CHALMERS
UNIVERSITY OF TECHNOLOGY

Network Modelling of Cooling of Hydro-Generators

A Thermal and Fluid Model

Master's thesis in Applied Mechanics

MARC CABRÉ GIMENO

Department of Applied Mechanics
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Master's thesis 2017:45

Network Modelling of Cooling of Hydro-Generators

A Thermal and Fluid Model

MARC CABRÉ GIMENO

Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
Gothenburg, Sweden 2017

Network Modelling of Cooling of Hydro-Generators
A Thermal and Fluid Model
MARC CABRÉ GIMENO

Supervisors: Håkan Nilsson, Bercelay Niebles Atencio, Daniel Rundström (Voith Hydro AB)
Examiner: Håkan Nilsson, Department of Applied Mechanics

Master's Thesis 2017:45
Department of Applied Mechanics
Division of Fluid Dynamics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2017

Network Modelling of Cooling of Hydro-Generators
A Thermal and Fluid Model
MARC CABRÉ GIMENO
Department of Applied Mechanics
Chalmers University of Technology

Abstract

Air-cooling is the most common way of cooling large generators like those used in hydropower. However, because these machines are quite complex and therefore, understanding the cooling of these machines is not easy. The details of the air flow inside of the generator is still a field in which the academics are doing research and developing more detailed and accurate CFD studies. The downside of these CFD studies is that they often involve the use of more computational resources and time. This lack of versatility of CFD studies makes its use impossible for daily small design tests. In this area, the lumped parameter thermal models are the tools used to get fast and good enough results that allow designers to test little modifications.
The present work consist of the development of a versatile lumped parameter model, implemented in Matlab/Simulink, to study the cooling of hydro-generators from a thermal and fluid perspective, contributing to future ventilation studies and research. The model created is based on a 3D network modelling of a Voith Hydro real generator, using lumped element strategy and is able to run transient simulations in a very short amount of time. The model is validated with real thermal data and is finally used to perform a parametric study of the outlets of the stator cooling ducts of the modelled generator.

Keywords: Hydropower, Generator, Cooling, Ventilation, Lumped element, Thermal network, Fluid network, Transient simulation, 3D network modelling, parametric study.

Acknowledgements

I would like to thank everyone that has contributed directly or indirectly in this master thesis; professors, colleagues, friends and family. In particular, I would like to thank my supervisors Håkan Nilsson and Bercelay Niebles Atencio for their support and assistance. I would also like to show my acknowledgement to Daniel Rundström and Voith Hydro for the data provided and the helpful assistance, especially in the validation.

Marc Cabré Gimeno, Gothenburg, June 2017

Contents

1 Introduction **1**
1.1 Background . . . 2
1.2 Parts of a generator . . . 2
1.2.1 Rotor . . . 3
1.2.2 Stator . . . 3
1.2.3 Cooling system . . . 4
1.3 The network model . . . 5
1.4 Scope of the thesis . . . 6

2 Thermal network model **7**
2.1 Basics of heat transfer . . . 7
2.2 General conduction equation . . . 8
2.3 Nodal implementation . . . 8
2.3.1 Thermal resistances . . . 8
2.3.2 Heat balance . . . 10
2.3.3 Heat generation . . . 12
2.3.4 Boundary conditions . . . 12

3 Ventilation model **17**
3.1 Fluid dynamics basics . . . 17
3.2 Heat transfer coefficient calculation . . . 21
3.3 Temperature increase in the channel . . . 22

4 Model overview **25**

5 Results **29**
5.1 Validation . . . 29
5.2 Effect of changing the outlet width . . . 32

6 Conclusion **37**
6.1 Validation . . . 37
6.2 Effect of changing the outlet width . . . 37
6.3 Possible future work . . . 38

Bibliography **39**

A Appendix **I**

A.1 inputgen.m I
A.2 executemod.m III
A.3 netmod.m V
A.4 netflowc.m XXXI
A.5 fhtc.m XXXVI
A.6 linknodes.m XXXVII

1
Introduction

Over the lasts years, the efficient production of energy has become one of the biggest challenges to become energetically sustainable. In order to reduce the emissions of CO_2 and stop the global warming, the International Energy Agency (IEA) [13] considers mandatory to stop increasing the usage of carbon based energy sources. In its world energy outlook, the IEA recommends to absorb the future demand of energy only by the use of renewable energy. In recent literature, the improvement of efficiency in old facilities is considered as one of the future sources of energy [7]. In this increasing of efficiency electric generators are crucial. Electric generators are machines that are used to transform mechanical energy into electrical energy. They are the main players in energy production present in nuclear, hydropower, wind-power, coal and gas energy. Three main sources of energy losses can be identified in the generator: electrical losses in the windings, magnetic losses mainly, and mechanical losses. The electrical losses are produced in the windings of the stator and rotor if they have. Electrical losses are produced by the resistance of the conductor

$$\dot{g} = R \cdot I^2 \tag{1.1}$$

being $\dot{g}$ the losses, I the intensity of current and R the resistance of the conductor that is affected by the temperature

$$R = R_0 \cdot (1 + \alpha \cdot (T - T_0)) \tag{1.2}$$

increasing its electrical resistance when the temperature increases. The magnetic losses are produced in the core and windings. On the other hand the mechanical losses occur in all the components that have friction, as the bearings and by the ventilation system. The losses generate heat, and the consequence is a rising temperature which has a negative effect on the generator: deterioration of the electrical insulation of the windings and an increase of the losses of the conductors (as equations 1.1 and 1.2 show) leading to a lower efficiency. For these reasons, it is important to include an efficient cooling system when designing the generator ensuring low temperatures and minimising the ventilation losses. As designing an electrical generator implies extensive multidisciplinary teams it is important to apply fast and easy to use tools for the design of the cooling system and to check the thermal behaviour in every step.

1.1 Background

Hydropower plants extract the power of flowing water due to a height difference from the water surface before and after the plants (with the usage of dams) or from the flow speed (Run-of-the-river plants). It is a renewable energy even if some facilities can have a negative impact in the ecosystem but can be mitigate with effort [10]. Nowadays, hydropower is responsible of 16% of the entire world electricity production [8] and it is predicted to increase in the future. Its role to balance the electric grid will be as well increased in the future. The balance is both in terms of storage and frequency [9]. This means that the hydropower electric generators will be working more in transient conditions than they were designed for.

The motivation to improve the performance of hydro-generators in Sweden is explained by the fact that hydropower represents 41.6% of the electricity production in Sweden [1]. During the period between 2003 to 2012 the redesign of Swedish hydro-generators has increased the hydropower production 5% [10].

As a main actor in charge to collect the energy as the electric generator is, that nowadays can produce dozens of megawatts, they are a target for improvements. The increase of its efficiency, the study of transient conditions and new upgrades to old generators increase the interest in new thermal and ventilation studies and research. Traditionally, thermal analysis can be classified in two big groups [6]:

- **Analytical methods**: The most common is the usage of lumped parameter networks combining a flow network and a thermal network. It has the disadvantage of being long to built and to calibrate but have the advantages of being fast to solve and easily modifiable.
- **Numerical methods**: Computational Fluid Dynamics (CFD) is the method mostly known for ventilation numerical methods. CFD can provide really accurate results and its a great tool for providing visual results that allow to the user the possibility to understand the behaviour of the fluid flow. On the other hand, with complex models CFD are very time and computational resources consuming.

The present work explains and develops a model of a hydro-generator based on lumped element network modelling. The main reason for using lumped parameter thermal models is the complexity of the geometry that makes CFD much less complex to modify and simulate than a network model. The lumped parameter thermal models are the tools used to get fast and good enough results that allow to designers to test modifications while designing. But in order to perform big validations it is mandatory to run CFD studies and even prototypes validations.

1.2 Parts of a generator

This section introduces the main parts of the generator (rotor, stator and cooling system) and some of the main geometrical parameters. Figure 1.3 is a sketch of a hy-

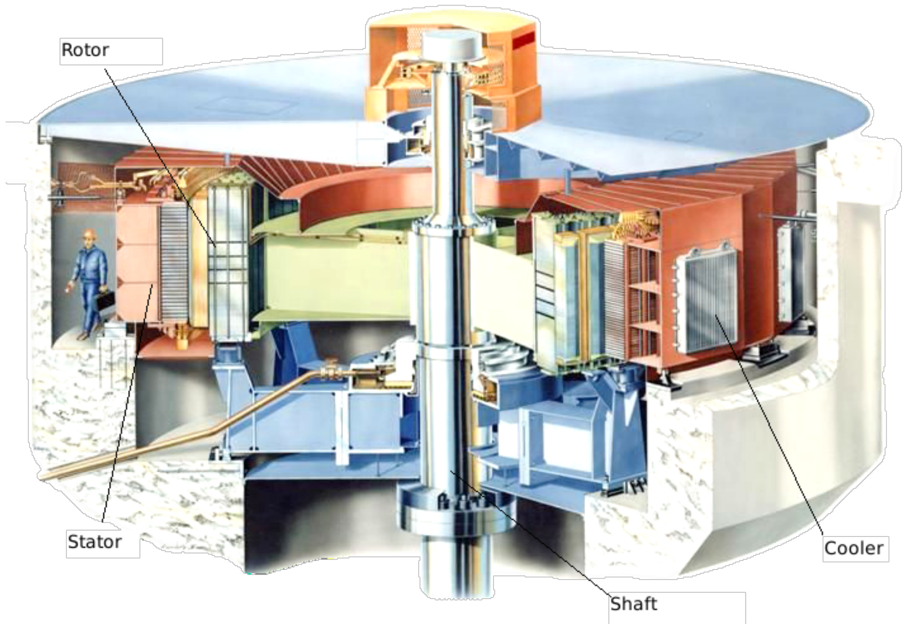

Figure 1.1: Sketch of a hydro-generator. Source:[12]

drogenerator, the size of the main parts can be compared with the man represented. In the centre of the figure is located the shaft that would be directly connected to the turbine under the sketch (not represented in the picture). Supporting the shaft, in yellow, are the bearings. In colour green is represented the rotor, directly attached to the shaft. In red is drawn the stator parts that are supported by the frame, in blue. Going further radially from the stator are located the coolers.

1.2.1 Rotor

As its name suggests, the rotor is the part of the generator that rotates thanks to the power transferred from the turbine by the shaft (see Figure 1.1). With its rotation a rotating magnetic field is created, responsible of inducing a current in the stator windings (Faraday's Law). This magnetic field can be produced by a current passing inside the exciting windings or by permanent magnets. In the studied case the rotor is conformed by exciting windings in a salient pole configuration, which is one of the most typical configurations in hydro-generators.

1.2.2 Stator

The stator is the stationary part of an electrical machine. Figure 1.2 represents an axial cut of an electrical generator. The left side is useful to visualise the relative positions of the rotor, stator and airgap (in blue on both sides of the Figure). The right side of the Figure 1.2 shows the elements of the stator: the windings (or coil), in yellow, the core, in grey, the frame that supports the structure and the wedge, in violet. The coil is the component where the current is induced thanks to the rotating magnetic field. It is put up between two teeth, the yoke and the wedge, this physical space is called slot. The modelled generator has 288 slots and two coils

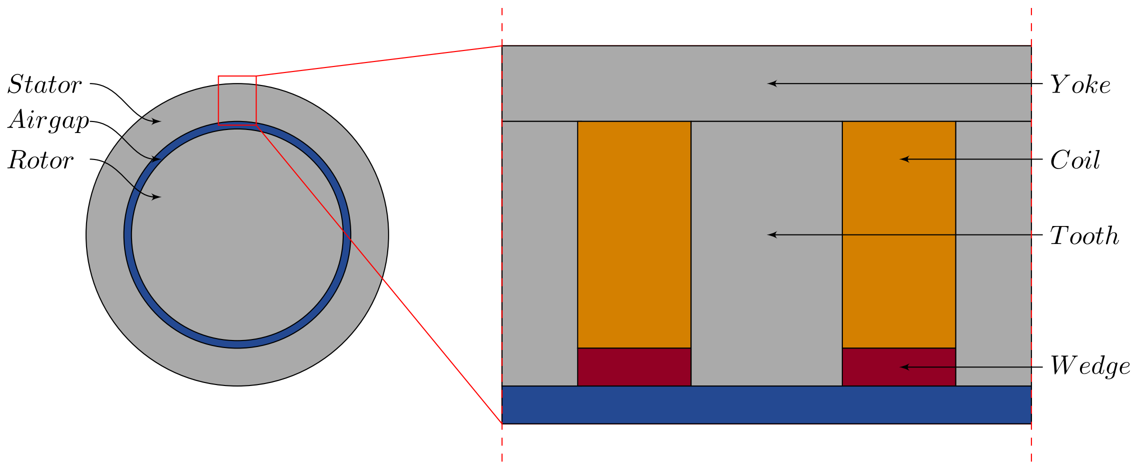

Figure 1.2: Representation of an axial section of a electric generator, on the right emphasis on the stator components

per slot. The core which includes tooth and yoke is made by stacked steel plates reducing the core losses. Those plates are interrupted by cooling channels which are created by separators placed between two plates (see Figure 1.3). The generator modelled has 37 axial parallel channels for each slot. The double coil configuration is separated by a separating wedge. Each coil has 2 columns and 29 rows of parallel rectangular wires made of copper wrapped in a thin layer of electrical insulation. It is in the copper that the electric losses and the corresponding heat generation occur, and from where the heat must be transferred. The coil is finally surrounded by a tape made of mica (as electric insulation too) and impregnated in a thermal conductor called round-packing that is in charge of filling the irregularities in the contact between the coil and the core. Finally the wedge is the structural element that keeps the windings inside the slot.

1.2.3 Cooling system

The cooling system is in charge of keeping the generator at a desired temperature. Figure 1.3 shows the ventilation path of a similar generator than the modelled one. The figure only represents half of the generator because it is a symmetric generator. It is possible to notice the typical components of the ventilation system: a recirculating system, a cooler and the ventilation channels. In this case, the re-circulation is made by two axial fans formed by blades directly mounted on the shaft of the generator. The fans introduce the air axially on the rotor and air-gap. The cooler cools the air exchanging the heat with the secondary cooling system. The modelled generator uses 8 coolers with a capacity of 5 m^3/s. The ventilation channels or stator ducts are the channels where the heat of the stator is transferred to the air. Starting from the fans the path that follows the air is the following: the airflow is introduced axially in to the generator by both axial sides of the machine, a part of this airflow is deviated to cool the end-windings (22% for our case).Then the air that follows the air gap cools the rotor and is introduced to the stator ducts changing the air direction from axial to radial. Because of that change of direction and the

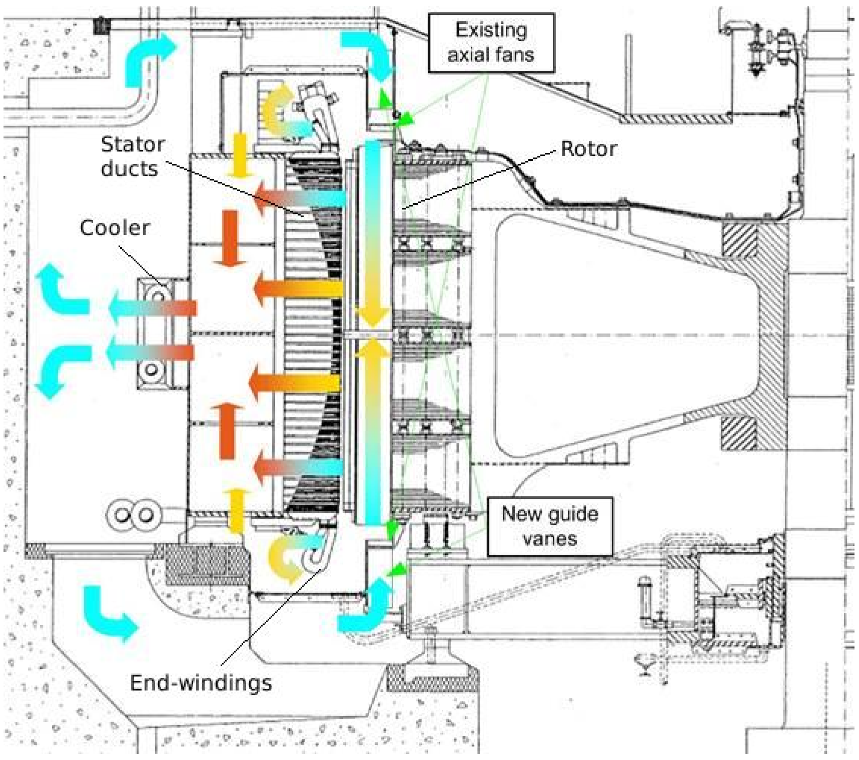

Figure 1.3: Cross-section of a hydro-generator with the ventilation flow represented. Source: Voith Hydro AB

tangential velocity produced by the rotation of the rotor, the study of the airflow in the generator is really complex. Finally all the air is collected behind the stator and introduced into the cooler that extracts the heat. After the heat is evacuated from the machine, the air restarts the whole process. It is a closed ventilation system.

1.3 The network model

The method chosen to model the generator is the lumped element model. A network built with the lumped element model allows to simplify systems with difficult geometries to a network of discrete elements. As the network will be an approximation of the reality the behaviour will be also approximated. The elements from a lumped element method are easy to parameterize and the models can be easily adapted to problems with similar conditions. In order to model the ventilation and thermal behaviour of the generator two network models have to be combined:

- **Thermal network model**: Based on a lumped element model, where the resistance is the thermal resistance between two nodes. Allows to calculate the temperature on the nodes and the heat transfer between them.
- **Fluid network model**: Based on a lumped element model, where the resistance is built with discrete flow resistances and pipe friction. Allows to determine the volume flow in each possible flow path.

An advantage of using the lumped element model is the similarity with the electric model. Table 1.1 shows the equivalence of those two models with the well known electrical model.

	Lumped element model		
Electrical model	Voltage	Current	Resistance
Thermal model	Temperature	Heat flow	Thermal resistance
Fluid model	Pressure	Airflow rate	Flow resistance

Table 1.1: Lumped element model equivalence

1.4 Scope of the thesis

In the present work, the goal is to model a hydro generator using the two network models described before (based on lumped elements). One model is used for the thermal behaviour and a second one for the fluid behaviour. The thermal network model uses basic concepts of heat transfer as the heat transfer modes or the concept of thermal equilibrium. The fluid network model uses concept fluid dynamics as conservation of mass-flow or pressure drop on pipes. With both models it is possible to study the ventilation and cooling of hydro-generators from a thermal and fluid point of view, contributing to future ventilation studies. One of the requirements of this work is to develop a parametric model easy to modify. To achieve this requirement, the model is developed as a software tool, implemented in Matlab/Simulink. The part of the generator modelled is a 3D sector of the stator around one of the slots that accommodate the coils. The generator modelled is a real generator designed by Voith Hydro AB, and in order to build the model real data from geometry and losses is used. The model is validated using data from a heat run test provided by the industrial partner. A heat run is an experimental method used to test generators applying real work conditions and acquiring all the thermal data possible. It is usually run in a prototype generator that has some few modifications to place sensors in components where it is impossible in a real generator. Finally, in order to test the behaviour of the model, a parametric test is performed using the model. The parametric test consists in variate the outlet of the ventilation channels and determinate how it affects to the temperature of the modelled components.

2

Thermal network model

This chapter starts with a short description of the basics of heat transfer, defining the heat transfer modes and the concept of heat balance. Then it is described how to adapt this concepts it to a network model that uses the lumped element method.

2.1 Basics of heat transfer

Heat is defined as the thermal energy that is transferred when there is a temperature gradient. In the atomic scale the heat transfer is the transference of energy in form of speed (or vibration) between particles. That transfer of speed varies the internal energy of the molecules. The temperature is the macroscopic measure of the internal energy. The heat transfer process ends when the difference of temperatures is zero. This state is called thermal equilibrium. Heat transfer can occur in three different modes:

- **Conduction:** Occurs in all mediums where there is a temperature gradient and direct contact with the particles. The heat transfers from the more energetic particles of the same medium to the less. The effects of conduction are usually more significant in solids than in other mediums. The capability to conduct heat of a material is called thermal conductivity. In the same medium, the unidirectional conduction equation (Fourier law) is written

$$q = -k\frac{dT}{dx} \ , \tag{2.1}$$

  where
  - q: Heat transferred $[W/m^2]$
  - k: Conductivity of the medium, $[W/mK]$
  - $T(x)$: Temperature distribution at the medium $[K]$
  - x: Direction in the space $[m]$

- **Convection:** Occurs between a solid and a fluid at a different temperature. The fluid has to move with a relative speed with respect to the solid. If the relative motion is produced by buoyancy effects its called natural convection and if it is produced by mechanical forcing it is called forced convection. Convection is described by

$$q = h(T_{solid} - T_{fluid}) \ , \tag{2.2}$$

where:
- q: Heat transferred between soild and fluid $[W/m^2]$
- h: Heat transfer coefficient $[W/m^2K]$
- T: Temperatures of solid contact surface and the fluid $[K]$

- **Thermal radiation**: Emission of energy through electromagnetic waves occurs at every surface at a temperature grater than 0 K. As in the other cases, heat exchange is produced between surfaces at different temperatures

$$q_{i-j} = \epsilon\sigma(T_i^4 - T_j^4) \ , \tag{2.3}$$

where:
- q_{i-j}: Heat transferred between surfaces i and j $[W/m^2]$
- ϵ: Emissivity of the surface
- σ: Stephan-Bolzmann constant $[W/m^2K^4]$
- T: Temperatures of the surfaces i and j $[K]$

In the present tool the radiation has not been considered in order to decrease the complexity of the problem and due to its low impact in the global heat transfer.

2.2 General conduction equation

The 2^{nd} Fourier's law or general conduction equation describes the heat balance in time in a medium. Heat balance is then the cause of the temperature variation in the medium

$$\nabla^2 T + \frac{\dot{g}}{k} = \frac{1}{\alpha}\frac{\partial T}{\partial t} \ , \tag{2.4}$$

where:
- T: Temperature distribution at the medium $[K]$
- $\dot{g}$: Heat generation $[W/m^3]$
- k: Thermal conductivity at the medium $[W/mK]$
- $\alpha = \frac{k}{\rho \ Cp}$: Diffusivity of the medium $[K]$
- t: Time $[s]$

2.3 Nodal implementation

As it was said before, one of the challenges is to develop a network model based on the lumped element model as parametric as possible in order to be able to perform changes on the geometry. For that reason, each node of the thermal network model is conceived as a replicable set of blocks that applies the theory of heat transfer.

2.3.1 Thermal resistances

The network model is made by discretizing the geometry into nodes with defined Δx, Δy and Δz. Those nodes (or elements) have assigned the physical properties of density, conduction and specific heat that correspond to its material. Figure 2.1

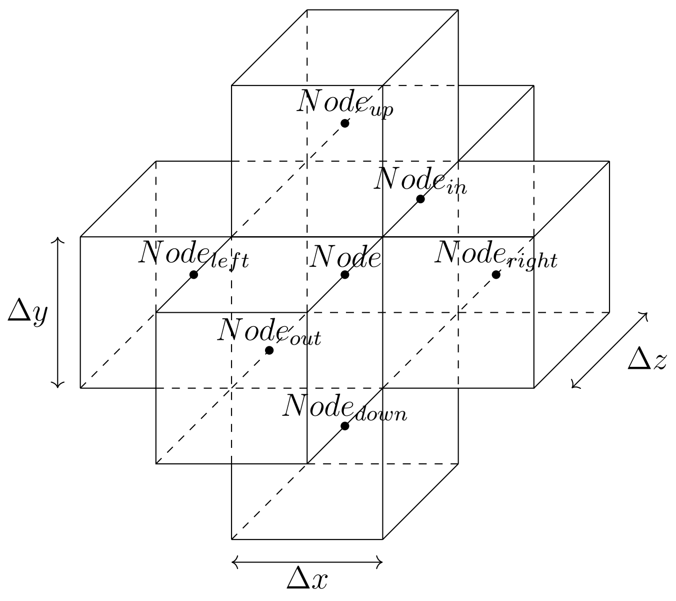

Figure 2.1: Node visualisation 3D with nodes names as they are used in the code

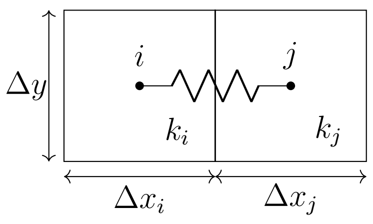

Figure 2.2: Conduction in x-direction representation in 2D

is an example of a 3D network presented with the names that are assigned to the neighbour nodes as they are in the code. For a 3D model each node will have 6 neighbours, two for each direction.

Any of the modes of heat transfer can occur between a node and its neighbours. For that reason, in order to standardize all the relations between nodes, the heat transfer modes are implemented in the thermal model as resistances. The heat transfer between any node is

$$q_{ij} = \frac{T_i - T_j}{R_{ij}} \ , \tag{2.5}$$

where R_{ij} is the nodal resistance between nodes i and j. For conduction and convection for a 3D nodal heat transfer (see Figure 2.2 and Figure 2.3, (z direction would be orthogonal to x and y)), resistance is calculated. For conduction, the resistance between two neighbour nodes in the x direction (as in Figure 2.2) is given by

$$R_{ij} = \frac{\Delta x_i}{2 \cdot k_i \cdot \Delta y \cdot \Delta z} + \frac{\Delta x_j}{2 \cdot k_j \cdot \Delta y \cdot \Delta z} \tag{2.6}$$

where k_i and k_j are the thermal conductivity of each node (for that reason is always preferable to make nodes that are all of the same material). $\Delta y \cdot \Delta z$ is the contact

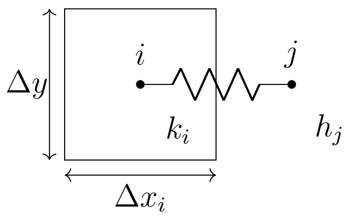

Figure 2.3: Node convection in x direction representation in 2D

surface between the two nodes. For the convection mode the resistance between two nodes in the x direction (represented in Figure 2.3) is

$$R_{ij} = \frac{\Delta x_i}{2 \cdot k_i \cdot \Delta y_i \cdot \Delta z_i} + \frac{1}{h_j \cdot \Delta y_j \cdot \Delta z_j} \tag{2.7}$$

where h_j is the heat transfer coeficient of the fluid of the node j. $\Delta y \cdot \Delta z$ is again the contact surface between the two nodes. To consider other directions the distance between nodes and the contact surface has to be modified to the appropriate.

2.3.2 Heat balance

It is possible to develop equation 2.4 for a differential volume to be applied to the nodal network defining it as

$$\int_0^t dT = \int_0^t \frac{Heat\ Balance + Heat\ Generated}{\rho\ Cp\ dV} \tag{2.8}$$

or for a short time interval

$$\Delta T = \frac{\sum_1^6 q_i + g_i \cdot \Delta V}{\rho\ Cp\ \Delta V} \tag{2.9}$$

or using Laplace transformation

$$T(s) = \frac{\sum_1^6 q_i + g_i \cdot \Delta V}{\rho\ Cp\ \Delta V \cdot s} \tag{2.10}$$

where 1/s is the integrator of Laplace that can easily be implemented in Simulink. Figure 2.4 represents the implementation of one 3D node with its characteristics sets of blocks (in Matlab/Simulink). The common Simulink elements are: the *'gain'* block represented by a triangle, allows to multiply the the signal. The *'tag'* block allows to connect blocks without a line, which would make the model more confuse. The *'sum'* block represented by a circle sums or subtract two or more signals. Finally the Laplace integrator which is named with a *'1/s'*. Two main regions can be noticed in the figure:

- The heat transfer set of blocks, located on the left side, that corresponds to the direct implementation of equation 2.5, where the two *'From'* tags are the nodes temperature (the light blue one is the analysed node), the gain block is 1/R (where R is the thermal resistance between the two nodes extracted from matrix Mr). Finally, the *'Goto'* tag coloured in orange is the heat transferred between the two nodes.

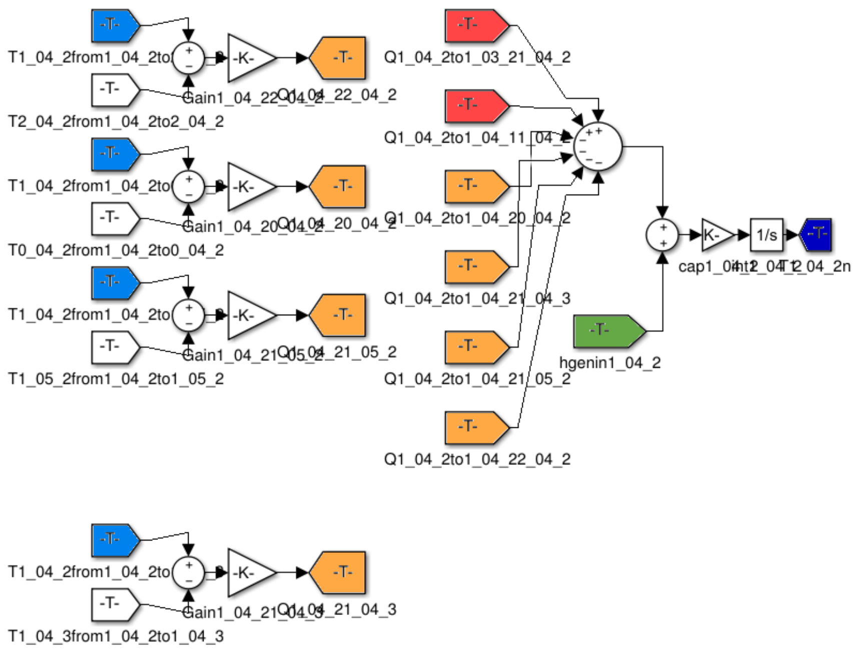

Figure 2.4: Simulink model of one 3D node, in the left side are the heat transfer blocks and in the right side the heat equilibrium ones

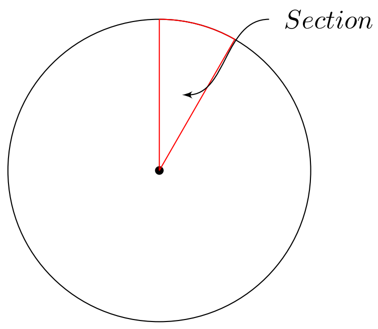

Figure 2.5: Generator periodicity

- The heat balance set of blocks, corresponds to the right side of the picture, is the implementation of equation 2.10: We can observe 6 *'From'* tags that are linked to heat transfer calculus where this node appears. In that case the orange tags come from the *'Goto'* tags that appear in the heat transfer set of this node. The other two heat transfer are located in other nodes because they have been calculated before in the code. It is important to notice that we can only include a heat transfer between two nodes only once in the model. The circle with the signs are sums or substractions depending on the provenance of the heat transfer (it has to agree with the sign criterion established with the equation 2.5). After this operation, the result is added to the generation assigned to this node (green block), the gain that corresponds to the $1/\rho \cdot Cp$ and the laplace integrator are applied. The result (blue tag) is the temperature of the node that Simulink will use for the next step of the simulation.

2.3.3 Heat generation

The heat generation is included as the losses. Which is one of the inputs of the model. As they are divided in iron losses, coil losses, rotor losses and ventilation losses, they are modelled in different ways. The ventilation losses and rotor losses increase the inlet air temperature of the model. The coil and iron losses are included in the nodes in a volume proportion, as

$$Losses(node) = \frac{\Delta V_{node}}{Total\ Volume} \cdot Total\ Losses \tag{2.11}$$

2.3.4 Boundary conditions

The boundary conditions of the thermal model are determined by two different types: air boundary conditions for which the temperature of the node is determined (see chapter 3) and periodic conditions, here described:

The generator, being a rotating machine with circumferential periodicity, can be treated as a portion of a sector that is repeated (Figure 2.5). The boundaries of this sector will be considered as periodic planes.

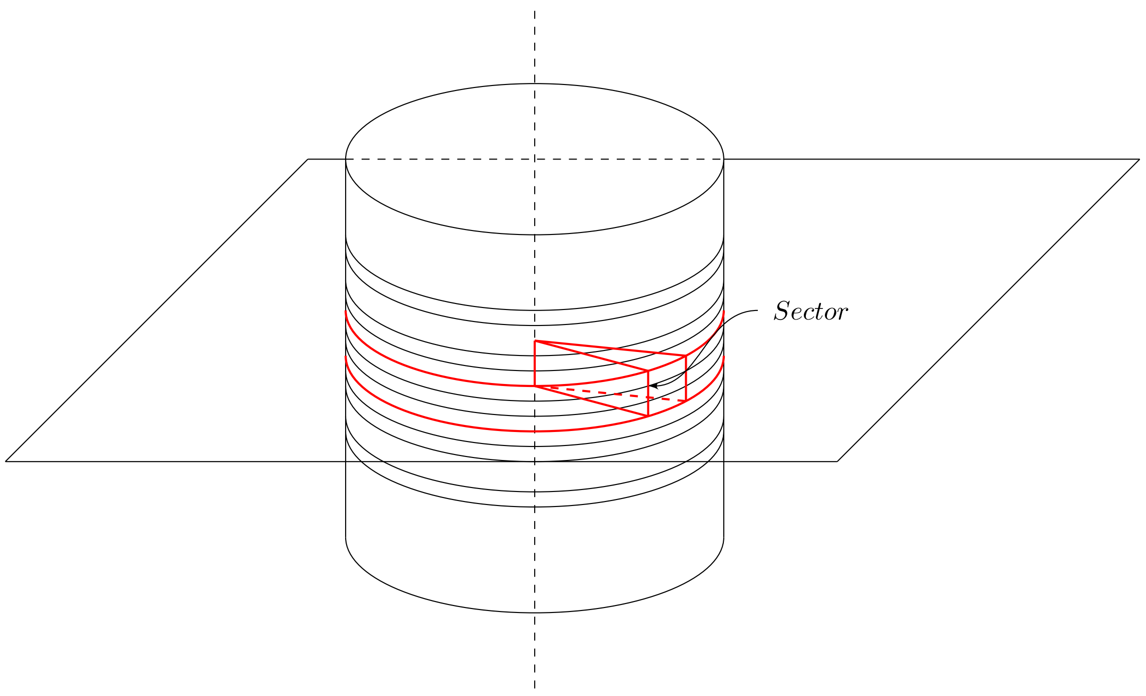

Figure 2.6: Representation of a generator with several parallel ventilation channels and a possible sector to model (in red).

The generator has 37 parallel ventilation rows and the study will be focused in only one of them, so the analysed sector can be represented as in figure 2.6. This sector will be the part of the generator modelled.

Figure 2.7 and 2.8 represent in a non-scaled discretization the studied sector. The colour legend for both figures is: yellow for the coils, grey for the core, violet for the wedges and blue for the air. Each horizontal line in Figure 2.8 represents the axial plane that cuts the layers of Figure 2.7. In Figure 2.7, the representations are the 3 axial layers that have to be at least used to perform the model: two (left and right representations) for the coil, tooth and wedge interaction. The other one (middle representation) is the interaction between coils and the air channels. Figure 2.8 represents the modelled sector as it is seen looking from the air gap through the air channels. The core bars that cross the ventilation channels are the separators and in the thermal network model are modelled as fins. In this representation it is easy to perceive the wedge that supports the coils is interrupted in the ventilation ducts to allow the air into.

Figure 2.9 is a representation of one of the separators with its neighbour channels. Because of the geometry, the heat flow (represented with red arrows) is horizontally symmetric. In heat transfer, the symmetry can be interpreted like an adiabatic surface boundary condition. This fact can be understood in the following way: if there is no temperature difference the heat exchanged is zero. That allows to model the separators as adiabatic fins with half of the length of the separators. The equations to model fins with an adiabatic end is given by

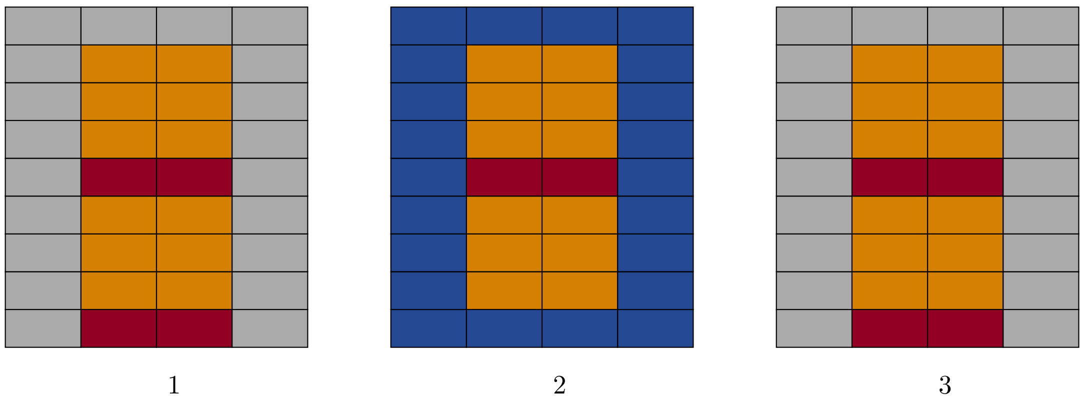

Figure 2.7: Representation of 3 layers of 4x9 nodes. The layers are obtained doing an axial cut of the stator. In Figure 2.8 are represented the cuts

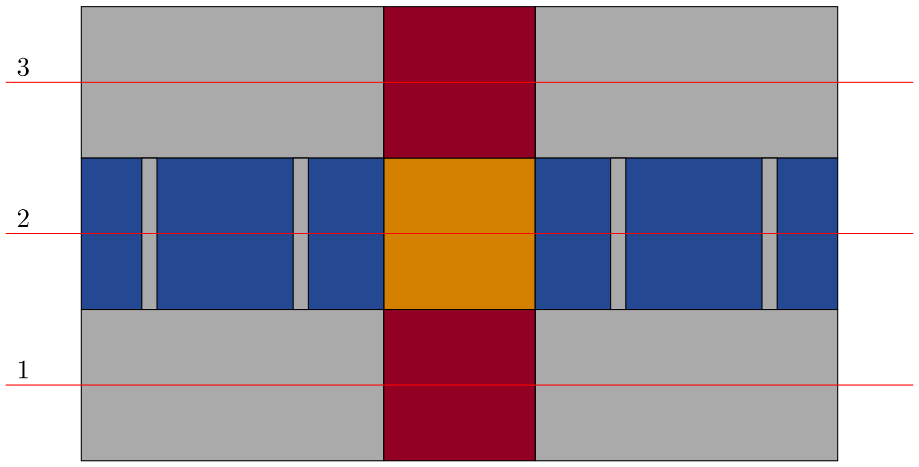

Figure 2.8: Representation of the generator as it is seen from the air gap, blue represents air, grey the iron, yellow the coil and violet for the wedges. Each axial cut is represented in Figure 2.7

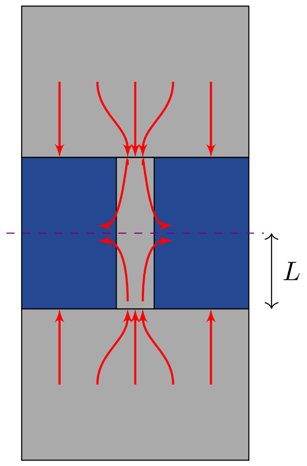

Figure 2.9: Representation of the heat flow in the separators, dashed line represents the symmetry of the model

$$q_f = \frac{\Delta T}{h(A_0 + e_f \cdot A_f)} \ , \tag{2.12}$$

where:

- ΔT: Temperature gradient between solid and fluid $[K]$
- h: Heat transfer coefficient $[W/m^2 \cdot K]$
- A_o: Solid surface excluding fin contact area $[m^2]$
- e_f: Efficiency of the fin
- A_f: Extended fin surface (in contact with the fluid) $[m^2]$

The efficiency of an adiabatic fin is calculated with the formula (see [2] or [5])

$$e_f = \frac{tanh(m \cdot L)}{m \cdot L} \ , \tag{2.13}$$

where

$$m = \sqrt{\frac{h \cdot P}{\lambda \cdot A}} \ , \tag{2.14}$$

where:

- L: Length of the fin (see Figure 2.9) $[m]$
- h: Heat transfer coefficient $[W/m^2 \cdot K]$
- A: Cross section of the fin $[m^2]$
- Λ: Thermal conductivity of the fin $[W/m \cdot K]$
- P: Cross perimeter of the fin $[m]$

3 Ventilation model

This chapter introduces firstly the basics of fluid dynamics required to implement the fluid network model. Then the correlations used to calculate the heat transfer coefficient used to model the interaction fluid solid. Finally the air boundary conditions are explained. In all the chapters is shown how it is implemented in the ventilation network model.

3.1 Fluid dynamics basics

The fluid used for cooling system of the modeled generator is air. It is introduced axially into the generator cooling firstly the rotor. Then it changes its direction from axial to radial in order to enter into the different stator channels. Figure 3.2 shows the representation of the fluid network model used for the modelled generator. Figure 3.1 represents an axial cut of the stator channels, in yellow is represented the coil and the limits of the core are drawn in grey. Each set of stator channels is composed by 5 channels that cool a single coil section. After the slot two of the channels merge into one only channel.

A fluid network model is used to deduce the volume flow in each ventilation channel. Every bend, section change and friction losse has to be modeled as a flow resistance where a pressure drop is produced. As the channels are connected at the inlet and at the outlet, the total pressure drop for every path has to be the same

$$\Delta P_n = \Delta P \ , \qquad \forall n \ , \tag{3.1}$$

and the total volume flow has to be conserved

$$\dot{V_{inlet}} = \sum_1^n V_i = \dot{V_{outlet}} \ . \tag{3.2}$$

The total pressure drop will be the sum of all the single flow resistances:

$$\Delta P_n = \sum_1^r \Delta P_{ri} \tag{3.3}$$

and the pressure drop for each flow resistance is modelled like

$$\Delta P_i = \frac{1}{2}\zeta_i v_i^2 \rho \tag{3.4}$$

where:

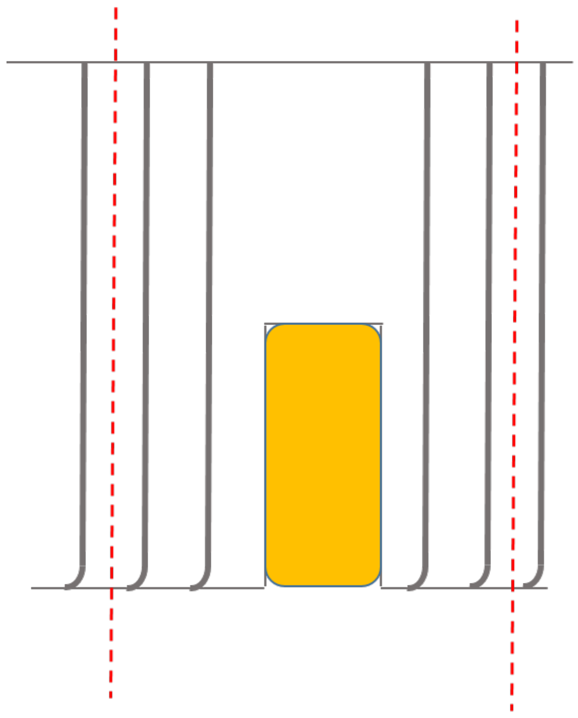

Figure 3.1: Representation of an axial cut of the ventilation channels.

- ΔP: Pressure drop at item i $[Pa]$
- ζ: Fluid resistance coefficient
- v: Air speed at item i $[m^2/s]$
- ρ : Air density $[kg/m^3]$

The fluid resistance coefficient has to be calculated using geometrical correlations for discrete (or singular) resistances (like section changes, bends or manifolds) or calculating the linear resistance due to friction in the channels. For the modeled case, three main types are used following the procedure described in [3]:

- Inlet (i): It is considered a reduction of the airflow cross section used for the entrance of all the ducts. The fluid resistance coefficient used as it is described in [3] is

$$k_{inlet} = 0.5 \cdot \left(1 - \frac{S_{in}}{S_{out}}\right)^{\frac{3}{4}} . \tag{3.5}$$

- Friction (f): It is a linear resistance (depends on the length of the duct) characterised by the friction between the air and the walls of the duct. As its calculation depends on the airspeed the network has to be solved in a iterative way. The fluid resistance coefficient from [3] used is

$$k_{friction} = \frac{\lambda \cdot l}{D_h} , \tag{3.6}$$

where λ is the friction coefficient calculated as

$$\lambda = \lambda_{lam} \cdot p_{lam} + \lambda_{sm} \cdot p_{sm} + \lambda_r \cdot p_r , \tag{3.7}$$

that is a probability calculation of laminar, smooth tubes and rough tube conditions, that depends on the airspeed, multiplied by the friction of each flow mode,

$$\lambda_{lam} = \frac{64}{Re} \ , \tag{3.8}$$

$$\lambda_{sm} = \frac{0.3164}{Re^{\frac{1}{4}}} \ , \tag{3.9}$$

$$\lambda_{r} = \frac{1}{\left(2 \cdot log\left(\frac{3.7}{\overline{\Delta}}\right)\right)^2} \ , \tag{3.10}$$

where $\overline{\Delta} = \frac{\Delta}{D_h}$ is the relative roughness of the duct (D_h is the hydraulic diameter, see formula 3.20). The probabilities are dependent on the Reynolds number,

$$p_{rt} = erf(\frac{Re - 2850}{\sqrt{2} \cdot 600} \ , \tag{3.11}$$

$$p_t = \frac{1}{2} + \frac{1}{2} \cdot p_{rt} \ , \tag{3.12}$$

$$p_{lam} = \frac{1}{2} - p_t \ , \tag{3.13}$$

$$p_{sm} = (1 - p_{rt}) \cdot p_t \ , \tag{3.14}$$

$$p_r = p_{rt} \cdot p_t \ , \tag{3.15}$$

- Outlet (o): Defined as an increase of the airflow cross section, it is used for the exit of all the ducts and the merging of tubes. The fluid resistance coefficient used is [3]

$$k_{outlet} = \left(1 - \frac{S_{in}}{S_{out}}\right)^{\frac{1}{2}} \ . \tag{3.16}$$

The network modeled in the netflow.m file (see A.4) is composed by 5 parallel entrances of different cross section. The two inlets at the sides of the coils merge into only one after it. A schematic representation of the system is the one represented in figure 3.2, where the correlations used in each resistance are marked with i, f or o. All the 4 ducts end in the outlet with different cross section than in the inlet.
For solving the network the netflowc.m function iterates the volume flow of each channel until the difference of pressure drop between all channels is in an acceptable range (modifying the *'erra'* value. It is possible to change the precision of the results and sometimes it is necessary to converge). The iteration is necessary because the friction coefficient depends on the volume flow.

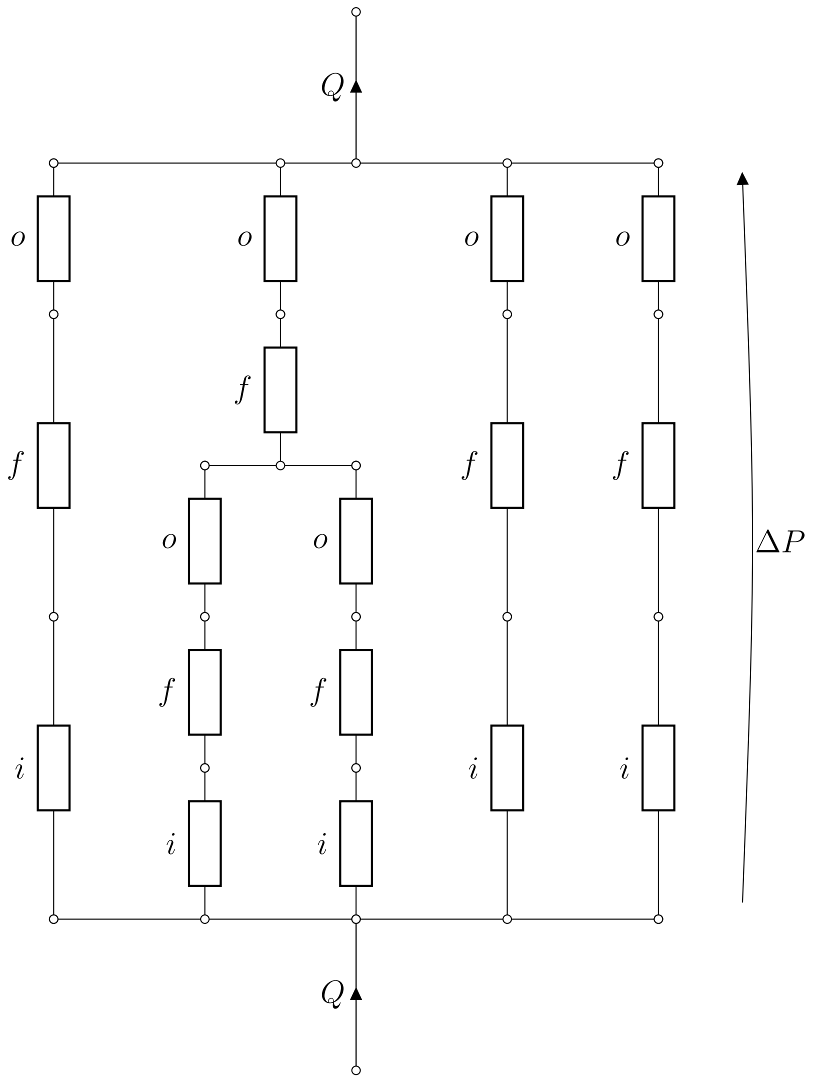

Figure 3.2: Representation of the fluid network model of the analysed generator, Q represents the airflow, ΔP the pressure drop for all the channels and the resistances are named with their type of resistance

3.2 Heat transfer coefficient calculation

As defined in chapter 2 the heat transfer coefficient is responsible for defining how well the heat is transferred between a solid and a fluid. In order to transfer this heat, the fluid needs a relative velocity with respect the solid and the heat transfer coefficient is very dependent of this relative velocity. While natural convection is the phenomenon that occurs when the relative motion of the fluid is produced by buoyancy forces due to differences of temperature in the fluid, the forced convection is the one that occurs when the movement of the fluid is 'artificially' induced (i.e: produced by fans or pumps or even the relative speed of a car in a motorway).

The following lines describe the correlation used to calculate the heat transfer coefficient in the model. It is important to notice than this value is referred to the mean air temperature.

The Nusselt number is defined as

$$Nu = \frac{h \cdot D_h}{k} \ , \tag{3.17}$$

where:
- Nu: Nusselt number
- D_h: Hydraulic diameter [m]
- h: Heat transfer coefficient $[W/m^2 \cdot K]$
- k : Thermal conductivity $[W/m \cdot K]$

The goal is to use a correlation to calculate the Nusselt number and finally deduce the heat transfer coefficient. A widely used correlation for large Reynolds numbers is the Gnielinski correlation [11]:

$$Nu = \frac{\frac{f}{8} \cdot (Re - 1000) \cdot Pr}{1 + 12.7 \cdot (\frac{f}{8})^{\frac{1}{2}} \cdot (Pr^{\frac{2}{3}} - 1)} \ , \tag{3.18}$$

where:
- Nu: Nusselt number
- f: Friction factor
- Re: Reynolds number
- Pr : Prandl number

The Prandl number is calculated as

$$Pr = \frac{Cp \cdot \mu}{k} \ , \tag{3.19}$$

where:
- Pr : Prandl number
- Cp: Nusselt number
- μ: Dynamic viscosity
- k: Thermal conductivity

As the ducts modelled are not cylindrical it is mandatory to use the hydraulic diameter formula to apply the correlations

$$Dh = \frac{4 \cdot S}{P} \ , \tag{3.20}$$

where:

- Dh : Hydraulic diameter
- S: Hydraulic section
- P: Wet perimeter

and the Reynolds number formula is

$$Re = \frac{v \cdot Dh}{\nu} \ , \tag{3.21}$$

where:

- Re : Reynolds number
- v: Average air speed
- Dh: Hydraulic diameter
- ν: Kinematic viscosity

3.3 Temperature increase in the channel

As the air nodes under study are successions of the same ventilation channel, the temperature of an air node is calculated considering the power exchanged in the previous nodes,

$$T_{i+1} = \frac{\sum q_i}{\rho \cdot Cp \cdot Q} + T_i \tag{3.22}$$

where:

- T : is the temperature of the node
- q_i: is the heat transfered to the node i
- ρ: is the density of the fluid
- Cp: is the specific heat capacity
- Q: The flow in the considered channel

Figure 3.3 shows how the formula is implemented in the model.The red *'From'* blocks are the heat exchanges that affect this air node. The orange *'Gain'* is multiplied by *cp rho*, the blue *'From'* block is the temperature of the air node and the green *'Goto'* tag is the temperature of the next node.

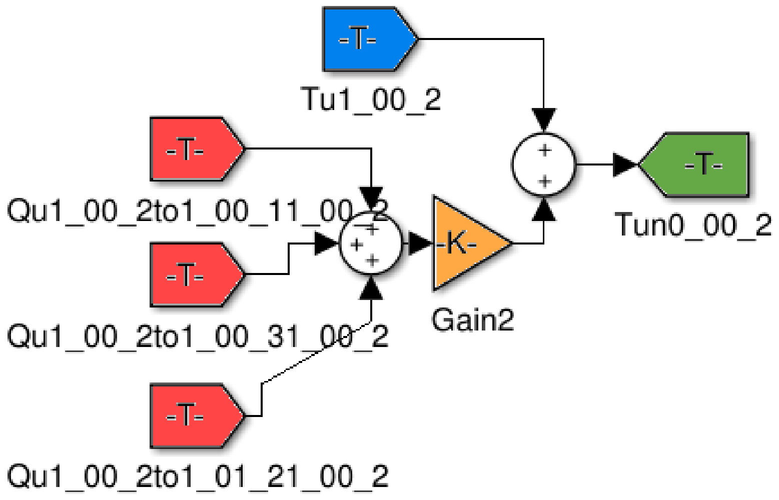

Figure 3.3: Temperature calculation for the air nodes

4
Model overview

The tool is developed using MATLAB and Simulink, so it is necessary a basic knowledge of it to fully understand this report and the code that can be found in the Appendix. Sufficient documentation and examples are explained in [4], the Matlab web-page.

The model is focused on obtaining the temperatures in the stator coil, where the hottest point of the generator is located. For that reason the modelled part in the thermal and fluid networks is only a 3D portion of the stator (see section 2.3.4) that corresponds to the ventilation channel of a slot. Figure 4.1 represents the behaviour of the different parts of the model as well as the dependence among them: the outputs of the fluid network model are inputs for the thermal network model. The fluid network calculates the airflow in each stator duct of the model. This airflow is used to calculate the heat transfer coefficient in each channel and both parameters are used by the generator thermal network. The losses of the generator are used to generate heat in different nodes of the generator (core and coils nodes) and as a preheating of the air.

Other inputs that have to be considered to run a simulation are:

- **Simulation time**: The simulation time can be defined in the executemod.m file or directly in the Simulink interface. It is the time in seconds that the model will take to simulate. The processing time of the model is directly

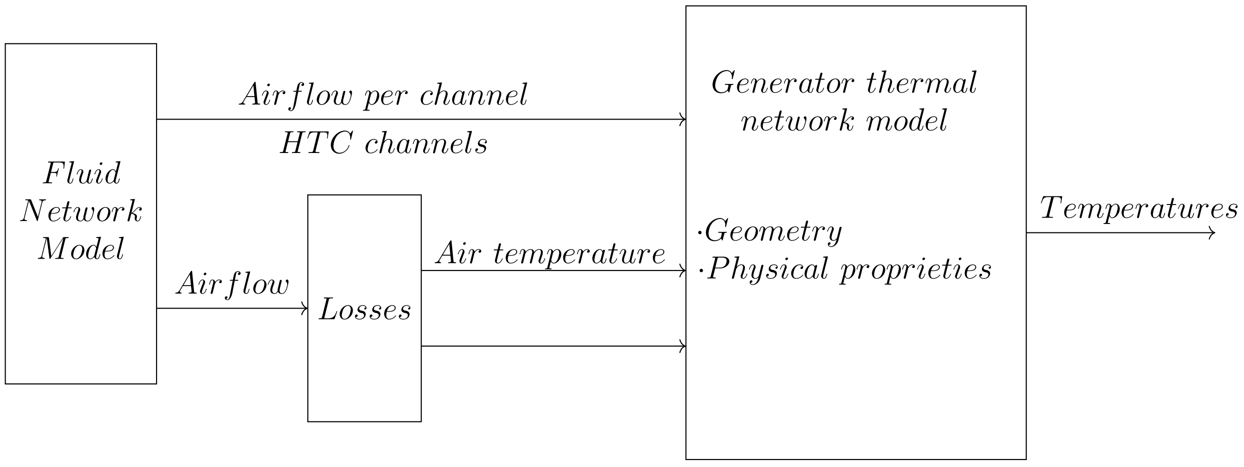

Figure 4.1: Model scheme representation

proportional to the simulation time but usually never exceeds the order of minutes. A typical value for a run to reach steady state conditions could be around 10000 seconds with a computation time of more or less 30 seconds.

- **Time step**: Time step is the simulation time between two consecutive calculations in Simulink. Usually the automatic step time mode of Simulink can be good enough for the simulation. It can be modified in the Simulink configuration interface. As it happens with the simulation time, when modifying the step time the computation time is affected, increasing when decreasing the step time.
- **Initial temperatures**: The initial temperatures of the components have to be determined for the integrator block (see figure 2.3) and are defined in the script (in the model they are defined at 20°C).
- **Mesh size**: The mesh/element size is one of the main parameters to decide before building a network model. This parameter can be modified in the netmod.m file. The *co* parameter is the number of nodes horizontally that will compose the coil, one more node per side will be built (for tooth or air channel). The *ro* parameter is the number of nodes for the coil vertically two more will be added too (for the wedge/yoke and air channels). Finally *zo* is the parameter that will determine the number of axial layers, the minimum and recommended value is 3.
- **Initial air temperature**: The temperature that the air has after the coolers. The air temperature which the model finally works takes in to account an increasing of the temperature due to the rotor losses, ventilation losses and friction losses.

As introduction to the code, Figure 4.2 shows the hierarchy of the files that makes the understanding of it easier. The structure of the code is one main script, the netmod.m file (see A.3), that creates a Simulink parametric block model where the thermal network is built. Before creating the thermal network model, the fluid conditions are calculated executing the netflow.m (see A.4) file which runs a function that solves the fluid model returning the airflow in each channel. The fluid model is solved iterating the volume flow in each channel until the total pressure drop in each path are equal. In each iteration the two channels around the coil have to be solved too to get the pressure drop in this path. With the volume flow of each channel, the fhtc.m (A.5) function calculates the heat transfer coefficient of each channel using the correlations described in section 3.2. The netmod.m script also calls the linknodes.m function that is in charge of returning the nodes that are in contact with its input node to build the thermal network. The parameters and data used in order to simulate and calibrate the model are provided by the industrial partner and are based on a real hydro-generator, all this data is stored in the inputgen.m file (see A.1), where all the others scripts/functions call them. The reason of having them all in a only one script is to help the user to have a clearer perception of the parameters he needs to run the simulation. The executemod.m file (see A.2) is a script that calls the netmod.m file creating the model, execute it with a simulation time, plots the main results on screen and saves them in a .csv file, it is useful to run several simulations in a row automatically.

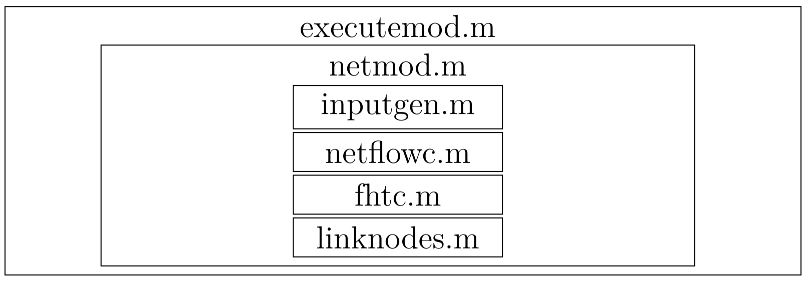

Figure 4.2: Code hierarchy

5
Results

5.1 Validation

In this first subsection, results of a simulation of the model are presented and will be validated by comparing them with the data provided by Voith Hydro AB. The data to be validated is the temperature of windings and core at steady state with the machine at full power.

The simulation is performed with the generator starting at ambient temperature (20°C) and with full load (all the losses from the beginning, modeled like a step block). The total inlet airflow is the maximum allowed by the coolers, but the one used as input in the model is only the part that corresponds to the section modeled, which is the total minus the part that bypasses via the end-windings and divided by the number of slots and the number of channels. The simulation time is set to 10000 seconds in order to reach the steady state. The steady state is defined in heat transfer as the state when there is not change of temperature and heat transfer in time. As it can be observed in Figure 5.2, Figure 5.3 and Figure 5.4, the temperatures and the heat transfer are horizontal, showing that the model is very near to steady state.

In Table 5.1 the results of the flow network and heat transfer coefficient calculation are presented for the channels as they are numbered in Figure 5.1. The airflow and heat transfer coefficients are lower in the channels around the coil as the entrances are smaller and the airflow has to do an extra change of path arround the coil.

Once the thermal model is executed, a plot of temperature versus time is returned by the simulation. As shown in Figure 5.2, the coil temperatures for the model are between around 88°C and 92°C. In Figure 5.3 the average temperatures against time for the elements are displayed and in Figure 5.4 the heat transfer extracted by the air in the different spots. As the coil is the part of the generator with the higher

Channel number	1	2	3	4	5
Volume Flow [m^3/s]	7.76E-4	2.5E-4	3.49E-4	7.76E-4	7.76E-4
Volume Flow rel [%]	26.5	8.6	11.9	26.5	26.5
Heat Transfer coefficient [$W/m^2 \cdot K$]	55.8	35.9	37.8	55.8	55.8
Pressure drop in stator ducts [Pa]	79.2				

Table 5.1: Fluid network and heat transfer coefficient calculations results

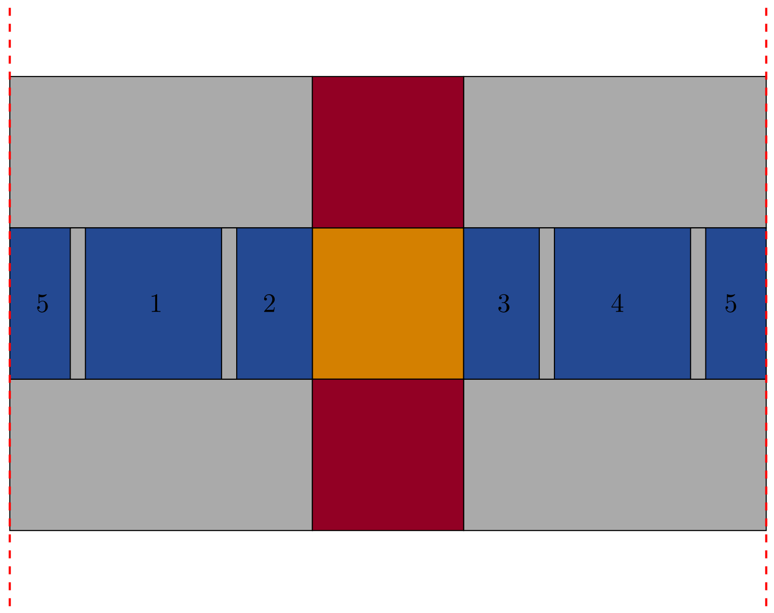

Figure 5.1: Ventilation channels numbered

loss density (for the modeled sector the coil receives 29.9W and the core 20.7W) and it is not one of the best cooled (around 2W from the air ducts behind the coil) it seems normal that the highest temperatures are located in the coil. The core is the element that evacuates more heat (almost all the air-gap and all the heat from core to channel that sum around 40W). From that fact we can conclude that the core is helping to evacuate the heat from the coil thanks to its big surface in contact with the air. On the other side, the wedges that are made with a material with very low thermal conductivity, have temperatures that are similar to their neighbours and evacuate a little amount of heat.

Table 5.2 shows a comparison between model and data provided by the industrial partner[1]. The model shows a good performance against the data from the real generator being the differences minimal (less than 2% of relative error) at the coil and little (less than 8% of relative error) at the core.

The higher error in the core could be due to different causes (or all at the same time): The resistance modelled between the coil and the core could be too high, the resistance between the core and the channels could be higher than in the real generator (too optimistic heat transfer coefficient calculation, contact between core and separators worse than expected...).

The difference between the radial temperature gradient in the core (Figure 5.2), from +0.7K in the model to -0.8K in the industrial data could be explained by how the losses are distributed. The losses in the model are distributed homogeneously in the volume while in a real generator the magnetic losses are higher at those locations where the magnetic field is stronger. As the magnetic field becomes weaker with

[1]The data provided was at different axial distances, here are shown the mean of these values as the boundary conditions of the model homogenise its characteristics axially

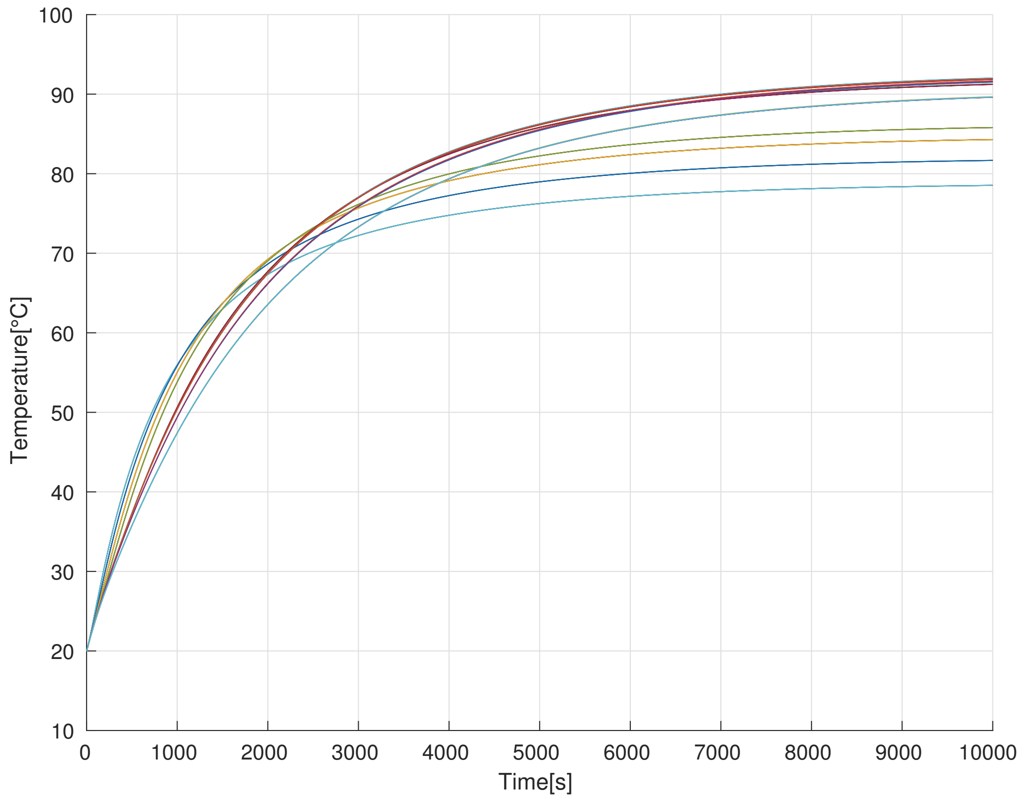

Figure 5.2: Temperature of the 48 coil nodes, where each line corresponds to the thermal behaviour of each node. All the coil nodes temperature are between 88°C and 92°C

	Model	Industrial data	Absolut error	Relative error
Max. coil temp. [$°C$]	91.8	91.9	0.1	0.1%
Mean coil temp. [$°C$]	86.8	88.3	1.5	1.7%
Centre tooth temp. [$°C$]	72	78.2	6.2	7.9%
Centre core temp. [$°C$]	72.7	77.4	4.7	6.1%
Mean core temp. [$°C$]	69.6	-		
Mean wedge sep. temp. [$°C$]	84.1	-		
Mean wedge temp. [$°C$]	57.6	-		

Table 5.2: Temperatures comparison between model and industrial data

	Channel 1	Channel 2+Channel 3	Channel 4	Channel 5
Normal	14	26	14	14
S1	4	56	4	4
S2	6	50	6	6
S3	8	44	8	8
S4	10	38	10	10
S5	12	32	12	12

Table 5.3: Set of width for the outlet of ventilation channels for a parametric study

the distance, it is reasonable to think that the major part of the core losses would really be in the tooth, which explains the gradient obtained for the industrial data. The gradient in the model is produced by the increasing of the temperature of the air in the radial direction (this also occurs in the real generator).

5.2 Effect of changing the outlet width

One useful way to use the network model that has been described so far is by performing parametric studies (i.e, the effect of changing certain parameters on the heat transfer coefficient). In this case, one goal is to compare the temperatures obtained in the different components with a variation in the width of the outlet of the ventilation ducts. That modification varies the flow resistance in the channels influencing the volume flow per channel and the heat transfer coefficient. As it was seen before the coil is actually cooled by the core. A way of increasing the efficiency of the cooling system could be to extract more heat directly from the coil. Table 5.3 presents the different widths of the channels (the total width has to be always the initial: 68 mm). The simulations are made from the current model decreasing the width of the 'tooth channels' (numbers 1, 4 and 5) by 2 mm in each simulation and increasing the width of the 'coil channels' (numbers 2 and 3).

Table 5.4 shows the results of the different simulations for the main parameters. As it was expected, decreasing the width of the 'tooth channels' increases the volume flow in the 'coil channels' and decreases it in the 'tooth channels' (see Figure 5.5) while increasing the total pressure drop (see Table 5.4). This airflow variation

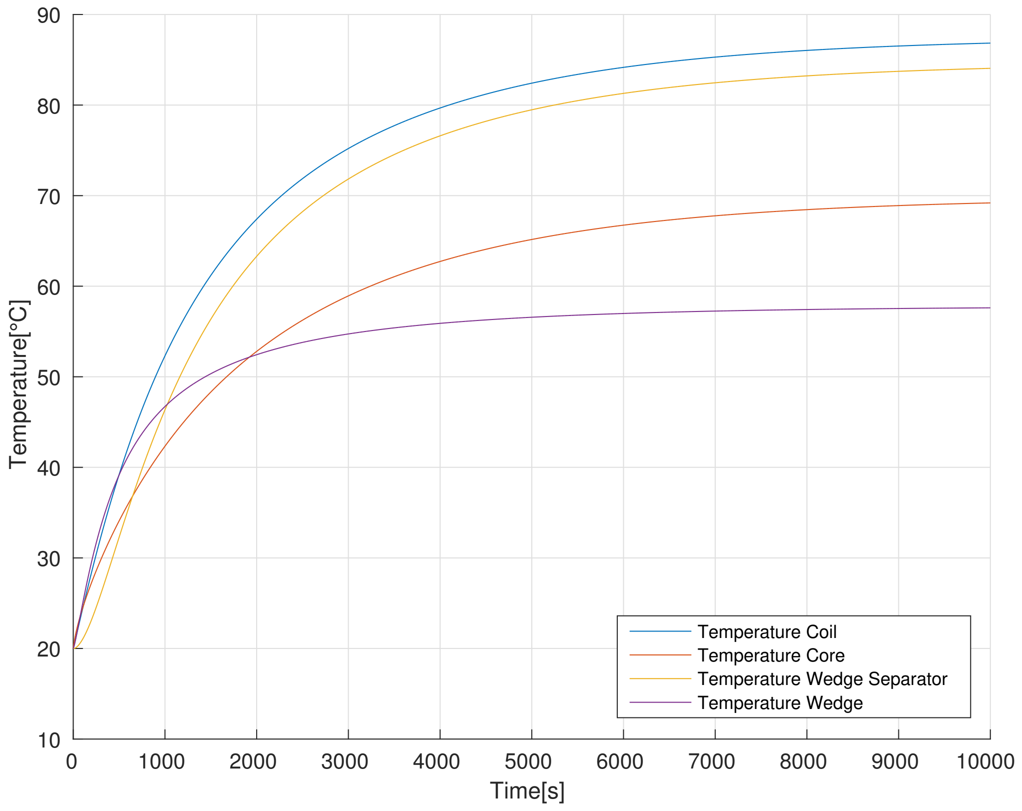

Figure 5.3: Mean temperatures in different elements

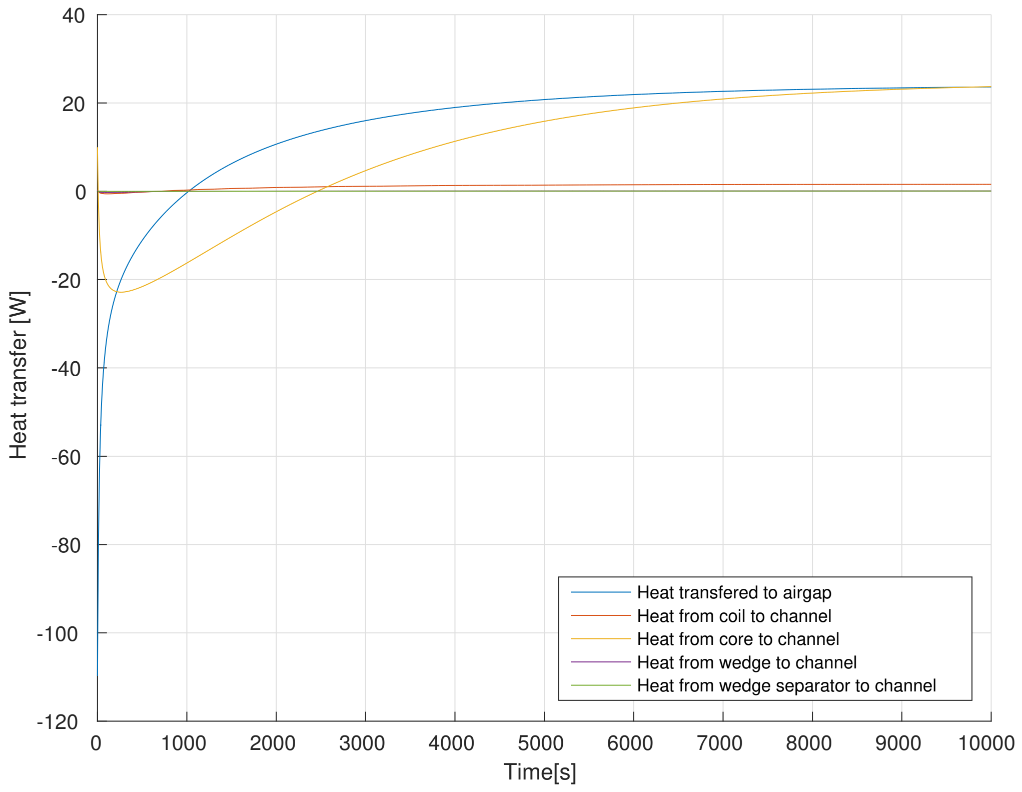

Figure 5.4: Heat transfer from different elements to air

	Normal	S1	S2	S3	S4	S5
Volume Flow Ch1 $[m^3/s]$	7.76E-4	3E-4	4.2E-4	5.4E-4	6.6E-4	7.72E-4
Volume Flow Ch2 $[m^3/s]$	2.5E-4	9E-4	7E-4	5.4E-4	3.9E-4	2.56E-4
Volume Flow Ch3 $[m^3/s]$	3.49E-4	12E-4	9.7E-4	7.5E-4	5.4E-4	3.56E-4
Pressure drop ducts $[Pa]$	79.2	103	98	93	89	83.5
HTC Ch1 $[W/m^2 \cdot K]$	55.8	13.9	26.7	37.5	47	55.4
HTC Ch2 $[W/m^2 \cdot K]$	35.9	138.6	113.7	88.7	63.3	37.2
HTC Ch3 $[W/m^2 \cdot K]$	37.8	136.5	112.4	88.3	63.9	38.8
Mean coil temperature $[°C]$	86.8	93.5	89.6	88.1	87.5	87.4
Mean core temperature $[°C]$	69.6	78.3	73.5	71.4	70.4	69.8
Heat transferred air-gap [W]	24.1	29.3	26.2	24.9	24.3	24
Heat transferred from coil [W]	0.5	4.2	3.5	3	2.4	1.6
Heat transferred from core [W]	25.9	17	20.7	22.6	23.8	24.8

Table 5.4: Behaviour comparison between different outlets width of the ventilation channel configuration

affects the heat transfer coefficient of the channels in the same way, reducing it for the 'tooth channels' and increasing it on the 'coil channels' (see Figure 5.6). The heat transferred from the coil to the channels increases around 4W but the heat transferred from the core to the channels decreases in almost 10W (see Figure 5.7). The behaviour of the heat transfer explains the mean temperatures for the coil, which rises almost 7 °C even if its better cooled, and the temperature of the core, that increases almost 9 °C. The rise of the core temperature is the reason of the increase of its heat transfer to the air-gap in 5W.

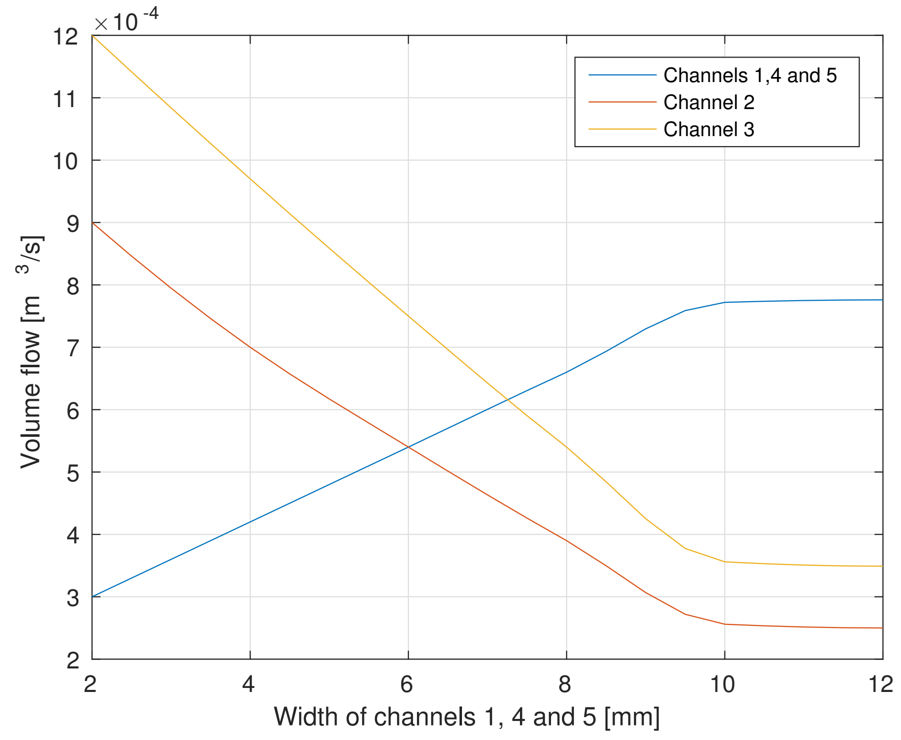

Figure 5.5: Airflow in the ventilation channels while varying the width of the channels outlet

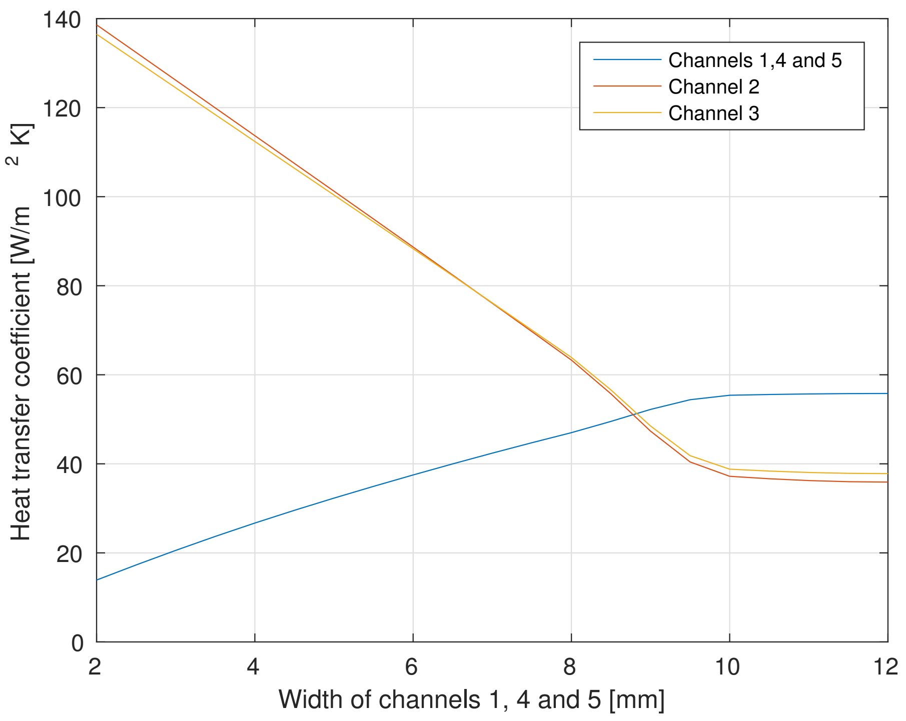

Figure 5.6: Heat transfer coefficient in the ventilation channels while varying the width of the channels outlet

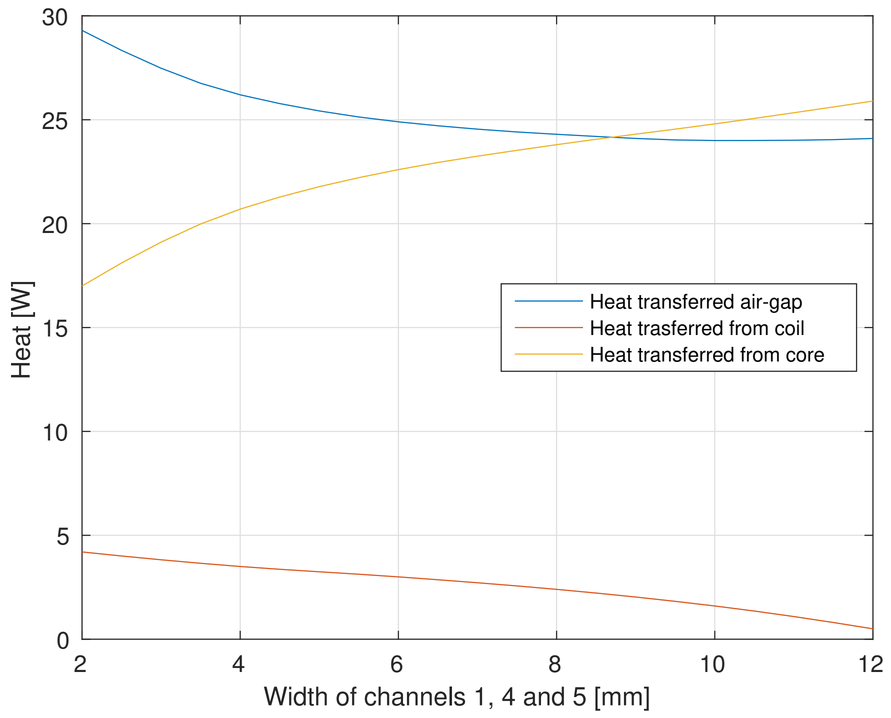

Figure 5.7: Heat transfer from different elements to air while varying the width of the channels outlet

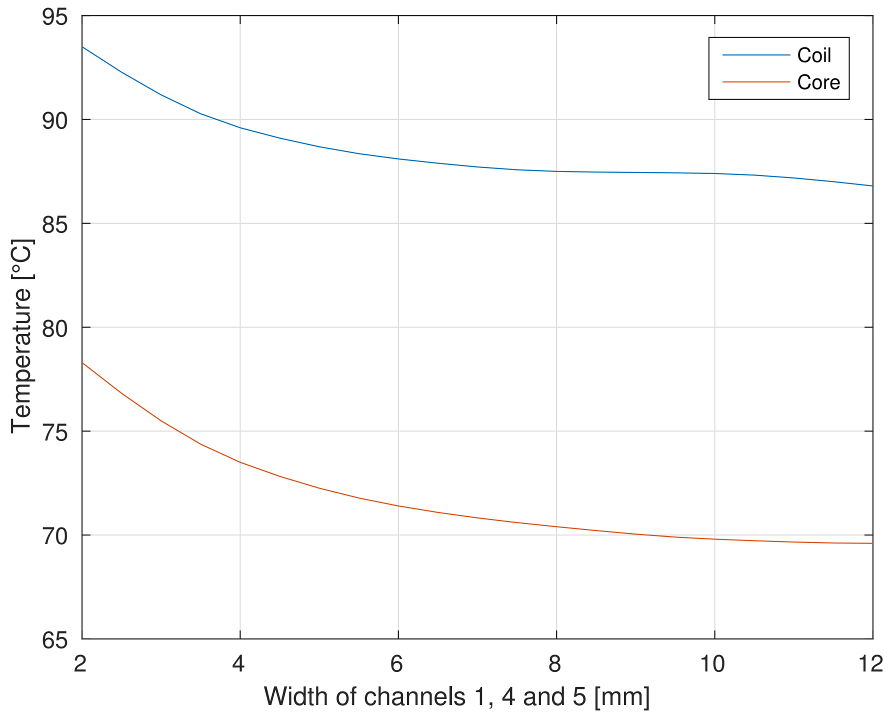

Figure 5.8: Mean temperatures of coil and core while varying the width of the channels outlet

6 Conclusion

As the thermal study of ventilation of generators and hydro-generator in particular is still a field of study in continuous development, the conception, calibration and performance of any tool for analysis should be considered carefully. In this Master thesis a model has been developed using a network of lumped elements validated against a real generator designed by Voith Hydro AB.

6.1 Validation

From the results presented in section 5.1, the model of the generator is validated (the relative errors are considered acceptable as the model is a simplification). Several calibration and validations against different load conditions and geometries would be recommended to assure a great performance of the model and a further understanding on how each parameter affects the whole model.

On the other hand, the validation performed confirms the utility and the accuracy of a simplified method like the network modelling for complex cases of studies as in the case of an electric generator. Executing the script to build a whole new model takes few minutes and a simulation to steady states less than one, confirming the versatility and simplicity that were the main goals of the model.

6.2 Effect of changing the outlet width

Modifying the outlet width of the ventilation ducts modifies the behaviour of the air in the ducts and then the heat transfer in the stator. A parametric study like the performed in this master thesis is the perfect example of the usefulness of an analytic method as the lumped element network. While designing a generator, the designer could want to test if increasing the volume flow in the ducts around the coil would help it cool better. As the system is complex, there is not an apparent solution and a simulation has to be done. Network models allow the user to simulate the system in few minutes and extract conclusions for his guessing. In the studied case, increasing the width of the outlet of the 'coil ducts' increases the coil temperature, being the behaviour of the cooling system worse.

6.3 Possible future work

As the only part of the generator implemented is a slot of the stator it would be interesting to model/implement the following parts:

- Rotor: Perform a thermal network of an equivalent section of the rotor.
- Full fluid network: Extend the current fluid network to the whole generator (axially) including all the packages of the generator (all the axial channel possibilities that has the fluid) as well as the geometry of the rotor, the fans, the returning channels and the coolers.
- Cooling system: Implement the thermal calculation of the primary and secondary cooling system, being able to calculate the temperatures of the cool air at different load conditions.
- Interface: It would be interesting to implement more friendly visual interface to watch the results because right now it is necessary to check the number or position of a concrete node in the model in order to know its temperature.
- Further validations: As it was said in section 6.1, to achieve the full potential of the model, further validations would be recommended.

Bibliography

[1] International Energy Agency, *Sweden: Electricity and Heat report for 2014.* `https://www.iea.org/statistics/statisticssearch/report/?year=2014&country=SWEDEN&product=ElectricityandHeat`

[2] Incropera, De Witt, Bergmann, Lavine. (2006) *Fundamentals of Heat and Mass Transfer.* 6th Edition

[3] I.E. Idelchik. (1966) *Handbook of Hydraulic Resistance.* 1st Edition

[4] *Mathworks: MATLAB webpage* `https://se.mathworks.com`

[5] Bonals, L.A. (2013) *Termotecnia.* 1st Edition (In Catalan)

[6] Boglietti, A. et al. *Evolution and Modern Approaches for Thermal Analysis of Electrical Machines.* Ieee Transactions on Industrial electronics, Vol. 56, No. 3, (March 2009). DOI: 10.1109/TIE.2008.2011622

[7] Chu, S. and Majumdar,A. *Opportunities and challenges for a sustainable energy future*, Nature 488, p.294–303 (16 August 2012) DOI:10.1038/nature11475 `https://www.nature.com/nature/journal/v488/n7411/pdf/nature11475.pdf`

[8] Moller, H. *Hydropower Continues Steady Growth*, Earth Policy Inst. (June 14, 2012), `http://www.earth-policy.org`

[9] International Energy Agency, *Technology Road-map: Hydropower*, (2012) `http://www.iea.org/publications/freepublications/publication/2012_Hydropower_Roadmap.pdf`

[10] Rudberg, P.M. et al. *Mitigating the Adverse Effects of Hydropower Projects: A Comparative Review of River Restoration and Hydropower Regulation in Sweden and the United States*, Georgetown International Environmental Law Review, 27(2), 251-274. (2015) `https://www.sei-international.org/publications?pid=2750`

[11] Gnielinski, V. *New Equations for Heat and Mass Transfer in Turbulent Pipe and Channel Flow*, Int. Chemical Engineering, 16 (1976), pp. 359–368

[12] *Swedish Hydropower webpage*: `http://www.elforsk.se/SVC/Forskningsprojekt/Elektromekanik/`

[13] International Energy Agency, *World Energy Outlook 2011* `http://www.worldenergyoutlook.org/`

A
Appendix

A.1 inputgen.m

```
 %INPUTS FOR THE GENERATOR THERMAL MODEL

%Geometry [IS units]

%Stator

Q=288;                    %Number of slots
Dsi=7.096;                %Stator core inner diameter [m]
lbr=2.250;                %Stator core length [m]
ln=2.065;                 %Active Stator core length [m]
lcoil=lbr;                %Coil total length [m]
hs=0.178;                 %Nominal slot depth [m]
bs=29E-3;                 %Nominal slot width [m]
hsstr=38E-3;              %Finger plate height [m]
wsstr=20E-3;              %Finger plate width [m]
hpa=2E-3;                 %Bare copper wire height [m]
bpa=10E-3;                %Bare copper wire width [m]
rpa=0.8E-3;               %Bare copper radius [m]
cr=29;                    %Number of height parallel wires
cc=2;                     %Number of width parallel wires
ns=2;                     %Number of conductors per slot
tmlh=9E-3;                %Separator between coils height [m]
ACu=1128.1E-6;            %Copper area in stator bar [m^2]
wci=0.1E-3;               %Column insulation width [m]
wmica=2E-3;               %Micalastic width [m]
ww=0.1E-3;                %Wire insulation width [m]
wcscs=0.1E-3;             %Coil shield corona shielding [m]
wccp=0.1E-3;              %Conductive cured paste width [m]
wsl=0.1E-3;               %Slot liner width [m]
tcore=54.3E-3;            %Core thicknes [m]
tc=1E-3;                  %Contact thicknes core [m]
ncp=5;                    %Parallel channels
npp=38;                   %Number of parallel packages
ho=5E-3;                  %Height channel [m]
```

```
Ro=3.953;                %Outer radius core [m]
Ri=3.548;                %Inner radius core [m]
hwedge=16.2E-3;          %Height wedge [m]
wtooth=48.2E-3;          %Width tooth [m] (Chord at the edge)
ws=2E-3;                 %Width spacers
Qpsd=0.78;               %Percentatge of the air going to the
    stator ducts
Sws=12.4E-6;             %Surface in the tooth for wedge slot
Stf=(((32+20.3)*15.3*2)/12)*1E-6;    %Surface frame in iron
   (total)
hi=tcore+ho;             %Height air inlet
wi=[11.4E-3,5E-3,7E-3,11.4E-3,11.4E-3];     %Channel
   entrance width
wo=[14E-3,26E-3,14E-3,14E-3];               %Channel outlet
   width



%Physical properties [IS units]

rCu=0.017241E-6;         %Volume resistivity for copper [ohm.
   m^2/m]
kCu=401;                 %Thermal conductivity copper [W/m.K]
kmica=0.27;              %Thermal conductivity mica [W/m.K]
kins=0.5;
kcol=0.5;
kccp=5;
cpCu=385;                %Specific heat copper [J/kg.K]
rhoCu=8960;              %Density copper [kg/m^3]
kCoilr=((hpa+2*ww)*cr*kCu*kins)/((hpa*cr*kins)+(2*ww*cr*kCu)
   ); %Equivalent radial conductivity coil (serie)
kCoiltg=(((bpa+2*ww)*cc+(cc-1)*wci)*kCu*kins*kcol)/((bpa*cc*
   kins*kcol)+(2*ww*cc*kCu*kcol)+((cc-1)*wci*kCu*kins));   %
   Equivalent tangential conductivity coil (serie)

%Air coditions

rhoAir=1;                %Density air [kg/m^3] @80C
cpAir=1.09E3;            %Specific heat air [J/kg.K]
kair=0.02;               %Conductivity air [W/m.K]

%Core

ksteel=37;               %Conductivity steel [W/m.K]
kCorea=(ksteel*kair*(tcore+tc))/(ksteel*tc+kair*tcore);
           %Equivalent axial conductivity
```

```
 cpsteel=466;              %Specific heat steel [J/kg.K]
 rhosteel=7800;            %Density steel [kg/m^3]

%Wedge

kwedge=0.5;                 %Conductivity wedge [W/m.K]
cpwedge=1400;               %Specific heat wedge [J/kg.K]
rhowedge=1900;             %Density wedge [kg/m^3]

%Thermal loses

 vl=207E3;                 %Ventilation losses [W]
 il=(295+37)*1E3;          %Iron losses [W]
 swl=222E3;                %Stator winding losses [W]
 rwl=418E3;                %Rotor winding losses [W]
 ll=18E3;                  %Additional load losses [W]
 ewl=(239+129)*1E3;        %Endwinding losses [W]

%Heat exchangers

 Tout=26.2;                %Outlet temperature air [C]
 Qat=5*8;                  %Air flow total [m^3/s]
 Tin=21.2;                 %Inlet temperature air [C](From
    Daniel heat test)
 Cap=220E3;                %Nominal capacity [W]
```

A.2 executemod.m

```
%Loads

netmod52;                                  %Loads variables from
    input_gen and makes network
set_param(nmodel, 'StopTime', '100000')
sim(nmodel);                           %Simulates the simulink
    model


%%%Plots
%scz=get(0,'ScreenSize');

 figure;
 hold on
 grid on
 xlabel('Time[s]')
 ylabel('Heat transfer [W]')
%title('Stator heat transfer against time')
```

```
plot(heat.time,heat.data);
print -depsc Heattransfer3

figure;
hold on;
grid on;
plot(tempcoil.time,tempcoil.data);
%title('Temperatures Coil against time')
xlabel('Time[s]')
ylabel('Temperature[ C ]')
print -depsc Tempcoil3

figure;
hold on;
grid on;
plot(meantemp.time,meantemp.data);
%title('Mean temperatures against time')
xlabel('Time[s]')
ylabel('Temperature[ C ]')
legend('Temperature Coil','Temperature Core','Temperature Wedge Separator','Temperature Wedge')
legend('Location','SouthEast')
print -depsc Tempmean3

figure;
hold on;
grid on;
plot(heatairgap.time,heatairgap.data,heatcoil.time,heatcoil.data,heatcore.time,heatcore.data,heatwedge.time,heatwedge.data,heatwedgesep.time,heatwedgesep.data);
%title('Mean temperatures against time')
xlabel('Time[s]')
ylabel('Heat transfer [W]')
legend('Heat transfered to airgap','Heat from coil to channel','Heat from core to channel','Heat from wedge to channel','Heat from wedge separator to channel')
legend('Location','SouthEast')
print -depsc Heatall3

warning('off','MATLAB:csvwrite:AddSheet');
filename='tempout.csv';
Heat=[heat.time,heat.data];
Temp=[meantemp.time,meantemp.data];
csvwrite(filename,Temp);
```

A.3 netmod.m

```
%Each node has two codifications the geometrical one with x_y_z and another
%with only one number ## to identify the thermal resistances and proprieties.

input_gen                                   %Generator geometry file
Ti=20;                                      %Initial conditions [C]
co=2;                                       %Number of columns --> MODIFY
ro=9;                                       %Number of rows   --> MODIFY (better odd)
zo=3;                                       %Number of z layers  --> MODIFY  (better odd)
zch=uint8(zo/2);                            %z layer where are situated the channels (middle)
rsc=uint8(ro/2);                            %y layer where are situated the coil separators (middle)
nmodel='model51';                           %Name of the simulink name
Mr=zeros((co+2)*(ro+2)*zo);                 %Mr --> Thermal resistances matrix and Heat generator on the diag.
rho=zeros((co+2)*(ro+2)*zo,1);              %Density matrix
cp=zeros((co+2)*(ro+2)*zo,1);               %Specific heat matrix
losses=swl;                                 %Thermal coil losses [W]
ilosses=il;                                 %Iron loses [W]
airnodes={};                                %Nodes that belong to an airchannel, add more if needed
coilnodes={};                               %Nodes that belong to the coil, add more if needed
wedgenodes={};                              %Nodes that belong to the wedge, add more if needed
ironnodes={};                               %Nodes that belong to the core, add more if needed
simnodes={};                                %Edges, add more if necessary
wsep={};                                    %Nodes that belong to the coil separator
```

```
qairgap={};                                          %Heat
   tranfer to the airgap (modified by the script)
cd=zeros((co+2)*(ro+2)*zo,3);                        %Matrix to
   code nodes number/name
Qc=(Qat*Qpsd)/((npp-1)*Q);                           %Airflow on
   the whole model
Qpc=netflowf(Qc,Ro,Ri,hs,hi,wi,ho,wo);               %Airflow per
    channel
Qpcl=Qpc(1)+Qpc(2)+Qpc(5)/2;                         %Total
   airflow on left channels
Qpcr=Qpc(3)+Qpc(4)+Qpc(5)/2;                         %Total
   airflow on right channels
htcc=fhtc(Qpc);
airTin=Tin+((vl+ewl)/(Qat*cpAir*rhoAir))+((rwl+ll)/(Qat*Qpsd
   *cpAir*rhoAir));  %Air inlet temperature [C] (Temp at
   airgap)
htcag=20;
Ptc=0;
Pti=0;

%Subsistems
new_system(nmodel);                                  %Creates the
    new Simulink system
add_block('simulink/Sources/Step',strcat(nmodel,'/Loses'),'
   After',num2str(losses),'position',[105,145,135,175]);
               %Loses Coil
add_block('simulink/Sources/Step',strcat(nmodel,'/LosesIron'
   ),'After',num2str(ilosses),'position',[105,275,135,305]);
            %Loses Iron
add_block('simulink/Sources/Constant',strcat(nmodel,'/
   AirinletT'),'Value',num2str(airTin),'position'
   ,[105,210,135,240]);      %Inlet air temp
add_block('simulink/Ports & Subsystems/Subsystem',strcat(
   nmodel,'/Network'),'position',[205,125,305,425]);
delete_line(strcat(nmodel,'/Network'),'In1/1','Out1/1');
set_param(strcat(nmodel,'/Network/In1'),'position'
   ,[-1005,-37,-975,-23]);
set_param(strcat(nmodel,'/Network/Out1'),'position'
   ,[-525,63,-495,77]);
add_block('simulink/Ports & Subsystems/In1',strcat(nmodel,'/
   Network/In2'),'position',[-850,200,-820,214]);
add_block('simulink/Ports & Subsystems/In1',strcat(nmodel,'/
   Network/In3'),'position',[-1205,200,-1175,214]);add_line(
   nmodel,'Loses/1','Network/1');
add_line(nmodel,'AirinletT/1','Network/2');
add_line(nmodel,'LosesIron/1','Network/3');
```

```
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'
    /Network/Out2'),'position',[-1420,450,-1400,464]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout2'),'inputs','1','position'
    ,[-1480,400,-1475,550]);
add_line(strcat(nmodel,'/Network'),'muxout2/1','Out2/1');
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout1'),'inputs','1','position'
    ,[-540,0,-535,140]);
add_line(strcat(nmodel,'/Network'),'muxout1/1','Out1/1');
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    tempcoil'),'variablename','tempcoil','position'
    ,[360,135,420,165]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/heat
    '),'variablename','heat','position',[715,185,775,215]);
add_line(nmodel,'Network/1','tempcoil/1');
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'
    /Network/Out3'),'position',[-420,550,-400,564]);
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'
    /Network/Out4'),'position',[-420,650,-400,664]);
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'
    /Network/Out5'),'position',[-420,750,-400,764]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout3'),'inputs','1','position'
    ,[-480,500,-475,600]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout4'),'inputs','1','position'
    ,[-480,600,-475,700]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout5'),'inputs','1','position'
    ,[-480,700,-475,800]);
add_line(strcat(nmodel,'/Network'),'muxout3/1','Out3/1');
add_line(strcat(nmodel,'/Network'),'muxout4/1','Out4/1');
add_line(strcat(nmodel,'/Network'),'muxout5/1','Out5/1');
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    tempcore'),'variablename','tempcore','position'
    ,[360,235,420,265]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    tempwedge'),'variablename','tempwedge','position'
    ,[360,285,420,315]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    tempwsep'),'variablename','tempwsep','position'
    ,[360,335,420,365]);
add_line(nmodel,'Network/3','tempcore/1');
add_line(nmodel,'Network/4','tempwedge/1');
add_line(nmodel,'Network/5','tempwsep/1');
```

```
add_block('dspstat3/Mean',strcat(nmodel,'/tmcoil'),'position',[535,120,590,160]);
add_block('dspstat3/Mean',strcat(nmodel,'/tmcore'),'position',[535,180,590,220]);
add_block('dspstat3/Mean',strcat(nmodel,'/tmwedge'),'position',[535,240,590,280]);
add_block('dspstat3/Mean',strcat(nmodel,'/tmwsep'),'position',[535,300,590,340]);
add_line(nmodel,'Network/1','tmcoil/1');
add_line(nmodel,'Network/3','tmcore/1');
add_line(nmodel,'Network/4','tmwedge/1');
add_line(nmodel,'Network/5','tmwsep/1');
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/muxmean'),'inputs','4','position',[675,201,680,239]);
add_line(nmodel,'tmcoil/1','muxmean/1');
add_line(nmodel,'tmcore/1','muxmean/2');
add_line(nmodel,'tmwedge/1','muxmean/3');
add_line(nmodel,'tmwsep/1','muxmean/4');
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/meantemp'),'variablename','meantemp','position',[715,204,775,234]);
add_line(nmodel,'muxmean/1','meantemp/1');
add_block('simulink/Signal Routing/Goto', char(strcat(nmodel,'/Network','/T','airgap')),'GotoTag',strcat('T','airgap'),'position',[-775,180,-750,196]);
add_line(strcat(nmodel,'/Network'),'In2/1',strcat('T','airgap','/1'));
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'/Network/Out6'),'position',[-1420,950,-1400,964]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/Network/muxout6'),'position',[-1480,900,-1475,1050]);
add_line(strcat(nmodel,'/Network'),'muxout6/1','Out6/1');
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'/Network/Out7'),'position',[-1420,750,-1400,764]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/Network/muxout7'),'inputs','1','position',[-1480,700,-1475,850]);
add_line(strcat(nmodel,'/Network'),'muxout7/1','Out7/1');
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'/Network/Out8'),'position',[-1420,850,-1400,864]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/Network/muxout8'),'inputs','1','position',[-1480,800,-1475,950]);
add_line(strcat(nmodel,'/Network'),'muxout8/1','Out8/1');
add_block('simulink/Ports & Subsystems/Out1',strcat(nmodel,'/Network/Out9'),'position',[-1420,1050,-1400,1064]);
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    Network/muxout9'),'inputs','1','position'
    ,[-1480,1000,-1475,1150]);
add_line(strcat(nmodel,'/Network'),'muxout9/1','Out9/1');
add_block('simulink/Signal Routing/Mux',strcat(nmodel,'/
    muxheat'),'inputs','5','position',[600,0,605,60]);
add_line(nmodel,'Network/2','muxheat/1');
add_line(nmodel,'Network/6','muxheat/2');
add_line(nmodel,'Network/7','muxheat/3');
add_line(nmodel,'Network/8','muxheat/4');
add_line(nmodel,'Network/9','muxheat/5');
add_line(nmodel,'muxheat/1','heat/1');
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
    ,'/Sum','2')),'Inputs','+','position',[500,190,510,200]);
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
    ,'/Sum','6')),'Inputs','+','position',[500,360,510,370]);
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
    ,'/Sum','7')),'Inputs','+','position',[500,420,510,430]);
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
    ,'/Sum','8')),'Inputs','+','position',[500,480,510,490]);
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
    ,'/Sum','9')),'Inputs','+','position',[500,540,510,550]);
add_line(nmodel,'Network/2','Sum2/1');
add_line(nmodel,'Network/6','Sum6/1');
add_line(nmodel,'Network/7','Sum7/1');
add_line(nmodel,'Network/8','Sum8/1');
add_line(nmodel,'Network/9','Sum9/1');
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    heatcoil'),'variablename','heatcoil','position'
    ,[560,190,620,220]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    heatairgap'),'variablename','heatairgap','position'
    ,[560,360,620,390]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    heatcore'),'variablename','heatcore','position'
    ,[560,410,620,440]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    heatwedgesep'),'variablename','heatwedgesep','position'
    ,[560,460,620,490]);
add_block('simulink/Sinks/To Workspace',strcat(nmodel,'/
    heatwedge'),'variablename','heatwedge','position'
    ,[560,510,620,540]);
add_line(nmodel,'Sum2/1','heatcoil/1');
add_line(nmodel,'Sum6/1','heatairgap/1');
add_line(nmodel,'Sum7/1','heatcore/1');
add_line(nmodel,'Sum8/1','heatwedgesep/1');
add_line(nmodel,'Sum9/1','heatwedge/1');


xn=bs/co;                                                %size of a
    node in each direction
yn=hs/ro;
zn=lbr/npp/zo;

linknode={'0','0','0','0','0','0'};

lco=length(num2str(co+2));
lro=length(num2str(ro+2));                                 %to code
    nodes
lzo=length(num2str(zo+1));
cx=0;
c=0;
agi=0;
ani=0;
for z = 0:zo+1
    for j = 0:(ro+2)
        for i = 0:(co+2)
            nodeco(1:lco)='0';
            nodero(1:lro)='0';
            nodezo(1:lzo)='0';
            lc=length(num2str(i));
            lr=length(num2str(j));
            lz=length(num2str(z));
            nodeco(end-lc+1:end)=num2str(i);
            nodero(end-lr+1:end)=num2str(j);
            nodezo(end-lz+1:end)=num2str(z);
            node=strcat(nodeco,'_',nodero,'_',nodezo);
            if z==zch
                zn=ho;
            else
                zn=lbr/npp/zo;
            end
            if j==0
               yn=hwedge;
            elseif j==ro+1
               yn=Ro-Ri-hs;
            elseif j==rsc
               yn=tmlh;
            else
               yn=(hs-tmlh-hwedge)/(ro-1);
            end
            if or(i==0,i==co+1)
```

```
                xn=wtooth/2;
            else
                xn=bs/co;
            end
            if or(or(z==0,z==zo+1),or(i==co+2,j==ro+2))
                simnodes=[simnodes,node];
            elseif and(or(or(j==0,i==0),or(j==ro+1,i==co+1))
              ,z==zch) %Creates the list of air nodes
                airnodes=[airnodes,node];
                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd
                    (3));
                cp(nnd,1)=cpAir;
                rho(nnd,1)=rhoAir;
            elseif and(j==rsc,and(i~=0,i~=co+1));
                wsep=[wsep,node];
                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd
                    (3));
                cp(nnd,1)=cpwedge;
                rho(nnd,1)=rhowedge*xn*yn*zn;

            elseif and(and(j>0,j<(ro+1)),and(i>0,i<(co+1)))
               %Creates the list of coil nodes
                coilnodes=[coilnodes,node];
                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd
                    (3));
                rho(nnd,1)=rhoCu*xn*yn*zn;
                cp(nnd,1)=cpCu;
            elseif and(and(j==0,and(i>0,i<(co+1))),z~=zch)
                wedgenodes=[wedgenodes,node];
                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd
                    (3));
                cp(nnd,1)=cpwedge;
                rho(nnd,1)=rhowedge*xn*yn*zn;

            else
                ironnodes=[ironnodes,node];
                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd
                    (3));
                cp(nnd,1)=cpsteel;
                rho(nnd,1)=rhosteel*xn*yn*zn;
            end
            end
        end
end

for z = 1:zo
        for j = 0:(ro+1)
            for i = 0:(co+1)
                nodeco(1:lco)='0';
                nodero(1:lro)='0';
                nodezo(1:lzo)='0';

                nodecoi(1:lco)='0';
                noderoi(1:lro)='0';
                nodezoi(1:lzo)='0';

                lc=length(num2str(i));
                lr=length(num2str(j));
                lz=length(num2str(z));

                nodeco(end-lc+1:end)=num2str(i);
                nodero(end-lr+1:end)=num2str(j);
                nodezo(end-lz+1:end)=num2str(z);
                node=strcat(nodeco,'_',nodero,'_',nodezo);

                nd=str2double(strsplit(node,'_'));
                nnd=nd(1)+1+(co+2)*nd(2)+((co+2)*(ro+2))*(nd(3))
                    ;

                cx=cx+1;
                cd(cx,1)=(i+1)+((co+2)*j)+((co+2)*(ro+2))*z;
                                                %For node identification open
                    cd var
                cd(cx,2)=i;
                cd(cx,3)=j;
                cd(cx,4)=z;
                if z==zch
                    zn=ho;
                else
                    zn=lbr/npp/zo;
                end
                if j==0
                    yn=hwedge;
                elseif j==ro+1
                    yn=Ro-Ri-hs;
                elseif j==rsc
                    yn=tmlh;
                else
                    yn=(hs-tmlh-hwedge)/(ro-1);
                end
                if or(i==0,i==co+1)
                    xn=wtooth/2;
                else
                    xn=bs/co;
                end
                cd(cx,5)=xn;
                cd(cx,6)=yn;
                cd(cx,7)=zn;
                if not(ismember(node,airnodes))
                    linknode=linknodes(node);
                        if ismember(node,coilnodes)
                                                        %Generation
                            @Coil volnode/totalvol
                            Mr(nnd,nnd)=(xn*yn*zn)/(ACu*ns*lcoil
                                *Q);
                            Ptc=Ptc+(Mr(nnd,nnd)*losses);
                            add_block('simulink/Math Operations/
                                Gain', strcat(nmodel,'/Network/
                                Gen',node),'Gain',strcat('Mr(',
                                num2str(nnd),',',num2str(nnd),')'
                                ),'position',[-940,-55+15*c
                                ,-915,-35+15*c]);
                            add_line(strcat(nmodel,'/Network'),'
                                In1/1',strcat('Gen',node,'/1'));
                            add_block('simulink/Signal Routing/
                                Goto',strcat(nmodel,'/Network/
                                hgen',node),'GotoTag',strcat('
                                hgen',node),'position'
                                ,[-910,-55+15*c,-880,-35+15*c]);
                            add_line(strcat(nmodel,'/Network'),
                                strcat('Gen',node,'/1'),strcat('
                                hgen',node,'/1'));
                        elseif ismember(node,ironnodes)
                                                    %Generation @Iron
                            volnode/totalvol
                            Mr(nnd,nnd)=(xn*yn*zn)/(ln*((pi()*(
                                Ro^2-Ri^2))-((hs*bs+Sws*2+Stf)*Q)
                                ));
                            Pti=Pti+(Mr(nnd,nnd)*ilosses);
                            add_block('simulink/Math Operations/
                                Gain', strcat(nmodel,'/Network/
                                Gen',node),'Gain',strcat('Mr(',
                                num2str(nnd),',',num2str(nnd),')'
                                ),'position',[-1150,200+15*c
                                ,-1120,215+15*c]);
                            add_line(strcat(nmodel,'/Network'),'
                                In3/1',strcat('Gen',node,'/1'));
                            add_block('simulink/Signal Routing/
                                Goto',strcat(nmodel,'/Network/
                                hgen',node),'GotoTag',strcat('
                                hgen',node),'position'
                                ,[-1115,200+15*c,-1085,215+15*c])
                                ;
                            add_line(strcat(nmodel,'/Network'),
                                strcat('Gen',node,'/1'),strcat('
                                hgen',node,'/1'));
                        end
                        c=c+1;

                        if ismember(node,coilnodes)
                            Rx=xn/(2*kCoiltg*yn*zn);
                            Ry=yn/(2*kCoilr*xn*zn);
                            Rz=zn/(2*kCu*yn*xn);
                            nod=1;
                            noda=2;
                        elseif ismember(node,wedgenodes)
                            Rx=xn/(2*kwedge*yn*zn);
                            Ry=yn/(2*kwedge*xn*zn);
                            Rz=zn/(2*kwedge*yn*xn);
                            nod=5;
                            noda=9;
                        elseif ismember(node,wsep)
                            Rx=xn/(2*kwedge*yn*zn);
                            Ry=yn/(2*kwedge*xn*zn);
                            Rz=zn/(2*kwedge*yn*xn);
                            nod=4;
                            noda=8;
                        elseif ismember(node,ironnodes)
                            Rx=xn/(2*ksteel*yn*zn);
                            Ry=yn/(2*ksteel*xn*zn);
                            Rz=zn/(2*kCorea*yn*xn);
                            nod=3;
                            noda=7;
                        end

                        for x=1:6           %2D 4 directions,
                            for 3D 6 directions
                            if and(linknode{x}~='X',not(ismember
                                (node,airnodes)))
```

```
if not(ismember(strcat(nmodel,'/Network','/Gain',linknode(x),node),find_system(strcat(nmodel,'/Network'),'Type','Block')))
   if not(ismember(linknode(x),simnodes));
        if not(linknode{x}=='airgap')
            ndx=str2double(strsplit(strjoin(linknode(x)),'_'));
            nndx=ndx(1)+1+(co+2)*ndx(2)+((co+2)*(ro+2))*(ndx(3));
            if ndx(3)==zch
                zn=ho;
            else
                zn=lbr/npp/zo;
            end
            if ndx(2)==0
               yn=hwedge;
            elseif ndx(2)==ro+1
               yn=Ro-Ri-hs;
            elseif ndx(2)==rsc
               yn=tmlh;
            else
               yn=(hs-tmlh-hwedge)/(ro-1);
            end
            if or(ndx(1)==0,ndx(1)==co+1)
               xn=wtooth/2;
            else
               xn=bs/co;
            end
        end
        if ismember(linknode(x),ironnodes)
           if or(x==3,x==4)

             %Built the thermal resistance for each node
```

```
                Mr(nnd,nndx)=Ry+yn/(2*ksteel*xn*zn);
                Mr(nndx,nnd)=Ry+yn/(2*ksteel*xn*zn);
            elseif or(x==1,x==2)
                Mr(nnd,nndx)=Rx+xn/(2*ksteel*yn*zn);
                Mr(nndx,nnd)=Rx+xn/(2*ksteel*yn*zn);
            elseif or(x==5,x==6)
                Mr(nnd,nndx)=Rz+zn/(2*kCorea*yn*xn);
                Mr(nndx,nnd)=Rz+zn/(2*kCorea*yn*xn);
            end
        elseif ismember(linknode(x),coilnodes)
            if or(x==3,x==4)
                Mr(nnd,nndx)=Ry+yn/(kCoilr*xn*zn*2);
                Mr(nndx,nnd)=Ry+yn/(kCoilr*xn*zn*2);
            elseif or(x==1,x==2)
                Mr(nnd,nndx)=Rx+xn/(kCoiltg*yn*zn*2);
                Mr(nndx,nnd)=Rx+xn/(kCoiltg*yn*zn*2);
            elseif or(x==5,x==6)
                Mr(nnd,nndx)=Rz+zn/(kCu*yn*xn*2);
                Mr(nndx,nnd)=Rz+zn/(kCu*yn*xn*2);
            end
        elseif ismember(linknode
                                              (x),wedgenodes)
                                              if or(x==3,x==4)
                                                  Mr(nnd,nndx)=Ry+
                                                     yn/(kwedge*xn
                                                     *zn*2);
                                                  Mr(nndx,nnd)=Ry+
                                                     yn/(kwedge*xn
                                                     *zn*2);
                                              elseif or(x==1,x==2)
                                                  Mr(nnd,nndx)=Rx+
                                                     xn/(kwedge*yn
                                                     *zn*2);
                                                  Mr(nndx,nnd)=Rx+
                                                     xn/(kwedge*yn
                                                     *zn*2);
                                              elseif or(x==5,x==6)
                                                  Mr(nnd,nndx)=Rz+
                                                     zn/(kwedge*yn
                                                     *xn*2);
                                                  Mr(nndx,nnd)=Rz+
                                                     zn/(kwedge*yn
                                                     *xn*2);
                                              end
                                           elseif ismember(linknode
                                              (x),wsep)
                                              if or(x==3,x==4)
                                                  Mr(nnd,nndx)=Ry+
                                                     yn/(kwedge*xn
                                                     *zn*2);
                                                  Mr(nndx,nnd)=Ry+
                                                     yn/(kwedge*xn
                                                     *zn*2);
                                              elseif or(x==1,x==2)
                                                  Mr(nnd,nndx)=Rx+
                                                     xn/(kwedge*yn
                                                     *zn*2);
                                                  Mr(nndx,nnd)=Rx+
                                                     xn/(kwedge*yn
                                                     *zn*2);
                                              elseif or(x==5,x==6)
                                                  Mr(nnd,nndx)=Rz+
                                                     zn/(kwedge*yn
                                                     *xn*2);
                                                  Mr(nndx,nnd)=Rz+
                                                     zn/(kwedge*yn
                                                     *xn*2);
```

```
                    end
                elseif ismember(linknode(x),airnodes)
                    ani=ani+1;
                    if x==3
                        Mr(nnd,nndx)=Ry+1/(htcc(2)*xn*zn);
                        Mr(nndx,nnd)=Ry+1/(htcc(2)*xn*zn);
                    elseif x==4
                        Mr(nnd,nndx)=Ry+1/(htcc(3)*xn*zn);
                        Mr(nndx,nnd)=Ry+1/(htcc(3)*xn*zn);
                    elseif x==1
                        Mr(nnd,nndx)=Rx+1/(htcc(3)*yn*zn);
                        Mr(nndx,nnd)=Rx+1/(htcc(3)*yn*zn);
                    elseif x==2
                        Mr(nnd,nndx)=Rx+1/(htcc(2)*yn*zn);
                        Mr(nndx,nnd)=Rx+1/(htcc(2)*yn*zn);
                    elseif or(x==5,x==6)
                        if ismember(node,ironnodes)
                            m=sqrt((htcc(4)*(2*ws+2*yn))/(ksteel*(ws*yn)));
                            ef=(tanh(m*(ho/2))/(m*(ho/2))); %Fin
                        efficiency (separators)
            Rc=3E-2;    %Contact resistance for steel fins typical value
            efRc=1/(1/ef+htcc(4)*(ho/2)*yn*Rc/(ws*yn));
            Mr(nnd,nndx)=Rz+1/(htcc(4)*((xn-2.5*ws)*yn+((ho/2)*yn)*2.5*efRc));
            Mr(nndx,nnd)=Rz+1/(htcc(4)*((xn-2.5*ws)*yn+((ho/2)*yn)*2.5*efRc));
        else
            Mr(nnd,nndx)=Rz+1/(htcc(4)*(xn*yn));
            Mr(nndx,nnd)=Rz+1/(htcc(4)*(xn*yn));
        end
```

```
end
add_block('simulink/
    Signal Routing/
    From', char(strcat
    (nmodel,'/Network'
    ,'/Qa',node,
    linknode(x))),'
    GotoTag',char(
    strcat('Q',node,
    linknode(x))),'
    position'
    ,[-1550,300+25*ani
    ,-1520,316+25*ani
    ]);
conea=get_param(
    strcat(nmodel,'/
    Network/muxout',
    num2str(noda)),'
    PortConnectivity')
    ;
bcona=conea.SrcBlock;
if bcona~=-1
     inp=str2double(
         get_param(
         strcat(nmodel,
         '/Network/
         muxout',
         num2str(noda))
         ,'inputs'));
     set_param(strcat(
         nmodel,'/
         Network/muxout
         ',num2str(noda
         )),'inputs',
         num2str(inp+1)
         );
     add_line(strcat(
         nmodel,'/
         Network'),char
         (strcat('Qa',
         node,linknode(
         x),'/1')),char
         (strcat('
         muxout',
         num2str(noda),
         '/',num2str(
                                    inp+1)))));
                                else
                                    add_line(strcat(
                                    nmodel,'/
                                    Network'),char
                                    (strcat('Qa',
                                    node,linknode(
                                    x),'/1')),char
                                    (strcat('
                                    muxout',
                                    num2str(noda),
                                    '/',num2str(1)
                                    )));
                                end

                            elseif strcmp(linknode{x
                                },'airgap')
                                agi=agi+1;
                                if x==3
                                    Mr(nnd,1)=Ry+1/(
                                        htcag*xn*zn);
                                    Mr(1,nnd)=Ry+1/(
                                        htcag*xn*zn);
                                elseif x==4
                                    Mr(nnd,1)=Ry+1/(
                                        htcag*xn*zn);
                                    Mr(1,nnd)=Ry+1/(
                                        htcag*xn*zn);
                                elseif x==1
                                    Mr(nnd,1)=Rx+1/(
                                        htcag*yn*zn);
                                    Mr(1,nnd)=Rx+1/(
                                        htcag*yn*zn);
                                elseif x==2
                                    Mr(nnd,1)=Rx+1/(
                                        htcag*yn*zn);
                                    Mr(1,nnd)=Rx+1/(
                                        htcag*yn*zn);
                                elseif or(x==5,x==6)
                                    Mr(nnd,1)=Rz+1/(
                                        htcag*(xn*yn)
                                        );
                                    Mr(1,nnd)=Rz+1/(
                                        htcag*(xn*yn)
                                        );
                                end
```

```
            set_param(strcat(
                nmodel,'/Network/
                muxout6'),'inputs'
                ,num2str(agi));
            add_block('simulink/
                Signal Routing/
                From', char(strcat
                (nmodel,'/Network'
                ,'/Qa',node,
                linknode(x))),'
                GotoTag',char(
                strcat('Q',node,
                linknode(x))),'
                position'
                ,[-1550,900+25*agi
                ,-1520,916+25*agi
                ]);
            add_line(strcat(
                nmodel,'/Network')
                ,char(strcat('Qa',
                node,linknode(x),'
                /1')),char(strcat(
                'muxout6/',num2str
                (agi))));
            qairgap=[qairgap,node
                ];
        end

        if or(and(ismember(node,
            coilnodes),not(
            ismember(linknode(x),
            coilnodes))),and(
            ismember(linknode(x),
            coilnodes),not(
            ismember(node,
            coilnodes))))
            if or(x==3,x==4)
                Mr(nnd,nndx)=Mr(
                    nnd,nndx)+
                    wmica/(kmica*
                    xn*zn)+wccp/(
                    kccp*xn*zn);
                Mr(nndx,nnd)=Mr(
                    nndx,nnd)+
                    wmica/(kmica*
                    xn*zn)+wccp/(
                                    kccp*xn*zn);
                          elseif or(x==1,x==2)
                              Mr(nnd,nndx)=Mr(
                                 nnd,nndx)+
                                 wmica/(kmica*
                                 yn*zn)+wccp/(
                                 kccp*yn*zn);
                              Mr(nndx,nnd)=Mr(
                                 nndx,nnd)+
                                 wmica/(kmica*
                                 yn*zn)+wccp/(
                                 kccp*yn*zn);
                          elseif or(x==5,x==6)
                              Mr(nnd,nndx)=Mr(
                                 nnd,nndx)+
                                 wmica/(kmica*
                                 xn*yn)+wccp/(
                                 kccp*xn*yn);
                              Mr(nndx,nnd)=Mr(
                                 nndx,nnd)+
                                 wmica/(kmica*
                                 xn*yn)+wccp/(
                                 kccp*xn*yn);
                          end
                      end
                      add_block('simulink/
                         Signal Routing/From',
                         char(strcat(nmodel,'/
                         Network','/T',node,'
                         from',node,'to',
                         linknode(x))),'GotoTag
                         ',strcat('T',node),'
                         position',[50+420*(j
                         -1),50+80*(x-1)+500*(i
                         -1)+2500*(z-1)
                         ,80+420*(j-1),70+80*(x
                         -1)+500*(i-1)+2500*(z
                         -1)]);
                      add_block('simulink/
                         Signal Routing/From',
                         char(strcat(nmodel,'/
                         Network','/T',linknode
                         (x),'from',node,'to',
                         linknode(x))),'GotoTag
                         ',char(strcat('T',
                         linknode(x))),'
```

```
          position',[50+420*(j
          -1),90+80*(x-1)+500*(i
          -1)+2500*(z-1)
          ,80+420*(j-1),110+80*(
          x-1)+500*(i-1)+2500*(z
          -1)]);
      add_block('simulink/Math
          Operations/Sum', char(
          strcat(nmodel,'/
          Network','/Sum',node,
          linknode(x))),'Inputs'
          ,'+-','position'
          ,[100+420*(j-1)
          ,70+80*(x-1)+500*(i-1)
          +2500*(z-1),120+420*(j
          -1),90+80*(x-1)+500*(i
          -1)+2500*(z-1)]);
      add_block('simulink/Math
          Operations/Gain', char
          (strcat(nmodel,'/
          Network','/Gain',node,
          linknode(x))),'Gain',
          char(strcat('1/Mr(',
          num2str(nnd),',',
          num2str(nndx),')')),'
          position',[135+420*(j
          -1),65+80*(x-1)+500*(i
          -1)+2500*(z-1)
          ,165+420*(j-1),95+80*(
          x-1)+500*(i-1)+2500*(z
          -1)]);
      add_block('simulink/
          Signal Routing/Goto',
          char(strcat(nmodel,'/
          Network','/Q',node,
          linknode(x))),'GotoTag
          ',char(strcat('Q',node
          ,linknode(x))),'
          position',[185+420*(j
          -1),67+80*(x-1)+500*(i
          -1)+2500*(z-1)
          ,220+420*(j-1),93+80*(
          x-1)+500*(i-1)+2500*(z
          -1)]);
      add_line(strcat(nmodel,'/
          Network'),char(strcat(
                                    'T',node,'from',node,'to',linknode(x),'/1')), char(strcat('Sum',node,linknode(x),'/1')));
                                add_line(strcat(nmodel,'/Network'),char(strcat('T',linknode(x),'from',node,'to',linknode(x),'/1')), char(strcat('Sum',node,linknode(x),'/2')));

                                    add_line(strcat(nmodel,'/Network'),char(strcat('Sum',node,linknode(x),'/1')),char(strcat('Gain',node,linknode(x),'/1')));
                                add_line(strcat(nmodel,'/Network'),char(strcat('Gain',node,linknode(x),'/1')), char(strcat('Q',node,linknode(x),'/1')));

                            end
                        end
                    end
                end
                ht=intersect(find_system(nmodel),{strcat(nmodel,'/Network/Q',node,char(linknode(1))),strcat(nmodel,'/Network/Q',char(linknode(1)),node),strcat(nmodel,'/Network/Q',node,char(linknode(2))),strcat(nmodel,'/Network/Q',char(linknode(2)),node),strcat(nmodel,'/Network/Q',node,char(linknode(3))),strcat(nmodel,'/Network/Q',char(linknode(3)),node),strcat(nmodel,'/Network/Q',node,char(linknode(4))),strcat(nmodel,'/Network/Q',char(linknode(4)),node),strcat(nmodel,'/Network/Q',node,char(linknode(5))),strcat(nmodel,'/Network/Q',char(
            linknode(5)),node),strcat(nmodel,'/
            Network/Q',node,char(linknode(6))),
            strcat(nmodel,'/Network/Q',char(
            linknode(6)),node)});
        b=length(ht);
        sig='';
        for n = 1:b
            tr=char(ht(n));
            tr=tr(end-2*(lro+lco+lzo)-3:end);
            add_block('simulink/Signal Routing/
               From', strcat(nmodel,'/Network','/
               Q',node,'to',tr),'GotoTag',strcat(
               'Q',tr),'position',[240+420*(j-1)
               ,50+50*(n-1)+500*(i-1)+2500*(z-1)
               ,280+420*(j-1),70+50*(n-1)+500*(i
               -1)+2500*(z-1)]);
            if tr(1:length(node))==node          %
               Heat transfer sign criterion
                sig=strcat(sig,'-');
            else
                sig=strcat(sig,'+');
            end
        end
        add_block('simulink/Math Operations/Sum',
            strcat(nmodel,'/Network','/Sum',node,
            'h'),'Inputs',sig,'position'
            ,[300+420*(j-1),120+500*(i-1)+2500*(z
            -1),320+420*(j-1),140+500*(i-1)+2500*(
            z-1)]);
        for m = 1:b
            tr=char(ht(m));
            tr=tr(end-2*(lro+lco+lzo)-3:end);
            add_line(strcat(nmodel,'/Network'),
               strcat('Q',node,'to',tr,'/1'),
               strcat('Sum',node,'h','/',num2str(
               m)));

        end
        add_block('simulink/Signal Routing/From',
            strcat(nmodel,'/Network/hgenin',node),
            'GotoTag',strcat('hgen',node),'
            position',[275+420*(j-1),240+500*(i-1)
            +2500*(z-1),320+420*(j-1),260+500*(i
            -1)+2500*(z-1)]);
        add_block('simulink/Math Operations/Sum',
            strcat(nmodel,'/Network','/Sum',node,
                'th'),'Inputs','++','position'
                ,[315+420*(j-1),180+500*(i-1)+2500*(z
                -1),335+420*(j-1),200+500*(i-1)+2500*(
                z-1)]);
            add_line(strcat(nmodel,'/Network'),strcat
                ('Sum',node,'h','/1'),strcat('Sum',
                node,'th','/1'));
            add_line(strcat(nmodel,'/Network'),strcat
                ('hgenin',node,'/1'),strcat('Sum',node
                ,'th','/2'));
            add_block('simulink/Math Operations/Gain'
                ,strcat(nmodel,'/Network/cap',node),'
                Gain',strcat('1/(rho(',num2str(nnd),'
                ,1)','*cp(',num2str(nnd),',1))'),'
                position',[350+420*(j-1),180+500*(i-1)
                +2500*(z-1),370+420*(j-1),200+500*(i
                -1)+2500*(z-1)]);
            add_line(strcat(nmodel,'/Network'),strcat
                ('Sum',node,'th/1'),strcat('cap',node,
                '/1'));
            add_block('simulink/Continuous/Integrator
                ',strcat(nmodel,'/Network/int',node),'
                InitialCondition','Ti','position'
                ,[380+420*(j-1),180+500*(i-1)+2500*(z
                -1),400+420*(j-1),200+500*(i-1)+2500*(
                z-1)]);
            add_line(strcat(nmodel,'/Network'),strcat
                ('cap',node,'/1'),strcat('int',node,'
                /1'));
            add_block('simulink/Signal Routing/Goto',
                 strcat(nmodel,'/Network','/T',node,'n
                '),'GotoTag',strcat('T',node),'
                position',[410+420*(j-1),180+500*(i-1)
                +2500*(z-1),430+420*(j-1),200+500*(i
                -1)+2500*(z-1)]);
            add_line(strcat(nmodel,'/Network'),strcat
                ('int',node,'/1'),strcat('T',node,'n',
                '/1'));
            add_block('simulink/Signal Routing/From',
                 strcat(nmodel,'/Network','/To',node,'
                n'),'GotoTag',strcat('T',node),'
                position',[-690,0+25*c,-660,16+25*c]);
            cone=get_param(strcat(nmodel,'/Network/
                muxout',num2str(nod)),'
                PortConnectivity');
            bcone=cone.SrcBlock;
```

```
                    if bcone~=-1
                        inp=str2double(get_param(strcat(
                           nmodel,'/Network/muxout',num2str(
                           nod)),'inputs'));
                        set_param(strcat(nmodel,'/Network/
                           muxout',num2str(nod)),'inputs',
                           num2str(inp+1));
                        add_line(strcat(nmodel,'/Network'),
                           char(strcat('To',node,'n','/1')),
                           char(strcat('muxout',num2str(nod),
                           '/',num2str(inp+1))));
                    else
                        add_line(strcat(nmodel,'/Network'),
                           char(strcat('To',node,'n','/1')),
                           char(strcat('muxout',num2str(nod),
                           '/',num2str(1))));
                    end
            end
        end
    end
end

sicag(1:length(qairgap))='+';
add_block('simulink/Math Operations/Sum', strcat(nmodel,'/
   Network','/Sum','airgap','a'),'Inputs',sicag,'position'
   ,[55+200*(-1),-620,75+200*(-1),-600]);
for n = 1:length(qairgap)
            add_block('simulink/Signal Routing/From', strcat
               (nmodel,'/Network','/Qu',qairgap{n},'airgap')
               ,'GotoTag',strcat('Q',qairgap{n},'airgap'),'
               position',[-5+200*(-1),-635+30*(n-1)
               ,25+200*(-1),-615+30*(n-1)]);
            add_line(strcat(nmodel,'/Network'),strcat('Qu',
               qairgap{n},'airgap','/1'),strcat('Sum','
               airgap','a','/',num2str(n)));
end
add_block('simulink/Math Operations/Gain', char(strcat(
   nmodel,'/Network','/Gain','airgap')),'Gain',char(strcat('
   1/(rhoAir*cpAir*Qc)')),'position',[85+200*(-1)
   ,-625,110+200*(-1),-595]);
add_block('simulink/Math Operations/Sum', char(strcat(nmodel
   ,'/Network','/Sum','airgaps')),'Inputs','++','position'
   ,[110+200*(-1),-640,130+200*(-1),-620]);
add_block('simulink/Signal Routing/From', char(strcat(nmodel
   ,'/Network','/Tu','airgap')),'GotoTag',strcat('T','airgap
   '),'position',[50+200*(-1),-685,80+200*(-1),-665]);
```

```
add_line(strcat(nmodel,'/Network'),char(strcat('Tu','airgap','/1')), char(strcat('Sum','airgaps','/1')));
add_line(strcat(nmodel,'/Network'),char(strcat('Sum','airgap','a','/1')), char(strcat('Gain','airgap','/1')));
add_line(strcat(nmodel,'/Network'),char(strcat('Gain','airgap','/1')), char(strcat('Sum','airgaps','/2')));
add_block('simulink/Signal Routing/Goto', char(strcat(nmodel,'/Network','/Tun',airnodes{2})),'GotoTag',char(strcat('T',airnodes{2})),'position',[150+200*(-1),-680,185+200*(-1),-660]);
add_line(strcat(nmodel,'/Network'),char(strcat('Sum','airgaps','/1')), char(strcat('Tun',airnodes{2},'/1')));
add_block('simulink/Signal Routing/Goto', char(strcat(nmodel,'/Network','/Tun',airnodes{3})),'GotoTag',char(strcat('T',airnodes{3})),'position',[150+200*(-1),-640,185+200*(-1),-620]);
add_line(strcat(nmodel,'/Network'),char(strcat('Sum','airgaps','/1')), char(strcat('Tun',airnodes{3},'/1')));


for ani=1:length(airnodes)
    airn=airnodes{ani};
    nair=strsplit(airn,'_');
    if not(and(str2double(nair{2})==(ro+1),and(str2double(nair{1})>0,str2double(nair{1})<(co+1))))
       alink=linknodes(airn);
       qair=intersect(find_system(nmodel),{strcat(nmodel,'/Network/Q',airn,char(alink(1))),strcat(nmodel,'/Network/Q',char(alink(1)),airn),strcat(nmodel,'/Network/Q',airn,char(alink(2))),strcat(nmodel,'/Network/Q',char(alink(2)),airn),strcat(nmodel,'/Network/Q',airn,char(alink(3))),strcat(nmodel,'/Network/Q',char(alink(3)),airn),strcat(nmodel,'/Network/Q',airn,char(alink(4))),strcat(nmodel,'/Network/Q',char(alink(4)),airn),strcat(nmodel,'/Network/Q',airn,char(alink(5))),strcat(nmodel,'/Network/Q',char(alink(5)),airn),strcat(nmodel,'/Network/Q',airn,char(alink(6))),strcat(nmodel,'/Network/Q',char(alink(6)),airn)});
       sic(1:length(qair))='+';
       add_block('simulink/Math Operations/Sum', strcat(nmodel,'/Network','/Sum',airn,'a'),'Inputs',sic,'position',[55+200*(ani-1),-620,75+200*(ani-1),-600]);
       for n = 1:length(qair)
          atr=char(qair(n));
```

```
            atr=atr(end-2*(lro+lco+lzo)-3:end);
            add_block('simulink/Signal Routing/From', strcat
                (nmodel,'/Network','/Qu',airn,'to',atr),'
                GotoTag',strcat('Q',atr),'position',[-5+200*(
                ani-1),-635+30*(n-1),25+200*(ani-1),-615+30*(
                n-1)]);
            add_line(strcat(nmodel,'/Network'),strcat('Qu',
                airn,'to',atr,'/1'),strcat('Sum',airn,'a','/'
                ,num2str(n)));
        end
        if str2double(nair(2))<=(co+2)/2
            Qch='Qpcl';
        else
            Qch='Qpcr';
        end
        add_block('simulink/Math Operations/Gain', char(
            strcat(nmodel,'/Network','/Gain',num2str(ani))),'
            Gain',char(strcat('1/(rhoAir*cpAir*',Qch,')')),'
            position',[85+200*(ani-1),-625,110+200*(ani-1)
            ,-595]);
        add_block('simulink/Math Operations/Sum', char(
            strcat(nmodel,'/Network','/Sum',num2str(ani))),'
            Inputs','++','position',[110+200*(ani-1)
            ,-640,130+200*(ani-1),-620]);
        add_block('simulink/Signal Routing/From', char(
            strcat(nmodel,'/Network','/Tu',airn)),'GotoTag',
            strcat('T',airn),'position',[50+200*(ani-1)
            ,-685,80+200*(ani-1),-665]);
        add_line(strcat(nmodel,'/Network'),char(strcat('Tu',
            airn,'/1')), char(strcat('Sum',num2str(ani),'/1')
            ));
        add_line(strcat(nmodel,'/Network'),char(strcat('Sum'
            ,airn,'a','/1')), char(strcat('Gain',num2str(ani)
            ,'/1')));
        add_line(strcat(nmodel,'/Network'),char(strcat('Gain
            ',num2str(ani),'/1')), char(strcat('Sum',num2str(
            ani),'/2')));
        nodeup=alink{3};
        nexta=nodeup;
        snair=strsplit(nexta,'_');
        if and(and(str2double(nair{1})~=(0),str2double(nair
            {1})~=(co+1)),str2double(nair{2})==0)
            if and(str2double(nair{1})>0,str2double(nair{1})
                <co)
                snair{1}=num2str(0);
            elseif str2double(nair{1})==(co)
```

```
                    snair{1}=num2str(co+1);
                end
                snair{2}=nair{2};
            elseif str2double(nair{2})==(ro+1)
                if str2double(nair{1})==0
                    snair{1}=num2str(1);
                elseif str2double(nair{1})==(co+1)
                    snair{1}=num2str(co);
                end
                snair{2}=nair{2};
            end
            nexta=strcat(snair(1),'_',snair(2),'_',snair(3));
          add_block('simulink/Signal Routing/Goto', char(
             strcat(nmodel,'/Network','/Tun',nexta)),'GotoTag'
             ,char(strcat('T',nexta)),'position',[150+200*(ani
             -1),-640,185+200*(ani-1),-620]);
          add_line(strcat(nmodel,'/Network'),char(strcat('Sum'
             ,num2str(ani),'/1')), char(strcat('Tun',nexta,'/1
             ')));
          sic='';
      end
end
open_system(nmodel);
```

A.4 netflowc.m

```
function Q=netflowf(Qt,Ro,Ri,hs,hi,wi,ho,wo)
    lg=Ro-Ri;                                          %Generator
       lenght
    li=hs;                                             %Coil height
    l=[lg,lg-li,lg,lg];
    rho=1;                                             %Air density
    visc=1.5E-5;                                       %Air
       viscosity
    Qo=[Qt/4,Qt/4,Qt/4,Qt/4];                          %First Q to
       calc.
    Qi=[Qo(1),Qo(2)/2,Qo(2)/2,Qo(3),Qo(4)];
    Si=hi*wi;                                          %Surface
       inlet
    %wo=[14E-3,26E-3,14E-3,14E-3];
    %wo=[17E-3,17E-3,17E-3,17E-3];
    %wo=[(34/3)*1E-3,34E-3,(34/3)*1E-3,(34/3)*1E-3];
    %wo=[20E-3,8E-3,20E-3,20E-3];
    So=ho*wo;                                          %Surface
       channels outlet
    Sci=ho*wi;
```

```
wm=[(wi(1)+wo(1))/2,wo(2),(wi(4)+wo(3))/2,(wi(5)+wo(4))
    /2];        %middle section
Sm=wm*ho;
Pm=2.*wm+2.*ho;
Pci=2.*wi+2.*ho;                                    %Wet
    perimeter
vi=Qi./Si;
vm=Qo./Sm;
ki=0.5*(1-(Sci./Si)).^(3/4);                        %Sharp edges
     inlet resistance
Dhci=4.*(Si./Pci);                                  %Hidraulic
    diameter
Dhm=4.*(Sm./Pm);
Rem=vm.*Dhm./visc;                                  %Reynolds
    number

%Friction coefficient calculation kfm
lambdalam=64./Rem;
lambdasm=0.3164./(Rem.^(1/4));
rou=1E-6;
roum=rou./Dhm;
lambdar=1./(2.*log(3.7./roum)).^2;
prt=erf((Rem-2850)./(sqrt(2)*600));
pt=0.5+0.5.*prt;
plam=1-pt;
psm=(1-prt).*pt;
pr=prt.*pt;
lambda=lambdalam.*plam+lambdasm.*psm+lambdar.*pr;
kfm=lambda.*l./Dhm;


Reci=vi.*Dhci./visc;                                %Reynolds
    number
%Friction coefficient calculation kfm
lambdalamci=64./Reci;
lambdasmci=0.3164./(Reci.^(1/4));
rouci=1E-6;
roumci=rouci./Dhci;
lambdarci=1./(2.*log(3.7./roumci)).^2;
prtci=erf((Reci-2850)./(sqrt(2)*600));
ptci=0.5+0.5.*prtci;
plamci=1-ptci;
psmci=(1-prtci).*ptci;
prci=prtci.*ptci;
lambdaci=lambdalamci.*plamci+lambdasmci.*psmci+lambdarci
    .*prci;
kfci=lambdaci.*li./Dhci;

koci=[0,(1-(Sci(2)/So(2)))^(0.5),(1-(Sci(3)/So(2)))
    ^(0.5),0,0];

ko=(1-(So./(wo.*hi))).^(0.5);                %Sharp edges
     outlet resistance
%Airflow first iteration
i=1;


n=1;
Q2=[1,Qo(end,2)/2,Qo(end,2)/2,1,1];
APm=1./2.*((Q2./Sci).^2+(ki+kfci+koci).*rho);

prec=2000;                                    %Precision
erra=APm(2)/prec;                             %Admisible
     error
ec=1/prec*20;                                 %Iteration
     multiplier


while abs(APm(n,2)-APm(n,3))>=(erra*10)
     if abs(APm(n,2)-APm(n,3))>=(erra*10)
         AQ=sqrt(abs(APm(n,2)-APm(n,3))/rho)*Sci(2)*ec;
         if (APm(n,2)-APm(n,3))>(erra*10)
             Q2(2)=Q2(2)-AQ;
             Q2(3)=Q2(3)+AQ;
         elseif (APm(n,1)-APm(n,k))<=(erra*10)
             Q2(2)=Q2(2)+AQ;
             Q2(3)=Q2(3)-AQ;
         end
     else
         Q2(2)=Q2(2);
         Q2(3)=Q2(3);
     end
     vci=Q2./Sci;
     Reci=vci.*Dhci./visc;
     lambdalamci=64./Reci;
     lambdasmci=0.3164./(Reci.^(1/4));
     prtci=erf((Reci-2850)./(sqrt(2)*600));
     ptci=0.5+0.5.*prtci;
     plamci=1-ptci;
     psmci=(1-prtci).*ptci;
     prci=prtci.*ptci;
     lambdaci=lambdalamci.*plamci+lambdasmci.*psmci+
        lambdarci.*prci;
     kfci=lambdaci.*li./Dhci;
     n=n+1;
     APm(n,:)=1./2.*((Q2./Sci).^2+(ki+kfci+koci).*rho);
     Q2(4:5)=1;
     Q2(1)=1;
end
ki=[ki(1),ki(2),ki(4),ki(5)];
AP=1./2*((Qo./So).^2+(ki+kfm+ko).*rho);
AP(2)=APm(end,2)+1./2*((Qo(2)./So(2)).^2+(kfm(2)+ko(2))
     .*rho);
cont=[];
j=1;
while or(or(abs(AP(j,1)-AP(j,2))>erra,abs(AP(j,1)-AP(j
     ,3))>erra),abs(AP(j,1)-AP(j,4))>erra)
     for k=2:4
         if abs(AP(j,1)-AP(j,k))>erra
             AQ=sqrt(abs(AP(j,1)-AP(j,k))/rho)*So(1)*ec;
             if (AP(j,1)-AP(j,k))>erra
                 Qo(1)=Qo(1)-AQ;
                 Qo(k)=Qo(k)+AQ;
             elseif (AP(j,1)-AP(j,k))<=erra
                 Qo(1)=Qo(1)+AQ;
                 Qo(k)=Qo(k)-AQ;
             end
         else
             Qo(k)=Qo(k);
         end
         n=1;
         Q2=[1,Qo(2)/2,Qo(2)/2,1,1];
         vci=Q2./Sci;
         Reci=vci.*Dhci./visc;
         lambdalamci=64./Reci;
         lambdasmci=0.3164./(Reci.^(1/4));
         prtci=erf((Reci-2850)./(sqrt(2)*600));
         ptci=0.5+0.5.*prtci;
         plamci=1-ptci;
         psmci=(1-prtci).*ptci;
         prci=prtci.*ptci;
         lambdaci=lambdalamci.*plamci+lambdasmci.*psmci+
             lambdarci.*prci;
         kfci=lambdaci.*li./Dhci;
         ki=0.5*(1-(Sci./Si)).^(3/4);                    %
             Sharp edges inlet resistance
         APm(n,:)=1./2.*((Q2./Sci).^2+(ki+kfci+koci).*rho
             );
```

```
Q2(4:5)=1;
Q2(1)=1;
prec=800;                                          % Precision
erra=APm(end,2)/prec;                              %Admisible error
ec=1/prec*10;                                      % Iteration multiplier
while abs(APm(n,2)-APm(n,3))>=(erra*10)
    if abs(APm(n,2)-APm(n,3))>=(erra*10)
        AQ=sqrt(abs(APm(n,2)-APm(n,3))/rho)*Sci(2)*ec;
        if (APm(n,2)-APm(n,3))>(erra*10)
            Q2(2)=Q2(2)-AQ;
            Q2(3)=Q2(3)+AQ;
        elseif (APm(n,1)-APm(n,k))<=(erra*10)
            Q2(2)=Q2(2)+AQ;
            Q2(3)=Q2(3)-AQ;
        end
    else
        Q2(2)=Q2(2);
        Q2(3)=Q2(3);
    end
    vci=Q2./Sci;
    Reci=vci.*Dhci./visc;
    lambdalamci=64./Reci;
    lambdasmci=0.3164./(Reci.^(1/4));
    prtci=erf((Reci-2850)./(sqrt(2)*600));
    ptci=0.5+0.5.*prtci;
    plamci=1-ptci;
    psmci=(1-prtci).*ptci;
    prci=prtci.*ptci;
    lambdaci=lambdalamci.*plamci+lambdasmci.*psmci+lambdarci.*prci;
    kfci=lambdaci.*li./Dhci;
    n=n+1;
    APm(n,:)=1./2.*((Q2./Sci).^2+(ki+kfci+koci).*rho);
    Q2(4:5)=1;
    Q2(1)=1;
end
j=j+1;
vo=Qo./So;
Reo=vo.*Dhm./visc;
lambdalam=64./Reo;
lambdasm=0.3164./(Reo.^(1/4));
```

```
                prt=erf((Reo-2850)./(sqrt(2)*600));
                pt=0.5+0.5.*prt;
                plam=1-pt;
                psm=(1-prt).*pt;
                pr=prt.*pt;
                lambda=lambdalam.*plam+lambdasm.*psm+lambdar.*pr
                    ;
                kfm=lambda.*l./Dhm;
                ki=[ki(1),ki(2),ki(4),ki(5)];
                AP(j,:)=1./2*((Qo./So).^2+(ki+kfm+ko).*rho);
                AP(j,2)=APm(end,2)+1./2*((Qo(2)./So(2)).^2+(kfm
                    (2)+ko(2)).*rho);

                cont(i,:)=[AQ,Qo,sum(Qo)];
            end
            i=i+k;
        end
    Q=[Qo(1),Q2(2),Q2(3),Qo(3),Qo(4)];
    AP(end,:)
end
```

A.5 fhtc.m

```
function htc=fhtc(Qc)
%HTC channels
ho=5E-3;                                          %Height channel
wi=[11.4E-3,5E-3,7E-3,11.4E-3,11.4E-3];           %Channel width

Sm=wi.*ho;
Pm=2.*wi+2.*ho;

vc=Qc./Sm;                                       %Airspeed per
    channel

visc=2E-5;
cp=1008;
k=0.029;
Dh=4.*Sm./Pm;
Pr=cp.*visc./k;
Re=vc.*Dh./visc;
f=0.184.*Re.^(-1/5);
Nu=((f./8).*(Re-1000).*Pr)./(1+12.7*(f./8).^(1/2).*(Pr
    .^(2/3)-1));
htc=Nu.*k./Dh;
end
```

A.6 linknodes.m

```
%linknode
function linkn=linknodes(node)
    linkn={'X','X','X','X','X','X'};
    for x=1:3
        lnode=strsplit(node,'_');
        l=length(strjoin(lnode(x)));
        xpn(1:l)='0';
        xmn(1:l)='0';
        a=str2double(lnode(x));
        xp=num2str(a+1);
        lp=length(xp);
        xm=num2str(a-1);
        lm=length(xm);
        if and(a-1<0,x~=2)
            nodem='X';
        elseif and(a-1<0,x==2)
            nodem='airgap';
        else
            xmn(end-lm+1:end)=xm;
            lnode(x)=cellstr(xmn);
            nodem=strjoin(lnode,('_'));
        end
        xpn(end-lp+1:end)=xp;
        lnode(x)=cellstr(xpn);
        nodep=strjoin(lnode,('_'));
        linkn(2*x-1:2*x)=[cellstr(nodep),cellstr(nodem)];
        xpn='0';
        xmn='0';
    end
end
```