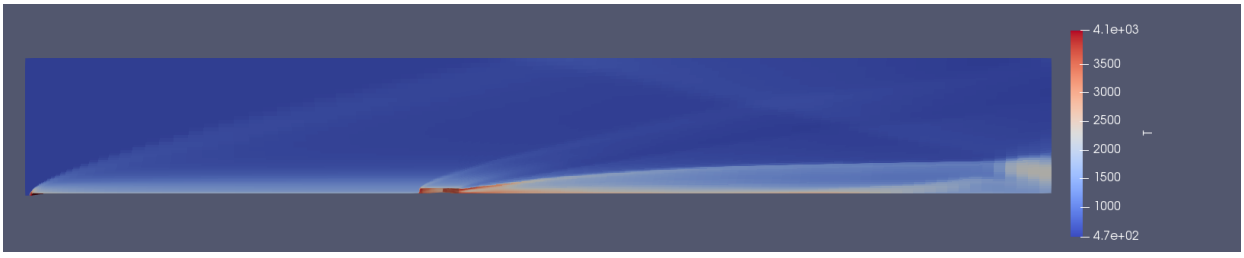


Figure 4.6: Temperature distribution of the y-z plane in the middle of the x-plane

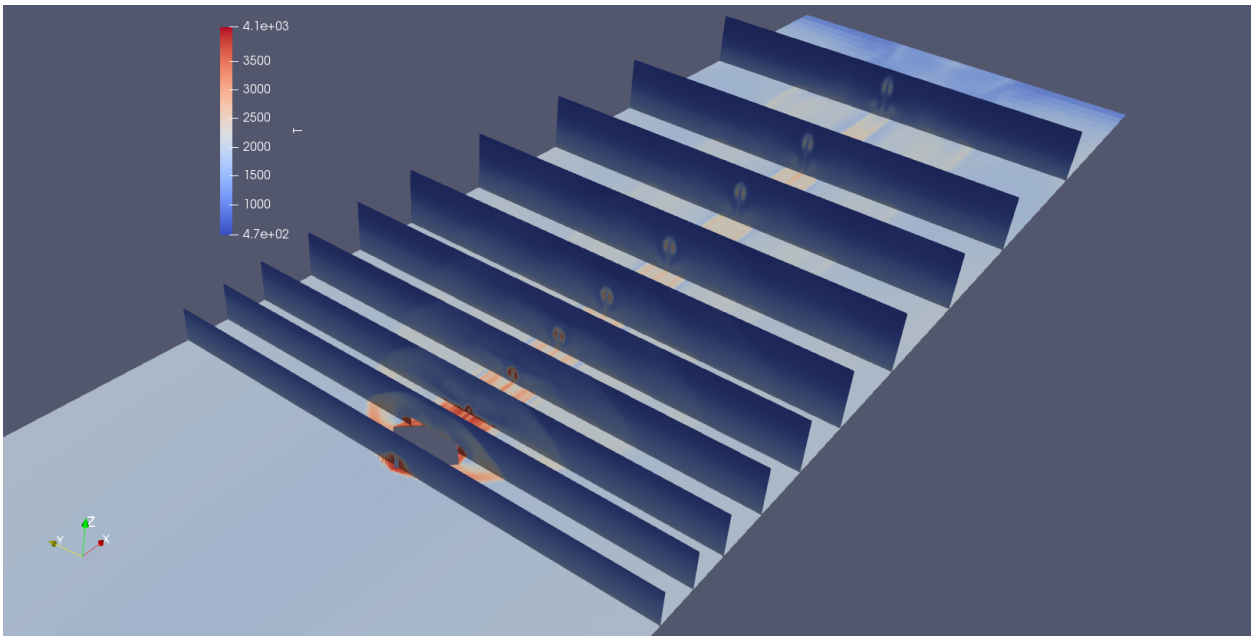
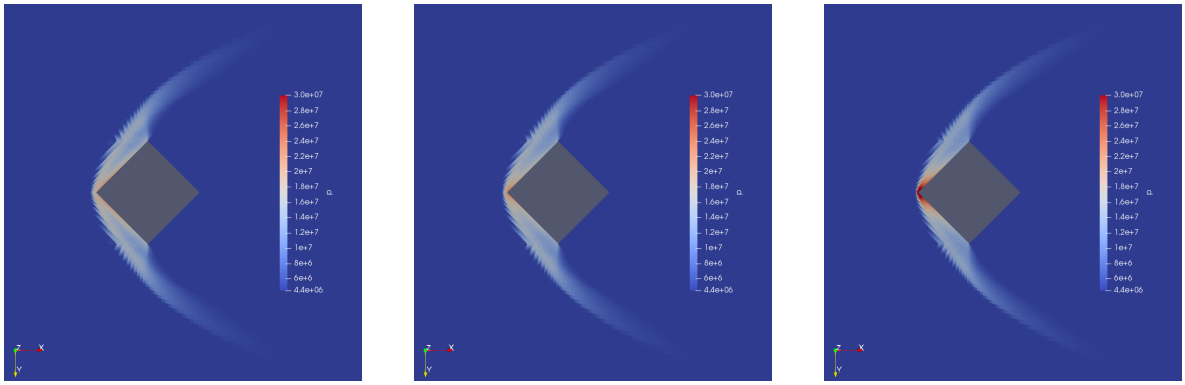
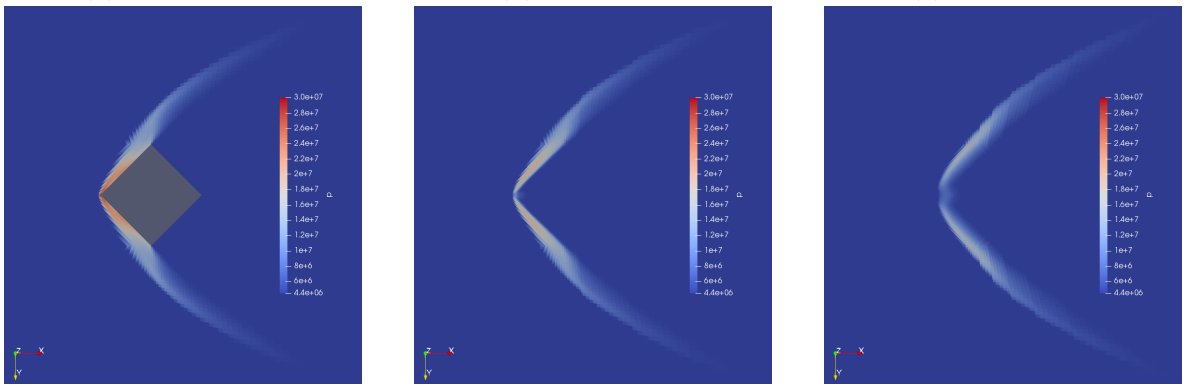


Figure 4.7: $3D$ representation of the temperature distribution



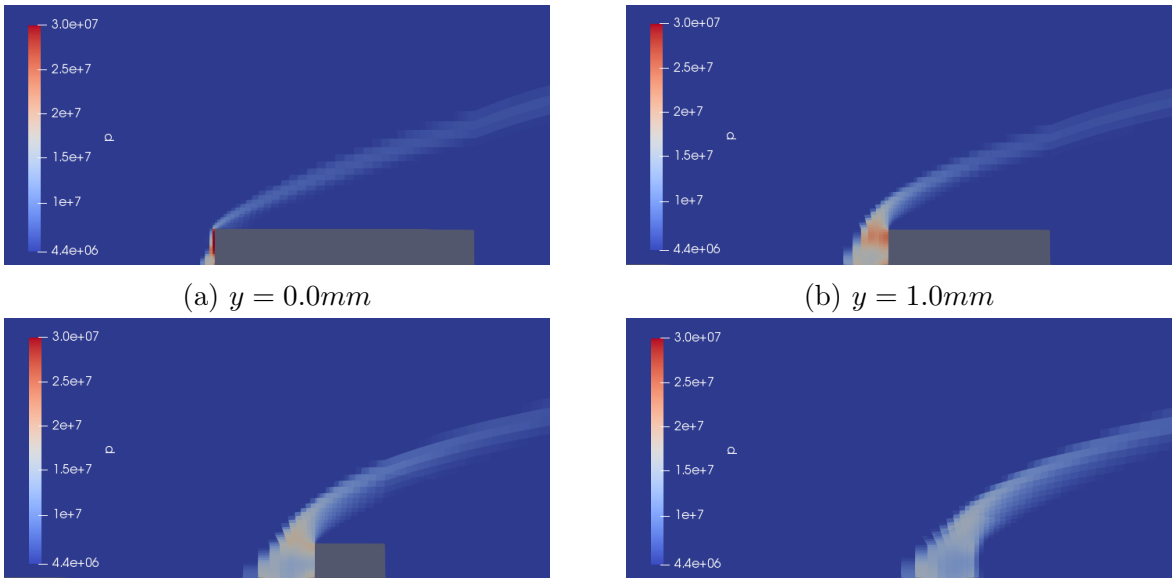
(a) $z = 0.01mm$ (b) $z = 0.2mm$ (c) $z = 0.4mm$



(d) $z = 0.8mm$ (e) $z = 1.0mm$ (f) $z = 1.5mm$

Figure 4.8: Pressure development around the element for different heights

Figure 4.8 shows the pressure distribution around the element for different heights where figure 4.8a is just above the plate ($z = 0.01mm$) and figure 4.8f is $0.7mm$ above the element ($z = 1.5mm$). One can see the really thin but very high pressure zone at the most left corner of the element (figure 4.8c) Especially close to the surface of the plate this edge forces the flow to go around the element (clockwise or counterclockwise) whereas further apart from the plate the flow also goes over the element. The shape of the boundary layer close to the surface also has a more blunt form and it gets sharper with increasing height. Remarkable is the lower pressure zone between the outside of the boundary layer and near the edge of the element (figures 4.8a, 4.8b and 4.8c) and the low pressure zone above the element at the left edge (figures 4.8e and 4.8f).



(a) $y = 0.0mm$ (b) $y = 1.0mm$

(c) $y = 2.0mm$ (d) $y = 3.0mm$

Figure 4.9: Pressure development around the element for different distances from the symmetry axis

Loads

Eilmer4 also has an option to calculate loads, such as heat flux or shear stress, on surfaces. Figure 4.10 shows the heat flux on the surface due to conduction of the plate and the upper surface of the element. The loads are calculated for the center of a cell and therefore these are plotted as markers with a color corresponding to the heat flux. Figure 4.11 shows the heat flux around the element in more detail.

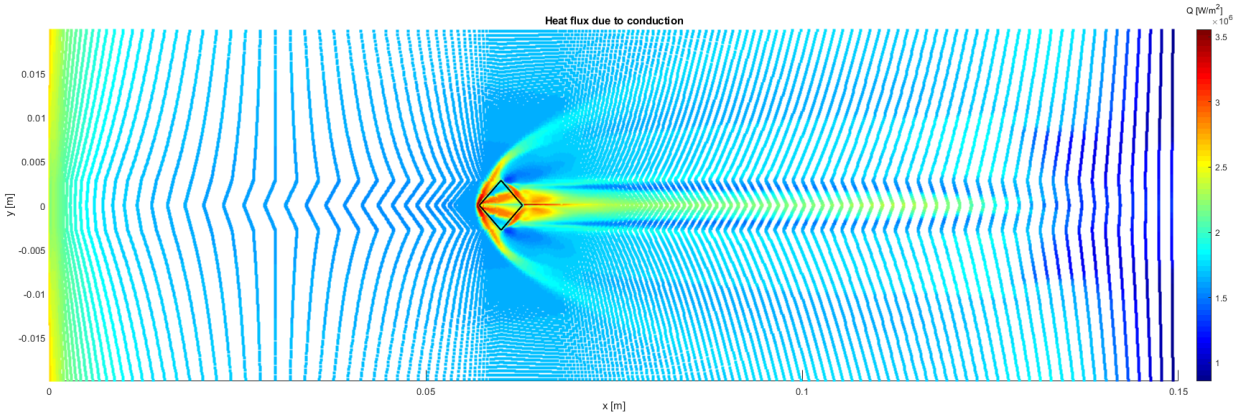


Figure 4.10: Heat flux on the surface of the plate and element

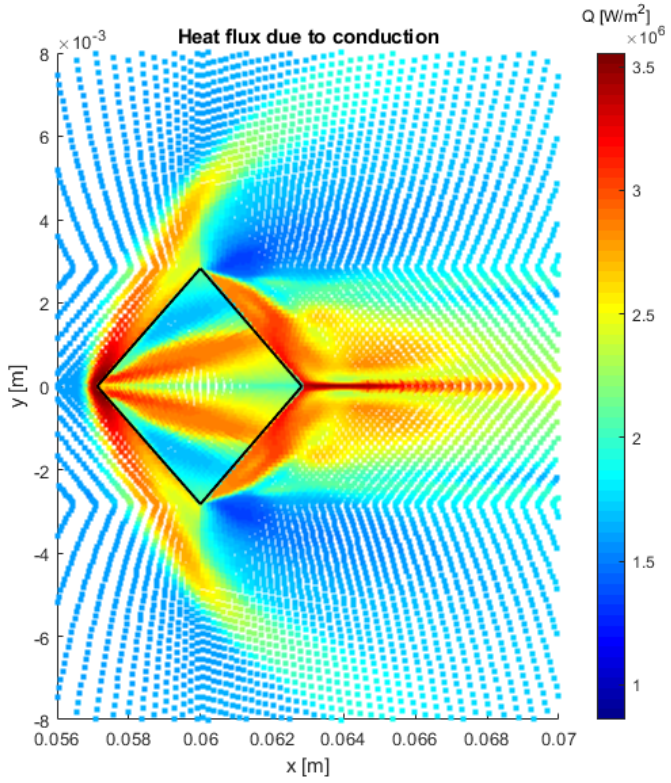


Figure 4.11: Detail of heat flux around the element

The heat flux is of the order $1E+06\ W/m^2$. Although Tirtey did not plot the heat flux for this element, he plotted it for a cylindrical element and the ramp. The heat fluxes calculated by his CFD simulation were of the order $1E+04\ W/m^2$.

A task group of NATO also did CFD simulations with Tirteys experiments as base [9]. The highest order heat flux in their simulations ($Q_w \sim 1E+06\ W/m^2$) corresponds to the heat

flux calculated by Eilmer4. This high order heat flux however is not seen often and the average heat flux is of order $1E+04\ W/m^2$. Both simulations show a order difference of $1E+02\ W/m^2$.

Tirtey also calculated the Modified Stanton number. Equation 4.1 shows how this dimensionless number is calculated. Q_w is the wall heat-flux and T_0 the stagnation temperature. The symbols C_p, ρ_{inf}, U_{inf} and T_w represent respectively the isobaric specific heat, the density of the free stream, the velocity of the free stream and the wall temperature.
The temperature values can be found in table 2.1. The other values are calculated by Eilmer4 and can be found in table 4.1. With these values the modified Stanton number can be calculated for every cell. The result is shown in figure 4.12. The order of the modified Stanton number calculated by Eilmer4 and the simulations done by Tirtey correspond. Figure 4.13 shows how the modified Stanton number varies along the x-axis ($y = 0$). In comparison with Tirtey there is still a difference of $\sim 3E+04$.

C_p	$1.1\ KJ/(kgK)$
ρ_{inf}	$21.595\ kg/m^3$
U_{inf}	$2945\ m/s$

Table 4.1: Values calculated by Eilmer4

$$C_{h,mod} = \frac{Q_w}{C_p \rho_{\text{inf}} U_{\text{inf}} (T_0 - T_w)} \tag{4.1}$$

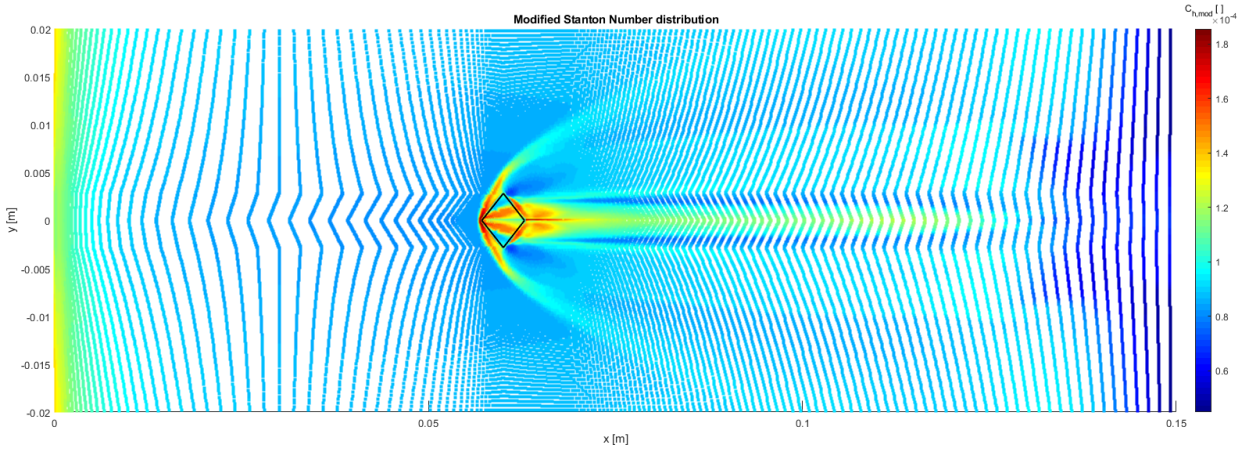


Figure 4.12: Modified Stanton number distribution

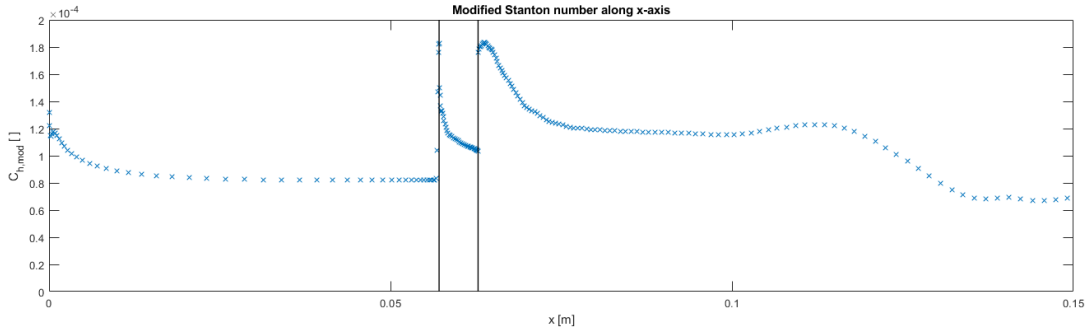


Figure 4.13: Modified Stanton number distribution along the x-axis. The black lines represent the area where the element is

Chapter 5

CONCLUSION

The experiments described in *Characterization of a Transitional Hypersonic Boundary Layer in Wind Tunnel and Flight conditions* [10] have been simulated in Eilmer4. In order to do so successfully a robust modeling system was invented. The model was easily adjustable and if bugs were present they could be solved without much effort. The system is not perfect for everybody, since this is dependant on personal preference. It can be used as an example to systematically build an own modeling system for a $3D$ problem in Eilmer4 however.
The results of the simulation look similar to the experiments Tirtey did. The flow looks the same as well as the boundary layer. Unfortunately there is only one picture of the square element experiment in Tirtey's paper [10]. In this picture the boundary layer is not clearly visible. The simulations performed by NATO [9] show a bow shock wave which is further away from the element. The values of the heat transfer do not correspond with both Tirtey's as NATO's experiments. While the Eilmer4 simulations give values of $Q_w \sim 1E+06\ W/m^2$, the other experiments give values of $Q_w \sim 1E+04\ W/m^2$. This difference is quite significant. Another comparison is made by comparing the modified Stanton numbers. The experiments now do correspond better, although there is still a difference of $\sim\ 3E+04$.
With this study Eilmer4 has been improved. A few bugs in the program were found during this study. These were fixed and also the program has been updated to decrease computational time. It is still a tedious work to model a $3D$ problem in Eilmer4. The method developed here can be used as an example on how to approach such a problem. If geometries get significantly more complicated one has to consider using other grid generating software.

Chapter 6

RECOMMENDATIONS

This study has improved the program Eilmer4, but since it is still in development, it always can be improved more. The results of this simulation seem to correspond with the experiments of Tirtey. This must be further investigated however. The final simulation already used almost 1.5 million cells, but the grid is still fairly coarse. The refinement has been focused around the element. This is obvious as here the flow changes the most. The domain at Tirtey's experiments is bigger and it shows that also interesting things happen after 150 mm. His simulations also used 21-32 million cells. Computational times would have skyrocketed if this kind of grids were used. To truly compare Eilmer4 with Tirtey's experiments an extra simulation needs to be performed with a 2 or even 4 times bigger mesh.

BIBLIOGRAPHY

[1] *About Eilmer*. 2018. URL: http://cfcfd.mechmining.uq.edu.au/eilmer/about/.

[2] *Blunt Waverider in X3 (Ne-H2 at* $9.25km/s$*)*. 2019. URL: http://hypersonics.mechmining.uq.edu.au/blunt-waverider-2.

[3] *Centre for Hypersonics profile*. 2018. URL: http://hypersonics.mechmining.uq.edu.au/.

[4] *GridPro*. 2019. URL: https://www.gridpro.com/.

[5] P. Jacobs and R. Gollan. "Guide to the geometry package". Sept. 2018. URL: http://cfcfd.mechmining.uq.edu.au/eilmer/documentation/.

[6] P. Jacobs and R. Gollan. "Guide to the transient flow solver". July 2018. URL: http://cfcfd.mechmining.uq.edu.au/eilmer/documentation/.

[7] P. Jacobs and R. Gollan. "Implementation of a Compressible-Flow Simulation Code in the D Programming Language". In: *Applied Mechanics and Materials* 846 (2016), pp. 54–60.

[8] *Lua language*. 2018. URL: https://www.lua.org/start.html.

[9] Multiple. "Extended Assessment of Stability and Control Prediction Methods for NATO Air Vehicles". In: *Technical Report RDP* (Nov. 2016). DOI: 10.14339/STO-TR-AVT-201.

[10] L. Walpot S.C. Tirtey O. Chazot. "Characterization of hypersonic roughness-induced boundary-layer transition". In: *Experiments in fluids* 50.2 (2011), pp. 407–418. DOI: 10.1007/s00348-010-0939-4.

[11] *Speed Regimes - Low Hypersonic*. 2019. URL: https://www.grc.nasa.gov/www/k-12/BGP/lowhyper.html.

[12] *Times higher education*. 2018. URL: https://www.timeshighereducation.com/student/best-universities/best-universities-australia.

[13] *University profile*. 2018. URL: http://www.uq.edu.au/about/university-profile.

Appendix A

CODE

```
1  -- Tirtey_hypersonic_boundary-layer_transition.lua
2  -- Simple 3D simulation of Hypersonic flow over a roughness element
3  -- Han Spee
4  -- This is an example on how to setup a structured way to describe a 3D
       flow problem.
5
6  -- General setup
7  config.title = "Simple 3D simulation of Hypersonic flow over a
       roughness element"
8  print(config.title)
9  config.dimensions = 3
10 axisymmetric = true
11
12 -- Configures loads so heat transfer can be calculated
13 config.compute_loads = true
14 config.dt_loads = 5.0e-6
15 config.boundary_group_for_loads = "loads"
16
17 -- THE GAS MODEL
18 -- This section loads the gas model. The gas model is setup in a
       different lua script where this section invokes the correct database
       .
19 nsp, nmodes = setGasModel('ideal-air-gas-model.lua')
20 print("GasModel set to perfect air. nsp= ", nsp, " nmodes= ", nmodes)
21
22 -- Setup begin conditions
23 p_inf = 3.1e6 -- Pressure of the flow in Pa
24 T_inf = 500 -- Temperature of the flow in degrees K
25 M_inf = 6 -- Mach number[]
26 Re = 7.5e6 -- Reynolds number[]
27 T_w = 300 -- Wall temperature in degree K
28
29 -- Verify simulation conditions
```

```
30 inflowtest = FlowState:new{p=p_inf, velx=0, T=T_inf}
31 U_inf = M_inf*inflowtest.a
32 mu = inflowtest.mu
33 rho = inflowtest.rho
34 inflow = FlowState:new{p=p_inf, velx=U_inf, T=T_inf}
35 noflow = FlowState:new{p=p_inf/10, velx=0.0, T=T_inf}
36 print("p_inf=", p_inf, "T=", T_inf, "M_inf=", M_inf, "Re=", Re)
37 print("U_inf=", U_inf, "mu=", mu, "rho=", rho)
38
39 -- mm conversion
40 mm = 1.0e-3 -- metres per mm
41
42 ----------------------------------------------------------------------
43 ----------------------------------------------------------------------
44 -- GENERAL GEOMETRY DOMAIN
45 -- In this section all points are defined which setup de basis of the
       whole geometry.
46 -- To do so, first a set of lengths is defined whereafter all points
       are made.
47 -- A specific strategy is used to define all coordinates in x, y and z-
       direction.
48 -- All points in the x-direction are marked with a small letter (a, b,
       c etc)
49 -- All points in the y-direction are marked with a capital letter (A, B
       , C etc)
50 -- All points in the z-direction are marked with a number (0, 1, 2 etc)
51 -- All markings are ascending in positive direction
52 -- Origin at symmetry axis at the beginning of the plate
53 ori = Vector3:new{x=0.0, y=0.0, z=0.0}
54
55 ----------------------------------------------------------------------
56 -- DEFINING ALL LENGTHS
57 -- Geometry of the element
58 a = 4.0*mm -- side length
59 k = 0.8*mm -- height
60 L_e = 0.5*(a^2+a^2)^0.5 -- half of the intersection
61 L_h = 0.5*L_e -- quarter of the intersection
62
```

```
63 -- x-direction
64 pos_FP = 60*mm -- length from center of the element to leading edge
65 pos_B = 1.0*mm -- length in front of the plate so boundary layer can
       start to develop
66 pos_AP = 90*mm -- length after center of the element
67
68 -- y-direction
69 W_e = L_e -- width of the element in y-direction
70 W_dom = 20*mm -- width of domain
71 W_h = 0.5*W_e -- half width of element
72 W_ce = 0.8*mm -- dividing point in element
73
74 -- z-direction
75 H_p = 0.25*mm -- height of plate
76 H_e = k -- height of element
77 H_dom = 20*mm -- height of domain
78
79 ----------------------------------------------------------------------
80 --DEFINING ALL POINTS
81
82 -- center of element
83 cen_el = ori+Vector3:new{x=pos_FP, y=0.0, z=0.0}
84
85 -- Leading edge plate (b)
86 --height z=1
87 bA1 = ori
88 bC1 = bA1+Vector3:new{x=0.0, y=W_h, z=0.0}
89 bD1 = bA1+Vector3:new{x=0.0, y=W_e, z=0.0}
90 bE1 = bA1+Vector3:new{x=0.0, y=W_dom, z=0.0}
91 --height z=0
92 bA0 = bA1-Vector3:new{x=0.0, y=0.0, z=H_p}
93 bC0 = bC1-Vector3:new{x=0.0, y=0.0, z=H_p}
94 bD0 = bD1-Vector3:new{x=0.0, y=0.0, z=H_p}
95 bE0 = bE1-Vector3:new{x=0.0, y=0.0, z=H_p}
96 --height z=2
97 bA2 = bA1+Vector3:new{x=0.0, y=0.0, z=H_e}
98 bC2 = bC1+Vector3:new{x=0.0, y=0.0, z=H_e}
99 bD2 = bD1+Vector3:new{x=0.0, y=0.0, z=H_e}
```

```
bE2 = bE1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
bA3 = bA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
bC3 = bC1+Vector3:new{x=0.0, y=0.0, z=H_dom}
bD3 = bD1+Vector3:new{x=0.0, y=0.0, z=H_dom}
bE3 = bE1+Vector3:new{x=0.0, y=0.0, z=H_dom}

-- Before leading edge plate (a)
--height z=1
aA1 = bA1-Vector3:new{x=pos_B, y=0.0, z=0.0}
aC1 = aA1+Vector3:new{x=0.0, y=W_h, z=0.0}
aD1 = aA1+Vector3:new{x=0.0, y=W_e, z=0.0}
aE1 = aA1+Vector3:new{x=0.0, y=W_dom, z=0.0}
--height z=0
aA0 = aA1-Vector3:new{x=0.0, y=0.0, z=H_p}
aC0 = aC1-Vector3:new{x=0.0, y=0.0, z=H_p}
aD0 = aD1-Vector3:new{x=0.0, y=0.0, z=H_p}
aE0 = aE1-Vector3:new{x=0.0, y=0.0, z=H_p}
--height z=2
aA2 = aA1+Vector3:new{x=0.0, y=0.0, z=H_e}
aC2 = aC1+Vector3:new{x=0.0, y=0.0, z=H_e}
aD2 = aD1+Vector3:new{x=0.0, y=0.0, z=H_e}
aE2 = aE1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
aA3 = aA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
aC3 = aC1+Vector3:new{x=0.0, y=0.0, z=H_dom}
aD3 = aD1+Vector3:new{x=0.0, y=0.0, z=H_dom}
aE3 = aE1+Vector3:new{x=0.0, y=0.0, z=H_dom}

-- Leading edge element (c)
--height z=1
cA1 = cen_el-Vector3:new{x=L_e, y=0.0, z=0.0}
--height z=2
cA2 = cA1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
cA3 = cA1+Vector3:new{x=0.0, y=0.0, z=H_dom}


```

```
-- negative half element (d)
--height z=1
dA1 = cen_el-Vector3:new{x=L_h, y=0.0, z=0.0}
dC1 = dA1+Vector3:new{x=0.0, y=W_h, z=0.0}
--height z=2
dA2 = dA1+Vector3:new{x=0.0, y=0.0, z=H_e}
dC2 = dC1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
dA3 = dA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
dC3 = dC1+Vector3:new{x=0.0, y=0.0, z=H_dom}

-- middle element (e)
--height z=1
eA1 = cen_el
eB1 = eA1+Vector3:new{x=0.0, y=W_ce, z=0.0}
eD1 = eA1+Vector3:new{x=0.0, y=W_e, z=0.0}
eE1 = eA1+Vector3:new{x=0.0, y=W_dom, z=0.0}
--height z=2
eA2 = eA1+Vector3:new{x=0.0, y=0.0, z=H_e}
eB2 = eB1+Vector3:new{x=0.0, y=0.0, z=H_e}
eD2 = eD1+Vector3:new{x=0.0, y=0.0, z=H_e}
eE2 = eE1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
eA3 = eA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
eB3 = eB1+Vector3:new{x=0.0, y=0.0, z=H_dom}
eD3 = eD1+Vector3:new{x=0.0, y=0.0, z=H_dom}
eE3 = eE1+Vector3:new{x=0.0, y=0.0, z=H_dom}

-- positive half element (f)
--height z=1
fA1 = cen_el+Vector3:new{x=L_h, y=0.0, z=0.0}
fC1 = fA1+Vector3:new{x=0.0, y=W_h, z=0.0}
--height z=2
fA2 = fA1+Vector3:new{x=0.0, y=0.0, z=H_e}
fC2 = fC1+Vector3:new{x=0.0, y=0.0, z=H_e}
--height z=3
fA3 = fA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
fC3 = fC1+Vector3:new{x=0.0, y=0.0, z=H_dom}
```

```
176
177 -- Trailing edge element (g)
178 --height z=1
179 gA1 = cen_el+Vector3:new{x=L_e, y=0.0, z=0.0}
180 --height z=2
181 gA2 = gA1+Vector3:new{x=0.0, y=0.0, z=H_e}
182 --height z=3
183 gA3 = gA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
184
185 -- End of domain plate (h)
186 --height z=1
187 hA1 = cen_el+Vector3:new{x=pos_AP, y=0.0, z=0.0}
188 hC1 = hA1+Vector3:new{x=0.0, y=W_h, z=0.0}
189 hD1 = hA1+Vector3:new{x=0.0, y=W_e, z=0.0}
190 hE1 = hA1+Vector3:new{x=0.0, y=W_dom, z=0.0}
191 --height z=2
192 hA2 = hA1+Vector3:new{x=0.0, y=0.0, z=H_e}
193 hC2 = hC1+Vector3:new{x=0.0, y=0.0, z=H_e}
194 hD2 = hD1+Vector3:new{x=0.0, y=0.0, z=H_e}
195 hE2 = hE1+Vector3:new{x=0.0, y=0.0, z=H_e}
196 --height z=3
197 hA3 = hA1+Vector3:new{x=0.0, y=0.0, z=H_dom}
198 hC3 = hC1+Vector3:new{x=0.0, y=0.0, z=H_dom}
199 hD3 = hD1+Vector3:new{x=0.0, y=0.0, z=H_dom}
200 hE3 = hE1+Vector3:new{x=0.0, y=0.0, z=H_dom}
201
202 ---------------------------------------------------------------------
203 -- DEFINING ALL VOLUMES
204 -- Volume names are defined with "vol_" followed with the outer points
       on each axis of the volume.
205 -- Volumes 01 (volumes beneath surface of the plate)
206 vol_abAC01 = TFIVolume:new{vertices={aA0, bA0, bC0, aC0, aA1, bA1, bC1,
       aC1}}
207 vol_abCD01 = TFIVolume:new{vertices={aC0, bC0, bD0, aD0, aC1, bC1, bD1,
       aD1}}
208 vol_abDE01 = TFIVolume:new{vertices={aD0, bD0, bE0, aE0, aD1, bD1, bE1,
       aE1}}
209
```

```
210 -- Volumes 12 (volumes on the surface of the plate)
211 -- x=a-b
212 vol_abAC12 = TFIVolume:new{vertices={aA1, bA1, bC1, aC1, aA2, bA2, bC2,
        aC2}}
213 vol_abCD12 = TFIVolume:new{vertices={aC1, bC1, bD1, aD1, aC2, bC2, bD2,
        aD2}}
214 vol_abDE12 = TFIVolume:new{vertices={aD1, bD1, bE1, aE1, aD2, bD2, bE2,
        aE2}}
215
216 -- x=b-e
217 vol_bdAC12 = TFIVolume:new{vertices={bA1, cA1, dC1, bC1, bA2, cA2, dC2,
        bC2}}
218 vol_beCD12 = TFIVolume:new{vertices={bC1, dC1, eD1, bD1, bC2, dC2, eD2,
        bD2}}
219 vol_beDE12 = TFIVolume:new{vertices={bD1, eD1, eE1, bE1, bD2, eD2, eE2,
        bE2}}
220
221 -- x=f-h
222 vol_fhAC12 = TFIVolume:new{vertices={gA1, hA1, hC1, fC1, gA2, hA2, hC2,
        fC2}}
223 vol_ehCD12 = TFIVolume:new{vertices={fC1, hC1, hD1, eD1, fC2, hC2, hD2,
        eD2}}
224 vol_ehDE12 = TFIVolume:new{vertices={eD1, hD1, hE1, eE1, eD2, hD2, hE2,
        eE2}}
225
226 -- Volumes 23 (volumes above the height of the top of the element)
227 -- x=a-b
228 vol_abAC23 = TFIVolume:new{vertices={aA2, bA2, bC2, aC2, aA3, bA3, bC3,
        aC3}}
229 vol_abCD23 = TFIVolume:new{vertices={aC2, bC2, bD2, aD2, aC3, bC3, bD3,
        aD3}}
230 vol_abDE23 = TFIVolume:new{vertices={aD2, bD2, bE2, aE2, aD3, bD3, bE3,
        aE3}}
231
232 -- x=b-e
233 vol_bdAC23 = TFIVolume:new{vertices={bA2, cA2, dC2, bC2, bA3, cA3, dC3,
        bC3}}
```

```
234 vol_beCD23 = TFIVolume:new{vertices={bC2, dC2, eD2, bD2, bC3, dC3, eD3,
        bD3}}
235 vol_beDE23 = TFIVolume:new{vertices={bD2, eD2, eE2, bE2, bD3, eD3, eE3,
        bE3}}
236
237 -- x=f-h
238 vol_fhAC23 = TFIVolume:new{vertices={gA2, hA2, hC2, fC2, gA3, hA3, hC3,
        fC3}}
239 vol_ehCD23 = TFIVolume:new{vertices={fC2, hC2, hD2, eD2, fC3, hC3, hD3,
        eD3}}
240 vol_ehDE23 = TFIVolume:new{vertices={eD2, hD2, hE2, eE2, eD3, hD3, hE3,
        eE3}}
241
242 -- element
243 vol_ceAC23 = TFIVolume:new{vertices={cA2, eA2, eB2, dC2, cA3, eA3, eB3,
        dC3}}
244 vol_dfBD23 = TFIVolume:new{vertices={dC2, eB2, fC2, eD2, dC3, eB3, fC3,
        eD3}}
245 vol_egAC23 = TFIVolume:new{vertices={eA2, gA2, fC2, eB2, eA3, gA3, fC3,
        eB3}}
246
247 ----------------------------------------------------------------------
248 ----------------------------------------------------------------------
249 -- NUMBER OF CELLS
250 -- In this section the number of cells in each direction is specified.
251 -- Only this section needs to be adjusted on order to make the grid
       coarser/finer.
252 -- The name starts with "n" followed by the outer points of the line
       segment
253 -- x-direction
254 nab = 3--10
255 nbc = 4--55
256 ngh = 5--110
257
258 -- y-direction
259 nAC = 3--20
260 nCD = 3--20
261 nDE = 6--70
```

```
262
263 -- z-direction
264 n01 = 3--10
265 n12 = 3--15
266 n23 = 6--55
267
268 ------------------------------------------------------------------------
269 -- Calculating the total number of cells
270 ncells_ab = nab*(nAC+nCD+nDE)*(n01+n12+n23)
271 ncells_bc = nbc*(nAC+nCD+nDE)*(n12+n23)
272 ncells_gh = ngh*(nAC+nCD+nDE)*(n12+n23)
273 ncells_ele = (nAC*nCD*2+nCD*nCD)*n23
274 ncells_tot = ncells_ab+ncells_bc+ncells_gh+ncells_ele
275 print("Number of cells= ", ncells_tot)
276
277 ------------------------------------------------------------------------
278 ------------------------------------------------------------------------
279 -- GRIDS
280 -- Grids are the most difficult part to define in a 3D simulation with
      Eilmer4.
281 -- In most cases one wants to apply grid refinements around critical
      parts in the simulation.
282 -- Robertsfunctions are used to make cells bigger or smaller along a
      line segment.
283 -- The difficulty is that the robertsfunction on a particular
      linesegment can be shared by up to 4 volumes.
284 -- One really has to make sure that the above is applied otherwise cell
       misalignments will occur.
285
286 -- Robertsfunctions are defined with "rcfxxYY00"
287 -- Where xx is the domain on the x-axis, YY the domain on the y-axis
      and 00 the domain on the z-axis.
288 -- If a single letter or number is used, then it is only valid on that
      particular line.
289 -- For example:
290 -- rcfabAE03
291 -- ab stands for the linesegment between a and b
```

```
292 -- AE means that it is valid for all ab linesegments between A and E on
        the y-axis
293 -- 03 means that it is valid for all ab linesegments between 0 and 3 on
        the z-axis
294
295 -- To collect all the robertsfunctions on the correct edges of a volume
    , they are stored in "cf_xxYY00" variable.
296 -- The same naming technic as for the robertsfunction itself is applied
    .
297
298 -- To adjust the gridrefinement only the robertsfunction needs to be
    adjusted.
299
300 -- Grids 01 (grids beneath surface of the plate)
301
302 rcfabAE03 = RobertsFunction:new{end0=false, end1=true, beta=1.05}
303
304 cf_abAE02 = {edge01=rcfabAE03, edge32=rcfabAE03, edge45=rcfabAE03,
    edge76=rcfabAE03}
305
306 grid_abAC01 = StructuredGrid:new{pvolume=vol_abAC01,
307                                  cfList=cf_abAE02,
308                                  niv=nab+1, njv=nAC+1, nkv=n01+1}
309 grid_abCD01 = StructuredGrid:new{pvolume=vol_abCD01,
310                                  cfList=cf_abAE02,
311                                  niv=nab+1, njv=nCD+1, nkv=n01+1}
312 grid_abDE01 = StructuredGrid:new{pvolume=vol_abDE01,
313                                  cfList=cf_abAE02,
314                                  niv=nab+1, njv=nDE+1, nkv=n01+1}
315
316
317 -- Grids 12 (grids on the surface of the plate)
318 -- ab
319
320 grid_abAC12 = StructuredGrid:new{pvolume=vol_abAC12,
321                                  cfList=cf_abAE02,
322                                  niv=nab+1, njv=nAC+1, nkv=n12+1}
323 grid_abCD12 = StructuredGrid:new{pvolume=vol_abCD12,
```

```
                                    cfList=cf_abAE02,
                                    niv=nab+1, njv=nCD+1, nkv=n12+1}
grid_abDE12 = StructuredGrid:new{pvolume=vol_abDE12,
                                    cfList=cf_abAE02,
                                    niv=nab+1, njv=nDE+1, nkv=n12+1}

-- be
rcfbcA12 = RobertsFunction:new{end0=true, end1=true, beta=1.01}
rcfbdC12 = RobertsFunction:new{end0=true, end1=true, beta=1.01}
rcfbeD12 = RobertsFunction:new{end0=true, end1=true, beta=1.01}
rcfbeE12 = RobertsFunction:new{end0=true, end1=true, beta=1.07}

-- For the specific case there is symmetry at the element and therefore
    the robertsfunctions for the edges
-- only have rcfYY00 markings. Adding xx markings will make things more
    complicated.
rcfAC12 = RobertsFunction:new{end0=true, end1=false, beta=1.1}

-- linesegment eDfC is special since it is the north edge of
   quadrilateral dC-eB-fC-eD and the
-- western edge of quadrilateral fC-hC-hD-eD.
-- Therefore in quadrilateral dC-eB-fC-eD eD to fC is positive in the
   local coordinate system (x-axis)
      -- and in quadrilateral fC-hC-hD-eD fC to eD is positive in the
         local coordinate system (y-axis)
-- Therefore the robertsfunction in the lignsegment is mirrored  and
   the beta is defined seperately
beta_CD12 = 1.1
rcfCD12 = RobertsFunction:new{end0=false, end1=true, beta=beta_CD12}
rcfCD12_mir = RobertsFunction:new{end0=true, end1=false, beta=beta_CD12
   }
rcfDE12 = RobertsFunction:new{end0=true, end1=false, beta=1.05}

cf_bdAC12 = {edge01=rcfbcA12, edge32=rcfbdC12, edge45=rcfbcA12, edge76=
   rcfbdC12, edge12=rcfAC12, edge56=rcfAC12}
cf_beCD12 = {edge01=rcfbdC12, edge32=rcfbeD12, edge45=rcfbdC12, edge76=
   rcfbeD12, edge12=rcfCD12, edge56=rcfCD12}
```

```
cf_beDE12 = {edge01=rcfbeD12, edge32=rcfbeE12, edge45=rcfbeD12, edge76=
   rcfbeE12, edge12=rcfDE12, edge56=rcfDE12}
grid_bdAC12 = StructuredGrid:new{pvolume=vol_bdAC12,
                                 cfList=cf_bdAC12,
                                 niv=nbc+1, njv=nAC+1, nkv=n12+1}
grid_beCD12 = StructuredGrid:new{pvolume=vol_beCD12,
                                 cfList=cf_beCD12,
                                 niv=nbc+1, njv=nCD+1, nkv=n12+1}
grid_beDE12 = StructuredGrid:new{pvolume=vol_beDE12,
                                 cfList=cf_beDE12,
                                 niv=nbc+1, njv=nDE+1, nkv=n12+1}
-- eh
rcfghA12 = RobertsFunction:new{end0=true, end1=false, beta=1.03}
rcffhC12 = RobertsFunction:new{end0=true, end1=false, beta=1.03}
rcfehD12 = RobertsFunction:new{end0=true, end1=false, beta=1.03}
rcfehE12 = RobertsFunction:new{end0=true, end1=false, beta=1.07}
cf_fhAC12 = {edge01=rcfghA12, edge32=rcffhC12, edge45=rcfghA12, edge76=
   rcffhC12, edge03=rcfAC12, edge47=rcfAC12}
cf_ehCD12 = {edge01=rcffhC12, edge32=rcfehD12, edge45=rcffhC12, edge76=
   rcfehD12, edge03=rcfCD12, edge47=rcfCD12}
cf_ehDE12 = {edge01=rcfehD12, edge32=rcfehE12, edge45=rcfehD12, edge76=
   rcfehE12, edge03=rcfDE12, edge47=rcfDE12}
grid_fhAC12 = StructuredGrid:new{pvolume=vol_fhAC12,
                                 cfList=cf_fhAC12,
                                 niv=ngh+1, njv=nAC+1, nkv=n12+1}
grid_ehCD12 = StructuredGrid:new{pvolume=vol_ehCD12,
                                 cfList=cf_ehCD12,
                                 niv=ngh+1, njv=nCD+1, nkv=n12+1}
grid_ehDE12 = StructuredGrid:new{pvolume=vol_ehDE12,
                                 cfList=cf_ehDE12,
                                 niv=ngh+1, njv=nDE+1, nkv=n12+1}

-- Grids 23 (grids above the height of the top of the element)
-- ab
rcf23 = RobertsFunction:new{end0=true, end1=false, beta=1.03} --for z-
   axis linesegments
cf_abAE23 = {edge01=rcfabAE03, edge32=rcfabAE03, edge45=rcfabAE03,
   edge76=rcfabAE03, edge04=rcf23, edge15=rcf23, edge26=rcf23, edge37=
```

```
   rcf23}
grid_abAC23 = StructuredGrid:new{pvolume=vol_abAC23,
                                 cfList=cf_abAE23,
                                 niv=nab+1, njv=nAC+1, nkv=n23+1}
grid_abCD23 = StructuredGrid:new{pvolume=vol_abCD23,
                                 cfList=cf_abAE23,
                                 niv=nab+1, njv=nCD+1, nkv=n23+1}
grid_abDE23 = StructuredGrid:new{pvolume=vol_abDE23,
                                 cfList=cf_abAE23,
                                 niv=nab+1, njv=nDE+1, nkv=n23+1}
-- be
rcfbc3 = RobertsFunction:new{end0=true, end1=true, beta=1.05}
rcfbd3 = RobertsFunction:new{end0=true, end1=true, beta=1.05}
rcfbe3 = RobertsFunction:new{end0=true, end1=true, beta=1.05}
cf_bdAC23 = {edge01=rcfbcA12, edge32=rcfbdC12, edge45=rcfbc3, edge76=
   rcfbd3, edge12=rcfAC12, edge04=rcf23, edge15=rcf23, edge26=rcf23,
   edge37=rcf23}
cf_beCD23 = {edge01=rcfbdC12, edge32=rcfbeD12, edge45=rcfbd3, edge76=
   rcfbe3, edge12=rcfCD12, edge04=rcf23, edge15=rcf23, edge26=rcf23,
   edge37=rcf23}
cf_beDE23 = {edge01=rcfbeD12, edge32=rcfbeE12, edge45=rcfbe3,
                  edge12=rcfDE12, edge04=rcf23, edge15=rcf23, edge26=
   rcf23, edge37=rcf23}
grid_bdAC23 = StructuredGrid:new{pvolume=vol_bdAC23,
                                 cfList=cf_bdAC23,
                                 niv=nbc+1, njv=nAC+1, nkv=n23+1}
grid_beCD23 = StructuredGrid:new{pvolume=vol_beCD23,
                                 cfList=cf_beCD23,
                                 niv=nbc+1, njv=nCD+1, nkv=n23+1}
grid_beDE23 = StructuredGrid:new{pvolume=vol_beDE23,
                                 cfList=cf_beDE23,
                                 niv=nbc+1, njv=nDE+1, nkv=n23+1}

-- eh
rcfgh3 = RobertsFunction:new{end0=true, end1=false, beta=1.05}
rcffh3 = RobertsFunction:new{end0=true, end1=false, beta=1.05}
rcfeh3 = RobertsFunction:new{end0=true, end1=false, beta=1.05}
```

```
414 cf_fhAC23 = {edge01=rcfghA12, edge32=rcffhC12, edge45=rcfgh3, edge76=
       rcffh3, edge03=rcfAC12, edge04=rcf23, edge37=rcf23}
415 cf_ehCD23 = {edge01=rcffhC12, edge32=rcfehD12, edge45=rcffh3, edge76=
       rcfeh3, edge03=rcfCD12, edge04=rcf23, edge37=rcf23}
416 cf_ehDE23 = {edge01=rcfehD12, edge32=rcfehE12, edge45=rcfeh3,
                    edge03=rcfDE12, edge04=rcf23, edge37=rcf23}
417 grid_fhAC23 = StructuredGrid:new{pvolume=vol_fhAC23,
418                                  cfList=cf_fhAC23,
419                                  niv=ngh+1, njv=nAC+1, nkv=n23+1}
420 grid_ehCD23 = StructuredGrid:new{pvolume=vol_ehCD23,
421                                  cfList=cf_ehCD23,
422                                  niv=ngh+1, njv=nCD+1, nkv=n23+1}
423 grid_ehDE23 = StructuredGrid:new{pvolume=vol_ehDE23,
424                                  cfList=cf_ehDE23,
425                                  niv=ngh+1, njv=nDE+1, nkv=n23+1}
426
427 -- element
428 rcf_ceA2 = rcfCD12_mir
429 rcf_deBC2 = rcfCD12_mir
430 rcf_egA2 = rcfCD12
431 rcf_efBC2 = rcfCD12
432 cf_ceAC23 = {edge01=rcf_ceA2,  edge32=rcf_deBC2,  edge03=rcfAC12,
                      edge04=rcf23, edge15=rcf23, edge26=rcf23, edge37=
       rcf23}
433 cf_dfBD23 = {edge01=rcf_deBC2, edge32=rcfCD12_mir, edge03=rcfCD12,
       edge12=rcf_efBC2, edge04=rcf23, edge15=rcf23, edge26=rcf23, edge37=
       rcf23}
434 cf_egAC23 = {edge01=rcf_egA2,  edge32=rcf_efBC2,
       edge12=rcfAC12,  edge04=rcf23, edge15=rcf23, edge26=rcf23, edge37=
       rcf23}
435 grid_ceAC23 = StructuredGrid:new{pvolume=vol_ceAC23,
436                                  cfList=cf_ceAC23,
437                                  niv=nCD+1, njv=nAC+1, nkv=n23+1}
438 grid_dfBD23 = StructuredGrid:new{pvolume=vol_dfBD23,
439                                  cfList=cf_dfBD23,
440                                  niv=nCD+1, njv=nCD+1, nkv=n23+1}
441 grid_egAC23 = StructuredGrid:new{pvolume=vol_egAC23,
442                                  cfList=cf_egAC23,
```

```
443                                     niv=nCD+1, njv=nAC+1, nkv=n23+1}
444
445 -- Flow blocks
446 -- The flow blocks are rather easy to define compared to the grid. The
    definition of the blocks is the same as
447 -- with the grids and volumes.
448 -- There are only two types of boundary conditions: Supersonic inflow
    and wall with no slip.
449 -- The sharp edge after the element (eD2 till gA2) gives troubles at
    the start if a
450 -- inflow initialstate is defined. Therefore a noflow initialstate is
    used with zero velocity.
451 -- To calculate heat transfer all walls in the xy-plane have a group
    where information about the loads is stored.
452 -- Block 01 (blocks beneath surface of the plate)
453 blk_abAC01 = FluidBlockArray{grid=grid_abAC01, initialState=inflow, nib
    =1, njb=1,
454                               bcList={west=InFlowBC_Supersonic:new{
                                      flowState=inflow},
455                                     east=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w}}} --b0000
456 blk_abCD01 = FluidBlockArray{grid=grid_abCD01, initialState=inflow, nib
    =1, njb=1,
457                               bcList={west=InFlowBC_Supersonic:new{
                                      flowState=inflow},
458                                     east=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w}}} --b0001
459 blk_abDE01 = FluidBlockArray{grid=grid_abDE01, initialState=inflow, nib
    =1, njb=1,
460                               bcList={west=InFlowBC_Supersonic:new{
                                      flowState=inflow},
461                                     east=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w}}} --b0002
462
463 -- Block 12 (blocks on the surface of the plate)
464 -- ab
465 blk_abAC12 = FluidBlockArray{grid=grid_abAC12, initialState=inflow, nib
    =1, njb=1,
```

```
466                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0003
467 blk_abCD12 = FluidBlockArray{grid=grid_abCD12, initialState=inflow, nib
    =1, njb=1,
468                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0004
469 blk_abDE12 = FluidBlockArray{grid=grid_abDE12, initialState=inflow, nib
    =1, njb=1,
470                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0005
471
472 -- be
473 blk_bdAC12 = FluidBlockArray{grid=grid_bdAC12, initialState=inflow, nib
    =1, njb=1,
474                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"},
475                                             east=WallBC_NoSlip_FixedT:new{
                                                Twall=T_w}}} --b0006
476 blk_beCD12 = FluidBlockArray{grid=grid_beCD12, initialState=inflow, nib
    =1, njb=1,
477                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"},
478                                             east=WallBC_NoSlip_FixedT:new{
                                                Twall=T_w}}} --b0007
479 blk_beDE12 = FluidBlockArray{grid=grid_beDE12, initialState=inflow, nib
    =1, njb=1,
480                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"}}} --b0008
481
482 -- eh
483 blk_fhAC12 = FluidBlockArray{grid=grid_fhAC12, initialState=noflow, nib
    =1, njb=1,
484                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"},
485                                             west=WallBC_NoSlip_FixedT:new{
                                                Twall=T_w},
486                                             east=OutFlowBC_Simple:new{}}} --
                                                b0009
```

```
487 blk_ehCD12 = FluidBlockArray{grid=grid_ehCD12, initialState=noflow, nib
    =1, njb=1,
488                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"},
489                                             west=WallBC_NoSlip_FixedT:new{
                                                Twall=T_w},
490                                             east=OutFlowBC_Simple:new{}}} --
                                                b0010
491 blk_ehDE12 = FluidBlockArray{grid=grid_ehDE12, initialState=noflow, nib
    =1, njb=1,
492                                 bcList={bottom=WallBC_NoSlip_FixedT:new{
                                        Twall=T_w, group="loads"},
493                                             east=OutFlowBC_Simple:new{}}} --
                                                b0011
494
495 -- Block 23 (blocks above the height of the top of the element)
496 -- ab
497 blk_abAC23 = FluidBlockArray{grid=grid_abAC23, initialState=inflow, nib
    =1, njb=1,
498                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0012
499 blk_abCD23 = FluidBlockArray{grid=grid_abCD23, initialState=inflow, nib
    =1, njb=1,
500                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0013
501 blk_abDE23 = FluidBlockArray{grid=grid_abDE23, initialState=inflow, nib
    =1, njb=1,
502                                 bcList={west=InFlowBC_Supersonic:new{
                                        flowState=inflow}}} --b0014
503
504 -- be
505 blk_bdAC23 = FluidBlockArray{grid=grid_bdAC23, initialState=inflow, nib
    =1, njb=1} --b0015
506 blk_beCD23 = FluidBlockArray{grid=grid_beCD23, initialState=inflow, nib
    =1, njb=1} --b0016
507 blk_beDE23 = FluidBlockArray{grid=grid_beDE23, initialState=inflow, nib
    =1, njb=1} --b0017
508
```

```
509 -- eh
510 blk_fhAC23 = FluidBlockArray{grid=grid_fhAC23, initialState=noflow, nib
      =1, njb=1,
511                               bcList={east=OutFlowBC_Simple:new{}}} --
                                    b0018
512 blk_ehCD23 = FluidBlockArray{grid=grid_ehCD23, initialState=noflow, nib
      =1, njb=1,
513                               bcList={east=OutFlowBC_Simple:new{}}} --
                                    b0019
514 blk_ehDE23 = FluidBlockArray{grid=grid_ehDE23, initialState=noflow, nib
      =1, njb=1,
515                               bcList={east=OutFlowBC_Simple:new{}}} --
                                    b0020
516
517 -- element
518 blk_ceAC23 = FluidBlockArray{grid=grid_ceAC23, initialState=inflow, nib
      =1, njb=1,
519                               bcList={bottom=WallBC_NoSlip_FixedT:new{
                                    Twall=T_w, group="loads"}}} --b0021
520 blk_dfBD23 = FluidBlockArray{grid=grid_dfBD23, initialState=inflow, nib
      =1, njb=1,
521                               bcList={bottom=WallBC_NoSlip_FixedT:new{
                                    Twall=T_w, group="loads"}}} --b0022
522 blk_egAC23 = FluidBlockArray{grid=grid_egAC23, initialState=inflow, nib
      =1, njb=1,
523                               bcList={bottom=WallBC_NoSlip_FixedT:new{
                                    Twall=T_w, group="loads"}}} --b0023
524
525 identifyBlockConnections()
526
527 -- History Points
528 -- History points are used to give more information over the
      development in time of critical points.
529
530 setHistoryPoint{x=0.0,y=0.0,z=0.0} -- bA1
531 setHistoryPoint{x=pos_FP-L_e,y=0.0,z=0.0} -- cA1
532 setHistoryPoint{x=pos_FP-L_e,y=0.0,z=k} -- cA2
533 setHistoryPoint{x=pos_FP,y=W_e,z=0.0} -- eD1
```

```
534 setHistoryPoint{x=pos_FP,y=W_e,z=k} -- eD2
535 setHistoryPoint{x=pos_FP+L_e,y=0.0,z=0.0} -- eD1
536 setHistoryPoint{x=pos_FP+L_e,y=0.0,z=k} -- eD2
537 setHistoryPoint{x=pos_FP+5*k,y=0.0,z=0.0} -- after element (5k)
538 setHistoryPoint{x=pos_FP+10*k,y=0.0,z=0.0} -- after element (10K)
539
540 -- Final configurations
541 mpiTasks = mpiDistributeBlocks(6)
542 config.gasdynamic_update_scheme = "euler"
543 config.flux_calculator = 'adaptive'
544 config.viscous = true
545 config.spatial_deriv_calc = 'divergence'
546 config.cfl_value = 0.3
547 config.max_time = 6.0e-5
548 config.max_step = 200000
549 config.dt_init = 1.0e-8
550 config.dt_plot = 5.0e-6
551 config.dt_history = config.max_time/100
```

Appendix B

REFLECTION

B.1 My time at UQ

The time flew during my internship. A little tense, I arrived at the campus at the University of Queensland to first meet Peter and before I knew it I already had my final meeting with him. It all went so fast and of course partly because I really enjoyed it. The Centre for Hypersonics is a great research group with high quality research on very interesting yet abstract subjects. I really liked their Thursday meetings. In this meetings one of the staff prepares a presentation about what he or she is doing, or what he has done since his last presentation. It can be about anything with only one conditions: it should not take anymore than around 15 min of preparation (although I doubt if they really spend this few time on it). PhD students told about their project development, the director Prof. Morgan told about a project he once did abroad in cooperation with NASA, but there was also a presentation about safety. There was a big variety and I really enjoyed it. For me it was a great way to get in touch with the people of the group and to learn more about the interesting things this part of science has to offer. For them it is also a good way to interact with their colleagues, learn from them or be critic to them. I think it only has advantages and I am happy that the Thermal and Fluid Engineering department also started doing these. The supervision of Peter and Rowan was really good. Although it sometimes was a bit hard to find a free spot in their crowded agendas, they helped me whenever they could.

I was placed in a room with many PhD students. Unfortunately they were from other departments and they also were not very talkative. Luckily the last few weeks some people from the Centre for Hypersonics group were moved to my room. It was great to brainstorm about some problems or to ask for a little assistance with Linux so that I did not have to contact Peter or Rowan for small problems. I was so sucked up by my project that I almost forgot to see the expansion tubes at work. Luckily on my final day I could visit one of the experiments.

B.2 Objectives

My main objective was to validate the results of Sandy Tirtey's experiment with a Eilmer4 simulation. To reach this I had to learn 2 new programming languages (LUA and D) and I had to learn to work with the Linux system. I also had to get familiar with this topic. Hypersonic flow was something completely new for me. Another objective for me personally

was to improve my English since I seldom speak it in the Netherlands.

B.3 Approach

The main objective itself does not seem like much work, but the conditions to reach this objective made it really big. I approached it very systematically. First I wanted to get familiar with the topic. Of course this is the most obvious thing to do. I started reading the papers and more background. It is irrelevant to mention these in this report, because I did not specifically use them, but they helped me to get more general knowledge about the topic.

I was lucky that Rowan was teaching a course about Eilmer4 to third year students, so I could attend his lectures. He told a lot about the background of the program. This was very recognizable with the Computational Fluid Dynamics course taught at the University of Twente.

With this lectures and with some examples listed in the Eilmer4 user-guide [6] and Geometry Package user-guide [5] the Lua language was learned quite fast. Also the D-language was not a really big problem mainly because you do not need to know that much about it to let Eilmer4 work. I wanted to set up the simulation very systematically. First the $2D$ simulation. It was not that hard to build the script for this simulation with the help of the user-guide and the examples in it. These first simulations gave me insight in how the program responded

Then we could move on to the $3D$ simulation. Before writing a single line of code I spend some time in drawing the problem and designing a system which is clear. The first simulations learned me that mistakes are easily made and that it should therefore be a robust system. I was really happy that I could run the simulation after only fixing 3 minor errors in the whole code. After analyzing some results I knew that there were more errors, but I made the system such that it was simply solvable.

Then another big challenge came. Now that the system worked, I had to refine the grid so I could get some results which are comparable with Tirtey's work. This computational time skyrocketed meaning that I had to wait a day for one simulation. At a point also my desktops computational power wasn't sufficient anymore and therefore it aborted my simulation. I moved on to High Performance Computing (HPC). I had to run my simulations through a virtual machine on the Goliath Faculty Cluster. I struggled a lot with the Linux environment and the commands. It is such a difference with the Windows environment. This meant that I had to search a lot for commands and ask a lot of help from Rowan and Peter. Nonetheless we managed to get the simulations running. Soon computation times reached 1.5 days and before I knew it I reached the end of my internship. Still not really satisfied with the accuracy

of the result, but my internship had ended, so I had to accept it.

B.4 Reflection

If I think back on my internship at the University of Queensland, I can be quite proud of myself solving this with not much assistance. In my opinion I really systematically approached the problem. Starting slow with small steps exploring the problem and not immediately diving straight into the problem. I constantly was critic to myself asking myself questions if what I produced was correct or if this was the ideal way to achieve things.

The thing I am less satisfied with is the end of my internship. I like to work things out myself. I do not want to bother other people with tasks assigned to me even though they are here to help me. I think I had to ask earlier for help with the Linux problems I had. This cost me some valuable time.

Another thing is that I only want to settle for the best and that I always keep trying to reach this. On itself this is a good property, but I think I got carried away with it. I constantly wanted to refine my mesh or adjust some settings to get better results. With computation times of almost 2 days time flew and before I knew it it was my final day. My last simulation only ended the day before I finished my internship. During the time the HPC was simulating I was constantly trying to think of ways to improve my simulation. I should not have spend so much time on this and instead started thoroughly analyzing the results and comparing them with Tirtey's work. Now this had to be done after my internship.

The last thing I want to mention here is my report. It is not in the objective of this research, but it is obviously a part of it. I knew I am not good at writing big reports. I should have started earlier with this. Now I started at the end of my internship with this, but I should have started way earlier and write it in small manageable pieces.

B.5 Goals

I am really happy with the way I approached the problem. This is one of the first times in your study that you actually can work on a bigger problem all by yourself. I am satisfied how I managed to systematically solve this and I definitely want to apply this in my graduation assignment or any other project.

The thing I want to improve is to ask for help earlier. It is good to try to work things out on my own, but it is no shame to ask for some help. Another thing is that I sometimes have to take things for granted. It is good to strive for the best, but if you end up with months of delay it is not worth it. I have to keep in mind the bigger picture.

The final point for improvement is my reporting. I know I have difficulties writing a big report, so I really have to split it in smaller manageable pieces. With my graduation assign-

ment of 9 months coming up I really want to finish chapters of my report within a week as I end certain parts of my research.

B.6 Feedback Peter

On my last day at the University of Queensland I had a lunch with Peter where we spoke about my internship. We spoke about the work I had done and how this contributed to the research they do. I felt that I did not achieved my goal to compare the research of Tirtey with my results, but Peter asked me where I was proud of. This is the robust system I developed to make a grid and system in $3D$ for an Eilmer4 simulation. He said that it was a good thing and that I should not worry much about the results. I have worked hard to do a complicated $3D$ simulation in program I had no knowledge about beforehand. The program is in development and with my help they removed some bugs. Another thing is that they want to add my simulation as an example so other people can see how to approach this kind of $3D$ problem in Eilmer4. He complimented me on my independency, but as said above I think I was a bit to independent and I should have asked for more help to get more results.

Figure B.1: Me at the final day in front of the beautiful University of Queensland campus