Cool-down and Warm-up of the Cryogenic Distribution Line at ESS

Riccard Andersson

LUND UNIVERSITY

Department of Automatic Control

MSc Thesis
ISRN LUTFD2/TFRT--5939--SE
ISSN 0280–5316

Department of Automatic Control
Lund University
Box 118
SE-221 00 LUND
Sweden

Abstract

The European Spallation Source, ESS, is a joint collaboration of 17 European countries, where the world's most powerful neutron source will be built for future research within a vast variety of fields. In order to deliver the highly energetic neutrons, protons are accelerated to energies in the GeV range and then collided with a spallation target. This acceleration process requires superconducting cavities cooled down to slightly over 2 K, and this cooling is done through the cryogenic system using helium.

In this project, the cryogenic system at ESS has been modeled in Dymola. Simulations have been made of the cool-down and warm-up of the superconducting part of the accelerator. This was done in separate simulations for the cryogenic distribution line and for an individual cryomodule. Additionally, a model was created for the helium discharge system, in order to size the vent line leading rapidly expanded helium from the cold masses back to the cryoplant. The mathematical tools and structure of the modeling are described in a separate chapter.

Acknowledgements

First of all, I would like to thank Dr. John G. Weisend II for taking on the role of being my supervisor at ESS and giving me the opportunity to dig into the subject of cryogenics, together with letting me take part in a large-scale European science project. I have received many valuable comments on my work and on the world of science and engineering in general.

Prof. Rolf Johansson, who has been my academic supervisor, has been very supportive in his comments on the progress of my work, the structure of scientific writing, and key points of developing useful results. In addition, Rolf has been quick at responding to my requests and considerations, which has made the development of this thesis as smooth as possible.

I want to thank Dr. Jaroslaw Fydrych for his helpful attitude towards guiding me in the field of accelerator physics and cryogenics. His expertise and pedagogical abilities have been a key pillar in my learning and this Master thesis. Without his helpful attitude, my work at ESS would not have reached the same level.

Other co-workers that have been helpful in creating this thesis are Wolfgang Hees, Philipp Arnold, and Dr. Xilong Wang, whose knowledge within cryogenics has been very useful inputs in the development of my work. I have also received a great amount of support from Dr. Carl Wilhelmsson at Modelon AB, who has helped me in the development of models in Dymola.

Contents

1

Introduction

1.1 The European Spallation Source

The European Spallation Source, ESS, is a high-power neutron source that will be built in Lund. It is a European project involving 17 countries, of which Sweden and Denmark are the host countries. The actual facility will be in Lund, while the data management center will be located at the Niels Bohr Institute in Copenhagen. Sweden and Denmark also hold a large portion of the company stock: 35% and 12.5%, respectively. To reach the 50% limit for the Scandinavian countries, Norway holds the remaining 2.5%, while the rest is shared by the other fourteen European countries involved. In 2013, ESS went into the *construction phase* of the project, being preceded by the *pre-construction phase* since its start. The current phase will continue until all of the construction is finished, which is planned to occur in 2025. First delivery of neutrons, however, is planned already in 2019. Even though full power has not been reached and all instruments have not been completed at that time [1, 2, 3].

Background

Since the construction of ISIS in Oxfordshire, UK, in 1985, there has been a clear need for a stronger neutron source to perform experiments in the front line of science. This made the European Neutron Scattering Association, ENSA, convince the Organisation for Economic Cooperation and Development, OECD, in 1999 that powerful neutron sources were necessary tools for the future, and that one facility should be placed in each of the continents of Europe, Asia, and America. The latter two already had such sources, and the turn had now come to Europe. In 2009, Lund was decided to be the place to build it. The year after, the company ESS AB formed, having Sweden as the owner of 75% of the stock and Denmark holding the remaining 25% [1].

Neutron Spallation

The process of neutron spallation was first discovered in 1937 by Glenn Seaborg. The process is performed by accelerating protons into a target, which in the case of ESS is made of tungsten. Upon being hit, the tungsten target then scatters some 20-30 neutrons per proton, and the neutrons are guided to their respective experiments using so called reflectors. The word *spallation* refers to the same spallation that can be seen on brick walls with peeled-off layers of concrete, or by knocking of small splinters from a rock using another rock. Just as neutrons are spalled off of the tungsten target by the protons.

Research

Within the partner countries, there are over 60 partner labs associated. In addition to the partner labs and their visits to the neutron source, there will be some 2000 to 3000 guest researchers coming each year. The research at ESS will be carried out within multiple disciplines and it will span fields such as life science, energy, environmental technology, culture and archeology, plastics, pharmaceuticals, molecular science, fundamental physics, engine technology, and more [3]. The power of the source will be up to 5 MW, and the brightness will surpass the second greatest neutron source, SNS in Oak Ridge, Tennessee [4], by 30 times. This is seen in Fig. 1.1, where the integral of the brilliance over time yields the brightness [5]. Another curious fact is that ESS will use long proton pulses, with a pulse length of 2.86 ms - something that has not been used before. With this technology, objects of size 10^{-11} to 10^{-6} meters can be resolved in time frames of 10^{-9} to 10^{-3} seconds. This makes it possible to study smaller, more complex objects in real time in the 22 instruments that will be available. As neutrons do not carry any charge, they only interact through the strong nuclear force. This means that they will leave the samples intact, and studies of bulky materials does not become a limitation [1].

Technical Details

The production of neutrons is done through a series of steps. The first step is to heat a hydrogen gas, which produces a plasma of free protons and electrons. The protons and electrons are separated in an electromagnetic field, and the protons are collected and pre-accelerated through 40 meters of normal conducting acceleration devices. Now the protons have gained enough energy to enter into the superconducting [7] part of the accelerator. This part consists of radio frequency niobium cavities, cooled to 2 Kelvin. After having been accelerated through these, which makes up about 305 meters, the protons have gained 2.5 GeV of energy. This means a velocity of 96% that of the speed of light at the point of hitting the spallation target. The target consists of neutron rich tungsten ($^{110}_{74}$W) [3] divided into 33 radial sections, which is rotating at a frequency of $\frac{1/33}{1/14} = 0.42$ revolutions per second [1, 2].

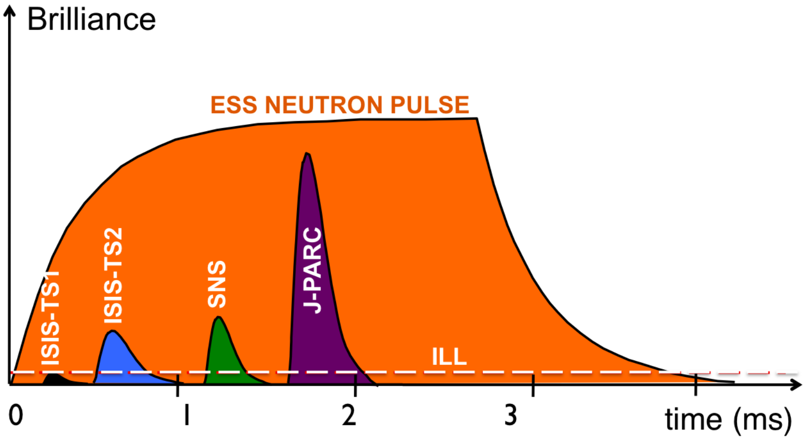


Figure 1.1 The brilliance of various neutron sources, as a function of time. ISIS in UK, SNS in U.S., ILL in France, and J-PARC in Japan are displayed together with ESS. Note the pulse length of ESS (orange), together with the high brilliance peak. Source: [6].

When the protons hit the target, neutrons are "knocked off" of the tungsten material. The produced neutrons then have a very high energy, and a velocity of about 10% of the speed of light. This is too high for usage within experiments, and the neutrons have to be slowed down to about the speed of sound using hydrogen moderators, and then guided to the right experimental hall. When the neutrons have been guided to their experiments, they scatter off of the sample in various directions and give rise to large amounts of data which is used for 3D image production and further analysis [1, 2].

There will be a maximum proton energy of 2.5 GeV and the current corresponds for 50 mA. The frequency of the proton pulse is 14 Hz, with a previously mentioned pulse length of 2.86 ms. The peak beam power is 125 MW, and the average is 5 MW. In total, the whole machine is 442.7 meters, with an allowance of 159.2 meters of extra space for future upgrades [2].

1.2 The Cryogenic System at ESS

The cryogenic distribution system for the linear accelerator at ESS has the main purpose to deliver the cooling medium to the superconducting cavities used in the acceleration of the proton beam. This is done by cold helium which is transported to the different parts of the accelerator [8]. The system consists of four major parts, including the accelerator cryoplant, the transfer line, the cryomodules, and the distribution line. A schematic view of the distribution line with the cryoplant and the transfer line can be seen in Fig. 1.2. Further, the distribution line contains a spoke cavity section, a medium-β section, and a high-β section, and can be divided into a modular setup where each module includes a valve box and jumper connection, further described below.

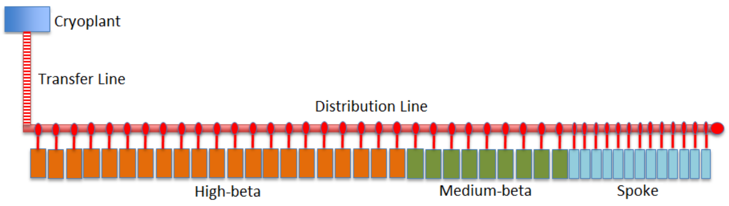


Figure 1.2 An overview of the cryogenic system for the superconducting accelerator part. The cryoplant, transfer line, and distribution line are named. The spoke cavity section is in light blue color, the medium-β is in green, and the high-β is in orange. Courtesy of Jaroslaw Fydrych.

The Cryoplant

In the cryoplant, the helium is being cooled and liquefied for the purpose of cooling the superconducting parts of the accelerator. It consists of a number of heat exchangers, compressors, and valves to yield the necessary amount and temperature of the helium. The cryoplant is aimed to work at an efficiency of at least 26% of the Carnot efficiency [9], and is expected to operate for at least 25 years. The cooling power will be specified as 3.3 kW at 2 K and 11.8 kW at 4.5 K equivalent [10, 11].

The Distribution Line

The cryogenic distribution line will stretch along the entire superconducting part of the accelerator and deliver and distribute the cold helium to the cryomodules. It has a total length of 305 meters, and will be placed in the accelerator tunnel. As there will be a total number of 43 cryomodules, the distribution line can be divided into 43 modules, or sections, each being of a similar structure. A sketch of a section and its structure with the distribution line, a valve box, and a jumper connection can be seen in Fig. 1.3. In addition, there will be an end box placed at the turnaround of the pipes at the very end of the distribution line. This end box contains connectors where the helium supply pipe connects to the vapor low pressure pipe and the thermal shield supply pipe connects to the thermal shield return pipe. There will also be a second choice of path for the cold helium circuit, where an electric heater is placed that will be used to add heat to the cold helium during steady-state operation, to increase the pressure of the returning helium vapor. A picture of the three sections of the distribution line, with its valve boxes, is seen in Fig. 1.4.

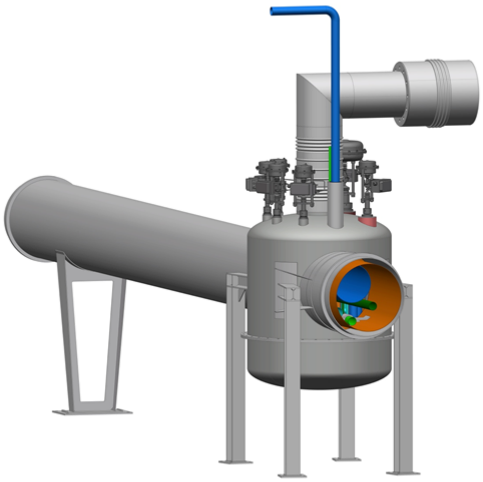


Figure 1.3 One section of the distribution line with the valve box and the jumper connection. Courtesy of Piotr Tereszkowski.

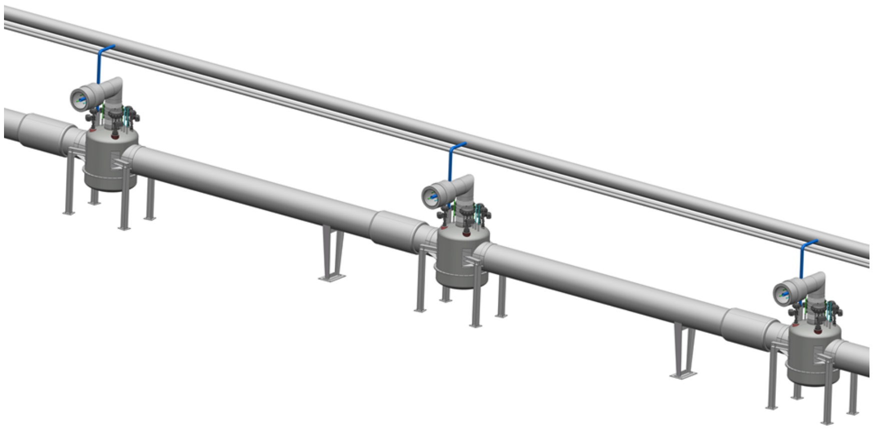


Figure 1.4 Three connected sections of the distribution line with the valve box and the jumper connection. Courtesy of Piotr Tereszkowski.

The Transfer Line

The cryogenic transfer line has the purpose of leading the helium from the cryoplant to the distribution line, and back again after the helium has passed through the system. The transfer line will have a length of 50 meters.

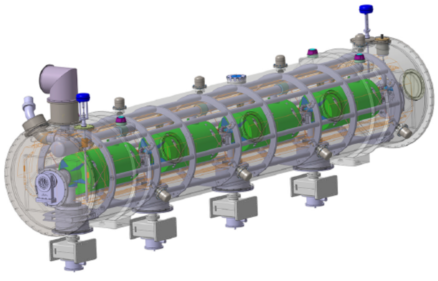


Figure 1.5 A transparent picture of an elliptical cryomodule with four cavities. Taken from [2].

Cryogenic Piping

Both the distribution line and the transfer line contain four process pipes. These are named the helium supply pipe (HS), the vapor low pressure return pipe (VLP), the thermal shield supply pipe (TSS) and the thermal shield return pipe (TSR). These process pipes are surrounded by a thermal shield, which absorbs radiation from the surroundings in order to minimize the heat loads to the pipes. This is then surrounded by the external envelope, sometimes referred to as the vacuum jacket, which keeps a high vacuum inside to avoid convective heat transfer. In the specifications for the distribution and transfer lines, the vacuum is of some 10^{-5} (at 300 K) to 10^{-6} (at 4.5 K) mbar [11]. While the TSR pipe is thermodynamically connected to the the thermal shield and thus keeps the same temperature, the other three pipes are only attached to occasional supports for keeping the pipes in place. In this way, these pipes will stay at lower temperatures and to a large extent be protected from heat loads.

The Cryomodules

The cryomodules are the advanced containers where the accelerating cavities are immersed in a bath of liquid helium to benefit from superconduction. The cryomodules are attached to the distribution line via jumper connections, and will have a constant inflow and outflow of helium. The incoming helium has a temperature of about 4.5 K, whereas the cryomodules keep an operating temperature of 2 K (helium liquefies at 2.18 K [12]). This means that each cryomodule has a process of lowering the helium temperature, where a so called Joule-Thomson cycle (see for example Reference [13]) is carried out. From the properties of cavities and the dynamics of the proton beam, the superconducting acceleration will be done in both spoke cavities and elliptical cavities. There will be a total number of 13 spoke cavity cryomodules, containing two cavities each, and 30 elliptical cavity cryomodules (9 medium-β and 21 high-β) containing four cavities each. A picture of such an elliptical cryomodule can be seen in Fig. 1.5.

2

Problem Formulation

This chapter describes the problems that have been elaborated and set up in this project. This has been divided into four main parts. The first part deals with helium being discharged into a relief system, and the arising pressures and possible mass flow rates have been studied to determine an appropriate sizing of the pipe. In the second part, the cool-down and warm-up have been modeled and simulated for the cryogenic distribution line excluding the cryomodules. The third part has been dedicated to the modeling and simulation of individual cryomodules, being both spoke and elliptical cryomodules. The simulations here have taken different mass flow rates into account in order to find a relation between cool-down and warm-up times and the rate of which helium has been supplied. Finally, there is a section on model comparisons between the Dymola models developed in this project and experimental and analytical results from published papers on the helium relief system for the LHC at CERN. The comparison section starts off with a comparison to a simulation in Excel with the HePak macro.

2.1 Helium Discharge to Relief System

The helium relief system is a common safety feature in large scale accelerator facilities, where a sudden temperature increase in the cold mass would be transferred to the helium contained in the system. Such a temperature increase could for instance be due to a quench [14], where the accelerating cavities lose their superconductivity and become normally conducting, which generates a large amount of resistive heat. The heated helium then undergoes a rapid expansion and pressure rise. To avoid unfortunate damages to the piping and other parts, there is typically one or more safety valves installed, where a pressure increase above a certain threshold causes the valve to open. The helium then discharges into the relief system where it is collected, and losses of helium into the environment is avoided. The procedure of helium inflow into the relief piping is simulated in order to find the appropriate dimensions.

The simulation was carried out by setting up a model of the relief system, as can be seen in Fig. 2.1. The simulations covered a span of pipe dimensions between DN50 and DN200 and was run for two different cases. The first case considers an inflow at one end of the pipe. For a predefined maximum pressure that arises, here 1.1 or 1.2 bar, the maximum mass flow rate is plotted against the pipe diameter to find for what flow a certain pipe dimension is suitable. For comparison reasons, as a second case, the flow was also simulated as divided between an inflow point at one end of the pipe, and another inflow point 10 m from the boundary between the insulated distribution line and the non-insulated transfer line.

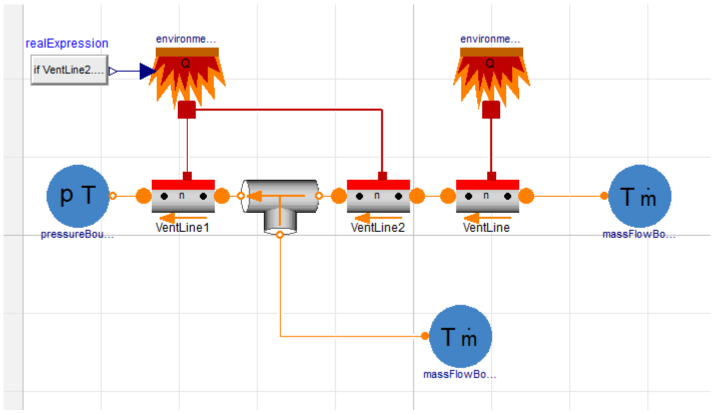


Figure 2.1 The relief system as set up in the Dymola graphical layer.

The pipes had a thickness of 3 mm and were made of stainless steel. The friction losses were found through the Darcy-Weisbach formulation with a friction loss coefficient found to be $\lambda = 0.03$. The heat flow at the boundary of the flowing helium and the pipe wall was solved with the Dittus-Bölter equation. Further, the first part with a length of 320 consisted of vacuum insulation where the heat flow from the ambience to the pipe was presumed to be 1 W/m^{2}K, while the last part of 80 m was non-insulated and had a heat flow of 68000 W/m^{2}K. The initial and ambient conditions are 300 K at a pressure of 1.05 bar.

2.2 Cool-down and Warm-up of the Distribution Line

Cool-down

In order to allow for the accelerating cavities to benefit from superconductivity, the entire cryogenic distribution line needs to be cooled down to the working temperature of 4.5 K. This will be studied for a few different cases. The task description for this Master thesis states that the model should involve cryogenic engineering, material properties, fluid properties of the cooling medium (helium), and be based on the current cryogenic system design. There should be an emphasis on considerations of the changing material and fluid properties from 300 K down to 4.5 K. In addition, the developed models should be designed so that materials, flow rates, and component designs can be easily used in new analyses.

The first problem is to simulate the cool-down of only the distribution line, meaning that no cryomodules or jumper connections are cooled at this point. The valves in the valve boxes connecting to the jumper connections and cryomodules are hence closed. These results are found in the second section of the results chapter, and are specified according to the pipe and component dimensioning below. Furthermore, the heat loads from radiation and conduction of the supports have been considered, while the conduction along the pipe walls themselves is not used in the simulations, due to the small temperature differences at every time instant of adjacent pipe sections. The helium inflow in the cold helium circuit is specified to be 99.3 g/s at an inflow temperature of 5.2 K. The thermal shield circuit has an inflow of 75.5 g/s at 40 K [15].

The cool-down is considered to be finished when the thermal shield has reached 40.5 K at its warmest point, and the cold helium circuit has reached 5.1 K at its warmest point. For the simulation, each component used $n = 100$ nodes, the time step size was set to $t_{step} = 2000$, and the tolerance was 10^{-8}.

Warm-up

This is also done in reverse, where the complete distribution system is at operating temperature, and it is then, using warmer helium, warmed back up to ambient temperature (300 K). The helium inflow is kept at 49.0 g/s for the cold helium circuit and maintained at 75.5 g/s in the thermal shield circuit [15].

The distribution system is considered warm when the temperature has reached 295 K at the coldest point. The node number, step size, and tolerance were set to the same numbers as for the cool-down.

The warm-up procedure is also simulated for static conditions, where the continuous helium supply is stopped for some reason. This causes a warm-up of the cryogenic system due to the so called static heat leaks, meaning radiation and to some extent conduction from the supports and connections to vacuum barriers.

2.3 Cool-down and Warm-up of One Cryomodule

Here, the cool-down is simulated for an individual cryomodule, with the masses and dimensions according to the section below. The helium inflow into the cryomodule is varied between 1.0 g/s and 3.0 g/s to find how the cool-down time changes with the supply of helium. Just as with the simulations of the distribution system, this is also run in reverse to find the warm-up times of the cryomodules when warmer helium is flowing in.

The superconducting cryomodules have four main components. The stainless steel piping, the helium tank made of titanium, the niobium cavities immersed in the helium tanks, and the thermal shields made from copper. A schematic figure of a cryomodule and how it is attached to the cryogenic distribution line is seen in Fig. 2.2. As is shown, each cryomodule contains four helium tanks with one cavity in each, and the thermal shield and helium supply pipes run once along the short side and then along the long side of the module. For the 13 spoke cryomodules, each module only contains two tanks with cavities.

The masses of the different materials for the four main components in an individual cryomodule are found in Table 4.5. The piping in the helium and thermal shield circuits has the dimension specified for the jumper connections in Table 4.3; that is, DN10.

Cool-down

Initially, the cryomodule is at 300 K and ambient pressure. Helium flows in according to the mass flow rates mentioned above, causing the components inside the cryomodule to cool with different rates. The inflow to the thermal shield circuit has a temperature of 40 K and the inflow to the cold helium circuit the temperature was set to 5 K. The cryomodule is considered cool when the temperature of the niobium cavities have reached 7 K.

A similar but less detailed study of the cool-down of a single spoke cryomodule has been carried out by the manufacturer at Institut Physique Nucléaire in Orsay, France. The helium mass needed for this process is seen in Fig. 2.3 and is given as at least 37.7 kg [16]. This is used as a comparison with the results given in the result chapter.

Warm-up

For this simulation, the cryomodule has an initial temperature of 5 K for the helium supply pipe, the titanium tank, and the niobium cavity. The thermal shield and its piping is at 40 K. Warm helium at a temperature of 300 K enters into both circuits and gradually warms the cryomodule up. The inflows are varied in the same fashion as for the cool-down, between 1.0 g/s and 3.0 g/s. The module is considered warm when it has reached a temperature of 290 K in the coldest point.

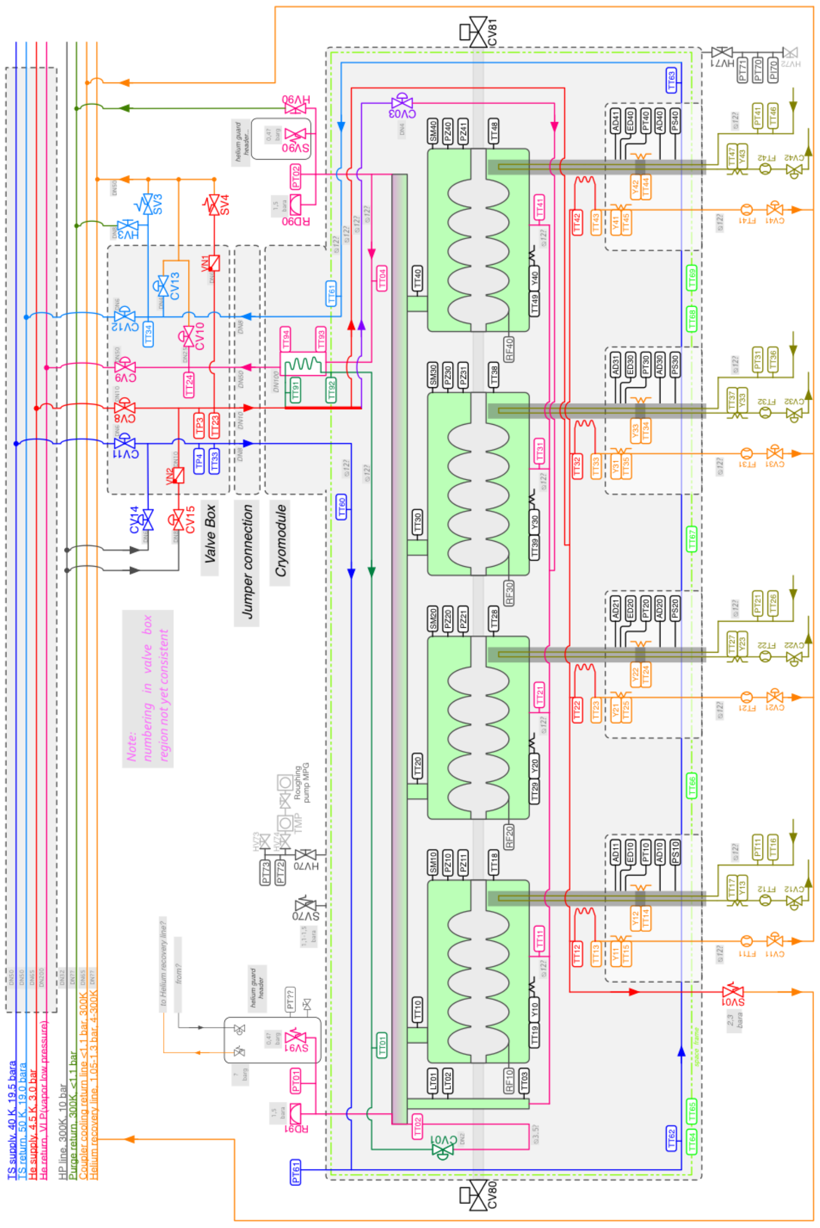

TS supply, 40 K, 19.5 bara
TS return, 50 K, 19.0 bara
He supply, 4.5 K, 3.0 bar
He return, VLP(vapor low pressure)
HP line, 300K, 10 bar
Purge return, 300K, <1.1 bar
Coupler cooling return line <1.1 bar, 300K
Helium recovery line, 1.05-1.3 bar, 4-300K
Valve Box
Jumper connection
Cryomodule
Note: numbering in valve box region not yet consistent
CV80
CV81
CV01
CV03
SV01
SV90
SV91
SV70
RD90
RD91
Roughing pump MPG
TMP
helium guard header
to Helium recovery line?
from?

Figure 2.2 An elliptical cryomodule with its component and attachment to the distribution line. Note the fours titanium tanks containing the superconducting cavities immersed in liquid helium. Courtesy of Dr. John G. Weisend II.

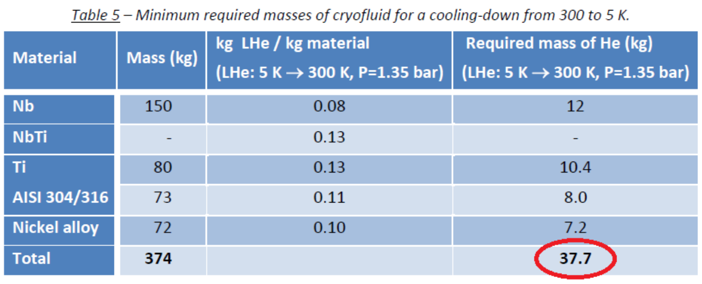


Table 5 – Minimum required masses of cryofluid for a cooling-down from 300 to 5 K.

Material	Mass (kg)	kg LHe / kg material (LHe: 5 K → 300 K, P=1.35 bar)	Required mass of He (kg) (LHe: 5 K → 300 K, P=1.35 bar)
Nb	150	0.08	12
NbTi	-	0.13	-
Ti	80	0.13	10.4
AISI 304/316	73	0.11	8.0
Nickel alloy	72	0.10	7.2
Total	**374**		**37.7**

Figure 2.3 The helium mass needed to cool down one spoke cavity. Source: [16]

2.4 Model Comparisons

In order to be able to trust the results generated from the simulations in this project, and to benchmark the models as valid, a robust cross-check has been made between the Dymola results and controlled calculations using Excel with the HePak [17] macro and two published papers with experimental results and simulations of the helium relief system for the Large Hadron Collider (LHC) at CERN. The Modelica components that were created and used in this section are all described in the Models chapter under Modelica Components while the descriptions of the standard Modelica components are found in [18].

Validation with Excel and HePak

The model for the Excel calculations consisted of a `CryoPipeDWReynolds` attached to a `Modelon.ThermoFluid.Sources.Environment_Q` for a constant heat inflow of 1 W/m^{2}K (vacuum insulation). On one side there was a `Modelon.ThermoFluid.Sources.MassFlowBoundary` with a helium mass flow of 0.5 kg/s, 5 K, and on the other side there was a `Modelon.ThermoFluid.Source-s.PressureBoundary` with the exit pressure of 1.05 bar. A picture of the Dymola component is seen in Fig. 2.4. The initial temperature and pressure in the pipe were 300 K and 1.05 bar, respectively. The pipe dimension was DN150 (168.28 mm), the pipe thickness was 3 mm, and it had a total length of 50 m. The simulation was then run for 250 seconds in Dymola. The pipe was modeled with $n = 50$ sections, each then corresponding to one meter of pipe, and the time discretization was $t_{step} = 0.05$ seconds per step.

For the Excel calculations, the same parameters have been used. It should also be pointed out that to find the velocity and properties of helium at a certain point, the pressure difference and absolute pressure of helium was taken from the Dymola simulations, in addition to the instantaneous temperature. This was then used to derive the other properties.

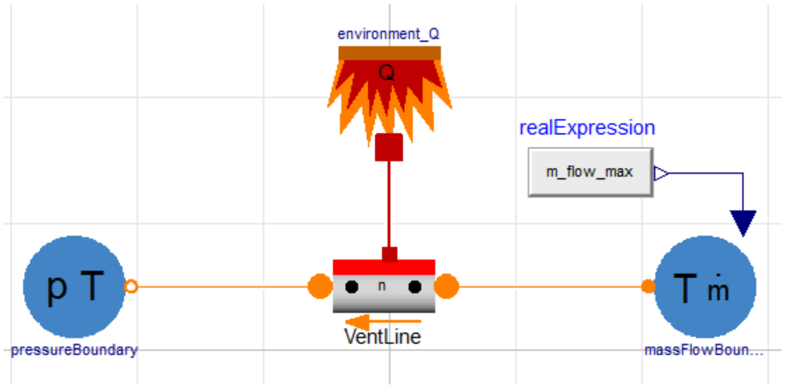


Figure 2.4 The Dymola graphical layer of the set-up used for the Excel validation simulations.

Comparison with Experimental Data from Nitrogen Experiment

This experiment was set up at the Wroclaw University of Technology with the purpose to validate a numerical model for the simulation of the helium relief system at LHC. A detailed description is found in [19], and it consists of a quick-opening valve and a copper pipe of length 25 m. The pipe has an inner diameter of 10 mm and is spiraled (for saving space) 13 laps in a coil with a diameter about 70 times that of the pipe's inner diameter. The friction losses due to the bent pipe are then not drastic, but still need to be considered [19]. The other side of the pipe is open to the atmosphere (1.05 bar, 300 K).

The experimental procedure is carried out by opening the valve, which connects the copper pipe to a pressurized nitrogen tank. Nitrogen is then flowing into the system with a temperature of 130 K and an initial pressure of 10 bar. This causes a decrease of the system pressure and pipe wall temperature as time passes. The attached sensors in the experiment also allow for measurements of the mass flow rate, and the three mentioned system properties are compared interchangeably in this validation. Due to the rapid cooling of the copper pipe, there will appear frosting on the outside of the copper, causing the heat transfer to go down from a value of 1700 W/m^{2}K at time t = 0 s to 500 W/m^{2}K at time t = 80 s. This procedure is given in [19]. The friction losses due to the spiraling of the pipe was found to have a Darcy-Weisbach friction loss coefficient (see Eq. (4.2)) of $\lambda = 0.5$.

Comparison with Experimental Data from Helium Relief System

The second paper that has been used for validation is also considering the LHC relief system, and is found in [20]. This one is a numerical simulation of a fast-moving situation where the magnet sector of the accelerator experiences a resistive transition, a so called *quench*, going from superconductivity to normal conduction, which causes large amounts of heat to be transfered to the cold helium stored in the system. The helium is heated up and flows through a pressure valve to the quench line, which is a 400 m long DN200 (219.08 mm) stainless steel pipe leading to four buffer volumes of 250 m^2 each. Initial values in the quench line and buffer volumes are 300 K and 1.3 bar. The helium inflow starts at 6 bar at 9 K, and while the inflow temperature decreases by a mere 0.3 K over the simulation time of 10 s, the inflow pressure gradually goes down to 4 bar after 10 s. These in-parameters for pressure and temperature are used for the Dymola simulation. Also in this simulation, the Dittus-Bölter equation is used for the heat transfer and the Darcy-Weisbach formulation is applied to the friction losses, with a friction coefficient value of $\lambda = 0.06$. The insulation of the pipe was taken to be vacuum insulation with a heat flow of 1 W/m^2.

3

Methods

The primary tools used for the modeling and simulations in this project are Dymola, which is a modeling tool for the Modelica language, Matlab, an environment for numerical computation and visualization, and Excel. While the modeling and main simulations were done in Dymola, Matlab has been used for additional side scripts and numerical calculations as well as for plotting some of the results. Excel was used for validation of the Dymola models, and has also been used for retrieving material properties and collective plotting of several results at the same time. Dymola, Modelica, and Matlab are described below, while it is assumed that Excel is known to the reader of this report and the seeker of more information is pointed to Refs. [21] and [22].

3.1 Dymola and Modelica

Dymola

Dymola is the simulation environment used as the main software in this project. It is maintained and developed by Dassault Systèmes AB, which is a small software company located in Lund. Dymola started off as a PhD project in 1978, which was later developed into the company Dynasim AB in 1991. Dynasim AB was bought by the French company Dassault Systèmes in 2006 and the idea was to implement Dymola as a tool within the multi-platform CAD software suite CATIA. It is now available both through the integration with CATIA and as a standalone product [23, 24]. For the project described in this thesis, the latter choice was used.

Dymola is a multi-engineering software, meaning that it can be used for modeling within a variety of different engineering fields, and even combining them into models that seamlessly connect the different areas into each other. It uses the Modelica modeling language, described below, and has combined the basic code (text) layer with a graphical modeling tool [25]. This allows for easy and quick modeling where the graphical layer gives an overview of the model or component and connections and flow schemes are visible, while the text layer allows for code writing and equation manipulation. This is a strong feature of the software, where the two layers are typically used interchangeably.

Modeling with Dymola typically involves drag-and-drop modeling with custom-made components from one or more of the available libraries. This is combined with setting the parameters and initial values of the components and connecting them in the way of choice. The added components then also appear in the text layer, where more detailed manipulation can be made together with the addition of features or equations. A picture of the modeling environment in Dymola is seen in Figs. 3.1 and 3.2, where the graphical layer is displayed in the first one and the text layer is seen in the second one.

Modelica

Modelica is an object-oriented modeling language. Its main feature, that also separates it from most other modeling languages, is that it is *acausal*. This means that the order of equations, and what variable is on which side of the equal sign, is irrelevant. Simulations are started by setting the initial values and then letting the simulation engine solve for the appropriate variables and calculate how the behavior of the system proceeds. This is also called that the language is declarative. The structure of the code is oriented so that individual components and their properties are written as separate objects within the more complex systems [26].

It should be noted that Modelica is a modeling language, rather than a traditional programming language. Models (programs) created are therefore first translated into standard C code, which is then being executed to obtain the results [27].

The Modelica language is free and is being maintained by the Modelica Association, which is a non-profit organization. It started in 1996 as an attempt by Hilding Elmqvist [24] to create a modeling language that was object-oriented and at the same time had the possibility of reusing code and find a more generic standard to model technical systems.

Dymola and Modelica

Using Dymola and Modelica together allows for an efficient way to build models and to create systems where much of the code can be reused for other components than the original one. In Modelica, the components are built, default parameters are set, and the equations specifying the behavior are given. This is done either graphically in the Dymola environment, in Modelica code, or combining both of them.

In Dymola, the parameters for the specific simulation, the initial values, and the simulation setup - meaning simulation time, step size, numerical solver method, and tolerance - are set. Plotting of the results is then made in the Dymola simulation environment, but the results can also easily be exported to other software, such as Matlab, for analysis and treatment.

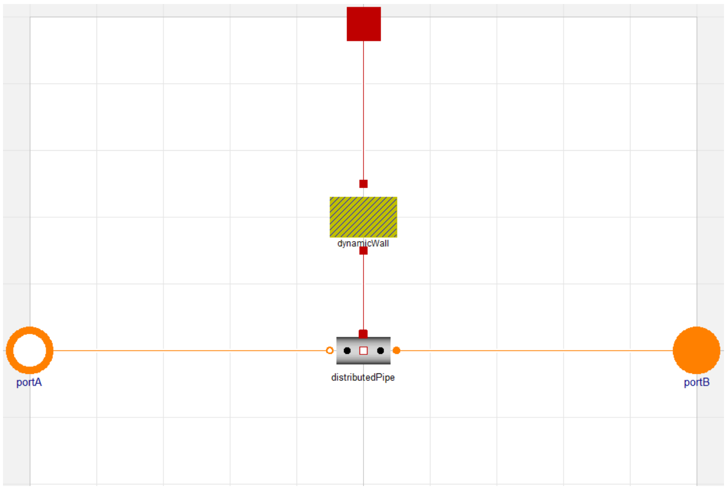


Figure 3.1 The Dymola graphical layer in the modeling environment.

The choice of Dymola and Modelica as the main simulation tools was based on previous development of similar models at the Target Division at ESS using these softwares. In addition, as Modelon AB is located in Lund, only a few hundred meters from the offices of ESS, the help and support provided was viewed as an advantage that could expand into future modeling tasks for ESS. The quick interfacing with Matlab, described below, was also a feature that supported the use of Dymola.

3.2 Matlab

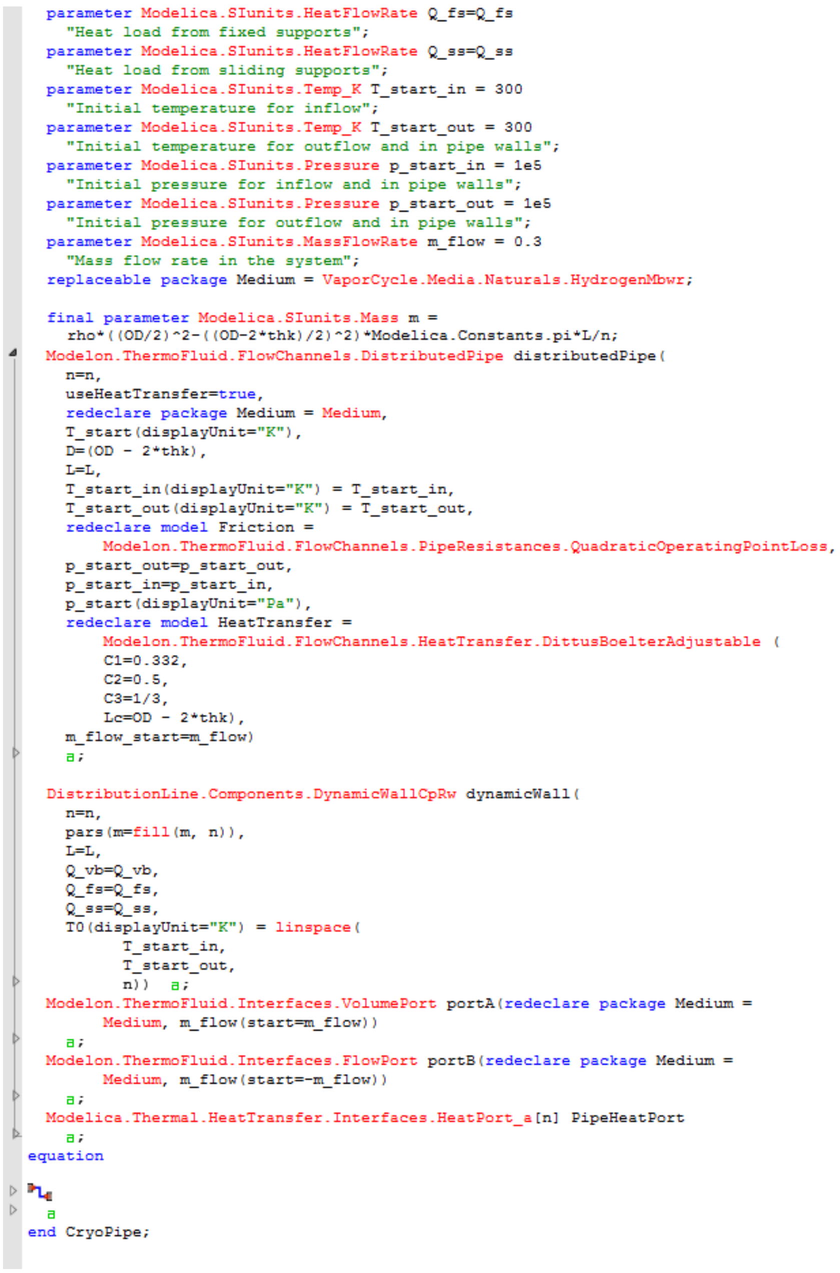
Matlab is an environment for numerical computation and programming. It is a high level language and its strength is within vector and matrix algebra, together with an extensive visualization toolbox. The name is an acronym for matrix laboratory and is developed by MathWorks Inc., which is an American company specialized in mathematical computing [28, 29].

Matlab was first started to be developed in the ’70s by Cleve Moler at the University of New Mexico. He used it as a tool to be able to use Fortran packages without having to learn Fortran itself, but after a visit by Moler to Stanford University, its possibilities as a commercial software was pointed out. Moler was joined by Jack

Little and Steve Bangert and together they formed MathWorks in 1984. The software was rewritten in C and the libraries have been updated since then [29].

Matlab has a wide possibility to be interfaced with other software and can call C or Fortran functions. IT can also call a variety of libraries written in e.g. Java or ActiveX [28, 29]. In this project, Matlab has been used to analyse data from Dymola, and for plotting some of the results. It has also served as a practical environment for developing scripts and functions who's results have been copied into some of the Dymola components.

For this project, it was chosen as an easy-to-use tool for quikc scripting at the same time as it could be interfaced with Dymola. Matlab has been a returning software within the course work in the studies at Lund Institute of Technology and is therefore also passed the initial adjustment and learning process for new software.



```
    parameter Modelica.SIunits.HeatFlowRate Q_fs=Q_fs
      "Heat load from fixed supports";
    parameter Modelica.SIunits.HeatFlowRate Q_ss=Q_ss
      "Heat load from sliding supports";
    parameter Modelica.SIunits.Temp_K T_start_in = 300
      "Initial temperature for inflow";
    parameter Modelica.SIunits.Temp_K T_start_out = 300
      "Initial temperature for outflow and in pipe walls";
    parameter Modelica.SIunits.Pressure p_start_in = 1e5
      "Initial pressure for inflow and in pipe walls";
    parameter Modelica.SIunits.Pressure p_start_out = 1e5
      "Initial pressure for outflow and in pipe walls";
    parameter Modelica.SIunits.MassFlowRate m_flow = 0.3
      "Mass flow rate in the system";
    replaceable package Medium = VaporCycle.Media.Naturals.HydrogenMbwr;

    final parameter Modelica.SIunits.Mass m =
      rho*((OD/2)^2-((OD-2*thk)/2)^2)*Modelica.Constants.pi*L/n;
    Modelon.ThermoFluid.FlowChannels.DistributedPipe distributedPipe(
      n=n,
      useHeatTransfer=true,
      redeclare package Medium = Medium,
      T_start(displayUnit="K"),
      D=(OD - 2*thk),
      L=L,
      T_start_in(displayUnit="K") = T_start_in,
      T_start_out(displayUnit="K") = T_start_out,
      redeclare model Friction =
          Modelon.ThermoFluid.FlowChannels.PipeResistances.QuadraticOperatingPointLoss,
      p_start_out=p_start_out,
      p_start_in=p_start_in,
      p_start(displayUnit="Pa"),
      redeclare model HeatTransfer =
          Modelon.ThermoFluid.FlowChannels.HeatTransfer.DittusBoelterAdjustable (
          C1=0.332,
          C2=0.5,
          C3=1/3,
          Lc=OD - 2*thk),
      m_flow_start=m_flow)
      a;

    DistributionLine.Components.DynamicWallCpRw dynamicWall(
      n=n,
      pars(m=fill(m, n)),
      L=L,
      Q_vb=Q_vb,
      Q_fs=Q_fs,
      Q_ss=Q_ss,
      T0(displayUnit="K") = linspace(
            T_start_in,
            T_start_out,
            n))  a;
    Modelon.ThermoFluid.Interfaces.VolumePort portA(redeclare package Medium =
          Medium, m_flow(start=m_flow))
      a;
    Modelon.ThermoFluid.Interfaces.FlowPort portB(redeclare package Medium =
          Medium, m_flow(start=-m_flow))
      a;
    Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a[n] PipeHeatPort
      a;
  equation

    a
  end CryoPipe;
```

Figure 3.2 The Dymola text layer in the modeling environment.

4 Models

Here, the background of the modeling is introduced. The major theoretical concepts are first presented, followed by the specifications of the Dymola models. The last section of the chapter specifies the dimensions of the main components as well as the pipes for the different parts of the system that has been modeled. The theory section gives an insight in how flow and heat transfer are dealt with in the models and how the material properties vary with changing temperature. In addition, the radiation models and the concept of multi-layer insulation are described. The Modelica component section gives a brief description of every component that has been used for the simulation of the results.

4.1 Theory

In this section, the physical equations behind the different models are described. As the models require attention to be paid to thermodynamics, material properties, and fluid dynamics, this theory section will comment on the assumptions and physical behavior that are not by default implemented into similar simulations as these performed here. All the approaches mentioned below will be applied where applicable unless stated differently for the models.

Friction Coefficient and Pressure Characteristics

It is known that the possible mass flow rate of helium from pressure differences is not arbitrary. It is rather highly dependent on the pressure difference together with the temperature of the helium. As it gets colder, the density rises and thus the mass that can flow through the pipes is increased. To explore the possibilities for mass flow rates given by this relation, the use of *pressure characteristics* were considered [30, 31, 32]. From a discretely increased mass flow rate, together with the inflow temperature and pressure, the arising pressure drop could be determined for a certain pipe geometry. The appearing characteristics could then be studied to find the possible mass flow rate for a specific pressure drop of interest. The procedure uses the Colebrook-White equation [33], which is an iterative procedure to find the Darcy-Weisbach friction coefficient for a flow in the turbulent regime [34], given by [35]

$$\frac{1}{\sqrt{\lambda}} = -2 \cdot \ln\left(\frac{2.51}{Re \cdot \sqrt{\lambda}} + \frac{r}{3.72 \cdot d_h}\right) \tag{4.1}$$

where λ is the dimensionless Darcy-Weisbach coefficient, Re is the dimensionless Reynold's number, r is the roughness of the pipe surface in m, and d_h is the (hydraulic) diameter in m. By iterating with Eq. (4.1) to find the appropriate λ, and then plug it into the friction modified Bernoulli equation, sometimes referred to as the Darcy-Weisbach equation for friction loss [35, 36],

$$\Delta P = \lambda \cdot \frac{L}{d_h} \cdot \frac{v^2}{2} \cdot \rho \tag{4.2}$$

the necessary pressure drop, ΔP, is found. In Eq. 4.2, L is the pipe length in m, v is the flow velocity in m/s, and ρ is the density in kg/m^3. The pressure drop can then be plotted against the mass flow rate to yield the characteristics for selected temperatures, as seen can be as an example in Fig. 4.1 for a 300 m long DN65 pipe. The maximum mass flow rate as a function of temperature, as derived by the pressure characteristics, is seen in Fig. 4.2. The Colebrook-White equation, (4.1),

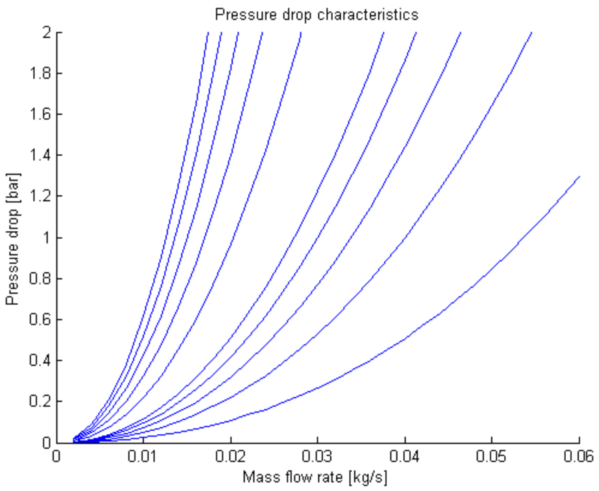


Figure 4.1 Characteristics for helium pressure drop as a function of mass flow rate for various temperatures and a pipe of nominal diameter DN65 and length 300 m.

together with the Bernoulli equation, (4.2), are implemented in the friction loss models in the pipes developed in Dymola for this project, and the mass flow rate and pressure relations make use of this relation for all applicable cases.

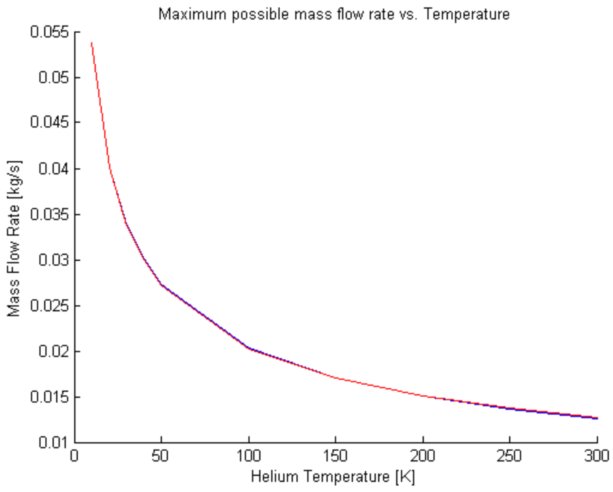


Figure 4.2 Possible mass flow rate as a function of helium temperature. Derived from the characteristics in Fig. 4.1. The red line shows points from the different curves in Fig. 4.1, and the blue line is a curve fit for these.

Choked Flow

Together with the pressure characteristics approach to verify possible mass flow rates, the phenomenon of *choked flow* was also examined. This was done using the following formula [37, 38, 39]

$$\dot{m} = \frac{A \cdot P}{\sqrt{T}} \sqrt{\frac{\gamma}{R_{He}}} \cdot \left(\frac{\gamma+1}{2}\right)^{\frac{\gamma+1}{2(\gamma-1)}} \tag{4.3}$$

where

$$\gamma = \frac{C_P}{C_V} \tag{4.4}$$

Here A is the pipe area in m^2, P is the pressure in Pa, T is the temperature in K, and $R_{He} = 2077$ J/kg·K is the gas constant for helium. C_P and C_V are the specific heat capacities in J/kg·K with constant pressure and constant volume, respectively. A plot for this, also using a DN65 pipe, can be seen in Fig. 4.3.

Heat Transfer Between Pipe Wall and Helium

For describing the heat flow that takes place on the boundary between the pipe walls and the flowing medium, the commonly accepted explicit equation of Bittus-Bölter [40, 12, 41] for finding the Nusselt number is implemented into the pipe models whenever there is flow present. This is motivated by the presence of forced

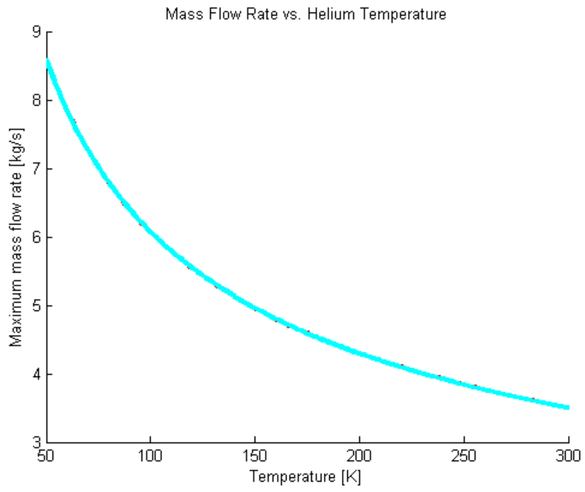


Figure 4.3 Maximum mass flow rate possible to avoid choked flow in a DN65 pipe, as a function of helium temperature.

convection in the pipe [9], and the equation used is the following

$$N_u = 0.0023 \cdot R_e^{0.8} \cdot P_r^{0.4} \tag{4.5}$$

where N_u is the Nusselt number, R_e is the Reynold's number, and P_r is the Prandtl number [41]. All three variables are dimensionless. This is applicable in the case where the lateral heat fluxes are dominant over the streamlined ones, which is the case in the pipe flow in these models [34]. In addition, the following inequalities need to hold [41]

$$\begin{aligned} &R_e > 10^4 \\ &0.7 < P_r < 160 \\ &L/D > 60 \end{aligned} \tag{4.7}$$

L and D are the length and diameter of the pipe in arbitrary units.

Solid Material Properties

For the vast temperature interval that is spanned in the cooling process, the solid materials show some noticeable change in material properties. In order to create a realistic model of the heat transfer from fluid to solid and solid to solid, the temperature dependence of specific heat capacity and thermal conductivity was taken into account.

Stainless Steel
For the stainless steel used in the cryogenic piping (SS304L), the data was taken from an Excel program called Cryocomp, created by Cryodata Inc. [42]. It is found that the specific heat capacity, in J/kg·K, of SS304L as a function of temperature has the following appearance

$$\begin{aligned} C_P^{ss} =& -0.00000000115115 \cdot T^5 + 0.00000112374 \cdot T^4 \\ & -0.000392079 \cdot T^3 + 0.0522538 \cdot T^2 + 0.141301 \cdot T \end{aligned} \tag{4.9}$$

The thermal conductivity (W/m·K) of stainless steel obtained from the same program is given by the formula

$$k_{ss} = 8 \cdot 10^{-7} \cdot T^3 - 0.0006 \cdot T^2 + 0.1477 \cdot T - 0.4952 \tag{4.10}$$

In the equations in this and the following section, T denotes the (average) temperature of the material.

Aluminum, Copper, Titanium, and Niobium
For the specific heat capacities of aluminum, copper, titanium, and niobium, the Selected Cryogenic Data Notebook from Brookhaven National Laboratory [43] was consulted, and from the tabulated values, appropriate formulae were obtained using a polynomial curve fit in Excel. The formulae obtained were:

$$\begin{aligned} C_P^{Al} =& -5.0858382990 \cdot 10^{-9} \cdot T^5 + 4.4933166438 \cdot 10^{-6} \cdot T^4 \\ & -0.0014348060969 \cdot T^3 + 0.18473605304 \cdot T^2 \\ & -3.4580884187 \cdot T + 10.842981412 \end{aligned} \tag{4.12}$$

$$\begin{aligned} C_P^{Cu} =& 1.2682110150 \cdot 10^{-11} \cdot T^6 - 1.4399163352 \cdot 10^{-8} \cdot T^5 \\ & + 6.3296511850 \cdot 10^{-6} \cdot T^4 - 1.3291319283 \cdot 10^{-3} \cdot T^3 \\ & + 0.12535751222 \cdot T^2 - 1.8004787189 \cdot T \\ & + 5.2456099913 \end{aligned} \tag{4.14}$$

$$\begin{aligned} C_P^{Ti} =& -3.5147820378 \cdot 10^{-9} \cdot T^5 + 3.0057506888 \cdot 10^{-6} \cdot T^4 \\ & -9.2045002329 \cdot 10^{-4} \cdot T^3 + 0.11191552397 \cdot T^2 \\ & -1.7076312852 \cdot T + 4.3306205877 \end{aligned} \tag{4.16}$$

and

$$\begin{aligned} C_P^{Nb} =& 2.0226423917 \cdot 10^{-11} \cdot T^6 - 2.0451534634 \cdot 10^{-8} \cdot T^5 \\ & + 7.9550021963 \cdot 10^{-6} \cdot T^4 - 0.0014601740994 \cdot T^3 \\ & + 0.11758205083 \cdot T^2 - 1.2495904095 \cdot T \\ & + 3.0875766442 \end{aligned} \tag{4.18}$$

The thermal conductivities for aluminum, copper, and titanium were found through the NIST website [44]. They are are obtained through the following general function

$$\begin{aligned} k_m =& 10^{a+b(log_{10}T)+c(log_{10}T)^2+d(log_{10}T)^3+e(log_{10}T)^4+f(log_{10}T)^5} \\ & {}^{+g(log_{10}T)^6+h(log_{10}T)^7+i(log_{10}T)^8} \end{aligned} \tag{4.20}$$

and the values for constants *a* - *i* are tabulated in Table 4.1 below.

Constant	Aluminum	Copper	Titanium
a	23.39172	1.8743	-2.398794842
b	-148.5733	-0.41538	8.970743802
c	422.1917	-0.6018	-29.19286973
d	-653.6664	0.13294	54.87139779
e	607.0402	0.26426	-59.67137228
f	-346.152	-0.0219	38.89321714
g	118.4276	-0.051276	-14.94175848
h	-22.2781	0.0014871	3.111616089
i	1.770187	0.003723	-0.270452768

Table 4.1 Table with the constants used in Eq. 4.20 for the different materials.

The thermal conductivity for niobium was not available through NIST, and was instead derived from tabulated values in [43]. Due to the curious behavior of the curve, it was split up into a low-temperature part up to 30 K, and a high-temperature part above 30 K. This was done for simplicity of the curve fitting, and the turn-over point of 30 K was chosen arbitrarily after inspection by eye. The formulae are stated below, in respective order of temperature region.

$$k_{Nb}^{lowT} = -0.11689520760 \cdot T^2 + 6.4368583162 \cdot T + 1.9644659451 \tag{4.21}$$

and

$$\begin{aligned} k_{Nb}^{highT} =& -3.8984111598 \cdot 10^{-10} \cdot T^5 + 3.9537530644 \cdot 10^{-7} \cdot T^4 \\ & - 1.5506337607 \cdot 10^{-4} \cdot T^3 + 0.029268800879 \cdot T^2 \\ & - 2.6465534609 \cdot T + 141.29752365 \end{aligned} \tag{4.23}$$

The Radiation Model

The thermal radiation that takes place between the different pipes in the cryogenic distribution line, together with that between pipes and the thermal shield, is quite a complicated problem that needs some special attention. The radiation model uses the Stefan-Boltzmann law of gray-body radiation [45, 46, 47],

$$\dot{Q}_{wc} = A_w E_w = \frac{1}{\frac{1}{\varepsilon_w} + \frac{1}{\varepsilon_c} - 1} A_w \sigma (T_w^4 - T_c^4) \tag{4.24}$$

where $\dot{Q}_{wc}$ is the heat flow in W/m^2 from the warm (w) to the cold (c) body, A_w is the area of the radiating body in m^2, ε_w is the emissivity of the warm (radiating) body and is dimensionless, ε_c is the emissivity of the cold (receiving) body, σ is the Stefan-Boltzmann constant in W/m^2·K^4, T_w is the temperature of the warm body in K, and T_c is the temperature of the cold body in K. In addition to the formula above, a combination of *view factors* (also known as F factors, shape factors, configuration factors, or form factors) have been used that take into account the geometric situation of the distribution pipe [46, 48, 49]. This means that certain bodies only "see" certain sections of the other bodies, whereas some sections are invisible and hence do not take part in the interacting thermal radiation. By including the view factors into Eq. (4.24), we have that [50]

$$\dot{Q}_{wc} = A_W F_{wc} E_w = \frac{1}{\frac{1}{\varepsilon_w} + \frac{1}{\varepsilon_c} - 1} A_w F_{wc} \sigma (T_w^4 - T_c^4) \tag{4.25}$$

The geometric situations that arise are those calling for the following view factor approximations [48, 51, 52, 46]

(a) Concentric cylinders of infinite length

(b) A long cylinder to itself when interior cylinders are present

(c) An infinitely long cylinder to non-concentric cylindrical enclosure

(d) Infinite (very long) parallel cylinders of different radii

The different geometries are displayed in Fig. 4.4. For all formulae stated below, subscript 1 refers to the emitting body while subscript 2 refers to the receiving body.

Approximation (d) is used for the radiation between the internal pipes of the distribution line. That is, between the helium supply, vapor low pressure (VLP), and thermal shield supply and return pipes. The formula used for this is [51]

$$\begin{aligned} F_{12} = & \frac{1}{2\pi} \left[C^2 - (R+1)^2\right]^{1/2} - \left[C^2 - (R-1)^2\right]^{1/2} \\ & + (R-1) \cdot \arccos\left(\frac{R-1}{C}\right) \\ & - (R+1) \cdot \arccos\left(\frac{R+1}{C}\right) \end{aligned} \tag{4.27}$$

where $R = r_2/r_1$, $C = 1 + R + S$, and the r refer to radii. Approximations (b) and (c) were used for the thermal shield radiating to the four internal pipes. It should be noted, however, that the default (c) view factor is for the inner pipe radiating to

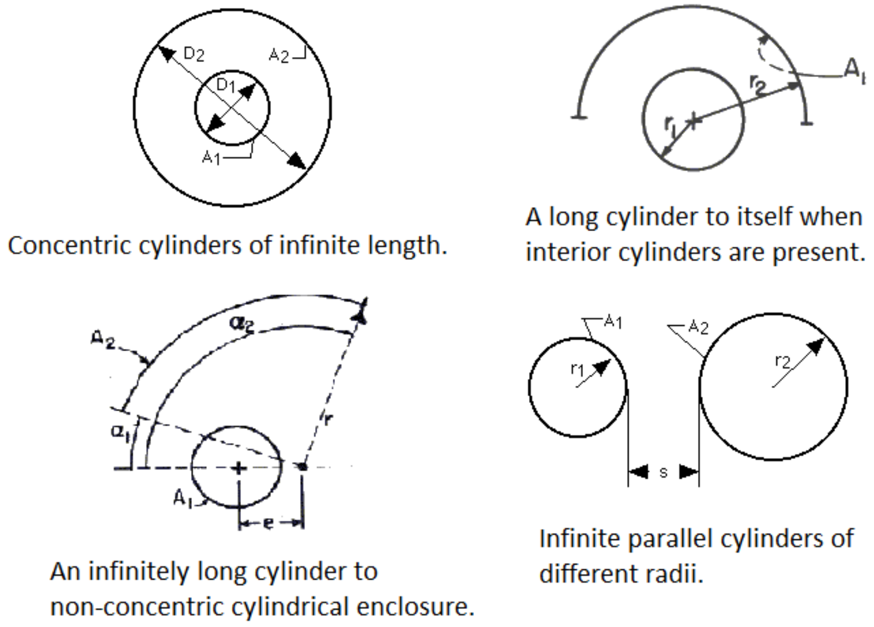

D2
A2
D1
A1
Concentric cylinders of infinite length.
A1
r2
r1
A long cylinder to itself when interior cylinders are present.
A2
α2
α1
r
A1
e
An infinitely long cylinder to non-concentric cylindrical enclosure.
A1
A2
r1
r2
s
Infinite parallel cylinders of different radii.

Figure 4.4 The four different cases for radiation in the distribution and transfer lines. The top left drawing corresponds to case (a), top right to (b), bottom left to (c), and the bottom right drawing corresponds to (d).

an enclosing pipe. The situation here is the opposite, and the formulae will take the following form, respectively [51],

$$F_{11} = 1 - \frac{2}{\pi}\left[(1-R^2)^{1/2} + R \cdot \arcsin(R)\right] \tag{4.28}$$

where $R = r_1/r_2$, and

$$\begin{aligned} F_{12} = &\frac{1}{2\pi}\left[\frac{\alpha_2-\alpha_1}{2} + \arctan\left(\frac{1+E}{1-E}\cdot\tan(\alpha_2/2)\right)\right. \\ &\left. -\arctan\left(\frac{1+E}{1-E}\cdot\tan(\alpha_1/2)\right)\right] \end{aligned} \tag{4.30}$$

where $E = e/r$, e is the distance between center points, r is the radius of the enclosing cylinder, α_1 is the angle from the internal pipe's horizontal position to the start of the enclosing cylinder, and α_2 is the total surrounding angle of the enclosure [51]. Note that for an entirely surrounding cylindrical enclosure, which is the case here, the term $\frac{\alpha_2-\alpha_1}{2} = \alpha_2/2 = \pi$, and $\tan(\alpha_1/2) = \tan(0) = 0$.

Approximation (a) calls for a slightly different treatment due to the character of the geometry. The view factor equations are as follows [47, 48]

$$\begin{aligned} F_{22} &= 1 - D_1/D_2 \\ F_{12} &= 1 \\ F_{21} &= D_1/D_2 \end{aligned} \tag{4.32}$$

with D_1 being the inner diameter and D_2 the outer diameter, which further yields the following formula for the complete form of A_1F_1 including emissivity [47, 48]

$$\tilde{A}_1\tilde{F}_1 = \frac{2\pi r_1 L}{\frac{1}{\varepsilon_1} + (\frac{1}{\varepsilon_2} - 1) \cdot \frac{r_1}{r_2}} \tag{4.33}$$

However, the case in the cryogenic system is opposite the standard form in the literature, which means that the index number of the radii are switched to adjust to that the outer cylinder is emitting to the inner cylinder. Thus, the formula in the models has the form

$$A_1F_1 = \frac{2\pi r_2 L}{\frac{1}{\varepsilon_2} + (\frac{1}{\varepsilon_1} - 1) \cdot \frac{r_2}{r_1}} \tag{4.34}$$

It is assumed that all internal pipes see half of all the other pipes, according to the placements in Fig. 4.5. In addition, the radiation of the thermal shield to itself and the radiation from the thermal shield to the internal pipes consider that all pipes are visible and affect the thermal shield radiation. A more detailed discussion on how complex geometries affect the view factors and radiative heat transfer can be found in [53].

Multi-Layer Insulation

The thermal radiation between two parallel plates is given by the formula [54]

$$q = \frac{\varepsilon_1\varepsilon_2}{\varepsilon_1 + \varepsilon_2 - \varepsilon_1\varepsilon_2} \tag{4.35}$$

If both plates are of the same material, and the emissivity is low ($\varepsilon_1 = \varepsilon_2 << 1$), this reduces to simply

$$q = \frac{\varepsilon}{2} \tag{4.36}$$

To make use of this relationship, it is common practice to wrap the process pipes with Multi-Layer Insulation (MLI) [12, 55, 47]. This drastically reduces the radiative heat loads, and an example of what this wrapping can look like is seen in Fig. 4.6. In the case of MLI of the pipes, repeated use of the formula above shows how the radiated heat transfer between the walls is efficiently reduced, and the following factor needs to be added to Eq. (4.25) [12, 56, 49]

$$q = \frac{\varepsilon}{2(N+1)} \tag{4.37}$$

where N is the number of insulation layers [12].

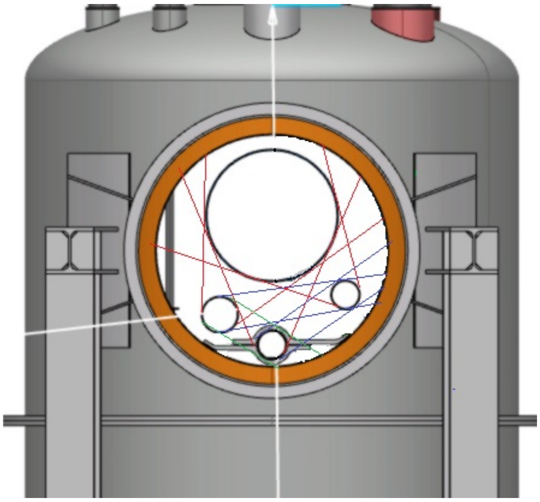


Figure 4.5 The placement of the four process pipes inside the distribution and transfer lines. Courtesy of Piotr Tereszkowski.

Figure 4.6 A cryostat wrapped with Multi-Layer Insulation. Source: [54].

The Conduction Calculations of Supports and Vacuum barriers

Even if relatively small, the heat loads from conduction through the fixed and sliding supports together with the vacuum barriers are still a non-negligible parameter in the calculations for the cool-down. Due to the similar appearance of all the supports and vacuum barriers, only a small set of equations will be used to define the heat loads from conduction through these solid attachments.

So called thermal conductivity integrals [12] in one dimension have been used, where the strong temperature dependence of the material properties is taken well into account. The conduction heat transfer is here described as

$$Q = \frac{-1}{\int_{x_1}^{x_2} \frac{dx}{A(x)}} \left[\int_{T_1}^{T_2} K(T)\, dT \right] = -G \cdot (\theta_2 - \theta_1) \tag{4.38}$$

where G is referred to as the geometry factor, $A(x)$ is the cross-sectional area, and θ_1 and θ_2 are the thermal conductivity integrals, defined as

$$\theta_i = \int_0^{T_i} K(T)\, dT \tag{4.39}$$

$K(T)$ is here the temperature dependent thermal conductivity. It should be noted that for the uniform geometries in this report, the geometry factor simply becomes $G = A/L$, where A is the cross-sectional area and L is the length of the section.

4.2 Modelica Components

For all components mentioned below, pipe geometries and initial parameters can be specified in the top component to allow for a high flexibility in the design and simulations. Further, the components are made independent of the others, and can be used, checked, tested, and simulated individually as well as collectively. The standard built-in components of the Dymola and Modelica libraries are explained in [57]. The components that were created in this project are found in the hierarchy diagram in Fig. 4.7. All components are displayed as pictures from the Dymola graphical layer in Appendix I. The Modelica source code is found in Appendix II.

CryoPipeDWReynolds

This component consists of a `Modelon.ThermoFluid.FlowChannels.DistributedPipe`, with a `DynamicWallSS` attached to simulate the pipe wall with its corresponding properties, such as mass, heat capacity, and thermal conductivity in the metal. The CryoPipe is included in all components that contain stainless steel piping, and the dimensions can be set through propagated parameters of the component. There is one inflow port and one outflow port for the medium, together with a heat port for thermal connections.

DynamicWallSS (...Nb/Ti/Al/Cu)

In order to obtain a temperature dependent specific heat capacity, C_p, and heat resistance, R_w, in the pipe walls in the `CryoPipe` component, the standard component `Modelon.ThermoFluid.Solids.DynamicWall` was modified in the `equation` section. The stainless steel component was then built upon to create corresponding components for modeling the niobium mass in the cavities, the titanium in the cryomodule helium tanks, the aluminum in the thermal shields, and copper for the usage in one of the validation models. The equations for C_p and R_w as given by Eqs. (4.9) and (4.10) were implemented for the `DynamicWallSS`, taking the instant temperature for each lump as an in-parameter. A corresponding procedure was made to find the modifications of the C_p and R_w of the other materials mentioned above, which makes use of Eqs. (4.12)-(4.12).

DistributionPipe

A component called `DistributionPipe` was written to model the entire distribution line, including the external envelope, or vacuum jacket, with the thermal shield and the four internal process pipes. This structure is seen in Fig. 4.8. The main problem here is to find a model which closely resembles the thermal radiation between the pipes and walls. The mathematical model of this is found in the theory section. The `DistributionPipe` component consists of four `CryoPipe` components, together with six `RadiationParallelPipes`, four `RadiationPipesInsidePipe`, one `RadiationConcentricPipes`, one `DynamicWallSS`, and one `DynamicWallAl`. The wall and pipe components correspond to the thermal shield and external envelope, and the four internal pipes, respectively. The radiation components model the thermal radiation in between them. In addition, the `DistributionPipe` is made available and adjustable within higher hierarchical structures by four flow ports, four volume ports, and one heat port.

TransferLine

The `TransferLine` is only a revision of the `DistributionPipe`. The component is adjusted to fit the specifications of the transfer line, that transports the helium from the cryoplant to the distribution line and back to the cryoplant with the liquefier, after the helium and thermal shield circuits are completed. As the dimensions of the pipes are the same as those for the distribution line, the main difference are the heat loads from the fixed and sliding supports together with the vacuum connections, that are now modified to be a total of 16, 16, and 8 respectively.

RadiationParallelPipes (...ConcentricPipes/PipesInsidePipe)

For the modeling of the radiation between the pipes and walls in the `Distributi-onPipe`, three different components are necessary to meet the requirements of the three different cases of thermal radiation. That is, between the internal pipes (`...ParallelPipes`), the thermal shield and the pipes (`...PipesInsidePipe`), and the external envelope and the thermal shield (`...ConcentricPipes`). The mathematics behind these cases is explained in the theory section, and the implementation into the radiation components is discretized and straight-forward. These components extend the `Modelica.Thermal.HeatTransfer.Components.Body-Radiation` with the additional necessary geometries of the pipes and the formulae for the radiation.

EndBox

The end box is where the internal process pipes ”turn around” and the supply pipes become return pipes. Further, there are a set of valves and circuits that can be adjusted for the different situations of cool-down, warm-up, or steady-state cooling of the cryogenic system. This component is modeled with one circuit for the thermal shield pipes, containing two 90 degree `Modelon.ThermoFluid.FlowResistances.FrictionLoss` and one `Modelon.ThermoFluid.Valves.ValveCompressible`. There are two circuits for the helium supply and VLP line pipes. The first circuit, used for initial cool-down, is similar to the thermal shield circuit and only takes friction losses and the on/off valve into account. The second circuit, on the other hand, also contains a heat exchanger, modeled by a `Modelon.ThermoFluid.FlowModifiers.SetTemperature`, that is used to increase the enthalpy of the returning helium. In addition, it also contains two friction loss components and an on/off valve. For the valves, there are two `Modelica.Blocks.Sources.RealExpression` to set the opening of the valves. Typically, one of the `RealExpression` is set to 1.0 for the two open valves, and the other is connected to the closed valve with a value of 0. There are also four flow ports for the helium inflow and outflow of the two circuits of the end box.

SimpleEndBox

The `EndBoxSimple` component is a simplified `EndBox`, where only the helium supply and VLP circuit for cool-down is present, and the valves are removed for numerical robustness in more complex cool-down and warm-up simulations. The thermal shield and cold helium circuits then only contain two 90 degree `Modelon.ThermoFluid.FlowResistances.FrictionLoss` in addition to the four flow ports.

CryoPlant

The `CryoPlant` component simply simulates the inflow from the cryoplant’s liquefiers and the outflow of the cryogenic system back into the cryoplant. It contains one `Modelon.ThermoFluid.Sources.MassFlowBoundary` for setting the mass flow rate and the temperature of the incoming helium into the thermal shield supply pipe. There is a `VaporCycle.Sources.TwoPhaseFlowSource` for setting the outflow enthalpy and mass flow rate of the helium supply pipe, which the allows for two-phase flow. It also contains a `Modelon.ThermoFluid.Sources.PressureBoundary` and a `VaporCycle.Sources.TwoPhasePressureSource` to simulate the outflow environment in terms of pressure and temperature. In addition, there are two volume ports (one for one-phase and one for two-phase) connected to the mass flow boundaries and two flow ports (one for one-phase and one for two-phase) connected to the pressure boundaries.

Cryomodule

This component simulated a cryomodule containing two loops: the cold helium loop and the thermal shield loop. Whereas the thermal shield loop contains two `Modelon.ThermoFluid.FlowResistances.FrictionLoss` and one `CMSteelPipingTwoPhase` and only considers one-phase flow (thermal shield stays above 40 K), the cold helium loop includes two-phase components. These are primarily a `CMSteelPipingTwoPhase`, a `TitaniumTankPipe`, and a `NiobiumCavity`. In between the piping and the thermal shield there are also one `RadiationParallelPipes` and two `RadiationPipesInsidePipe` to simulate the radiation between the different components. As the niobium cavity is immersed in the titanium tank, the heat port is connected directly to the fluid port in the titanium tank. For external connection, there are also one of each of `Modelon.ThermoFluid.Interfaces.FlowPort`, `Modelon.ThermoFluid.Interfaces.VolumePort`, `Modelon.ThermoFluid.Interfaces.ApplicationSpecific.TwoPhaseFlowPort`, and `Modelon.ThermoFluid.Interfaces.ApplicationSpecific.TwoPhaseVolumePort`.

TitaniumTankPipe

The `TitaniumTankPipe` is used within the cryomodule component. It has two `Modelon.ThermoFluid.Interfaces.ApplicationSpecific.TwoPhaseFlowPort`, one `Modelon.ThermoFluid.FlowChannels.DistributedTwoPhase` to act as the volume and helium container, one `DynamicWallTi` being the titanium wall of the tank, one `Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a` to connect to the outside radiation components, and one `Modelon.ThermoFluid.Interfaces.FlowHeatPort` being attached to the fluid on one end.

CMSteelPipingTwoPhase

This component is similar to a `DistributedPipeDWReynolds` in terms of components. The difference is that the `CMSteelPipingTwoPhase` has a `Modelon.ThermoFluid.FlowChannels.DistributedTwoPhase` pipe instead of a regular distributed channel. This allows for two-phase flow through the pipes, which occurs in the cryomodule where the component is used.

CMThermalShields / Niobium Cavity

These two components simply contain one `DynamicWallAl` (or `Nb`) and one `Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a`. This is to simulate the cold mass of the thermal shields in the cryomodule, or the niobium cavity.

4.3 Specification of Pipes and Components

By implementing the features and equations in the theory section into the components in the component section, the models are then given the parameters described in this section. It should be noted, however, that the final decision has not been made on the exact dimensions of all the pipes, valves, cavities, tanks, shields, and so on, when this project was finalized. The development of the technical specifications has reached far into the process though, and only minor changes can be expected [8]. The pipe dimensions are given as both mm and using the European designation of nominal diameters, DN (diamètre nominal). More about the DN convention can be found in [58], for example.

Piping

The piping in the transfer line and distribution line have the same dimensions, and will be of the kind specified in Table 4.2. All process pipes will be made of SS304 stainless steel. In addition, the thermal shield surrounding the internal process pipes

Pipe	Nominal Diameter	Outer Diameter (mm)	Thickness (mm)
Helium Supply	DN65	76.1	2.3
VLP Return	DN250	273.05	3.0
TS Supply	DN50	60.33	2.3
TS Return	DN50	60.33	2.3

Table 4.2 Table of pipe dimensions used in the transfer and distribution lines.

has a diameter of FI430 (Outer Diameter: 430 mm) and a thickness of 2 mm. This thermal shield is made from Al6060 aluminum. The external envelope, also called vacuum jacket, will have a diameter of DN550, corresponding to 558.8 mm.

In the jumper connections, which connects the individual cryomodules to the distribution line, the following dimensions are used, as seen in Table 4.3.

Pipe	Nominal Diameter	Outer Diameter (mm)	Thickness (mm)
Helium Supply	DN10	17.15	1.4
VLP Return	DN50	60.33	1.6
TS Supply	DN10	17.15	1.4
TS Return	DN10	17.15	1.4

Table 4.3 Table of pipe dimensions used in the jumper connections.

Valves

The cryogenic valves, used to control the helium flow in the distribution system, are placed in the valve boxes and have the properties stated in Table 4.4.

Valve Connection	Valve Name	Nominal Diameter	Kv Value
Helium Supply	CV8	DN10	1.77
VLP Return	CV9	DN50	66
TS Supply	CV11	DN10	0.9
TS Return	CV12	DN10	1.0

Table 4.4 Table of valve dimensions and Kv values for the control valves in the valve boxes. The flow coefficient value, Kv, is described briefly in e.g. [59].

Cryomodules

The cryomodules have been modeled as components with specific masses and their behaviors in the cool-down and warm-up processes are modeled through the bahavior of the materials that they contain. From [10], one obtains the mass setup for the three kinds of superconducting cryomodules seen in Table 4.5. The cavities are made from niobium, the helium tanks from titanium, the thermal shields from aluminum, and the piping from stainless steel.

Cryomodule Type	Cavities	Helium Tanks	Thermal Shields	Piping
Spoke	150	74	61	89
Medium-β	240	132	120	48
High–β	220	134	120	46

Table 4.5 Table of specified masses for the different components in the cryomodules. All masses in kg.

Conduction from Supports

The heat conduction from solid supports that keep the pipes in place have been calculated separately in [60] using Eq. (4.38). The results from this are seen in Table 4.6, and the mathematical description of the calculation procedure is found in the theory section.

Support Type	Fixed	Sliding	Vacuum Barrier
Number in TL	10	20	2
Number in DL	34	55	3
Heat load HS	0.486	0.162	0.648
Heat load VLP	1.5148	0.505	2.02
Heat load TSS	0.317	0.106	0.422
Heat load TSR	15.133	5.044	20.18

Table 4.6 Table of heat transfer through fixed and sliding supports and the supports for the vacuum barrier. The results are taken from [60]. Heat loads are given in W per support.

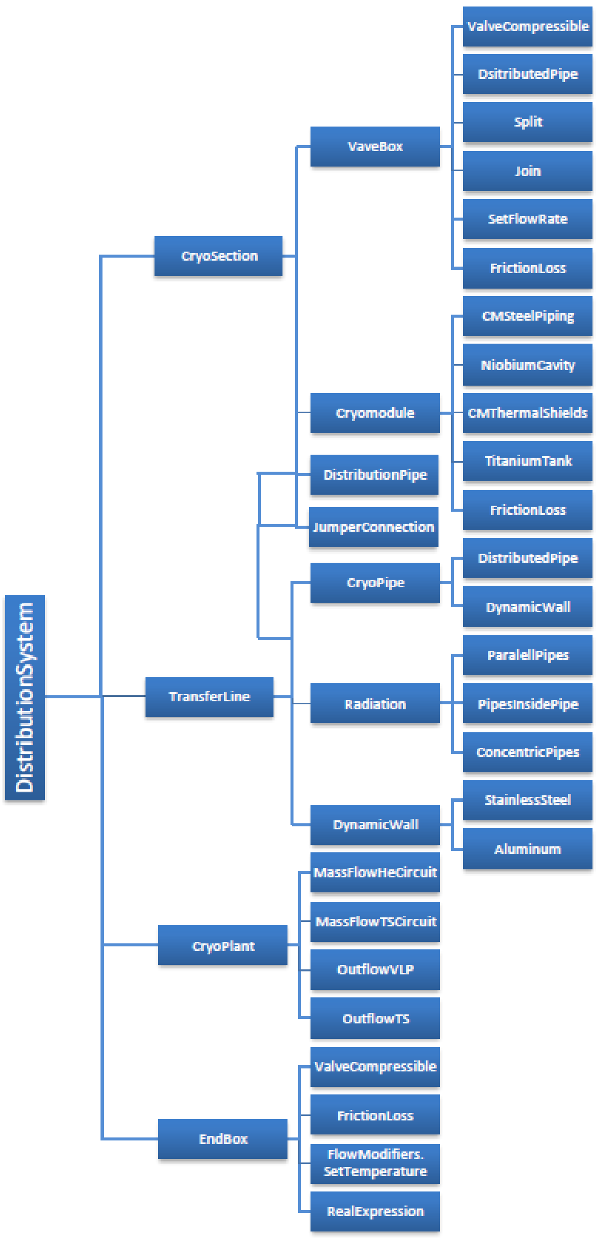


Figure 4.7 The component hierarchy for the cryogenic distribution system.

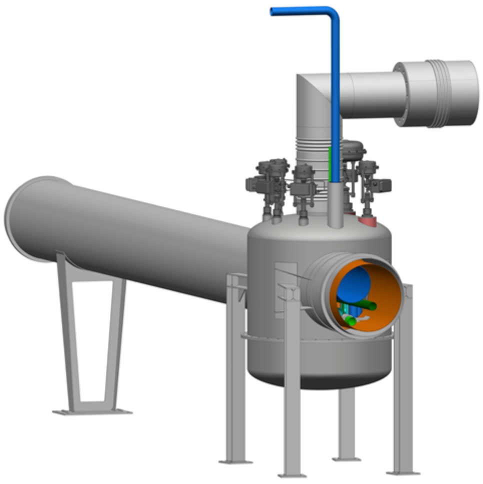


Figure 4.8 The structure of the distribution pipe with the external envelope (outer), the thermal shield (orange), and the four process pipes inside.

5

Results

In this chapter, the results from the primary simulations are presented. The first section deals with the helium discharge into a relief system, and the sizing of the vent line used in this system. This is followed by the cool-down and warm-up procedures for the distribution line without jumper connections and cryomodules; that is, with the valves going out of the distribution line closed. Then the results for the cool-down and warm-up of one cryomodule are presented. Finally, there is a comparison of the models used in this project with calculations in Excel and two published articles. These results do not have any other purpose than comparing and validating the other results, together with showing on some limitations for the other models.

5.1 Helium Discharge to Relief System

The results from the different simulations for the helium discharge into the relief system are seen in Fig. 5.1. It is found that the inflow at two separate points (red line) allows for a much higher mass flow rate. The green line gives the allowed mass flow rates for an allowed pressure of 1.2 bar, while the blue line allows for a pressure rise up to 1.1 bar. For an expected mass flow rate of 180 g/s at a maximum pressure of 1.2 bar, it is found that a DN150 pipe should be sufficient.

5.2 Cool-down of the Cryogenic Distribution Line

The total cool-down time of the distribution system with the jumper connections and cryomodules disconnected is 6.93 hours, or 416 minutes. If one only considers the cold helium circuit to reach 5.1 K, the cool-down time is 6.04 hours, or 362 minutes. The cool-down plot of some pipe sections, including the coldest one, is seen in Fig. 5.2.

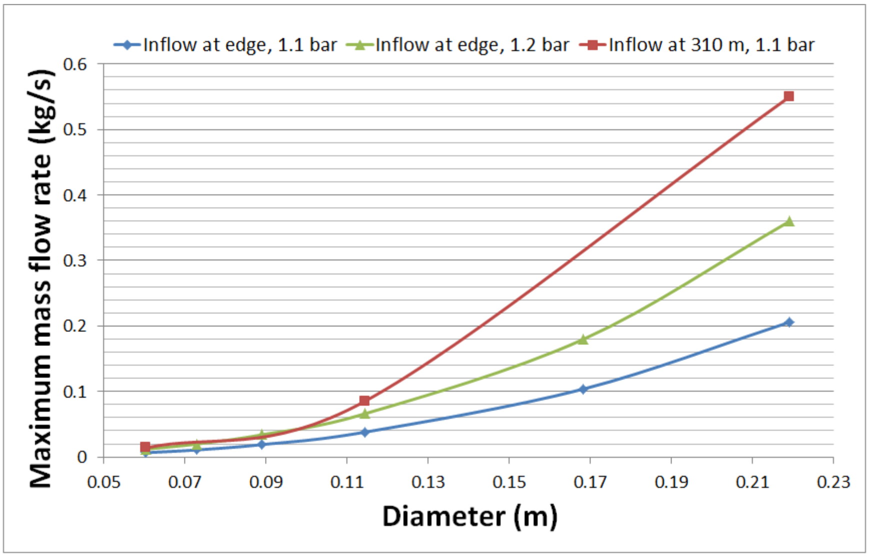


Figure 5.1 The maximum allowed mass flow rates (y axis) as functions of the pipe diameter (x axis). It should be noted that the red line, showing the inflow from two different points, allows for a higher mass flow rate than inflow from only one edge of the pipe.

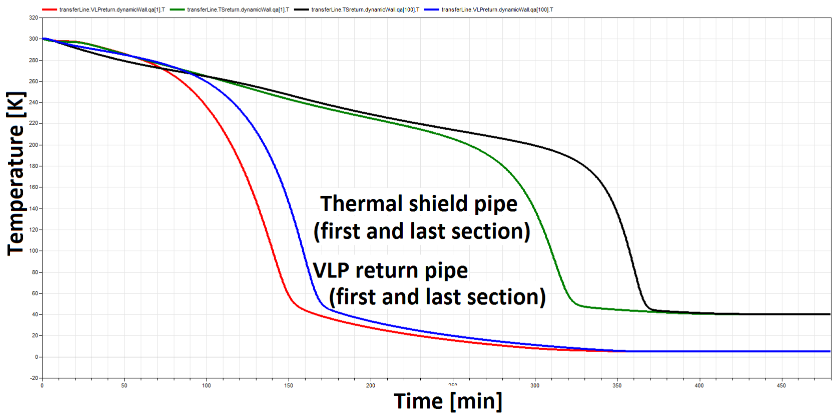


Figure 5.2 Cool-down of the distribution line, with jumper connections and cryomodules deattached. The plot shows the time evolution of the temperature (y axis in K) in the pipe wall of the VLP pipe (blue and red) and the thermal shield return pipe (green and black). The x axis displays time in minutes.

5.3 Warm-up of the Cryogenic Distribution Line

With Helium Flow

The warm-up time with 300 K helium flowing through the distribution system is 19.87 hours, or 1192 minutes. These results are found in Fig. 5.3.

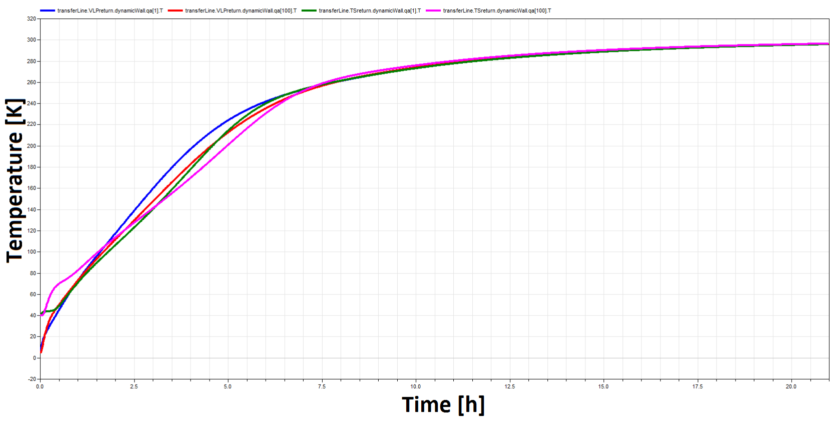


Figure 5.3 Warm-up of the distribution line without jumper connections and cryomodules for flowing helium of 300 K. The plot shows the temperature in K (y axis) as a function of time for the transfer line's first and last section in the outer pipe wall of the VLP pipe (blue and red) and the thermal shield return pipe (green and pink). The x axis displays time in hours.

Without Flow and Only Static Heat Loads

For the warm-up with only static heat loads present, the warm-up time is way beyond two weeks (336 hours), and the temperature plot is seen in Fig. 5.4.

5.4 Cool-down of One Cryomodule

The cool-down time for one spoke cryomodule was between 5.96 (3 g/s) and 15.68 (1 g/s) hours. The process of the cool-down for the 1 g/s simulation is seen in Fig. 5.5. For an elliptical cryomodule, the cool-down time is between 9.76 (3 g/s) and 21.84 (1 g/s) hours. The simulated cool-down times for both the spoke and the elliptical cryomodules for four different inflow rates are collected in Table 5.1, and the results are plotted in Fig. 5.6. The total amount of helium needed for cooling a spoke cryomodule with an inflow of 3 g/s is then found to be $m_{He} = \dot{m}_{He} \cdot t_{cool} = 3 \cdot 10^{-3} \cdot 5.96 \cdot 3600 = 64.4$ kg.

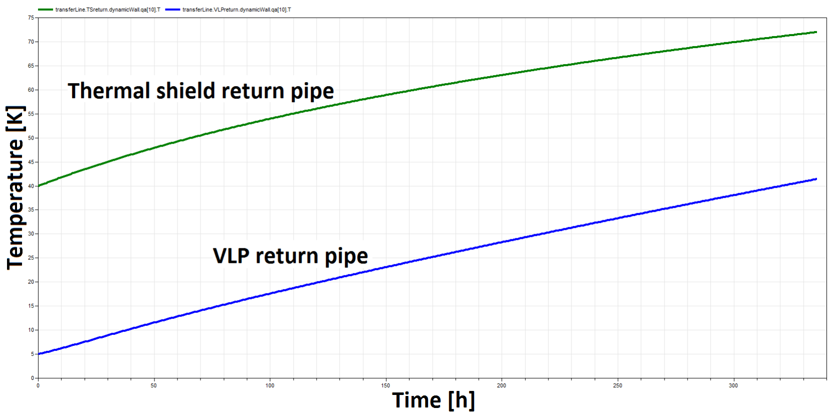


Figure 5.4 Warm-up of the distribution line without jumper connections and cryomodules with only the static heat leaks present. These include radiation between the process pipes and thermal shield and the conduction from fixed, sliding, and vacuum barrier supports. The plot shows the temperature (y axis in K) as a function of time for the transfer line's first and last section in the outer pipe wall of the VLP pipe (blue and red) and the thermal shield return pipe (green and pink). The x axis shows time in hours.

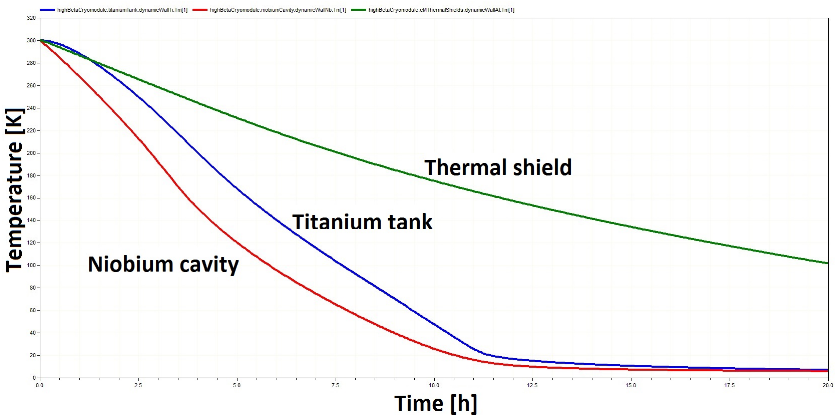


Figure 5.5 The time evolution of the cool-down of one cryomodule. The y axis shows temperature in K and the x axis shows time in hours.

5.5 Warm-up of One Cryomodule

The warm-up time for one spoke cryomodule was between 9 and 25 hours for a mass flow rate of 3 g/s and 1g/s, respectively. The warm-up for the simulation of

Mass Flow Rate	Spoke	Elliptical
1.0 g/s	15.68 h	21.84 h
1.3 g/s	12.12 h	17.52 h
2.0 g/s	8.20 h	12.80 h
3.0 g/s	5.96 h	9.76 h

Table 5.1 Cool-down times for a spoke and an elliptical cryomodule for different mass flow rates.

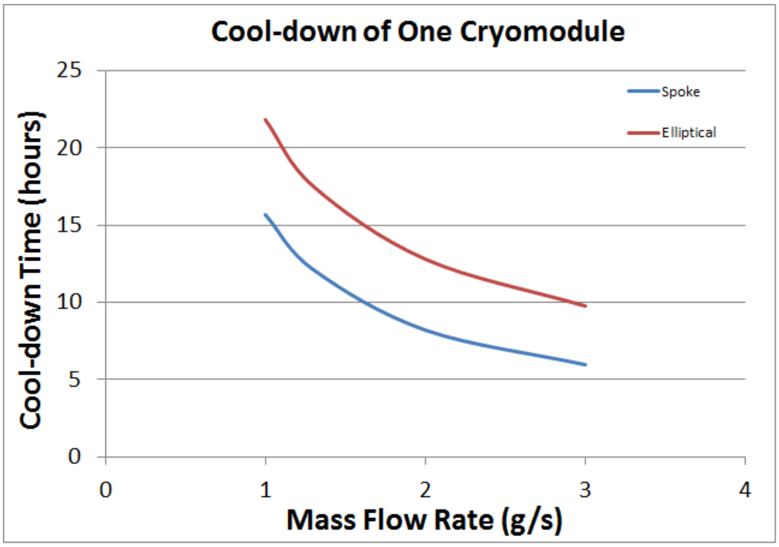


Figure 5.6 The cool-down times of one cryomodule as functions of mass flow rates into the module. The calculations are made for both spoke and elliptical cryomodules.

the 1 g/s case is seen in Fig. 5.7. For an elliptical cryomodule, the warm-up time is instead between 14.64 (3 g/s) and 37.68 (1 g/s) hours. The simulated warm-up times for both the spoke and the elliptical cryomodules for four different inflow rates are collected in Table 5.2, and the results are plotted in Fig. 5.8.

5.6 Model Comparisons

Validation with Excel and HePak

The results from the Dymola simulations for the helium and pipe wall temperatures are seen in Figs. 5.9 and 5.10, respectively. In Figs. 5.11 and 5.12, the mass flow rate and the heat flow between the helium and the pipe wall are seen. For these figures

Mass Flow Rate	Spoke	Elliptical
1.0 g/s	25.0 h	37.68 h
1.3 g/s	19.65 h	29.92 h
2.0 g/s	13.15 h	20.56 h
3.0 g/s	9.0 h	14.64 h

Table 5.2 Warm-up times for a spoke and an elliptical cryomodule for different mass flow rates.

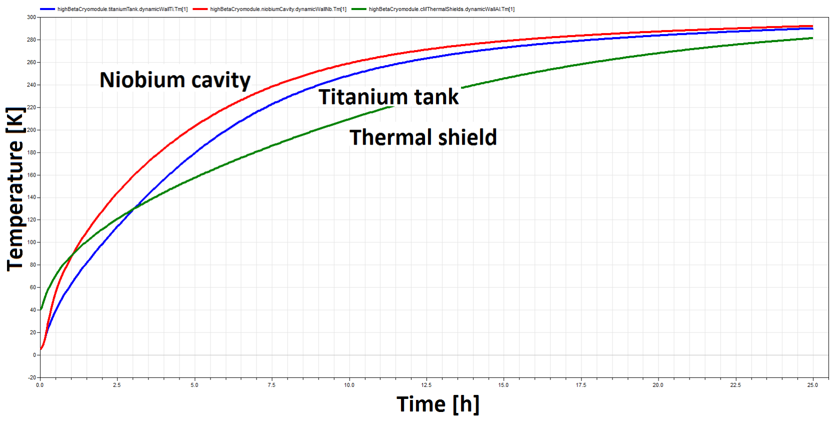


Figure 5.7 The time evolution of the warm-up of one cryomodule. The y axis shows temperature in K and the x axis shows time in hours.

below, the evolutions are displayed from the inlet of the pipe (first line) with even 10 m steps through the end of the pipe (bottom or rightmost line).

The Excel calculations give an inflow velocity of 48.9 m/s, which means that the entire length of the pipe is filled in just over one second. This can be viewed and compared in the Dymola simulation in Fig. 5.11, where the steep and quick rises in mass flow rate for all the sections are seen. Given the inflow temperature (5 K) and pressure (1.05195 bar, seen in Fig. 5.13), the speed of sound in the medium is given by HePak to be 967.3 m/s, and the helium velocity is by that safely under 0.05 Mach, meaning that it can be modeled as an incompressible flow. This makes it reasonable to use the models described above for friction and medium propagation [61, 18, 62]. In the top graph of Fig. 5.15 it is shown that the helium temperature in the first pipe section decreases to 28.7 K in 0.1 seconds. Given the enthalpy differences between the stationary helium at 300 K and 1.05 bar and the flowing helium of 5 K and 1.1 bar, assuming a pressure of 1.1 bar, gives the helium temperature 26.1 K in close accordance to the Dymola results. In addition, it was given that the heat transfer coefficient from the Dittus-Bölter equation for the first time instants

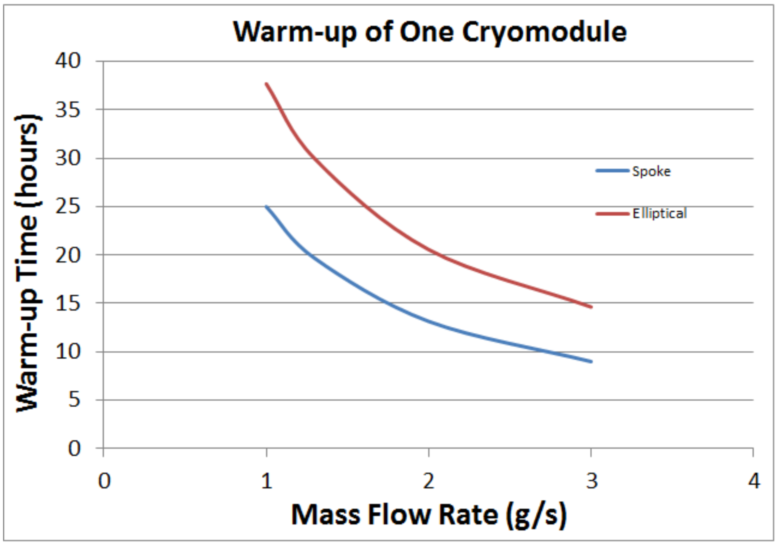


Figure 5.8 The warm-up times of one cryomodule as functions of mass flow rates into the module. The calculations are made for both spoke and elliptical cryomodules.

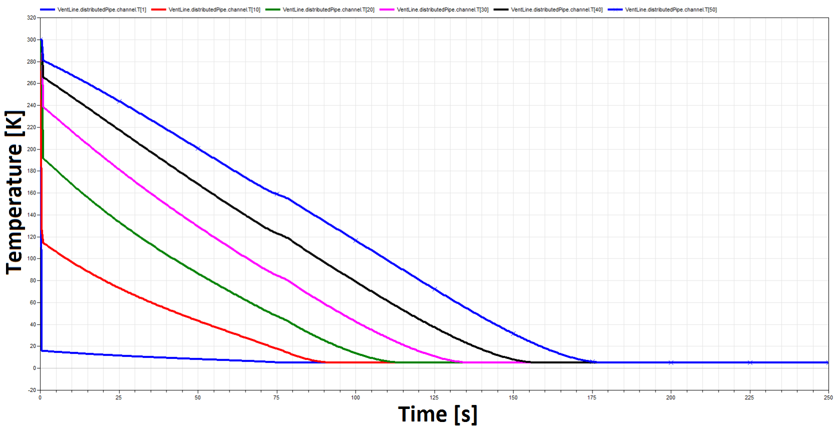


Figure 5.9 Helium temperature for the simulation. The inlet is the leftmost blue line and the outlet is the blue crossed line to the right. The y axis shows temperature in K and the x axis shows time in seconds.

equals 215 W/m^{2}K. Using this value and the pipe area, heat flow after the first 0.1 seconds, where the temperature difference amounts to 276 K, is according to the

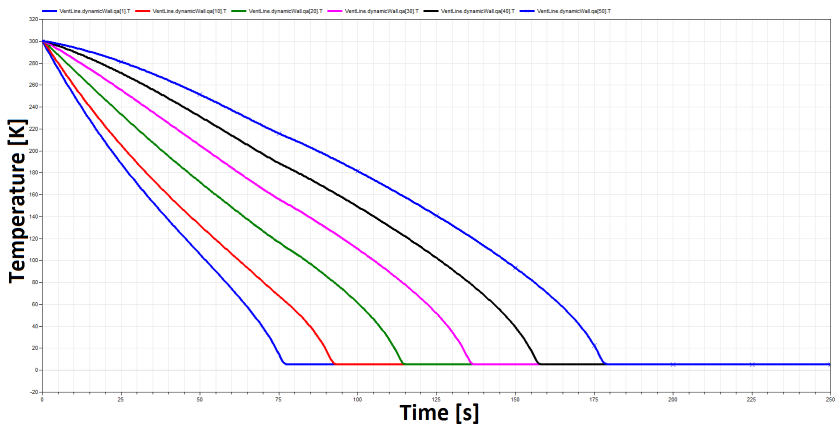


Figure 5.10 Pipe wall temperature for the simulation. Inlet is blue, and the outlet is blue with crosses. The y axis shows temperature in K and the x axis shows time in seconds.

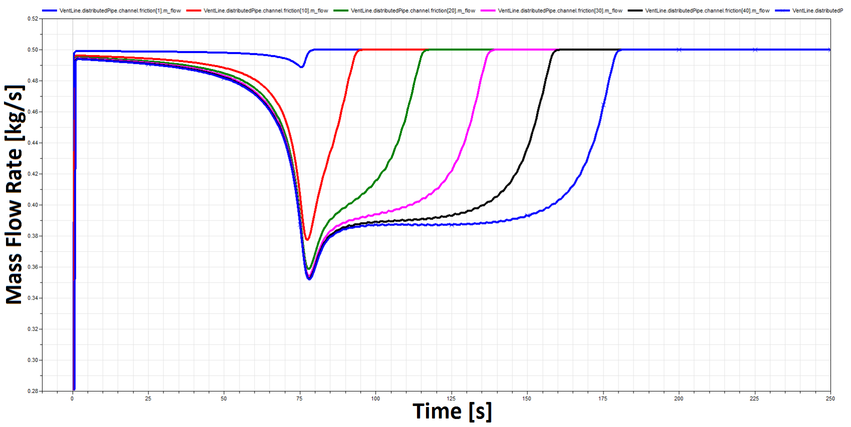


Figure 5.11 Helium mass flow rate. The inlet is shown in blue with only a small bump at 75 s, and the outlet has the last jump to steady-state at 0.5 kg/s. The y axis shows mass flow rate in kg/s and the x axis shows time in seconds.

Excel calculations 15.7 kJ. By looking at Fig. 5.12, one sees that this agrees with the Dymola simulation.

Comparison with Experimental Data from Nitrogen Experiment

The experimental results as given in the paper are seen in Fig. 5.16. The corresponding plots for the Dymola simulation, which used the pressures as in/parameters, are

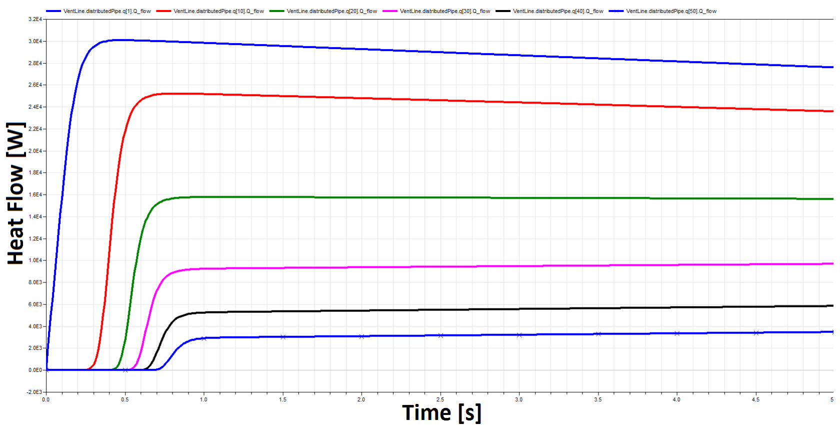


Figure 5.12 Heat flow between helium and the pipe wall. The inlet is the blue top line, and the outlet is the bottom blue crossed line. The y axis shows heat flow in W and the x axis shows time in seconds.

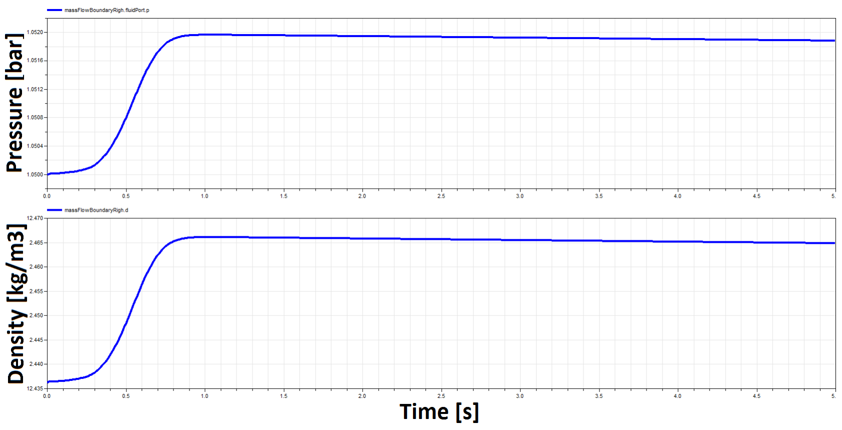


Figure 5.13 Pressure (top, Pa) and density (bottom, kg/m^3) at the inlet for the first five seconds. The x axis shows time in seconds.

seen in Figs. 5.17, 5.18, and 5.19. In Fig. 5.16, it is shown how the pressure starts with an even distribution from 9 bar down to slightly over 2 bar. This appearance and slope is then maintained and lowered as time passes and after 80 seconds the pressures in the extremes are between 4 and about 1.8 bar. The temperature distribution is initially quite concave and spans between about 215 K to 280 K. It gradually becomes more linear and after 80 seconds it resembles a linear graph, with minimum at 130 K and maximum at 270 K. The mass flow rate starts off just below 40

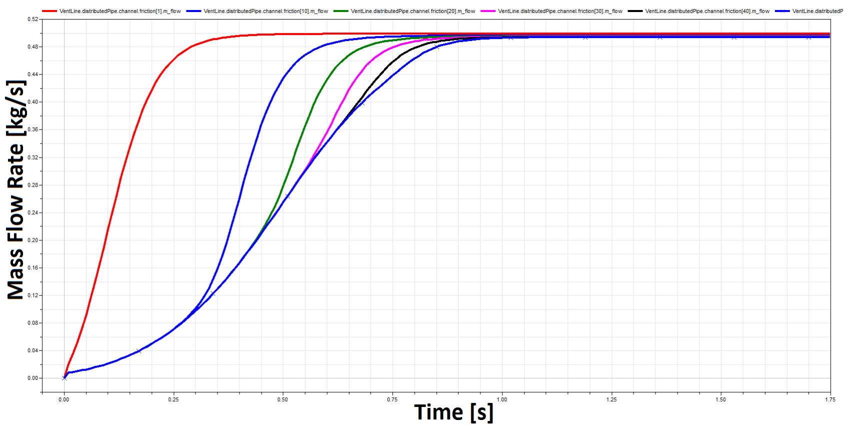


Figure 5.14 Helium mass flow rate for the first 1.5 seconds, where the rapid rise in mass flow rate in the last section (blue crossed line, rightmost) can be seen, as predicted by the calculated velocity. The y axis shows mass flow rate in kg/s and the x axis shows time in seconds.

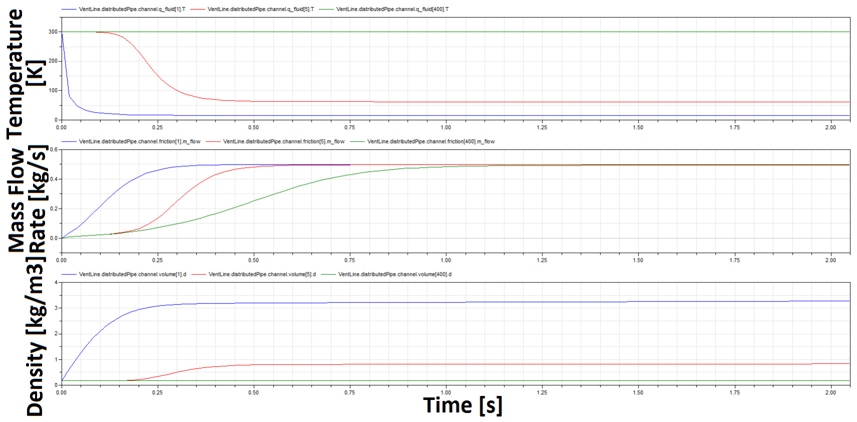


Figure 5.15 Some helium properties for the first two seconds of the simulation. In the top graph, the helium temperature is seen in K, the second graph shows the mass flow rate in kg/s, and the last graph shows the helium density in kg/m^3. The x axis displays time in seconds. For the three plots, blue is at the inlet, red is after 5 m, and green is at the outlet (at 400 m).

kg/s at the beginning and declines down to 18 kg/s after 80 seconds.

The pressure in the Dymola simulation also gradually lowers but keeps the same shape of the curve. The values in the beginning and end of the pipe go from 9 to 4 bar and 2.5 to 2.1 bar, respectively. The temperature distribution is initially concave and gradually becomes convex, where the temperature of the first section of the pipe goes from 215 K to 130 K while the last section starts at 270 K and reaches 180 K. The mass flow rate has a slightly concave shape, starts at slightly below 40 kg/s, and ends at 20 kg/s.

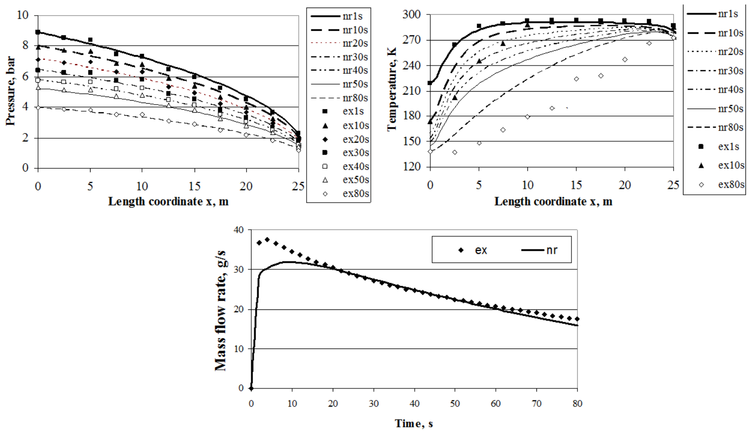


Figure 5.16 The results for pressure drops (top left) and temperature (top right) increases as functions of the pipe position, together with the mass flow rate versus time (bottom). Source: [19].

Comparison with Experimental Data from Helium Relief System

The results from the paper are seen in Fig. 5.23. These are compared to the simulated results in Dymola which yield the plots in Figs. 5.20, 5.21, and 5.22. The analysis of the quench line in the refered paper gives a declining pressure in the entire line, with a maintained, slightly concave curve dropping about 1.5 bar over the whole simulation of 10 seconds. The temperature has initially an increase in the pipe end due to the compression and the maximum temperature is about 350 K. This then goes down to 30 K after 10 seconds and the distribution is an almost straight line from 8.7 to 30 K. The velocities in the quench line stay very high throughout the first 10 seconds, and the maximum velocity is reached after one second being a little bit over 600 m/s in the pipe end. After 10 seconds, the maximum velocity has gone down to about 170 m/s.

The Dymola results for the pressure distribution shows on a similar appearance, where the pressure in the end of the pipe stays somewhere between 2 and 3 bar during the ten seconds of simulation. The pressure in the beginning of the pipe goes down from 5.5 bar to 4 bar. The temperature distributions maintain the straight slope line, and go from a value of about 335 K in the end of the pipe at one second, to

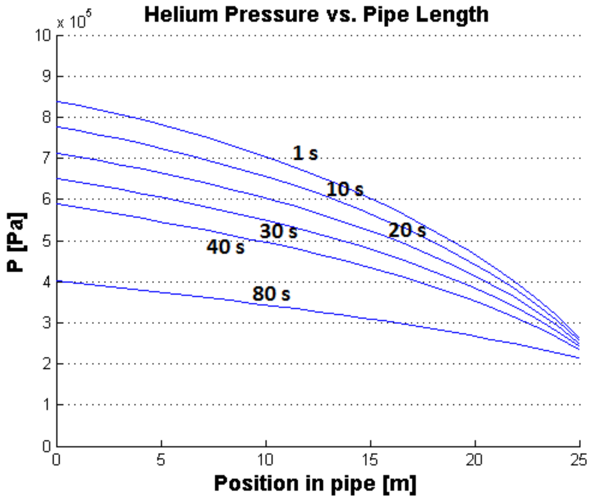


Figure 5.17 Results from the Dymola simulation for the pressure drop.

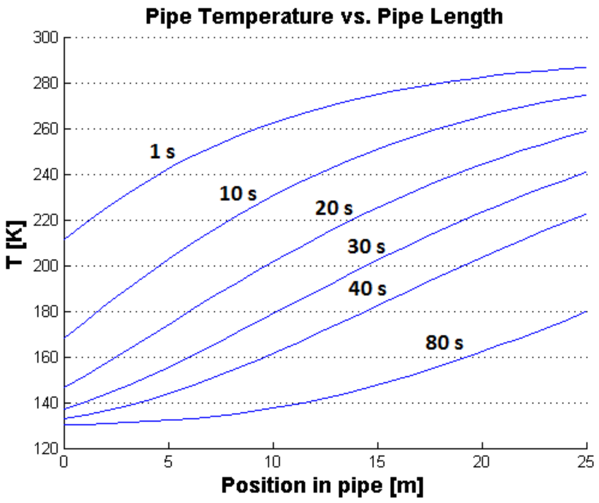


Figure 5.18 Results from the Dymola simulation for the temperature increase.

slightly above 100 K at the same place, whereas the initial lump of the pipe stays around 10 K. The velocity profile for one second goes high up to over 600 m/s at one second, while the curves for five and ten seconds are concave reaching somewhat above 200 m/s at five seconds and about 120 m/s after ten seconds.

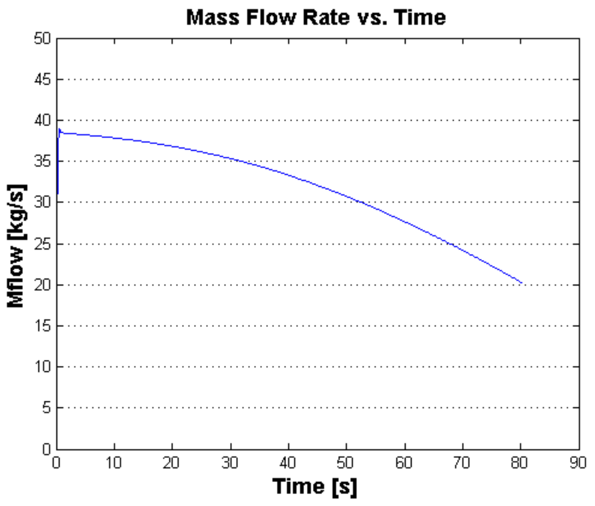


Figure 5.19 Results from the Dymola simulation for the mass flow rate development.

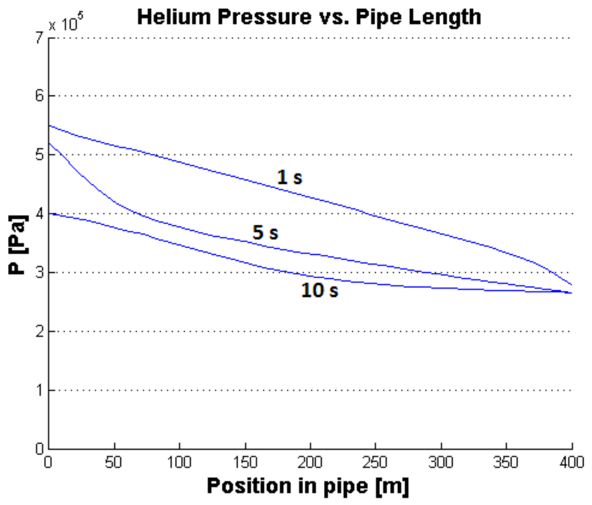


Figure 5.20 Results from the Dymola simulation for the distribution of pressures in the quench line.

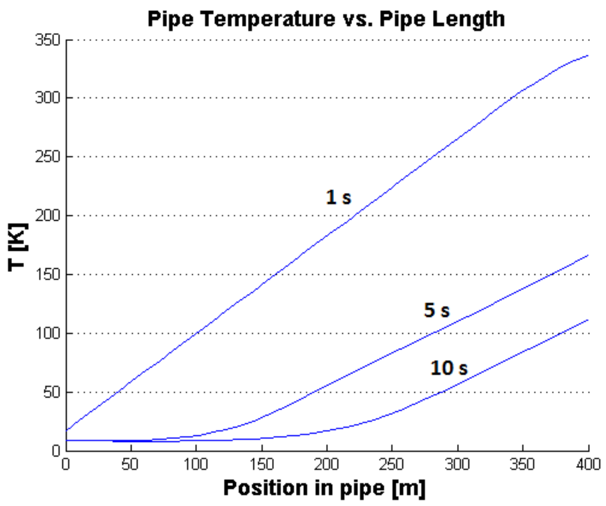


Figure 5.21 Results from the Dymola simulation for the distribution of temperatures in the quench line.

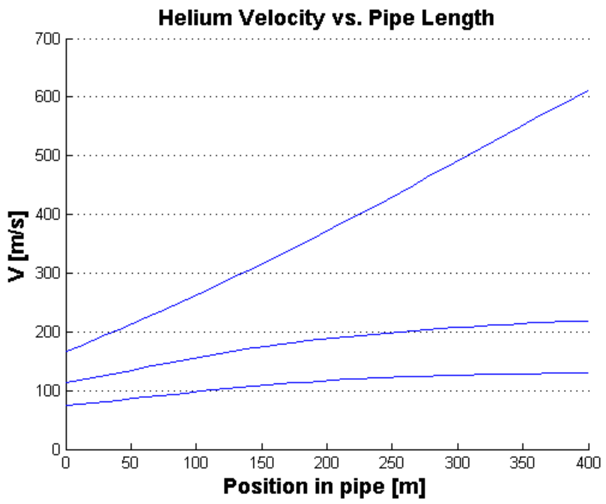


Figure 5.22 Results from the Dymola simulation for the velocity distribution in the quench line.

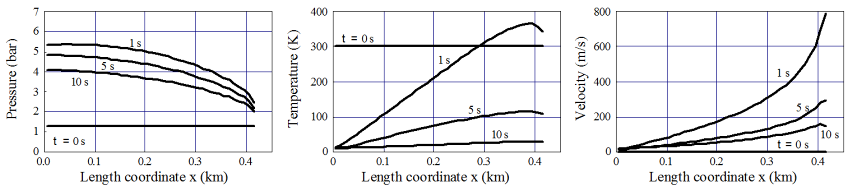


Figure 5.23 The results for pressure drops (left), temperature increases (center), and velocity (right) as functions of the position in the pipe, as given in the experimental article of comparison. Source: [20].

6 Discussion

6.1 Conclusions

Helium Discharge to Relief System

The result for this simulation shows that the worst case scenario of peak pressures appear when all of the helium inflow appears in one place. A design where the maximum pressure can be 1.2 bar can then allow for at least 180 g/s of helium to flow into the discharge system and vent line with a DN150 pipe. This will save time and effort in the installation, since the initial design was for a DN200 pipe, at the same time as money will be saved due to the smaller material costs.

Cool-down and Warm-up of the Cryogenic Distribution System

The cool-down of the non-dynamic part of the cryogenic system meets any requirements to reach operating temperature in less than a day. Even if the thermal shields need to be at their coldest temperatures, the cool-down is less than seven hours. If one also takes the cryomodules into account, the cool-down will be less than 24 hours given that the helium flow into the cryomodules can be kept steady at 1 g/s for all 43 of them. The cool-down of one single cryomodule takes 15.68 hours, which is an important time scale for the downtime of the machine due to a need of warming a module for maintenance, replacements or repairs.

The warm-up of the distribution line is almost three times longer than the cool-down and is therefore the most important feature in the downtime of the accelerator, in the case of a complete shut-down and warm-up. Including the cryomodules, this warm-up takes 25 hours. For the warm-up of one single cryomodule, the downtime due to warming it up is also 25 hours, which is notably longer than the cool-down.

For the combined downtime for warming a cryomodule up, repairs and maintenance, and then cooling it down, the warm-up procedure is the most time consuming. The same goes for the entire distribution line. The combined downtime only due to warming and cooling using flowing helium is given as 41 hours for a single cryomodule.

Cool-down and Warm-up of One Cryomodule

The simulations for the cool-down of a single cryomodule shows how the total cool-down time is strongly dependent on the inflow rate. This has an almost linear behavior for the mass flow rates considered in this project, thus suggesting that the aim should be to have as a high of a flow as possible in the transient cool-down period. By calculating the total mass of helium needed for the cool-down, it is found that it agrees with the specifications provided by the manufacturer in Fig. 2.3.

The warm-up is also for an individual cryomodule a more time demanding process than the cool-down. The total time is strongly dependent on the mass flow rate into the module, and this should be considered at the manufacturing of the cryomodules, if downtime is an important factor.

Validations

As mentioned in the results for the Excel validation, the results between Excel with HePak and the Dymola simulation agree well for all variables. The simulation shows on an important event where the initialization of rapid inflow of cold helium into a warm pipe can be handled well. This validation is helpful to support the results for the helium discharge calculations.

For the validation with the nitrogen flow through a copper pipe, the pressure and mass flow plots lie within a 10% difference between the experimental results and the Dymola simulation. The temperature differs more, however. In the first sections of the pipe, the Dymola simulation closely resembles that of the experimental results. However, as time passes, the end sections differ more and more in temperature. The Dymola simulation gives a final temperature of 180 K, while the experimental results yield only a minor decrease down to 270 K. This means that there is a difference of 90 K, or an absolute error of 33%. This error must be due to some factor in the heat transfer that has not been taken care of. A reasonable guess would be that the heat transfer between the pipe wall and the surrounding air is less prominent than in the Dymola model, most likely due to frosting on the outside of the pipe wall. Another reason could be that the Dymola model does not take the exact geometry of the piping into account in the Dittus-Bölter friction model. Finally, the measurement of temperature in the experiment was performed with a thermometer placed in the nitrogen flow. This causes a slight pressure increase on the upstream part of the thermometer and a possible registration of a higher temperature.

The comparison of the helium discharge into a relief system for ten seconds receives a similar result as the nitrogen simulation above. While the pressures stay closely the same, the appearance of the temperature distributions become more and more different as time passes. Once again, this should be due to a different modeling of the heat transfer between the ambient and the pipe wall, as the heat transfer between the pipe wall and the flowing helium follows the Dittus-Bölter procedure and is a lesser subject of discussion. One likely reason for the temperature differences is simply that the insulation in reality differs from that received by word-of-mouth for the Dymola simulation. Since the modeling in the paper was done some years ago and the paper did not specify the specific type insulation used, discussions lead to using standard vacuum insulation, which seems to have been too conservative.

The results from the validations show that the pressure and flow of the helium (and nitrogen) follow the same evolutions as for the benchmarked models, and lie within 10% for all of the simulations. The heat transfer has a larger difference at times, most likely due to that the heat transfer between the pipe and the surroundings has not been outlined in detail in the papers and therefore show a different behavior for the Dymola models and the results. For the actual simulations of the cryogenic distribution line, however, the convective heat transfer is already considered in the inflow of heat from the Dymola components, and the appearing errors in the validations models should not affect the other simulations to a large extent. In addition, the cool-down and warm-up simulations of the piping occur within vacuum insulation, causing a possible erroneous convective heat flow to be of negligible importance.

6.2 Future Work

The work that has been carried out in this Master thesis has involved setting up a simulation environment for cryogenic models, creating cooperation and contact with Modelon AB as a delivery firm for libraries, medium properties, and assistance in the modeling process, and running simulations using Dymola. The models created are available for individual or composite use for simulations of cryogenic situations that might arise. It should be noted that there have been additional Dymola components created that have not been used in the primary simulations presented in this report, that could be integrated into future models as well.

The idea within the cryogenic group at ESS is to maintain a cryogenic student project over time. The foundation for this has been laid in this project, and the models can very well be developed further and expanded into more sophisticated tools. There are still many problems to be set up and solved within the scope of cryogenics, and Dymola will be used for this to a large extent. The personnel in the Specialized Technical Services Group at ESS has come into contact with Modelica and Dymola, and these tools will be used for detailed simulations in the near and far future. The models created here together with the additional work input from future engineers and students will be part of creating the ESS cryogenic system.

Suggestions for future work is to expand on the model of the cryomodules, and perform more detailed simulations of the cool-down and warm-up process thereof. As was seen in the results, the cool-down times depend heavily on the mass flow rate of helium into the modules, and a rigorous analysis of possible flow rates would be of great importance. Even though the cool down of the major parts of the cryogenic system has been simulated in this project, a detailed simulation of all cold masses at once is yet to be performed. This is an advanced task numerically, in terms of modeling, and when it comes to initialisation. The effort should therefore be based on the models created but taking system considerations into account in order to be able to carry out the simulations.

7 References

[1] *European Spallation Source ESS AB*. URL: `http://europeanspallationsource.se` (visited on 10/07/2013).

[2] Peggs, Steve (ed.) *ESS Technical Design Report*. ESS-doc-274. April 23, 2012.

[3] Lundh, Emil and Jonathan Moberg. *Fatigue analysis of the ESS target wheel in accordance with RCC-MRx*. Degree Project. Media-Tryck, Lund, 2013.

[4] Oak Ridge National Laboratory. URL: `http://netrons.ornl.gov` (visited on 10/07/2013).

[5] Pynn, Roger. *An introduction to neutron scattering*. Presentation at Indiana University and Oak Ridge National Lab. URL: `http://neutrons.ornl.gov/conf/nxs2010/pdf/lectures/IntroductoryLectures_Pynn_ORNL.pdf`.

[6] URL: `http://europeanspallationsource.se/sites/default/files/ess\_pulse\_1.png`.

[7] Hott, Roland, et.al. "Review on superconducting materials". *Handbook of Applied Superconductivity* (2013). URL: `http://arxiv.org/abs/1306.0429`.

[8] Fydrych, Jaroslaw and Piotr Tereszkowski. *Ess cryogenic infrastructure - distribution system for the ess linac*. ESS Presentation. November, 2013.

[9] Böckh, Peter von and Thomas Wetzel. *Heat Transfer - Basics and Practice*. Springer-Verlag Berlin Heidelberg, 2012.

[10] Wang, Xilong L. et.al. *Heat load estimates of the ess accelerator cryogenic plant*. Doc.nr. ESS/AD/TN0052. November, 2013.

[11] Wang, Xilong. Cryogenics Engineer, ESS AB. Private discussions. 2013/2014.

[12] Weisend II, John G. *Handbook of Cryogenic Engineering*. Taylor & Francis, USA, 1998.

[13] Princeton University. *Joule-thomson effect.* URL: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Joule%E2%80%93Thomson_effect.html.

[14] Fydrych, Jaroslaw, Bartosz Zajaczkowski, and Maciej Chorowski. *Helium flows from the tf coils to the cold quench tank after the fast energy discharge.* Presentation at Wroclaw University of Technology. 2011.

[15] Weisend II, John G. Flow Specifications. 2014.

[16] IPN Orsay. *Cryogenic distribution aspects of the Spoke cryomodule - Sizing.* Technical Note IPNO-DA-ESS-NT-20140217, 2014.

[17] HePak. URL: http://www.htess.com/hepak.htm.

[18] *Modelica Language Specification, Version 3.2.* 2010.

[19] Chrowski, M., et.al. "Experimental validation of the lhc helium relief system flow modeling". In: CEC-ICMC, Keystone, Colorado, USA. 2005.

[20] Chrowski, M., et.al. "Flow and thermo-mechanical analysis of the lhc sector helium relief system". In: ICEC20, Beijing, China. 2004.

[21] Microsoft Office, Excel. URL: http://office.microsoft.com/en-001/excel/.

[22] Wikipedia. *Microsoft excel.* URL: http://en.wikipedia.org/wiki/Microsoft_Excel.

[23] *Dymola: Environment for Object-oriented Modeling of Physical Systems.* 2005. URL: http://www.inf.ethz.ch/personal/fcellier/Res/Soft/Dymola_engl.html (visited on 11/18/2013).

[24] Dynasim. URL: http://www.ida.liu.se/labs/pelab/realsim/dynasim.php3.

[25] Dassault Systèmes. URL: http://www.3ds.com/.

[26] Wikipedia. *Modelica.* URL: http://en.wikipedia.org/wiki/Modelica (visited on 11/18/2013).

[27] Wilhelmsson, Carl. Product Manager and Senior simulation engineer, Modelon AB. Private discussions. 2013/2014.

[28] MathWorks. *Matlab - the language of technical computing.* URL: http://www.mathworks.se/products/matlab/.

[29] Wikipedia. *Matlab.* URL: http://en.wikipedia.org/wiki/MATLAB.

[30] The Engineering Toolbox. *Dynamic pressure.* URL: http://www.engineeringtoolbox.com/dynamic-pressure-d_1037.html (visited on 01/24/2014).

[31] Tritton, D.J. *Physical Fluid Dynamics.* 2nd ed. Oxford University Press, 1988.

[32] Landau, L.D. and E.M. Lifshitz. *Fluid Mechanics.* 2nd ed. Pergamon Press, 1987.

[33] The Engineering Toolbox. *Colebrook equation*. URL: `http://www.engineeringtoolbox.com/colebrook-equation-d\_1031.html` (visited on 01/24/2014).

[34] Libby, Paul A. *Introduction to Turbulence*. Taylor & Francis, Washington, 1996.

[35] Idelchik, I.E. *Hanbook of Hydraulic Resistance*. 3rd ed. Jaico Publishing House, Mumbai, 2003.

[36] White, Frank M. *Fluid Mechanics*. 7th ed. McGraw Hill, NY, 2008.

[37] The Engineering Toolbox. *Individual universal gas constant*. URL: `http://www.engineeringtoolbox.com/individual-universal-gas-constant-d\_588.html` (visited on 09/12/2013).

[38] Benson, Tom. *Beginner's guide to aerodynamics*. URL: `http://www.grc.nasa.gov/WWW/k-12/VirtualAero/BottleRocket/airplane/mflchk.html` (visited on 10/17/2013).

[39] T. Benson.

[40] Annaratone, Donatello. *Engineering Heat Transfer*. Springer, 2010.

[41] Sundén, Bengt. *Värmeöverföring*. Sudentlitteratur, Lund, 2006.

[42] Horizon Technologies. *Cryodata inc.* URL: `http://www.htess.com/cryodata.htm`.

[43] Jensen, J.E., et.al. *Brookhaven National Laboratory Selected Cryogenic Data Notebook, Volume I*. Brookhaven National Laboratory, 1980.

[44] NIST Cryogenics Technologies Group. *Material properties*. URL: `http://cryogenics.nist.gov/MPropsMAY/materialproperties.htm` (visited on 12/09/2013).

[45] The Engineering Toolbox. *Radiation heat transfer*. URL: `http://www.engineeringtoolbox.com/radiation-heat-transfer-d\_431.html` (visited on 10/10/2013).

[46] Modest, Michael F. *Radiative Heat Transfer*. Elsevier Inc., 2013.

[47] Flynn, Thomas M. *Cryogenic Engineering*. 2nd ed. CRC Press, 2004.

[48] Leuenberger, H., et.al. *Compilation of radiation shape factors for cylindrical assemblies*. The American Society of Mechanical Engineers, 1957.

[49] Basiński, Paweł. *Model Obliczeniowy Konfiguracji Wielowarstwowej Izolacji Próżniowej MLI Wielokanałowych Linii Kriogenicznych*. Politechnika Wrocławska, 2012.

[50] Massachusetts Institute of Technology. *Radiation heat transfer between black surfaces of arbitrary geometry*. URL: `http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node137.html` (visited on 09/12/2013).

[51] Howell, John R. *A catalog of radiation heat transfer configuration factors*. URL: `http://www.engr.uky.edu/rtl/Catalog/tablecon.html` (visited on 03/06/2014).

[52] Martinez, Isidoro. *Radiative View Factors*. 1995.

[53] Byun, Do Yung. "Investigation of radiative heat transfer in complex geometries using blocked-off, multiblock, and embedded boundary treatments". *Numerical Heat Transfer, Part A: Applications* (2003). URL: `http://dx.doi.org/10.1080/713838148`.

[54] Weisend II, John G. *Cryogenics lectures at michigan state univesity*. 2012.

[55] Hofman, A. "The thermal conductivity of cryogenic insulation materials and its temperature dependence". *Elsevier Cryogenics* **46:11** (2006).

[56] Xie, J.G. "Study on the heat transfer of high-vacuum-multilayer-insulation tank after sudden, catastrophic loss of insulating vacuum". *Elsevier Cryogenics* (2010).

[57] Modelica Association. *Modelica Language Specification, Version 3.3*. 2012.

[58] Wikipedia. *Nominal pipe size*. URL: `http://en.wikipedia.org/wiki/Nominal_Pipe_Size`.

[59] Valvias. *Flow coefficient definition*. URL: `http://www.valvias.com/flow-coefficient.php`.

[60] Fydrych, Jaroslaw. *Preliminary heat load estimations for the Cryogenic Distribution System of the ESS accelerator*. ESS Technical Note ESS/AD/TN/0051, 2013.

[61] Aksel, Haluk and Cahit Eralp. *Gas Dynamics*. Prentice Hall International, 1994.

[62] Williamson Jr., K.D. and Frederick J. Edeskuty. *Liquid Cryogens*. CRC Press Inc., Boca Raton, Florida, 1983.

8

Appendices

8.1 Appendix I - Dymola Components, Graphical Layer

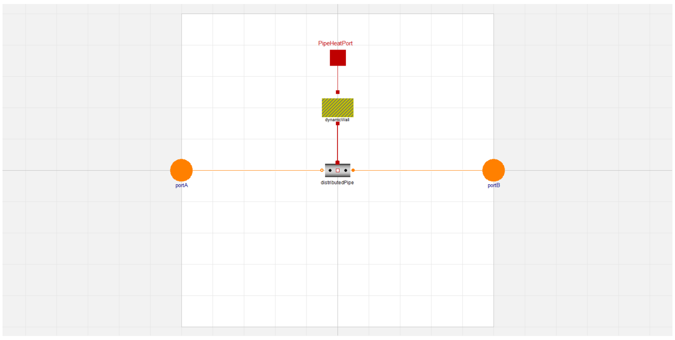


Figure 8.1 CryoPipeDWReynolds.

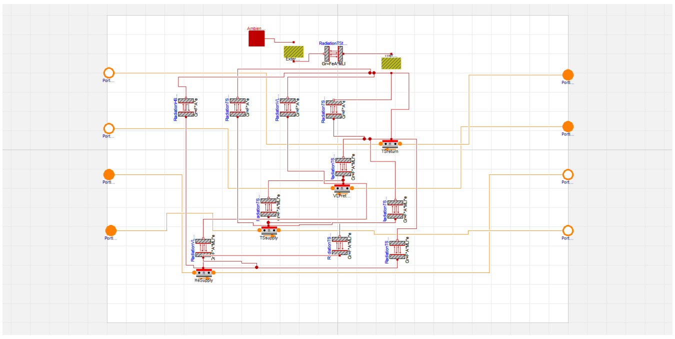


Figure 8.2 DistributionPipe.

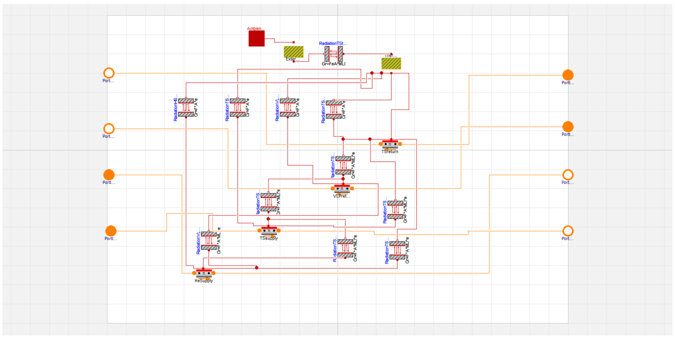


Figure 8.3 TransferLine.

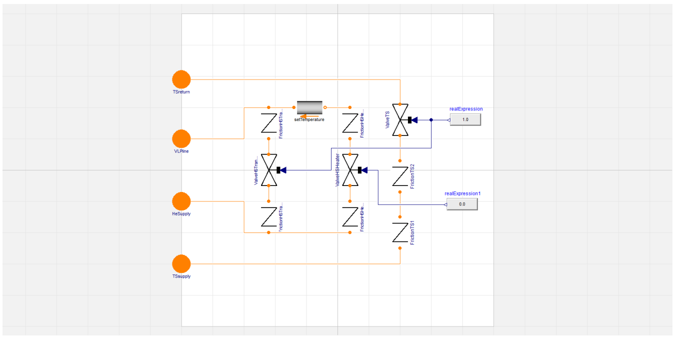


Figure 8.4 EndBox.

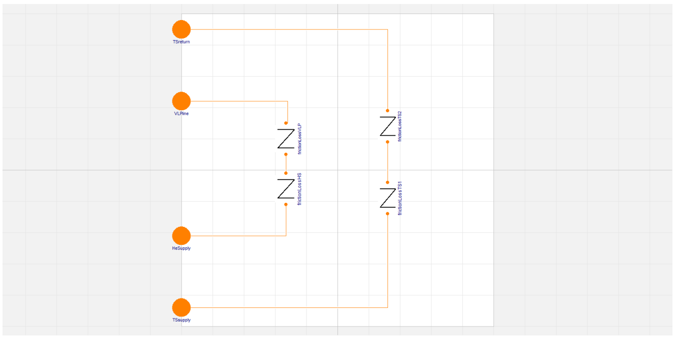


Figure 8.5 SimpleEndBox.

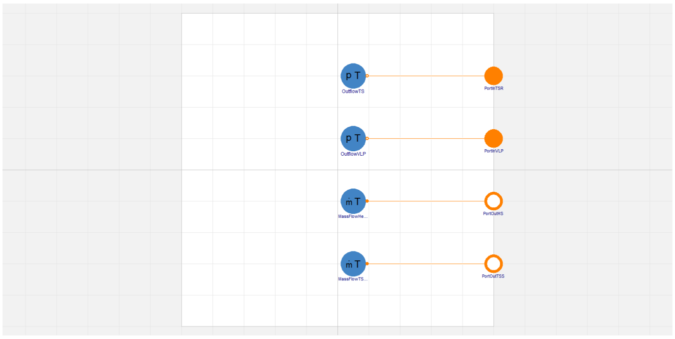


Figure 8.6 Cryoplant.

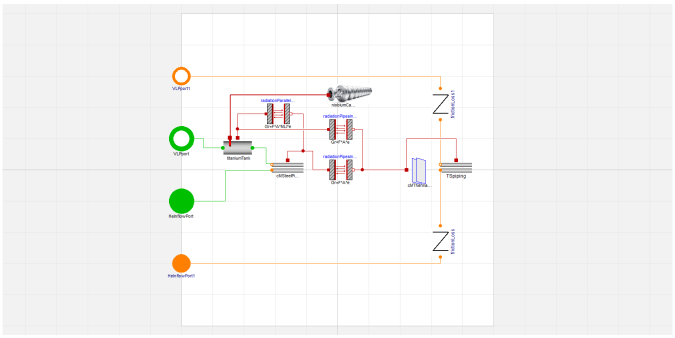


Figure 8.7 Cryomodule.

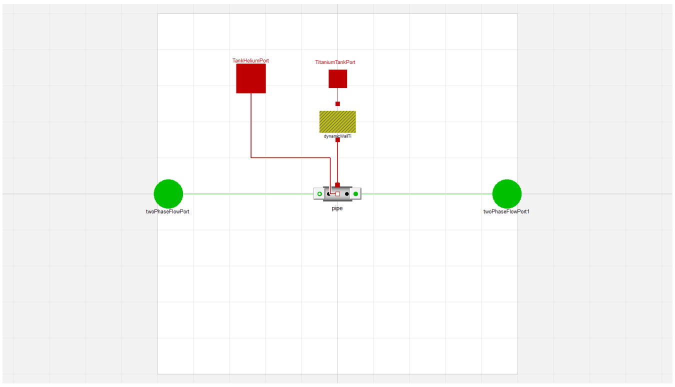


Figure 8.8 TitaniumTankPipe.

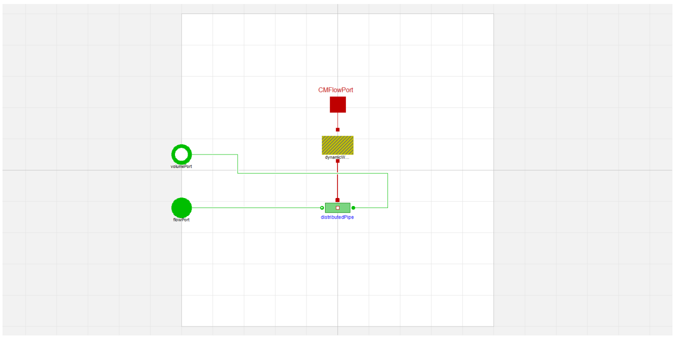


Figure 8.9 CMSteelPipingTwoPhase.

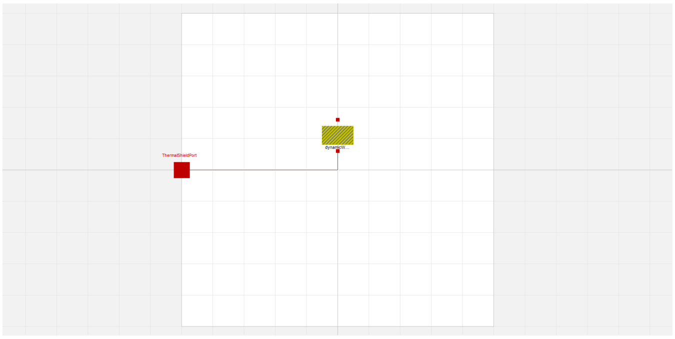


Figure 8.10 CMThermalShields and NiobiumCavity.

8.2 Appendix II - Dymola Components, Modelica Code

CryoPipeDWReynolds

```
model CryoPipeDWReynolds
  "Simple pipe with heat capacitance through the DynamicWall
  component."
  parameter Integer n=10 "Number of control volumes";
  parameter Modelica.SIunits.Length L "Pipe length";
  parameter Modelica.SIunits.Length OD "Outer pipe diameter";
  parameter Modelica.SIunits.Length thk "Thickness of pipe, used
  for Rw";
  parameter Modelica.SIunits.Density rho=7900 "Density of the
  pipe wall";
  parameter Modelica.SIunits.HeatFlowRate Q_vb=Q_vb
    "Heat load from vaccum barrier";
  parameter Modelica.SIunits.HeatFlowRate Q_fs=Q_fs
    "Heat load from fixed supports";
  parameter Modelica.SIunits.HeatFlowRate Q_ss=Q_ss
    "Heat load from sliding supports";
  parameter Modelica.SIunits.Temp_K T_start_in = 300
    "Initial temperature for inflow";
  parameter Modelica.SIunits.Temp_K T_start_out = 300
    "Initial temperature for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_in = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_out = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.MassFlowRate m_flow = 0.3
    "Mass flow rate in the system";
  parameter Real zeta=0.03*L/n/OD;
  parameter Modelica.SIunits.CoefficientOfHeatTransfer
  alpha0=200;

  replaceable package Medium =
  VaporCycle.Media.Naturals.HydrogenMbwr;

  final parameter Modelica.SIunits.Mass m =
  rho*((OD/2)^2-((OD-2*thk)/2)^2)*Modelica.Constants.pi*L/n;

  Modelon.ThermoFluid.FlowChannels.DistributedPipe
  distributedPipe(n=n,
    redeclare package Medium = Medium,
    L=L,
    T_start_in(displayUnit="K") = T_start_in,
    T_start_out(displayUnit="K") = T_start_out,
    p_start_out=p_start_out,
    p_start_in=p_start_in,
    useHeatTransfer=true,
    D=(OD - 2*thk),
    redeclare model Friction =
        Modelon.ThermoFluid.FlowChannels.PipeResistances.
        VariableZeta (zeta=zeta),
    redeclare model HeatTransfer =
        Modelon.ThermoFluid.FlowChannels.HeatTransfer.
        DittusBoelterAdjustable)
    annotation (Placement(
    transformation(extent={{-10,-10},{10,10}})));

  WallComponents.DynamicWallSS dynamicWall(
    n=n,
    Q_vb=Q_vb,
    Q_fs=Q_fs,
    Q_ss=Q_ss,
    T0(displayUnit="K") = linspace(
          T_start_in,
          T_start_out,
          n),
    thk=thk,
    A=Modelica.Constants.pi*OD*L/n,
    m=fill(m, n))
                annotation (Placement(transformation(extent=
                {{-10,30},{10,50}})));
  Modelon.ThermoFluid.Interfaces.FlowPort
  portA(redeclare package Medium =
        Medium, m_flow(start=m_flow))
    annotation (Placement(transformation(extent={{-112,-12},
    {-88,12}}),
        iconTransformation(extent={{-110,0},{-70,40}})));
  Modelon.ThermoFluid.Interfaces.FlowPort portB(redeclare package
  Medium = Medium, m_flow(start=-m_flow))
    annotation (Placement(transformation(extent={{88,-12},{112,
    12}}),
        iconTransformation(extent={{90,0},{130,40}})));
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a[n]
  PipeHeatPort
    annotation (Placement(transformation(extent={{-10,62},
    {10,82}}),
        iconTransformation(extent={{-10,62},{10,82}})));
equation

  connect(portA, distributedPipe.portA) annotation (Line(
      points={{-100,0},{-10,0}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(distributedPipe.portB, portB) annotation (Line(
      points={{10,0},{100,0}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(dynamicWall.qa, PipeHeatPort) annotation (Line(
      points={{0,50},{0,72}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(distributedPipe.q, dynamicWall.qb) annotation (Line(
      points={{0,5},{0,30}},
      color={191,0,0},
      thickness=0.5,
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,-100},{100,100}}), graphics), Icon(
  coordinateSystem(
          preserveAspectRatio=false, extent={{-100,-100},
          {100,100}}),
        graphics={
        Rectangle(
          extent={{-70,60},{90,-20}},
          lineColor={0,0,0},
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder),
        Line(
          points={{-50,-36},{70,-36}},
          color={255,128,0},
          thickness=0.5,
          smooth=Smooth.None),
        Polygon(
          points={{70,-36},{34,-24},{34,-48},{70,-36}},
          lineColor={255,128,0},
          lineThickness=0.5,
          fillPattern=FillPattern.HorizontalCylinder,
          smooth=Smooth.None,
          fillColor={255,128,0}),
        Text(
          extent={{-90,-40},{110,-80}},
          lineColor={0,0,0},
          textString="%name"),
        Ellipse(
          extent={{-48,28},{-32,12}},
          lineColor={0,0,0},
          fillPattern=FillPattern.Solid,
          fillColor={0,0,0}),
        Ellipse(
          extent={{52,28},{68,12}},
          lineColor={0,0,0},
          fillPattern=FillPattern.Solid,
          fillColor={0,0,0}),
        Text(
          extent={{-10,30},{30,10}},
          lineColor={0,0,0},
          fillColor={0,0,0},
          fillPattern=FillPattern.Solid,
          textString="%n"),
        Rectangle(
          extent={{-70,62},{90,40}},
          lineColor={255,0,0},
          fillColor={255,0,0},
          fillPattern=FillPattern.Solid)}));
end CryoPipeDWReynolds;
```

DynamicWallSS

```
model DynamicWallSS "Dynamic wall with heat resistance"
  extends Modelon.ThermoFluid.Solids.Interfaces.Wall(
                                                      qa(T(start=T0)),
                                                      qb(T(start=T0)));
  extends Modelon.ThermoFluid.Icons.WallDyn;

  parameter Modelica.SIunits.Mass[n] m "Metal mass per lump";
  parameter Modelica.SIunits.SpecificHeatCapacity Cp0=494
    "Initial metal heat capacity";
  parameter Modelica.SIunits.ThermalResistance Rw0=thk/(A*16)
    "Initial heat resistance in metal";
  DistributionSystem.Units.Temp_K[n] Tm(start=T0)
    "Metal mean temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Temperature[n] T0=ones(n)*300.0
    "Metal start temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Length thk "Thickness of the pipe
  wall";
  parameter Modelica.SIunits.Length A "Area of _one lump_ of the
  pipe wall";
  parameter Modelica.SIunits.HeatFlowRate Q_vb=Q_vb
    "Heat load from vaccum barrier";
  parameter Modelica.SIunits.HeatFlowRate Q_fs=Q_fs
    "Heat load from fixed supports";
  parameter Modelica.SIunits.HeatFlowRate Q_ss=Q_ss
    "Heat load from sliding supports";
  Modelica.SIunits.SpecificHeatCapacity[n] CpVar
    "Elementwise Cp from temperature";
  Modelica.SIunits.ThermalResistance[n] RwVar
    "Elementwise Rw from inverse thermal conductivity based on
    temperature";
equation
  for i in 1:n loop
    CpVar[i]=-0.00000000115115*((qa[i].T+qb[i].T)/2)^5+
    0.00000112374*((qa[i].T+qb[i].T)/2)^4-0.000392079*(
    (qa[i].T+qb[i].T)/2)^3+0.0522538*((qa[i].T+qb[i].T)/2)^2+
    0.141301*((qa[i].T+qb[i].T)/2);
    RwVar[i]=thk/(A*(0.0000008*((qa[i].T+qb[i].T)/2)^3-0.0006*(
    (qa[i].T+qb[i].T)/2)^2+0.1477*((qa[i].T+qb[i].T)/2)-0.4952));

    m[i]*CpVar[i]*der(Tm[i]) = qa[i].Q_flow + qb[i].Q_flow;

    qa[i].Q_flow = (qa[i].T - Tm[i])*2/RwVar[i] +
    (Q_vb+Q_fs+Q_ss)/n; //Added support and vacuum
    barrier heat loads
    qb[i].Q_flow = (qb[i].T - Tm[i])*2/RwVar[i] +
    (Q_vb+Q_fs+Q_ss)/n; //Added support and vacuum
    barrier heat loads
  end for;
annotation(Evaluate=true,
            Diagram(coordinateSystem(preserveAspectRatio=false,
            extent={{-100,-100},{100,100}}), graphics));
```

```
end DynamicWallSS;
```

DynamicWallNb

```
model DynamicWallNb "Dynamic wall with conduction heat
resistance."
  extends Modelon.ThermoFluid.Solids.Interfaces.Wall(
     qa(T(start=T0)), qb(T(start=T0)));
  extends Modelon.ThermoFluid.Icons.WallDyn;

  parameter Modelica.SIunits.Mass[n] m "Metal mass per lump";
  parameter Modelica.SIunits.SpecificHeatCapacity Cp0=494
    "Initial metal heat capacity";
  parameter Modelica.SIunits.ThermalResistance Rw0=thk/(A*16)
    "Initial heat resistance in metal";
  DistributionSystem.Units.Temp_K[n] Tm(start=T0)
    "Metal mean temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Temperature[n] T0=ones(n)*300.0
    "Metal start temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Length thk "Thickness of the pipe
  wall/specimen";
  parameter Modelica.SIunits.Length A "Area of _one lump_ of the
  pipe wall";
  Modelica.SIunits.SpecificHeatCapacity[n] CpVar
    "Elementwise Cp from temperature";
  Modelica.SIunits.ThermalResistance[n] RwVar
    "Elementwise Rw from inverse thermal conductivity based on
    temperature";

equation
  for i in 1:n loop
    if Tm[i] > 20 then
      CpVar[i]=2.0226423917e-11*Tm[i]^6-2.0451534634e-8*Tm[i]^5+
      7.9550021963e-6*Tm[i]^4-0.0014601740994*Tm[i]^3+
      0.11758205083*Tm[i]^2-1.2495904095*Tm[i]+3.0875766442;
    else
      CpVar[i]=0.0009*Tm[i]^3 + 0.0076*Tm[i]^2 + 0.0449*Tm[i]
      +0.0422;
    end if;
    if Tm[i] >= 30 then
      RwVar[i]=thk/(A*(-3.8984111598e-10*Tm[i]^5+3.9537530644e-
      7*Tm[i]^4-1.5506337607e-
      4*Tm[i]^3+0.029268800879*Tm[i]^2-2.6465534609*Tm[i]+
      141.29752365));
    else
      RwVar[i]=thk/(A*(-0.11689520760*Tm[i]^2+6.4368583162*Tm[i]
      +1.9644659451));
    end if;

    m[i]*CpVar[i]*der(Tm[i]) = qa[i].Q_flow + qb[i].Q_flow;

    qa[i].Q_flow = (qa[i].T - Tm[i])*2/RwVar[i];
    qb[i].Q_flow = (qb[i].T - Tm[i])*2/RwVar[i];

  end for;
end DynamicWallNb;
```

DynamicWallTi

```
model DynamicWallTi "Dynamic wall with heat resistance"
  extends Modelon.ThermoFluid.Solids.Interfaces.Wall(
     qa(T(start=T0)), qb(T(start=T0)));
  extends Modelon.ThermoFluid.Icons.WallDyn;

  parameter Modelica.SIunits.Mass[n] m "Metal mass per lump";
  parameter Modelica.SIunits.SpecificHeatCapacity Cp0=494
    "Initial metal heat capacity";
  parameter Modelica.SIunits.ThermalResistance Rw0=thk/(A*16)
    "Initial heat resistance in metal";
  DistributionSystem.Units.Temp_K[n] Tm(start=T0)
    "Metal mean temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Temperature[n] T0=ones(n)*300.0
    "Metal start temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Length thk "Thickness of the pipe
  wall/specimen";
  parameter Modelica.SIunits.Length A "Area of _one lump_ of the
  pipe wall";
  Modelica.SIunits.SpecificHeatCapacity[n] CpVar
    "Elementwise Cp from temperature";
  Modelica.SIunits.ThermalResistance[n] RwVar
    "Elementwise Rw from inverse thermal conductivity based on
    temperature";
  constant Real a = -2.398794842;
  constant Real b = 8.970743802;
  constant Real c = -29.19286973;
  constant Real d = 54.87139779;
  constant Real e = -59.67137228;
  constant Real f = 38.89321714;
  constant Real g = -14.94175848;
  constant Real h = 3.111616089;
  constant Real k = -0.270452768;

equation
  for i in 1:n loop
    if Tm[i] > 20 then
      CpVar[i]=-3.5147820378e-9*Tm[i]^5+3.0057506888e-6*Tm[i]^4-
      9.2045002329e-4*Tm[i]^3+0.11191552397*Tm[i]^2-
      1.7076312852*Tm[i]+4.3306205877;
    else
      CpVar[i] = 0.0008*Tm[i].^3 - 0.0029*Tm[i].^2 + 0.0742*Tm[i]
      + 0.005;
    end if;

    RwVar[i]=thk/(A*10^(a+b*(log10(Tm[i]))+c*(log10(Tm[i]))^2+
    d*(log10(Tm[i]))^3+e*(log10(Tm[i]))^4+f*(log10(Tm[i]))^5+
    g*(log10(Tm[i]))^6+h*(log10(Tm[i]))^7+k*(log10(Tm[i]))^8));

    m[i]*CpVar[i]*der(Tm[i]) = qa[i].Q_flow + qb[i].Q_flow;

    qa[i].Q_flow = (qa[i].T - Tm[i])*2/RwVar[i];
    qb[i].Q_flow = (qb[i].T - Tm[i])*2/RwVar[i];
  end for;
end DynamicWallTi;
```

DynamicWallAl

```
model DynamicWallAl "Dynamic wall with heat resistance"
  extends Modelon.ThermoFluid.Solids.Interfaces.Wall(
      qa(T(start=T0)), qb(T(start=T0)));
  extends Modelon.ThermoFluid.Icons.WallDyn;

  parameter Modelica.SIunits.Mass[n] m "Metal mass per lump";
  parameter Modelica.SIunits.SpecificHeatCapacity Cp0=494
    "Initial metal heat capacity";
  parameter Modelica.SIunits.ThermalResistance Rw0=thk/(A*16)
    "Initial heat resistance in metal";
  DistributionSystem.Units.Temp_K[n] Tm(start=T0)
    "Metal mean temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Temperature[n] T0=ones(n)*300.0
    "Metal start temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Length thk "Thickness of the pipe
  wall/specimen";
  parameter Modelica.SIunits.Length A "Area of _one lump_ of the
  pipe wall";
  Modelica.SIunits.SpecificHeatCapacity[n] CpVar
    "Elementwise Cp from temperature";
  Modelica.SIunits.ThermalResistance[n] RwVar
    "Elementwise Rw from inverse thermal conductivity based on
    temperature";
  constant Real a = 23.39172;
  constant Real b = -148.5733;
  constant Real c = 422.1917;
  constant Real d = -653.6664;
  constant Real e = 607.0402;
  constant Real f = -346.152;
  constant Real g = 118.4276;
  constant Real h = -22.2781;
  constant Real k = 1.770187;

equation
  for i in 1:n loop
    if Tm[i] > 20 then
      CpVar[i]=-5.0858382990e-9*Tm[i]^5+4.4933166438e-6*Tm[i]^4-
      0.0014348060969*Tm[i]^3+0.18473605304*Tm[i]^2-
      3.4580884187*Tm[i]+10.842981412;
    else
```

```
      CpVar[i]=0.0011*Tm[i]^3-0.0029*Tm[i]^2+0.0622*Tm[i]-0.011;
    end if;
    RwVar[i]=thk/(A*10^(a+b*(log10(Tm[i]))+c*(log10(Tm[i]))^2+
    d*(log10(Tm[i]))^3+e*(log10(Tm[i]))^4+
    f*(log10(Tm[i]))^5+g*(log10(Tm[i]))^6+h*(log10(Tm[i]))^7+
    k*(log10(Tm[i]))^8));

    m[i]*CpVar[i]*der(Tm[i]) = qa[i].Q_flow + qb[i].Q_flow;

    qa[i].Q_flow = (qa[i].T - Tm[i])*2/RwVar[i];
    qb[i].Q_flow = (qb[i].T - Tm[i])*2/RwVar[i];
  end for;
end DynamicWallAl;
```

DynamicWallCu

```
model DynamicWallCu "Dynamic wall with heat resistance"
  extends Modelon.ThermoFluid.Solids.Interfaces.Wall(
        qa(T(start=T0)), qb(T(start=T0)));
  extends Modelon.ThermoFluid.Icons.WallDyn;

  parameter Modelica.SIunits.Mass[n] m "Metal mass per lump";
  parameter Modelica.SIunits.SpecificHeatCapacity Cp0=494
    "Initial metal heat capacity";
  parameter Modelica.SIunits.ThermalResistance Rw0=0.003/(A*16)
    "Initial heat resistance in metal";
  DistributionSystem.Units.Temp_K[n] Tm(start=T0)
    "Metal mean temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Temperature[n] T0=ones(n)*300.0
    "Metal start temperature, direction along n as on side qa";
  parameter Modelica.SIunits.Length thk "Thickness of the pipe
  wall";
  parameter Modelica.SIunits.Length A "Area of _one lump_ of the
  pipe wall";
  parameter Modelica.SIunits.HeatFlowRate Q_vb=Q_vb
    "Heat load from vaccum barrier";
  parameter Modelica.SIunits.HeatFlowRate Q_fs=Q_fs
    "Heat load from fixed supports";
  parameter Modelica.SIunits.HeatFlowRate Q_ss=Q_ss
    "Heat load from sliding supports";
  Modelica.SIunits.SpecificHeatCapacity[n] CpVar
    "Elementwise Cp from temperature";
  Modelica.SIunits.ThermalResistance[n] RwVar
    "Elementwise Rw from inverse thermal conductivity based on
    temperature";
  constant Real a = 1.8743;
  constant Real b = -0.41538;
  constant Real c = -0.6018;
  constant Real d = 0.13294;
  constant Real e = 0.26426;
  constant Real f = -0.0219;
  constant Real g = -0.051276;
  constant Real h = 0.0014871;
  constant Real k = 0.003723;

equation
  for i in 1:n loop
    CpVar[i]=1.2682110150e-11*Tm[i]^6-1.4399163352e-8*Tm[i]^5+
    6.3296511850e-6*Tm[i]^4-1.3291319283e-3*Tm[i]^3+
    0.12535751222*Tm[i]^2-1.8004787189*Tm[i]+5.2456099913;
    RwVar[i]=thk/(A*10^(a+b*(log10(Tm[i]))+c*(log10(Tm[i]))^2+
    d*(log10(Tm[i]))^3+e*(log10(Tm[i]))^4+f*(log10(Tm[i]))^5+
    g*(log10(Tm[i]))^6+h*(log10(Tm[i]))^7+k*(log10(Tm[i]))^8));

    m[i]*CpVar[i]*der(Tm[i]) = qa[i].Q_flow + qb[i].Q_flow;

    qa[i].Q_flow = (qa[i].T - Tm[i])*2/RwVar[i] +
    (Q_vb+Q_fs+Q_ss)/n;
    qb[i].Q_flow = (qb[i].T - Tm[i])*2/RwVar[i] +
    (Q_vb+Q_fs+Q_ss)/n;
  end for;
end DynamicWallCu;
```

DistributionPipe

```
model DistributionPipe
  parameter Integer n=10;
  parameter Modelica.SIunits.Length L=50 "Pipe length";
  parameter Modelica.SIunits.Density rhoSS=rhoSS
    "Density of the stainless steel pipes";
  parameter Modelica.SIunits.Length ODHS=ODHS
    "Outer diameter of helium supply pipe";
  parameter Modelica.SIunits.Length ODVLP=ODVLP
    "Outer diameter of VLP return pipe";
  parameter Modelica.SIunits.Length ODTSS=ODTSS
    "Outer diameter of thermal shield supply pipe";
  parameter Modelica.SIunits.Length ODTSR=ODTSR
    "Outer diameter of thermal shield return pipe";
  parameter Modelica.SIunits.Length thkHS=thkHS
    "Thickness of helium supply pipe";
  parameter Modelica.SIunits.Length thkVLP=thkVLP
    "Thickness of VLP return pipe";
  parameter Modelica.SIunits.Length thkTSS=thkTSS
    "Thickness of thermal shield supply pipe";
  parameter Modelica.SIunits.Length thkTSR=thkTSR
    "Thickness of thermal shield return pipe";
  parameter Real eSS=eSS "Emissivity of stainless steel (0.07)";
  parameter Real eAl=eAl "Emissivity of aluminum or copper
  (same 0.04)";
  parameter Modelica.SIunits.Length ODTS=ODTS
    "Outer diameter of thermal shield ";
  parameter Modelica.SIunits.Length tTS=tTS "Thickness of thermal
  shield";
  parameter Modelica.SIunits.Density rhoTS=rhoTS
```

```
    "Density of thermal shield material";
  parameter Modelica.SIunits.Length ODEE=ODEE
    "Outer diameter of external envelope";
  parameter Modelica.SIunits.Length tEE=tEE "Thickness of
  external envelope";
  parameter Modelica.SIunits.Density rhoEE=rhoEE
    "Density of external envelope material";
  parameter Modelica.SIunits.CoefficientOfHeatTransfer zeta
    "Heat transfer coefficient for constant heat transfer between
    pipe and medium";
  parameter Modelica.SIunits.Pressure p_start_inHS = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outHS = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inTSS = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outTSS = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inTSR = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outTSR = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inVLP = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outVLP = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Temp_K T_startHS = 300
    "Start temperature of the helium supply pipe";
  parameter Modelica.SIunits.Temp_K T_startVLP = 300
    "Start temperature of the vapor low pressure pipe";
  parameter Modelica.SIunits.Temp_K T_startTSS = 300
    "Start temperature of the thermal shield supply pipe";
  parameter Modelica.SIunits.Temp_K T_startTSR = 300
    "Start temperature of the thermal shield return pipe";
  parameter Modelica.SIunits.MassFlowRate m_flow = 0.3
    "Mass flow rate in the system";
  replaceable package Medium =
  VaporCycle.Media.Naturals.HydrogenMbwr;

  final parameter Modelica.SIunits.Mass mTS = ((ODTS/2)^2-
  ((ODTS-2*tTS)/2)^2)*Modelica.Constants.pi*L/n*rhoTS;
  final parameter Modelica.SIunits.Mass mEE = ((ODEE/2)^2-
  ((ODEE-2*tEE)/2)^2)*Modelica.Constants.pi*L/n*rhoEE;

  CryoPipeDWReynolds HeSupply(
    redeclare package Medium = Medium,
    L=L,
    thk=thkHS,
    OD=ODHS,
    rho=rhoSS,
    Q_vb=3*0.648/n,
    Q_fs=34*0.486/n,
    Q_ss=55*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inHS,
    p_start_out(displayUnit="Pa") = p_start_outHS,
    T_start_in=T_startHS,
    T_start_out=T_startHS,
    n=n,
    m_flow=m_flow,
    zeta=zeta*L/ODHS,
    alpha0=220)          annotation (Placement(transformation(
    extent={{-150,
            -140},{-130,-120}})));
  CryoPipeDWReynolds VLPreturn(
    redeclare package Medium = Medium,
    L=L,
    thk=thkVLP,
    OD=ODVLP,
    rho=rhoSS,
    p_start_in(displayUnit="Pa") = p_start_inVLP,
    p_start_out(displayUnit="Pa") = p_start_outVLP,
    T_start_in=T_startVLP,
    T_start_out=T_startVLP,
    n=n,
    m_flow=m_flow,
    zeta=zeta*L/ODVLP,
    Q_vb=3*0.648/n,
    Q_fs=34*0.486/n,
    Q_ss=55*0.162/n,
    alpha0=20)
    annotation (Placement(transformation(extent=
    {{16,-52},{-4,-32}})));
  CryoPipeDWReynolds TSsupply(
    redeclare package Medium = Medium,
    L=L,
    thk=thkTSS,
    OD=ODTSS,
    rho=rhoSS,
    Q_vb=3*0.648/n,
    Q_fs=34*0.486/n,
    Q_ss=55*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inTSS,
    p_start_out(displayUnit="Pa") = p_start_outTSS,
    T_start_in=T_startTSS,
    T_start_out=T_startTSS,
    n=n,
    m_flow=m_flow,
    zeta=zeta*L/ODTSS,
    alpha0=500)
    annotation (Placement(transformation(extent=
    {{-82,-96},{-62,-76}})));
  CryoPipeDWReynolds TSreturn(
    redeclare package Medium = Medium,
    L=L,
    thk=thkTSR,
    OD=ODTSR,
    rho=rhoSS,
    Q_vb=3*0.648/n,
    Q_fs=34*0.486/n,
    Q_ss=55*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inTSR,
    p_start_out(displayUnit="Pa") = p_start_outTSR,
    T_start_in=T_startTSR,
    T_start_out=T_startTSR,
    n=n,
    m_flow=m_flow,
    zeta=zeta*L/ODTSR,
    alpha0=500)
    annotation (Placement(transformation(extent=
    {{66,-6},{46,14}})));
  WallComponents.DynamicWallSS ThermalShield(
    n=n,
    thk=tTS,
    m=fill(mTS, n),
    Q_vb=0,
    Q_fs=0,
    Q_ss=0,
    A=Modelica.Constants.pi*L*ODTS/n,
    T0(displayUnit="K") = ones(n)*T_startTSR)
    annotation (Placement(transformation(extent=
    {{46,100},{66,80}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInHS(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{-248,-36},{-228,-16}}),
        iconTransformation(
          extent={{-248,-36},{-228,-16}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInTSS(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{-246,-94},{-226,-74}}),
        iconTransformation(
          extent={{-246,-94},{-226,-74}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInVLP(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{230,14},{250,34}}),
        iconTransformation(extent={{230,14},
            {250,34}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInTSR(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{230,68},{250,88}}),
        iconTransformation(extent={{230,68},
            {250,88}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutTSR(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{-248,70},{-228,90}}),
        iconTransformation(extent={{-248,70},
            {-228,90}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutVLP(
redeclare package Medium
    = Medium) annotation (Placement(
      transformation(extent={{-248,12},{-228,32}}),
      iconTransformation(extent={{-248,12},
          {-228,32}})));
Modelon.ThermoFluid.Interfaces.VolumePort PortOutTSS(
redeclare package Medium
    = Medium) annotation (Placement(
      transformation(extent={{230,-94},{250,-74}}),
      iconTransformation(extent={{230,-94},
          {250,-74}})));
Modelon.ThermoFluid.Interfaces.VolumePort PortOutHS(
redeclare package Medium
    = Medium) annotation (Placement(
      transformation(extent={{230,-36},{250,-16}}),
      iconTransformation(extent={{230,-36},
          {250,-16}})));
Radiation.RadiationParallelPipes RadiationVLPtoHS[n](
  r1=ODVLP/2,
  r2=ODHS/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={-140,-102})));
Radiation.RadiationParallelPipes RadiationTSStoVLP[n](
  r1=ODTSS/2,
  r2=ODVLP/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={-72,-60})));
Radiation.RadiationParallelPipes RadiationTSRtoVLP[n](
  r1=ODTSR/2,
  r2=ODVLP/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={6,-18})));
Radiation.RadiationParallelPipes RadiationTSStoHS[n](
  r1=ODVLP/2,
  r2=ODHS/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={2,-100})));
  Radiation.RadiationParallelPipes RadiationTSRtoHS[n](
    r1=ODTSR/2,
    r2=ODHS/2,
    s=0.2,
    L=L,
    e1=eSS,
    e2=eSS) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={62,-104})));
  Radiation.RadiationParallelPipes RadiationTSRtoTSS[n](
    r1=ODTSR/2,
    r2=ODTSS/2,
    s=0.2,
    L=L,
    e1=eSS,
    e2=eSS) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={60,-62})));
  Radiation.RadiationPipesInsidePipe RadiationHSout[n](
    r1=ODTS/2,
    r2=ODHS/2,
    L=L,
    e=eSS,
    s=0.12) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-158,44})));
  Radiation.RadiationPipesInsidePipe RadiationTSSout[n](
    r1=ODTS/2,
    r2=ODTSS/2,
    L=L,
    e=eSS,
    s=0.11) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-104,44})));
  Radiation.RadiationPipesInsidePipe RadiationVLPout[n](
    r1=ODTS/2,
    r2=ODVLP/2,
    L=L,
    e=eSS,
    s=0.3) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-52,44})));
  Radiation.RadiationPipesInsidePipe RadiationTSRout[n](
    r1=ODTS/2,
    r2=ODTSR/2,
    L=L,
    e=eSS,
    s=0.11) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-4,42})));
  Modelon.ThermoFluid.Interfaces.FlowHeatPort AmbiencePort[n]
    annotation (Placement(transformation(extent=
    {{-94,106},{-74,126}})));
  WallComponents.DynamicWallSS ExternalEnvelope(
    n=n,
    thk=tEE,
    m=fill(mEE, n),
    Q_vb=0,
    Q_fs=0,
    Q_ss=0,
    A=Modelica.Constants.pi*L*ODEE/n,
    T0(displayUnit="K") = ones(n)*T_startTSR)
    annotation (Placement(transformation(extent=
    {{-56,92},{-36,112}})));
  Radiation.RadiationConcentricPipes RadiationTStoEE[n](
    e2=eSS,
    r1=ODTS/2,
    r2=ODEE/2,
    L=L/n,
    e1=eAl) annotation (Placement(transformation(extent=
    {{6,90},{-14,110}})));
equation
  connect(PortInHS, HeSupply.portA) annotation (Line(
      points={{-238,-26},{-162,-26},{-162,-128},{-149,-128}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(HeSupply.portB, PortOutHS)  annotation (Line(
      points={{-129,-128},{158,-128},{158,-26},{240,-26}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSsupply.portB, PortOutTSS)  annotation (Line(
      points={{-61,-84},{38,-84},{38,-88},{136,-88},
      {136,-84},{240,-84}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(VLPreturn.portA,PortInVLP)  annotation (Line(
      points={{15,-40},{128.5,-40},{128.5,24},{240,24}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSreturn.portA,PortInTSR)  annotation (Line(
      points={{65,6},{136.5,6},{136.5,78},{240,78}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortOutTSR, TSreturn.portB) annotation (Line(
      points={{-238,80},{-74,80},{-74,6},{45,6}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortOutVLP, VLPreturn.portB)  annotation (Line(
      points={{-238,22},{-114,22},{-114,-40},{-5,-40}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortInTSS, TSsupply.portA) annotation (Line(
      points={{-236,-84},{-178,-84},{-178,-66},{-130,-66},
      {-130,-84},{-81,
          -84}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort,RadiationTSStoHS. port_a)
  annotation (Line(
      points={{-140,-122.8},{-140,-110},{2,-110}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSStoHS.port_b, TSsupply.PipeHeatPort)
  annotation (Line(
      points={{2,-90},{2,-76},{-72,-76},{-72,-78.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSStoVLP.port_b, VLPreturn.PipeHeatPort)
  annotation (Line(
      points={{-72,-50},{-72,-32},{6,-32},{6,-34.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort,RadiationVLPtoHS. port_a)
  annotation (Line(
      points={{-140,-122.8},{-84,-122.8},{-84,-118},
      {-100,-118},{-100,-116},
          {-140,-116},{-140,-112}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoHS.port_a, HeSupply.PipeHeatPort)
  annotation (Line(
      points={{62,-114},{62,-122.8},{-140,-122.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoVLP.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
      points={{6,-8},{6,11.2},{56,11.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort,RadiationTSRtoTSS. port_a)
  annotation (Line(
      points={{-72,-78.8},{-40,-78.8},{-40,-78},{-6,-78},
      {-6,-76},{60,-76},{60,-72}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort , RadiationTSStoVLP.port_a)
  annotation (Line(
      points={{-72,-78.8},{-72,-70}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(VLPreturn.PipeHeatPort,RadiationTSRtoVLP. port_a)
  annotation (Line(
      points={{6,-34.8},{6,-28}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationVLPtoHS.port_b, VLPreturn.PipeHeatPort)
  annotation (Line(
      points={{-140,-92},{-140,-72},{30,-72},{30,-34.8},
      {6,-34.8}},color={191,0,0},smooth=Smooth.None));
  connect(RadiationTSRtoHS.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
      points={{62,-94},{62,-82},{82,-82},{82,11.2},{56,11.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoTSS.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
      points={{60,-52},{60,-12},{34,-12},{34,11.2},{56,11.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort, RadiationHSout.port_a)
  annotation (Line(
      points={{-140,-122.8},{-150,-122.8},{-150,-120},
      {-158,-120},{-158,34}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort, RadiationTSSout.port_a)
  annotation (Line(
      points={{-72,-78.8},{-88,-78.8},{-88,-76},
      {-104,-76},{-104,34}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(VLPreturn.PipeHeatPort, RadiationVLPout.port_a)
  annotation (Line(
      points={{6,-34.8},{-14,-34.8},{-14,-22},{-52,-22},
      {-52,34}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSreturn.PipeHeatPort, RadiationTSRout.port_a)
  annotation (Line(
      points={{56,11.2},{28,11.2},{28,14},{-4,14},{-4,32}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRout.port_b, ThermalShield.qa)
  annotation (Line(
      points={{-4,52},{56,52},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationVLPout.port_b, ThermalShield.qa)
  annotation (Line(
      points={{-52,54},{-52,76},{38,76},{38,80},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSSout.port_b, ThermalShield.qa)
  annotation (Line(
      points={{-104,54},{-104,84},{24,84},{24,84},
      {34,84},{34,80},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationHSout.port_b, ThermalShield.qa)
  annotation (Line(
      points={{-158,54},{-158,66},{-166,66},{-166,76},
      {-56,76},{-56,80},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(ThermalShield.qb, RadiationTStoEE.port_a)
  annotation (Line(
      points={{56,100},{6,100}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTStoEE.port_b, ExternalEnvelope.qb)
  annotation (Line(
      points={{-14,100},{-30,100},{-30,92},{-46,92}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(ExternalEnvelope.qa, AmbiencePort)
  annotation (Line(
      points={{-46,112},{-66,112},{-66,116},{-84,116}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSreturn.PipeHeatPort, ThermalShield.qa)
  annotation (Line(
      points={{56,11.2},{56,42},{74,42},{74,80},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-240,
            -180},{240,140}}), graphics), Icon(coordinateSystem(
          preserveAspectRatio=false, extent=
          {{-240,-180},{240,140}}), graphics={
          Text(
          extent={{-258,-78},{266,-154}},
          lineColor={0,0,127},
          lineThickness=0.5,
          fillColor={0,0,0},
          fillPattern=FillPattern.Solid,
          textString="%name"),
        Rectangle(
          extent={{-224,74},{226,-84}},
          lineColor={0,0,0},
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder),
        Rectangle(
          extent={{-168,106},{-74,74}},
          lineColor={165,0,0},
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={168,0,0})}));
end DistributionPipe;
```

TransferLine

```
model TransferLine
  parameter Integer n=10;
  parameter Modelica.SIunits.Length L=50 "Pipe length";
  parameter Modelica.SIunits.Density rhoSS=rhoSS
    "Density of the stainless steel pipes";
  parameter Modelica.SIunits.Length ODHS=ODHS
    "Outer diameter of helium supply pipe";
  parameter Modelica.SIunits.Length ODVLP=ODVLP
    "Outer diameter of VLP return pipe";
  parameter Modelica.SIunits.Length ODTSS=ODTSS
    "Outer diameter of thermal shield supply pipe";
  parameter Modelica.SIunits.Length ODTSR=ODTSR
    "Outer diameter of thermal shield return pipe";
  parameter Modelica.SIunits.Length thkHS=thkHS
    "Thickness of helium supply pipe";
  parameter Modelica.SIunits.Length thkVLP=thkVLP
    "Thickness of VLP return pipe";
  parameter Modelica.SIunits.Length thkTSS=thkTSS
    "Thickness of thermal shield supply pipe";
  parameter Modelica.SIunits.Length thkTSR=thkTSR
    "Thickness of thermal shield return pipe";
  parameter Real eSS=0.07 "Emissivity of stainless steel";
  parameter Real eAl=0.04 "Emissivity of aluminum or copper
  (same)";
  parameter Modelica.SIunits.Length ODTS=ODTS
    "Outer diameter of thermal shield ";
  parameter Modelica.SIunits.Length tTS=tTS "Thickness of thermal
  shield";
  parameter Modelica.SIunits.Density rhoTS=rhoTS
    "Density of thermal shield material";
  parameter Modelica.SIunits.Length ODEE=ODEE
    "Outer diameter of external envelope";
  parameter Modelica.SIunits.Length tEE=tEE "Thickness of
  external envelope";
  parameter Modelica.SIunits.Density rhoEE=rhoEE
    "Density of external envelope material";
  parameter Modelica.SIunits.CoefficientOfHeatTransfer zeta
    "Heat transfer coefficient for constant heat transfer between
    pipe and medium";
  parameter Modelica.SIunits.Pressure p_start_inHS = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outHS = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inTSS = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outTSS = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inTSR = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outTSR = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_inVLP = 1e5
    "Initial pressure for inflow and in pipe walls";
  parameter Modelica.SIunits.Pressure p_start_outVLP = 1e5
    "Initial pressure for outflow and in pipe walls";
  parameter Modelica.SIunits.Temp_K T_startHS = 300
    "Start temperature of the helium supply pipe";
  parameter Modelica.SIunits.Temp_K T_startVLP = 300
    "Start temperature of the vapor low pressure pipe";
  parameter Modelica.SIunits.Temp_K T_startTSS = 300
    "Start temperature of the thermal shield supply pipe";
  parameter Modelica.SIunits.Temp_K T_startTSR = 300
    "Start temperature of the thermal shield return pipe";
  parameter Modelica.SIunits.MassFlowRate m_flow = 0.3
    "Mass flow rate in the system";
  replaceable package Medium =
  VaporCycle.Media.Naturals.HydrogenMbwr;
  final parameter Modelica.SIunits.Mass mTS = ((
  ODTS/2)^2-((ODTS-2*tTS)/2)^2)*Modelica.Constants.pi*L/n*rhoTS;
  final parameter Modelica.SIunits.Mass mEE = ((
  ODEE/2)^2-((ODEE-2*tEE)/2)^2)*Modelica.Constants.pi*L/n*rhoEE;

  CryoPipeDWReynolds HeSupply(
    redeclare package Medium = Medium,
    L=L,
    thk=thkHS,
    OD=ODHS,
    rho=rhoSS,
    p_start_in(displayUnit="Pa") = p_start_inHS,
    p_start_out(displayUnit="Pa") = p_start_outHS,
    m_flow=m_flow,
    n=n,
    T_start_in(displayUnit="K") = T_startHS,
    T_start_out(displayUnit="K") = T_startHS,
    zeta=zeta*L/ODHS,
    Q_vb=2*0.648/n,
    Q_fs=10*0.486/n,
    Q_ss=20*0.162/n,
    alpha0=220)
    annotation (Placement(transformation(extent=
    {{-150,-140},{-130,-120}})));
  CryoPipeDWReynolds VLPreturn(
    redeclare package Medium = Medium,
    L=L,
    thk=thkVLP,
    OD=ODVLP,
    rho=rhoSS,
    Q_vb=2*0.648/n,
    Q_fs=10*0.486/n,
    Q_ss=20*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inVLP,
    p_start_out(displayUnit="Pa") = p_start_outVLP,
    m_flow=m_flow,
    n=n,
    T_start_in(displayUnit="K") = T_startVLP,
    T_start_out(displayUnit="K") = T_startVLP,
    zeta=zeta*L/ODVLP,
    alpha0=20)
    annotation (Placement(transformation(extent=
    {{16,-52},{-4,-32}})));
  CryoPipeDWReynolds TSsupply(
    redeclare package Medium = Medium,
    L=L,
    thk=thkTSS,
    OD=ODTSS,
    rho=rhoSS,
    Q_vb=2*0.648/n,
    Q_fs=10*0.486/n,
    Q_ss=20*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inTSS,
    p_start_out(displayUnit="Pa") = p_start_outTSS,
    m_flow=m_flow,
    n=n,
    T_start_in(displayUnit="K") = T_startTSS,
    T_start_out(displayUnit="K") = T_startTSS,
    zeta=zeta*L/ODTSS,
    alpha0=500)
    annotation (Placement(transformation(extent=
    {{-82,-96},{-62,-76}})));
  CryoPipeDWReynolds TSreturn(
    redeclare package Medium = Medium,
    L=L,
    thk=thkTSR,
    OD=ODTSR,
    rho=rhoSS,
    Q_vb=2*0.648/n,
    Q_fs=10*0.486/n,
    Q_ss=20*0.162/n,
    p_start_in(displayUnit="Pa") = p_start_inTSR,
    p_start_out(displayUnit="Pa") = p_start_outTSR,
    m_flow=m_flow,
    n=n,
    T_start_in(displayUnit="K") = T_startTSR,
    T_start_out(displayUnit="K") = T_startTSR,
    zeta=zeta*L/ODTSR,
    alpha0=500)
    annotation (Placement(transformation(extent=
    {{66,-6},{46,14}})));
  WallComponents.DynamicWallSS ThermalShield(
    n=n,
    thk=tTS,
    m=fill(mTS, n),
    Q_vb=0,
    Q_fs=0,
    Q_ss=0,
    A=Modelica.Constants.pi*ODTS*L/n,
    T0(displayUnit="K") = ones(n)*T_startTSR)
    annotation (Placement(transformation(extent=
    {{46,100},{66,80}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInHS(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{-248,-36},{-228,-16}}),
        iconTransformation(
          extent={{-248,-36},{-228,-16}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInTSS(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{-246,-94},{-226,-74}}),
        iconTransformation(
          extent={{-246,-94},{-226,-74}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInVLP(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{230,14},{250,34}}),
        iconTransformation(extent={{230,14},
            {250,34}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInTSR(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{230,68},{250,88}}),
        iconTransformation(extent={{230,68},
            {250,88}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutTSR(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{-248,70},{-228,90}}),
        iconTransformation(extent={{-248,70},
            {-228,90}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutVLP(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{-248,12},{-228,32}}),
        iconTransformation(extent={{-248,12},
            {-228,32}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutTSS(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{230,-94},{250,-74}}),
        iconTransformation(extent={{230,-94},
            {250,-74}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutHS(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{230,-36},{250,-16}}),
        iconTransformation(extent={{230,-36},
            {250,-16}})));
```

```
Radiation.RadiationParallelPipes RadiationVLPtoHS[n](
  r1=ODVLP/2,
  r2=ODHS/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={-134,-94})));
Radiation.RadiationParallelPipes RadiationTSStoVLP[n](
  r1=ODTSS/2,
  r2=ODVLP/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={-72,-54})));
Radiation.RadiationParallelPipes RadiationTSRtoVLP[n](
  r1=ODTSR/2,
  r2=ODVLP/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={6,-16})));
Radiation.RadiationParallelPipes RadiationTSStoHS[n](
  r1=ODVLP/2,
  r2=ODHS/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={8,-102})));
Radiation.RadiationParallelPipes RadiationTSRtoHS[n](
  r1=ODTSR/2,
  r2=ODHS/2,
  s=0.2,
  L=L,
  e1=eSS,
  e2=eSS) annotation (Placement(transformation(
      extent={{-10,-10},{10,10}},
      rotation=90,
      origin={62,-104})));
Radiation.RadiationParallelPipes RadiationTSRtoTSS[n](
  r1=ODTSR/2,
  r2=ODTSS/2,
  s=0.2,
    L=L,
    e1=eSS,
    e2=eSS) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={60,-62})));
  Radiation.RadiationPipesInsidePipe RadiationHSout[n](
    r1=ODTS/2,
    r2=ODHS/2,
    L=L,
    e=eSS,
    s=0.12) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-158,44})));
  Radiation.RadiationPipesInsidePipe RadiationTSSout[n](
    r1=ODTS/2,
    r2=ODTSS/2,
    L=L,
    e=eSS,
    s=0.11) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-104,44})));
  Radiation.RadiationPipesInsidePipe RadiationVLPout[n](
    r1=ODTS/2,
    r2=ODVLP/2,
    L=L,
    e=eSS,
    s=0.3) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-52,44})));
  Radiation.RadiationPipesInsidePipe RadiationTSRout[n](
    r1=ODTS/2,
    r2=ODTSR/2,
    L=L,
    e=eSS,
    s=0.11) annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-4,42})));
  Modelon.ThermoFluid.Interfaces.FlowHeatPort AmbiencePort[n]
    annotation (Placement(transformation(extent=
    {{-94,106},{-74,126}})));
  WallComponents.DynamicWallSS ExternalEnvelope(
    n=n,
    thk=tEE,
    m=fill(mEE, n),
    Q_vb=0,
    Q_fs=0,
    Q_ss=0,
    A=Modelica.Constants.pi*ODEE*L/n,
    T0(displayUnit="K") = ones(n)*T_startTSR)
    annotation (Placement(transformation(extent=
    {{-56,92},{-36,112}})));
  Radiation.RadiationConcentricPipes RadiationTStoEE[n](
    e2=eSS,
    r1=ODTS/2,
    r2=ODEE/2,
    L=L/n,
    e1=eAl) annotation (Placement(transformation(extent=
    {{6,90},{-14,110}})));
equation
  connect(PortInHS, HeSupply.portA) annotation (Line(
      points={{-238,-26},{-162,-26},{-162,-128},{-149,-128}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(HeSupply.portB, PortOutHS)  annotation (Line(
      points={{-129,-128},{158,-128},{158,-26},{240,-26}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSsupply.portB, PortOutTSS)  annotation (Line(
      points={{-61,-84},{38,-84},{38,-88},{136,-88},
      {136,-84},{240,-84}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(VLPreturn.portA,PortInVLP)  annotation (Line(
      points={{15,-40},{128.5,-40},{128.5,24},{240,24}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSreturn.portA,PortInTSR)  annotation (Line(
      points={{65,6},{136.5,6},{136.5,78},{240,78}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortOutTSR, TSreturn.portB) annotation (Line(
      points={{-238,80},{-74,80},{-74,6},{45,6}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortOutVLP, VLPreturn.portB)  annotation (Line(
      points={{-238,22},{-114,22},{-114,-40},{-5,-40}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(PortInTSS, TSsupply.portA) annotation (Line(
      points={{-236,-84},{-178,-84},{-178,-66},{-130,-66},
      {-130,-84},{-81,
          -84}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort,RadiationTSStoHS. port_a)
  annotation (Line(
      points={{-140,-122.8},{-140,-112},{8,-112}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSStoHS.port_b, TSsupply.PipeHeatPort)
  annotation (Line(
      points={{8,-92},{8,-72},{-72,-72},{-72,-78.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSStoVLP.port_b, VLPreturn.PipeHeatPort)
  annotation (Line(
      points={{-72,-44},{-72,-30},{6,-30},{6,-34.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort,RadiationVLPtoHS. port_a)
  annotation (Line(
      points={{-140,-122.8},{-84,-122.8},{-84,-120},
      {-134,-120},{-134,-104}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoHS.port_a, HeSupply.PipeHeatPort)
  annotation (Line(
      points={{62,-114},{62,-122.8},{-140,-122.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoVLP.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
      points={{6,-6},{6,11.2},{56,11.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort,RadiationTSRtoTSS. port_a)
  annotation (Line(
      points={{-72,-78.8},{-4,-78.8},{-4,-80},{60,-80},{60,-72}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort, RadiationTSStoVLP.port_a)
  annotation (Line(
      points={{-72,-78.8},{-72,-64}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(VLPreturn.PipeHeatPort,RadiationTSRtoVLP. port_a)
  annotation (Line(
      points={{6,-34.8},{6,-26}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationVLPtoHS.port_b, VLPreturn.PipeHeatPort)
  annotation (Line(
      points={{-134,-84},{-134,-68},{42,-68},{42,-34.8},
      {6,-34.8}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(RadiationTSRtoHS.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
      points={{62,-94},{62,-82},{82,-82},{82,11.2},{56,11.2}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationTSRtoTSS.port_b, TSreturn.PipeHeatPort)
  annotation (Line(
        points={{60,-52},{60,-12},{34,-12},{34,11.2},{56,11.2}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(HeSupply.PipeHeatPort, RadiationHSout.port_a)
  annotation (Line(
        points={{-140,-122.8},{-150,-122.8},{-150,-120},
        {-158,-120},{-158,34}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(TSsupply.PipeHeatPort, RadiationTSSout.port_a)
  annotation (Line(
        points={{-72,-78.8},{-88,-78.8},{-88,-76},{-104,-76},
        {-104,34}},color={191,0,0},smooth=Smooth.None));
  connect(VLPreturn.PipeHeatPort, RadiationVLPout.port_a)
  annotation (Line(
        points={{6,-34.8},{-26,-34.8},{-26,-16},{-52,-16},
        {-52,34}},color={191,0,0},smooth=Smooth.None));
  connect(TSreturn.PipeHeatPort, RadiationTSRout.port_a)
  annotation (Line(
        points={{56,11.2},{6,11.2},{6,14},{-4,14},{-4,32}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationTSRout.port_b, ThermalShield.qa)
  annotation (Line(
        points={{-4,52},{56,52},{56,80}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationVLPout.port_b, ThermalShield.qa)
  annotation (Line(
        points={{-52,54},{-52,74},{46,74},{46,80},{56,80}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationTSSout.port_b, ThermalShield.qa)
  annotation (Line(
        points={{-104,54},{-104,84},{24,84},{24,64},{36,64},
        {36,80},{56,80}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationHSout.port_b, ThermalShield.qa)
  annotation (Line(
        points={{-158,54},{-158,70},{30,70},{30,80},{56,80}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(ThermalShield.qb, RadiationTStoEE.port_a)
  annotation (Line(
        points={{56,100},{6,100}},
        color={191,0,0},
        smooth=Smooth.None));
  connect(RadiationTStoEE.port_b, ExternalEnvelope.qb)
  annotation (Line(
      points={{-14,100},{-30,100},{-30,92},{-46,92}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(ExternalEnvelope.qa, AmbiencePort) annotation (Line(
      points={{-46,112},{-66,112},{-66,116},{-84,116}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TSreturn.PipeHeatPort, ThermalShield.qa) annotation
  (Line(points={{56,11.2},{56,42},{74,42},{74,80},{56,80}},
      color={191,0,0},
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-240,
          -180},{240,140}}), graphics), Icon(coordinateSystem(
          preserveAspectRatio=false, extent=
          {{-240,-180},{240,140}}), graphics={
        Rectangle(
          extent={{-168,106},{-74,74}},
          lineColor={165,0,0},
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={168,0,0}),          Rectangle(
          extent={{-218,74},{228,-64}},
          lineColor={0,0,0},
          fillColor={255,255,255},
          fillPattern=FillPattern.Forward),
                                          Text(
          extent={{-192,-54},{214,-136}},
          lineColor={0,0,127},
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={215,215,215},
          textString="%name")}));
end TransferLine;
```

RadiationParallelPipes

```
model RadiationParallelPipes
  extends Modelica.Thermal.HeatTransfer.Components.BodyRadiation(
  Gr=F*A*MLI*e);
  parameter Modelica.SIunits.Length r1 "Radius of emitting pipe";
  parameter Modelica.SIunits.Length r2 "Radius of receiving
  pipe";
  parameter Modelica.SIunits.Length s "Distance between pipes";
  parameter Modelica.SIunits.Length L "Pipe length";
  parameter Real e1 "Emissivity of emitting pipe material";
  parameter Real e2 "Emissivity of receiving pipe material";
  parameter Integer Nmli=10 "Number of MLI layers around pipes";
  final parameter Real e=1/(1/e1+1/e2-1);
  final parameter Real MLI=2*1/(2*(Nmli+1)) "For MLI around both
  pipes";
  final parameter Modelica.SIunits.Area A =
  r1*2*Modelica.Constants.pi*L;
  final parameter Real R = r2/r1;
  final parameter Real S = s/r1;
  final parameter Real C = 1+R+S;
  final parameter Real F = -1/(2*Modelica.Constants.pi)*((C^2-
  (R+1)^2)^(1/2)-(C^2-(R-1)^2)^(1/2)+(R-1)*acos((R-1)/C)-
  (R+1)*acos((R+1)/C));
end RadiationParallelPipes;
```

RadiationConcentricPipes

```
model RadiationConcentricPipes
  extends Modelica.Thermal.HeatTransfer.Components.BodyRadiation(
  Gr=FeA*MLI);
  parameter Modelica.SIunits.Length r1 "Radius of inner
  (receiving) pipe";
  parameter Modelica.SIunits.Length r2 "Radius of outer
  (emitting) pipe";
  parameter Modelica.SIunits.Length L "Pipe length";
  parameter Real e1 "Emissivity of receiving pipe material";
  parameter Real e2 "Emissivity of emitting pipe material";
  parameter Integer Nmli=40
    "Number of MLI layers around pipe or thermal shield";
  final parameter Real MLI=1/(2*(Nmli+1));
  final parameter Real FeA = 2*Modelica.Constants.pi*r2*L/(1/e2+
  (1/e1-1)*(r2/r1));
end RadiationConcentricPipes;
```

RadiationPipesInsidePipe

```
model RadiationPipesInsidePipe
  extends Modelica.Thermal.HeatTransfer.Components.BodyRadiation(
  Gr=F*A*e);
  parameter Modelica.SIunits.Length r1 "Radius of thermal
  shield";
  parameter Modelica.SIunits.Length r2 "Radius of inner pipe";
  parameter Modelica.SIunits.Length s
    "Distance between centers of TS and the inner pipe";
  parameter Modelica.SIunits.Length L "Pipe length";
  parameter Real e "Emissivity of emitting pipe material";
  parameter Integer Nmli = 10 "Layers of MLI around inner pipe";
  final parameter Modelica.SIunits.Area A =
  r1*2*Modelica.Constants.pi*L;
  final parameter Real MLI=1/(2*(Nmli+1));
  final parameter Real R = r1/r2;
  final parameter Real F1 = 1-1/(2*Modelica.Constants.pi)*
  (Modelica.Constants.pi-atan((1+s/r1)/(1-s/r1)*tan(
  Modelica.Constants.pi)));
  final parameter Real F = F1*MLI;
end RadiationPipesInsidePipe;
```

EndBox

```
model EndBox
  replaceable package Medium=
  VaporCycle.Media.Naturals.HydrogenMbwr;
  parameter Modelica.SIunits.MassFlowRate m_flow = 0.3
    "Mass flow rate in the system";
  Modelon.ThermoFluid.Interfaces.VolumePort
      TSreturn(redeclare package Medium = Medium)
    annotation (Placement(transformation(extent=
    {{-110,48},{-90,68}})));
  Modelon.ThermoFluid.Interfaces.VolumePort
      VLPline(redeclare package Medium = Medium)
    annotation (Placement(transformation(extent=
    {{-110,10},{-90,30}})));
  Modelon.ThermoFluid.Interfaces.FlowPort
  HeSupply(redeclare package Medium =
        Medium)
    annotation (Placement(transformation(extent=
    {{-110,-30},{-90,-10}})));
  Modelon.ThermoFluid.Interfaces.FlowPort
  TSsupply(redeclare package Medium =
        Medium)
    annotation (Placement(transformation(extent=
    {{-110,-70},{-90,-50}})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  FrictionHSTransient1(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=m_flow - 0.01)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-44,-24})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  FrictionHSTransient2(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=m_flow - 0.02)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-44,24})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss FrictionTS1(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=m_flow)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={40,-40})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss FrictionTS2(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=m_flow)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={40,-4})));

  Modelon.ThermoFluid.Valves.ValveCompressible ValveTS(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    positiveFlow=true,
    Av=0.1,
    redeclare model FlowCharacteristic =
        Modelon.ThermoFluid.Valves.Characteristics.Quadratic(
          minimumOpening=0.01),
    flowCoeff=Modelon.ThermoFluid.Choices.CvTypes.Kv,
    Kv=66,
    x(fixed=false, start=0.46),
    xs(fixed=false, start=0.46),
    mflow_start=m_flow,
    p(start=150000, fixed=false))
        annotation (Placement(transformation(
        extent={{-10,10},{10,-10}},
        rotation=90,
        origin={40,32})));
  Modelica.Blocks.Sources.RealExpression realExpression(y=1.0)
  annotation (
      Placement(transformation(
        extent={{10,-10},{-10,10}},
        rotation=0,
        origin={82,32})));
  Modelon.ThermoFluid.Valves.ValveCompressible ValveHSTransient(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    positiveFlow=true,
    redeclare model FlowCharacteristic =
        Modelon.ThermoFluid.Valves.Characteristics.Quadratic,
    flowCoeff=Modelon.ThermoFluid.Choices.CvTypes.Kv,
    Kv=66,
    x(fixed=false, start=0.46),
    xs(fixed=false, start=0.46),
    p(start=150000, fixed=false),
    mflow_start=m_flow - 0.01)
annotation (
      Placement(transformation(
        extent={{-10,10},{10,-10}},
        rotation=90,
        origin={-44,0})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  FrictionHSHeater1(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=0.01)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={8,-24})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  FrictionHSHeater2(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        QuadraticOperatingPointLoss,
    mflow_start=0.01)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={8,24})));

  Modelon.ThermoFluid.Valves.ValveCompressible ValveHSHeater(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    positiveFlow=true,
    Av=0.1,
    redeclare model FlowCharacteristic =
        Modelon.ThermoFluid.Valves.Characteristics.Quadratic,
    flowCoeff=Modelon.ThermoFluid.Choices.CvTypes.Kv,
    Kv=66,
    x(fixed=false, start=0.46),
    xs(fixed=false, start=0.46),
    p(start=150000, fixed=false),
    mflow_start=0.01) annotation (
      Placement(transformation(
        extent={{-10,10},{10,-10}},
        rotation=90,
        origin={8,0})));
  Split split(
    redeclare package Medium = Medium,
    redeclare model FrictionB =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        LinearOperatingPointLoss,
    redeclare model FrictionC =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        LinearOperatingPointLoss)
    annotation (Placement(transformation(extent=
    {{-54,-64},{-34,-84}})));

  Modelon.ThermoFluid.SplitsAndJoins.Join
       join(
    redeclare package Medium = Medium,
    T_start(displayUnit="K"))
    annotation (Placement(transformation(extent=
    {{-54,58},{-34,38}})));
  Modelon.ThermoFluid.FlowModifiers.SetFlowRate
  setFlowRate(redeclare package
      Medium = Medium,
      flowDefinition=Modelon.ThermoFluid.Choices.FlowDefinition.
      m_flow,m_flow=0.29)
    annotation (Placement(transformation(extent=
    {{-10,-10},{10,10}},
        rotation=90,
        origin={-44,-48})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  FrictionHSTransient3(
    redeclare package Medium = Medium,
    T_start(displayUnit="K") = 300,
    mflow_start=m_flow,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        LinearOperatingPointLoss
        (dp0(displayUnit="Pa"), m_flow0=m_flow))
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={-74,34})));
equation
  connect(TSsupply, FrictionTS1.portA) annotation (Line(
      points={{-100,-60},{40,-60},{40,-50}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(ValveTS.opening, realExpression.y)
  annotation (Line(
      points={{49,32},{71,32}},
      color={0,0,127},
      smooth=Smooth.None));
  connect(ValveTS.portB, TSreturn) annotation (Line(
      points={{40,42},{40,58},{-100,58}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(ValveTS.portA, FrictionTS2.portB)
  annotation (Line(
      points={{40,22},{40,6}},
      color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(FrictionTS1.portB, FrictionTS2.portA)
  annotation (Line(
        points={{40,-30},{40,-14}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(FrictionHSTransient1.portB, ValveHSTransient.portA)
  annotation (
        Line(
        points={{-44,-14},{-44,-10}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(FrictionHSTransient2.portA, ValveHSTransient.portB)
  annotation (
        Line(
        points={{-44,14},{-44,10}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(FrictionHSHeater1.portB, ValveHSHeater.portA)
  annotation (Line(
        points={{8,-14},{8,-10}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(ValveHSHeater.portB, FrictionHSHeater2.portA)
  annotation (Line(
        points={{8,10},{8,14}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(ValveHSTransient.opening, realExpression.y)
  annotation (Line(
        points={{-35,-4.44089e-016},{-6,-4.44089e-016},
        {-6,70},{58,70},{58,32},{71,
            32}},
        color={0,0,127},
        smooth=Smooth.None));
  connect(FrictionHSTransient2.portB, join.portB)
  annotation (Line(
        points={{-44,34},{-44,38}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(join.portA, FrictionHSHeater2.portB)
  annotation (Line(
        points={{-34,48},{8,48},{8,34}},
        color={255,128,0},
        pattern=LinePattern.None,
        smooth=Smooth.None));
  connect(split.portA, HeSupply) annotation (Line(
      points={{-54,-74},{-74,-74},{-74,-20},{-100,-20}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(ValveHSHeater.opening, realExpression.y)
  annotation (Line(
      points={{17,-4.44089e-016},{28,-4.44089e-016},
      {28,14},{58,14},{58,32},{71,
          32}},
      color={0,0,127},
      smooth=Smooth.None));
  connect(join.portC, FrictionHSTransient3.portB)
  annotation (Line(
      points={{-54,48},{-74,48},{-74,44}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(FrictionHSTransient3.portA, VLPline)
  annotation (Line(
      points={{-74,24},{-74,20},{-100,20}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(split.portC, setFlowRate.portA)
  annotation (Line(
      points={{-44,-64},{-44,-58}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(setFlowRate.portB, FrictionHSTransient1.portA)
  annotation (Line(
      points={{-44,-38},{-44,-34}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(split.portB, FrictionHSHeater1.portA)
  annotation (Line(
      points={{-34,-74},{8,-74},{8,-34}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(
  preserveAspectRatio=false, extent={{-100,
            -100},{100,100}}), graphics), Icon(graphics={
          Rectangle(
            extent={{-82,100},{80,-74}},
            lineColor={255,255,255},
            fillColor={255,255,255},
            fillPattern=FillPattern.VerticalCylinder),
                                Text(
        extent={{-164,-122},{158,-64}},
        textString="%name",
        lineColor={0,0,255}),
        Polygon(
          points={{54,76},{54,-66},{174,-36},{174,44},{54,76}},
          lineColor={0,0,127},
          smooth=Smooth.None,
          fillColor={66,132,197},
          fillPattern=FillPattern.Solid)}));
end EndBox;
```

SimpleEndBox

```
model SimpleEndBox
  parameter Modelica.SIunits.Length ODHS "He supply diameter";
  parameter Modelica.SIunits.Length ODTS "TS diameter";
  parameter Modelica.SIunits.Length thkHS "Thickness of HS
  wall";
  parameter Modelica.SIunits.Length thkTS "Thickness of TS
  walls";
  parameter Modelica.SIunits.Length L=3 "Lengths of bend
  sections";
  parameter Modelica.SIunits.Temp_K T_i = 300
    "Initial temperature of components";
  parameter Real zeta_90bend=0.5 "90 degree bend friction loss
  coefficient";

  replaceable package Medium =
  VaporCycle.Media.Naturals.HydrogenMbwr;

  Modelon.ThermoFluid.Interfaces.FlowPort HeSupply(
  redeclare package Medium =
        Medium)
    annotation (Placement(transformation(extent=
    {{-110,-52},{-90,-32}})));
  Modelon.ThermoFluid.Interfaces.FlowPort VLPline(
  redeclare package Medium = Medium)
    annotation (Placement(transformation(extent=
    {{-110,34},{-90,54}})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  frictionLossHS(
    redeclare package Medium = Medium,
    positiveFlow=true,
    T_start=T_i,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        VariableZeta (
          zeta=zeta_90bend*L/ODHS,
          A=Modelica.Constants.pi*ODHS^2/4))
          annotation (Placement(transformation(
        extent={{-10,-11},{10,11}},
        rotation=90,
        origin={-33,-12})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  frictionLossVLP(redeclare
      package Medium = Medium,
    positiveFlow=true,
    T_start=T_i,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        VariableZeta (
          zeta=zeta_90bend*L/ODHS,
          A=Modelica.Constants.pi*ODHS^2/4)) annotation(
      Placement(transformation(
        extent={{-10,-11},{10,11}},
        rotation=90,
        origin={-33,20})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  frictionLossTS1(redeclare
      package Medium = Medium,
    positiveFlow=true,
    T_start=T_i,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        VariableZeta (
          zeta=zeta_90bend*L/ODTS,
          A=Modelica.Constants.pi*ODTS^2/4)) annotation(
      Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={32,-18})));

  Modelon.ThermoFluid.FlowResistances.FrictionLoss
  frictionLossTS2(redeclare
      package Medium = Medium,
    positiveFlow=true,
    T_start=T_i,
    redeclare model Friction =
        Modelon.ThermoFluid.FlowResistances.FrictionModels.
        VariableZeta (
          zeta=zeta_90bend*L/ODTS,
          A=Modelica.Constants.pi*ODTS^2/4)) annotation(
      Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={32,28})));

  Modelon.ThermoFluid.Interfaces.FlowPort TSsupply(
  redeclare package Medium =
        Medium)
    annotation (Placement(transformation(extent=
    {{-110,-98},{-90,-78}})));
  Modelon.ThermoFluid.Interfaces.FlowPort TSreturn(
  redeclare package Medium =
        Medium)
    annotation (Placement(transformation(extent=
    {{-110,80},{-90,100}})));
equation
```

```
  connect(HeSupply, frictionLossHS.portA)
      annotation (Line(
      points={{-100,-42},{-33,-42},{-33,-22}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSsupply, frictionLossTS1.portA)
      annotation (Line(
      points={{-100,-88},{32,-88},{32,-28}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(frictionLossTS1.portB, frictionLossTS2.portA)
      annotation (Line(
      points={{32,-8},{32,18}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(frictionLossVLP.portB, VLPline)
      annotation (Line(
      points={{-33,30},{-32,30},{-32,44},{-100,44}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(frictionLossHS.portB, frictionLossVLP.portA)
      annotation (Line(
      points={{-33,-2},{-33,10}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(frictionLossTS2.portB, TSreturn)
      annotation (Line(
      points={{32,38},{32,90},{-100,90}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,
            -100},{100,100}}), graphics), Icon(coordinateSystem(
          preserveAspectRatio=false, extent=
          {{-100,-100},{100,100}}),graphics={Text(
          extent={{-90,-58},{102,-124}},
          lineColor={0,0,127},
          lineThickness=0.5,
          fillColor={0,0,0},
          fillPattern=FillPattern.Solid,
          textString="%name"),
          Rectangle(
            extent={{-86,100},{76,-74}},
            lineColor={255,255,255},
            fillColor={255,255,255},
            fillPattern=FillPattern.VerticalCylinder),
        Polygon(
          points={{50,76},{50,-66},{170,-36},{170,44},{50,76}},
          lineColor={0,0,127},
          smooth=Smooth.None,
          fillColor={66,132,197},
          fillPattern=FillPattern.Solid)}));
end SimpleEndBox;
```

Cryoplant

```
model Cryoplant
   replaceable package
   Medium=VaporCycle.Media.Naturals.HydrogenMbwr;

   parameter Modelica.SIunits.Temp_K T_inHS = 5.2
   "Inflow temperature";
   parameter Modelica.SIunits.Temp_K T_inTS = 40
   "Inflow temperature";
   parameter Modelica.SIunits.MassFlowRate m_flow_He = 0.0993
    "Mass flow rate for cold helium circuit";
   parameter Modelica.SIunits.MassFlowRate m_flow_TS = 0.0775
    "Mass flow rate for thermal shield circuit";
   parameter Modelica.SIunits.Pressure P_outVLP = 27e2
   "Outflow pressure";
   parameter Modelica.SIunits.Pressure P_outTS = 19e5
   "Outflow pressure";

  Modelon.ThermoFluid.Sources.PressureBoundary OutflowVLP(
    T(displayUnit="K") = 100,
    redeclare package Medium = Medium,
    N=1,
    p=P_outVLP) annotation (Placement(transformation(extent=
    {{0,10},{20,30}})));
  Modelon.ThermoFluid.Sources.PressureBoundary OutflowTS(
    N=1,
    T(displayUnit="K") = 100,
    redeclare package Medium = Medium,
    p=P_outTS) annotation (Placement(transformation(extent=
    {{0,50},{20,70}})));
  Modelon.ThermoFluid.Sources.MassFlowBoundary MassFlowHeCircuit(
    m_flow=m_flow_He,
    T(displayUnit="K") = T_inHS,
    redeclare package Medium = Medium,
    flowDefinition=
    Modelon.ThermoFluid.Choices.FlowDefinition.m_flow)
    annotation (Placement(transformation(extent=
    {{0,-30},{20,-10}})));
  Modelon.ThermoFluid.Sources.MassFlowBoundary
  MassFlowTScircuit(m_flow=m_flow_TS, T(
        displayUnit="K") = T_inTS,
    redeclare package Medium = Medium,
    flowDefinition=
    Modelon.ThermoFluid.Choices.FlowDefinition.m_flow)
    annotation (Placement(transformation(extent=
    {{0,-70},{20,-50}})));
  Modelon.ThermoFluid.Interfaces.FlowPort
  PortInTSR(redeclare package Medium = Medium
  annotation (Placement(
        transformation(extent={{90,50},{110,70}}),
        iconTransformation(extent={{92,70},
            {112,90}})));
  Modelon.ThermoFluid.Interfaces.FlowPort PortInVLP(
  redeclare package Medium =
        Medium) annotation (Placement(
        transformation(extent={{90,10},{110,30}}),
        iconTransformation(extent={{92,16},
            {112,36}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutHS(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{90,-30},{110,-10}}),
        iconTransformation(extent={{92,-34},
            {112,-14}})));
  Modelon.ThermoFluid.Interfaces.VolumePort PortOutTSS(
  redeclare package Medium
      = Medium) annotation (Placement(
        transformation(extent={{90,-70},{110,-50}}),
        iconTransformation(extent={{92,-92},
            {112,-72}})));
equation
  connect(OutflowTS.fluidPort[1], PortInTSR) annotation (Line(
      points={{19,60},{100,60}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(OutflowVLP.fluidPort[1], PortInVLP) annotation (Line(
      points={{19,20},{100,20}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(MassFlowHeCircuit.fluidPort, PortOutHS) annotation
  (Line(
      points={{19,-20},{100,-20}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(MassFlowTScircuit.fluidPort, PortOutTSS) annotation
  (Line(
      points={{19,-60},{100,-60}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,-100},{100,100}}), graphics),
  Icon(coordinateSystem(
          preserveAspectRatio=false, extent=
          {{-100,-100},{100,100}}),
        graphics={
```

```
        Ellipse(
          extent={{-100,-56},{100,-98}},
          lineColor={0,0,0},
          fillPattern=FillPattern.VerticalCylinder,
          fillColor={215,215,215}),
        Rectangle(
          extent={{-100,44},{100,-76}},
          lineColor={0,0,0},
          fillPattern=FillPattern.VerticalCylinder,
          fillColor={215,215,215}),
        Ellipse(
          extent={{-100,84},{100,8}},
          lineColor={0,0,0},
          fillColor={255,255,255},
          fillPattern=FillPattern.Solid),
        Ellipse(
          extent={{-60,68},{58,-44}},
          lineColor={0,0,0},
          fillPattern=FillPattern.Sphere,
          fillColor={215,215,215}),
        Ellipse(
          extent={{8,22},{-10,4}},
          lineColor={156,0,0},
          fillColor={156,0,0},
          fillPattern=FillPattern.Solid,
          visible=internalHeatResistance),
          Text(
          extent={{-236,-90},{240,-154}},
          lineColor={0,0,127},
          lineThickness=0.5,
          fillColor={0,0,0},
          fillPattern=FillPattern.Solid,
          textString="%name")}));
end Cryoplant;
```

Cryomodule

```
model Cryomodule
  replaceable package Medium=
  DistributionSystem.Media.Helium_RefProp;
  parameter Integer n=2;
  parameter Modelica.SIunits.Temp_K T0=300;
  parameter Modelica.SIunits.Temp_K T0_TS=300;
  parameter Modelica.SIunits.Pressure p_start_inHe=1e5;
  parameter Modelica.SIunits.Pressure p_start_outHe=1e5;
  parameter Modelica.SIunits.Pressure p_start_inTS=1e5;
  parameter Modelica.SIunits.Pressure p_start_outTS=1e5;
  parameter Modelica.SIunits.Length L=8 "Length of the
  cryomodule";
  parameter Modelica.SIunits.Length OD=0.01715
    "Diameter of the helium supply pipe";
  parameter Real zeta_bend=0.5
```

```
    "Friction loss coefficient for the pipe bends in the module";
  parameter Modelica.SIunits.Length D_TS = 0.5;
  parameter Modelica.SIunits.Length D_tank = 0.3;
  parameter Modelica.SIunits.Mass m_Ti = 134;
  parameter Modelica.SIunits.Mass m_Nb = 220;
  parameter Modelica.SIunits.Mass m_Al = 120;
  parameter Modelica.SIunits.Mass m_SS = 46;
  parameter Modelica.SIunits.Mass m_SS_TS = 0;

  final parameter Real eSS = 0.07;
  final parameter Real eTi = 0.19;
  final parameter Real eAl = 0.04;

  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseFlowPort HeInflowPort(redeclare package Medium
      =Medium) annotation (Placement(transformation(extent=
      {{-110,-30},{-90,-10}}),
                          iconTransformation(extent={{-106,-30},
                          {-86,-10}})));
  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseVolumePort VLPport(redeclare package Medium =
                 Medium) annotation (Placement(transformation(
                 extent={{-110,10},{-90,30}}),
                 iconTransformation(extent={{-106,8},{-86,28}})));
  CryomoduleComponents.CMSteelPipingTwoPhase
                                           cMSteelPiping(
    thkSS=0.003,
    m_SS=m_SS,
    T0=T0,
    L_CMpipe=L,
    D_CMpipe=OD,
    redeclare package Medium = Medium,
    p_start_in=p_start_inHe,
    p_start_out=p_start_outHe,
    n=n)
    annotation (Placement(transformation(extent={{-42,-8},
    {-22,12}})));
  CryomoduleComponents.CMThermalShields cMThermalShields(
    thkAl=0.003,
    m_Al=m_Al,
    T0=T0_TS,
    L_TS=L)
           annotation (Placement(transformation(extent={{44,-10},
           {64,10}})));
  CryomoduleComponents.NiobiumCavity niobiumCavity(
    thkNb=0.002,
    m_Nb=m_Nb,
    T0=T0) annotation (Placement(transformation(extent={{-6,38},
    {14,58}})));
  CryomoduleComponents.TitaniumTankPipe
                                          titaniumTank(
    thkTi=0.003,
    m_Ti=m_Ti,
    T0=T0,
    redeclare package Medium = Medium)
           annotation (Placement(transformation(extent={{-74,4},
           {-54,24}})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss frictionLoss(
  redeclare package Medium = Medium,
    positiveFlow=true,
    redeclare model Friction =
    Modelon.ThermoFluid.FlowResistances.FrictionModels.
    VariableZeta
        (A=Modelica.Constants.pi*L*OD/n, zeta=zeta_bend*L/n/OD),
    T_start(displayUnit="K") = T0_TS)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={66,-46})));
  Modelon.ThermoFluid.FlowResistances.FrictionLoss frictionLoss1(
  redeclare package Medium = Medium,
    positiveFlow=true,
    redeclare model Friction =
    Modelon.ThermoFluid.FlowResistances.FrictionModels.
    VariableZeta
        (A=
            Modelica.Constants.pi*L*OD/n, zeta=zeta_bend*L/n/OD),
    T_start(displayUnit="K") = T0_TS)
    annotation (Placement(transformation(
        extent={{-10,-10},{10,10}},
        rotation=90,
        origin={66,42})));
  Modelon.ThermoFluid.Interfaces.FlowPort HeInflowPort1(
      redeclare package Medium=Medium)
      annotation (Placement(transformation(extent={{-110,
            -70},{-90,-50}}),
                       iconTransformation(extent=
                       {{-106,-68},{-86,-48}})));
  Modelon.ThermoFluid.Interfaces.VolumePort VLPport1(
               redeclare package Medium =
               Medium) annotation (Placement(transformation(
               extent={{-110,50},{-90,70}}),
            iconTransformation(extent={{-106,46},
            {-86,66}})));
  Radiation.RadiationPipesInsidePipe radiationPipesInsidePipe(
    s=0.15,
    L=L,
    Nmli=1,
    e=eAl,
    r1=D_TS/2,
    r2=OD/2)
            annotation (Placement(transformation(extent={{-6,-8},
            {10,8}})));
  Radiation.RadiationParallelPipes radiationParallelPipes(
    s=0.1,
    L=8,
    e1=eSS,
    Nmli=1,
    r1=OD/2,
    r2=D_tank/2,
    e2=eTi)
    annotation (Placement(transformation(extent={{-46,28},
    {-30,44}})));
  Radiation.RadiationPipesInsidePipe radiationPipesInsidePipe1(
    s=0.15,
    L=L,
    Nmli=1,
    r1=D_TS/2,
    r2=D_tank/2,
    e=eTi)  annotation (Placement(transformation(extent={{-6,18},
    {10,34}})));
  CryomoduleComponents.CMSteelPiping TSpiping(
    thkSS=0.003,
    m_SS=m_SS_TS,
    L_CMpipe=L,
    redeclare package Medium = Medium,
    D_CMpipe=D_TS,
    T0=T0_TS,
    n=n,
    p_start_in=p_start_inTS,
    p_start_out=p_start_outTS)
    annotation (Placement(transformation(extent={{66,-8},
    {86,12}})));
equation
  connect(HeInflowPort1, frictionLoss.portA) annotation
  (Line(points={{-100,-60},{66,-60},{66,-56}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(VLPport1, frictionLoss1.portB) annotation (Line(
      points={{-100,60},{66,60},{66,52}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(titaniumTank.twoPhaseFlowPort, VLPport) annotation
  (Line(
      points={{-73.4,14},{-86,14},{-86,20},{-100,20}},
      color={0,190,0},
      smooth=Smooth.None));
  connect(titaniumTank.twoPhaseFlowPort1,
  cMSteelPiping.volumePort) annotation (Line(
      points={{-54.6,14},{-46,14},{-46,4},{-42,4},{-42,3.2}},
      color={0,190,0},
      smooth=Smooth.None));
  connect(cMThermalShields.ThermalShieldPort,
  radiationPipesInsidePipe.port_b)
    annotation (Line(
      points={{44,0},{10,0}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(radiationPipesInsidePipe.port_a,
  cMSteelPiping.CMFlowPort)
```

```
    annotation (Line(
      points={{-6,0},{-16,0},{-16,12},{-32,12},{-32,6.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(titaniumTank.TankHeliumPort,
  niobiumCavity.NiobiumCavityPort)
    annotation (Line(
      points={{-68.8,20.4},{-68.8,48},{-6,48}},
      color={191,0,0},
      thickness=0.5,
      smooth=Smooth.None));
  connect(titaniumTank.TitaniumTankPort,
  radiationParallelPipes.port_a)
    annotation (Line(
      points={{-64,20.4},{-64,36},{-46,36}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(radiationParallelPipes.port_b,
  cMSteelPiping.CMFlowPort) annotation (
      Line(
      points={{-30,36},{-22,36},{-22,12},{-32,12},{-32,6.2}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(radiationPipesInsidePipe1.port_b,
  radiationPipesInsidePipe.port_b)
    annotation (Line(
      points={{10,26},{16,26},{16,0},{10,0}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(radiationPipesInsidePipe1.port_a,
  radiationParallelPipes.port_a)
    annotation (Line(
      points={{-6,26},{-64,26},{-64,36},{-46,36}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(frictionLoss.portB, TSpiping.flowPort)
  annotation (Line(
      points={{66,-36},{66,-0.4}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSpiping.volumePort, frictionLoss1.portA)
  annotation (Line(
      points={{66,3.2},{66,32}},
      color={255,128,0},
      pattern=LinePattern.None,
      smooth=Smooth.None));
  connect(TSpiping.CMFlowPort,
  cMThermalShields.ThermalShieldPort)
    annotation (Line(
      points={{76,6.2},{76,20},{44,20},{44,0}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(HeInflowPort, cMSteelPiping.flowPort)
  annotation (Line(
      points={{-100,-20},{-72,-20},{-72,-0.4},{-42,-0.4}},
      color={0,190,0},
      smooth=Smooth.None));
  annotation (Icon(coordinateSystem(
  preserveAspectRatio=false, extent={{
            -100,-100},{100,100}}),
                   graphics={
      Polygon(
        points={{-94,-60},{-74,-40},{106,-40},{86,-60},
        {-94,-60}},
        lineColor={0,0,255},
        fillColor={235,235,235},
        fillPattern=FillPattern.Solid),
      Text(
        extent={{-116,101},{124,46}},
        lineColor={255,0,0},
        textString="%name"),
        Ellipse(
          extent={{-42,50},{-80,-48}},
          lineColor={0,0,0},
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={215,215,215}),             Rectangle(
          extent={{-60,50},{80,-48}},
          pattern=LinePattern.None,
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder,
          lineColor={0,0,0}),
        Ellipse(
          extent={{96,50},{64,-48}},
          lineColor={95,95,95},
          fillPattern=FillPattern.Solid,
          fillColor={255,255,255}),
        Rectangle(
          extent={{-28,4},{16,-12}},
          lineColor={165,0,0},
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={168,0,0})}), Diagram(coordinateSystem(
          preserveAspectRatio=false,
          extent={{-100,-100},{100,100}}), graphics));
end Cryomodule;
```

TitaniumTankPipe

```
model TitaniumTankVolume
  replaceable package Medium=
  DistributionSystem.Media.Helium_RefProp;
  parameter Integer n=1;
  parameter Modelica.SIunits.Volume V=0.5;
  parameter Modelica.SIunits.Length thkTi "Thickness of titanium
  tank";
  parameter Modelica.SIunits.Mass m_Ti "Titanium mass";
  parameter Modelica.SIunits.Temp_K T0 "Initial temperature";
  parameter Modelica.SIunits.Length D_Ti=0.4 "Tank diameter";
  parameter Modelica.SIunits.Length L_Ti=6 "Tank length";
  WallComponents.DynamicWallTi dynamicWallTi(
    thk=thkTi,
    A=Modelica.Constants.pi*D_Ti*L_Ti,
    m=fill(m_Ti/n, n),
    T0=ones(n)*T0,
    n=1)        annotation (Placement(transformation(extent=
    {{-10,30},{10,50}})));
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a
  TitaniumTankPort
  annotation (Placement(transformation(extent=
  {{-10,54},{10,74}}),
  iconTransformation(extent={{-10,54},{10,74}})));
  Modelon.ThermoFluid.Volumes.TwoPhaseVolume volume(
    redeclare package Medium = Medium,
    V=V,
    initOpt=Modelon.ThermoFluid.Choices.InitOptions.noInit)
    annotation (Placement(transformation(extent=
    {{-10,-10},{10,10}})));
  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseFlowPort
    twoPhaseFlowPort(redeclare package Medium = Medium)
    annotation (Placement(transformation(extent=
    {{-104,-10},{-84,10}}),
  iconTransformation(extent={{-104,-10},{-84,10}})));
  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseFlowPort
    twoPhaseFlowPort1(redeclare package Medium = Medium)
    annotation (Placement(transformation(extent=
    {{84,-10},{104,10}}),
        iconTransformation(extent={{84,-10},{104,10}})));
  Modelon.ThermoFluid.Interfaces.FlowHeatPort TankHeliumPort
  annotation (
      Placement(transformation(extent={{-58,54},{-38,74}}),
        iconTransformation(extent={{-58,54},{-38,74}})));
equation
  connect(twoPhaseFlowPort, volume.portA) annotation (Line(
      points={{-94,0},{-10,0}},
      color={0,190,0},
      smooth=Smooth.None));
  connect(volume.portB, twoPhaseFlowPort1) annotation (Line(
      points={{10,0},{94,0}},
      color={0,190,0},
      smooth=Smooth.None));
  connect(volume.q, dynamicWallTi.qb[1]) annotation (Line(
      points={{0,0},{0,30}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(dynamicWallTi.qa[1], TitaniumTankPort)
  annotation (Line(points={{0,50},{0,64}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(TankHeliumPort, volume.q) annotation (Line(
      points={{-48,64},{-24,64},{-24,14},{0,14},{0,0}},
      color={191,0,0},
      thickness=0.5,
      smooth=Smooth.None));
  connect(TankHeliumPort, TankHeliumPort) annotation (Line(
      points={{-48,64},{-48,64}},
      color={191,0,0},
      thickness=0.5,
      smooth=Smooth.None));
  annotation (Diagram(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,
            -100},{100,100}}), graphics), Icon(coordinateSystem(
          preserveAspectRatio=false, extent=
          {{-100,-100},{100,100}}),graphics={Rectangle(
          extent={{-90,40},{90,-42}},
          pattern=LinePattern.None,
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder,
          lineColor={0,0,0}),Text(extent={{90,-50},{-70,-90}},
            textString="%name"),
        Line(
          points={{-48,12},{-48,54}},
          color={255,0,0},
          thickness=1,
          smooth=Smooth.None),
        Line(
          points={{-46,12},{-46,54},{-46,54},{-46,14},{-46,12},
          {-48,12},{-46,12},{-48,54},{-46,36},{-48,58},
          {-46,52},{-46,48},{-48,28},{-48,14},{-46,20},
          {-48,24},{-48,20},{-46,12},{-48,12}},
          color={255,0,0},
          thickness=1,
          smooth=Smooth.None),
        Line(
          points={{-48,12},{-46,56},{-48,48},{-46,54}},
          color={255,0,0},
          thickness=1,
          smooth=Smooth.None),
        Line(
          points={{-46,12},{-48,12}},
          color={255,0,0},
          thickness=1,
          smooth=Smooth.None),
        Rectangle(
          extent={{-46,12},{-46,54}},
          lineColor={255,0,0},
          lineThickness=1,
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={215,215,215}),
        Rectangle(
          extent={{-52,54},{-44,10}},
          lineColor={255,0,0},
          lineThickness=1,
          fillPattern=FillPattern.HorizontalCylinder,
          fillColor={255,0,0})}));
end TitaniumTankVolume;
```

CMSteelPipingTwoPhase

```
model CMSteelPipingTwoPhase
  replaceable package
  Medium=VaporCycle.Media.Naturals.HydrogenMbwr;
  parameter Modelica.SIunits.Length thkSS "Thickness of steel
  pipe";
  parameter Modelica.SIunits.Mass m_SS "Stainless steel mass";
  parameter Modelica.SIunits.Temp_K T0 "Initial temperature";
  parameter Modelica.SIunits.Length L_CMpipe
    "Length of pipes inside cryomodule";
  parameter Modelica.SIunits.Length D_CMpipe
    "Inner diameter of pipes inside cryomodule";
  parameter Integer n=1 "Mass sections";
  parameter Modelica.SIunits.Pressure p_start_in=p_start_in;
  parameter Modelica.SIunits.Pressure p_start_out=p_start_out;
  parameter Real zeta=0.03 "Friction loss coefficient";
  parameter Modelica.SIunits.CoefficientOfHeatTransfer
  alpha0 = 300;
  VaporCycle.Pipes.Pipe distributedPipe(
    L=L_CMpipe,
    D=D_CMpipe,
    n=n,
    redeclare package Medium = Medium,
    p_start(displayUnit="Pa"),
    T_start(displayUnit="K"),
    p_in_start=p_start_in,
    p_out_start=p_start_out,
    T_in_start=T0,
    T_out_start=T0,
    includeStaticHead=true,
    redeclare model Friction =
        VaporCycle.Pipes.SubComponents.FlowResistance.
        OnePhaseColebrook,
    redeclare model HeatTransfer =
        Modelon.ThermoFluid.FlowChannels.HeatTransfer.
        DiscretizedHeatTransfer.ConstantCoefficient
        (alpha0=alpha0))
    annotation (Placement(transformation(extent=
    {{-10,-34},{10,-14}})));

  WallComponents.DynamicWallSS dynamicWallSS(n=n,
    m=fill(m_SS/n,n),
    T0=ones(n)*T0,
    thk=thkSS,
    A=Modelica.Constants.pi*D_CMpipe*L_CMpipe/n,
    Q_vb=0,
    Q_fs=0,
    Q_ss=0)
    annotation (Placement(transformation(extent=
    {{-10,6},{10,26}})));
  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseFlowPort
  flowPort(redeclare package Medium = Medium)
  annotation (Placement(
        transformation(extent={{-108,-32},{-92,-16}}),
        iconTransformation(
          extent={{-110,-34},{-90,-14}})));
  Modelon.ThermoFluid.Interfaces.ApplicationSpecific.
  TwoPhaseVolumePort
      volumePort(redeclare package Medium = Medium)
      annotation (Placement(
        transformation(extent={{-108,2},{-92,18}}),
        iconTransformation(extent={{
            -110,2},{-90,22}})));
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a CMFlowPort
    annotation (Placement(transformation(extent=
    {{-10,32},{10,52}}),
        iconTransformation(extent={{-10,32},{10,52}})));
equation
  connect(distributedPipe.q, dynamicWallSS.qb) annotation (Line(
      points={{0,-19},{0,6}},
      color={191,0,0},
      thickness=0.5,
      smooth=Smooth.None));
  connect(dynamicWallSS.qa[1], CMFlowPort) annotation (Line(
      points={{0,26},{0,42}},
      color={191,0,0},
      smooth=Smooth.None));
  connect(flowPort, distributedPipe.portA) annotation (Line(
      points={{-100,-24},{-10,-24}},
      color={0,190,0},
      smooth=Smooth.None));
  connect(distributedPipe.portB, volumePort) annotation (Line(
      points={{10,-24},{32,-24},{32,-2},{-64,-2},{-64,10},
      {-100,10}},
      color={0,190,0},
      smooth=Smooth.None));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,
          -100},{100,100}}),
                          graphics={
        Rectangle(
          extent={{-100,-12},{100,-36}},
          lineColor={0,0,0},
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder),
          Text(extent={{84,-40},{-76,-80}},
            textString="%name"),
        Rectangle(
          extent={{-100,24},{100,0}},
          lineColor={0,0,0},
          fillColor={215,215,215},
          fillPattern=FillPattern.HorizontalCylinder)}),
          Diagram(
        coordinateSystem(preserveAspectRatio=false,
        extent={{-100,-100},{100,100}}),
        graphics));
end CMSteelPipingTwoPhase;
```

CMThermalShields

```
model CMThermalShields
  parameter Modelica.SIunits.Length thkAl "Thickness of aluminum
  shield";
  parameter Modelica.SIunits.Mass m_Al "Aluminum mass";
  parameter Modelica.SIunits.Temp_K T0 "Initial temperature";
  parameter Integer nn=1 "Mass sections";
  parameter Modelica.SIunits.Length D_TS=0.5 "Thermal shield
  diameter";
  parameter Modelica.SIunits.Length L_TS=6 "Thermal shield total
  length";
  WallComponents.DynamicWallAl dynamicWallAl(
    n=nn,
    T0=ones(nn)*T0,
    thk=thkAl,
    m=fill(m_Al/nn, nn),
    A=Modelica.Constants.pi*D_TS*L_TS)
    annotation (Placement(transformation(extent=
    {{-10,12},{10,32}})));
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a
  ThermalShieldPort
    annotation (Placement(transformation(extent=
    {{-110,-10},{-90,10}})));
equation
  connect(ThermalShieldPort, dynamicWallAl.qb[1])
  annotation (Line(
      points={{-100,0},{0,0},{0,12}},
      color={191,0,0},
      smooth=Smooth.None));
  annotation (Icon(graphics={
      Polygon(
        points={{-64,-76},{-64,74},{-4,56},{-4,-92},{-64,-76}},
        lineColor={0,0,255},
        fillColor={235,235,235},
        fillPattern=FillPattern.Solid),
      Polygon(
        points={{-36,-74},{-36,76},{24,58},{24,-90},{-36,-74}},
        lineColor={0,0,255},
        fillColor={235,235,235},
        fillPattern=FillPattern.Solid),
        Text(extent={{68,-82},{-92,-122}},
            textString="%name")}), Diagram(coordinateSystem(
            preserveAspectRatio=
           false, extent={{-100,-100},{100,100}}), graphics));
end CMThermalShields;
```

NiobiumCavity

```
model NiobiumCavity
  parameter Modelica.SIunits.Length thkNb "Thickness of niobium
  cavity";
  parameter Modelica.SIunits.Mass m_Nb "Niobium mass";
  parameter Modelica.SIunits.Temp_K T0 "Initial temperature";
  parameter Integer nn=1 "Mass sections";
  parameter Modelica.SIunits.Length D_Nb=0.3 "Cavity diameter";
  parameter Modelica.SIunits.Length L_Nb=6 "Cavities' total
  length";
  Modelica.Thermal.HeatTransfer.Interfaces.HeatPort_a
  NiobiumCavityPort
    annotation (Placement(transformation(extent=
    {{-110,-10},{-90,10}})));
  WallComponents.DynamicWallNb dynamicWallNb(
    n=nn,
    T0=ones(nn)*T0,
    thk=thkNb,
    m=fill(m_Nb/nn, nn),
    A=Modelica.Constants.pi*D_Nb*L_Nb)
                annotation (Placement(transformation(extent=
                {{-10,10},{10,30}})));
equation
  connect(NiobiumCavityPort, dynamicWallNb.qb[1]) annotation
  (Line(points={{-100,0},{0,0},{0,10}},
      color={191,0,0},
      smooth=Smooth.None));
  annotation (Icon(coordinateSystem(preserveAspectRatio=false,
  extent={{-100,-100},{100,100}}), graphics={Bitmap(
          extent={{-120,90},{190,-70}},
          imageSource="",
          fileName="modelica://DistributionSystem/../../../
          DESY_Picture.jpg"),
            Text(extent={{78,-46},{-82,-86}},
            textString="%name")}), Diagram(coordinateSystem(
            preserveAspectRatio=
           false, extent={{-100,-100},{100,100}}), graphics));
end NiobiumCavity;
```

Lund University **Department of Automatic Control** **Box 118** **SE-221 00 Lund Sweden**	*Document name* MASTER THESIS *Date of issue* March 2014 *Document Number* ISRN LUTFD2/TFRT--5939--SE
Author(s) Riccard Andersson	*Supervisor* John Weisend, ESS Rolf Johansson, Dept. of Automatic Control, Lund University, Sweden (examiner) *Sponsoring organization*

Title and subtitle

Cool-down and Warm-up of the Cryogenic Distribution Line at ESS

Abstract

The European Spallation Source, ESS, is a joint collaboration of 17 European countries, where the world's most powerful neutron source will be built for future research within a vast variety of fields. In order to deliver the highly energetic neutrons, protons are accelerated to energies in the GeV range and then collided with a spallation target. This acceleration process requires superconducting cavities cooled down to slightly over 2 K, and this cooling is done through the cryogenic system using helium.

In this project, the cryogenic system at ESS has been modeled in Dymola. Simulations have been made of the cool-down and warm-up of the superconducting part of the accelerator. This was done in separate simulations for the cryogenic distribution line and for an individual cryomodule. Additionally, a model was created for the helium discharge system, in order to size the vent line leading rapidly expanded helium from the cold masses back to the cryoplant. The mathematical tools and structure of the modeling are described in a separate chapter.

Keywords

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title 0280-5316		*ISBN*
Language English	*Number of pages* 1-124	*Recipient's notes*
Security classification		

http://www.control.lth.se/publications/