Using a Thermal Flow Sensor as a Thermal Conductivity Sensor

Daniël Geert Bijsterveld

June 30, 2020

Bachelor Thesis

University of Twente
Faculty of Electrical Engineering, Mathematics and Computer Science
Research group: Integrated Devices and Systems

Assignment committee:
Dr.ir. R.J. Wiegerink
Prof.dr.ir. J.C.Lötters
Dr.ir. J. Groenesteijn
Dr.ir. R.A.R. van der Zee

Contents

Chapter 1

Introduction

The goal of this Bachelor assignment is to see whether an IQ+ flow sensor [1] from Bronkhorst can be used as a thermal conductivity sensor. The IQ+ flow sensor is a microchip thermal flow sensor which consists of two heaters and three temperature sensors. One of the temperature sensors is sandwiched between the heaters the others are upstream and downstream from the heaters. The flow is measured by the difference in temperature of the upstream and downstream heaters. This is then adjusted with the temperature sensor between the heaters to account for total temperature of the chip affecting the temperature difference between the upstream and downstream heaters. This measurement is dependent on the thermal properties of the measured medium. For this reason the gas and its thermal properties being measured must be known. Adding a thermal conductivity sensor can be used to compensate for the influence of the thermal conductivity. A thermal conductivity sensor can also possibly be used to determine which gas is being measured so the correct thermal properties can be used. To use in addition to the thermal flow sensor would require the thermal conductivity sensor to be able to determine the thermal conductivity independent of flow.

In this thesis first some background information will be given. Then the microchip including the flow channel will be modelled. First in two dimensions and in three dimensions without flow, then a model for flow will be added to the 3D model. the models for the three scenarios (2D, 3D and 3D with flow) will programmed in Matlab and be simulated this way. Using the simulation results it will be looked into whether the IQ+ Flow sensor can be used as a thermal conductivity sensor. Some limited measurements will be done and the results of these will be analyzed and compared to the results of the simulations.

Figure 1.1: The IQ+ Flow Module [1]

Chapter 2

Theory

2.1 Thermal flow sensors in general

A Thermal flow sensor is made to measure the mass flow of a gas. Thermal flow sensors consist of one or more heaters and sensors. It is possible to combine a heater and sensor in one sensor element.

When using hot wire anemometry a single wire element is both the heater and thermal sensor. The heated wire is cooled by the flowing gas and its temperature is measured. The temperature can be measured as the resistance of the wire changes as its temperature does. By knowing the power input and the temperature of the wire the amount of heat dissipated by the flowing gas can be calculated and used to calculate the mass flow. [2]

It is also possible to have a structure that measures the flow velocity. This can be done by having a heater upstream and a sensor downstream at a known distance from one another. The gas will be heated in pulses by the heater. By measuring the time it takes for these pulses to arrive at the sensor the flow velocity can be determined. [2]

Calorimetric flow sensors work by measuring the amount of energy needed to heat the flowing gas. To do this at least a heater upstream and a sensor downstream is needed. Typically an extra sensor upstream is added. This way the temperature difference of the two sensors can be compared to the power that was supplied by the heater.[2]

2.2 IQ+ Flow Sensor

The IQ+ flow sensor is a microchip thermal flow sensor consisting of two heaters and three thermal sensors. These are structured as a sensor upstream then the two heaters with a sensor positioned between them and then the downstream sensor, so sensor—heater-sensor-heater—sensor. A schematic image can be seen in figure 2.1. The IQ+ is a calorimetric flow meter in that it uses the difference in temperature of its outer sensors to determine the mass flow. It uses it sensor that sits between the heaters to determine the temperature of the heaters. This is used to correct for factors as the thermal conductivity of the gas or power supplied by the heaters. These factors determine the absolute temperature at the heaters which amplifies the temperature difference between the outer sensors.

The three thermal sensors are thermopiles referenced to the rim of the chip, this point of reference can be seen as Trim in figure 2.1. This means the thermal sensors measure the temperature difference between themselves and the rim of the chip. The thermal sensors and heaters are placed upon a rectangular borosilicate glass membrame of 2500 by 3000 μm which forms a square of 2500 by 2500 μm with on one side which has a 500 μm expansion to attach bondpads. The membrane has a thickness of 40 μm. The chip rests on a silicon rim. The rim is placed along the outer edges of the borosilicate glass leaving a square of 976 by 976 μm exposed to gas at the bottom of the membrane. The silicon rim is straight at the edges but on the inside it taperes down with an angle of 35.3 degrees as seen normal vector from the membrame. The chip is placed in a cavity in a flow channel. The cavity is made in a way that the top of the flow channel sits flush with the walls of the flow channel. the flow channel is 200 μm high and 6 mm deep for which the height is orthogonal to the surface of the chip and the depth is parallel to the surface of the chip. This creates a cavity under the chip. There is connection between this cavity and the flow channel which causes the cavity to be filled with the same gas as flows through the channel after some time. The flow through this this cavity is negligible for other purposes. [3]

2.3 Thermal conductivity sensors

A thermal conductivity sensor is made to measure the thermal conductivity of a material. In this bachelor assignment however only the sensing of thermal conductivities of gases will be considered.

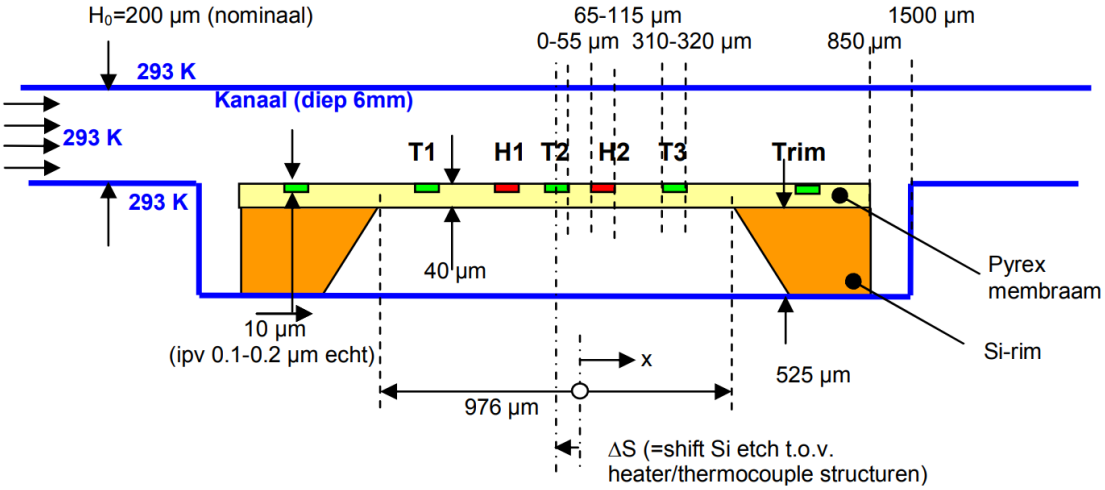


Figure 2.1: A schematic image of the IQ+ flow sensor (not to scale) [3]

This can be done by having two temperature sensors separated by a gas and a heater at on of the temperature sensors. If both the temperature difference and the power supplied heater is known and stable, the thermal conductivity of the gas can be determined.

Another way to determine the thermal conductivity is a thermal conductivity detector (TCD). The thermal conductivity detector measures the heat dissipation through the gas by having a heated element of which the temperature is sensed. This heated element is put into an enclosed channel to not have outside influence and to allow a gas to be injected so it can be measured. By knowing the power provided to and the temperature of the heated element the thermal dissipation can be determined. If the heat heat dissipation is known then the thermal conductivity of the gas can be determined. Usually the thermal conductivity detector has gas flowing through it, to supply the gas to be measured, this flow can also dissipate heat by means of convection. To compensate for the heat dissipation caused by flow two identical channels with heated elements are used. Trough on channel a carrier gas carrying the gas to be measured flows. Through the other channel only the carrier gas flows. The gases are supplied with the same flow rate through both channels. This way the channel with the gas to be measured can be referenced to the channel with only the carrier gas. Doing this the flow rate and thermal conductivity of the carrier gas is being compensated for and the thermal conductivity of the gas which should be measured can be measured. [4]

2.4 The IQ+ Flow Sensor as Thermal Conductivity Sensor

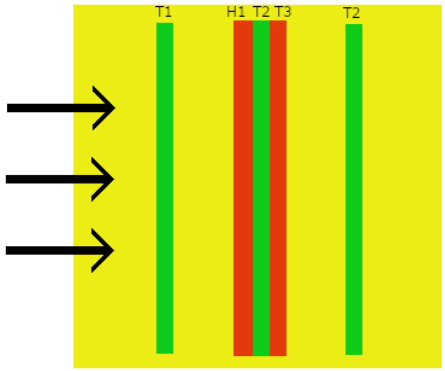
The goal of this bachelor thesis is to find whether the IQ+ flow sensor can be used as a thermal conductivity sensor. The sensor has to be able to have a gas flowing through it and sense the thermal conductivity, so it must be able to determine the thermal conductivity independent of the flow. To do this three setups may be considered: The original chip with no modification, the chip rotated 90 degrees in the flow channel and a series arrangement with the chip in its original orientation followed by the next chip rotated 90 degrees. These scenarios can be seen in figure 2.2.

In an ideal scenario a signal that is independent of flow and can be used to determine the thermal conductivity can be found. In this case only some readout electronics changes may need to be made. This may allow to, with some changes in electronics, use an IQ+ flow module as both a flow sensor and thermal conductivity sensor. This scenario is shown in figure 2.2a.

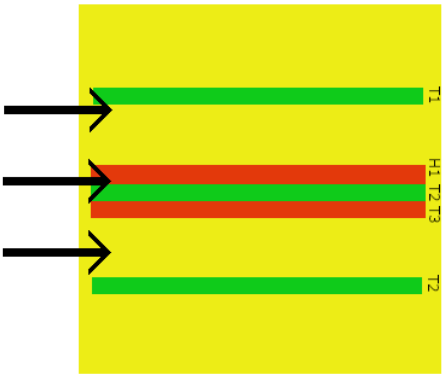
So the chip may be rotated 90 degrees. This way there is no flow induced difference between T1 and T3. This makes sensing the thermal conductivity less complex as there is no longer a difference between T1 and T3 which has to compensated for. This way it is hoped that a flow independent signal that can be

used to determine the thermal conductivity can be found. If this can be found a modified IQ+ module can be used as a thermal conductivity sensor. This scenario is shown in figure 2.2b.

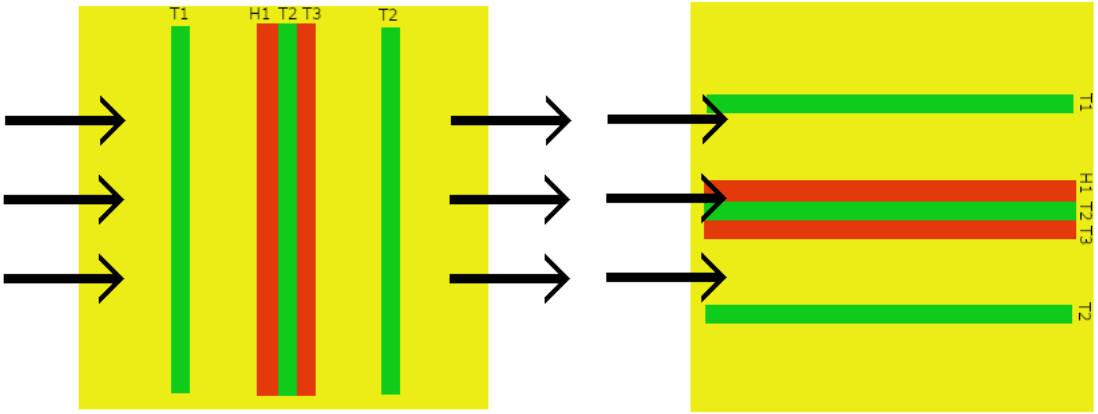
Putting two chips in series may be done when a signal is found on the 90 degrees rotated chip which can be used to determine the thermal conductivity but is dependent on the mass flow of the gas. In this case a chip in its original orientation may be placed before or after the 90 degrees rotated chip. Signals from the originally oriented chip may be used to compensate for the flow dependency of the signal from the 90 degrees rotated chip. When the thermal conductivity is known this can even be used to get a flow measurement from the originally rotated chip. If this configuration works an unmodified chip and a modified chip in series can be used to determine the thermal conductivity and mass flow rate. This scenario is shown in figure 2.2c.



(a) The chip in original configuration, the black arrows represent the flow direction of the gas



(b) The chip in rotated 90 degrees, the black arrows represent the flow direction of the gas



(c) The chip in original configuration and the chip rotated 90 degrees in series, the black arrows represent the flow direction of the gas

Figure 2.2: Possible configurations of the IQ+ flow sensor chips to measure the thermal conductivity (schematic and not to scale).

Chapter 3

Modelling the IQ+ Flow Sensor

3.1 Introduction

To determine whether the IQ+ flow sensor can be used as a thermal conductivity sensor, the IQ+ flow sensor is first modelled. The modelling starts at 0 g/h mass flow and constant power supplied to the heaters. This makes the temperature dependent on the heat conduction through the chip and gas. This conduction is first modelled in a 2d model, refined and later expanded to a 3d model to refine it further. When this 3d model is completed the effect of the mass flow will be modeled. All this modelling will first model the chip as flow sensor as Velthuis has done.[3] All these models will be validated using Velthuis measurements as this is the most complete data that can be used to validate the models. When this is done an extra model will be made in which the direction of flow is rotated 90 degrees.

To do this the same dimensions of the chip will be used as Velthuis so the 2500 by 2500 μm square will be trimmed at the edges to a 1700 by 1700 μm square. This loses some silicon of the original chip and thus some heat conduction at the dges of the chip. As later will be seen the heat conduction of the silicon rim is high enough to be like a short and the there will not be a significant heat difference between the edges of the chip and the environment temperature.

The eventual complete models (The model with the original direction of flow and the one with the 90 degree rotated flow) will be used to find a theoretical expression for the the thermal conductivity of the gas that flows trough the chip. This expression should only use signals from the chip so it can practically be used to measure the thermal conductivity of a gas.

3.2 Modelling Heat Conduction

For the initial models heat conduction will fully define the behaviour of the chip. After convection is introduced to the model the convection adds to the behaviour already defined by the heat conduction throughout the chip. The heat conduction throughout the chip is modelled by an equivalent electrical circuit. This was inspired by the modelling of the V-Grooved Pirani sensor by van Baar in his Thesis [5].

The heat conduction throughout the chip can be modelled as an electrical equivalent conductance as heat conduction is mathematically similar to electrical conduction. The equation for heat conduction can be seen in equation 3.1. [6] where Q is the amount of heat conducted, A is the area it is being conducted through and k is the material that conducts the heat. x is the path along which the heat is conducted.

$$\frac{Q}{\Delta t} = -kA\frac{dT}{dx} \tag{3.1}$$

By defining ΔT to be the heat difference between two points and in the opposite direction of dT. Taking the length L which is the distance between these two points and A the area between these points. Then by settig k as the thermal conductivity constant of the material between these points we can write equation 3.1 as equation 3.2 where $\dot{Q}$ is the heat flux.

$$\dot{Q} = \frac{kA}{L}\Delta T \tag{3.2}$$

This equation 3.2 can be rewritten as equation 3.3.

$$\Delta T = \frac{L}{kA}\dot{Q} \tag{3.3}$$

Which is very similar to Ohms law which can be seen in equation 3.4.

$$V = RI \tag{3.4}$$

This means the equivalent resistance of the thermal resistance between two points would be described by equation 3.5. Where the heat flux would be considered as the current and the temperature difference as the voltage.

$$R = \frac{L}{kA} \tag{3.5}$$

All the models will be divided in to points in space of the chip and the flow channel. The equivalent resistance as can be seen in equation 3.5 will be used to model the thermal conductances between these points.

3.3 2D Model

The first model that is made is the two dimensional model. This model can be seen in figure 3.1. This model has 9 nodes (points) and a reference node. The 9 nodes are the top and bottom walls of the flow channel, the left Trim (as seen in the figure), T1, H1, T2, H2, T3, the right Trim (as seen in the figure). The location of these points is taken exactly in the middle of where the Trim, T1, H1, T2, H2, T3 are positioned on the chip, as all these take up up area on the actual chip. Between the walls of the flow channel and the ground a voltage source is placed to force the wall nodes to be 293 K since it is assumed that the wall nodes have the environment temperature which is set at 293 K. Between the top wall and the nodes at the membrane (Trim Left, T1, H1, T2, H2, T3, Trim Right) there are resistances modelling the thermal conductance from the membrane to the top wall trough the gas. These resistances are denoted as variable as they are dependant on the thermal conductivity of the gas. The height of the flow channel is taken as length for equation 3.5. For the area the width of the chip is multiplied with the corresponding width of the node. The corresponding width of the node is measured as being the length between the two halfway points to the neighbouring horizontal nodes. Between Trim Left, T1, H1, T2, H2, T3 and Trim Right there are the resistances that model the membrane. To determine these resistances the length between these points is filled in to equation 3.5, together with the area of the membrane between these points which is the thickness of the membrane multiplied by the width of the chip. Between the nodes at the membrane and the bottom wall there are resistances modelling the thermal conductance of the silicon rim. The resistance from Trim (on both left and right side) to the walls is different from the other nodes on the membrane in its connection to the walls. They have a direct connection with the walls through the silicon and are modeled by taking the distance between the bottom wall and the membrane as length and the average thickness of the silicon times the width of the chip as area. The resistances between the other points of the membrane are harder to model. They exist of two thermal resistances in series the first being the thermal resistance from the middle of the membrane to the rim of the membrane and the second resistance being from the rim of the membrane to the bottom wall. The resistances of the membrane are taken by taking the distance between the middle of the membrane to the rim as length and the thickness of the membrane multiplied with the corresponding node width as area. The resistance of the silicon rim are taken by taking the height of the rim as length and the average thickness of silicon rim multiplied by the corresponding node width (of T1 etc) as area. When these resistances are modeled only one group of resistances is left. The resistances modeling the gas between the membrane and the bottom wall. There are only resistances between T1, H1, T2, H2 and T3 and the bottom wall as there is no cavity beneath the rim of the chip. These resistors are modeled by taking the height of the silicon rim as length and the corresponding width of the node (of T1 etc) multiplied with the width of the cavity as area. As last part two current sources are added, one for each heater. both current sources are set at the same power and are set at a combined power of 26mW as this corresponds to the power density that was used by Han Velthuis in his simulations [3]. All thermal conductivity constants used in equation 3.5 are the conductivities of the material that they model.

All material properties used in the modelling in this thesis can be found in the appendix in section 7.1.

The 2d model is further refined after this by taking the resistances of the gas in the flow channel and splitting them in two resistances in series between each point at the membrane and the top wall. They are split evenly to create nodes in the middle of the flow channel. These nodes are connected with resistances between each other to model the horizontal thermal conduction of the gas in the flow channel. They are modeled by taking the distance between the points as length and the area between each other

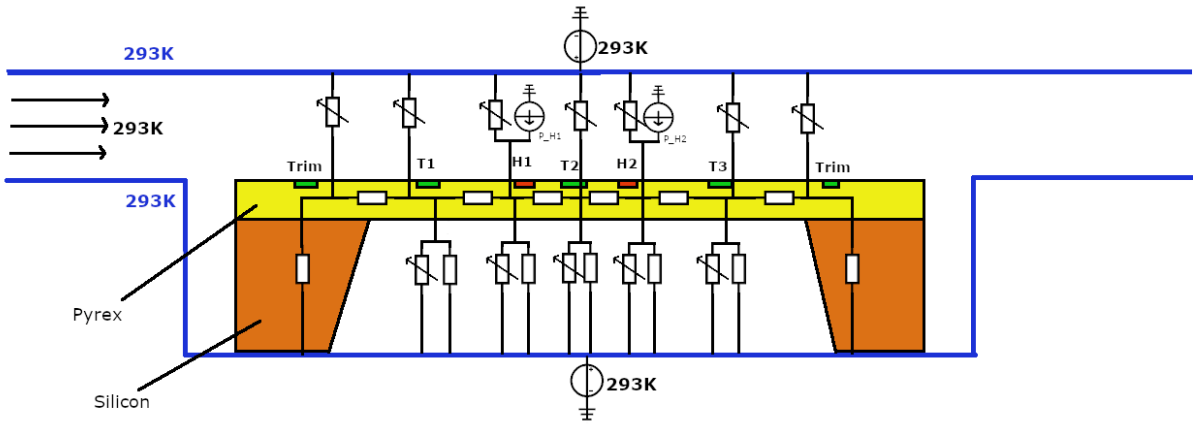


Figure 3.1: The 2d model of the IQ+ flow sensor

as area and plugging this into equation 3.5 together with the thermal conductivity of the gas to calculate the resistance.

3.4 3D Model

To make a more accurate model the chip is now modelled in three dimensions. This allows the addition of the horizontal thermal conduction of the gas in the cavity and the horizontal thermal conduction of silicon rim to the model. This requires an extra dimension as splitting the resistances in two does not create the same nodes for the gas and the silicon rim. The silicon resistances do not connect to the same nodes as the gas resistances. This creates an extra dimension. However it was decided not to stop there but make a full 3D interpratation of the chip.

To make this full 3D interpratation first 7 nodes will be defined as the main nodes. These main nodes form the structure of the 3D model. These nodes are Trim Left, T1, H1, T2, H2, T3 and Trim Right. The nodes are the same nodes as their namesake in the 2D model. They lie when looking into the depth of the chip (The dimension orthogonal to the paper in figure 3.1 and 2.1, this dimension will from now on be called depth) in the middle of the chip. Then along to the depth we will also have the same nodes shifted in to the paper to lie on the middle of the rim. These are called the [main nodes]_in. As the chip is symmetrical there will also be the same nodes shifted out alongside the depth to lie on the middle of the rim. These are called [main nodes]_out. Then the main nodes and their in and out variants also have their variants but shifted up to halfway the flow channel. These are called [main nodes]_up, [main nodes]_up_in and [main nodes]_up_out. The same thing is done to create a lower dimension. The main, in and out nodes also have a variant shifted down to halfway the height of the silicon rim. These nodes are called [main nodes]_down, [main nodes]_down_in and [main nodes]_down_out.

Now that the nodes are defined the resistances can be calculated. To do this the areas taken for resistances must be defined. The areas for a node in the vertical direction of the middle nodes (in the depth direction) will be defined as the width of the cavity (976 μm) times the distance between the halfway points of its neighbours in the horizontal direction (This will be called node width). At the edges it will be the distance between the edge and the halfway point of the neighboring node as seen from the node at the edge (This will extend to node with). For the other areas in the vertical direction, the width of the cavity will be replaced with the width of the silicon rim. For the areas in the horizontal direction this will be divided into three parts, upper, membrane and down. The membrane areas will be the thickness of the membrane times the cavity width for the middle (in the depth direction) area and times the silicon rim width (at its maximum) for the inner and outer (also in the depth direction) areas. The upper and down areas follow the previously mentioned areas replacing the thickness of the membrane with the height of the flow channel for the upper areas and with the height of the silicon rim for the down areas. For the down areas the width of the silicon rim at its maximum will be replaced with the average width of the silicon rim. The lengths of the resistances will be the distance between the nodes. The thermal conductivity will be taken as the thermal conductivity of the material between

the points. This leads to the complication that at the down level for the resistances between the points in the rim and the points in the cavity. Part of the material between the points is silicon and part is gas. This will be solved by calculating two resistances and adding them up. The resistances have the original area but the length that corresponds with the length of the path that goes through one of the two respective material, then the thermal conductivity constant of that material can be used.

3.5 Convection

Through the flow channel of the chip there may be a mass flow (as the chip is a flow sensor). The influence of a mass flow in the flow channel is modelled by convection.

To model the convection it is assumed that the flow is laminar over the chip and that the chip has a flat surface. It is also assumed that the gas properties are constant and do not change with temperature. Doing this makes it possible to use the formulas given by Y.A. Cengel in his textbook about heat transfer [7]. This gives the equation for het transfer by convection:

$$Q = hA_s\Delta T \tag{3.6}$$

From this equation it can be seen, by using the same method as in section 3.2, that we get a thermal resistance.

$$R = \frac{1}{hA_s} \tag{3.7}$$

To determine h Cengel gives the equation.

$$h = \frac{k}{L}Nu \tag{3.8}$$

Where the Nusselt number is gotten by taking the average Nusselt number over the part of the surface that h is determined for. The Nusselt number of which the average should be taking is also given by Cengel.

$$Nu_x = 0.332Re_x^{0.5}Pr^{\frac{1}{3}} \tag{3.9}$$

As is the Reynolds number.

$$Re_x = \frac{\rho V x}{\mu} \tag{3.10}$$

Where ρ is the density of the gas and μ is the viscosity of the gas.

Using these equations the convection is modeled. This is done by taking the surface as the middle nodes (in height) so the main and in and out nodes. For every one of these surface nodes there will be a resistance to ground (as the gas is assumed to be at room temperature) to model the convection. Then determining the flow direction as either coming from the left of the chip as can be seen in figure 2.1 or rotating the chip 90 degrees so that the flow would now be in the depth direction of the figure 2.1. When the flow is determined the edge from which the flow enters will be considered x = 0 and x will increase as the flow moves along the chip. Using this the Reynolds number and the Nusselt number can be determined. Then for every node the average of the Nusselt number will be taken for every node for the length corresponding with the node in the direction of flow. These lengths are in accordance to the areas taken for every node to determine the resistance of the vertical resistance of the upper gas attached to that node. The lengths are also used as length to calculate h where k is the thermal conductance. the areas used for the resistance calculation are the previously mentioned areas corresponding to the nodes.

Chapter 4
Simulating the IQ+ Flow Sensor

4.1 Introduction

In this chapter the simulations will be discussed. It will go into how the simulations were done and in what they resulted to conclude with using the results of the simulation to find an expression for the thermal conductivity. The results of these simulations will be compared to the simulations carried out by Han Velthuis [3] as these are the most complete reference data available at the time of doing the simulations.

All the simulations were done using Matlab [8]. At first a script was written to determine the resistance values for the 2d model. These resistances were then used to build the 2d model in LTspice [9], after this a DC analysis was run in LTspice. The resulting voltages at the nodes represent the temperatures. This process however is very time consuming as for every change and different setting of gas a certain number of resistance values have to be changed by hand in LTspice. To make the process less time consuming a node voltage solver was programmed in Matlab. This way a resistor network could be generated by a script and then solved by this node voltage solver. This method of first generating a resistor network via a script after which this resistor network is handed to the node voltage solver to be solved is also used for the 3D model and the 3D model with convection. These models are large enough to make building these models in LTspice and inserting the values not a practically viable option. Both models have more than 150 resistors which would need to be put into LTspice.

4.2 Node Voltage Analysis

To be able to both create and solve resistor networks in Matlab a simple circuit solver is made in Matlab. As all models purely contain resistors and independent current sources this can be done by using the node and creating a node admittance matrix. This admittance matrix has the form.[10]

$$\begin{bmatrix} Y_{11} & Y_{12} & ... & Y_{1,n-1} \\ Y_{21} & Y_{22} & ... & Y_{2,n-1} \\ : & : & ... & : \\ Y_{n-1,1} & Y_{n-1,2} & ... & Y_{n-1,n-1} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ : \\ V_{n-1} \end{bmatrix} = \begin{bmatrix} i_1 \\ i_2 \\ : \\ i_{n-1} \end{bmatrix} \tag{4.1}$$

Where n is the number of nodes not including the ground node and Y are the admittance values. The indexes of of the admittance values show their relation to the nodes. The admittance values are the negative sum of the admittances between the nodes described in their indexes so for the index 12 it is the negative sum of admittances between the nodes 1 and 2. The diagonal nodes of course have admittances no between their indexes (as they are the same nodes). To calculate these admittance value the positive sum of all the admittances attached to the doubly described node should be taken. The currents of every node are 0 unless a current source is attached to the node. If the current from the current source flows into the node it is added to the node when it flows out of the node it is subtracted. [10]

To transfer this in to Matlab a function was written. This function takes as input the amount of nodes of the network it has to solve, All resistors in an matrix containing for every resistor its resistor value and the two nodes it is attached to and as last input it takes all the current sources also in an array with for every current source its value and the nodes it is attached to. The resistor array is organized as follows Resistor_Array(Number, Value, First node, Second Node). The current source array is organized similarly Current_Source_Array(Source Number, Value, First Node, Second Node) where the current direction is from the second node towards the first node. The numbers are the way to identify individual resistors. When having these inputs the script calculates the admittance values for every entry of the matrix as described in the previous paragraph. It then ads the current sources in the current section as described as also described in the previous paragraph. When the admittance matrix and the current vector are complete they are given as input to the linsolve function in Matlab. The output of this function are is then the vector of voltages. These voltages are the solution of the network and thus given as output. The code can be seen in section 7.2.

4.3 2D Simulation

To simulate the 2D model of section 3.3 a Matlab script was written. This script can be seen in section 7.3. It changes the model as described by removing the voltage sources and setting the ground at the walls of the channel which become one node. This means that the reference for the temperature difference now has to be the environment temperature which is 293 K. First it declares all the chip properties so these can be changed without going through the entire chip. It has the thermal conductance properties of the gases N_2, Ar, CO_2, He, H_2, Kr, Ne and Xe to accommodate convenient switching between these gases. Then it gives every node a number. It generates the resistor network by generating resistors per group. Every resistor has its resistance value calculated as described in the modelling section and is then assigned the nodes it is connected to. This resistor network is given as input to the Node voltage solver. The output of the node voltage solver is then converted to temperature in Kelvin and the relevant nodes are taken and then plotted. The plot for the gas N_2 can be seen in figure 4.1. The power of the chip is set at 26 mW as this corresponds to the power density used by Han Velthuis in his report.

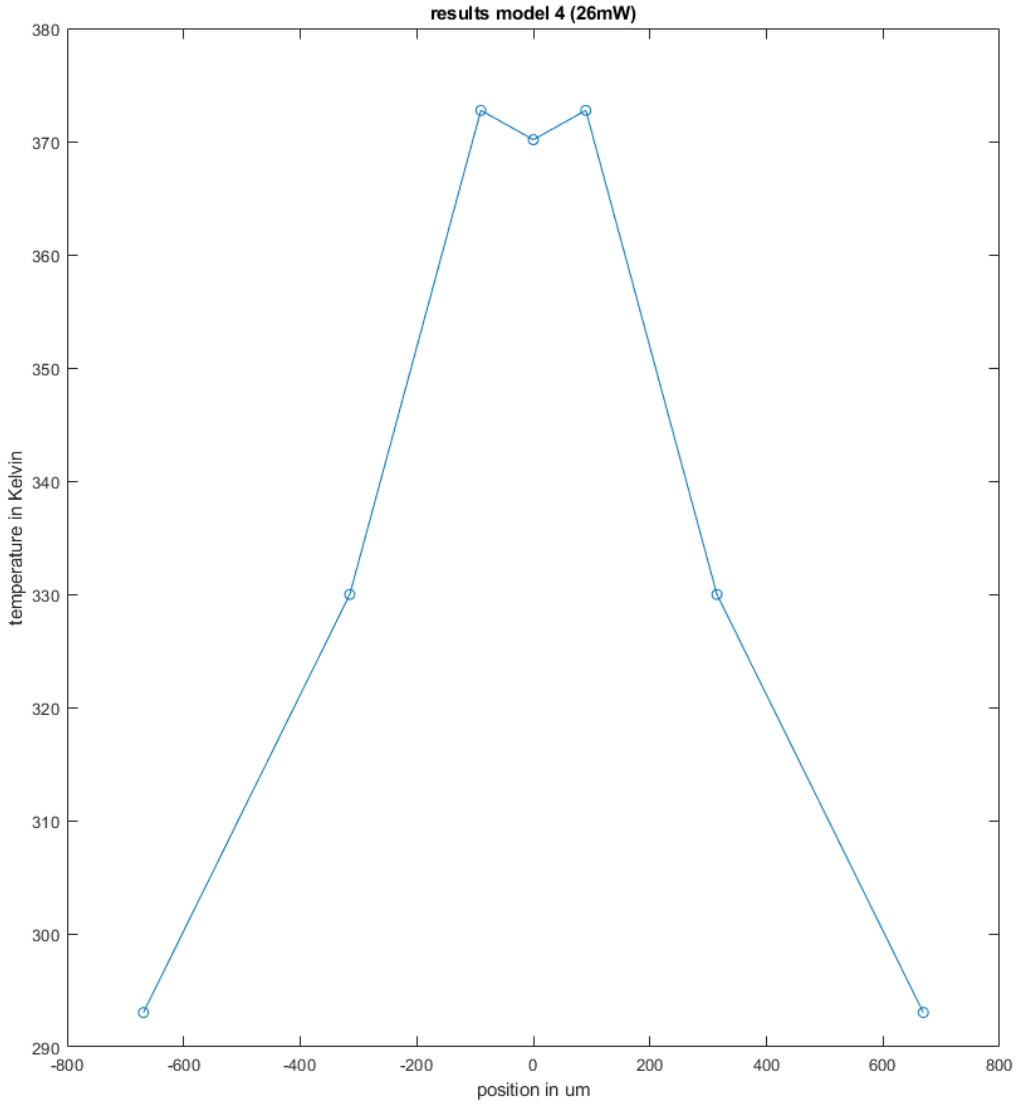


Figure 4.1: The temperature distribution of the 2D model for N_2 at 26 mW heater power, the dots represent the nodes (Trim Left, T1, H1, T2, H2, T3, Trim Right)

The Result of the 2d model which can be seen in figure 4.1 show the temperature distribution over the chip. It can be seen that at the heaters the chip is hottest and it cools down towards the edges which is expected. The temperature of the chip seems to be slightly colder, around 10K, than the results of Velthuis in earlier simulations. As the model will still be expanded to three dimensions this was deemed acceptable.

4.4 3D Simulation

For the 3D model a function was created which did the same as the script for the 2D model with some changes. The function also has the parameters of the chip and declares this however it gets the thermal conductivity as input. It also does not plot the relevant nodes but gives these as output. Then a main function was written which lets the function run for the multiple gases (by setting their thermal conductivity as input of the function) and plots the results. These plots can be seen in figure 4.2 and 4.3.

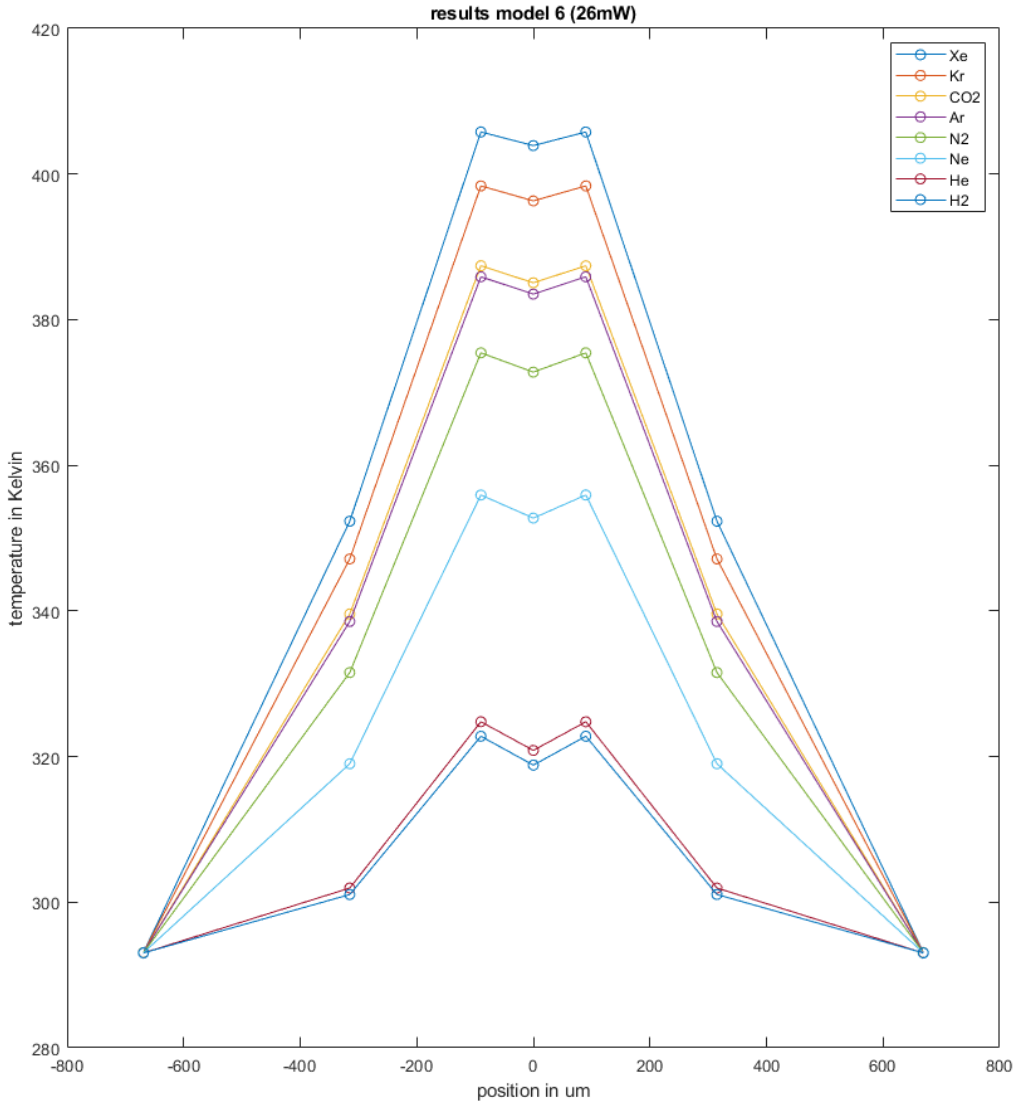


Figure 4.2: Results of the 3D model, The temperature distribution of the chip is shown, the dots represent the nodes (Trim Left, T1, H1, T2, H2, T3, Trim Right), the combined heater power is 26 mW

In figure 4.2 the temperature distribution over the is shown. It can be seen that with the gases that have a higher thermal conductivity the chip runs hotter as less heat is dissipated through the gas. It can also be seen that for the gases with low thermal conductivity the temperature distribution from the heaters to the edge is more linear than for the gases with high thermal conductivity. The linearity is caused by the heat dissipation through the membrane which is a resistor divider along the position of the chip and has a constant resistance per length as its thickness and material properties are constant. The gas however dissipates the most heat where the heat is highest (so in the middle of the chip) this gives rise to the heat conduction of the gas to cause nonlinear behaviour for the temperature distribution of the chip. As the membrane remains constant in the chip the heat conduction through the gas can make the heat distribution from heater to edge either more or less linear. Where higher thermal conductivity of the gas causes more heat conduction through the gas and thus less linearity.

In figure 4.3 The temperatures of different nodes and expressions of these are shown.It can be seen that most nodes decrease in temperature depending on the thermal conductivity. It can also be seen that when looking at the ratio between (T2 - Trim)/(T1 - Trim) it is linearly dependent on the thermal conductivity. This can be explained by the fact that the non linearity of the temperature distribution is caused by the heat conduction of the gas which increases linear with the thermal conductivity. So looking at the ratio of T2/T4 referenced to the edge of the chip (which tends to be around the environment temperature) which shows the linearity or non linearity of the temperature distribution gives a result dependent on the thermal conductivity.

4.5 3D Simulation with Convection

To simulate the 3D model with convection two functions were used. One function having the flow as shown in figure 2.1 and on with the flow rotated 90 degrees. Both functions are the function of the 3D model with convention added to it. For the convection to be added the function needs extra gas properties as input. It needs as extra properties the density of the gas, the viscosity of the gas, the Prandtl number of the gas and it needs the flow velocity of the gas. Using these parameters the resistances modelling the convection can be added as described in section 3.5. These functions are called in the main function for the gases Xe, Kr, CO_2, Ar, N_2, Ne, He and H_2. These gases are each run for a range of flow velocities of 0 to 30 m/s with 0.1 m/s increments, each gas is subjected to the entire range of velocities. This data is then saved to a file. Running both functions 8 x 301 = 2408 times when changing some plot settings or doing analysis on the data is impractical. A plotting function then reads the data from the file and plots it. The plots can be seen in figure 4.4 and 4.5. In these plots a point T4 can be seen. This isn't a real point, it is the average of T1 and T3 so $T4 = \frac{T1+T3}{2}$. It most also be noted that to simulate the real signals the chip gives, all signals are referenced to Trim. This is done because the real chip does this. Trim is in every case around between 293K and 294K as it is almost thermally shorted to the flow channel walls which are 293K. To compare it to the other measurements 293K must thus be added to the signals.

In figure 4.4 the results with the flow in the original direction are shown. When comparing this to the model of Velthuis T2 has similar values (370 K at 0 m/s and 340 K at 30 m/s for N_2 and having added 293K to the model to compensate for the reference point). So the total heat dissipation of the flow seems to have been modelled correctly. It can be seen that T3-T1 however has a difference with the model of Velthuis of a factor of around 4 but still has the same behaviour. This indicates that modelling the flow purely by convection is not sufficient to model the local temperature differences.

When looking at figure 4.5 which has the flow rotated 90 degrees it became clear that with the exception of T1-T3 and what dependant on this the results where exactly the same. T4 has the exact same value as the previous model. As T1 and T3 are more than half of the length of the flow length over the chip this is more likely to depend on the total dissipation trough convection and thus being modelled correctly. This would indicate that T4 in the previous model was also correct and only the distribution of temperature over T3 and T1 was incorrect.

It can be seen that T2/T4 (which is already referenced to the edge of the chip in this model) gives a again its linear relation with the thermal conductivity. The flow however also has an influence on this. It works in the same way as the thermal conduction does for the gas however its dissipation increases in

a square root form with the flow.

4.6 Thermal Conductivity Sensing

The 3D model with convection will be used to find a way to determine the the thermal conductivity of the gas using only the temperature signals of the thermal sensors. All signals from the thermal piles in the model are referenced to Trim to simulate the actual signals the chip will put out.

A starting point to determine would be taking T4/T2 as this is already has a linear dependence on the thermal conductivity. However just linearly fitting this signal and creating a conversion formula using this fit will be dependent on flow since T4/T2 is dependent on flow. So a correction for flow must be made. To do this T3-T1/T2 is taken to represent flow (and will be called F from now on). This function is the function that is taken as signal for the flow calculations by Han Velthuis and is a square root dependent on flow. Simply multiplying this signal with T2/T4 however does not yield the hoped result of a mostly flow independent signal that is dependent on the thermal conductivity. This is caused by the fact that the signal is dependent on the membrane, the gas and the flow. So (T2/T4)(1-aF) was tried where a is just a constant that can be set. The 1 takes care of the signal caused by the membrane and gas while a times F adjusts for the influence of flow on the signal. As both the signal (T2/T4) and F are square root dependent on the flow F can be plugged in without correcting for the square root dependence. By trial and error the factor 5 was taken for a. This made the function mostly independent of flow. To determine the thermal conductivity using this signal the known thermal conductivity of the gasses was linearly fitted to the signal. Then using the fit the estimated thermal conductivity was calculated from the function.

A first method to determine the thermal conductivity was created with trial and error to determine its constant a. It was decided to determine a in a more precise way. The function (T2/T4)(1-aF) was taken again. and k was fitted again to this function and using this fit an estimated k value for the signal (T2/T4)(1-aF) was made. This time however a was varied from 0.01 to 20 with steps of 0.01. Then the sum of the error squared where the error is the difference between the estimated and actual thermal conductivity was taken. The value of a for which this sum was lowest is considered the best value for a which was 5.01 which is almost exactly the same as the outcome of the trial and error. This signal and the estimated thermal conductivities can be seen in figure 4.6. It must be noted that since T3-T1 differs considerably from the values of Velthuis that the factor a will probably differ from the actual factor that should be used with a real flow sensor.

To determine that $(T2/T4)(1-aF)$ is the correct form to use, some other forms where tested. These forms where $(T2/T4)(1-a\sqrt{(}F))$, $(T2/T4)(1-aF-bF^2)$ and $(T2/T4)(1-aF-b\sqrt{(}F))$. The parameters of these functions where found in the same way as a was determined in the previous paragraph. The lowest sum of square errors was also taken for each signal and compared to the sum of square errors of $(T2/T4)(1-aF)$. No improvement was found. The signals $(T2/T4)(1-aF-bF^2)$ and $(T2/T4)(1-aF-b\sqrt{(}F))$ got equal performance to $(T2/T4)(1-aF)$ while $(T2/T4)(1-a\sqrt{(}F))$ had worse performance (higher sum of square errors). This means that $(T2/T4)(1-aF)$ is the best signal of these tested signals as all signals getting equal performance are more complicated and the other signal performed worse.

The magnitude of the error of the estimated thermal conductivity (using the best fitted signal: $(T2/T4)(1-5.01F)$) was plotted as percentage of the correct value for the thermal conductivity in figure 4.6. It can be seen that for low mass flow and small thermal conductivities the error is considerable. As the best fit signal was taken over the entire range of flows the extreme ends are expected to show more error. Since the influence of the flow follows a square root relation this make the difference in the low flows the largest resulting in more error for the low flows. This could be improved by making a special fit for low flows or having logarithmic steps for the flow when determining the fit. The larger error for low thermal conductivities is caused by the fact that the same error in thermal conductivity is a larger relative error in thermal conductivity.

The simulations indicate that the unmodified chip can be used to determine the thermal conductivity independent of flow. Using the signal $(T2/T4)(1-5.01(\frac{T3-T1}{T2}))$ and linearly fitting the thermal conductivity to this a function for the thermal conductivity can be created that is independent of flow.

Measurements will have to be done to test whether this holds in reality. As the difference between T3-T1 is most likely incorrect, the value 5.01 will most likely have to be adjusted based on the measurements. Interesting in measurements would be whether the temperature T4 is independent of the mass flow direction (at a 90 degree angle). If T4 is independent of flow direction one flow sensor module with the flow in the original direction can be used as thermal conductivity sensor. If T4 is in measurements not independent of flow direction it might be necessary to have an unmodified flow sensor in series with a 90 degrees rotated chip.

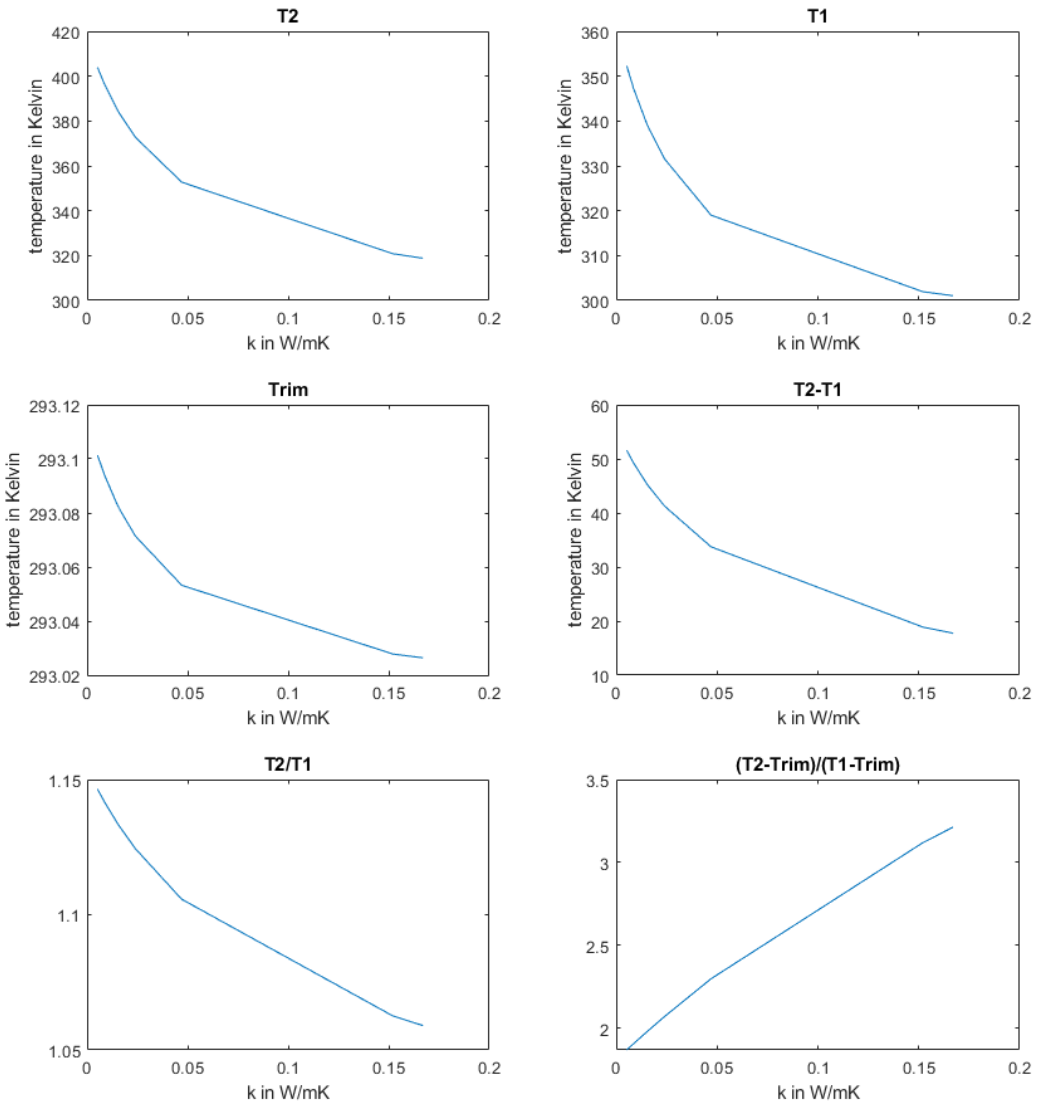


Figure 4.3: Results of the 3D model, the temperatures and functions of these temperatures are shown against the thermal conductivity, the combined heater power is 26 mW

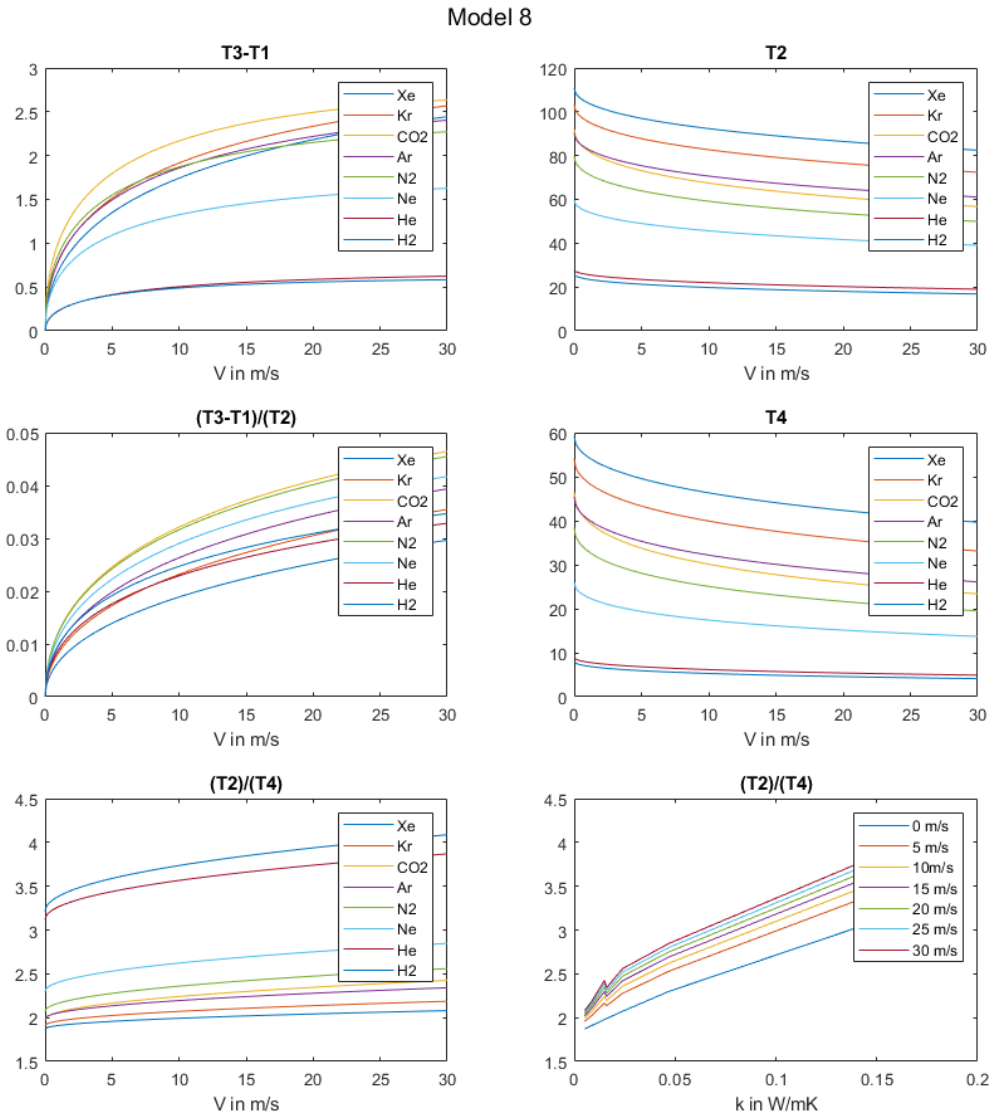


Figure 4.4: The results from the 3D model with convection with flow from the original direction as can be seen in figure 2.1, Heater power at 26mW

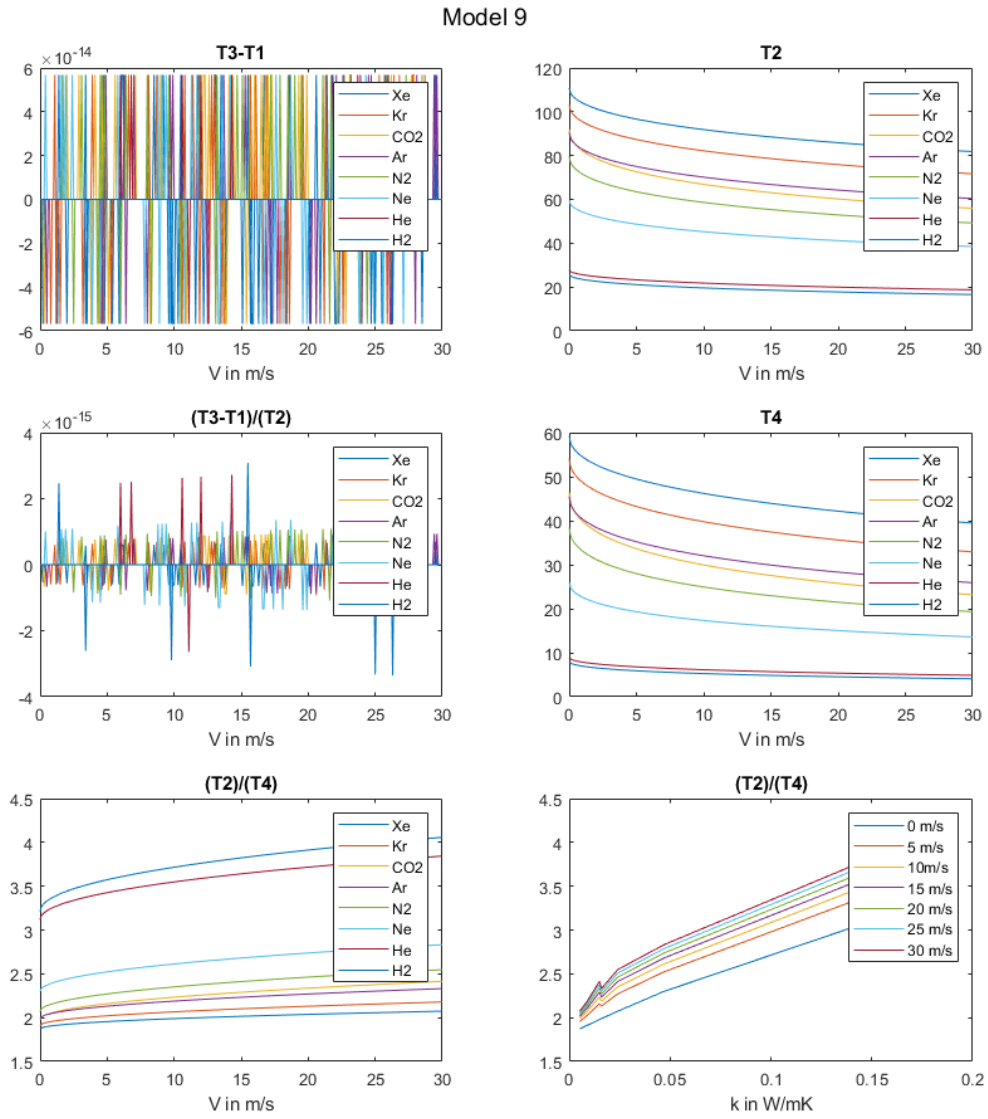


Figure 4.5: The results from the 3D model with convection with the flow direction rotated 90 degrees from the original flow direction. Heater power at 26mW

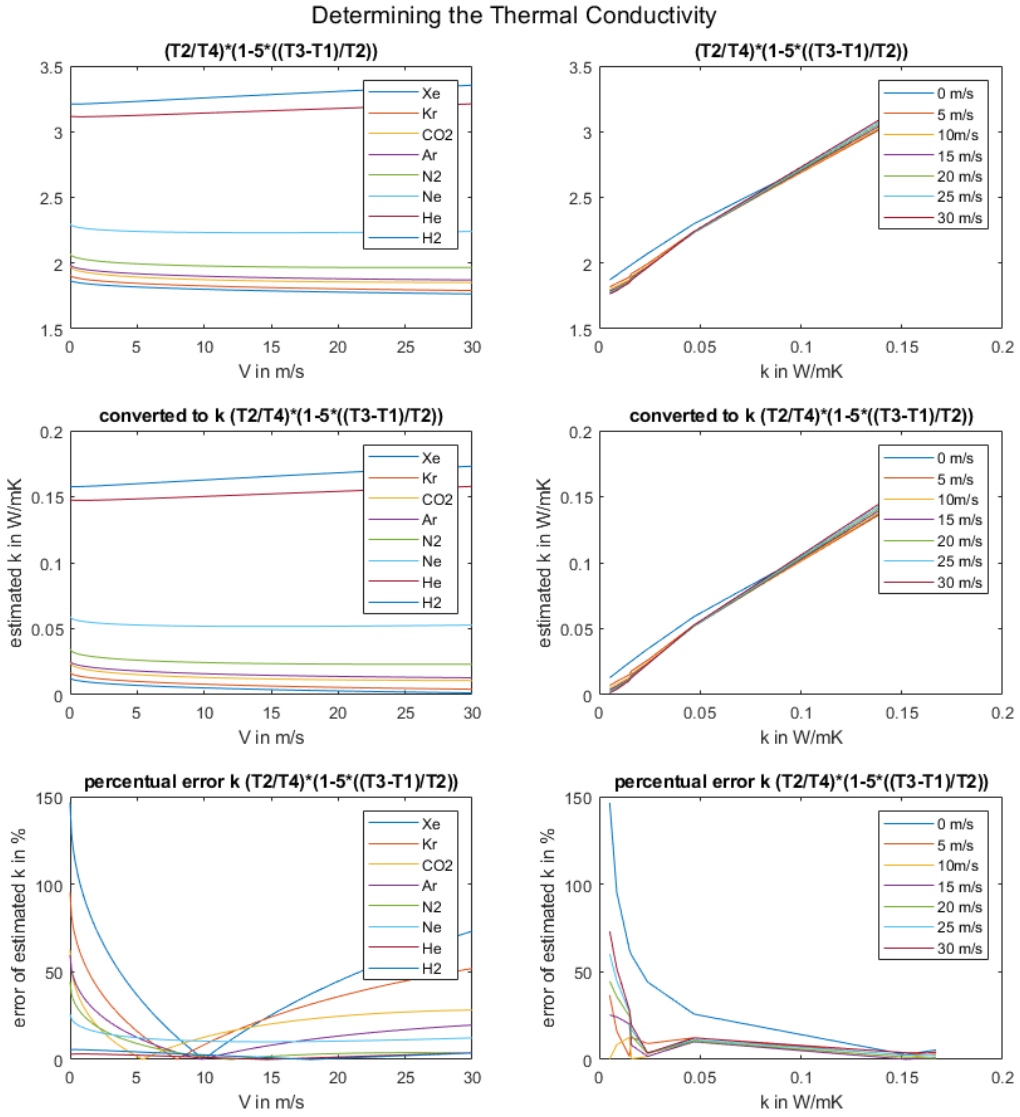


Figure 4.6: The signal used to determine the thermal conductivity

Chapter 5

Measurements

There were measurements done to test how accurate the models are. These measurements are not enough to fully test the models as a standard IQ+flow module with its standard read-out electronics was used. The standard readout electronics only read out the temperature at T2 referenced to Trim and the temperature difference between T1 and T3. The other data the IQ+ module gives is its measurement which will not be that useful as we are interested in the temperatures of the chip.

5.1 Setup

Three gases were taken to be used for the measurements: N_2, Ar and He. These gases were put through a Coriolis flow controller. This flow controller controls the flow and this flow is fed to the IQ+ flow sensor. The IQ+ flow sensors output is left open so the gas flows into the measurement lab. So it is the gas supply supplies the Coriolis flow controller which supplies the IQ+ sensor which outputs the gas into the environment. A thermal sensor is placed on the base plate of the IQ+ module to measure the absolute temperature. This temperature should be close to Rim temperature of the chip to which all the temperature signals from the IQ+ sensor are referenced. The Coriolis flow controller and the IQ+ flow sensor were controlled from a script in python. This script first flushes the the setup 2 minutes with the gas the measurements will be taken with to get rid of any residue gas of previous measurements with the setup. Then in steps a range of mass flows is measured. Starting at 0 g/h to the maximum maximum mass flow that the Coriolis mass flow sensor can reach. The measurement data is saved by this script and later loaded in Matlab for analysis.

5.2 Results

The measurement results were loaded in Matlab so some analysis could be done. The mass flow measurement of the Coriolis flow controller was considered to be the mass flow. For comparison the flow velocity in m/s was calculated from the mass flow. The first seconds of the N_2 and Ar measurements still showed effects from the flushing of the chip so the first 5 seconds of these measurements were removed. The results were then plotted as can be seen in figure 5.1, 5.2 and 5.3. The theoretical values of T2 and T3-T1 were plotted against the same flow range as the measured values to make comparison easier, these plots can be seen in figure 5.4.

5.3 Discussion

It can be seen that the measured temperature of T2 at 0 m/s flow is very similar to the simulation for both N_2 (84 K vs 80 K)(measurement vs simulated) and Ar (90 K vs 90 K) but not for He (41 K (first value in time was taken) vs 28K). This could be caused by helium warming up the top wall of the flow channel causing less heat conduction. When looking at the temperature difference at maximum flow to zero flow this is for N_2(6 K vs 11), Ar(6 K vs 12 K) and He(5 K vs 5 K). It can be seen that the predicted temperature decrease caused by convection is off by a factor 2 for both N_2 and Ar but is correct for He. When looking at T3-T1 it can be seen that for N_2(2.75 K vs 1.25 K) and Ar(2.5 K vs 1.25 K) the theoretical prediction was half the measured value however He(3 K vs 0.5 K) has the simulated prediction being 6 times lower than the actual measured value. Helium is in all these cases the outlier, as only three gases where tested more gases should be tested to see whether this is helium is truly an outlier or that the model is by coincidence off by the same factors for Ar and N2. When looking at Ar and N_2 the heat dissipation dissipation through convection is two times higher in the simulations than it should be. The difference in temperature of T1 and T3 however is two times lower in simulations than it should be.

As T1 and T3 could not be read out by the read-out electronics $T2/T4$ and $(T2/T4)(1 - aF)$ could not be tested. Instead T2 was plotted against the thermal conductivity as can be seen in figure 5.5. Both signals show the same type of behaviour (logarithmic decay) but with the simulated values having

a greater response to the change in thermal conductivity. It was tried to fit the thermal conductivity to two measured values of T2 (at 0 m/s flow) using an exponential function $a \cdot e^{b \cdot k}$ (where k is the thermal conductivity and a and b are fit parameters). The values of argon and helium were taken, as they have the largest difference of temperature and thermal conductivity, to minimize the influence of noise on the fit. Then this fit was used to estimate the thermal conductivity value of N_2. The estimated thermal conductivity value was 0.0212 W/mK where the known thermal conductivity value of N_2 is 0.024 W/mK. This is an error of 11.5% between the estimated and known value of N2. When having more data with which such a fit can be made the influence of noise on the fit should decrease and thus the error should also decrease. There is another factor which also contributes to the error. The known thermal conductivity values are valid at 293 K, the temperature of the gases in the chip are considerably higher (up to around 90 K higher) this affects the thermal conductivity of the gases and to get the error down the thermal conductivity of the gases should be taken as function of temperature.

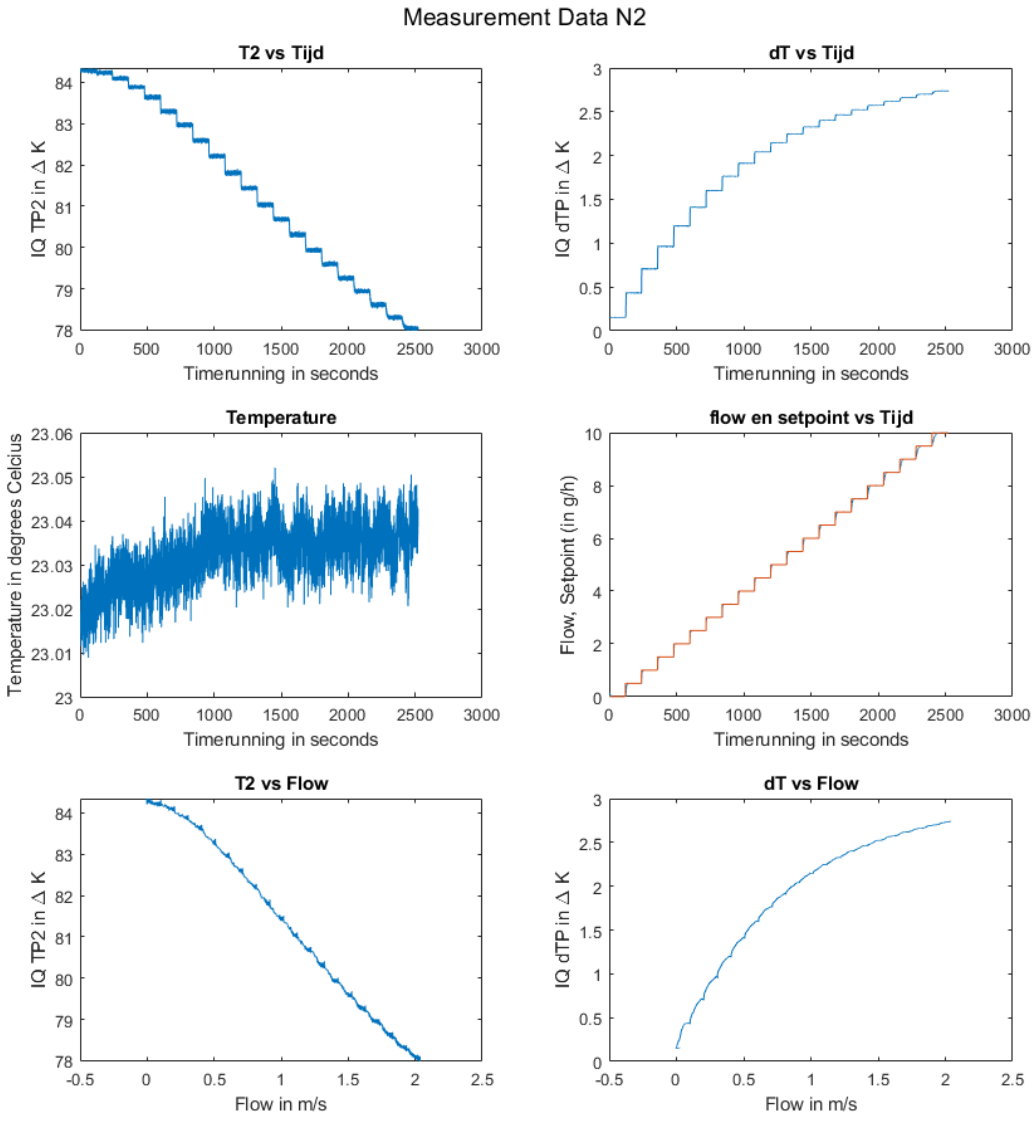


Figure 5.1: The measurement results of the gas N_2

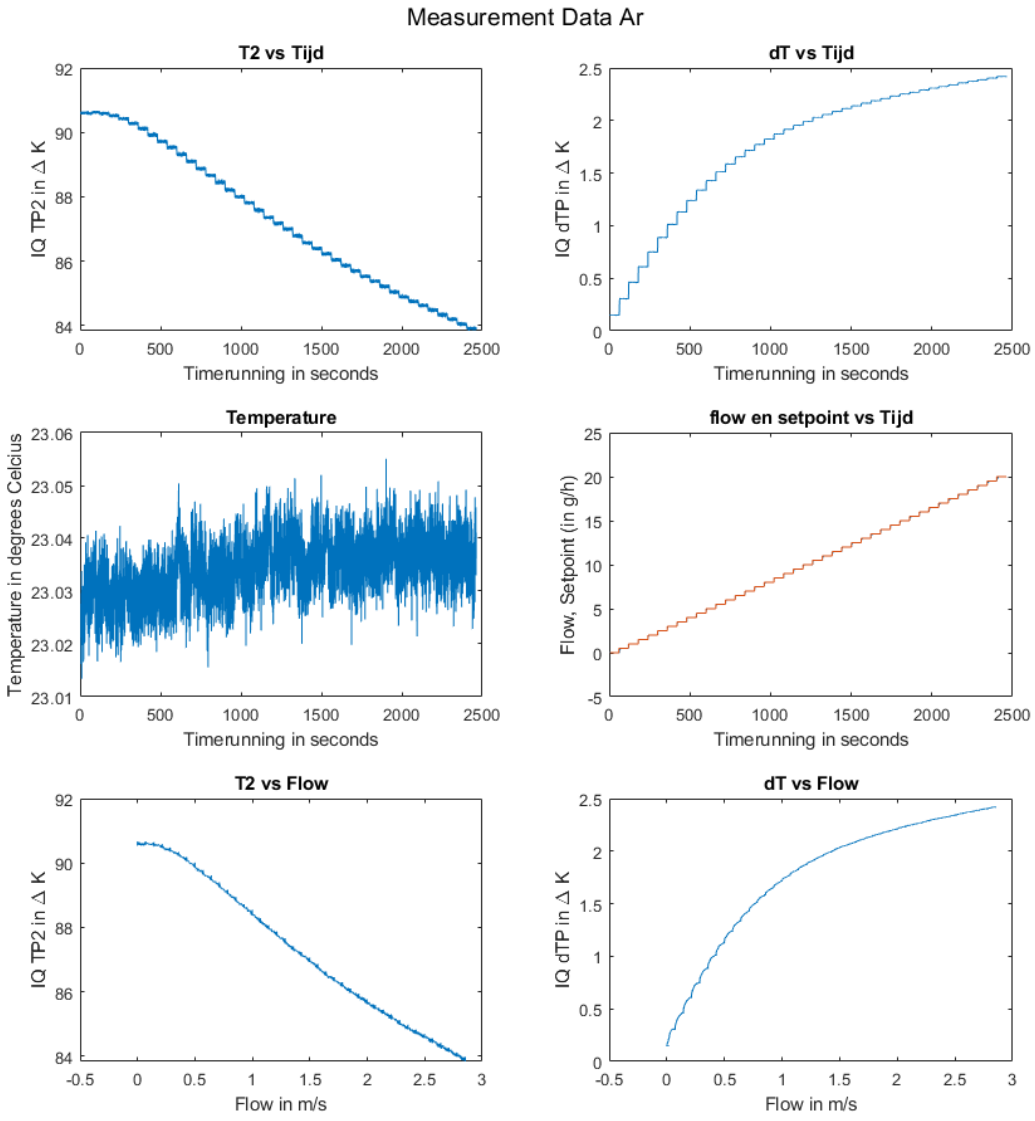


Figure 5.2: The measurement results of the gas Ar

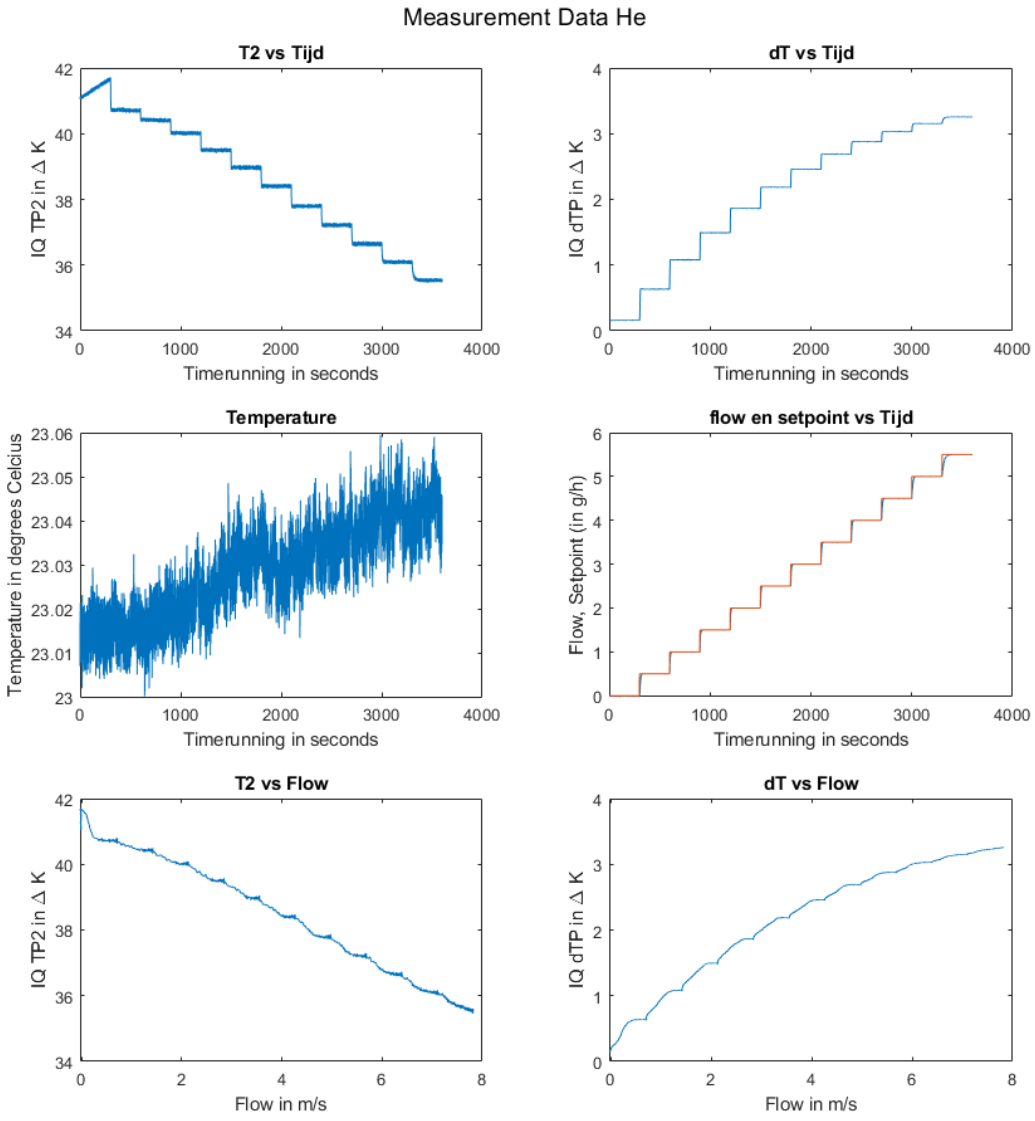


Figure 5.3: The measurement results of the gas He

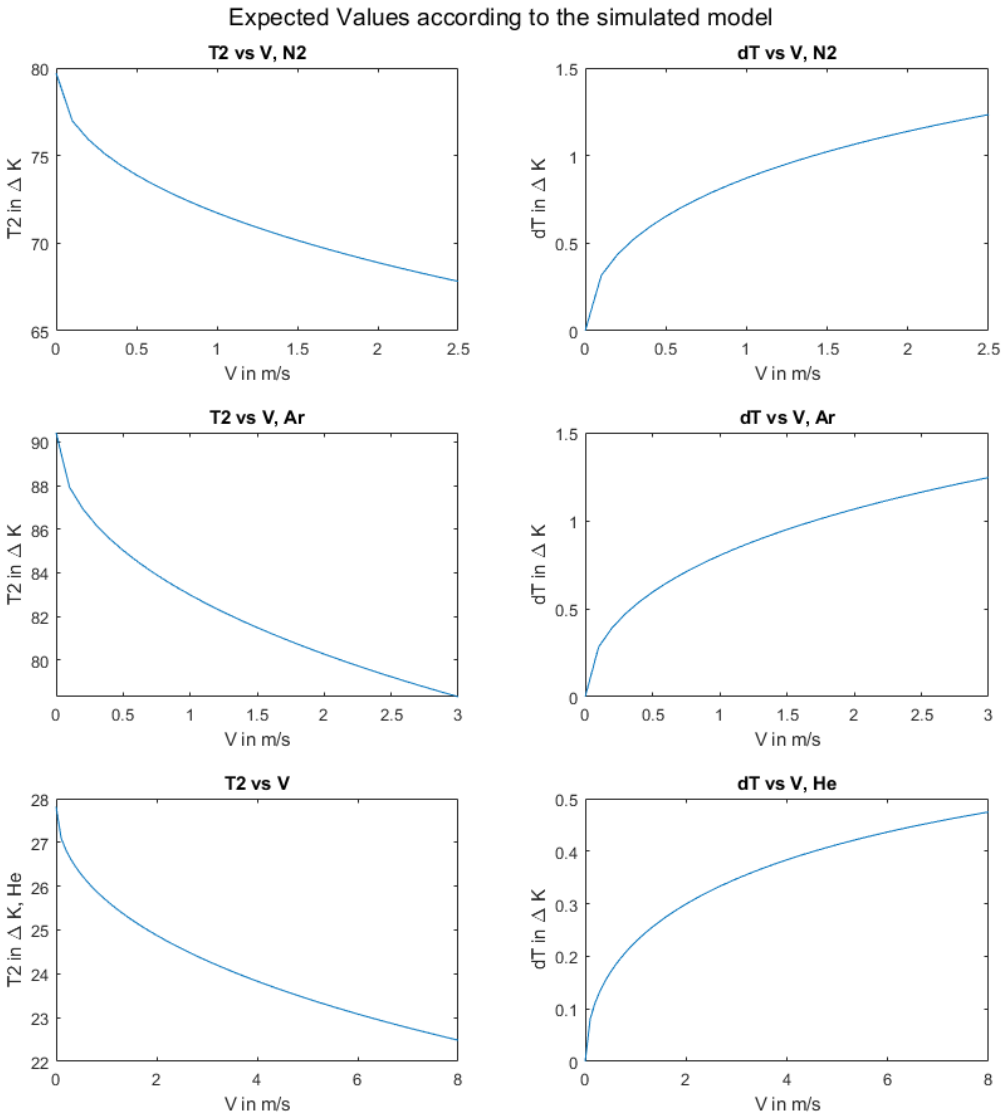


Figure 5.4: The results of the simulation for comparison, dT is T3-T1

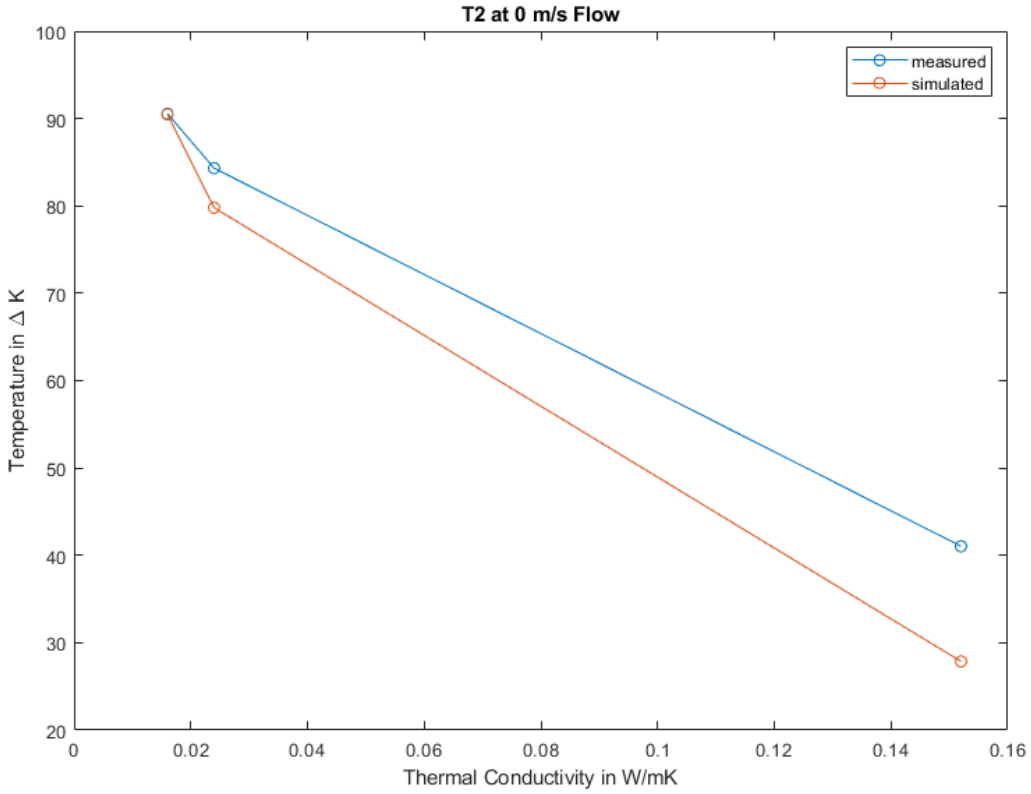


Figure 5.5: The temperature of T2 at zero flow for measured and simulated values plotted against the thermal conductivity

Chapter 6
Conclusion

The goal of this project was to find whether an IQ+ flow sensor could be used as a thermal conductivity sensor. Where the thermal conductivity sensor is independent of flow.

When looking at the results of the simulation the thermal conductivity can be determined independent of flow. This can be done with a IQ+ Flow module with only the readout electronics changed. This is the ideal case scenario as described in section 2.4. This could be done by taking the ratio of T2/T4 and compensating this for the flow by multiplying this with $(1 - 5.01\frac{T3-T1}{T2})$ to compensate for the heat dissipation caused by the flow resulting in a signal that is linearly dependent on the thermal conductivity but independent of flow. Considering this correct the only changes that would need to be made to the IQ+ module would be adding readout electronics to read out T1 and T3.

There is a disparity between the limited measurement results and the simulated results. This disparity indicates that the signal as described would need at least its parameter 5.01 changed. The type of behaviour for every signal was correct however indicating that with this change of parameter it is possible to use the IQ+ flow sensor as a flow independent thermal conductivity sensor. To determine whether this is the case more extensive measurements should be done.

Bibliography

[1] Bronkhorst, "Iq+flow ultra compact mass flow meters / controllers," 27-06-2020. [Online]. Available: https://www.bronkhorst.com/int/products/gas-flow/iq-flow/

[2] D. F. Reyes Romero, "Development of a medium independent flow measurement technique based on oscillatory thermal excitation," Ph.D. dissertation, Albert-Ludwigs-Unversität Freiburg, July 2014.

[3] H. Velthuis, *Flowconversie IQ+ module.* TNO, March 2009.

[4] A. N. Abarca Prouza, "High precision flow compensated thermal conductivity detector for gas sensing with read-out circuit," Master's thesis, Technische Universiteit Delft, September 2015.

[5] J. J. van Baar, "Distributed thermal micro sensors for fluid flow," Ph.D. dissertation, Universiteit Twente, November 2002.

[6] D. V. Schroeder, *An Introduction to Thermal Physics.* San Francisco, CA: Addison Wesley Longman, 2000.

[7] Y. A. Cengel, *Heat Transfer: A Practical Approach. 2nd Edition.* New York: McGraw Hill, 2002.

[8] MathWorks, "Matlab r2020a," 24-06-2020. [Online]. Available: https://nl.mathworks.com/products/matlab.html

[9] A. Devices, "Ltspice xvii," 24-06-2020. [Online]. Available: https://www.analog.com/en/design-center/design-tools-and-calculators/ltspice-simulator.html

[10] "Node voltage method," 25-06-2020. [Online]. Available: https://inst.eecs.berkeley.edu/ ee100/fa04/lecture_notes/EE100supplementarynoteweek03-2.pdf

Chapter 7

Appendix

7.1 Material Properties

Gas	ρ [kg/m3]	k [W/mK]	μ [Pa·s]	C$_p$[J/Kg]	Pr [-]
Ar	1.623	0.016	2.125E-5	520.64	0.7
CO2	1.788	0.015	1.37E-5	840.37	0.794
He	0.163	0.152	1.99E-5	5193	0.68
H2	0.082	0.167	0.8411E-5	14280	0.719
Kr	3.404	0.0087	2.5E-5	284.14	0.816
Ne	0.82	0.047	3.13E-5	1030	0.694
N2	1.138	0.024	1.663E-5	1041	0.715
Xe	5.333	0.0052	2.28E-5	158.4	0.695

Table 7.1: The physical properties of the gases at environment temperature [3]

Solid	k [W/mK
Borosilicate Glass	1
Si	150

Table 7.2: The Thermal conductivity of the solid materials at environment temperature [3]

7.2 Node Voltage Matlab

```
function nodes = nodal(amount_nodes, resistance, current_source)
    A = zeros(amount_nodes, amount_nodes);

    for i = 1:amount_nodes
        list_1_i = find(resistance(:,1) == i);
        list_2_i = find(resistance(:,2) == i);

        for j = 1:amount_nodes
            list_1_j = find(resistance(:,1) == j);
            list_2_j = find(resistance(:,2) == j);
            cross_list_1 = intersect(list_1_i, list_2_j);
            cross_list_2 = intersect(list_2_i, list_1_j);
            Q = 0;

            for k = 1:length(cross_list_1)
                Q = Q+1/resistance(cross_list_1(k),3);
            end

            for k = 1:length(cross_list_2)
                Q = Q+1/resistance(cross_list_2(k),3);
            end

            if Q ~= 0
            A(i,j) = -Q;
            end
        end
    end

    for i = 1:amount_nodes
        current(i) = 0;
        list_1_current = find(current_source(:,1) == i);
        list_2_current = find(current_source(:,2) == i);
        list_1_resistance = find(resistance(:,1) == i);
        list_2_resistance = find(resistance(:,2) == i);
        for k = 1:length(list_1_current)
            current(i) = current(i) + current_source(list_1_current(k),3);
        end
        for k = 1:length(list_2_current)
            current(i) = current(i) - current_source(list_2_current(k),3);
        end
```

```
        for k = 1:length(list_1_resistance)
            A(i,i) = A(i,i) + 1/resistance(list_1_resistance(k),3);
        end
        for k = 1:length(list_2_resistance)
            A(i,i) = A(i,i) + 1/resistance(list_2_resistance(k),3);
        end
    end

    current = current';

    nodes = linsolve(A,current);

end
```

7.3 2D model Matlab

```
close all;
clear all;

%% chip parameters

Lu = 200*10^-6;             % stroomkanaal in M
Ld = 525*10^-6;             % holte onder chip in M
Bchip = 1*10^-3;            % in meter assumptie van 1mm breed
thicknes_pyrex = 40*10^-6;
Am = Bchip*thicknes_pyrex;  % oppervlakte membraan in M^2
angle  = 35.3*(2*pi/360);
Overstaand = Ld*atan(angle);
Effect_Thickness_Silic = (Overstaand/2 + (850 - 976/2 - Overstaand))*10^-6;  % correcting for the angle of the silicon rim
total_power = 26*10^-3;     % totaal vermogen verstookt door de chip in W

%% thermal conductivities
% all in W/Km

k_N2 = 0.024;
k_Ar = 0.016;
k_CO2 = 0.015;
k_He = 0.152;
k_H2 = 0.167;
k_Kr = 0.0087;
k_Ne = 0.047;
k_Xe = 0.0052;

k_pyrex = 1;
k_silicon = 150;
k_gas = k_N2;

%% lengtes tussen de nodes

Lm1 = 354*10^-6; % in meter
Lm2 = 225*10^-6; % in meter
Lm3 = 90*10^-6; % in meter
Ledge = 181*10^-6;

Lm(1) = Lm1;
Lm(2) = Lm2;
Lm(3) = Lm3;
Lm(4) = Lm3;
Lm(5) = Lm2;
Lm(6) = Lm1;

%% wijdtes bovenplaat gassen
Wu1 = Ledge + Lm1/2;
Wu2 = Lm1/2+Lm2/2;
Wu3 = Lm2/2 + Lm3/2;
Wu4 = Lm3;

Wu(1) = Wu1;
Wu(2) = Wu2;
Wu(3) = Wu3;
Wu(4) = Wu4;
Wu(5) = Wu3;
Wu(6) = Wu2;
Wu(7) = Wu1;

%% nodes

gnd = 0;
TrimL = 1;
T1 = 2;
H1 =3;
T2 = 4;
H2 = 5;
T3 = 6;
```

```
TrimR = 7;
% 8 - 14 trimL_up to TrimR_up

amount_nodes = 14;

%% heaters

I(1, :) = [H1, gnd, total_power/2];
I(2, :) = [H2, gnd, total_power/2];

%% start the resistors

%% resistor counter initialization
n_R = 1;

%% the membrame resitors

for i = 1:6
    R(n_R,:) = [i, i+1, Lm(i)/(k_pyrex*Am)];
    n_R = n_R + 1;
end

%% Resistors Silicium

R(n_R, :) = [1, gnd, Ld/(k_silicon*Bchip*Effect_Thickness_Silic)];
n_R = n_R+1;
for i = 2:6
    Rtemp = Ld/(k_silicon*Wu(i)*Effect_Thickness_Silic)+(Bchip/2)/(k_pyrex*thicknes_pyrex*Wu(i));
    R(n_R, :) = [i, gnd, Rtemp/2];
    n_R = n_R + 1;
end
R(n_R, :) = [7, gnd, Ld/(k_silicon*Bchip*Effect_Thickness_Silic)];
n_R = n_R+1;

%% resistors gas beneden

for i = 1:5
    R(n_R, :) = [i+1, gnd, Ld/(k_gas*Bchip*Wu(i+1))];
    n_R = n_R+1;
end

%% resistors gas boven verticaal

for i = 1:7
    R(n_R, :) = [i, i+7, Lu/(2*k_gas*Bchip*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+7, gnd, Lu/(2*k_gas*Bchip*Wu(i))];
    n_R = n_R+1;
end

%% resistors boven horizontaal

Au = Bchip*Lu;
for i = 1:6
    R(n_R, :) = [i+7, i+8, Lm(i)/(k_gas*Au)];
    n_R = n_R+1;
end

%% nodal

nodes = nodal(amount_nodes, R, I);
temperatures = nodes + 293;

%% plot
x = [-669 -315 -90 0 90 315 669];
T = temperatures(1:7, 1);

figure;
plot (x,T, '-o');
xlabel('position in um');
ylabel('temperature in Kelvin');
title('results model 4 (26mW)');
```

7.4 3D Model Matlab

7.4.1 3D Model Function

```
function y = model_6_fun(k_input)

%% chip parameters

Lu = 200*10^-6;             % stroomkanaal in M
Ld = 525*10^-6;             % holte onder chip in M
```

```
BchipMin = 976*10^-6;       % exclusief rim
BchipMax = 850*2*10^-6;     % inclusief rim
thicknes_pyrex = 40*10^-6;
% Overstaand = Ld*atan(pi/4);
SiRimMax = (850-976/2)*10^-6;
% Effect_Thickness_Silic = Overstaand/2 + (850 - 976/2 - Overstaand);
total_power = 26*10^-3;     % totaal vermogen verstookt door de chip in W

%% thermal conductivities
% all in W/Km

k_pyrex = 1;
k_silicon = 150;
k_gas = k_input;

%% lengtes tussen de nodes

Lm1 = 354*10^-6; % in meter
Lm2 = 225*10^-6; % in meter
Lm3 = 90*10^-6; % in meter
Ledge = 181*10^-6;
%Bchip = 2*(Lm1+Lm2+Lm3);

Lm(1) = Lm1;
Lm(2) = Lm2;
Lm(3) = Lm3;
Lm(4) = Lm3;
Lm(5) = Lm2;
Lm(6) = Lm1;

Ldepth = Lm(1)+Lm(2)+Lm(3);

%% wijdtes bovenplaat gassen
Wu1 = Ledge + Lm1/2;
Wu2 = Lm1/2+Lm2/2;
Wu3 = Lm2/2 + Lm3/2;
Wu4 = Lm3;

Wu(1) = Wu1;
Wu(2) = Wu2;
Wu(3) = Wu3;
Wu(4) = Wu4;
Wu(5) = Wu3;
Wu(6) = Wu2;
Wu(7) = Wu1;

%% nodes

gnd = 0;
TrimL = 1;
T1 = 2;
H1 =3;
T2 = 4;
H2 = 5;
T3 = 6;
TrimR = 7;
offset_in = 8 - 1;                  % 8 - 14 trimL_in to TrimR_in
offset_out = 15 - 1;                % 15 - 21 trimL_out to TrimR_out
offset_up = 22 - 1;                 % 22 - 28 trimL_up to TrimR_up
offset_up_in = 29 - 1;              % 29 - 35 trimL_up_in to TrimR_up_in
offset_up_out = 36 - 1;             % 36 - 42 trimL_up_out to TrimR_up_out
offset_down = 43 - 1;               % 43 - 49 trimL_down to TrimR_down
offset_down_in = 50 - 1;            % 50 - 56 trimL_down_in to TrimR_down_out
offset_down_out = 57 - 1;           % 57 - 63 trimL_down_in to TrimR_down_out

amount_nodes = 63;

%% heaters

I(1, :) = [H1, gnd, total_power/2];
I(2, :) = [H2, gnd, total_power/2];

%% start the resistors

%% resistor counter initialization
n_R = 1;

%% the membrame resitors horizontal
Am = BchipMin*thicknes_pyrex;                       % oppervlakte membraan in M^2
Am_in = (BchipMax - BchipMin)*thicknes_pyrex/2;     % oppervlakte membraan in M^2
Am_out = (BchipMax - BchipMin)*thicknes_pyrex/2;    % oppervlakte membraan in M^2
% n_R
for i = 1:6
    R(n_R,:) = [i, i+1, Lm(i)/(k_pyrex*Am)];
    n_R = n_R + 1;
    R(n_R,:) = [i+offset_in, i+offset_in+1, Lm(i)/(k_pyrex*Am_in)];
    n_R = n_R + 1;
```

```
    R(n_R,:) = [i+offset_out, i+offset_out+1, Lm(i)/(k_pyrex*Am_out)];
    n_R = n_R + 1;
end


%% the membrame resitors depth
% n_R
for i = 1:7
    R(n_R,:) = [i, i+offset_in, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
    R(n_R,:) = [i, i+offset_out, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
end


%% Resistors Silicium verticaal
% correcting for the angle of the silicon rim
angle  = 35.3*(2*pi/360);
Overstaand = (Ld/2)*atan(angle);
Effect_Thickness_Silic_up = SiRimMax - Overstaand/2;
Effect_Thickness_Silic_down = SiRimMax - 3*Overstaand/2;

% n_R
R(n_R, :) = [1, 1 + offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [1 + offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;
for i = 1:7
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_in, i+offset_down_in, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_in, gnd, Rtemp];
    n_R = n_R + 1;

    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_out, i+offset_down_out, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_out, gnd, Rtemp];
    n_R = n_R + 1;
end
R(n_R, :) = [7, 7+offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;

%% Resistors Silicium Horizontaal

Overstaand = Ld*atan(angle);
Effect_Thickness_Silic = SiRimMax - Overstaand/2;
for i = 1:6
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_in, i+offset_down_in+1, Rtemp];
    n_R = n_R + 1;
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_out, i+offset_down_out+1, Rtemp];
    n_R = n_R + 1;

end

%% Resistors silicium diepte

Rtemp = Ldepth/(k_silicon*Ld*Effect_Thickness_Silic);
R(n_R, :) = [1+offset_down, 1+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [1+offset_down, 1+offset_down_out, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [7+offset_down, 7+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Rtemp];
n_R = n_R + 1;

%% resistors gas beneden verticaal

for i = 1:5
    R(n_R, :) = [i+1, i+offset_down+1, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down+1, gnd, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
end

%% resistors gas beneden horizontaal

Rtemp = (Lm(1) - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 2+offset_down, Rtemp];
n_R = n_R+1;
```

```
for i = 1:4
    R(n_R, :) = [i+offset_down + 1, i+offset_down+2, Lm(i+1)/(k_gas*BchipMin*Ld)];
    n_R = n_R+1;
end
R(n_R, :) = [6+offset_down, 7+offset_down, Rtemp];
n_R = n_R+1;

%% resistors gas beneden diepte

Rtemp = (Ldepth - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 1+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
R(n_R, :) = [1+offset_down, 1+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
for i = 1:5
    R(n_R, :) = [i+offset_down + 1, i+offset_down_in+1, Rtemp];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down + 1, i+offset_down_out+1, Rtemp];
    n_R = n_R+1;
end
R(n_R, :) = [7+offset_down, 7+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;

%% resistors gas boven verticaal

for i = 1:7
    R(n_R, :) = [i, i+offset_up, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, gnd, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_in, i+offset_up_in, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_in, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_out, i+offset_up_out, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_out, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
end


%% resistors gas boven horizontaal

for i = 1:6
    R(n_R, :) = [i+offset_up, i+offset_up+1, Lm(i)/(k_gas*BchipMin*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_in, i+offset_up_in+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_out, i+offset_up_out+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;
end

%% resistors gas boven diepte

for i = 1:7
    R(n_R, :) = [i+offset_up, i+offset_up_in, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, i+offset_up_out, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
end

%% nodal

nodes = nodal(amount_nodes, R, I);

%% results
temperatures = nodes + 293;
T = temperatures(1:7, 1);
y = T;
end
```

7.4.2 3D model Main

```
close all;
clear all;


k_Xe = 0.0052;
k_Kr = 0.0087;
```

```
k_CO2 = 0.015;
k_Ar = 0.016;
k_N2 = 0.024;
k_Ne = 0.047;
k_He = 0.152;
k_H2 = 0.167;
k = [k_Xe k_Kr k_CO2 k_Ar k_N2 k_Ne k_He k_H2];

x = [-669 -315 -90 0 90 315 669];
for i = 1:length(k)
    T(i,:) = model_6_fun(k(i));
end

T2 = T(:, 4);
T1 = T(:, 2);
Trim = T(:, 1);



figure;
for i = 1:length(k)
    plot(x,T(i,:), '-o');
    hold on
end
xlabel('position in um');
ylabel('temperature in Kelvin');
title('results model 6 (26mW)');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');

figure;
subplot(3,2,1)
plot(k,T2);
xlabel('k in W/mK');
ylabel('temperature in Kelvin');
title('T2');

subplot(3,2,2)
plot(k,T1);
xlabel('k in W/mK');
ylabel('temperature in Kelvin');
title('T1');

subplot(3,2,3)
plot(k,Trim);
xlabel('k in W/mK');
ylabel('temperature in Kelvin');
title('Trim');

subplot(3,2,4)
plot(k,T2-T1);
xlabel('k in W/mK');
ylabel('temperature in Kelvin');
title('T2-T1');

subplot(3,2,5)
plot(k,T2./T1);
xlabel('k in W/mK');
title('T2/T1');

subplot(3,2,6)
plot(k,(T2-Trim)./(T1-Trim));
xlabel('k in W/mK');
title('(T2-Trim)/(T1-Trim)');
```

7.5 Complete Model Matlab

7.5.1 Flow as IQ+ Flow Function

```
function y = model_8_fun(k_input, Cp, rho, mu, Pr, V)

%% chip parameters

Lu = 200*10^-6;             % stroomkanaal in M
Ld = 525*10^-6;             % holte onder chip in M
BchipMin = 976*10^-6;       % exclusief rim
BchipMax = 850*2*10^-6;     % inclusief rim
thicknes_pyrex = 40*10^-6;
SiRimMax = (850-976/2)*10^-6;
total_power = 26*10^-3;     % totaal vermogen verstookt door de chip in W

%% thermal conductivities
% all in W/Km
```

```
k_pyrex = 1;
k_silicon = 150;
k_gas = k_input;

%% lengtes tussen de nodes

Lm1 = 354*10^-6; % in meter
Lm2 = 225*10^-6; % in meter
Lm3 = 90*10^-6; % in meter
Ledge = 181*10^-6;

Lm(1) = Lm1;
Lm(2) = Lm2;
Lm(3) = Lm3;
Lm(4) = Lm3;
Lm(5) = Lm2;
Lm(6) = Lm1;

Ldepth = Lm(1)+Lm(2)+Lm(3);

%% wijdtes bovenplaat gassen
Wu1 = Ledge + Lm1/2;
Wu2 = Lm1/2+Lm2/2;
Wu3 = Lm2/2 + Lm3/2;
Wu4 = Lm3;

Wu(1) = Wu1;
Wu(2) = Wu2;
Wu(3) = Wu3;
Wu(4) = Wu4;
Wu(5) = Wu3;
Wu(6) = Wu2;
Wu(7) = Wu1;

%% nodes

gnd = 0;
TrimL = 1;
T1 = 2;
H1 =3;
T2 = 4;
H2 = 5;
T3 = 6;
TrimR = 7;
offset_in = 8 - 1;                      % 8 - 14 trimL_in to TrimR_in
offset_out = 15 - 1;                    % 15 - 21 trimL_out to TrimR_out
offset_up = 22 - 1;                     % 22 - 28 trimL_up to TrimR_up
offset_up_in = 29 - 1;                  % 29 - 35 trimL_up_in to TrimR_up_in
offset_up_out = 36 - 1;                 % 36 - 42 trimL_up_out to TrimR_up_out
offset_down = 43 - 1;                   % 43 - 49 trimL_down to TrimR_down
offset_down_in = 50 - 1;                % 50 - 56 trimL_down_in to TrimR_down_out
offset_down_out = 57 - 1;               % 57 - 63 trimL_down_in to TrimR_down_out

amount_nodes = 63;

%% heaters

I(1, :) = [H1, gnd, total_power/2];
I(2, :) = [H2, gnd, total_power/2];

%% start the resistors

%% resistor counter initialization
n_R = 1;

%% the membrame resitors horizontal
Am = BchipMin*thicknes_pyrex;                           % oppervlakte membraan in M^2
Am_in = (BchipMax - BchipMin)*thicknes_pyrex/2;         % oppervlakte membraan in M^2
Am_out = (BchipMax - BchipMin)*thicknes_pyrex/2;        % oppervlakte membraan in M^2
% n_R
for i = 1:6
    R(n_R,:) = [i, i+1, Lm(i)/(k_pyrex*Am)];
    n_R = n_R + 1;
    R(n_R,:) = [i+offset_in, i+offset_in+1, Lm(i)/(k_pyrex*Am_in)];
    n_R = n_R + 1;
    R(n_R,:) = [i+offset_out, i+offset_out+1, Lm(i)/(k_pyrex*Am_out)];
    n_R = n_R + 1;
end


%% the membrame resitors depth
% n_R
for i = 1:7
    R(n_R,:) = [i, i+offset_in, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
    R(n_R,:) = [i, i+offset_out, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
```

```
end


%% Resistors Silicium verticaal
% correcting for the angle of the silicon rim
angle  = 35.3*(2*pi/360);
Overstaand = (Ld/2)*atan(angle);
Effect_Thickness_Silic_up = SiRimMax - Overstaand/2;
Effect_Thickness_Silic_down = SiRimMax - 3*Overstaand/2;

% n_R
R(n_R, :) = [1, 1 + offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [1 + offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;
for i = 1:7
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_in, i+offset_down_in, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_in, gnd, Rtemp];
    n_R = n_R + 1;

    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_out, i+offset_down_out, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_out, gnd, Rtemp];
    n_R = n_R + 1;
end
R(n_R, :) = [7, 7+offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;

%% Resistors Silicium Horizontaal
% n_R
Overstaand = Ld*atan(angle);
Effect_Thickness_Silic = SiRimMax - Overstaand/2;
for i = 1:6
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_in, i+offset_down_in+1, Rtemp];
    n_R = n_R + 1;
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_out, i+offset_down_out+1, Rtemp];
    n_R = n_R + 1;

end

%% Resistors silicium diepte

Rtemp = Ldepth/(k_silicon*Ld*Effect_Thickness_Silic);
R(n_R, :) = [1+offset_down, 1+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [1+offset_down, 1+offset_down_out, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [7+offset_down, 7+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Rtemp];
n_R = n_R + 1;

%% resistors gas beneden verticaal
% n_R
for i = 1:5
    R(n_R, :) = [i+1, i+offset_down+1, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down+1, gnd, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
end

%% resistors gas beneden horizontaal
% n_R
Rtemp = (Lm(1) - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 2+offset_down, Rtemp];
n_R = n_R+1;
for i = 1:4
    R(n_R, :) = [i+offset_down + 1, i+offset_down+2, Lm(i+1)/(k_gas*BchipMin*Ld)];
    n_R = n_R+1;
end
R(n_R, :) = [6+offset_down, 7+offset_down, Rtemp];
n_R = n_R+1;

%% resistors gas beneden diepte
% n_R
Rtemp = (Ldepth - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 1+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
```

```
R(n_R, :) = [1+offset_down, 1+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
for i = 1:5
    R(n_R, :) = [i+offset_down + 1, i+offset_down_in+1, Rtemp];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down + 1, i+offset_down_out+1, Rtemp];
    n_R = n_R+1;
end
R(n_R, :) = [7+offset_down, 7+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;

%% resistors gas boven verticaal
% n_R
for i = 1:7
    R(n_R, :) = [i, i+offset_up, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, gnd, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_in, i+offset_up_in, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_in, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_out, i+offset_up_out, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_out, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
end


%% resistors gas boven horizontaal
% n_R

for i = 1:6
    R(n_R, :) = [i+offset_up, i+offset_up+1, Lm(i)/(k_gas*BchipMin*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_in, i+offset_up_in+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_out, i+offset_up_out+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;
end

%% resistors gas boven diepte
% n_R
for i = 1:7
    R(n_R, :) = [i+offset_up, i+offset_up_in, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, i+offset_up_out, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
end

%% add flow resitors

x = [1:1:1700];
x = x.*10^-6;
Re = (rho*V.*x)/mu;
Nu = 0.332*nthroot(Pr,3).*sqrt(Re);
h = (Nu.*k_gas)./x;
for i = 1:7
    Lmin = 0;
    for j = 1:7
        if j < i
            Lmin = Lmin + Wu(j)*10^6;
        end
    end
    Lmin = int32(Lmin)+1;
    Lmax = int32(Lmin + Wu(i)*10^6)-1;
    h_avg(i) = trapz(x(Lmin:Lmax),h(Lmin:Lmax));
end

Qmiddle = h_avg.*BchipMin;
Qedge = h_avg.*SiRimMax;


for i = 1:7
    R(n_R, :) = [i, gnd, 1/Qmiddle(i)];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_in, gnd, 1/Qedge(i)];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_out, gnd, 1/Qedge(i)];
    n_R = n_R+1;
end
```

```
%% nodal

nodes = nodal(amount_nodes, R, I);
temperatures = nodes + 293;

T = temperatures(1:7, 1);
y = T;
end
```

7.5.2 Flow at a 90 degree angle Function

```
function y = model_9_fun(k_input, Cp, rho, mu, Pr, V)

%% chip parameters

Lu = 200*10^-6;             % stroomkanaal in M
Ld = 525*10^-6;             % holte onder chip in M
BchipMin = 976*10^-6;       % exclusief rim
BchipMax = 850*2*10^-6;     % inclusief rim
thicknes_pyrex = 40*10^-6;
SiRimMax = (850-976/2)*10^-6;
total_power = 26*10^-3;     % totaal vermogen verstookt door de chip in W

%% thermal conductivities
% all in W/Km

k_pyrex = 1;
k_silicon = 150;
k_gas = k_input;


%% lengtes tussen de nodes

Lm1 = 354*10^-6; % in meter
Lm2 = 225*10^-6; % in meter
Lm3 = 90*10^-6; % in meter
Ledge = 181*10^-6;

Lm(1) = Lm1;
Lm(2) = Lm2;
Lm(3) = Lm3;
Lm(4) = Lm3;
Lm(5) = Lm2;
Lm(6) = Lm1;

Ldepth = Lm(1)+Lm(2)+Lm(3);

%% wijdtes bovenplaat gassen
Wu1 = Ledge + Lm1/2;
Wu2 = Lm1/2+Lm2/2;
Wu3 = Lm2/2 + Lm3/2;
Wu4 = Lm3;

Wu(1) = Wu1;
Wu(2) = Wu2;
Wu(3) = Wu3;
Wu(4) = Wu4;
Wu(5) = Wu3;
Wu(6) = Wu2;
Wu(7) = Wu1;

%% nodes

gnd = 0;
TrimL = 1;
T1 = 2;
H1 =3;
T2 = 4;
H2 = 5;
T3 = 6;
TrimR = 7;
offset_in = 8 - 1;                      % 8 - 14 trimL_in to TrimR_in
offset_out = 15 - 1;                    % 15 - 21 trimL_out to TrimR_out
offset_up = 22 - 1;                     % 22 - 28 trimL_up to TrimR_up
offset_up_in = 29 - 1;                  % 29 - 35 trimL_up_in to TrimR_up_in
offset_up_out = 36 - 1;                 % 36 - 42 trimL_up_out to TrimR_up_out
offset_down = 43 - 1;                   % 43 - 49 trimL_down to TrimR_down
offset_down_in = 50 - 1;                % 50 - 56 trimL_down_in to TrimR_down_out
offset_down_out = 57 - 1;               % 57 - 63 trimL_down_in to TrimR_down_out

amount_nodes = 63;

%% heaters
```

```
I(1, :) = [H1, gnd, total_power/2];
I(2, :) = [H2, gnd, total_power/2];

%% start the resistors

%% resistor counter initialization
n_R = 1;

%% the membrame resitors horizontal
Am = BchipMin*thicknes_pyrex;                           % oppervlakte membraan in M^2
Am_in = (BchipMax - BchipMin)*thicknes_pyrex/2;         % oppervlakte membraan in M^2
Am_out = (BchipMax - BchipMin)*thicknes_pyrex/2;        % oppervlakte membraan in M^2
% n_R
for i = 1:6
    R(n_R,:) = [i, i+1, Lm(i)/(k_pyrex*Am)];
    n_R = n_R + 1;
    R(n_R,:) = [i+offset_in, i+offset_in+1, Lm(i)/(k_pyrex*Am_in)];
    n_R = n_R + 1;
    R(n_R,:) = [i+offset_out, i+offset_out+1, Lm(i)/(k_pyrex*Am_out)];
    n_R = n_R + 1;
end


%% the membrame resitors depth
% n_R
for i = 1:7
    R(n_R,:) = [i, i+offset_in, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
    R(n_R,:) = [i, i+offset_out, Ldepth/(k_pyrex*thicknes_pyrex*Wu(i))];
    n_R = n_R + 1;
end


%% Resistors Silicium verticaal
% correcting for the angle of the silicon rim
angle  = 35.3*(2*pi/360);
Overstaand = (Ld/2)*atan(angle);
Effect_Thickness_Silic_up = SiRimMax - Overstaand/2;
Effect_Thickness_Silic_down = SiRimMax - 3*Overstaand/2;

% n_R
R(n_R, :) = [1, 1 + offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [1 + offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;
for i = 1:7
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_in, i+offset_down_in, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_in, gnd, Rtemp];
    n_R = n_R + 1;

    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_up);
    R(n_R, :) = [i+offset_out, i+offset_down_out, Rtemp];
    n_R = n_R + 1;
    Rtemp = (Ld/2)/(k_silicon*Wu(i)*Effect_Thickness_Silic_down);
    R(n_R, :) = [i+offset_down_out, gnd, Rtemp];
    n_R = n_R + 1;
end
R(n_R, :) = [7, 7+offset_down, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_up)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, gnd, (Ld/2)/(k_silicon*BchipMin*Effect_Thickness_Silic_down)];
n_R = n_R+1;

%% Resistors Silicium Horizontaal
% n_R
Overstaand = Ld*atan(angle);
Effect_Thickness_Silic = SiRimMax - Overstaand/2;
for i = 1:6
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_in, i+offset_down_in+1, Rtemp];
    n_R = n_R + 1;
    Rtemp = Lm(i)/(k_silicon*Ld*Effect_Thickness_Silic);
    R(n_R, :) = [i+offset_down_out, i+offset_down_out+1, Rtemp];
    n_R = n_R + 1;

end

%% Resistors silicium diepte

Rtemp = Ldepth/(k_silicon*Ld*Effect_Thickness_Silic);
R(n_R, :) = [1+offset_down, 1+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [1+offset_down, 1+offset_down_out, Rtemp];
n_R = n_R + 1;
```

```
R(n_R, :) = [7+offset_down, 7+offset_down_in, Rtemp];
n_R = n_R + 1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Rtemp];
n_R = n_R + 1;

%% resistors gas beneden verticaal
% n_R
for i = 1:5
    R(n_R, :) = [i+1, i+offset_down+1, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down+1, gnd, (Ld/2)/(k_gas*BchipMin*Wu(i+1))];
    n_R = n_R+1;
end

%% resistors gas beneden horizontaal
% n_R
Rtemp = (Lm(1) - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 2+offset_down, Rtemp];
n_R = n_R+1;
for i = 1:4
    R(n_R, :) = [i+offset_down + 1, i+offset_down+2, Lm(i+1)/(k_gas*BchipMin*Ld)];
    n_R = n_R+1;
end
R(n_R, :) = [6+offset_down, 7+offset_down, Rtemp];
n_R = n_R+1;

%% resistors gas beneden diepte
% n_R
Rtemp = (Ldepth - Effect_Thickness_Silic + Ledge)/(k_gas*BchipMin*Ld) + (Effect_Thickness_Silic - Ledge)/(k_silicon*BchipMin*Ld);
R(n_R, :) = [1+offset_down, 1+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
R(n_R, :) = [1+offset_down, 1+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
for i = 1:5
    R(n_R, :) = [i+offset_down + 1, i+offset_down_in+1, Rtemp];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_down + 1, i+offset_down_out+1, Rtemp];
    n_R = n_R+1;
end
R(n_R, :) = [7+offset_down, 7+offset_down_in, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;
R(n_R, :) = [7+offset_down, 7+offset_down_out, Ldepth/(k_silicon*Effect_Thickness_Silic*Ld)];
n_R = n_R+1;

%% resistors gas boven verticaal
% n_R
for i = 1:7
    R(n_R, :) = [i, i+offset_up, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, gnd, (Lu/2)/(k_gas*BchipMin*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_in, i+offset_up_in, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_in, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_out, i+offset_up_out, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up_out, gnd, (Lu/2)/(k_gas*SiRimMax*Wu(i))];
    n_R = n_R+1;
end


%% resistors gas boven horizontaal
% n_R

for i = 1:6
    R(n_R, :) = [i+offset_up, i+offset_up+1, Lm(i)/(k_gas*BchipMin*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_in, i+offset_up_in+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;

    R(n_R, :) = [i+offset_up_out, i+offset_up_out+1, Lm(i)/(k_gas*SiRimMax*Lu)];
    n_R = n_R+1;
end

%% resistors gas boven diepte
% n_R
for i = 1:7
    R(n_R, :) = [i+offset_up, i+offset_up_in, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_up, i+offset_up_out, Ldepth/(k_gas*Lu*Wu(i))];
    n_R = n_R+1;
end
```

```
%% add flow resitors


x = [1:1:1700];
x = x.*10^-6;
x_index1 = int32(SiRimMax*10^6);
x_index2 = int32(SiRimMax*10^6+BchipMin*10^6);
x_index3 = int32(BchipMax*10^6);
Re = (rho*V.*x)/mu;
Nu = 0.332*nthroot(Pr,3).*sqrt(Re);
h = (Nu.*k_gas)./x;
h1 = trapz(x(1:x_index1),h(1:x_index1));
h2 = trapz(x(x_index1:x_index2),h(x_index1:x_index2));
h3 = trapz(x(x_index2:x_index3),h(x_index2:x_index3));
for i = 1:7
    Q1(i) = h1*Wu(i);
    Q2(i) = h2*Wu(i);
    Q3(i) = h3*Wu(i);
end

for i = 1:7
    R(n_R, :) = [i+offset_in, gnd, 1/Q1(i)];
    n_R = n_R+1;
    R(n_R, :) = [i, gnd, 1/Q2(i)];
    n_R = n_R+1;
    R(n_R, :) = [i+offset_out, gnd, 1/Q3(i)];
    n_R = n_R+1;
end


%% nodal

nodes = nodal(amount_nodes, R, I);
temperatures = nodes + 293;

T = temperatures(1:7, 1);
y = T;

end
```

7.5.3 Running The models and saving the Data

```
close all;
clear all;

%% gas values
k_Xe = 0.0052;
k_Kr = 0.0087;
k_CO2 = 0.015;
k_Ar = 0.016;
k_N2 = 0.024;
k_Ne = 0.047;
k_He = 0.152;
k_H2 = 0.167;

Cp_Xe = 158.4;
Cp_Kr = 284.14;
Cp_CO2 = 840.37;
Cp_Ar = 520.64;
Cp_N2 = 1041;
Cp_Ne = 1030;
Cp_He = 5193;
Cp_H2 = 14280;


Pr_Xe = 0.695;
Pr_Kr = 0.816;
Pr_CO2 = 0.794;
Pr_Ar = 0.7;
Pr_N2 = 0.715;
Pr_Ne = 0.694;
Pr_He = 0.68;
Pr_H2 = 0.719;

mu_Xe = 2.28*10^-5;
mu_Kr = 2.5*10^-5;
mu_CO2 = 1.37*10^-5;
mu_Ar = 2.125*10^-5;
mu_N2 = 1.663*10^-5;
mu_Ne = 3.13*10^-5;
mu_He = 1.99*10^-5;
mu_H2 = 0.8411*10^-5;

density_Xe = 5.333;
```

```
density_Kr = 3.404;
density_CO2 = 1.788;
density_Ar = 1.623;
density_N2 = 1.138;
density_Ne = 0.82;
density_He = 0.163;
density_H2 = 0.082;


k = [k_Xe k_Kr k_CO2 k_Ar k_N2 k_Ne k_He k_H2];
mu = [mu_Xe mu_Kr mu_CO2 mu_Ar mu_N2 mu_Ne mu_He mu_H2];
Pr = [Pr_Xe Pr_Kr Pr_CO2 Pr_Ar Pr_N2 Pr_Ne Pr_He Pr_H2];
density = [density_Xe density_Kr density_CO2 density_Ar density_N2 density_Ne density_He density_H2];
Cp = [Cp_Xe Cp_Kr Cp_CO2 Cp_Ar Cp_N2 Cp_Ne Cp_He Cp_H2];

% membrame positions
x1 = [-669 -315 -90 0 90 315 669];

% gas speeds
V = [0:0.1:30];

for j = 1:length(k)
for i = 1:length(V)
    T9(j,i,:) = model_9_fun(k(j), Cp(j), density(j), mu(j), Pr(j), V(i));
    T8(j,i,:) = model_8_fun(k(j), Cp(j), density(j), mu(j), Pr(j), V(i));
end
j
end
save('model10_var', 'T9', 'T8', 'k', 'V', 'x1');
```

7.5.4 Plotting and analyzing the data

```
close all;
clear all;

A = load('model10_var.mat');

T9 = A.T9;
T8 = A.T8;
V = A.V;
k = A.k;
x1 = A.x1;

error = sum(sum(sum(T9 - T8)));

Trim_9 = 0.5*(T9(:, :, 1) + T9(:, :, 7));
T1_9 = T9(:, :, 2) - Trim_9;
T2_9 = T9(:, :, 4) - Trim_9;
T3_9 = T9(:, :, 6) - Trim_9;
T4_9 = 0.5*(T1_9 + T3_9);

Trim_8 = 0.5*(T8(:, :, 1) + T8(:, :, 7));
T1_8 = T8(:, :, 2) - Trim_8;
T2_8 = T8(:, :, 4) - Trim_8;
T3_8 = T8(:, :, 6) - Trim_8;
T4_8 = 0.5*(T1_8 + T3_8);

%% plotting M9

figure;
sgtitle('Model 9');
subplot(3,2,1);
for i = 1:length(k)
plot(V,T3_9(i,:)-T1_9(i,:));
hold on
end
title('T3-T1');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,2);
for i = 1:length(k)
plot(V,T2_9(i,:));
hold on
end
title('T2');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,3);
for i = 1:length(k)
plot(V,(T3_9(i,:)-T1_9(i,:))./(T2_9(i,:)));
hold on
end
title('(T3-T1)/(T2)');
```

```
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,4);
for i = 1:length(k)
plot(V,T4_9(i,:));
hold on
end
title('T4');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,5);
for i = 1:length(k)
plot(V,(T2_9(i,:))./(T4_9(i,:)));
hold on
end
title('(T2)/(T4)');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,6);
for i = 1:50:length(V)
plot(k,(T2_9(:,i))./(T4_9(:,i)));
hold on
end
title('(T2)/(T4)');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
xlabel('k in W/mK');


%% plotting M8
figure;
sgtitle('Model 8');
subplot(3,2,1);
for i = 1:length(k)
plot(V,T3_8(i,:)-T1_8(i,:));
hold on
end
title('T3-T1');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,2);
for i = 1:length(k)
plot(V,T2_8(i,:));
hold on
end
title('T2');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,3);
for i = 1:length(k)
plot(V,(T3_8(i,:)-T1_8(i,:))./(T2_8(i,:)));
hold on
end
title('(T3-T1)/(T2)');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,4);
for i = 1:length(k)
plot(V,T4_8(i,:));
hold on
end
title('T4');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,5);
for i = 1:length(k)
plot(V,(T2_8(i,:))./(T4_8(i,:)));
hold on
end
title('(T2)/(T4)');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(3,2,6);
for i = 1:50:length(V)
plot(k,(T2_8(:,i))./(T4_8(:,i)));
hold on
end
title('(T2)/(T4)');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
xlabel('k in W/mK');
```

```
%% Surface plot
Z = (T2_9)./(T4_9);

Y = (T3_8-T1_8)./(T2_8);


%% some more trying



for i = 1:2000
diff = 1 + (-0.01*i).*Y;

Temp = Z.*diff;
Temp_avg = mean(Temp');
k_fit = polyfit(Temp_avg, k, 1);
k_estimated = k_fit(1).*Temp + k_fit(2);
for m = 1: length(V)
errortemp(m,:) = (k - k_estimated(:,m)').^2;
end
error_lin(i) = sum(sum(errortemp));
least_error_lin = min(error_lin);
best_coefficient_lin =  -0.01*find(error_lin == min(error_lin));
end

for i = 1:200
diff = 1 + (-0.01*i).*sqrt(Y);

Temp = Z.*diff;
Temp_avg = mean(Temp');
k_fit = polyfit(Temp_avg, k, 1);
k_estimated = k_fit(1).*Temp + k_fit(2);
for m = 1: length(V)
errortemp(m, :) = (k - k_estimated(:,m)').^2;
end
error_sqrt(i) = sum(sum(errortemp));
least_error_sqrt = min(error_sqrt);
best_coefficient_sqrt = -0.01*find(error_sqrt == min(error_sqrt));
end

for i = 1:200
    for j = 1:200
        diff = 1 + (-0.1*i).*(Y) + (-0.1*j).*(Y).^2;

        Temp = Z.*diff;
        Temp_avg = mean(Temp');
        k_fit = polyfit(Temp_avg, k, 1);
        k_estimated = k_fit(1).*Temp + k_fit(2);
        for m = 1: length(V)
            errortemp(m,:) = (k - k_estimated(:,m)').^2;
        end
        error_poly(i,j) = sum(sum(errortemp));

        least_error_poly = min(min(error_poly));
        [best_coefficient_poly_i, best_coefficient_poly_j] = find(error_poly == min(min(error_poly)));
        test = error_poly(best_coefficient_poly_i, best_coefficient_poly_j);
        best_coefficient_poly_i = (-0.1*best_coefficient_poly_i);
        best_coefficient_poly_j = (-0.1*best_coefficient_poly_j);
    end
end

%% plotting

diff = 1 + best_coefficient_sqrt.*sqrt(Y);
Temp = Z.*diff;
Temp_avg = mean(Temp');
k_fit = polyfit(Temp_avg, k, 1);
k_estimated = k_fit(1).*Temp + k_fit(2);

figure;
sgtitle('sqrt');
subplot(2,2,2);
for i = 1:50:length(V)
plot(k,Temp(:,i));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,1);
for i = 1:length(k)
plot(V,Temp(i,:));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
```

```
xlabel('V in m/s');

subplot(2,2,4);
for i = 1:50:length(V)
plot(k,k_estimated(:,i));
hold on
end
title('fitted to k (T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
ylabel('estimated k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,3);
for i = 1:length(k)
plot(V,k_estimated(i,:));
hold on
end
title('fitted to k (T2/T4)*(1-5*((T3-T1)/T2))');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');
ylabel('estimated k in W/mK');

diff = 1 + best_coefficient_lin.*Y;
Temp = Z.*diff;
Temp_avg = mean(Temp');
k_fit = polyfit(Temp_avg, k, 1);
k_estimated = k_fit(1).*Temp + k_fit(2);

figure;
sgtitle('Determining the Thermal Conductivity');
subplot(2,2,2);
for i = 1:50:length(V)
plot(k,Temp(:,i));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,1);
for i = 1:length(k)
plot(V,Temp(i,:));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(2,2,4);
for i = 1:50:length(V)
plot(k,k_estimated(:,i));
hold on
end
title('converted to k (T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
ylabel('estimated k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,3);
for i = 1:length(k)
plot(V,k_estimated(i,:));
hold on
end
title('converted to k (T2/T4)*(1-5*((T3-T1)/T2))');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');
ylabel('estimated k in W/mK');

diff = 1 + best_coefficient_poly_i.*Y + best_coefficient_poly_j.*(Y).^2;
Temp = Z.*diff;
Temp_avg = mean(Temp');
k_fit = polyfit(Temp_avg, k, 1);
k_estimated = k_fit(1).*Temp + k_fit(2);

figure;
sgtitle('poly');
subplot(2,2,2);
for i = 1:50:length(V)
plot(k,Temp(:,i));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,1);
for i = 1:length(k)
plot(V,Temp(i,:));
hold on
end
title('(T2/T4)*(1-5*((T3-T1)/T2))');
```

```
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');

subplot(2,2,4);
for i = 1:50:length(V)
plot(k,k_estimated(:,i));
hold on
end
title('fitted to k (T2/T4)*(1-5*((T3-T1)/T2))');
xlabel('k in W/mK');
ylabel('estimated k in W/mK');
legend('0 m/s', '5 m/s', '10m/s', '15 m/s', '20 m/s', '25 m/s', '30 m/s');
subplot(2,2,3);
for i = 1:length(k)
plot(V,k_estimated(i,:));
hold on
end
title('fitted to k (T2/T4)*(1-5*((T3-T1)/T2))');
legend('Xe', 'Kr', 'CO2', 'Ar', 'N2', 'Ne', 'He', 'H2');
xlabel('V in m/s');
ylabel('estimated k in W/mK');
```