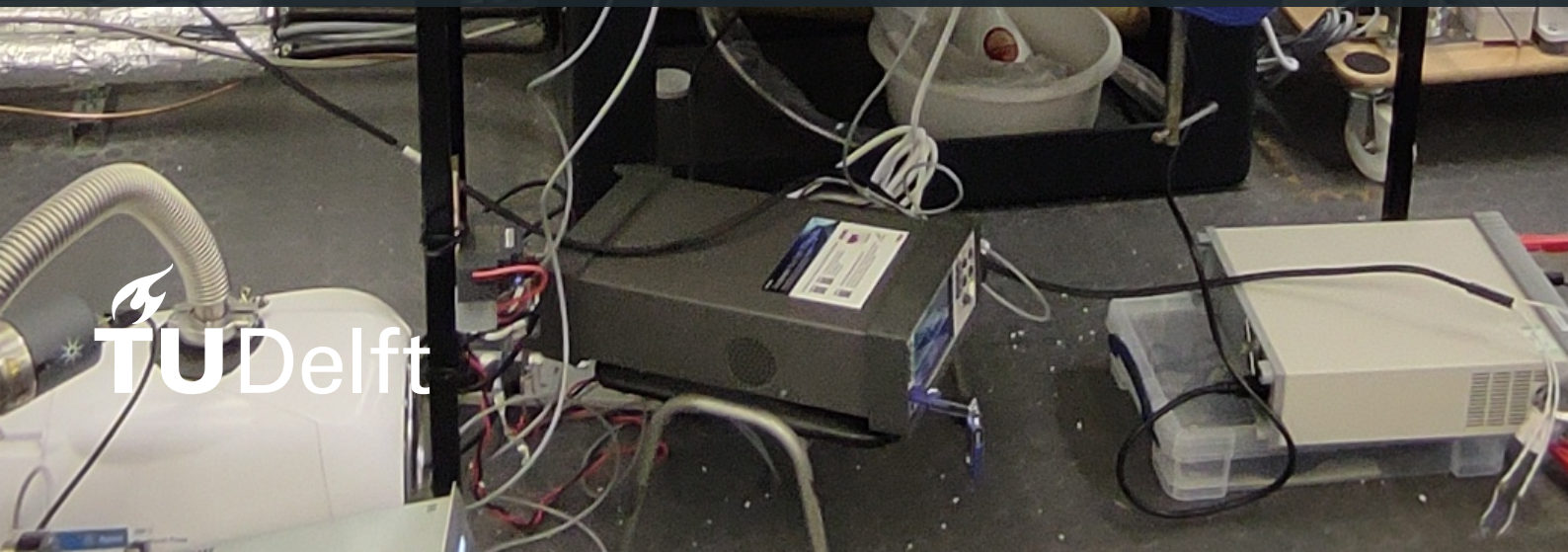
Thermal Management of Superconducting DC Link Terminations: Design and Experimental Validation of a Cryogenic Heat Sink for Hydrogen-Electric Aircraft

AE5222

Kunal Gupta



TU Delft

Thermal Management of Superconducting DC Link Terminations: Design and Experimental Validation of a Cryogenic Heat Sink for Hydrogen-Electric Aircraft

AE5222

MSc. Thesis

by

Kunal Gupta

to obtain the degree of Master of Science
at the Delft University of Technology.

Supervisors:	Dr. Chiara Falsetti (TU Delft)
	Dr. Pedro Barusco & Camille Chaper (Airbus UpNext)
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	March, 2025 - December, 2025
Student number:	4998715

Faculty of Aerospace Engineering · Delft University of Technology

Copyright © Kunal Gupta, 2025
All rights reserved.

Preface

This thesis is submitted in partial fulfillment of the requirements for the Master of Science degree in Aerospace Engineering, within the Flight Propulsion and Power track, at the Technical University of Delft.

The topic of this thesis and its scope were defined by Airbus UpNext, which supported and supervised the work throughout its entire duration. I am immensely grateful for the opportunity to have worked with such a dedicated and knowledgeable team.

Within Airbus, I would like to extend my sincere gratitude to my supervisors. My deepest thanks go to Dr. Pedro Barusco, who served as my mentor. His guidance was invaluable, and he consistently challenged me to go above and beyond my own capabilities. I am also incredibly thankful to Camille Chaper, who helped tremendously with the mechanical aspects of this project; his expertise was a critical asset. Furthermore, a special thanks must go to Dr. Swapnil Kharche, without whom the experimental test campaign would not have proceeded as smoothly as it did.

From the academic side, I wish to express my deepest thanks to my university supervisor, Dr. Chiara Falsetti. I first met Dr. Falsetti during the ”Internal Flows” lecture, where I greatly enjoyed her teaching style. It was for this reason, combined with her extensive experience in heat transfer and heat exchangers, that I requested her to be my thesis supervisor when the time came. Despite the geographical distance, with myself in Toulouse, France, and her in Delft, Netherlands, she provided constant support, kept me on track, and checked in regularly for updates. Her guidance and encouragement were essential to this project’s success.

Finally, I want to thank my friends and family for their unwavering support, not just throughout this thesis but during my entire time at the university. Their encouragement has been a constant source of strength.

Kunal Gupta
November, 2025.

Summary

The aviation industry is undergoing an important shift towards net-zero emissions, with hydrogen-electric aircraft (HEA) emerging as a leading solution. To enable megawatt scale power distribution on these platforms, superconducting (SC) DC systems are essential for their high efficiency and low weight. However, these systems possess a critical limitation: the termination where the cryogenic superconducting cable must connect to a current lead (CL) which in turn exits the cold environment and leads to the fuel cell. This generates a significant thermal load from both Joule heating and thermal conduction. If the heat generated here is not managed, it will cause the local temperature of the superconductor to exceed its critical limit, causing a ”quench” and a system power failure. This thesis directly addresses this issue by detailing the design, multi-physics analysis, and experimental validation of an optimized heat sink. The component is designed to be integrated into the superconducting DC link for the Airbus UpNext Cryoprop demonstrator.

The research methodology followed a structured progression from analytical modeling to high-fidelity simulation and experimental validation. First some 1D analytical models based on the McFee optimization method and general heat transfer equations were developed to get a preliminary idea on the feasibility of the design. These 1D codes provided the length to area ratio for an optimized geometry for a 1.5 kA Copper (RRR-10) current lead and specified the primary thermal load that the heat sink must dissipate: 80 W per current lead. With this heat load defined, a 3D-printed copper heat sink was designed and iterated using COMSOL Multiphysics simulations. This iterative process showed that the simple axial fin designs were not able to meet the project requirements, creating thermal bottlenecks and fluid stagnation at the current lead base. The key design innovation of this work was the development of a novel concentric fin topology. This design, which arranges the cooling fins radially outward from the CL base, creates a highly efficient, short thermal path to the 42 K supercritical helium. In simulation, this topology successfully met all project constraints, maintaining the HTS termination temperature below the 50 K critical limit with a predicted pressure drop of 90 Pa.

The design’s performance was then validated through an experimental campaign on a 3D-printed copper prototype at the University of Bath’s cryogenic test facility. The results provided two important yet contrasting outcomes that form the core conclusion of this work. The thermal model was proven to be accurate. Under the nominal 80 W load, the sensors measured a peak HTS termination temperature of 49.6 K. This result successfully meets the primary project requirement, validating the concentric fin topology as a viable solution for the Cryoprop demonstrator. However, a significant discrepancy was found in the pressure drop. The experiment measured 187 Pa across the heat sink assembly. This resulted in an underestimation of 45% compared to the 129 Pa predicted by the COMSOL simulation. This under prediction is attributed to real-world factors that were not captured in the ”perfect” CAD model such as the complex, non-uniform flow in the headers, a higher than estimated surface roughness ($> 120\ \mu m$) from the additive manufacturing process, and minor losses from the silver brazing assembly.

In conclusion, this thesis provides an important component for the operation of a superconducting powertrain for a hydrogen electric aircraft. More importantly, it quantitatively demonstrates that for high performance cryogenic hardware, the manufacturing and assembly are not just secondary concerns but primary performance drivers that must be taken into account in the design process.

Contents

Nomenclature

List of Abbreviations

ASCEND	Advanced Superconducting & Cryogenic Experimental powertrain Demonstrator
BSCCO	Bismuth Strontium Calcium Copper Oxide
CAD	Computer-Aided Design
CFD	Computational Fluid Dynamics
CHF	Critical Heat Flux
CL	Current Lead
CO2	Carbon Dioxide
CORC	Conductor on Round Core
CTE	Coefficient of Thermal Expansion
DC	Direct Current
DOF	Degrees of Freedom
EASA	European Union Aviation Safety Agency
FAA	Federal Aviation Administration
FEM	Finite Element Method
GHe	Gaseous Helium
GUI	Graphical User Interface
HEA	Hydrogen-Electric Aircraft
HTS	High-Temperature Superconductor
ICAO	International Civil Aviation Organization
LH2	Liquid Hydrogen
LHe	Liquid Helium
LN2	Liquid Nitrogen
LTAG	Long-Term Aspirational Goal
LTS	Low-Temperature Superconductor
MCU	Motor Control Unit
MLI	Multi-Layer Insulation
MW	Megawatt
NOx	Nitrogen Oxides
OFHC	Oxygen-Free High Conductivity
PEM	Proton Exchange Membrane
PPLP	Polypropylene Laminated Paper
RANS	Reynolds-Averaged Navier-Stokes
REBCO	Rare-Earth Barium Copper Oxide
RRR	Residual Resistivity Ratio
SAF	Sustainable Aviation Fuel
SC	Superconducting
SFCL	Superconducting Fault Current Limiter
TCR	Thermal Contact Resistance
TIM	Thermal Interface Material
TPMS	Triply Periodic Minimal Surfaces

List of Symbols

ΔP	Pressure drop, Pa
ΔT	Temperature difference, K
$\dot{m}$	Mass flow rate, kg/s
$\dot{Q}$	Heat transfer rate or heat load, W
$\dot{Q}_L$	Heat load at the cold end, W
ϵ	Emissivity, dimensionless
ϵ	Surface roughness, μm
ϵ	Turbulent dissipation rate, m^2/s^3
μ	Dynamic viscosity, Pa·s
μ_t	Turbulent viscosity, Pa·s
ρ	Electrical resistivity, Ω·m
ρ	Fluid density, kg/m^3
σ	Electrical conductivity, S/m

σ	Stefan-Boltzmann constant, W/(m^2·K^4)
A	Cross-sectional area, m^2
Bi	Biot number, dimensionless
c	Minimum spacing between poles, mm
C_d	Discharge coefficient, dimensionless
c_p	Specific heat capacity at constant pressure, J/(kg·K)
D	Diameter, m or mm
D_H	Hydraulic diameter, m
D_p	Pole diameter, mm
E	Electric field, V/m
F	Body force vector, N/m^3
f	Darcy-Weisbach friction factor, dimensionless
g	Clearance from heat sink edge, mm
H	Enthalpy, J/kg
h	Convective heat transfer coefficient, W/(m^2·K)
H_c	Critical magnetic field, A/m
h_f	Fin height, mm
I	Electric current, A
I_{nom}	Nominal operating current, A
I_{opt}	Optimal operating current, A
J	Current density, A/m^2
J_c	Critical current density, A/m^2
k	Thermal conductivity, W/(m·K)
k	Turbulent kinetic energy, m^2/s^2
L	Length, m or mm
L_0	Lorenz number, W·Ω/K^2
L_p	Length to peak temperature, m
n	Number of poles
N_f	Number of fins
Nu	Nusselt number, dimensionless
p	Pressure, Pa or bara
P_k	Production of turbulent kinetic energy
Pr	Prandtl number, dimensionless
Q	Volumetric heat source, W/m^3
q	Heat flux vector, W/m^2
R	Electrical resistance, Ω
Re	Reynolds number, dimensionless
Ri	Richardson number, dimensionless
T	Temperature, K
T_c	Critical temperature, K
t_f	Fin thickness, mm
T_H	Hot end temperature, K
T_L	Cold end (Lower) temperature, K
T_p	Peak temperature, K
U	Averaged velocity field, m/s
u	Fluid velocity vector, m/s
V	Electric potential, V

List of Figures

List of Tables

1

Introduction

The aviation sector is a significant contributor to global, human-induced, climate change. Although responsible for approximately 2.5% of global carbon dioxide (CO2) emissions [1], its total climate impact is considerably larger when effects such as nitrogen oxides (NOx) emissions and the formation of contrails are accounted for potentially scaling upto around 4% of total climate warming [1] and contributing up to 2/3 of aviation's total contribution [2]. With the demand in air travel expected to grow with years, the environmental footprint of the aviation sector will increase further. Looking back, the demand for air travel has often exceeded the improvements in aircraft efficiency [3]. Without significant changes, aviation's role in global emissions is expected to increase even further by 2050 [1].

Seeing this, many international bodies and the aviation industry have established ambitious targets where they aim for net-zero carbon emissions by 2050 [1]. Achieving these goals requires more than imporvements in the current designs or switching to Sustainable Aviation Fuels (SAFs). It requires a change in the system architecture itself. While SAFs play an important role in the transition, they face challenges related to their availability, production energy requirements, and production of CO2 emissions during flight, making them insufficient on their own to reach zero emissions [2]. Therefore, many global initiatives like the European Green Deal [4], the International Civil Aviation Organization's (ICAO) goal (LTAG) for net zero emissions [4], and strategies pursued by the U.S. Federal Aviation Administration (FAA) and the European Union Aviation Safety Agency (EASA) [4], show the move towards exploring, financing and certifying new solutions with hydrogen propulsion as the leading candidate.

1.1. Hydrogen-Electric Aircraft as a pathway to zero emission

Hydrogen-Electric Aircraft (HEA) represent a promising solution for achieving true zero in-flight carbon emissions in the aviation sector [5]. The fundamental concept involves using hydrogen as the primary fuel to generate electrical power, which then drives electric propellers. This involves the Proton Exchange Membrane (PEM) fuel cells which convert the onboard hydrogen and atmospheric oxygen into electricity and water. Concepts involving hydrogen combustion in modified gas turbines also exist but carry the problem of high temperature combustion [2]. Fuel cell based systems offer the advantage of eliminating both CO2 and NOx emissions during operation. This approach is very different from battery powered electric propulsion, which has its own limitations for larger commercial aircraft due to insufficient specific energy (energy per unit mass) of batteries, restricting range to only a few hundred kilometers.

The motivation for pursuing HEA is to reduce an aircraft's direct climate impact from CO2 emmisions and NOx emissions [2]. Hydrogen has a significantly higher specific energy (gravimetric energy density) compared to conventional jet fuel, approximately three times higher [5]. However, realizing this potential involves overcoming significant challenges. Hydrogen's low volumetric energy density, even when liquefied at cryogenic temperatures (approx. 20 Kelvin), means that it requires roughly four times the storage volume for the same energy content as kerosene [2]. This requires new aircraft designs with larger fuselages or alternative tank placements and design. Furthermore, the handling, storage, and transfer of cryogenic liquid hydrogen (LH2) present significant logistical and safety complexities for any ground fueling operations [5].

The core subsystems of a typical HEA include:

- Hydrogen storage: Cryogenics storage tanks to store liquid hydrogen at approximately 20 K.

- Power Generation: Fuel cells to generate the required power with the hydrogen on board.
- Power distribution: A high power electrical network, often configured as a DC link, to transmit electricity from the fuel cells to the electric motor [6].
- Electric propulsion: Electric motor to turn the propellers.
- Thermal management system: Required to manage the waste heat from the fuel cells and other electric components (often also needed to vaporize the LH2) [5].

1.2. Superconducting DC Power Systems

Future electric aircraft will require power systems that are capable of handling megawatt (MW) levels of power efficiently and with minimal weight penalty [2]. Conventionally, copper and aluminium based wiring is used for low power levels. However, these choices have significant limitations in terms of power density (in kW/kg) when scaled to high power levels [5]. The weight and volume associated with conventional conductors capable of carrying MWs of power comes with heavy weight penalties on the aircraft. This is where the benefits of the superconducting (SC) technology become more visible as they allow to overcome these limitations, particularly for the high power DC distribution network (the DC link) and also for motors and generators [5]. Superconductors offer two main advantages over their conventional counter parts:

- High current density: Superconducting wires can carry current densities over 100 times greater than copper or aluminum conductors of the same cross-section[7]. This allows to drastically reduce the size of the conductor and hence the weight for transmitting the same amount of power [8].
- Zero DC resistance: Below their critical temperature, superconductors show almost no electrical resistance to direct current. This eliminates the resistive (I^2R) power losses during DC power transmission, leading to a much higher system efficiency compared to the conventional conductors, where the losses can be very high over long distances or at high currents [8].

Furthermore, the high current carrying capacity of superconducting cables allows for a power system that can operate at lower voltages (e.g., below 1 kV or even 500 V [8]) while still transmitting megawatt level power. This is unlike the conventional systems that often require higher voltages to reduce current and minimize resistive losses and conductor weight. Lower voltage operation can simplify insulation requirements, reducing the risks associated with electrical arcing and improve the overall system safety [8].

An important factor that allows the practical application of superconducting technology in HEA is the availability of liquid hydrogen (LH2) on board the aircraft. Superconductors require cryogenic temperatures to operate (typically below 90 K or even liquid nitrogen temperature for High-Temperature Superconductors (HTS)) [7]. The LH2 onboard at approximately 20 K provides cooling power that can be used to cool the superconducting components [5]. This removes the need for heavy, complex, and power consuming cryocoolers, which are often a big hurdle for implementing superconducting systems in aerospace applications [9]. This combination of LH2 availability and the performance benefits of superconductivity helps address the critical bottleneck of electrical system weight and efficiency that currently affect the development of large-scale electric aircraft [5].

While superconducting cables offer significant advantages for the main power transmission lines operating entirely within the cryogenic environment, a critical interface exists where these cables must connect to other components operating at higher temperatures. This connection is made with a component known as a current lead [10]. Current leads bridge the large temperature differential between the cryogenic superconducting cable and devices operating at intermediate cryogenic temperatures (e.g., cryocooled power electronics at 77-100 K [10]) or, more challenging, at ambient temperatures (e.g., power sources or conventional busbars near 300 K) [10].

The design of the current leads presents a conflict between the electrical and thermal requirements. They must possess high electrical conductivity to efficiently transport large operating currents (in kiloampere range for a megawatt scale system [10]) with minimal voltage drop. Simultaneously, they must also exhibit low thermal conductivity to minimize the heat leak from the warmer environment into the cryogenic system [11]. These two requirements are fundamentally opposite in most conventional materials, as described by the Wiedemann-Franz Law, which links electrical and thermal conductivity [11]. The three main thermal loads on the interface come from thermal conduction, Joule heating, and contact resistance.

- Thermal conduction: Heat conduction through the current lead due to the large temperature gradient (ΔT), spaning over 200 K [10]. The magnitude depends on the current lead's material thermal properties and its geometry (cross-sectional area A and length L).
- Joule heating: As the high operating current (I) flows through the electrically resistive (non-superconducting) sections of the current lead, heat is generated according to I^2R, where R is the electrical resistance of that section [10]. This becomes increasingly significant at higher currents.
- Contact resistance: Imperfect electrical connections at the warm and cold ends of the current leads create contact resistance, leading to additional localized Joule heating [12].

It is important to remove all the aforementioned heat loads, especially near the connection point to the superconducting cable. This is because any excessive heat flowing from the current lead into the superconducting cable can raise the local temperature of the HTS above the superconductor's critical temperature (T_c), causing a transition to the normal, resistive state resulting in a "quench" [13]. A quench can lead to rapid heating, potential damage to the superconductor, and disruption of power transmission. As a result, an efficient heat sink capable of dissipating the heat loads described above is essential to ensure the reliable, safe, and optimal operation of the superconducting DC link.

1.3. Research objective and scope

This research framework comprises a central research objective accompanied by specific research questions, which are mentioned below. The research object is as follows:

Design and analyze an optimized heat sink to manage the thermal load at the junction between a superconducting DC link cable and its associated current lead within the power system architecture of a conceptual hydrogen electric aircraft.

To support this research objective a research question has been formulated and presented here:

Research question: "How do the required coolant mass flow rate, system pressure drop, heat sink length, diameter and mass vary with the nominal operating current for a superconducting hydrogen electric aircraft?"

To achieve this objective and find an answer to the research question, intermediate steps are required. These are included in the scope of this thesis and presented below. Items that are related to the work but not included within the scope of this thesis are also given below. The work assumes that the heat sink is being designed for a hydrogen-electric aircraft with a superconducting DC power distribution system cooled by gaseous helium, which in turn is cooled by onboard liquid hydrogen.

Within the scope of this thesis:

- Literature study on superconductivity and electrical distribution architecture of hydrogen-electric powered systems.
- Development of a thermal model for estimation of losses and heat load on the heat sink
- Selection and analysis of heat sink materials suitable for cryogenic aerospace applications.
- Geometric design and optimization of the heat sink.
- Thermal performance analysis of the heat sink under defined operating conditions in COMSOL Multiphysics.
- Experimental validation of the designed heat sink.

Outside the scope of this thesis:

- Detailed design of the overall hydrogen-electric aircraft architecture.
- Design and optimization of the superconducting cable itself.
- Detailed design of the current lead (its thermal load characteristics are considered inputs).
- Design of the primary cryogenic cooling system (e.g., LH2 tanks, pumps, primary heat exchangers).

2

Literature Review

2.1. Hydrogen Electric Aircraft Technologies

The pursuit of sustainable aviation has led to intensive research on alternative propulsion systems, with hydrogen-electric concepts receiving significant attention. These concepts differ fundamentally from conventional aircraft and require the integration of new technologies across multiple domains.

2.1.1. Hydrogen-Electric Aircraft Architectures

Several distinct architectures are being explored for HEA, each with its own characteristics and challenges. Firstly, the Fuel cell powered architecture utilizes fuel cells, typically Proton Exchange Membrane (PEM) fuel cells, to convert chemical energy stored in hydrogen directly into electrical energy with high efficiency [3]. The electricity then powers electric motors for propulsion. PEM fuel cells offer the significant advantage of producing zero CO2 and NOx emissions during operation, with water as the only byproduct [2]. However, they currently face challenges such as a lower power density compared to gas turbines [5], a substantial waste heat generation (often comparable in magnitude to the useful electrical power produced [5]) which then requires a large and heavy thermal management systems [5]. Furthermore, PEM fuel cells have a limited operational life (estimated around 10,000 hours [2]) and require a complex balance-of-plant systems for air compression, humidification, and water management, especially at high altitudes [2].

An alternative system architecture is the one used onboard Turbo-electric aircrafts. In this configuration gas turbines (potentially fueled by hydrogen combustion) drive electrical generators, which in turn supply power to multiple electric motors driving the propellers [5]. Hydrogen combustion turbines avoid the complexity of fuel cells but still produce NOx emissions [2] and face material challenges like hydrogen embrittlement in turbine components [2]. Closed-cycle concepts, such as the Closed Brayton Cycle, are also proposed to potentially mitigate some of these issues [2]. A key advantage of the turbo-electric approach is the ability to decouple the turbine speed from the propeller speed, allowing each to operate at its optimal efficiency point [14].

Lastly, there also exists the Hybrid-electric architecture. This concept combine a primary power source (gas turbine or fuel cell) with an energy storage system, typically batteries, to provide additional power during high demand phases (like takeoff) or to enable engine optimization during cruise [2]. Hybridization can offer moderate reductions in fuel consumption (ranging from 3 to 10% depending on the architecture and mission [2]). However, they do not achieve zero emissions (if using gas turbines) and are still limited by the significant weight and energy density of current battery technology [5]. The added complexity of managing multiple power sources and the thermal management of the battery also creates more challenges [15].

The choice of architecture significantly influences the design requirements for the electrical power system. Fuel cell systems tend to operate at lower voltages and higher currents, impacting the design of the DC link and associated components. They also introduce a large thermal management burden at relatively low temperatures [5]. Turbo-electric systems might operate at higher voltages but introduce different thermal signatures and potentially NOx emissions if using hydrogen combustion. Hybrid systems add the complexity of battery integration and thermal control [15]. Therefore, the specific HEA architecture defines the operating environment, power characteristics, and thermal loads that the DC link components must dissipate.

2.1.2. Liquid Hydrogen Storage

The hydrogen gas is the lightest material in the world and therefore it has a extremely low density. To make up for this low density, storing it as a cryogenic liquid (LH2) at approximately 20 K (-253 °C) is considered one of the most viable approach for achieving acceptable energy density by volume for aviation applications [5]. Other ways of storing hydrogen onboard the aircraft include cryocompressed (cch2) and compressed hydrogen (cH2) as shown in Figure 2.1. Despite liquefying hydrogen gas, LH2 still occupies about four times the volume of kerosene for the same energy content [5]. This means that larger fuel tanks compared to conventional aircraft are required for hydrogen powered flight. These tanks also need to be well insulated (using vacuum jackets and multi-layer insulation) to minimize boil-off of the cryogenic fuel during storage and flight [16]. Integrating these large, nonconventional cryogenic tanks into aircraft structures creates several design challenges, often leading to considerations of blended-wing-body or other non-standard airframe configurations[2]. Another concern with LH2 is the on ground refueling infrastructure. It needs to be carefully handled due to its highly explosive nature [5].

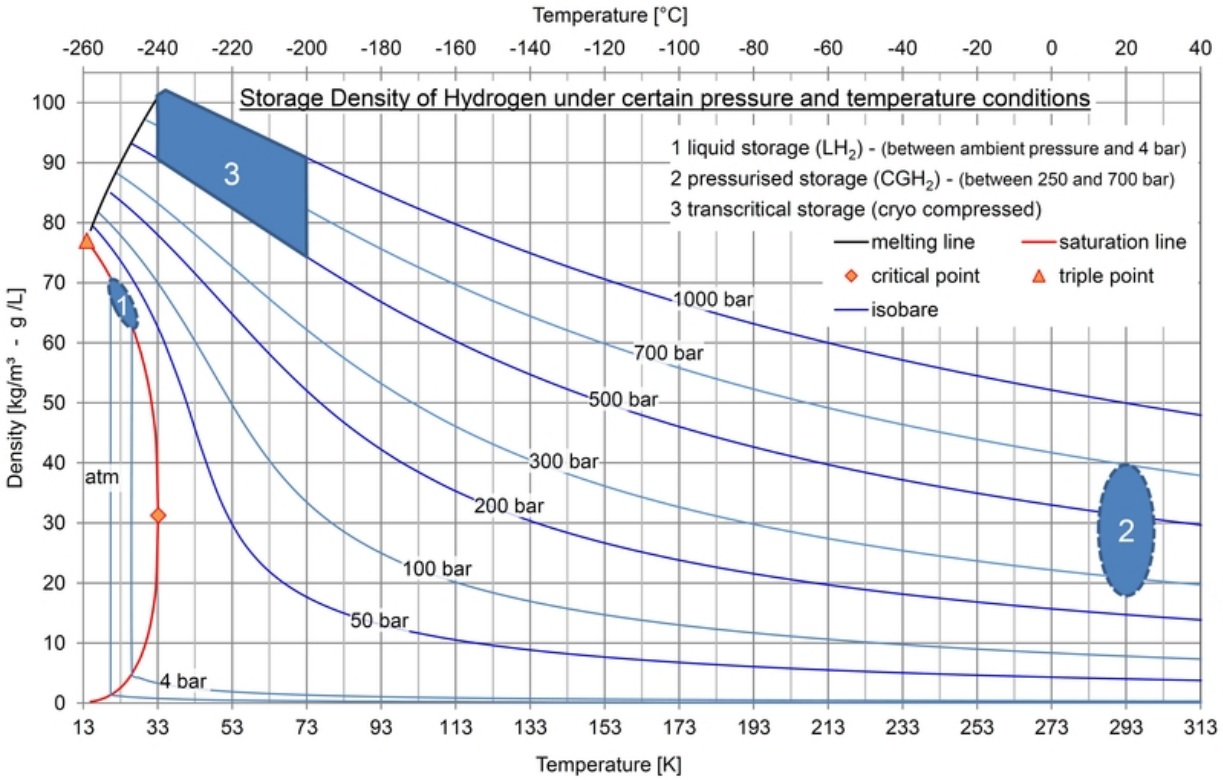


Figure 2.1: Storage density of hydrogen under various pressures and temperatures. [17]

2.2. Superconducting Power Systems for Aerospace Applications

The need for applying superconducting technologies comes from the limitations in the power density (mainly due to the weight of the cables) and efficiency of conventional electrical systems required for megawatt scale aircraft propulsion system.

Superconductivity is a phenomenon observed in certain materials below a critical temperature (T_c), where they exhibit zero DC electrical resistance and expel magnetic fields (Meissner effect) [18]. The superconducting state the material is also limited by a critical magnetic field (H_c) and a critical current density (J_c). Exceeding any of these parameters can causes the material to revert to its normal, resistive state [19].

Superconductors are generally categorized into two main categories of Low Temperature Superconductors (LTS) and High Temperature Superconductors (HTS) with the exception of magnesium di-boride (MgB_2) which lies in between both.

- LTS: These are materials like Niobium-Titanium (NbTi) and Niobium-Tin (Nb3Sn) that normally have a T_c below 23 K. Therefore they are cooled with liquid helium (LHe) at 4.2 K [20]. They are mature technologies and used in applications like MRI magnets, particle accelerators, and many existing fusion reactors.
- HTS: The high temperature superconductors include ceramic materials like Bismuth Strontium Calcium Copper Oxide (BSCCO, specifically Bi-2223) and Rare-Earth Barium Copper Oxide (REBCO, often YBCO) discovered in the 1980s [21]. They exhibit a critical temperature above the boiling point of liquid nitrogen (77 K), with some even exceeding 90 K [7]. This allows for cooling with LN2 or cryocoolers operating at higher and more easily achievable temperatures compared to liquid helium [22]. The REBCO conductors that are typically manufactured as coated tapes, generally offer superior performance in the presence of magnetic fields. This is especially true at higher temperatures, compared to BSCCO [7].
- MgB_2: The MgB2 has a critical temperature around 39 K [7]. It operates in a temperature range achievable with LH2 cooling (20 K) or cryocoolers, offering a potential intermediate option between LTS and HTS in terms of performance and cooling requirements [7]. Nevertheless, in practice they are often used at 20 K.

2.2.1. Advantages & challenges of superconducting aircraft power systems

The application of superconducting technology, particularly HTS materials cooled by onboard LH2 (directly or indirectly), offers many advantages for future aircraft electrical systems. For instance, the superconductors can carry large amounts of current at cryogenic temperatures allowing for the cables to be significantly smaller and lighter than their conventional counterparts [8]. The potential power densities for superconducting motors and generators may exceed 20 kW/kg, potentially reaching as high as 40-55 kW/kg, compared to 4-8 kW/kg for advanced conventional machines [5]. This weight reduction is significant in aerospace applications.

Furthermore, the elimination of DC resistive losses in superconducting cables leads to highly efficient power transmission [8]. Superconducting motors and generators made with superconducting cables can also achieve high efficiencies, exceeding 98-99%, due to the reduction of the winding losses [8]. This reduces the total energy consumption and the waste heat generation within the electrical system itself. Finally, since superconducting cables can carry very high currents, it becomes feasible to transmit MW level power at relatively lower voltages (e.g., < 1 kV) [8]. This leads to a simpler insulation design and reduces the risk of arcing, improving the overall system safety compared to the conventional high voltage systems [20].

Despite the aforementioned benefits of the superconducting cables, implementing superconducting systems in aircraft involves considerable challenges. Maintaining low operating temperatures requires a highly complex cryogenic cooling system which also needs to be lightweight [9]. Even though the onboard LH2 serves as the cold source for cooling the superconducting cables, integrating the cooling loops, managing the cryogen (LH2 or an intermediate like gaseous helium), and ensuring system reliability under flight conditions adds more complexity to the system. If dedicated cryocoolers are needed (e.g., for components operating at temperatures different from LH2, or in non-LH2 aircraft concepts), their weight, power consumption, and reliability become critical concerns [9].

Quench is another risk for an aircraft with superconducting cables. Superconducting systems are vulnerable to quenching (an abrupt loss of the superconducting state if temperature, magnetic field, or current density exceed critical limits) [22]. Quenches can be triggered by electrical faults (causing current surges), transient events, localized heating, or even failures in the cooling system. A quench results in a rapid increase in resistance of the superconductor and heat generation. This can damage the superconductor, or worse burn it. Robust quench detection and protection systems, possibly including superconducting fault current limiters (SFCLs), are therefore essential for safe operation [22]. This requires the designs to incorporate sufficient temperature and current margins to minimize the probability of a quench [9].

Furthermore, superconductors carrying alternating current or exposed to time-varying magnetic fields (as in motors, generators, transformers, or AC cables) experience AC losses [8]. These losses are primarily due to magnetic hysteresis in the superconductor and eddy currents in the material. This phenomenon also leads to heat generation within the cryogenic environment. This heat must also be removed by the

cooling system, resulting in a significant thermodynamic penalty (refrigeration power required per watt of heat removed increases dramatically at lower temperatures [23]). Minimizing AC losses through conductor design and machine topology is crucial for AC superconducting applications [8]. However, DC power transmission cables inherently avoid these AC losses in the superconductor itself.

Finally, from a mechanical aspect, HTS materials, particularly REBCO coated conductors, are ceramic-based and relatively brittle. They are sensitive to mechanical stress, strain, and bending, requiring careful handling during manufacturing and integration into components like cables and machine windings [24]. The large thermal contractions experienced during cooldown from room temperature to cryogenic temperatures also induce mechanical stresses that must be managed in the system design [25].

Integrating superconducting components, cryogenic systems, and protection schemes results in a highly complex system [24]. Compared to conventional power systems, superconducting technology has low maturity in the aerospace environment. This often increases the risks and development costs for projects including superconductivity [26].

2.2.2. Superconducting cables & cryogenic cooling

Superconducting DC power cables typically consist of HTS tapes (e.g: REBCO or Bi-2223) wound helically around a flexible former to form the conductor layers [27]. Electrical insulation is then applied over the conductor, often using lapped tapes of cryogenic-compatible dielectric materials like polypropylene laminated paper (PPLP) or specialized polymers (e.g., CryoFlex™) [27]. This ”cold dielectric” design, where the insulation operates at cryogenic temperature, helps manage thermomechanical stresses arising from the difference in thermal contraction between the conductor and insulation [25]. The entire cable core is housed within a cryostat, which is essentially a vacuum insulated pipe to minimize heat leak from the ambient environment. Cooling is achieved by flowing a cryogen through the cable assembly, either within the conductor former as seen in Figure 2.2, or within the cryostat completely surrounding the cables (also known as immersed cooling as seen in Figure 2.3 [28]. For HEA applications, the coolant would likely be LH2 itself or gaseous helium (GHe) that has been cooled by LH2 via a heat exchanger. Conductor on Round Core (CORC) cables, which utilize HTS tapes wound on a small diameter round former, are a specific topology being considered for compact, high-current applications as seen in Figure 2.4 [10].

Mechanical cryocoolers (e.g: Stirling, Pulse Tube, Brayton cycle refrigerators) can provide cooling at various temperature levels but face challenges in aerospace regarding weight, reliability, and vibration [9]. Efficiencies for cryocoolers operating around 50 K might be in the range of 10-20% of Carnot, with targets needing to reach 30% for feasibility in some aircraft concepts [9]. To avoid the weight and inefficiency of dedicated cryocoolers for the main superconducting components, use of GHe(cooled by LH2) or LH2 directly looks highly promising but requires careful design of the cryogenic fluid management system and heat exchangers [28]. A comparison between a superconducting and a conventional power system for an aircraft is presented in Table 2.1.

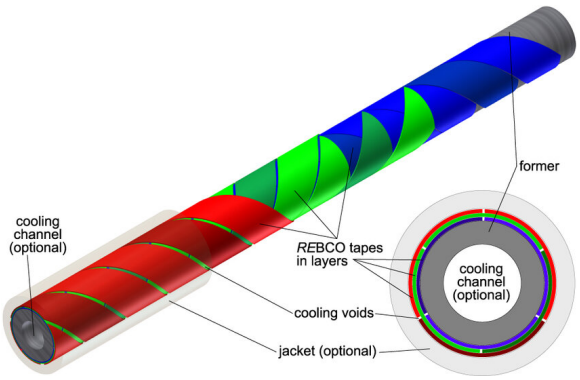


Figure 2.2: Schematic drawing of a CORC cable with REBCO tapes wound around a former. The cable can be fitted with a hollow former for forced flow cooling. [29]

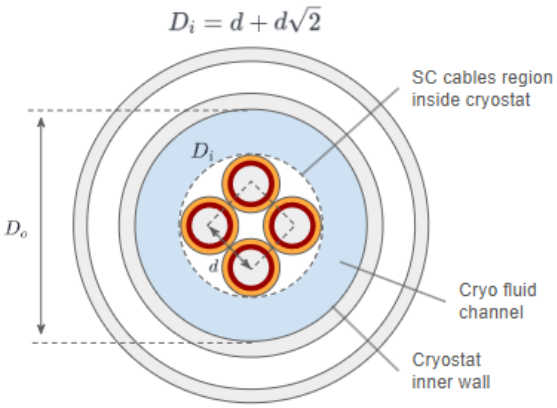


Figure 2.3: A Schematic diagram of the CORC cables under immersed cooling.

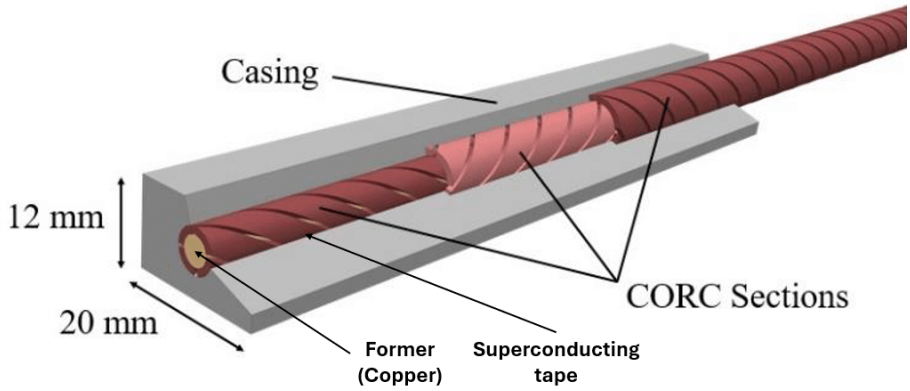


Figure 2.4: A simplified overview of a CORC cable. [30]

Table 2.1: Comparison of Superconducting vs Conventional Power System Components for Aircraft.

Components	Parameter	Conventional (State-of-Art)	Superconducting (Projected)	References
Power Cables (DC link)	Current density	1-5 A/mm2 (Cu/Al)	>100 A/mm2 (HTS)	[7]
	Efficiency (DC)	<100% (Resistive losses)	∼100% (Zero DC resistance)	[8]
	Weight/Volume	High for megawatt scale	Significantly reduced	[8]
	Voltage level	High (to reduce current)	Low (as higher currents are allowed)	[8]
	Cooling	Ambient / Forced (air or liquid)	Cryogenic (LH2/GHe)	[16]
	Complexity	Moderate	High	[27]
	TRL (Aviation)	High (9)	Low-Medium (3-5)	[26]
Motor/Generator	Specific power	4-8 kW/kg	>20 kW/kg	[5]
	Efficiency	∼95-97 %	>98-99.9 %	[8]
	Cooling	Air / Liquid	Cryogenic (LH2/GHe)	[9]
	Weight/Volume	Baseline	Significantly reduced	[5]
	Complexity	High	Very High (SC windings, cryo integration, AC losses)	[19]
	TRL (Aviation)	High (7-9)	Low (2-4)	[26]

2.3. Current Leads

Current leads are part of almost every superconducting system, providing the electrical connection between the cryogenic superconducting elements and the external power supplies or loads operating at room temperature [10]. In the context of an HEA DC link, they connect the superconducting cable (40 K) to components potentially at ambient (300 K) on the fuel cell side or intermediate cryogenic temperatures (77-100 K) on the MCU (Motor Control Unit) side. Figure 2.5 shows a simplified view of the electrical powertrain for the ASCEND (Advanced Superconducting & Cryogenic Experimental powertraiN Demonstrator) project carried out at Airbus. This includes a superconducting distribution system (cables and protection system), a cryogenically cooled motor control unit, a superconducting motor, and a cryogenic cooling system.

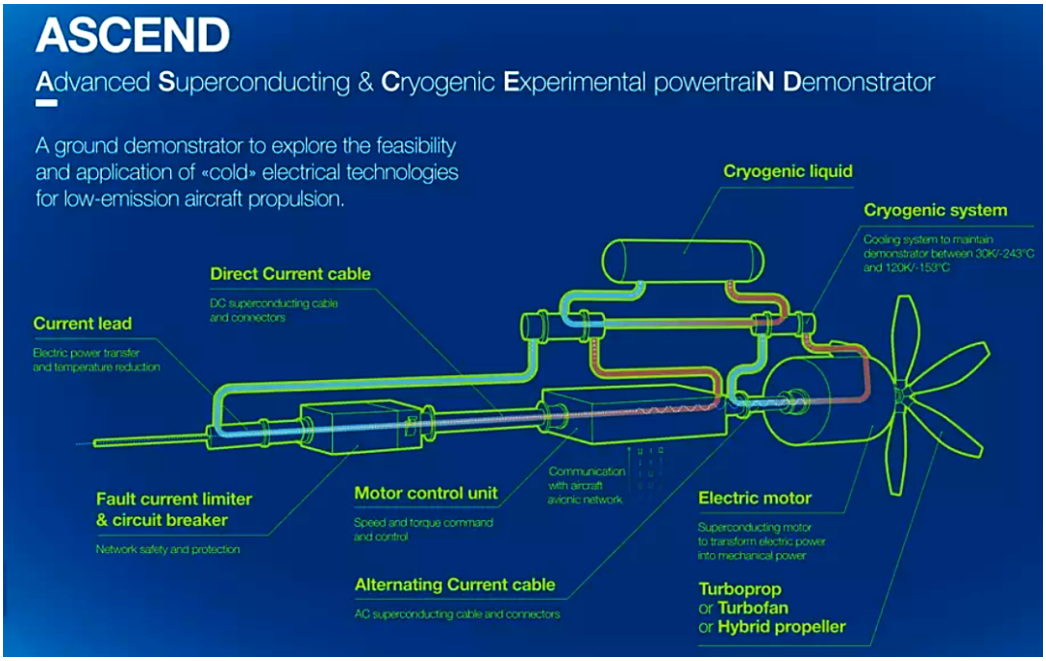


Figure 2.5: ASCEND system architecture. [31]

2.3.1. Design principles and optimization

The fundamental design goal for a current lead is to efficiently transfer the electrical current while minimizing the amount of heat transferred from the warmer end to the colder, cryogenic end. This highlights the two main sources of heat leaks into the system for current leads: conduction and Joule heating. Conduction heat loss in the current lead is the heat transfer along the current lead due to the temperature difference (ΔT) between the warm and cold ends. It is governed by the material's thermal conductivity and the current lead's geometry [10]. This is the main heat leak mechanism when no current is flowing [32]. The Joule heating losses, however, are due to the electrical resistance (R) of the current lead when carrying current (I). The total Joule heat depends on the current, the lead geometry (length-to-cross-sectional area ratio), and the electrical resistivity. This component becomes the primary heat source at high operating currents. The secondary sources of heat leak into the system include environmental radiation [33], electrical contact resistance between components [12], and finally, flow friction, which causes the temperature of the coolant to increase as it flows through channels with finite roughness.

In order to minimize the main heat load onto the current lead, an optimization process to balance the heat flow from thermal conduction and Joule heating generated within the current lead needs to be established [10]. For purely resistive (non-superconducting) current leads cooled only at the ends (conduction-cooled), the Wiedemann-Franz Law (which states that the ratio of thermal conductivity 'k' to electrical conductivity 'σ' is proportional to temperature T, k/σ ≈ $L_0 T$, where L_0 is the Lorenz number) provides a theoretical basis for optimization [11]. This law implies that for a given material and temperature difference (T_{hot} to T_{cold}), there exists an optimal ratio of length (L) to cross-sectional area (A) that minimizes the heat leak (Q_{cold}) per unit current (I). The minimum heat leak for such an optimized lead is approximately proportional to Equation 2.1. It is largely independent of the specific material chosen, although the optimal L/A ratio itself is still dependent on the material [32]. Practical optimization often makes use of numerical methods such as the McFee method, which account for the actual temperature-dependent properties of materials [10].

$$Q_{min} = \sqrt{T_{hot}^2 - T_{cold}^2} \tag{2.1}$$

Cooling the current lead along its length, typically using the boil-off vapor from a liquid cryogen bath (vapor cooling or self-cooling) can significantly reduce the heat leak compared to pure conduction cooling [32]. However this is not always possible due to volume and space constraints. An alternative is to introduce intermediate heat sinks connected to either cryocooler stages or separate cooling loops to reduce

the heat load reaching the coldest stage [11]. One may also integrate the heat sink directly into the existing cooling loop for the HTS (immersed in cold helium flow), as is being done at CRYOPROP, Airbus UpNext.

2.3.2. Materials for current leads

The choice of material is critical for achieving a high performance current lead. The two materials often used are either a conventional resistive copper (with high purity) or an extended HTS section. An extended HTS section however is mostly used when going from cold temperatures (4-20 K) to intermediate temperatures (50-80 K). Oxygen-Free High Conductivity (OFHC) copper is frequently used for going to the warm end. It has excellent electrical conductivity, but requires geometric optimization (specific L/A ratio) to minimize heat leak from conduction [11]. The Residual Resistivity Ratio (RRR), which indicates purity, affects both electrical and thermal conductivity at low temperatures. Lower RRR copper or alloys like phosphor bronze or brass have lower thermal conductivity, reducing the standby (zero current) heat leak, but their higher electrical resistivity leads to increased Joule heating at the design current. Figure 2.6 shows how the resistivity of copper varies with its RRR value. The choice depends on the relative importance of no-current versus operational(current-carrying phase) heat loads.

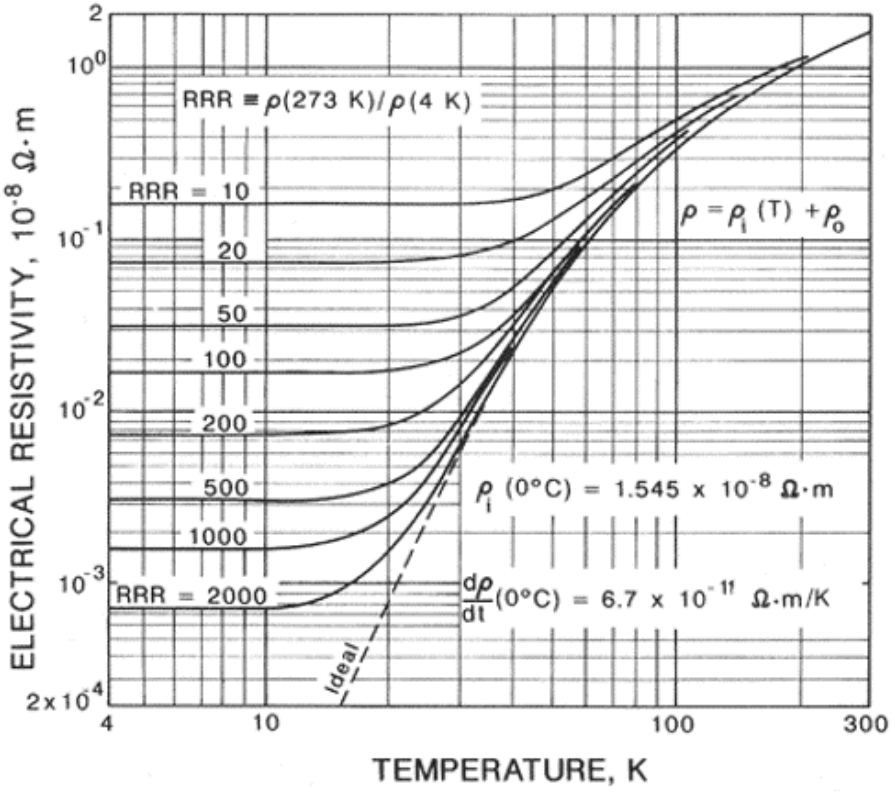


Figure 2.6: Electrical resistivity of copper for varying purity (RRR value) vs temperature. [34]

2.4. Fundamentals of cryogenic heat transfer

Understanding the heat transfer mechanisms at cryogenic temperatures is essential in designing and analyzing the thermal performance of the current lead interface and the associated heat sink. While the general principles of conduction, convection, and radiation still apply, material properties and fluid (cold helium) behaviour change rapidly at cold temperatures.

Heat conduction through solids is still governed by Fourier's law, Q = -kA(dT/dx), where Q is the heat transfer rate, k is the thermal conductivity, A is the cross-sectional area, and dT/dx is the temperature gradient. However, the thermal conductivity (k) and the specific heat (c_p) of the material are strongly dependent on the temperature at low temperatures. For pure metals such as copper and aluminum, k typically increases as the temperature decreases from room temperature until a peak is reached (around

20-50 K), after which it starts to decrease. The c_p follows an inverse trend where it starts at zero and rises with temperatures until it reaches 150-200 K where the gradient of the curve starts to decrease. This phenomenon can be seen in Figure 2.7 for Copper (RRR-100) and Aluminium (RRR-100) respectively. The data for these curves was obtained from CryoComp [35]. Accurate calculation of heat conduction across large temperature differences requires integrating k(T) over the temperature range, often expressed as the thermal conductivity integral, ∫k(T)dT.

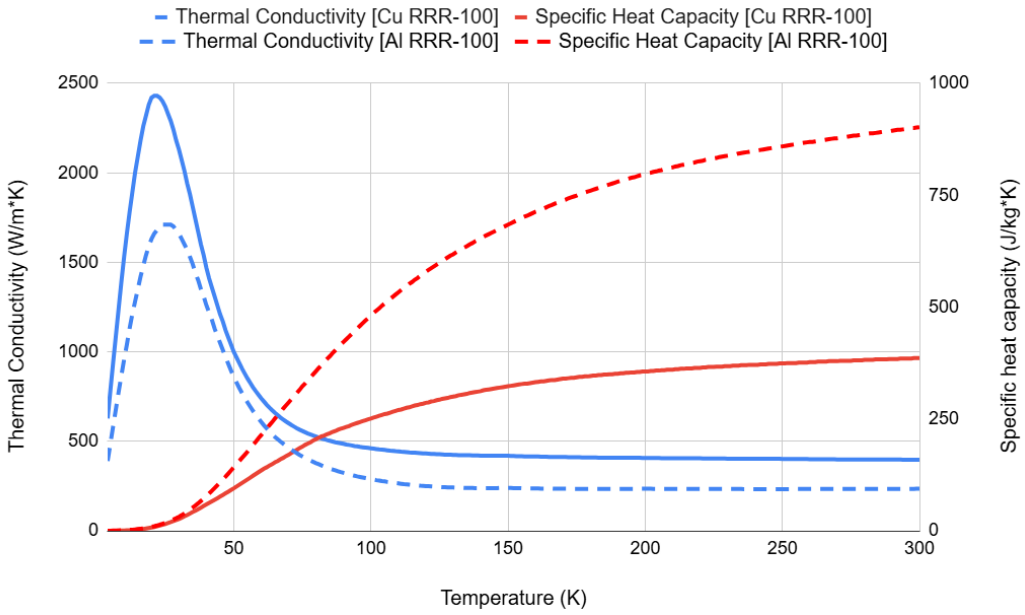


Figure 2.7: Thermal conductivity and Specific heat capacity of Copper RRR 100 and Aluminium RRR 100 versus temperature.

The convective heat transfer between a surface and a fluid is described by Newton's Law of Cooling, $Q = h \cdot A \cdot (T_{surface} - T_{fluid})$, where h is the convective heat transfer coefficient. In cryogenic systems, convection can be natural or forced, and can involve single-phase fluids or phase change (boiling/condensation). When a cryogen (like LH2, LN2, or GHe) is pumped through channels (forced convection) or over surfaces as in heat exchangers or cable cooling passages, the heat transfer coefficient 'h' depends strongly on the fluid properties (viscosity, density, thermal conductivity and specific heat all of which vary significantly with temperature and pressure at cryogenic conditions), the flow velocity, the flow regime (laminar or turbulent), and the geometry of the flow path. However, natural convection occurs purely due to a difference in density, which is often caused by a temperature gradient. In a system where the fluid has a very slow velocity (such as inside the LH2 storage tank), the Richardson number can be calculated to estimate the main heat transfer means. Typically, the natural convection is negligible when Ri < 0.1, forced convection is negligible when Ri > 10, and neither is negligible when 0.1 < Ri < 10 [36].

If the heat flux into a liquid cryogen causes the fluid to boil, very high heat transfer coefficients can be achieved in the nucleate boiling regime [37]. Nevertheless, exceeding a certain Critical Heat Flux (CHF) leads to the formation of a vapor film at the surface (film boiling), which drastically reduces the heat transfer coefficient [37]. This, however, does not apply to cryogenic fluids that are in the supercritical phase. If the cryogenic fluid is maintained at a pressure above its critical pressure (e.g: supercritical helium used for cooling HTS at Cryoprop, Aibus UpNext), boiling does not occur. Heat transfer behaves similarly to single-phase gas convection, but fluid properties can change very quickly near the "pseudocritical" temperature [38].

Finally, the thermal radiation between surfaces is governed by the Stefan-Boltzmann Law, Q = εσA$(T_{hot}^4 - T_{cold}^4)$, where ε is the effective emissivity of the surfaces and σ is the Stefan-Boltzmann constant. The T^4 relationship makes radiation a significant heat transfer mechanism at high temperature differences, but it can be easily overcome with the help of Multi-Layer Insulation (MLI). It consists of multiple layers of reflective foil separated with spacers in a vacuum [33].

2.5. Heat sink design for cryogenic applications

Heat sinks are an important component in any thermal system as they help maintain a safe operating temperature by dissipating the heat generated (or caused) by other components. The goal of a heat sink is to effectively dissipate the heat load into the coolant. This involves maximizing the effective surface area available for heat transfer, using high conductivity (thermal) materials, and optimizing the geometry to increase the heat transfer coefficient by altering the flow velocity. In aerospace applications, these thermal performance goals must be achieved while strictly adhering to constraints on weight and volume, as any excess mass or size directly penalizes aircraft performance [39]. Furthermore, the heat sink must have sufficient mechanical strength to withstand operational loads, vibrations, and thermal stresses induced by large temperature cycles in cryogenic systems.

2.5.1. Material selection & manufacturing of cryogenic heat sinks

To achieve a high-performing heat sink, specific guidelines must be followed to get the right material. The chosen material should display high thermal conductivity across the operating temperature range. This is important to minimize any temperature gradients within the heat sink itself. Materials like OFHC copper and high purity aluminium (AL-1100 or AL-6061) are common choices as they have relatively high thermal conductivity at low cryogenic temperatures [40]. Another material property that needs to be considered is the specific heat capacity. A low specific heat capacity allows the heat sink to cool down much faster and requires less cryogen. However, the low c_p also implies that in case of temperature fluctuations in the system, the heat sink may not be able to serve as a thermal buffer. The c_p value of most metals decreases significantly at low cryogenic temperatures as seen in Figure 2.7. The last important aspect to be considered in the material choice for a heat sink is its coefficient of thermal expansion (CTE). Minimizing the mismatch in CTE between the heat sink material and the components it connects to (e.g., current lead, cooling structure) is important to reduce the thermomechanical stress during cooldown and warmup cycles [25]. Large CTE mismatches can lead to high stresses, potential joint failure, or increased thermal contact resistance over time. Table 2.2 provides an overview of the relevant properties of certain materials for the heat sink and of stainless steel (SS-304L) as a baseline. Meanwhile the curves for the coefficient of thermal expansions as a function of temperature are provided in Figure 2.8.

Table 2.2: Overview of thermal properties of candidate materials for the heat sink.

Material	Property	Value at 20 K	Value at 77 K	Value at 300 K	Units	References
OFHC Copper (RRR = 100)	Thermal conductivity (k)	∼1500	∼500	∼400	W/(m*K)	[40]
	Specific heat (cp)	∼1	∼150	∼385	J/(kg*K)	[40]
	Density	8960			kg/m3	[40]
Aluminium 6061-T6	Thermal conductivity (k)	∼30	∼80	∼170	W/(m*K)	[40]
	Specific heat (cp)	∼5	∼350	∼900	J/(kg*K)	[40]
	Density	2700			kg/m3	[40]
Aluminium 1100	Thermal conductivity (k)	∼800	∼300	∼220	W/(m*K)	[40]
	Specific heat (cp)	∼5	∼360	∼900	J/(kg*K)	[40]
	Density	2710			kg/m3	[40]
Stainless Steel 304L	Thermal conductivity (k)	∼0.5	∼8	∼15	W/(m*K)	[40]
	Specific heat (cp)	∼2	∼200	∼500	J/(kg*K)	[40]
	Density	8000			kg/m3	[40]

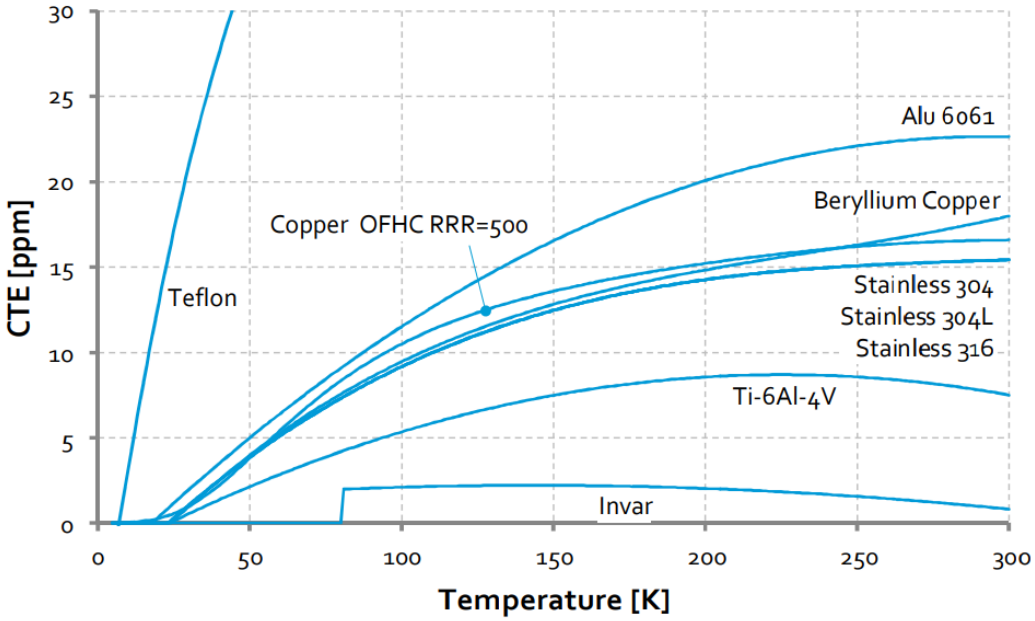


Figure 2.8: Coefficient of thermal expansion of various materials as a function of temperature [41].

Once the correct material is selected, several manufacturing process exist to make simple fin-type heat sink, such as water jet cutting, mechanical cutting, etc. However, advanced manufacturing techniques like additive manufacturing (like 3D printing), which allows for complex, optimized geometries that are difficult to produce conventionally [39] are often used to maximize the surface area to volume ratio. Metal foam type structure can significantly improve the heat transfer coefficient by making the flow more turbulent and offering higher surface area. However, it comes at the cost of a higher pressure drop and low thermal conductivity as seen for the 'Foam on Fins' design developed by L. Bromberg et al. for their superconducting cooling system at MIT's Plasma Science and Fusion Center [42]. The ability to effectively join the heat sink to the heat source and cooling structure (e.g., via brazing, bonding, or clamping with TIMs) is another aspect to consider for minimizing interface resistance.

2.5.2. Use of heat sinks in cryogenic systems

Heat sinks or integrated heat exchangers are commonly employed at current lead terminations. Designs often involve flowing the system cryogen (LHe, LN2, GHe) through channels or structures thermally coupled to the warm end of the lead to intercept the heat [28]. Examples include copper blocks with simple extruding fins, internal channels, perforated plates to enhance turbulence [12], or more advanced designs like foam-on-fin heat exchangers developed for efficient turbo-Brayton cryocoolers [42]. Conduction cooling to cryocooler stages is used in cryogen-free magnet systems, requiring thermal linking [43]. Solid cryogens have also been considered as thermal buffers or temporary heat sinks in case of cryocooler failure [33]. Commercial HTS leads often incorporate features for heat sinking at the warm end, sometimes recommending specific interface materials like sapphire [44]. A simplified 2D representation of the heat sink envisioned for use in the DC link at Airbus UpNext is presented in Figure 2.9.

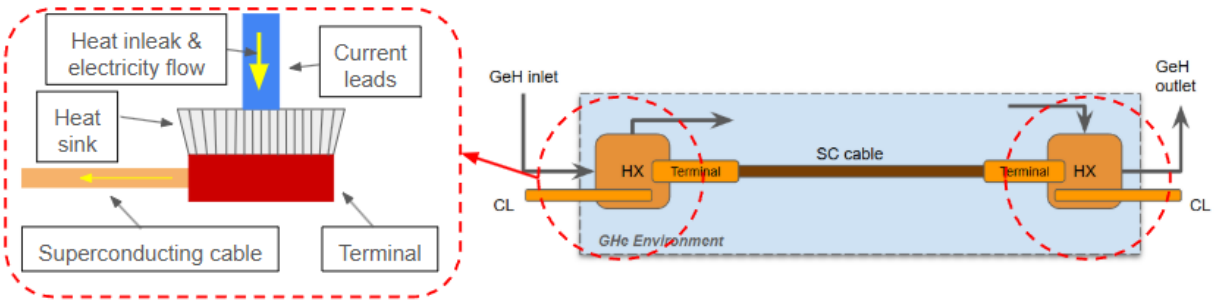


Figure 2.9: A simplified representation of the heat sink to be integrated into the Cryoprop DC link.

2.6. Thermal interface management in cryogenic systems

In any assembly where heat must be conducted between two solid components, the thermal resistance at the interface can significantly impede heat flow. This Thermal Contact Resistance (TCR) is particularly critical in cryogenic systems where efficient heat transfer is necessary, and TCR can become an important factor in the overall thermal budget [45].

2.6.1. Thermal contact resistance and its cause

The thermal contact resistance (TCR) occurs mainly because real connecting surfaces are not perfectly smooth. Even highly polished surfaces exhibit microscopic roughness (asperities). When two such surfaces are pressed together, actual physical contact occurs only at the peaks of these asperities, resulting in a contact area that is typically much smaller than the nominal contact area [45]. Heat flowing across the interface is constricted to these small contact spots, leading to a temperature drop across the junction, resulting in a TCR. The gaps between the contact spots are usually filled with the surrounding fluid (e.g., air, cryogen), which may contribute a parallel path for heat transfer. This however is often negligible in vacuum or low-conductivity fluids.

There are 4 main factors influencing the thermal contact resistance. These are the contact pressure, contact surface, material properties, and temperature. Increasing the mechanical force presses the surfaces together causes the asperities to deform plastically or elastically, increasing the number and size of contact spots and thus the real contact area. This then leads to a significant reduction in TCR [45]. The other factor influencing the TCR is the contact roughness. Surface roughness, flatness, and waviness directly impact the geometry of the contact interface. Generally, smoother and flatter surfaces result in lower TCR. Furthermore any substance present in the gaps between contact spots can affect the heat transfer. This may include oxide layers or surface contaminants which act as additional thermal barriers, increasing TCR [45]. On the other hand, intentionally introducing a Thermal Interface Material (TIM) can drastically reduce TCR.

The material properties also have a significant impact on the TCR between two components. The bulk thermal conductivity of the contacting materials influences heat spreading near the tiny spikes at the contact area (asperities). Material hardness also affects the deformation of asperities under pressure which results in more or less resistance [45]. Finally, the TCR is strongly temperature dependent. This is even more the case at cryogenic temperatures. As temperature decreases, materials become harder and stiffer, reducing asperity deformation for a given pressure. Furthermore, the material properties also change significantly at cryogenic temperature. Typically, TCR increases significantly as temperature is lowered [45].

2.6.2. Thermal interface materials (TIMs) for cryogenic use

To lower the TCR, TIMs are often applied between contacting surfaces. Suitable TIMs for cryogenic applications easily fit inbetween the surface irregularities and have good thermal conductivity at low temperatures while also remaining stable under thermal cycling. Some TIMs currently used in the industry are presented below:

- Indium foil: Indium is a very soft, malleable metal that files any irregularities on the surfaces under moderate pressure, creating a large contact area. It also maintains good thermal conductivity at

cryogenic temperatures and is widely used for cryogenic applications [45]. It often provides the lowest TCR, especially at higher clamping forces [46].

- Cryogenic greases (e.g: Apiezon): These hydrocarbon based greases easily flow into and fill microscopic gaps improving the overall thermal contact. They are used for sensitive equipement which can not be placed under clamping forces (for example cryogenic temperature sensors) [45]. They are also relatively easy to apply but may outgas in vacuum, and their performance is often inferior to indium at higher pressures or very low temperatures [46].
- Metallic coatings (e.g: Gold): Thin coatings of soft, conductive, and non-oxidizing metals like gold can also be used to reduce the TCR by improving asperity contact and preventing the formation of oxide layers [47].
- Low melting point solders: For permanent or semi-permanent joints, solders like lead-tin (PbSn) or indium-based solders can provide excellent thermal (and electrical) contact [48]. Liquid metal alloys like Galinstan (gallium-indium-tin) are able to take it a step further as they are liquid at room temperature and solidify at cryogenic temperatures creating a demountable interface [49].
- Filled polymers: Composites consisting of a polymer matrix (like epoxy) filled with thermally conductive particles (e.g: metallic powders, ceramics, or advanced fillers like graphene) can also be used as adhesives or gap fillers much like the conductive grease [50]. However, their performance depends heavily on the filler material, its concentration in the polymer, and the way it was manufactured.

2.7. Thermal Management at the SC Cable/Current Lead Junction

This section of the report covers the heat load analysis between the current leads and the superconducting cables, the cooling strategies at the interface to account for the heating, a detailed review of existing designs, and finally, the research gaps and the contributions of the thesis.

2.7.1. Heat load analysis at the interface

The total heat load that needs to be managed at this junction is the sum of the resistive heat load of the current leads, the conduction heat load brought in by the current leads, the contact resistance between the super conducting tapes and the solid former of the superconducting cables, and finally, the contact resistance between the superconducting cables and the heat sink connecting to the current leads. It is important to note that the resistive and conduction heat load dominate the design, which is why the current lead geometry needs to be optimized.

Firstly, the Mcfee formulation is used to find the optimum geometry of the current and hence minimize the heat load onto the heat sink. This formulation is presented in Equation 2.2 where L is the integration length, I is the current carried through the leads, A is the cross section area of the current leads, k is the material thermal conductivity, ρ is the electrical resistivity of the material, T_H is the temperature at the warm end of the current lead, and T_L is the temperature at the cold end of the current lead [51].

$$\left(\frac{IL}{A}\right)_{\text{opt}} = \int_{T_L}^{T_H} \frac{k}{\sqrt{2\int_{T}^{T_H} k\rho\, dT}}\, dT \tag{2.2}$$

This, when performed for different levels of purity of copper for two different warm temperatures of 100 K and 290 K versus the temperature at the cold end, results in Figure 2.11. Figure 2.11 gives a general idea of the dimensions of the current lead for the chosen temperatures on the warm and cold ends in an ideal scenario. Similarly, the minimum heat load onto the cold end of the current lead is calculated using Equation 2.3 [51]. The minimum heat load per current (in W/A) for a copper RRR 100 current lead versus the temperature at the cold end is shown in Figure 2.10.

$$\left(\dot{Q}_L\right)_{\text{min}} = I\sqrt{2\int_{T_L}^{T_H} k\rho\, dT} \tag{2.3}$$

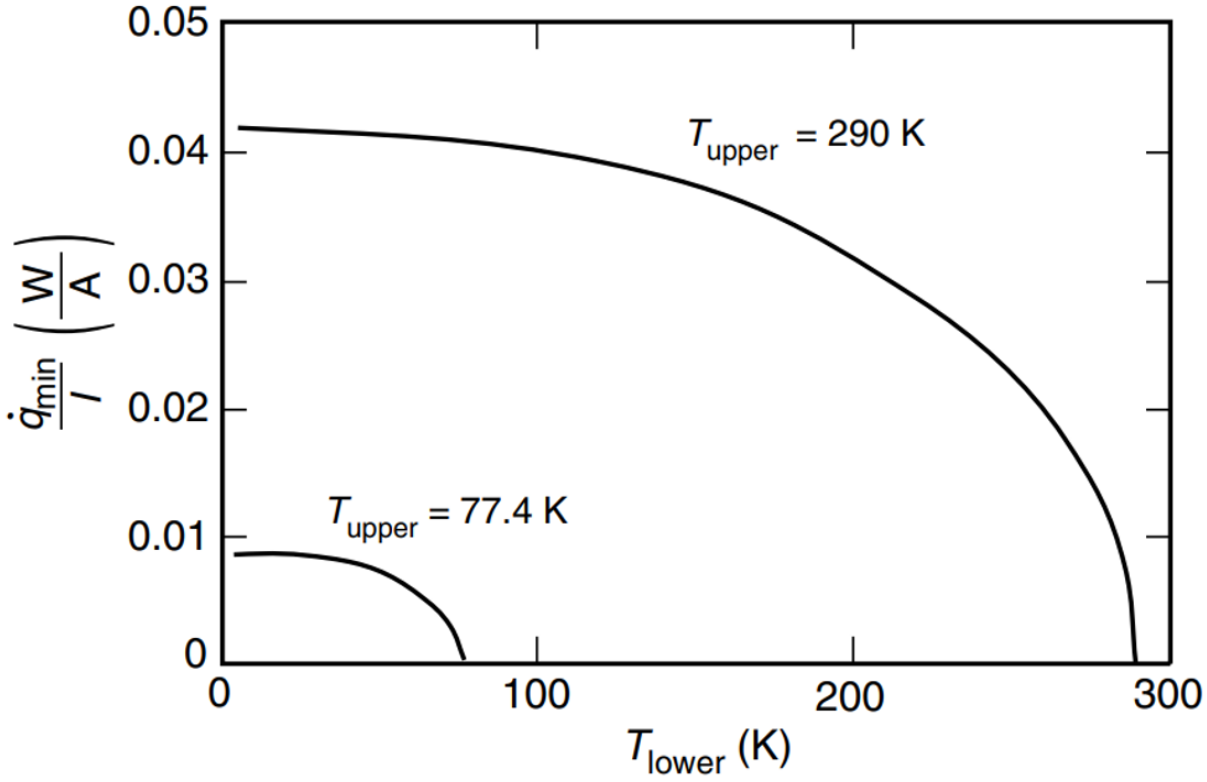


Figure 2.10: Minimum heat load from McFee estimation for Copper RRR 100 at T_H = 77.4 K and 290 K. [52]

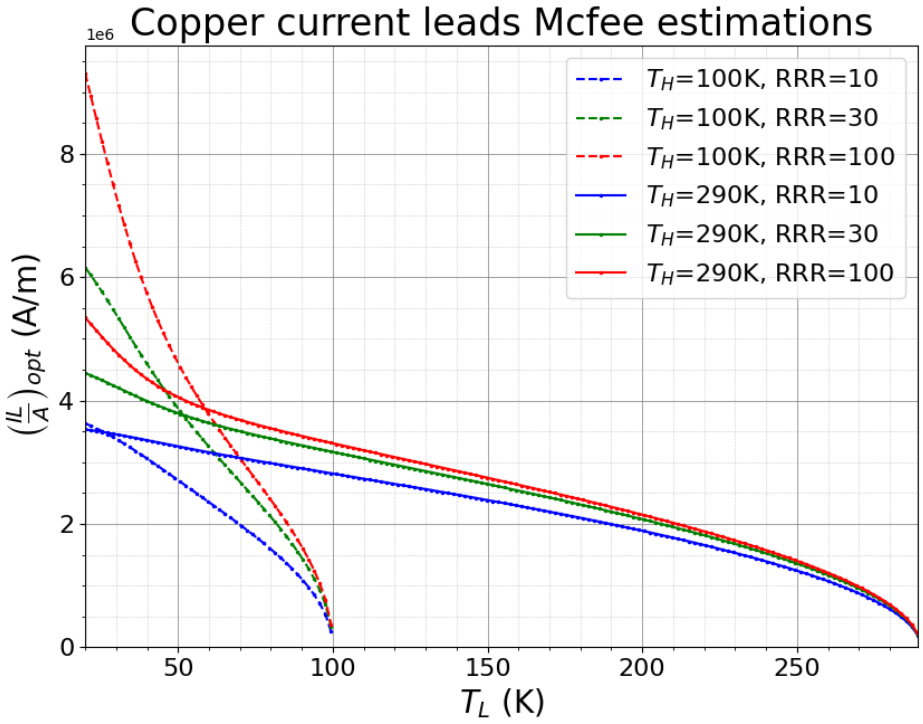


Figure 2.11: McFee estimation for optimum current lead geometry for Copper RRR 10, 30, 100 at T_H = 100 K and 290 K.

2.7.2. Review of existing designs

Several projects and products can help provide examples of thermal management of the current leads. Some of these projects are presented below.

- Fusion magnet leads (ITER, W7-X, JT-60SA): These high-current (10s of kA) leads use a binary design with a Bi-2223 HTS section (e.g: 4.5 K to 65 K) and a GHe cooled copper heat exchanger section (e.g: 65 K to 300 K) [53]. The focus is on high reliability and managing large heat loads in large scale installations.
- Airbus ASCEND demonstrator: This project used a superconducting DC cable (CORC) cooled by LN2 (65-70 K) for a 500 kW aircraft powertrain demonstrator [31]. The connecting devices feature conduction cooled current leads (copper for 295K-68K, flexible braid for 100K-68K) with integrated LN2 heat exchangers and vacuum insulation [10]. The target heat load removal at the 68 K interface for the 295 K lead is 80 W per lead [10].
- Commercial HTS current leads (e.g: HTS-110 CryoSaver): Designed primarily for cryogen-free LTS and HTS magnets (e.g: MRI, NMR), these current leads typically operate between a warm heat sink (64 K provided by a cryocooler) and the cold magnet (4.2 K) [44]. They emphasize the critical need for efficient heat sinking at the 64 K end and recommend sapphire interfaces for optimal thermal contact combined with electrical isolation [44].
- US Navy HTS cable terminations: The US Navy has also focused its R&D efforts on developing compact and lightweight terminations for GHe cooled (50 K) HTS power cables for shipboard applications with a goal of scalable terminations (1 - 4 kA) significantly smaller than traditional designs with the help of McFee optimization principles [54].
- MIT 2-stage current lead system: The 2-stage current lead system at MIT serves as the closest comparison to the present work, as it too makes use of small heat exchangers/ heat sinks to dissipate 80 W with their low-temperature heat exchanger using 40 g/s mass flow of gaseous neon at 3 bara inlet pressure and approximately 50 K inlet temperature, all while causing a pressure drop of almost 500 Pa. At this state, neon has roughly the same density as helium at the inlet conditions of the DC link for the Cryoprop demonstrator(42 K, 15 bara) [42].

2.8. Research gaps and thesis contribution

While considerable work has been done on current leads for various applications, such as nuclear fusion, naval ships, and particle accelerators, a specific gap still exists. This gap corresponds to the thermal management design for the SC cable/current lead junction within the context of a hydrogen-electric aircraft using a separate GHe loop to cool the heat sink. Existing designs are often based on different temperature boundaries (4 K, 77 K), coolants (LHe, LN2, specific GHe loops), or application constraints (large-scale fusion, static lab magnets, power grid). Commercial current leads typically assume specific cryocooler interfaces [44]. Fusion and power grid solutions are generally too large and heavy for aerospace [54]. The ASCEND project from Section 2.7.2 is highly relevant as it is also the predecessor of Cryoprop at Airbus UpNext (the project under which the thesis is taking place) but utilizes LN2 cooling [10].

Therefore, this thesis addresses the need for a dedicated heat sink design optimized to meet the requirements for:

- The specific temperature regime (approx: 40 K cold end, interfacing towards 110 K on the Motor Control Unit (MCU) side or 290 K on the fuel cell side).
- Integration with an LH2/GHe cooling system.
- Strict requirements for weight, volume, pressure drop, and operation under dynamic flight conditions.

The thermal management solution (heat sink) cannot be designed alone. Its performance is linked to the optimized design of the current lead itself (which determines the heat input) and the capabilities of the aircraft's cryogenic cooling system (which determines the heat rejection potential). An effective design requires an integrated approach that considers the heat load of the current lead, the heat transfer characteristics of the sink, and the properties and flow parameters of the GHe coolant. This thesis aims to contribute to this integrated design challenge by focusing on the critical heat sink component at the interface between the current leads and the superconducting cables.

3

Methodology

This chapter discusses the methodology implemented to achieve the desired goal and answer the research questions/objectives of this thesis. Since the research consists of a numerical part and an experimental part, the methodology section is divided into two parts. In particular, Section 3.1 details the methodology for the numerical models and simulations, and Section 3.2 focuses on the experimental setup.

3.1. Numerical methodology

The first step in design is to lay out the requirements and the objective. This also follows for the design of a superconducting termination. First the DC link subsystem requirements pertaining to the heat sink and the current lead set forth by the Cryoprop project at Airbus UpNext are discussed in Section 3.1.1. Then, a 1D PYTHON model was created to provide a preliminary design of the heat sink. This PYTHON model is explained in Section 3.1.2, starting with the heat transfer equations along with the equations used to approximate the pressure drop across the heat sink. Simultaneously, another 1D numerical model was built to size the Current Leads (CL) and estimate the heat load falling on the heat sink. This 1D model for CL is also discussed in Section 3.1.2 along with the heat sink design.

Based on the initial 1D PYTHON model for the heat sink and a preliminary estimate of the heat load from the CL numerical model a basic heat sink geometry was created in CATIA V5. This CAD model was later tested with a Multiphysics model in COMSOL. Here several iterations were performed on the geometry of the heat sink based on the COMSOL simulation results. This iterative approach is discussed in Section 3.1.4 along with the boundary conditions and an overview on the mesh generation for the multiphysics model is provided in Section 3.1.3. Once the final heat sink design was converged upon in the COMSOL simulations and the requirements were met, the focus of the activities was moved to the development of the experimental setup to test the complete superconducting termination (including the Current lead and the heat sink).

3.1.1. Project requirements and constraints

Cryoprop is a project dedicated to demonstrating the performance of a superconducting cryogenic powertrain with aircraft specifications defined by project Zero E [55]. The Cryoprop team is divided into four main teams, namely: DC distribution, Superconducting Motor, MCU & Power electronics, and Cryogenics. To allow for an interfacing between the different teams and parts of the project, each team establishes requirements for their components and, more importantly, downstream of their section. This allows the teams working on components further downstream to assume consistent inlet conditions. Similarly, the entirety of the DC distribution received a certain cooling budget and a pressure drop budget which could not be exceeded. In addition to the project-imposed requirements, the DC link had requirements for its own components. An overview of all the requirements and constraints is provided below.

- **Constraints**
  - Constraint 1: The cooling fluid is Helium.
  - Constraint 2: The inlet fluid temperature is 42 K.
  - Constraint 3: The operating fluid pressure is set at 15 bara.
  - Constraint 4: The maximum available mass flow rate of the fluid is 18 g/s.

  - Constraint 5: The length of the superconducting terminations is set at 10 cm.
- **Requirements**
  - Requirement 1: The pressure drop across the heat sink should be kept below 100 Pa for nominal operation.
  - Requirement 2: The temperature at the HTS termination of the heat sink must not exceed 50 K.
  - Requirement 3: The temperature difference across the superconducting cable termination (HTS termination) must not exceed 1 K.

3.1.2. Preliminary 1-D analytical modeling

To rapidly explore the design space and establish a viable baseline, two independent 1D numerical models were developed in PYTHON. These models provided initial sizing for the heat sink and current leads and quantified the primary heat loads and volume.

Heat sink model

The first 1D model focused on the design of the heat sink. Initially, the model was created for a basic flat plate heat sink as shown in Figure 3.1. However, the heat sink must be installed inside a cylindrical tube. Therefore, the original model was modified to a cylindrical heat sink as seen in Figure 3.1. This heat sink was split into 4 sections, each containing one termination of a superconducting cable for a complete 4-cable monopole design. As the name suggests, in the monopole design, each superconducting cable has one single termination, also seen in Figure 3.1. This is unlike the coaxial design with both a positive and a negative termination on the same superconducting cable as seen in Figure 3.2. In the Cryoprop demonstrator, the monopole cable design will be used.

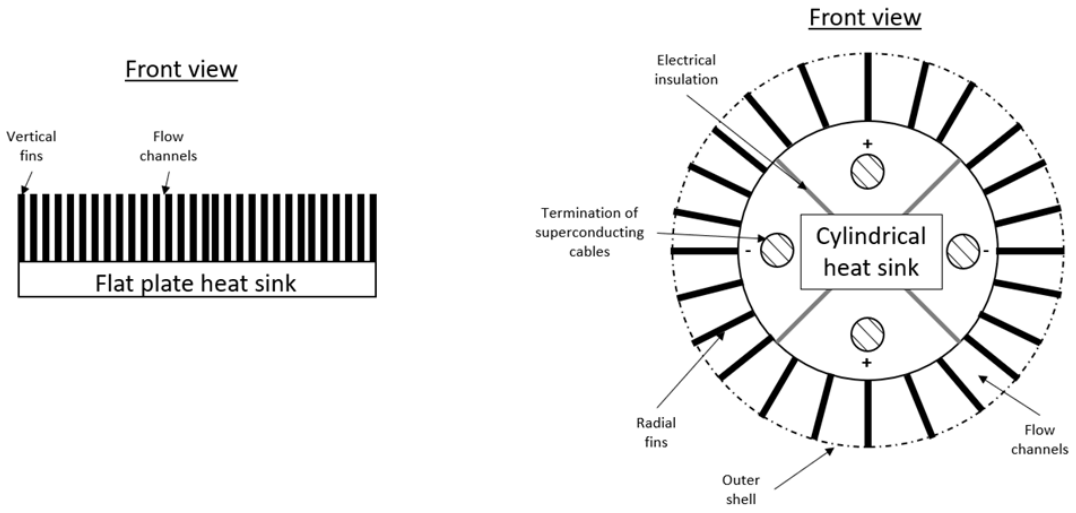


Figure 3.1: A simplified frontal view of a flat plate heat sink and a cylindrical heat sink.

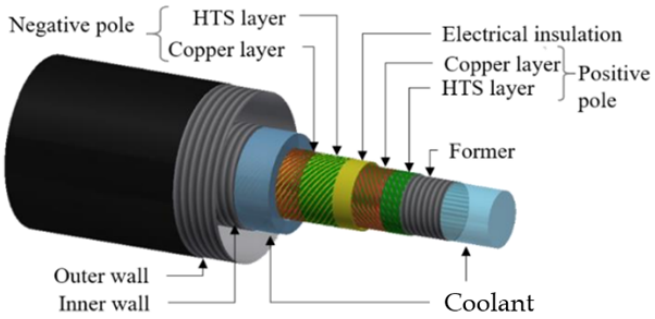


Figure 3.2: Complete view of a superconducting coaxial cable with its different layers. [56]

The base diameter of the cylindrical heat sink was set to be a function of the number of poles (n), the diameter of the terminations of the superconducting cables (D_p), the clearance of these terminations from the circular edge of the heat sink (g), and the minimum spacing between two adjacent poles (c). This is presented in Figure 3.3 along with an overview of all the variables used in the heat sink design in Table 3.1.

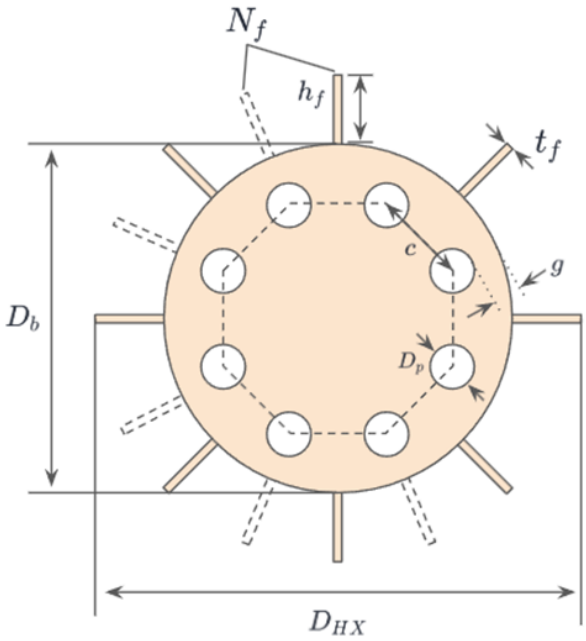


Figure 3.3: Variables of the 1D heat sink model.

Table 3.1: Input variables for the 1D heat sink model.

Input variables	Symbol	Units
Fin thickness	t_f	mm
Number of poles	n	-
Pole diameter	D_p	mm
Length of heat sink	L	mm
Fin height	h_f	mm
Edge clearance	g	mm
Pole spacing	c	mm
Number of fins (range)	i < N_f < j	-
Mass flow rate	$\dot{m}$	kg/s
Inlet temperature	T_{in}	K
Inlet pressure	P_{in}	Bara
Heat load	Q	Watts
Fluid	-	-

The heat sink base diameter is then calculated via Equation 3.1. Here the complete heat sink diameter is calculated by simply adding 2 x fin height to the base diameter and the hydraulic diameter for flow through a single channel is calculated using Equation 3.2 [57].

$$\text{Base diameter } (D_b) = \begin{cases} 2 \cdot \left(\frac{c}{2} + D_p + g\right); & \text{if n = 2} \\ 2 \cdot \left(\frac{(c/2)+(D_p/2)}{cos\left(0.5 \cdot (180 - \frac{360}{n})\right)} + \frac{D_p}{2} + g\right); & \text{if n > 2} \end{cases} \tag{3.1}$$

$$\text{Hydraulic diameter } (D_H) = 4 \cdot \frac{V_{channel}}{A_{wetted}} \tag{3.2}$$

Knowing the hydraulic diameter, some basic calculations can now be made regarding the pressure drop of the flow through the heat sink and the heat transfer between the radial-fin heat sink and helium. These thermo-hydraulic calculations are based on a set of simplifications made for the 1D model. The set of assumptions used in the 1D heat sink model are as follows:

- **Assumption 1**: The flow of helium is distributed evenly throughout all the flow channels at the inlet of the heat sink.
- **Assumption 2**: The temperature of the fins is assumed to be constant along the height of the fin and equal to the base temperature (on the cylindrical surface) of the heat sink (due to the short height of the fins).
- **Assumption 3**: The temperature of the fins is assumed to be constant along the thickness as it is much smaller than the height of the fins.
- **Assumption 4**: No heat dissipation is assumed through the flat ends of the cylindrical heat sink.
- **Assumption 5**: A uniform heat dissipation is assumed through the cylindrical surface of the heat sink.

The fluid properties such as the fluid conductivity, density, enthalpy and viscosity are extracted from CoolProp for a given pressure and temperature [58]. The flow velocity through each channel is calculated via Equation 3.3. Next, the Reynolds number of the flow is calculated through Equation 3.4 using the previously calculated hydraulic diameter.

$$\text{velocity } (u) = \frac{\dot{m}}{\rho_{fluid}} \cdot \frac{1}{A_{\text{cross-section area/channel}}} \tag{3.3}$$

$$\text{Re} = \frac{4 \cdot \dot{m}}{\pi \cdot \mu \cdot D_H} \tag{3.4}$$

The Reynolds number dictates whether the flow is of a turbulent or laminar nature. The flow regime then dictates which equations are to be used to estimate the pressure drop. For tubulent flow, the pressure drop equation is given in Equation 3.5 where f is the Darcy-Weisbach friction factor and L is the length of the flow segment [59]. Since the equation for the Darcy-Weisbachch friction factor is an implicit equation, as shown in Equation 3.6, it requires an initial guess to iterate over. This initial guess is provided through the Haaland equation given below in Equation 3.7 where the surface roughness is set as 30μm [60].

$$\Delta P = f \cdot \frac{L}{D_H} \cdot \left(\frac{\rho \cdot u^2}{2} \right) \tag{3.5}$$

$$\frac{1}{\sqrt{f_{\text{Darcy–Weisbach}}}} = -2 \cdot log_{10} \left(\frac{\epsilon}{3.7 \cdot D_H} + \frac{2.51}{Re \cdot \sqrt{f_{\text{Darcy–Weisbach}}}} \right) \tag{3.6}$$

$$\frac{1}{\sqrt{f_{\text{Haaland}}}} = -1.8 \cdot log_{10} \left(\left\{ \frac{\epsilon/D_H}{3.7} \right\}^{1.11} + \frac{6.9}{Re} \right) \tag{3.7}$$

As the system is under forced flow of supercritical helium, its is assumed that the conductive resistance is negligible as compared to the heat convection between helium and the wetted surfaces of the heat sink. This assumption is further reinforced with the calculation of the Biot number in Equation 3.8. If the Biot number is less than 0.1, one can assume that the internal resistance of the solid is negligible compared to the external convective resistance. For a base diameter of 37.5 mm, a total of 50 fins or more, a mass flow rate of 5 g/s or more, the convective heat transfer coefficient is of the order of 600 - 800 W/m^2K. Furthermore, the thermal conductivity of copper (RRR 30) at 42 K is 871 $W/m \cdot K$ [35]. Assuming a fin height (h_f) of 10 mm, it can be seen that the Biot number is indeed very small.

$$Bi = \frac{h_c}{k_{solid}} \cdot h_f \quad \rightarrow \quad \frac{600\ [W/m^2K]}{871\ [W/m \cdot K]} \cdot 0.01\ [m] \ << 1 \tag{3.8}$$

The convective heat transfer coefficient is calculated via Equation 3.9. Here the nusselt number (Nu) is a function of the friction factor, the reynolds number, and the prandtl number (Pr). It is derived through the Gnielinski correlation in Equation 3.10 [61]. The prandtl number is simply taken from the CoolProp database for the given fluid temperature and pressure.

$$h_c = \frac{k_{fluid} \cdot Nu}{D_H} \tag{3.9}$$

$$Nu_D = \frac{(f/8)(Re_D - 1000) \cdot Pr}{1 + 12.7(f/8)^{1/2}\left(Pr^{2/3} - 1\right)} \tag{3.10}$$

A user specified heat load is imposed onto the heat sink. For the heat sink to operate in steady state condition, the constant heat load is absorbed by the constant flow of helium. The user specifies an inlet fluid temperature and pressure, as a result the inlet fluid enthalpy, and the outlet fluid temperature is calculated using Equation 3.11 where the enthalphy of the fluid increases as it absorbs all the heat load. Knowing the fluid enthaply at the outlet, and the pressure drop across the heat sink, one can get the outlet fluid temperature with the help of CoolProp.

$$H_{\text{fluid outlet}} = \frac{Q_{\text{per channel}}}{\dot{m}_{\text{per channel}}} + H_{\text{fluid inlet}} \tag{3.11}$$

Next, the mean wall temperature is calculated via the traditional heat transfer equation below in Equation 3.12. This is rearranged to get the mean temperature of the wall along the length of the heat sink for the mean fluid temperature along the heat sink as seen in Equation 3.13.

$$Q = h_c \cdot A_{\text{wetted}} \cdot \Delta T \tag{3.12}$$

$$T_{\text{mean, wall}} = \frac{Q}{h_c \cdot A_{\text{wetted}}} + T_{\text{mean, fluid}} \tag{3.13}$$

Instead of performing a bulk calculation, the heat sink is discretized along its length into 100 cells, each of which perform the aforementioned calculations to get the fluid and wall temperatures for each cell. The outlet of the 1st cell becomes the inlet of the 2nd cell and so on. This is represented graphically in Figure 3.4 for a clearer understanding.

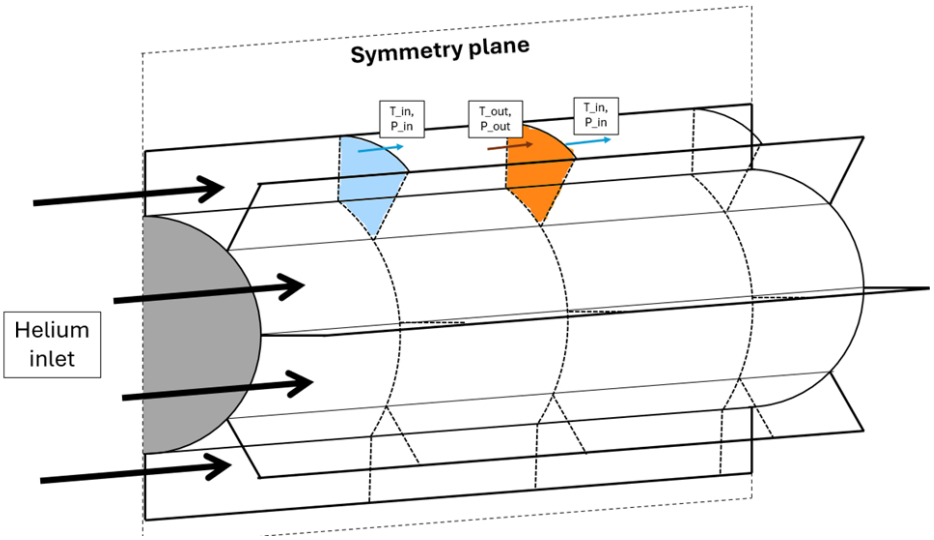


Figure 3.4: A simplified representation of the discretized 1D Python model.

Current lead model

Along with the 1D model for the heat sink, a separate one-dimensional model was created for sizing of the current lead. This is important as the current lead (CL) sizing code outputs the heat load imposed onto the system for a specified combination of the applied current, length to area ratio, and the temperatures on both ends of the CL. This heat load is then fed back into the heat sink model to get a better approximation of its size and geometry.

The numerical current lead model is based on the research conducted by Ho-Myung Chang & Min Jee Kim on the optimization of conduction-cooled current leads [51]. Three main operating conditions for a current lead are highlighted by Chang and Kim, namely the optimal operation, the under-current operation, and the over-current operation. A simplified representation of the temperature profile along the current lead in all the operation modes is presented in Figure 3.5 where T_p is the peak temperature on the current lead and L_p is the length along the current lead where it occurs.

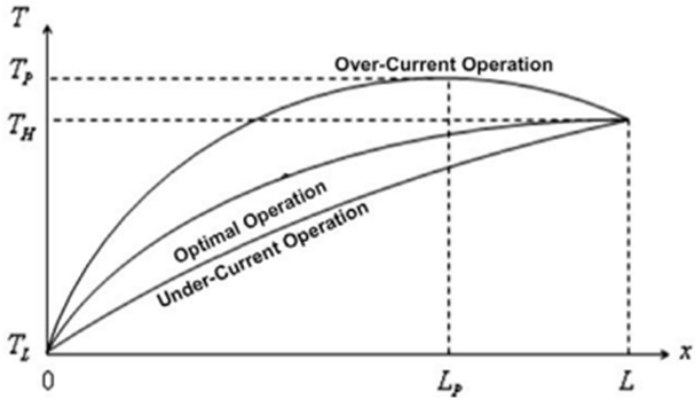


Figure 3.5: Temperature profiles of conduction-cooled current lead.[51]

Under optimal operating conditions the heat load due to electrical resistance is equal to the heat load due to pure conduction from the hot end to the cold end. An energy balance for a small control volume of length dx of the current lead as shown in Figure 3.6 is given by Equation 3.14 where $\dot{Q}$ is the rate of heat transfer to the cold end at T_L temperature, rho is the electrical resistivity of the material at the given temperature, A is the crossectional area of the current lead, and I is the current flowing through the conductor.

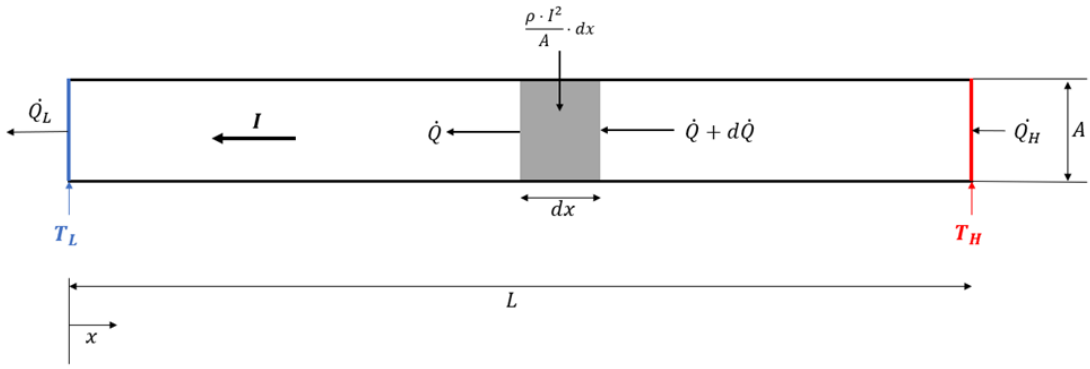


Figure 3.6: Simplified 1D representation of a conduction-cooled current lead.

$$d\dot{Q} = -\frac{\rho \cdot I^2}{A} \cdot dx \tag{3.14}$$

Combining the fouriers conduction law $Q_{dot} \cdot dx = k \cdot A \cdot dT$ with Equation 3.14 results in Equation 3.15 where k is the electrical conductivity and ρ is the elctrical resistivity of the material. To get the minimum heat load on the cold end of the current lead, Equation 3.15 can be integrated from T_L (temperature on the cold end) to T_H (temperature on the hot end) and the $\dot{Q}_H$ is set to zero as seen in Equation 2.3. This is the case when the T_H is set to be the same as the ambient temperature on the hot end side.

$$\dot{Q} \cdot d\dot{Q} = -k \cdot \rho \cdot I^2 \cdot dT \tag{3.15}$$

As mentioned before in Section 2.7.1, a specific length over area (L/A) ratio exists for the optimal operation and can be determined by integrating the Fourier's law from (x,T) = (0,T_L) to (L,T_H). The $L/A_{optimum}$ then results in Equation 2.2.

In under-current operation the current passing through the current lead is lower than what the current lead was sized for as shown by Equation 3.16. Here instead of assuming a $\dot{Q}_H$ = 0, the integration is done from T_L to an arbitrary temperature T as seen in Equation 3.17. Rearranging this to get an expression for $\dot{Q}(T)$ results in Equation 3.18.

$$I < I_{\text{opt}} = \frac{A}{L} \int_{T_L}^{T_H} \frac{k}{\sqrt{2 \int_{T_L}^{T_H} k\rho \, dT}} \, dT \tag{3.16}$$

$$\int_{\dot{Q}_L}^{\dot{Q}} \dot{Q} \, d\dot{Q} = \int_{T_L}^{T} -k \, \rho \, I^2 \, dT \tag{3.17}$$

$$\dot{Q} = \sqrt{\dot{Q}_L^2 - 2I^2 \int_{T_L}^{T} k \cdot \rho \cdot dT} \tag{3.18}$$

Now the fourier's law can be integrated again from (0,T_L) to (L,T_H). The $\dot{Q}$ is now substituted in from Equation 3.18 into Equation 3.19 to get the L/A ratio for under-current operation at constant current in Equation 3.20.

$$\int_0^L \left(\frac{dx}{A}\right) = \int_{T_L}^{T_H} \left(\frac{k}{\dot{Q}} \cdot dT\right) \tag{3.19}$$

$$\int_0^L \left(\frac{dx}{A}\right) = \int_{T_L}^{T_H} \left(\frac{k}{\dot{Q}} \cdot dT\right) \quad \rightarrow \quad \frac{L}{A} = \int_{T_L}^{T_H} \left(\frac{k}{\sqrt{\dot{Q}_L^2 - 2 \cdot I^2 \cdot \int_{T_L}^{T} (k \cdot \rho \cdot dT)}}\right) \cdot dT \tag{3.20}$$

Finally, for over-current operation the applied current exceeds the optimum current that the current lead is sized. Therefore a temperature overshoot occurs in the current lead reaching a peak temperature T_p at length L_p as shown in Figure 3.5. Since T_p is assumed to be the hottest point in the current lead, the heat ingress at that point (x = L_p) is equal to 0. Now Equation 3.15 is once again integrated from T_L to T_p where the $\dot{Q} = 0$ at $x = L_p$ to get Equation 3.21.

$$\frac{1}{2}\dot{Q}_L^2 = I^2 \int_{T_L}^{T_p} \rho \cdot k \cdot dT \tag{3.21}$$

In order to get the expression for $\dot{Q}$ before and after $x = L_p$, Equation 3.15 is integrated from T_L to an arbitrary T for $0 < x < L_p$ and from T_p to arbitrary T for $L_p < x < L$. This yields a discontinuous function shown in Equation 3.22. Following the same process as for under-current, the fourier's law is integrated for the bounds given in Equation 3.22 to get the expression for L/A in over-current operation shown in Equation 3.23.

$$\dot{Q} = \begin{cases} \sqrt{\dot{Q}_L^2 - 2I^2 \int_{T_L}^{T} k \cdot \rho \cdot dT} & \text{for } 0 \leq x \leq L_p \\ -\sqrt{2I^2 \int_{T}^{T_p} k \cdot \rho \cdot dT} & \text{for } L_p \leq x \leq L \end{cases} \tag{3.22}$$

$$\int_0^{L_p} \left(\frac{dx}{A}\right) + \int_{L_p}^{L} \left(\frac{dx}{A}\right) = \int_{T_L}^{T_p} \left(\frac{k}{\dot{Q}_{(0<x<L_p)}} \cdot dT\right) + \int_{T_p}^{T_H} \left(\frac{k}{\dot{Q}_{(L_p<x<L)}} \cdot dT\right) \tag{3.23}$$

3.1.3. Mesh generation: COMSOL

In COMSOL Multiphysics, as in all FEM software, the mesh serves two main functions. First, it provides a piecewise approximation of the CAD geometry. If a mesh fails to accurately capture critical geometric features, such as small fillets, sharp corners, or thin layers, it is often unable to give a reliable physical solution. Second, the mesh discretizes the solution field itself. It defines a set of discrete points in space, also known as nodes, at which the approximate solution is calculated. The solution is then interpolated across the volume of each element using polynomial basis functions.

The density of the cells is another factor to consider for the accuracy of the mesh. The number and size of the mesh's elements directly determine the number of nodes and the number of Degrees of Freedom (DOF) in the model [62]. As the elements are made smaller and increased in quantity, the piecewise polynomial approximation becomes better able to capture complex variations in the solution field. This allows the computed solution to get closer to the analytical solution [62]. However, this comes at a significant computational price and time. To assist with this the software has a set of tools to mesh the geometry all the way from fully automated procedures to manual manipulation of the mesh.

Physics controlled vs User controlled

COMSOL has two main ways for mesh generation, a physics controlled mesh and a user controlled mesh. The physics controlled mesh is the default way. This is completely automatic where the software generates a mesh best suited to the requirements of the selected physics interfaces [63]. For instance, when modeling a fluid flow problem, the physics controlled setting will automatically create a finer mesh that includes boundary layer elements along the no-slip walls to accurately capture the high velocity gradients in these regions [64]. This method provides a good starting point, making it particularly valuable for initial analyses and standard problems. But as the geometry starts becoming more and more complex, it is better to switch to a user controlled mesh.

The user controlled mesh allows the user complete control over every aspect of the meshing process. It allows for the precise use of different meshing operations, element types, and size controls to more complex geometries. A user controlled mesh is constructed through a Meshing Sequence, which is a list of operations executed sequentially as they appear in the Model Builder [65]. The order of this process is important as when a meshing operation is applied to a geometry, the resulting mesh on that CAD model becomes a fixed boundary condition for any new meshing on adjacent parts [66]. This sequential dependency often leads to low quality elements at the interfaces between different meshed regions. For example, if a domain with a coarse mesh is meshed first, its boundary with an adjacent, unmeshed domain is fixed with large elements. If the next mesh then attempts to apply a fine mesh to the adjacent domain, it is forced to generate highly skewed and poor quality elements to transition from the fine interior mesh to the existing coarse boundary mesh [66]. An example of this can be seen in Figure 3.7 and Figure 3.8. In Figure 3.7 the solid domain was meshed first which forced the fluid side facing the solid domain to have a coarser mesh. To mitigate this, the recommended practice is to first mesh the domains where a finer element size and then move onto more coarse mesh parts as seen in Figure 3.8.

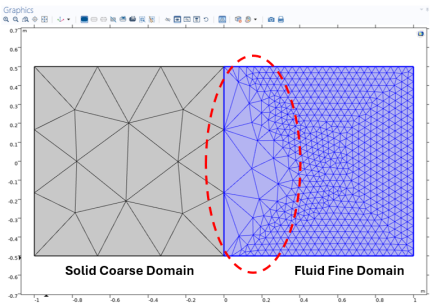


Figure 3.7: Meshing of the coarse domain before the fine domain.

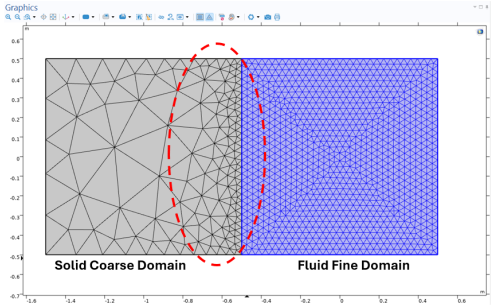


Figure 3.8: Meshing of the fine domain before the coarse domain.

Types of mesh elements

The choice of element type is another important decision that influences the meshing of a geometry. It also affects the accuracy of the solution and the overall computational efficiency. COMSOL Multiphysics has a range of element types, each with its own advantages and applications as listed below.

- Tetrahedra: These cells have four nodes and four faces and are the default element type in COMSOL. Their main advantage is their geometric flexibility. They are able to mesh any 3D volume regardless of its shape. This is done using an unstructured tetrahedral mesh generator [66]. They are also the only element type that support solver based adaptive mesh refinement [67][68] [69]. However, a drawback of the Tetrahedra elements is that the linear (first-order) tetrahedral elements can have excessive stiffness in structural mechanics simulations. They also require a higher number of elements to achieve the same level of accuracy as other element types. This then leads to a higher meshing and computing time [70][68] [69].
- Hexahedra: Hexahedra elements have eight nodes and six faces. They are used for their efficiency and accuracy. For a given number of nodes, a hexahedral mesh generally gives more accurate results than a tetrahedral mesh. In structural mechanics and in computational fluid dynamics (mainly in the boundary layers of the fluid), a hexahedral mesh is likely to give better results than the default Tetrahedra mesh [70] [68] [69]. Hexahedra elements can also be stretched to high aspect ratios (the ratio of the longest edge to the shortest edge) without a huge degradation in quality. This makes them ideal for meshing thin structures or regions where the gradient of a solution is anisotropic [67] [68] [69]. The main limitation of hexahedra elements is the lack of a fully automatic, ”free” meshing algorithm. They can only be generated using manually structured meshing techniques like Swept or Mapped operations. Such operations are only applicable to geometries with a regular and simple shapes (constant cross section extruded parts). This is also the element chosen for the extruded fins of the heat sink and the fluid volume within the simulation.
- Prisms: This is a six noded element composed of two triangular and three quadrilateral faces. They act as links between triangular and quadrilateral surface meshes and a structured volume mesh. They are mainly used in generating Boundary Layer meshes, where they are grown from a triangular surface mesh, and in creating Swept meshes that are extruded from a triangular face [71].
- Pyramids: The final element type is the pyramid elements. They have five nodes with a quadrilateral base and four triangular faces. Their sole function within COMSOL is to act as transition elements by connecting a region of hexahedral elements to a region of tetrahedral elements. They are mainly used for creating hybrid meshes.

Mesh quality assessment

The quality of the mesh elements has a direct impact on the stability and accuracy of the numerical solution. A mesh containing highly distorted or degenerated elements can lead to solver convergence failures or produce inaccurate results [72]. Hence why the mesh on the CAD model was evaluated on two different criteria: the **skewness** and **volume vs length ratio**.

Skewness is the measure of deviation from an ideal, equiangular element shape. A mesh quality of 1 (normalized) in terms of skewness is the best possible in COMSOL. In practice, a skewness above 0.9 is

often considered to be good while a skewness above 0.5 is acceptable. Figure 3.9 and Figure 3.10 show that the average skewness of the mesh for the heat sink is around 0.7 - 0.8 which is deemed acceptable. Although some elements are below a skewness of 0.5, they lie in the core of the metal and not the fins and therefore have less of an impact on the heat transfer between the helium and the fins.

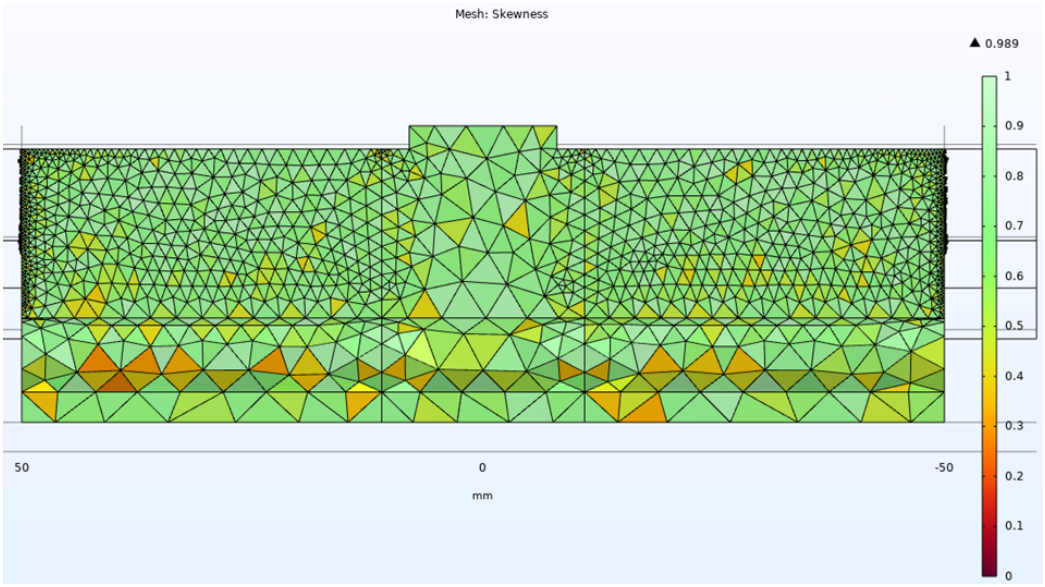


Figure 3.9: Mesh element skewness on the side cut view of the heat sink.

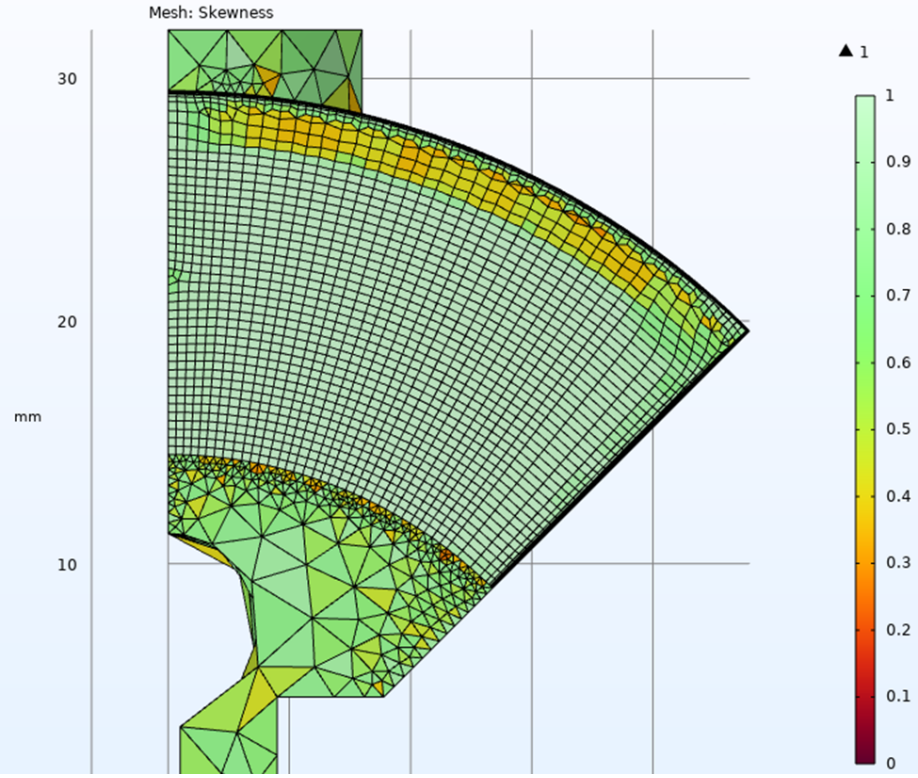


Figure 3.10: Mesh element skewness on the front view of the heat sink.

The second criteria for the mesh quality assessment is the volume to length ratio of the mesh cells. This metric relates the element's volume to the length of its longest edge. It is an effective tool for identifying

elements that are stretched or compressed in one direction. Any normalized Volume vs Length value lower than 0.1 is said to be poor from a meshing point of view. COMSOL aims to keep this value above 0.4 in its auto generated meshes [73]. Figure 3.11 shows that the heat sink satisfies the Volume vs Length criteria where almost all the elements are above 0.4. However, the fluid volume which can be seen in the front view (at the inlet of the heat sink) in Figure 3.12 shows that the Volume vs Length value is between 0.1-0.4 for the fluid elements. This does not qualify as a good mesh but is acceptable for the simulation as no element falls below the value of 0.1. Increasing the Volume vs Length value for the fluid volume is possible by increasing the number of cells and decreasing the length of each cell. This however comes at the price of a higher computational time and therefore was not done.

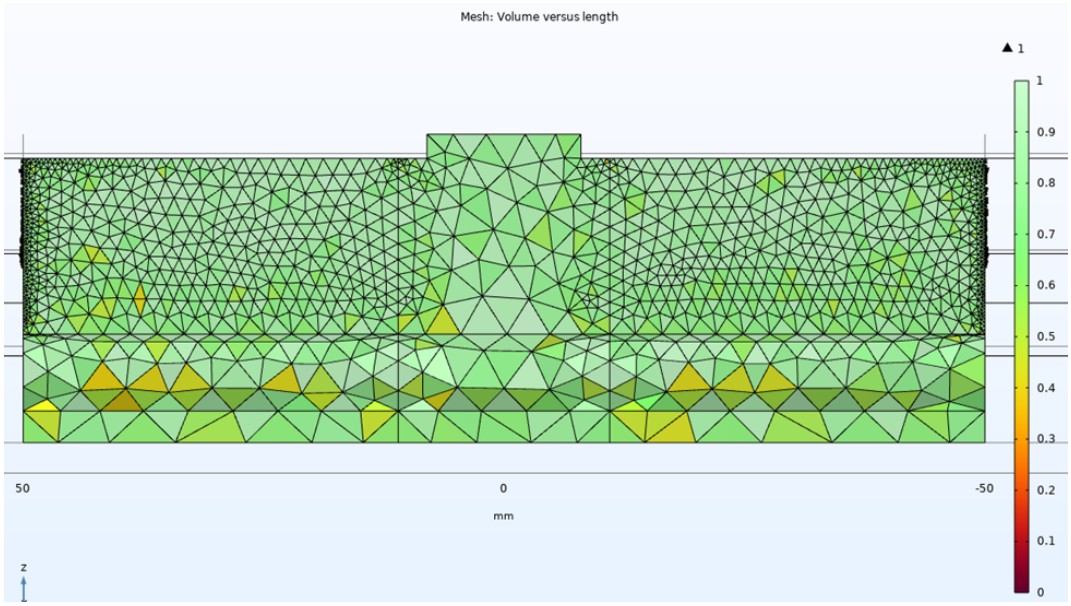


Figure 3.11: Mesh element volume/length ratio on the side cut view of the heat sink.

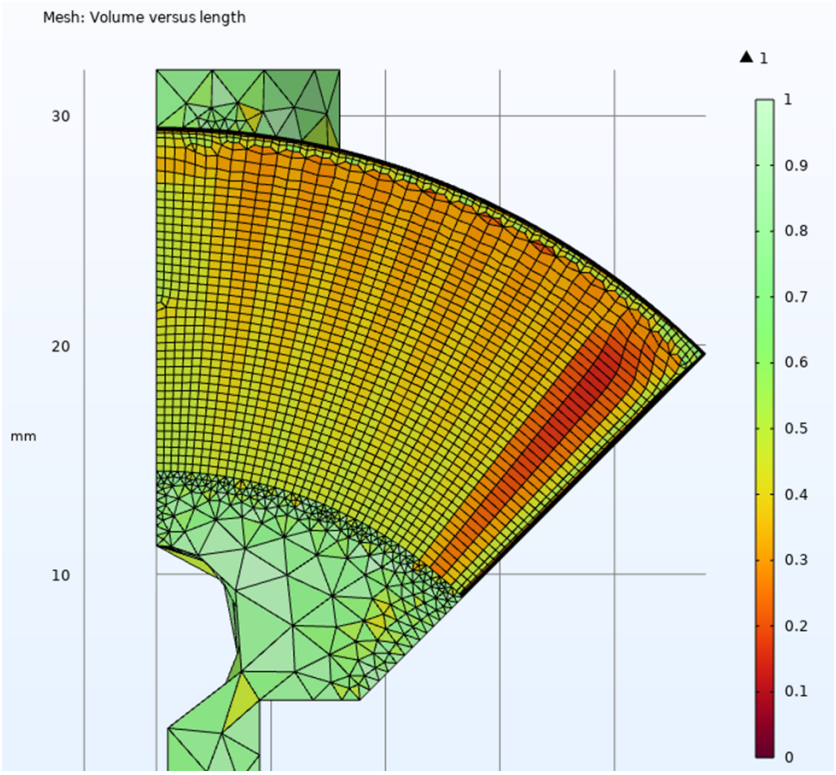


Figure 3.12: Mesh element volume/length ratio on the front view of the heat sink.

3.1.4. Boundary conditions

Once an acceptable mesh has been created for the CAD model, the next step in any simulation is to set the boundary conditions. This section provides a detailed description of the boundary conditions and the underlying physical models used in the simulation. It begins by establishing the governing equations and the multiphysics framework used to couple them. It then details the constitutive models and material property data which are fundamental to the accuracy of the simulation. Finally, it presents an overview of the specific boundary and initial conditions for the superconducting termination model applied to each physical domain, providing a physics-based reason for every modeling decision. The physics interfaces involved in this work are the Electric Currents, Heat Transfer in Solids and Fluids, and Turbulent Flow interfaces. These interfaces are coupled together to simulate the complete electro-thermal-fluidic behavior of the system [74].

Governing physics & Multiphysics framework

The simulation is constructed upon three fundamental physics interfaces. These interfaces are linked with each other through predefined multiphysics couplings to represent the interactions between the fluid and the solid domains. These are the electromagnetic, fluid dynamics, and the heat transfer interfaces.

In the electromagnetic interface the distribution of electrical potential and current within the heat sink and attached current lead is modeled using the electric currents interface. This interface solves the charge conservation equation, a derivative of Maxwell's equations for stationary currents as shown in Equation 3.24. Here, J is the current density vector. The current density is related to the electric field E and the material's electrical conductivity(σ) through Ohm's law, $J = \sigma \cdot E$. The electric field is defined as the negative gradient of the electric potential(V) such that $E = -V$. The interface solves for the potential field V throughout the solid domains.

$$\nabla \cdot J = 0 \tag{3.24}$$

For the fluid dynamics interface, the fluid flow of the supercritical helium coolant is modeled using the 'Turbulent Flow: (k-ε)' interface. This k-ε model is one of the most widely used turbulence models

for industrial applications due to its consistency and reasonable accuracy for a wide range of flows, and was therefore also used for this simulation. This interface is used for single-phase flows at high Reynolds numbers and solves the Reynolds-Averaged Navier-Stokes (RANS) equations for conservation of momentum, along with the continuity equation for conservation of mass. For a steady, incompressible (or mildly compressible) fluid, these equations are shown in Equation 3.25 and Equation 3.26.

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} = \nabla\cdot[-p\mathbf{I} + \mu(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)] + \mathbf{F} \tag{3.25}$$

$$\rho\nabla\cdot\mathbf{u} = 0 \tag{3.26}$$

Here, ρ is the fluid density, u is the velocity vector, p is the pressure, I is the identity matrix, μ is the dynamic viscosity, and F is a body force term (e.g., gravity), which is neglected in this model. When the flow becomes turbulent, all the quantities fluctuate in space and time. The Reynolds-averaged representation of turbulent flows divides the flow quantities into an averaged value and a fluctuating part as seen in Equation 3.27 and Equation 3.28 where U is the averaged velocity field, u' is the fluctuation term, and $\otimes$ is the outer vector product [75].

$$\rho\frac{\partial \mathbf{U}}{\partial t} + \rho\mathbf{U}\cdot\nabla\mathbf{U} + \nabla\cdot\overline{(\rho\mathbf{u}'\otimes\mathbf{u}')} = -\nabla P + \nabla\cdot\mu(\nabla\mathbf{U} + (\nabla\mathbf{U})^T) + \mathbf{F} \tag{3.27}$$

$$\rho\nabla\cdot\mathbf{U} = 0 \tag{3.28}$$

The K-epsilon model, which is used for the fluid domain, is a two-equation model, meaning it solves two additional transport equations to capture the turbulent properties of the flow. The model introduces the following turbulent quantities: Turbulent kinetic energy (k) and Turbulent dissipation rate (ϵ). The Turbulent kinetic energy represents the mean kinetic energy per unit mass of the turbulent fluctuations in the flow. It is a measure of the intensity of turbulence. The Turbulent dissipation rate signifies the rate at which the turbulent kinetic energy is converted into thermal energy due to viscous forces.

By solving transport equations for both k and ϵ, the model can determine the turbulent viscosity. This is an important parameter for modeling the Reynolds stresses. The transport equation for Turbulent Kinetic Energy is given by Equation 3.29 and the transport equation for Turbulent Dissipation Rate is given by Equation 3.30.

$$\frac{\partial(\rho k)}{\partial t} + \frac{\partial(\rho k u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_k}\right)\frac{\partial k}{\partial x_j}\right] + P_k - \rho\epsilon \tag{3.29}$$

$$\frac{\partial(\rho \epsilon)}{\partial t} + \frac{\partial(\rho \epsilon u_i)}{\partial x_i} = \frac{\partial}{\partial x_j}\left[\left(\mu + \frac{\mu_t}{\sigma_\epsilon}\right)\frac{\partial \epsilon}{\partial x_j}\right] + C_{1\epsilon}\frac{\epsilon}{k}P_k - C_{2\epsilon}\rho\frac{\epsilon^2}{k} \tag{3.30}$$

The terms on the left hand side of Equation 3.29 and Equation 3.30 represent the rate of change of k and ϵ over time and their transport by the mean flow (convection) respectively. The first term on the right hand side for both equations is the diffusion term and describes the transport of k and ϵ due to turbulent diffusion. P_k accounts for the generation of turbulent kinetic energy from the mean flow due to velocity gradients. The dissipation of turbulent kinetic energy is shown by $\rho\cdot\epsilon$. The equations include several constants ($C_{1\epsilon}$, $C_{2\epsilon}$, σ_k, σ_ϵ) that are determined empirically from experimental data for a wide range of turbulent flows. Finally, the turbulent viscosity (μ_t) is calculated using the previously calculated k and ϵ via Equation 3.31.

$$\mu_t = \rho\cdot C_\mu\cdot\frac{k^2}{\epsilon} \tag{3.31}$$

The flow in the region near the walls is solved using wall functions, which bridge the viscous sublayer and the fully turbulent region.

The final interface that is used in the simulation is the heat transfer interface. The temperature distribution throughout all solid and fluid domains is governed by the heat equation, which is solved using the Heat Transfer in Solids and Fluids interface [76]. This interface accounts for heat transfer by conduction and convection. The general form of the steady-state heat equation solved is presented in Equation 3.32.

$$\rho \cdot c_p \cdot \mathbf{u} \cdot \nabla T + \nabla \cdot \mathbf{q} = Q \tag{3.32}$$

Here, c_p is the specific heat at constant pressure, T is the temperature, **q** is the heat flux vector, and Q is a volumetric heat source. The heat flux q is defined by Fourier's law of heat conduction, $q = -k\nabla T$, where k is the thermal conductivity of the material. In the solid domains, the convective term ($\rho \cdot c_p \cdot \mathbf{u} \cdot \nabla T$) is zero, and heat transfer is purely by conduction. In the fluid domain, both conduction and convection are active, with the velocity field u provided by the fluid dynamics interface.

Multiphysics couplings

The accuracy of the simulation relies on correctly solving for the two-way interactions between the constituent physics. This is achieved through specific multiphysics coupling features.

The Electromagnetic Heating (Joules Heating) multiphysics coupling provides a link between the Electric Currents and Heat Transfer interfaces [74]. The primary effect of current is the generation of heat through the electrical resistance in the solid material. The electrical solution for current density J and electric field E is used to define a volumetric heat source, Q, in the thermal model. This source term is $Q = J \cdot E$. Using Ohm's law, this can be expressed in terms of the current density and electrical conductivity as shown by Equation 3.33.

$$Q = \frac{|\mathbf{J}|^2}{\sigma} \tag{3.33}$$

This term is automatically added to the heat equation in all domains where the electric current interface is active. The Electromagnetic Heating coupling also contains a feedback mechanism from the thermal to the electrical interface. The electrical conductivity, σ, of the copper conductors is a function of temperature. Therefore, the coupling ensures that the local temperature, T, calculated by the heat transfer interface is used to calculate the appropriate electrical conductivity, $\sigma(T)$. Any significant temperature gradients along the conductor will therefore alter the local resistivity, which can in turn, redistribute the current density [77].

The other important multiphycis coupling is the Nonisothermal flow. This links the Turbulent Flow, k-ε and Heat Transfer interfaces to model the conjugate heat transfer between the solid heat sink and the supercritical helium coolant. The velocity field, **u**, calculated by the fluid dynamics interface is used in the heat transfer interface to compute the convective heat transport. This is represented by the convective term, $\rho C_p \mathbf{u} \cdot \nabla T$, in the heat equation for the fluid domain. This term describes the transport of thermal energy by the bulk motion of the fluid and is the main mechanism responsible for heat removal from the heat sink. The Nonisothermal flow coupling also ensures that the local temperature and pressure values are continuously used to update these fluid properties throughout the domain. As the helium heats up along the flow path, its density and viscosity change, which directly impacts the momentum balance, altering the velocity profile and the overall pressure drop.

Material properties

The simulation is highly dependent on the accuracy of the material property data. Since the local heat transfer rates and pressure drops are calculated directly from first principles, any inaccuracies in the material properties will propagate directly into the final results. Therefore, dedicated databases were used to gather the material properties.

The operating conditions (inlet at 15 bara and 42 K) place the helium coolant in the supercritical fluid region. In this thermodynamic state, its properties exhibit non-linear dependence on both temperature and pressure. The use of simplified models, such as the ideal gas law or constant property assumptions, could lead to errors in the calculation of density, heat capacity, and other transport properties. Therefore, the thermophysical properties for helium were sourced from the NIST REFPROP and Coolprop database [78] [79]. Within the COMSOL model, the material properties for helium—density (ρ), isobaric specific heat (c_p),

thermal conductivity (k), and dynamic viscosity (μ) are implemented as functions of the local temperature (T) and pressure (p). The properties of helium at the specified inlet condition are summarized in Table Table 3.2.

Table 3.2: Thermophysical properties of supercritical helium at inlet conditions (15 bara, 42 K) [79].

Property	Symbol	Value	Units
Density	ρ	16.63	kg/m^3
Isobaric Specific Heat Capacity	c_p	5345	$J/(kg \cdot K)$
Thermal Conductivity	k	0.0443	$W/(m \cdot K)$
Dynamic Viscosity	μ	$6.01 \cdot 10^{-6}$	$Pa \cdot s$

For the copper heat sink, it is important to note that the thermal and electrical performance of the solid conductors at cryogenic temperatures is highly influenced by their purity. This effect is captured through the use of temperature dependent material properties that are parameterized by the material's Residual Resistivity Ratio (RRR). The RRR is defined as the ratio of the material's electrical resistivity at room temperature (typically 295 K or 300 K) to its residual resistivity at liquid helium temperature (typically 4.2 K) [80]. At cryogenic temperatures, the scattering of conduction electrons, which governs both electrical and thermal resistance, is dominated by interactions with impurities and lattice defects rather than by thermal vibrations (phonons) [81] [82]. Consequently, a higher RRR value corresponds to a purer material with lower electrical resistivity and higher thermal conductivity at low temperatures [80]. For the multiphysics model, the precise material properties for a chosen RRR of Copper were extracted from CryoComp [35]. Another important note to make about the solid domain is for the current lead and the attached busbar. For both the current lead and the busbar which may not be protected by MLI (multi layered insulation) inside the cryostat radiation losses are also another source of heating. The most significant contributor to the radiation losses is the temperature difference, however, the second most important contribution is from the emissivity of the material. Emissivity of a metal varies with temperature. This variation is even more pronounced for an object like the Current lead, which has a large temperature gradient across it. As not much data is available on the emissivity of pure copper at cryogenic temperature, the values for the emissivity of copper with respect to temperature were sourced from multiple different sources and averaged out into blue dots in the curve shown in Figure 3.13 [83] [84] [85]. A linear interpolation was then performed for areas between the blue dots.

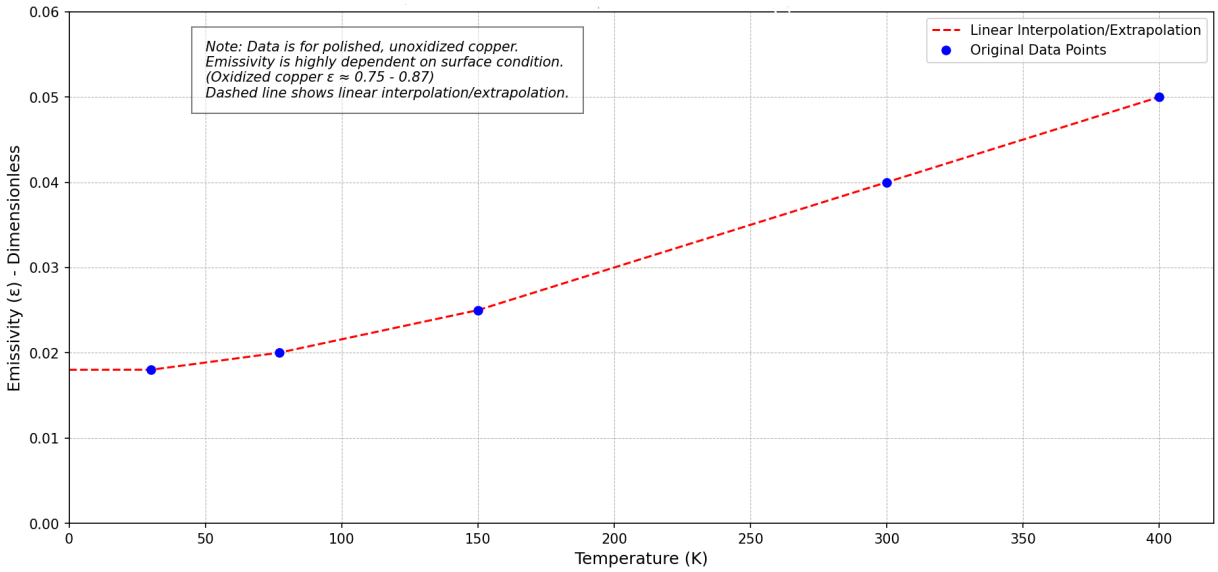


Figure 3.13: Emissivity of copper w.r.t temperature used in the COMSOL model.

Boundary and initial conditions

The governing equations are solved with a set of boundary conditions that define the physical constraints and interactions. These conditions are specified for each physics interface. At the entrance of the helium channel a Mass Flow Inlet condition is applied, specifying a constant mass flow rate. This formulation is more reliable than specifying an inlet velocity, as it ensures mass conservation even as the fluid density changes due to heating. A Pressure Outlet condition is applied at the channel exit. The pressure level is set to a reference value of 15 bara. This condition allows the pressure to be freely computed within the domain, with the pressure drop emerging as a result of the simulation. The condition is configured to prevent backflow. Despite the fact that in the Airbus Demonstrator the inlet should have a pressure of 15 bara, this condition is applied to the outlet as it provides a more stable convergence of the solver. Furthermore, a change of a few pascals at the entrance does not bring a significant change into the fluid properties which allows this change. A no slip wall condition, ($\mathbf{u} = 0$), is applied to all internal surfaces of the heat sink that are in contact with the helium. The flow in this area is modeled using wall functions [86].

The inlet temperature of the supercritical helium is set to 42 K at the inlet of the helium channel while the outlet fluid temperature is calculated via the simulation. A heat load is placed onto the heatsink through the natural convection condition imposed onto the warm end of the current lead. The ambient temperature is set to 293.15 K and the heat sink is initialized at the same temperature as the inlet helium temperature of 42 K. In the electromagnetic domain, a Terminal condition is applied at the hollow cylinder in the heat sink which will be in contact with the superconducting cables. The warm end of the heat sink is defined as a Ground boundary. All other sides of the heat sink and the current lead are insulated to ensure that the current travels through the intended path. A summary of all the boundary conditions applied to the model is presented in Table 3.3.

Table 3.3: Boundary conditions applied to the COMSOL model.

Physics Interface	Boundary Location / Domain	Condition Type	Value and Justification
Turbulent Flow, k-ϵ	Helium Channel Inlet	Mass Flow Inlet	$\dot{m} = 0.018$ kg/s. Specifies the coolant flow rate.
	Helium Channel Outlet	Pressure Outlet	$p = 15$ bara. Defines a reference pressure outlet.
	Helium-Solid Interfaces	Wall (No-Slip)	$\mathbf{u} = 0$. Standard condition for viscous flow at solid surfaces; near-wall flow resolved with wall functions.
Electric Currents	HTS cable termination	Terminal	Specifies total electrical current.
	Current lead warm end	Ground	$V = 0$. Defines the reference electric potential.
	All other external conductor surfaces	Electric Insulation	$\mathbf{n} \cdot \mathbf{J} = 0$. Confines current to the intended path.
Heat Transfer	Helium Channel Inlet	Temperature	$T = 42$ K. Defines the inlet temperature of the coolant.
	Helium Channel Outlet	Outflow	Open boundary condition; calculated by the solver.
	Helium-Solid Interfaces (Continuity)	No condition applied.	Physics interface enforces $T_{\text{solid}} = T_{\text{fluid}}$ and flux continuity automatically.
	Busbar External Surfaces	Heat Flux	Mixed-mode heat rejection. Radiation: emissivity = $\varepsilon(T)$, Flat plate natural convection $T_{\text{ext}} = 293.15$ K (Ambient Temperature).
	All other external cryogenic surfaces	Thermal Insulation	$q_0 = 0$. Assumes operation in a vacuum cryostat, neglecting external heat transfer.

Geometry modifications

Upon running the simulation for the axial fin type configuration of the heat sink, it was seen that the placement of the current lead in the centre was blocking several channels and causing issues with simulation convergence. In order to resolve this issue, a circular clearance was introduced around the base of the current lead. The idea behind this modification was to allow the helium to continue flowing around the current lead base and back into the central channels. This idea however neglected the fact that the main source of heat load is the current lead, and by disconnecting the fins from the current lead base the heat load enters straight into the core of the heat sink increasing the temperature even further.

In order to allow a smooth flow of helium through the central channel while also extracting the heat load from the current lead base a thick central fin was introduced. This thick fin ran down the length of the heat sink and was connected to the current lead. This fin thickness was set to twice the thickness of the other fins. This iteration is shown along with the others in Figure 3.14. Despite the addition of the thick central fin, the temperature of the heat sink core was too high. To dissipate more heat from the current lead, and consequently lower the heat sink core temperature, a small rectangular-curved plate was added to the top of the fins around the current lead. This significantly improved the temperature distribution on the heat sink, however, the temperature at the core was still too high. Therefore, instead of using axial fins which help dissipate heat from the core of the heat sink, the design was changed to dissipate heat directly from the base of the current lead. This introduced the idea of concentric fins as seen in figure

Figure 3.14. The thickness of the concentric fins and the fin-spacing was varied based on the results from the COMSOL simulations until a uniform distribution of the heat load was achieved and the requirements set in Section 3.1.1 were met.

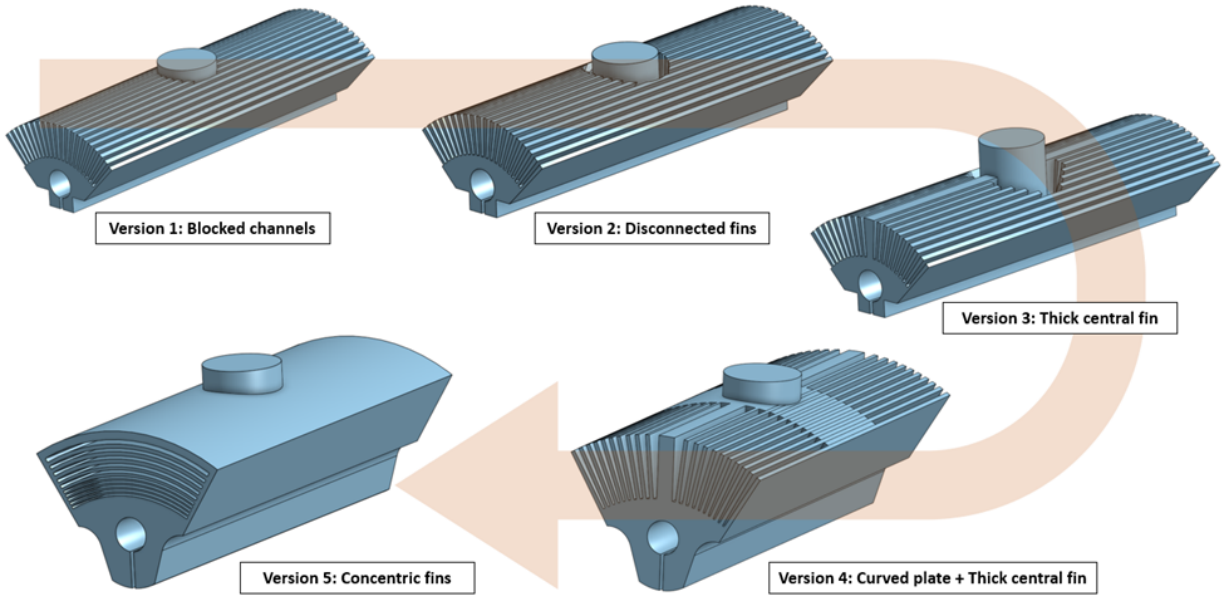


Figure 3.14: Development of the heat sink design.

3.2. Experimental research and methodology

Once the design satisfied the project requirements in the 3D simulations, an experiment was devised with the final part printed in copper. The goal of this experiment was to show that the results of the pressure drop and temperature gradient across the heat sink from the final COMSOL simulation correspond to the real life data and to satisfy the original set of requirements. This section of the report covers the complete experimental setup. Section 3.2.1 talks about the setup itself which includes an overview of the parts and connections used on the setup. Next, Section 3.2.2 starts with an overview of the required measurements described in detail, along with the set of instruments needed to capture the data.

3.2.1. Experimental setup

For the heat sink to be tested, an appropriate connection (also referred to as the header) was 3D printed in copper to connect the inlet and outlet of the heat sink to 12 x 14 mm copper tubes. A male and a female VCR 1/2 inch coupling was attached to the ends of the copper tubes to connect the setup to the helium loop at the University of Bath, UK. This is where the heat sink was tested. A 2 x 4 mm one-meter-long copper tube was brazed onto the inlet and the outlet copper tubes to act as the access point for the differential pressure sensors. All the connections on the setup were done with silver brazing (40% Ag, 60% Cu). A CAD drawing of the heat sink setup is shown in Figure 3.16. Two final setups were created for the test in Bath. These setups are shown in Figure 3.15. Setup B includes a flexible corrugated copper tube instead of the rigid copper tube used in setup A. This was done on purpose, as any mismatch with the setup in Bath could easily be compensated for by the flexible tube on setup B.

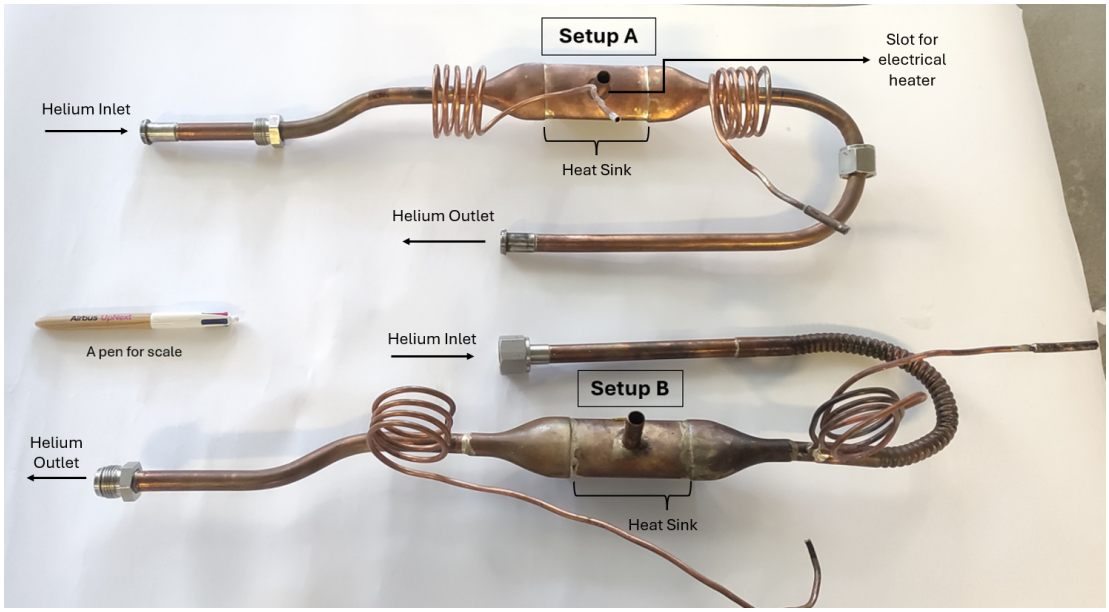


Figure 3.15: The final setup A and setup B for the heat sink tests at the University of Bath, Uk.

Once the setups were completed, they were subjected to pressure tests. Firstly, a high-pressure test was performed at 20 bara. During this test, the outlet tube of the setup was closed off with another female VCR coupling. Then the setup was pressurized to 20 bara and left inside a closed bunker for 30 minutes while the pressure was constantly monitored. This was done to ensure the structural integrity and check for system leaks at high-pressure operation. Once it was seen that the pressure did not change, the setup had passed the high-pressure test. Next, the pressure was lowered to 2 bara and a helium sniffer was used to detect any small leaks at the brazed joints. It is important to note that both the aforementioned tests were conducted at room temperature. After having passed the low-pressure leak test at room temperature, the setup was subjected to a cold high-pressure test. Here, the setup was pressurized to 20 bara with helium gas and slowly cooled with cold nitrogen vapors before being lowered into a liquid nitrogen bath to see if the silver brazed joints would crack under the thermal shock. Photos of this final test are presented in Figure 3.17 and Figure 3.18. Both setups were subject to the same tests.

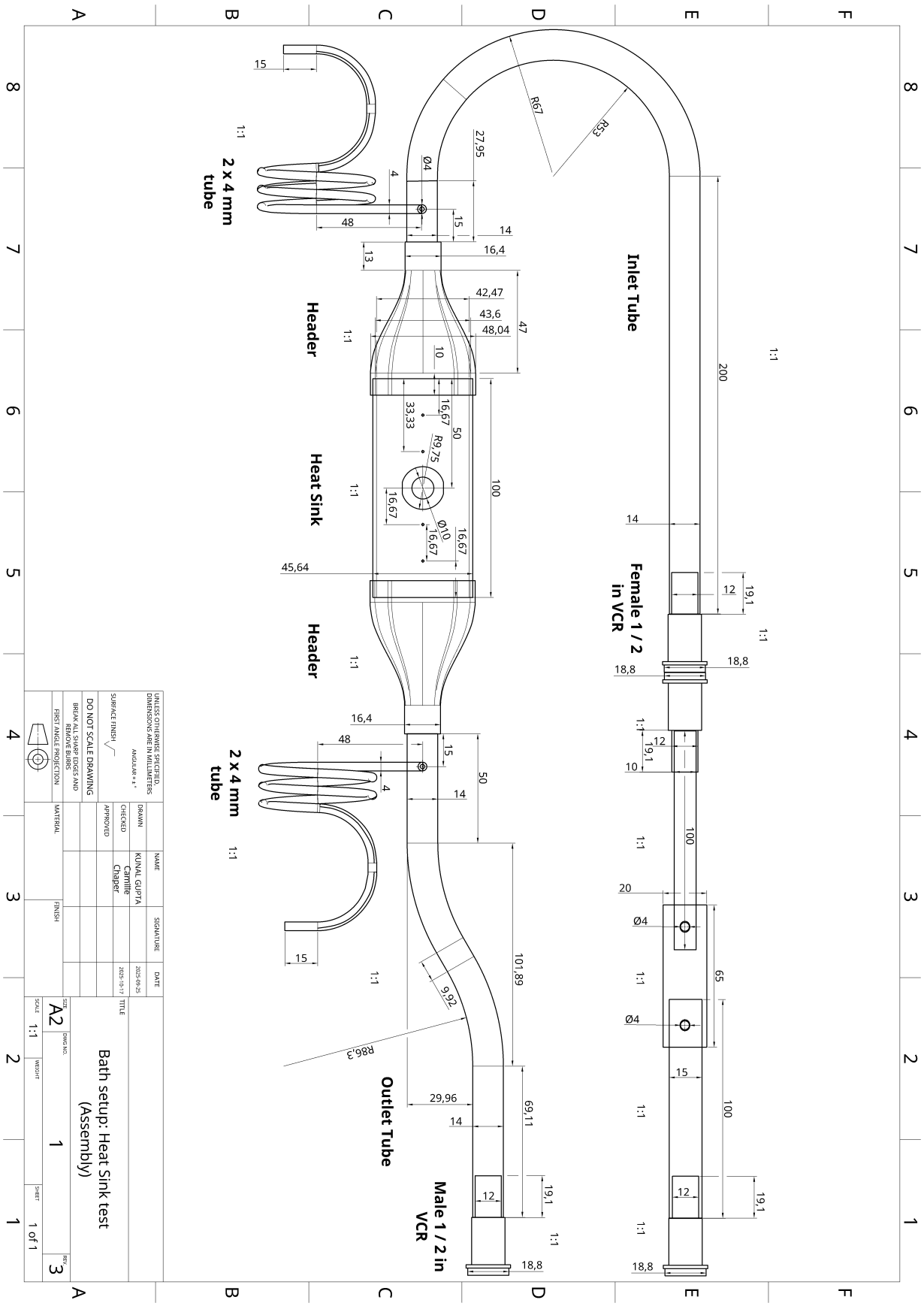


Figure 3.16: A CAD drawing of setup A for testing in Bath, UK.

Figure 3.17: 20 bara high-pressure test under cold nitrogen vapor cooling.

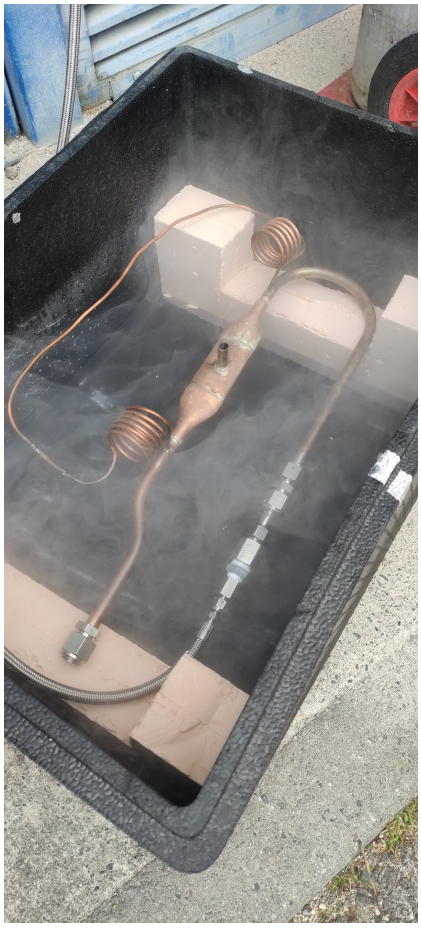


Figure 3.18: 20 bara high-pressure test under liquid nitrogen cooling.

After successfully passing the leaks and pressure tests, the setup was transported to Bath for testing with supercritical helium. During the installation at Bath only setup B could be tested due to time limitations. It was mounted with a venturi meter to measure the flow rate of the supercritical helium using the pressure measurements at the throat and inlet of the venturi meter as seen in Figure 3.19. Next, the first half of the outer shell (cryostat) was placed around the setup and clamped to the ISO K DN-200 flange at the inlet. This can be seen in Figure 3.20 alongside Figure 3.21 that shows all the CERNOX cryogenic temperature sensors and electrical heater mounted onto the heat sink with Apiezon grease (conductive grease) and Kapton tape.

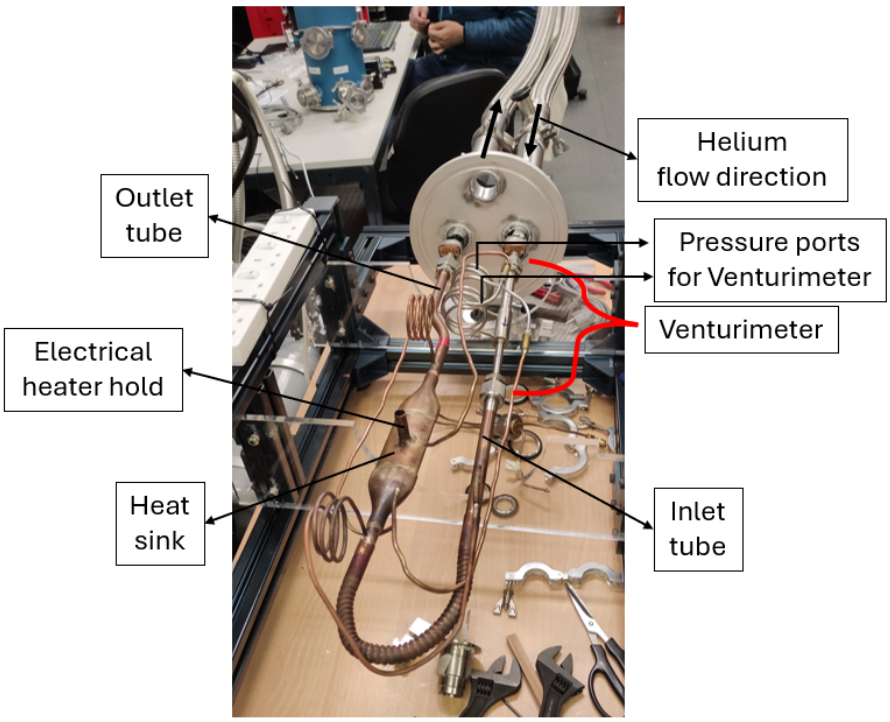


Figure 3.19: Installation of setup B with the helium loop at the University of Bath.

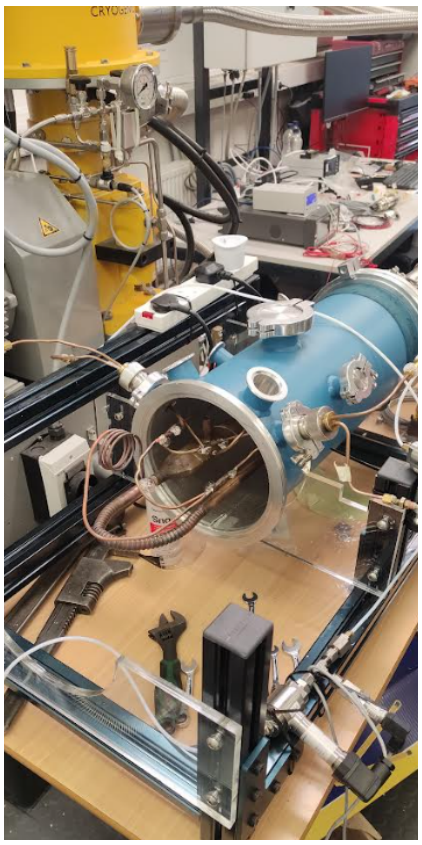


Figure 3.20: Image showing the first half of the cryostat encasing the heat sink setup B.

Figure 3.21: Image showing the CERNOX cryogenic temperature sensor and electrical heating mounting onto the heat sink.

The complete setup with the Stirling cryocooler and the heat sink is presented in Figure 3.22. This is an industrial cryocooler (SPC-1T) with a 30 W cooling power at 15 K. This cryocooler uses the Noorden Wind cryofan to circulate the helium gas through the setup [87]. The main heat exchnager of the Stirling cryocooler is integrated within the cold head. The cooldown of this cold head is achieved through the reverse Stirling cycle of the helium gas in the closed circuit. This Stirling cycle is executed by the piston cylinder mechanism driven by the electric motor shown in Figure 3.22. The expansion of the gas cools the cold head while the heat generation from the iso-choric compression is ejected into the cooling water supply as shown in Figure 3.24. This cold head then cools the helium gas which is being pumped through the inlet of the test setup. This circuit is complete isolated from the test setup and is part of the Stirling machine. The frequency modulator also shown in Figure 3.22 is used to tune the cooling power of the cryocooler.

The pressure sensor measurements on the test setup were done with the Keithley DAQ6510, while the vacuum gauge measurements were recorded using the Agilent Technology's AGC-100. The measurements of the temperature sensors were recorded through the Cryocon 14i.

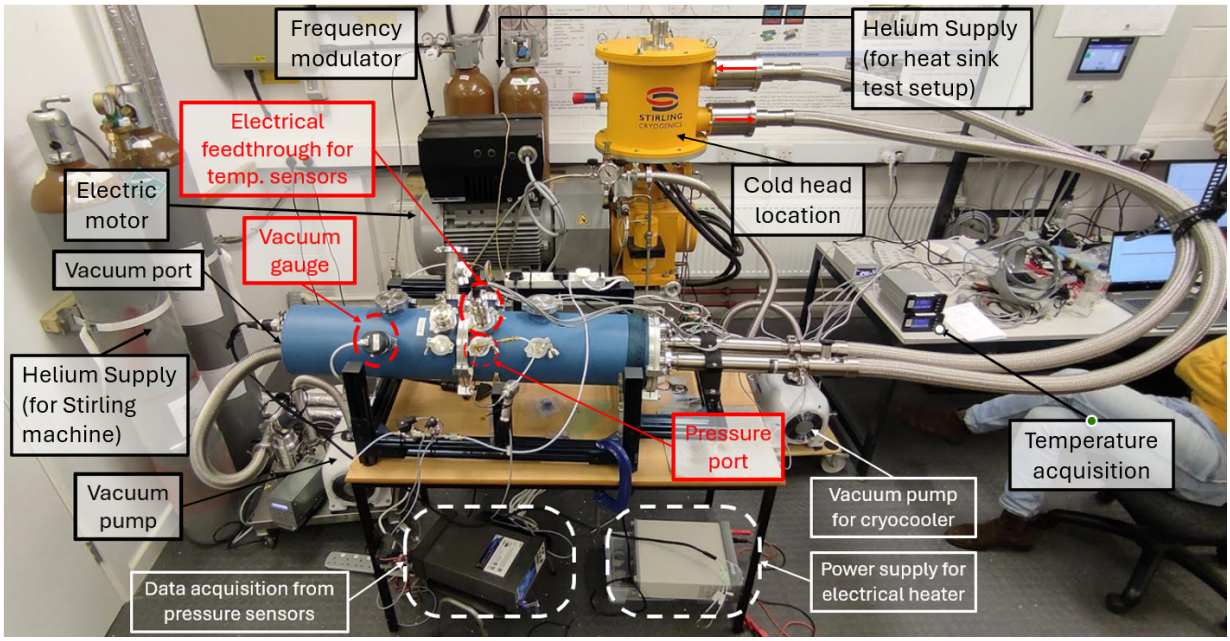


Figure 3.22: Complete experimental setup with the Sterling cryocooler at Bath.

Two CryoCon monitors were used to connect the seven temperature sensors. However, during the experiment, it was noticed that four of the seven CERNOX temperature sensors were giving erroneous readings. The sensors mounted at the helium inlet, outlet, second temperature measurement port from the inlet, and third temperature measurement port from the inlet were the four temperature sensors giving erroneous results. Therefore, the readings from these sensors were ignored and the inlet and outlet temperature readings were replaced with the readings from the inlet and outlet of the Sterling cryocooler at TT 25.73 and TT 25.71, respectively. These temperature ports on the Sterling cryocooler are shown in Figure 3.24. Since achieving a vacuum level in the range of 10^{-3} to 10^{-4} mbar at the highest available pumping speed required more than two hours, followed by approximately 90 minutes to cool the system to nearly 40 K, the malfunctioning temperature sensors mounted on the heat sink were not removed during the initial test day (22/10/2025).

During the second test day (23/10/2025), the vacuum chamber was vented and the setup returned to ambient temperature. As no additional CERNOX temperature sensors were available to replace those with unreliable measurements, the three functional sensors were repositioned to be symmetrically distributed along the heat sink. The inlet and outlet helium temperatures continued to be monitored using the TT 25.73 and TT 25.71 integrated into the Stirling cryocooler.Figure 3.23 shows the placement of the working temperature sensors on both days of testing.

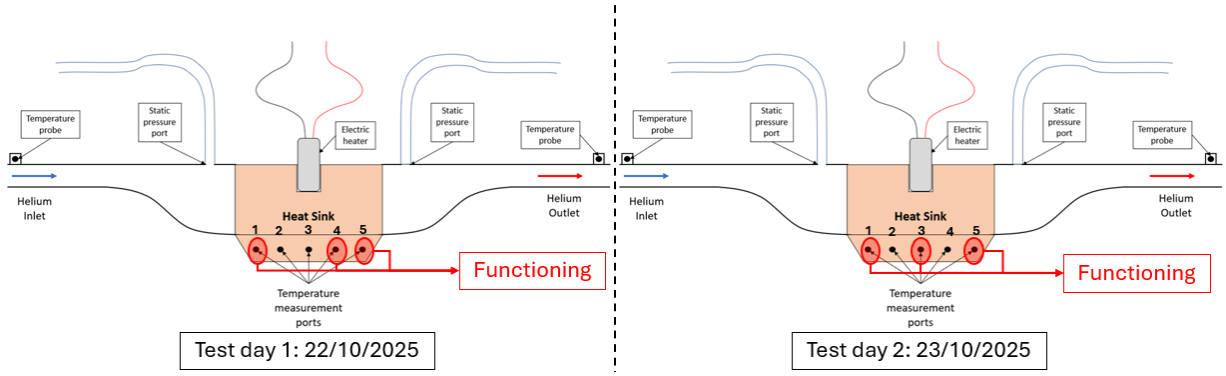


Figure 3.23: Set of functioning temperature sensors on both days of testing.

To characterize the heat sink at different mass flow rates, the experiment was conducted at three different cryofan speeds: 8036 rpm, 13100 rpm, and 17680 rpm. The exact location of the cryofan is indicated by component 23.74 in Figure 3.24. At each cryofan speed, different heat load scenarios were tried by regulating the power to the electrical heater and letting the system stabilize. On day one, four heat load scenarios were tried at a cryofan speed of 8036 rpm: 10 watts, 30 watts, 50 watts, and 80 watts. On day two of the test campaign, three different heat load scenarios were tried at a cryofan speed of 13100 rpm and 17680 rpm each: 30 watts, 50 watts, and 80 watts. An important note to make is that even though the cryofan speed is constant over the varying heat loads, the mass flow rate of helium is not constant. The cryofan controls the volumetric flow through the setup; however, the fluid density changes as more heat load is added to the system, leading to an increased helium inlet temperature. This change in temperature and density results in a varying mass flow rate at a constant cryofan speed.

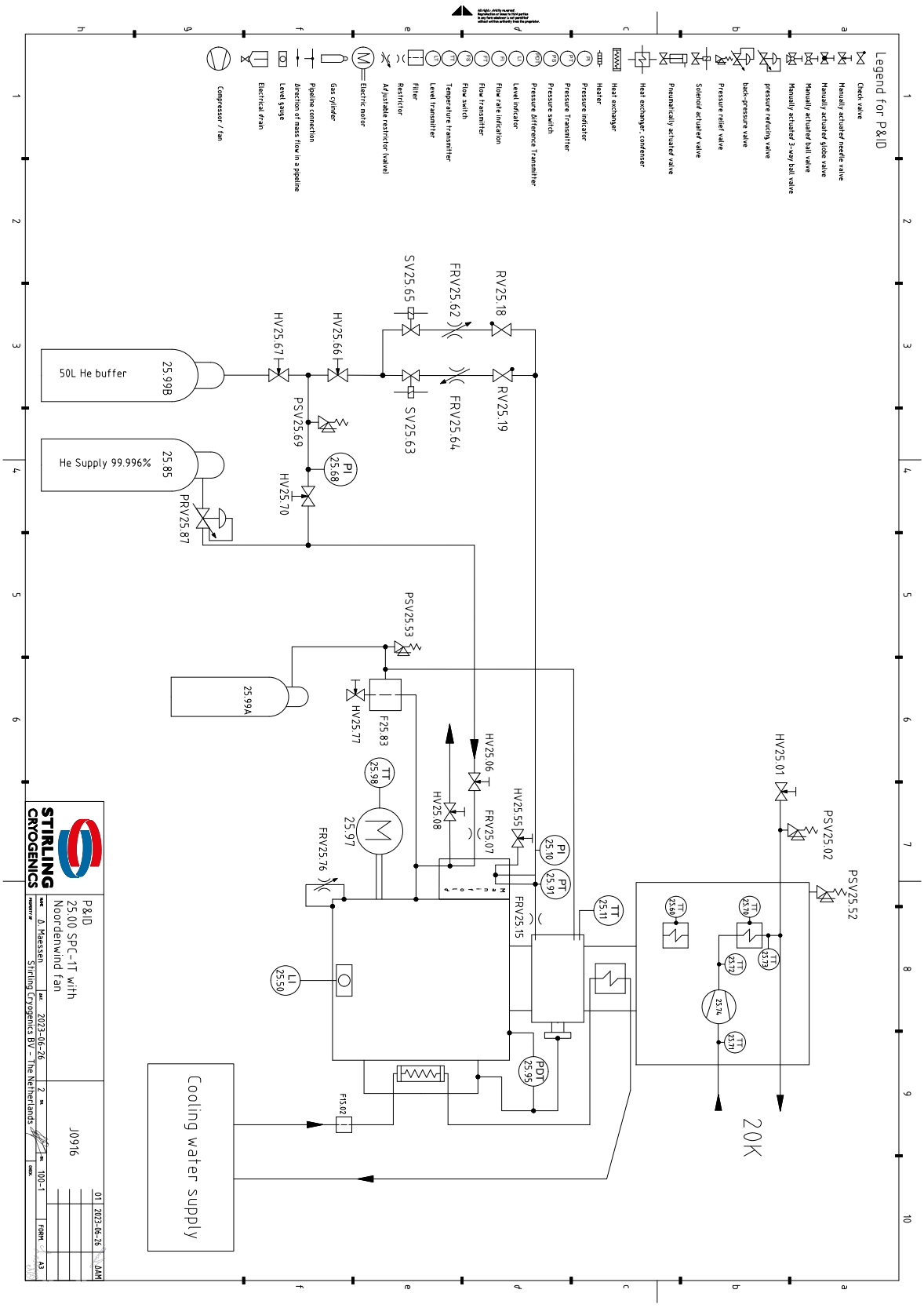


Figure 3.24: The piping and instrumentation diagram of the Sterling cryocooler.

3.2.2. Measurements

The two main measurements required from the experiment are the temperature and pressure measurements. This section covers instruments used for the temperature and pressure measurements along with their placement on the set up.

Temperature measurements

The temperature at the inlet and outlet of the cooling fluid (helium) is measured along with five other points along the length of the heat sink to observe the temperature gradient along its length. The temperature measurements are done with the CX-AA Cernox temperature sensors as shown in Figure 3.25. These sensors are calibrated for a temperature range of 4 K to 325 K with a tolerance of ± 3 mK [88].

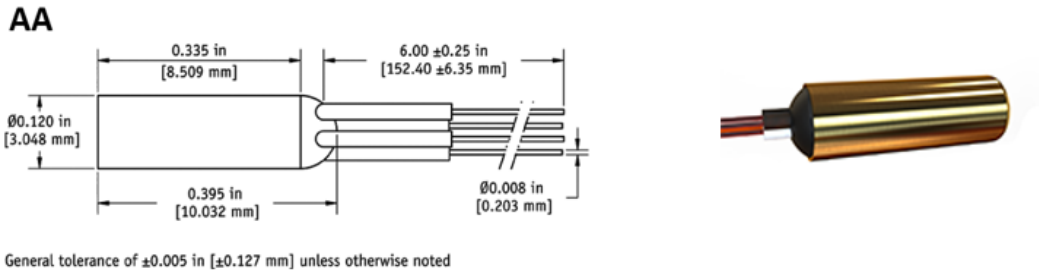


Figure 3.25: CX-AA Cernox cryogenic temperature sensors. [88]

The inlet and outlet fluid temperature measurements were initially planned to be performed externally through contact with the external wall of the fluid-carrying tubes. This decision was taken to avoid any leaks resulting from attempts to make a hole in the tube for submerged temperature measurements. It is understood that the temperature measurements conducted externally may not represent the exact temperature of the fluid inside, however, the measured temperatures were assumed to be close to the fluid bulk temperature due to the relatively low thickness of tube. Nevertheless, these temperature measurements were later ignored due to the malfunctioning sensors as discussed in Section 3.2.1. The temperature measurements along the heat sink are done by inserting the Cernox sensors inside five equally spaced holes made along the length of the heat sink. A graphical representation of the temperature sensor placement is shown in Figure 3.26.

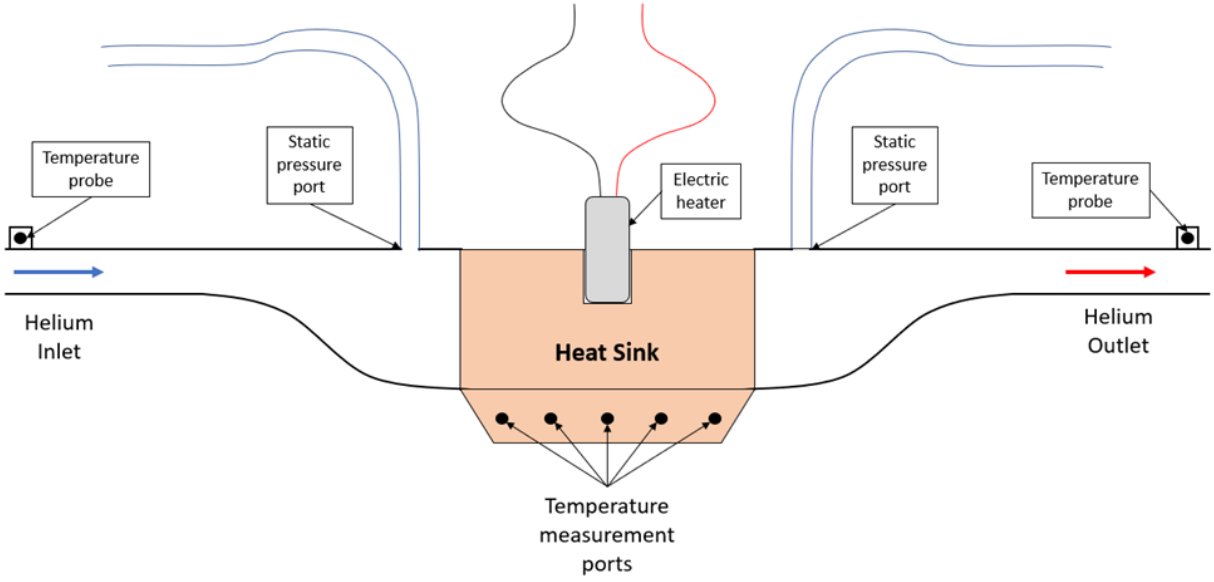


Figure 3.26: A graphical representation of the temperature sensor location on the setup.

Pressure measurements

The pressure difference across the heat sink is measured using the PD-23X differential pressure sensor, also shown in Figure 3.27. The static ports, as shown in Figure 3.26, are used to access the pressure before and after the heat sink to get the measurements on the pressure drop across it. The performance and specifications on the PD-23X differential pressure sensor can be found here [89]. Further upstream the setup a static pressure sensor, PA-21Y - shown in Figure 3.28, is mounted together with a differential pressure sensor, PD-23X, to get the static pressure of the loop along with the mass flow rate of helium flowing through the setup. The specification of PA-21Y can be found here [90]. These pressure sensors read the voltage across them and relate that directly to the presssure readings. The PD-23X has a direct linear correlation between the voltage measurements and the pressure readings.

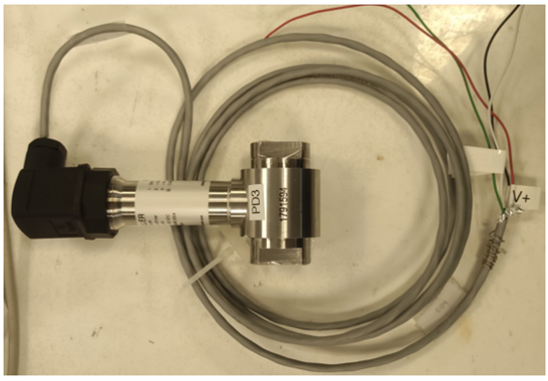


Figure 3.27: PD-23X differential pressure sensor.

Figure 3.28: PA-21Y static pressure sensor.

In addition to the differential and static pressure sensors, a vacuum gauge was used on the cryostat to monitor the vacuum level inside. For this purpose, the Agilent FRG-700 gauge was used. The vacuum gauge is shown in Figure 3.29 and Figure 3.30.

Figure 3.29: Side view of the FRG-700 vacuum gauge.

Figure 3.30: Top view of the FRG-700 vacuum gauge.

4

Results & Discussion

This chapter provides the results and their interpretation for the 1D numerical models in Section 4.1, the COMSOL simulations in Section 4.2, and talks about the experiments performed at the superconducting laboratory at the University of Bath in Section 4.3. Section 4.4 provides a thorough comparison between the simulated result and the experimental results. Next, the limitations from the experiment are discussed in Section 4.5 and the research question is answered in Section 4.6.

4.1. 1D numerical model results

This section provides the results from the 1D numerical model of the current lead and the heat sink. Firstly, the results from the current lead model are shown in Section 4.1.1. This 1D numerical model for the current lead shows the thermal performance of the current lead at different cross-section to length ratios and the operating current by calculating the heat load due to conduction and Joule heating. Secondly, the outputs of the heat sink model are presented in Section 4.1.2 along with a set of input variables presented in Table 4.1. The heat sink model calculates the convective heat transfer coefficient, the wall temperature of the heat sink, and the pressure drop across to provide an initial indication of its size and geometry. The main goal of these 1D numerical models is to get a preliminary estimate of the heat load falling onto the heat sink and an idea of the size of the component.

4.1.1. 1D numerical model results: Current lead

To get an estimate for the primary heat load on the heat sink, a 1D numerical model for the operation of the current lead was developed. The graphic user interface (GUI) created for the current lead numerical model is shown in Figure 4.1. This shows the input parameters for this model, such as the material, the temperature at the warm end of the current lead, a set of temperatures at the cold end, the RRR value of the material, and a set of operating currents.

For a copper current lead with a RRR value of 10 and with its ends at T_{cold} = 50 K and T_{Hot} = 293 K, carrying a nominal operating current of 1.5 kilo amps, a specific length to cross-section (L/A) ratio exists requiring the lowest cooling power as seen in Figure 4.2. This L/A is 24.46 with a required cooling power of 67.6 watts per current lead. To account for any manufacturing defects or an over-current scenario, an additional 12.5% of cooling power was added as a safety margin to reach a total of 80 W heat load per current lead. A total of 4 current leads are required for a 4-pole design, leading to a total of 320 watts of heat load onto the heat sink.

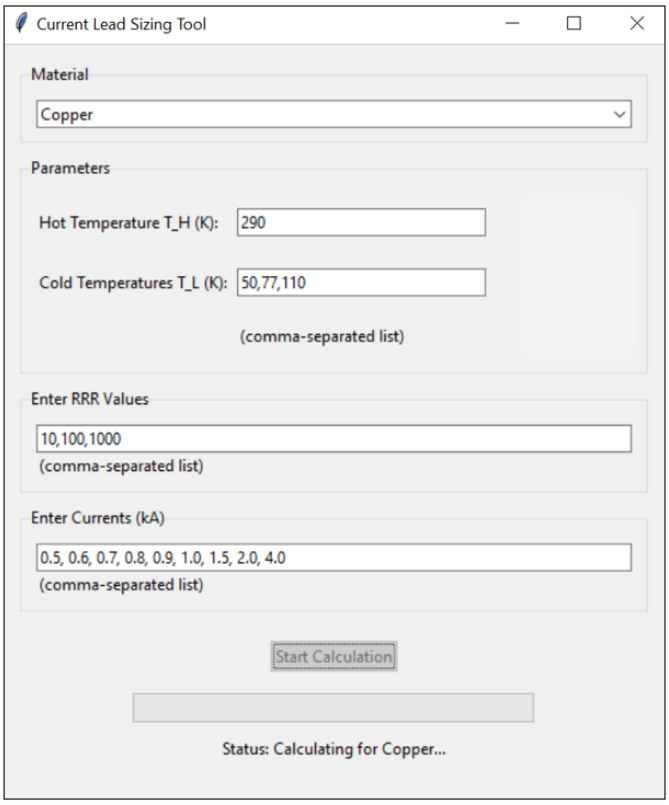


Figure 4.1: The GUI for the 1D current lead numerical model.

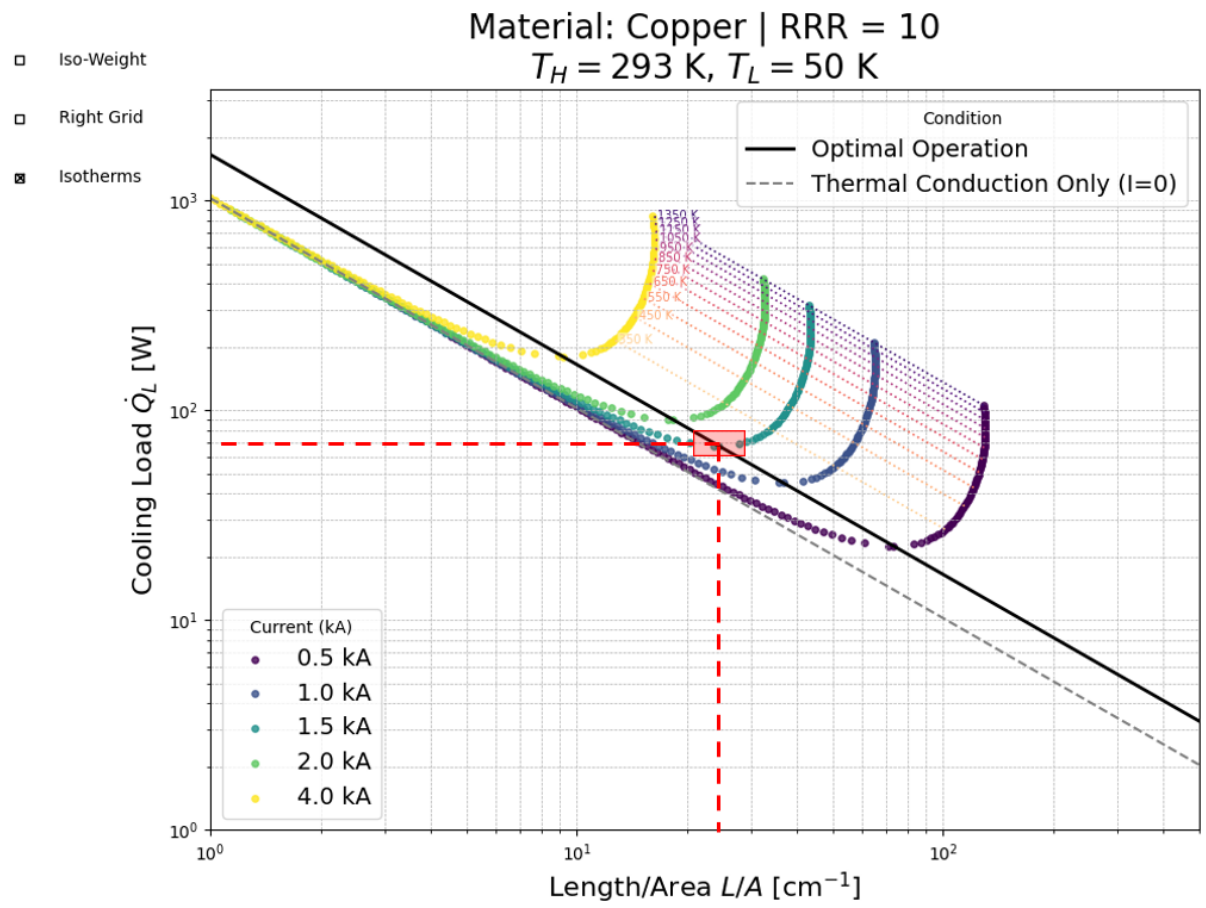


Figure 4.2: The output of the 1D current lead sizing tool. RRR = 10

Figure 4.2 shows not only the design heat load on the heat sink but the entire design envelope for the current lead, including off-design cases (over current scenario). The isotherms on Figure 4.2 help gauge the temperature of the current lead in case of over-current. For example, if a current lead is sized for 1.5 kilo amps nominal current with a T_H of 293 K and a T_C of 50 K, it will operate at a maximum temperature of 293 K. However, if the same current lead (same L/A) is subjected to a higher current of 2 kilo amps, the temperature of the current lead will immediately rise to almost 350 K as seen in Figure 4.2. If the purity of the material is increased, the RRR value, the optimum L/A ratio for the current lead also increases as seen in Figure 4.3, implying that the current lead can have a lower cross-section area for the same length, reducing its weight. However, the current lead also becomes highly sensitive to the operating current. In case of over-current scenarios with the high-purity current lead, the temperature climbs much faster.

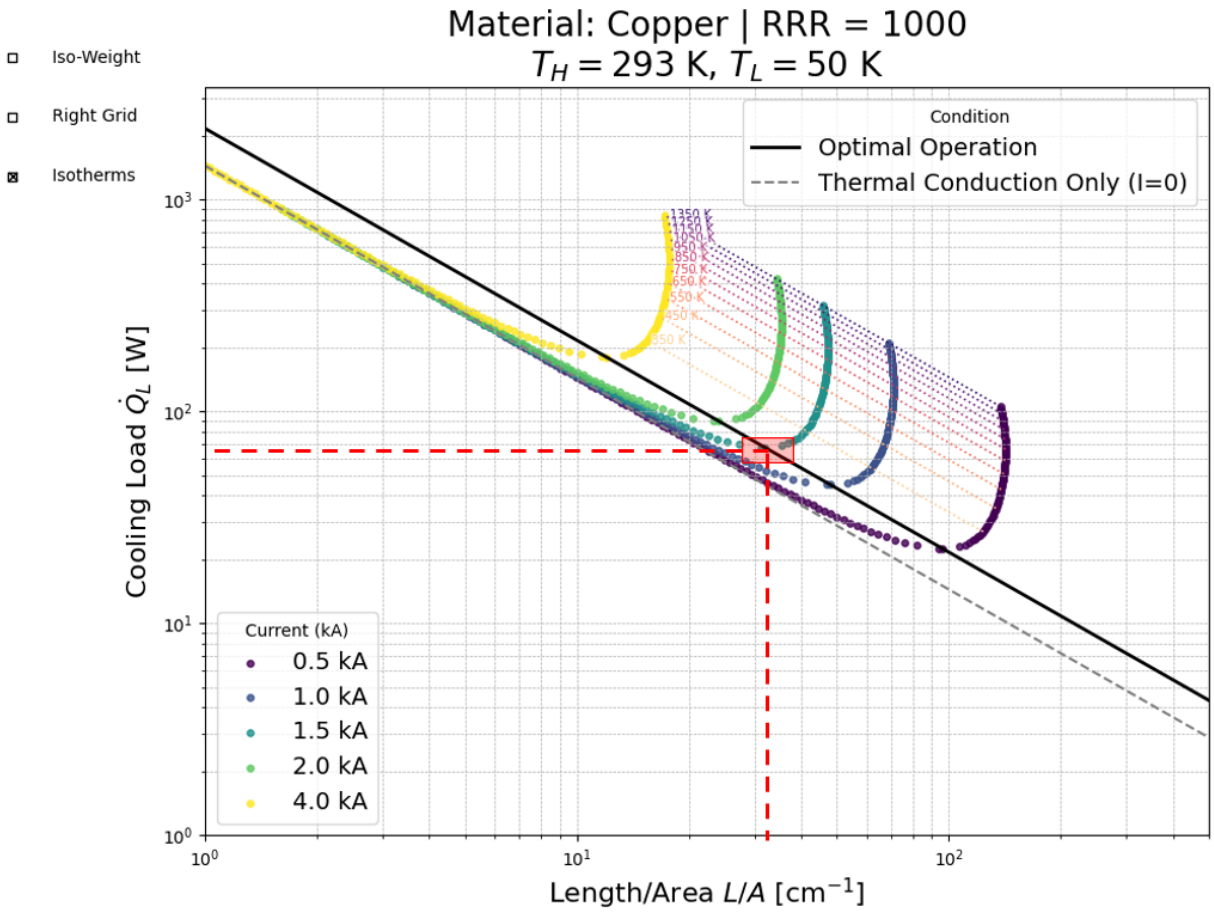


Figure 4.3: The output of the 1D current lead sizing tool. RRR = 1000

4.1.2. 1D numerical model results: Heat sink

The heat sink model discussed here is based on the original axial fin design around a cylindrical heat sink as shown in Figure 3.1. The reason behind choosing a cylindrical heat sink design is the integration of the heat sink into a cylindrical tube-like structure. A graphical user interface (GUI) was created to allow the user to input the desired properties of the fluid, the range of mass flow rate of the desired fluid, and the range of the number of fins. Figure 4.4 shows a screenshot of the GUI for the heat sink model with all the input parameters summarised in Table 4.1.

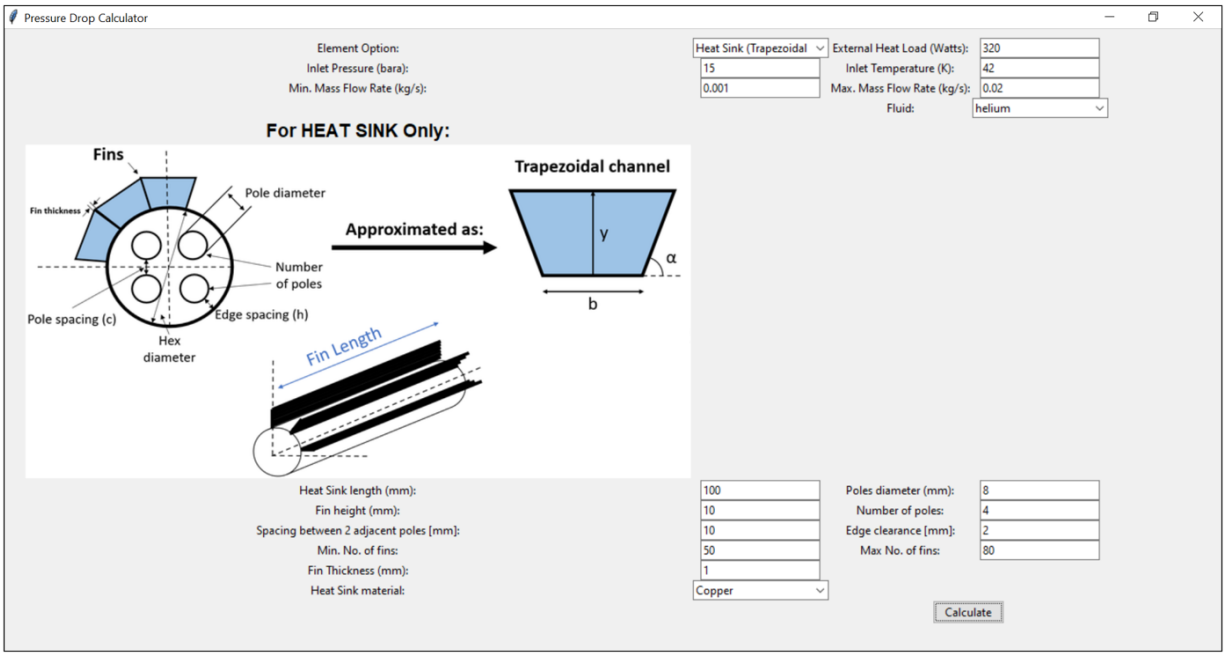


Figure 4.4: The GUI of the 1D heat sink model.

Table 4.1: Input variables for the 1D heat sink model.

Input Variable	Value	Units
Inlet pressure	15	Bara
Minimum mass flow rate	0.001	kg/s
Maximum mass flow rate	0.010	kg/s
Fluid	Helium	-
Inlet fluid temperature	42	K
Heat sink length	100	mm
Fin height	15	mm
Space between two adjacent poles	10	mm
Pole diameter	8	mm
Number of poles	4	-
Edge clearance	2	mm
Minimum number of fins	50	-
Maximum number of fins	80	-
Heat sink material	Copper	-
Heat load	320	Watts

Parameters like the *Heat sink length*, *Inlet fluid temperature*, *Inlet pressure*, *Fluid*, *Number of poles*, and *Pole diameter*, are set by the project. However, the values for the *Edge clearance* or *Space between two adjacent poles* are based on the internal expertise at Airbus Upnext. The heat load value is taken from the current lead model in Section 4.1.1. This heat load is then imposed uniformly onto the heat sink. Next, the fluid velocity through the axial fins of the heat sink is calculated for a range of equally spaced fins on the x-axis and a range of coolant mass flow rates on the y-axis as seen in Figure 4.5. Once the base diameter of the heat sink is calculated via Equation 3.1, the maximum number of fins that can fit around the heat sink without overlapping can be calculated for a given fin thickness. This value is then used as the maximum for the number of fins. On the y-axis, the mass flow rate range is shown, starting with a

user-defined minimum and up to the user-defined maximum. For any single mass flow rate value, the fluid velocity increases with the increasing number of fins around the heat sink as seen in Figure 4.5. This is as expected due to the reduction in flow area with the increase in the quantity of fins.

Similarly, the convective heat transfer coefficient calculated via the equations mentioned in Section 3.1.2 is also plotted against the number of fins and the mass flow rates. Since the convective heat transfer coefficient is coupled with the flow velocity, it too increases with the increasing number of fins. At a mass flow rate of 18 g/s the convective heat transfer coefficient ranges from 800 $W/m^2 \cdot K$ for 50 fins upto 3000 $W/m^2 \cdot K$ for 80 fins. This can be seen in Figure 4.6.

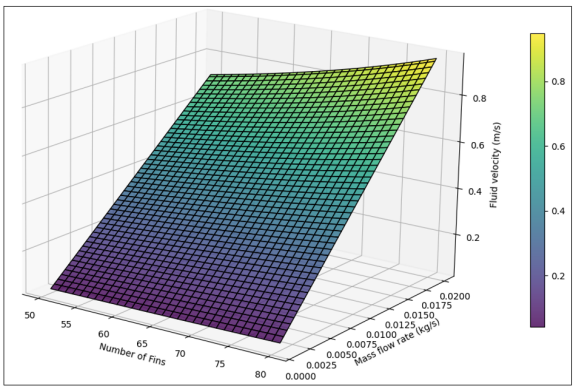


Figure 4.5: Velocity for a combination of mass flow rates and quantity of fins as stated in Table 4.1.

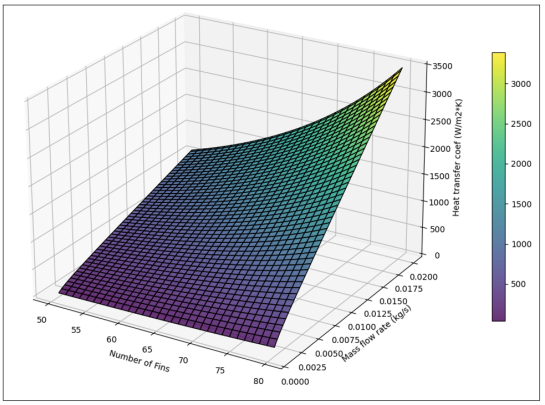


Figure 4.6: The convective heat transfer coefficient for a combination of mass flow rates and quantity of fins as stated in Table 4.1.

Based on this range of heat transfer coefficient, a combination of number of fins and mass flow rates was found that would ensure a mean wall temperature of 46 K for a uniformly distributed heat load of 320 W. This was done by intersecting the numerical solution of the mean wall temperature for the set of inputs given in Table 4.1 with a z-plane at z = 46 K as shown by Figure 4.7. The intersecting points are all the possible numerical solutions that ensure a wall temperature of 46 K for different mass flow rates and numbers of fins. A vertical plane (red plane) was then erected from the intersecting line in Figure 4.7 and intersected with the pressure drop surface as seen in Figure 4.8. This was done to find which possible solution from Figure 4.7 caused the lowest pressure drop over the length of the heat sink. It was seen that at the maximum available flow rate of 18 g/s (project constraint), a total of 78 fins are required for a cylindrical heat sink with a base diameter of 37.5 mm to maintain a mean wall temperature of the heat sink at 46 K, causing 15 Pa of pressure drop. With this, an initial design was obtained that satisfied the project requirements. While this model's simplicity (e.g., assuming uniform flow, uniform heat load imposition, and constant temperature across the heat sink) was its limitation, its value was in rapidly confirming the feasibility and establishing a baseline geometry for the 3D multiphysics simulation.

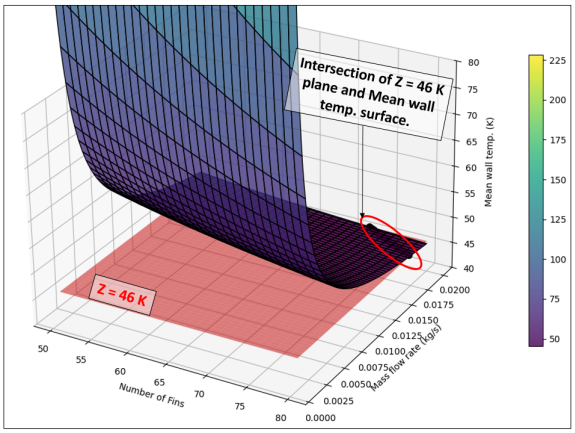


Figure 4.7: The mean wall temperature for a combination of mass flow rates and quantity of fins as stated in Table 4.1.

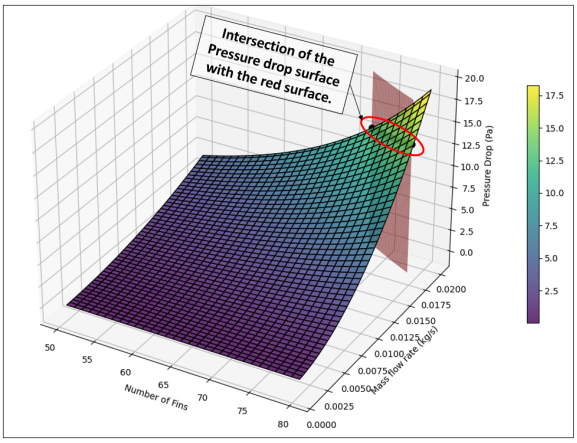


Figure 4.8: The pressure drop for a combination of mass flow rates and quantity of fins as stated in Table 4.1.

4.2. COMSOL Multipysics simulation results

The first simulation was done with the thick central fin configuration as shown in Figure 3.14. Instead of simulating the complete cylindrical heat sink, just half of one of the 4 heat sinks (one per pole) was simulated. This is because the heat sink is symmetrical. Hence, the heat load and the coolant mass flow rates were divided by eight. Since one current lead is sized to carry 1500 amps, the imposed current onto the model is halved as only half of the current lead end is simulated. This can be seen in Figure 4.9. A heat load of 320 watts is imposed onto the end of the current lead as shown by the red arrow in Figure 4.9. This was done to account for the heat load from the current lead, which was not part of the first simulation. An outlet pressure of 15 bara was fixed while the upstream pressure was calculated via the simulation. For a steady state, time-invariant simulation, the temperature profiles along the length of the heat sink at three different locations are presented in Figure 4.11. These locations are shown in Figure 4.10 where the red cross indicates a line going through the top of the center fin, the green cross indicates the line just above the superconducting termination, and the blue cross indicates the line just below the superconducting terminations.

The pressure field across the heat sink is shown in Figure 4.12. This pressure field shows a gradual decrease in pressure as helium moves further downstream along the length of the heat sink. This is in line with the expected pressure field distribution along the heat sink due to the expansion of the fluid as it extracts heat from the fins. The expansion of fluid along the heat sink length also causes the flow velocity to increase towards the heat sink outlet. Furthermore, the central channels (from the inlet to the center of heat sink) near the current lead show a higher static pressure than the flow channels further away from the current lead base. This stems from the blockage due to the current lead base in the center. However, after the current lead base, the static pressure in the central channels is slightly lower than in those further away. This is because the helium flowing in the central channels is closer to the heat source, which raises its temperature more than the temperature of helium flowing between in the outer channels (away from CL). This increased helium temperature in the central channels after the current lead causes an increase in the flow velocity and therefore a slightly lower static pressure in comparison. Finally, the pressure drop between the inlet fluid face and the outlet fluid face was calculated to be 51 Pa.

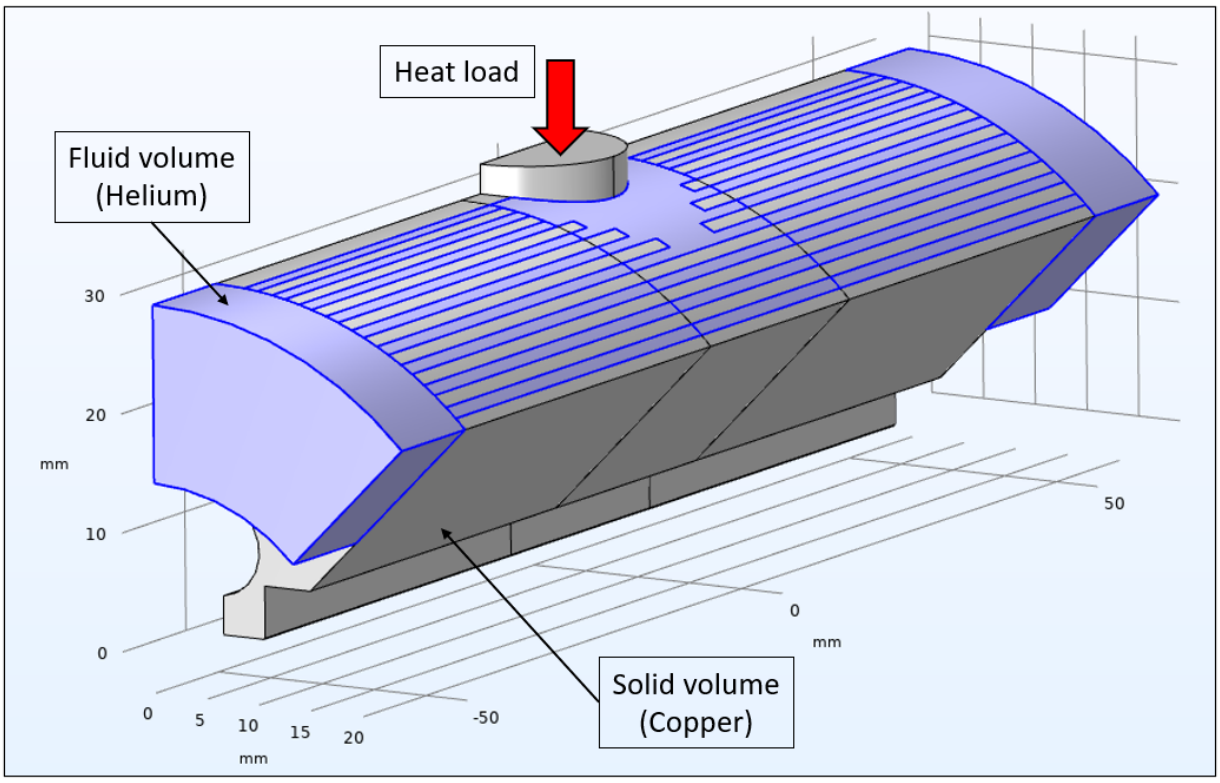


Figure 4.9: The CAD model of the heat sink to be simulated in COMSOL.

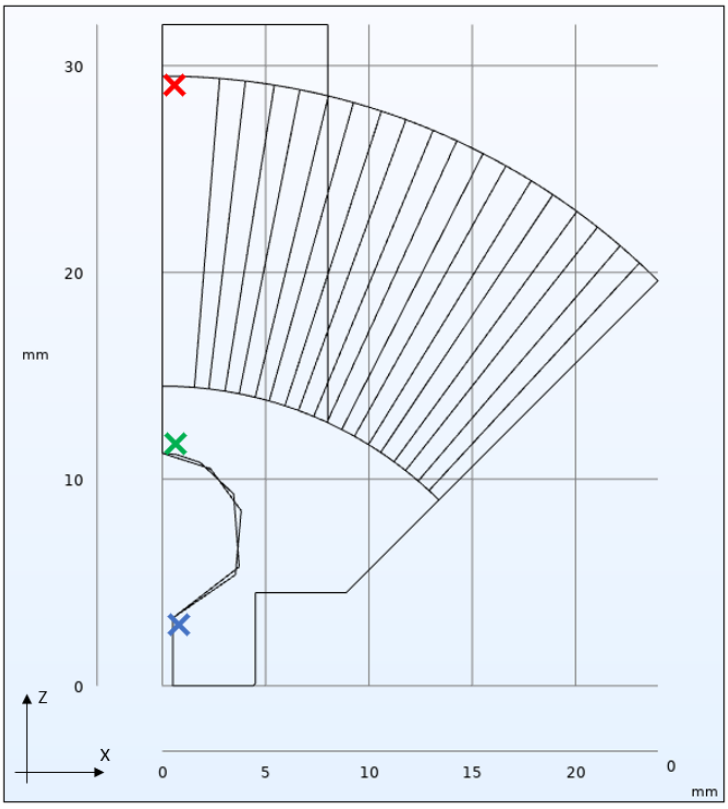


Figure 4.10: Location of the temperature probe in COMSOL.

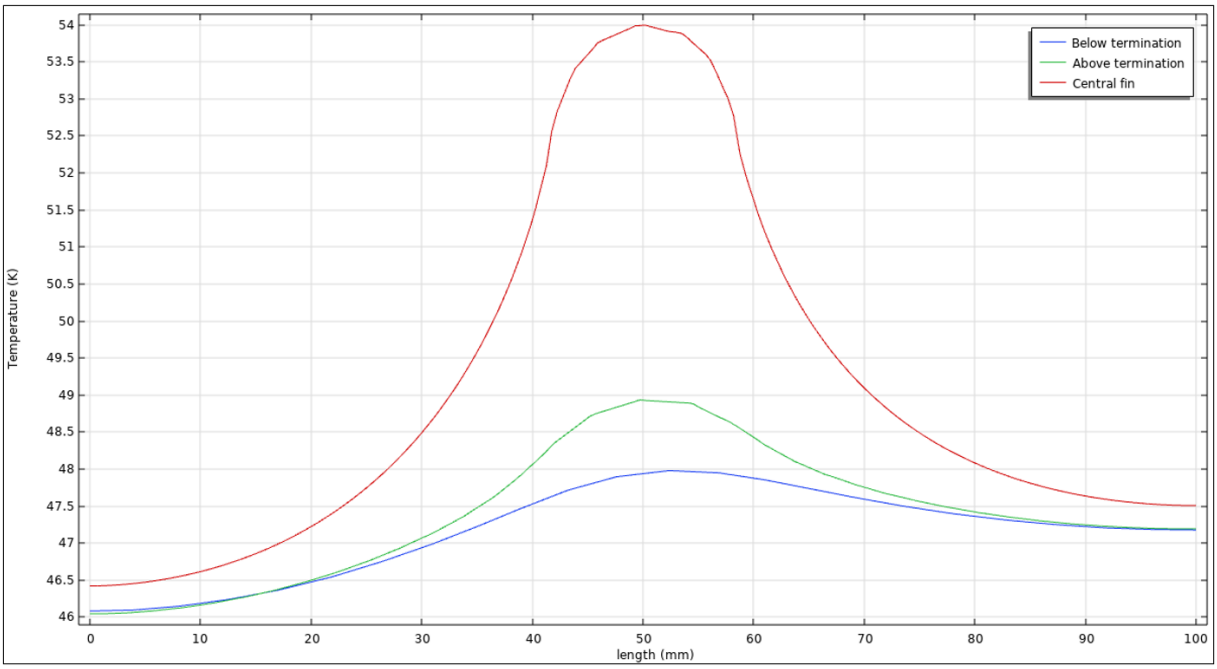


Figure 4.11: Temperature profiles along the length of the heat sink.

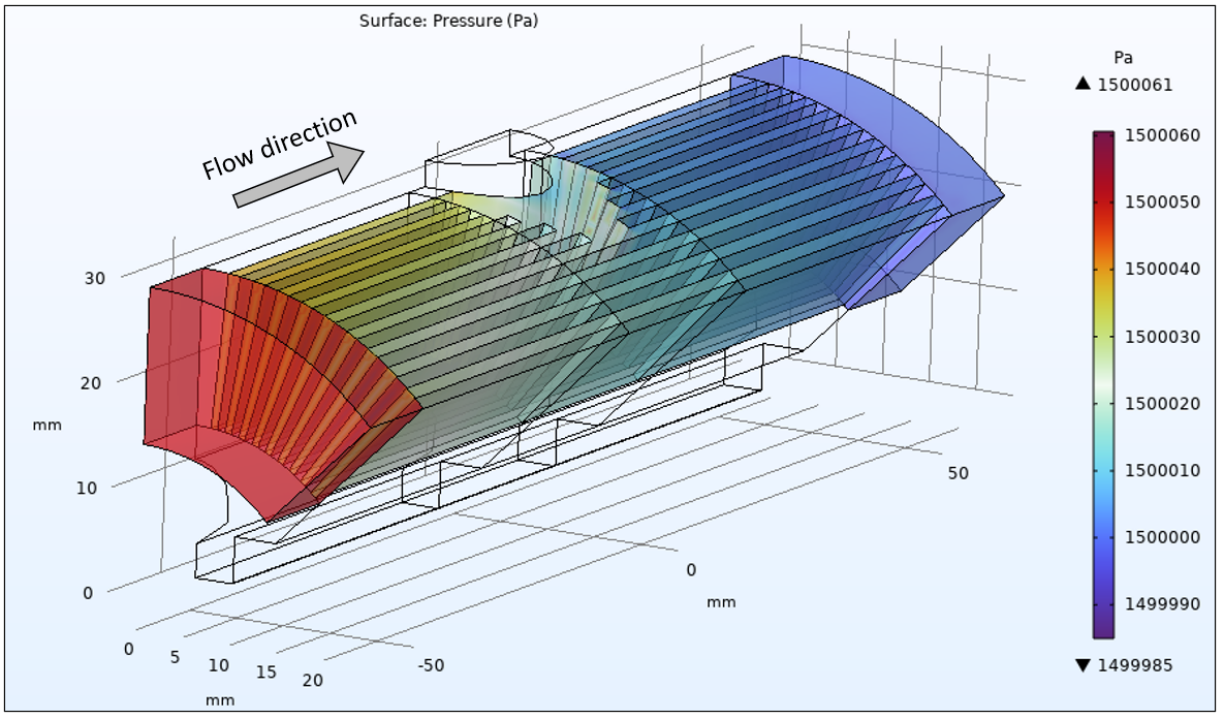


Figure 4.12: Pressure field in helium along the length of the heat sink.

Since the temperature at the current lead base was still too high, above 54 K, two copper plates were attached near the current lead base to dissipate the heat load and lower the temperature at the base. This modification is shown in Figure 4.13. This modification resulted in a lower temperature at the current lead base at the expense of a higher pressure drop. The general temperature distribution over the modified heat sink is shown in Figure 4.14 along with the new pressure field in Figure 4.15. Here, the pressure field shows a similar trend as the one in Figure 4.12 with a more pronounced difference in the static pressure

between the central and outer channels at the inlet. In comparison to the previous simulation, the pressure drop increased to 79 pa, while the temperature at the base of the current lead dropped below 50 K as seen in Figure 4.16. This decrease in temperature resulted by connecting all the fins to the current lead base instead of just the central thick fin. This significantly reduces the thermal path and therefore the thermal resistance between the current lead base (source of primary heat load) and the other axial fins. Another note to make regarding the curved plate heat sink design is the difference in thickness between the thicker curved copper plate at the top versus the thinner curved copper plate placed below. This is because the temperature at the junction between the current lead and the heat sink was higher than the temperature further down. This required a bigger cross-sectional area in contact with the current lead at the junction.

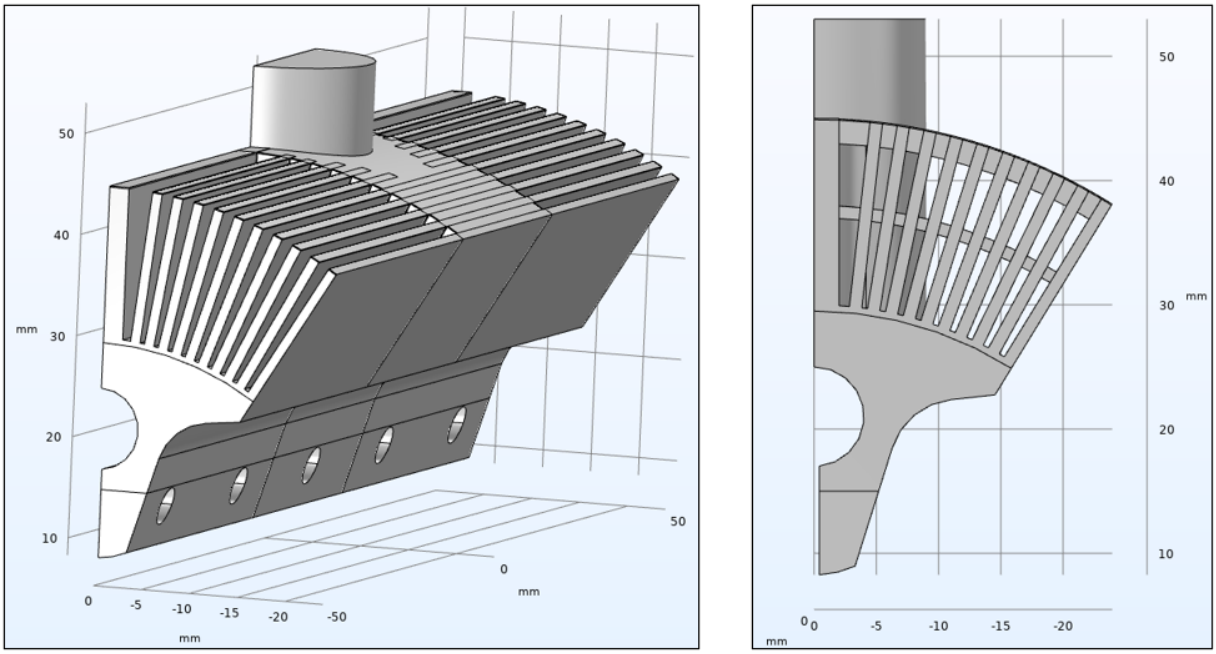


Figure 4.13: Curved copper plates added to the heat sink.

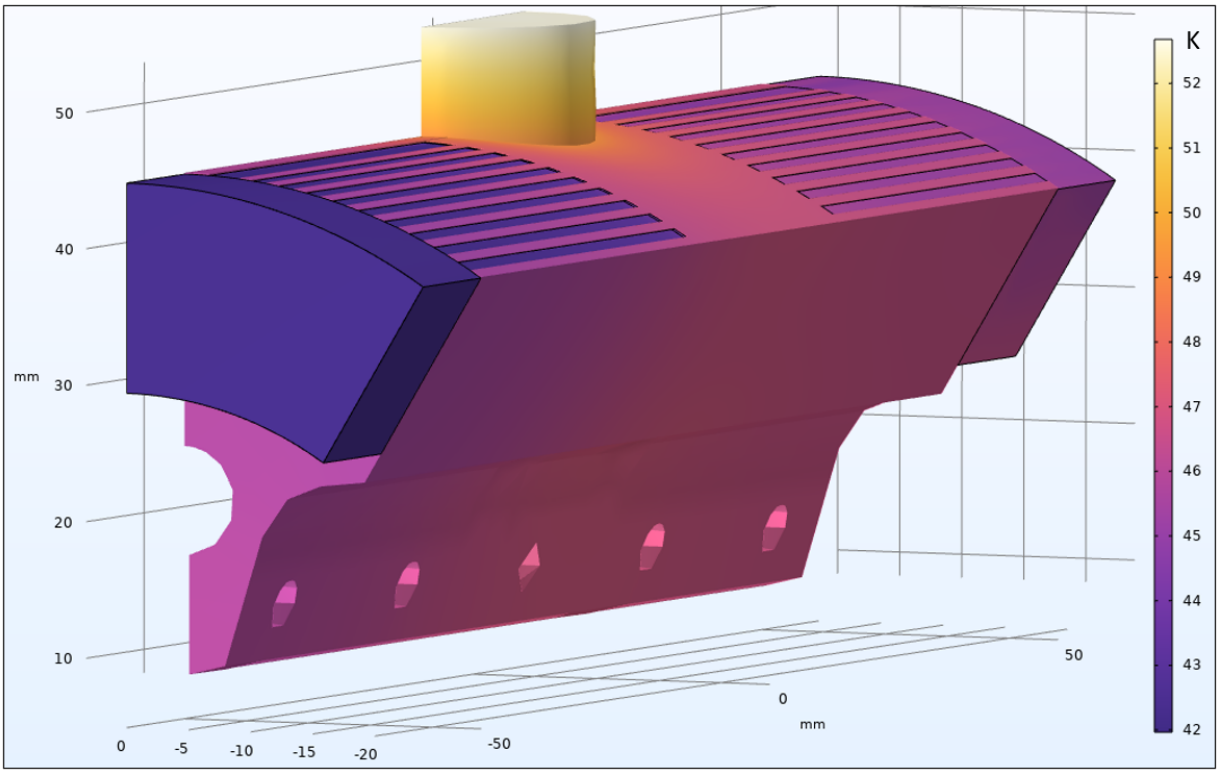


Figure 4.14: General temperature distribution over the heat sink with curved copper plates.

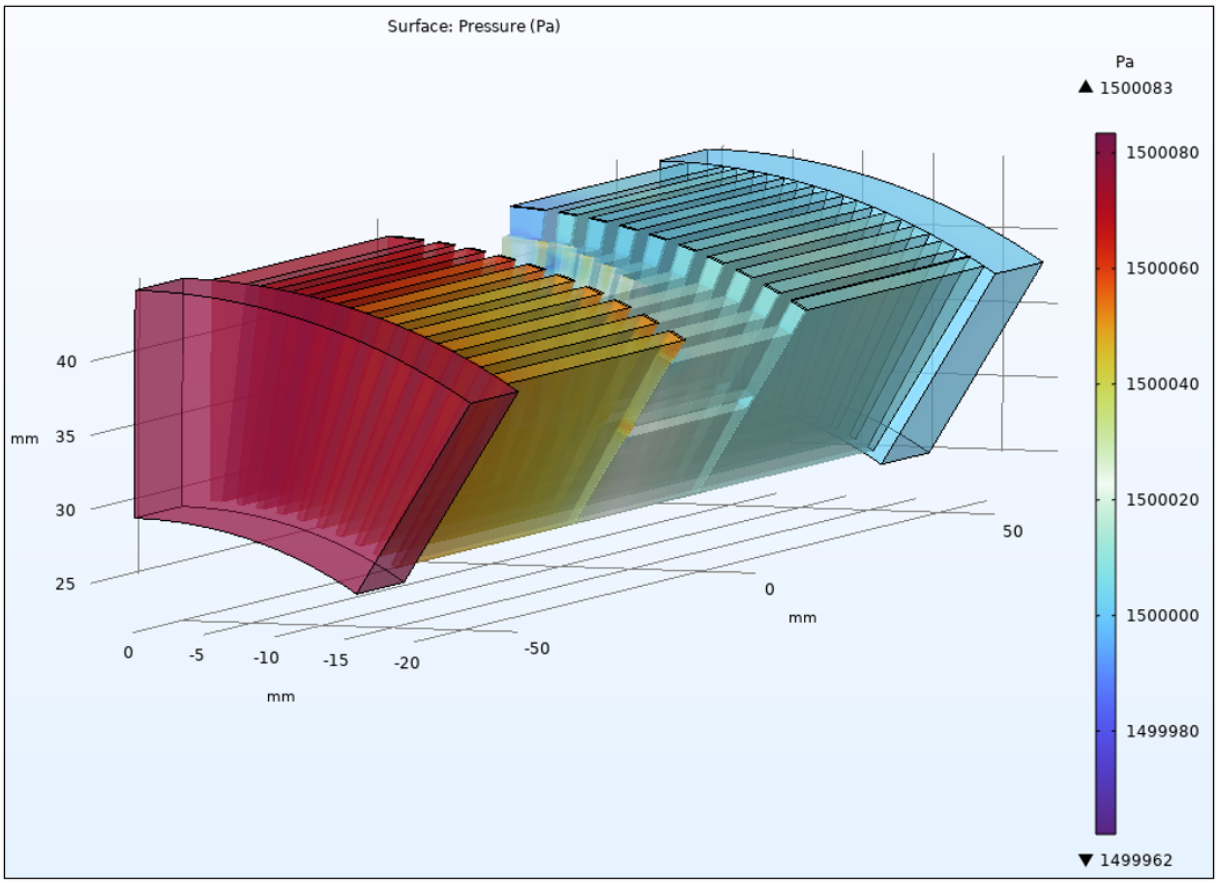


Figure 4.15: Pressure field along the heat sink with curved copper plates.

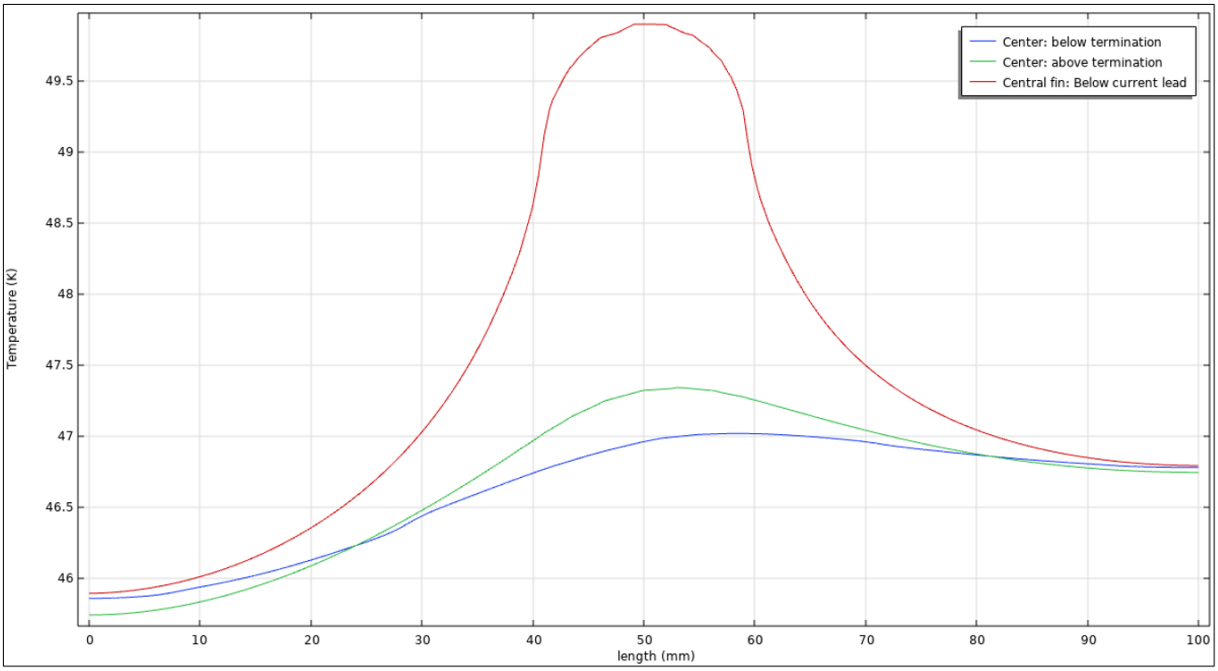


Figure 4.16: Temperature profiles along the length of the heat sink with curved copper plates.

As the idea of adding curved plates to the heat sink lowered the temperature at the base of the current lead, the decision was taken to move away from the axial fins and onto the concentric fin design. Here, the distance of the outermost fin to the core of the cylindrical heat sink was kept the same as the fin height from the axial fin design at 15 mm. Figure 4.17 shows the modified concentric fin geometry of the heat sink with the locations for temperature probes.

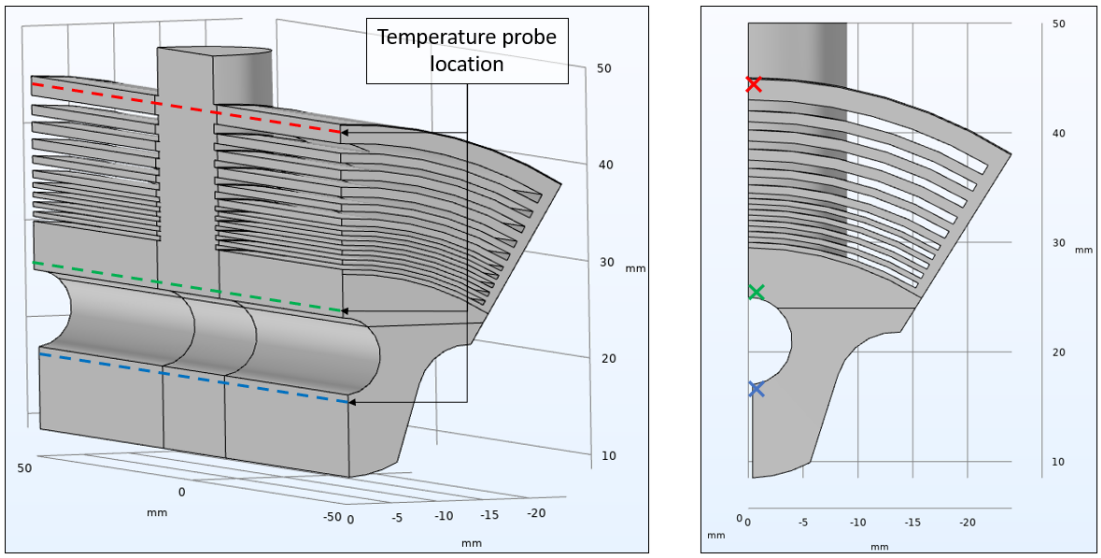


Figure 4.17: The CAD model of the heat sink with concentric fins design and locations of temperature probes.

It can be seen in Figure 4.17 that the fin spacing and thickness vary with increasing radius. This was done on purpose to have thicker fins towards the top where the current lead enters the heat sink, as its the warmest location in contact with the heat sink. Similarly, the fin spacing was also increased as the radius increased to account for the higher pressure drop between the outer fins due to higher temperature and hence a lower fluid density. This ensured a more uniform distribution of the mass flow rate of helium in all the channels (between two adjacent fins).

This concentric fin heat sink was subjected to the same boundary conditions as the previous models, and resulted in the temperature distribution seen in Figure 4.18. The temperature at the base of the current lead reduced even further to just over 48.6 K, while the peak termination temperature fell to 46.5 K as seen in Figure 4.19.

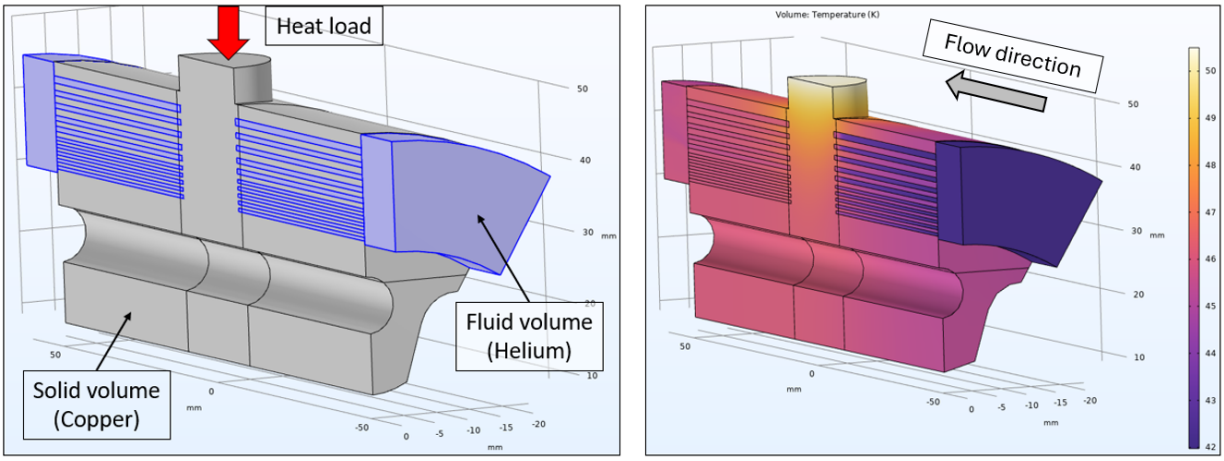


Figure 4.18: The CAD model of the concentric fins heat sink with solid and fluid domain on left and the general temperature distribution on right.

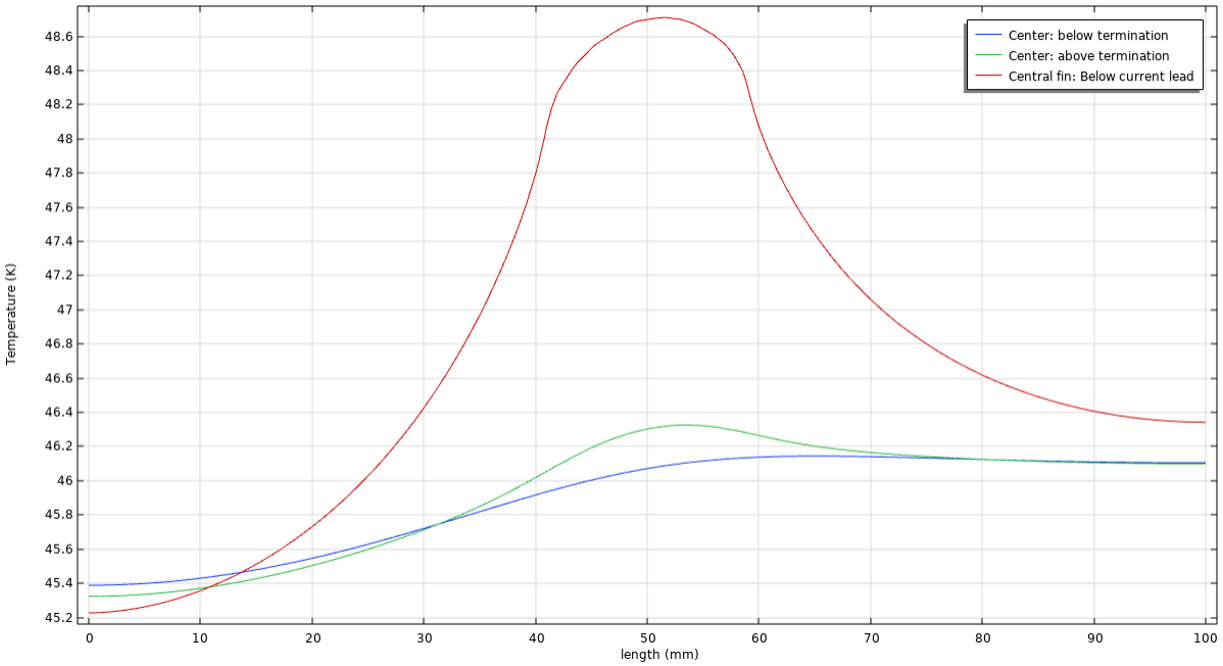


Figure 4.19: Temperature profiles along the length of the concentric fin heat sink.

In addition to the reduction in temperature, the temperature difference between the inlet and the outlet of the termination on the heat sink is now below 1K, satisfying the previously set requirement in Section 3.1.1. Furthermore, the pressure drop from the inlet to the outlet of the heat sink also decreased to 66 Pa. This can also be seen in the pressure field across the heat sink in Figure 4.20. Figure 4.20 also shows a difference in the static pressure field between the flow channels closer to the core, with a higher static pressure than the channels radially outwards (closer to CL base). This means that the velocity of helium is higher in these outer channels near the base of the current lead, allowing for a higher convective heat transfer coefficient between the fins and helium and lowering the overall temperature of the heat sink.

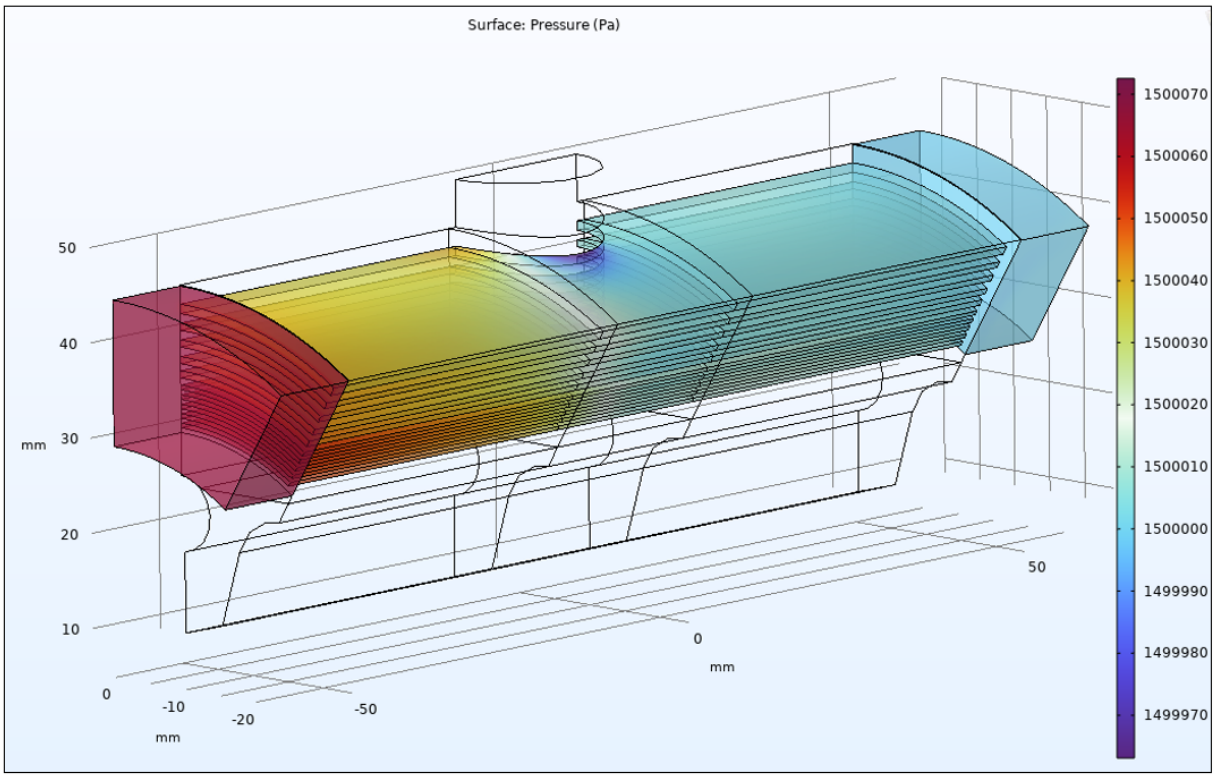


Figure 4.20: Pressure field along the heat sink with concentric fin design.

The next step was to simulate the entire termination together, including the heat sink, the current lead, and the busbar at room temperature. Here the constant heat load of 320 watts was removed and two different branches of simulation were created; the first branch forced a fixed temperature boundary condition onto the busbar, the second branch removed this fixed temperature boundary condition and placed a natural convection boundary condition on the busbar with a room temperature of 293.15 K, along with radiation on all the surfaces of the busbar. In total four fixed temperature simulations and seven natural convection simulations were performed. An overview of the fixed temperature boundary condition simulations and natural convection boundary condition simulations is presented in Table 4.2 and Table 4.3. Figure 4.21 shows the temperature distribution for Sim_convec_1, a simulation of a superconducting termination with a current lead length of 30 cm and 6.25 mm in radius, L/A of 24.45, carrying 1500 amps of nominal current.

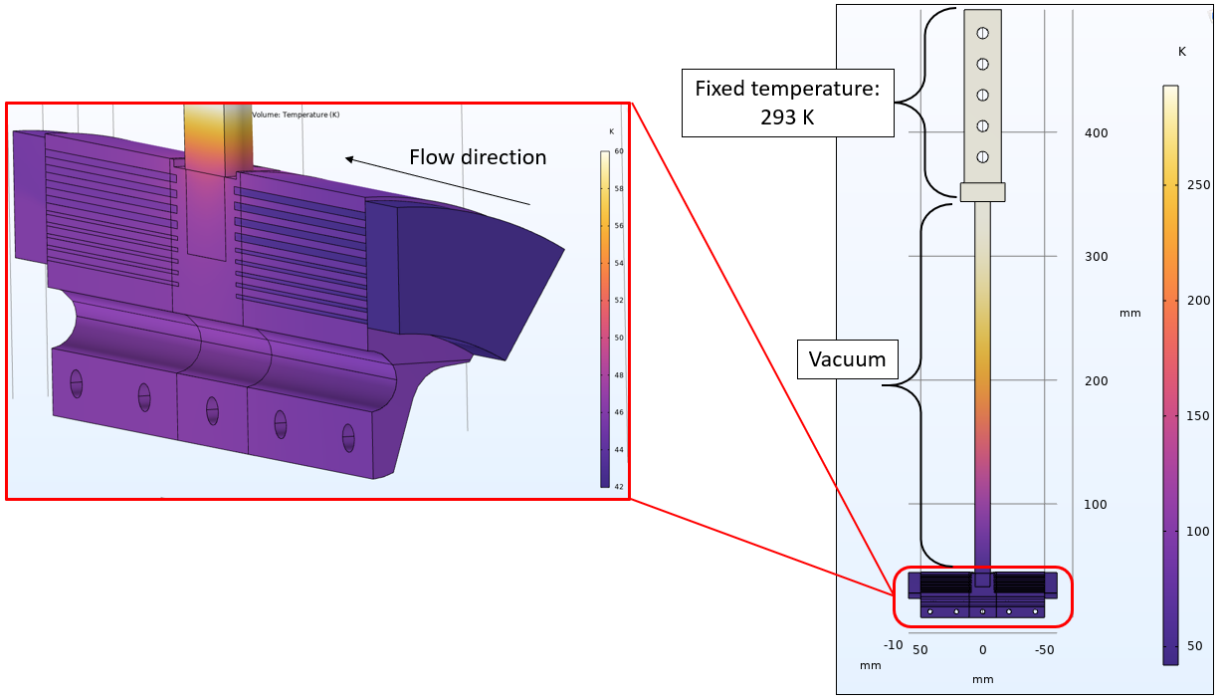


Figure 4.21: Temperature distribution over a superconducting termination carrying 1500 amps.

Much like the previous simulations, the temperature profile at the three critical locations (as seen in Figure 4.18) has been plotted in Figure 4.22. This temperature profile looks similar to Figure 4.19 with a slightly sharper peak for the temperature along the top of the outermost fin and a termination temperature that is approximately 0.3 K lower. Here, the current lead was set to have the properties of copper RRR-10, while the heat sink was set to have the properties of copper RRR-30. The pressure drop through the heat sink remained almost the same at 60 Pa. This can be seen in the pressure field through the heat sink in Figure 4.23. While the inlet fluid temperature is set to 42 K, the average outlet fluid temperature in this simulation was equal to 45.12 K. With a working fixed-temperature boundary condition model in place, parameters such as the current and helium mass flow rate were varied to observe the sensitivity and performance of the heat sink in off-design operation. All other results for simulations with fixed-temperature boundary conditions are presented in Appendix A. A summary of their results is presented in Table 4.2 and Table 4.3.

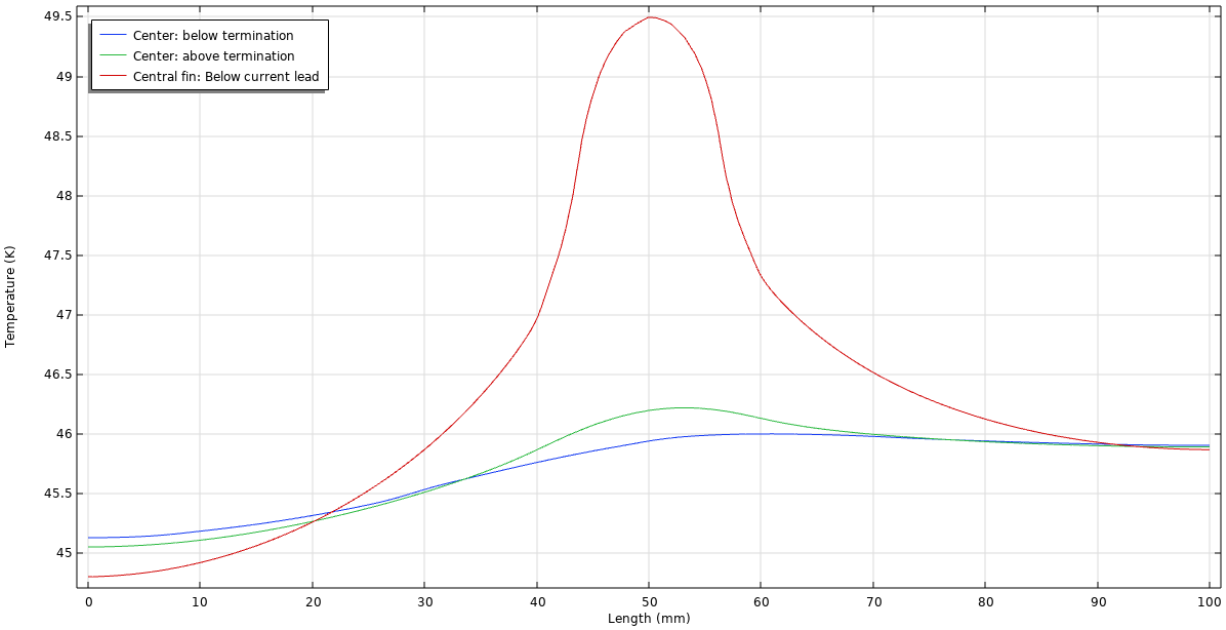


Figure 4.22: Temperature profiles along the length of the concentric fin heat sink with a fixed temperature condition on the busbar.

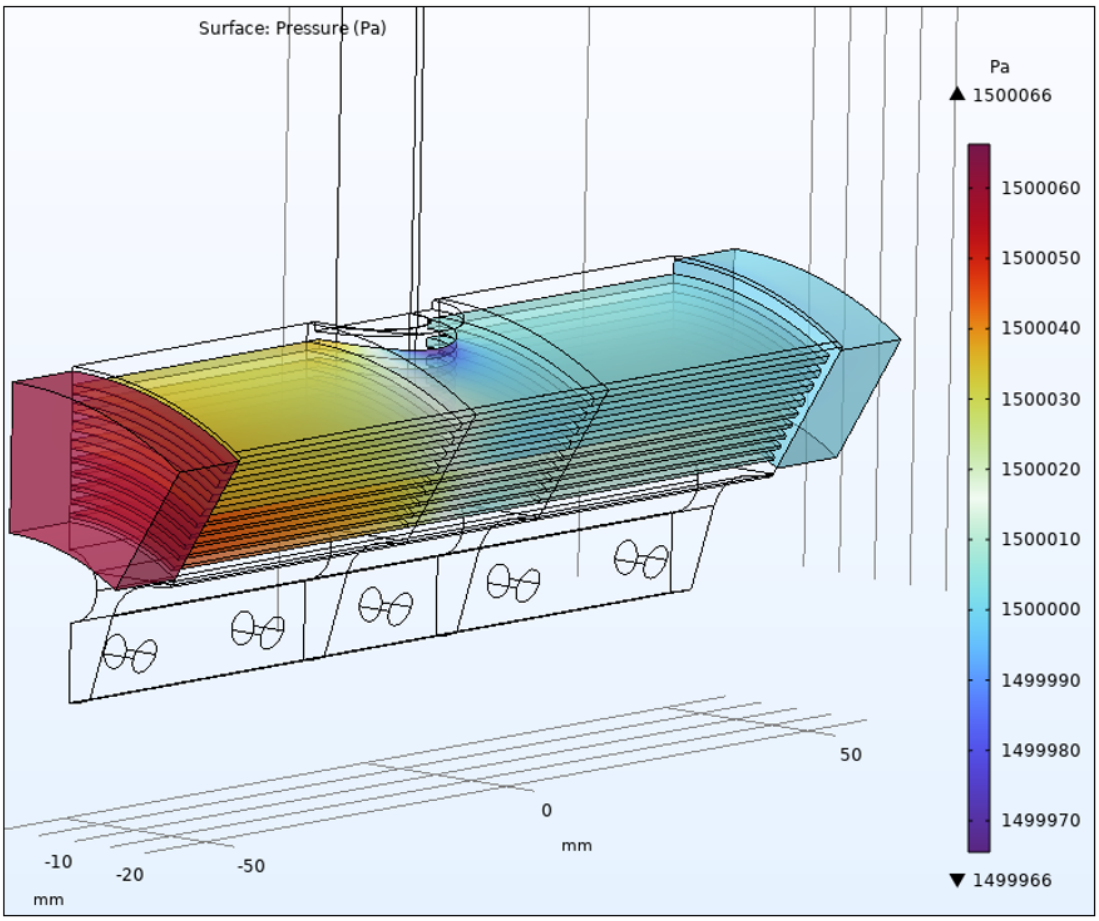


Figure 4.23: The pressure field in the concentric fin heat sink with a fixed temperature condition on the busbar.

After the fixed-temperature boundary condition simulations, the second branch of simulations dealing with the natural convection and radiation boundary condition was investigated. Sim_convec_1 simulation is the closest representation to the real-life nominal operation of the heat sink. Here, the busbar is subjected to natural convection with the environment at 293.15 K, all while radiating heat to the environment through its surface. Figure 4.24 shows the temperature distribution for Sim_convec_1.

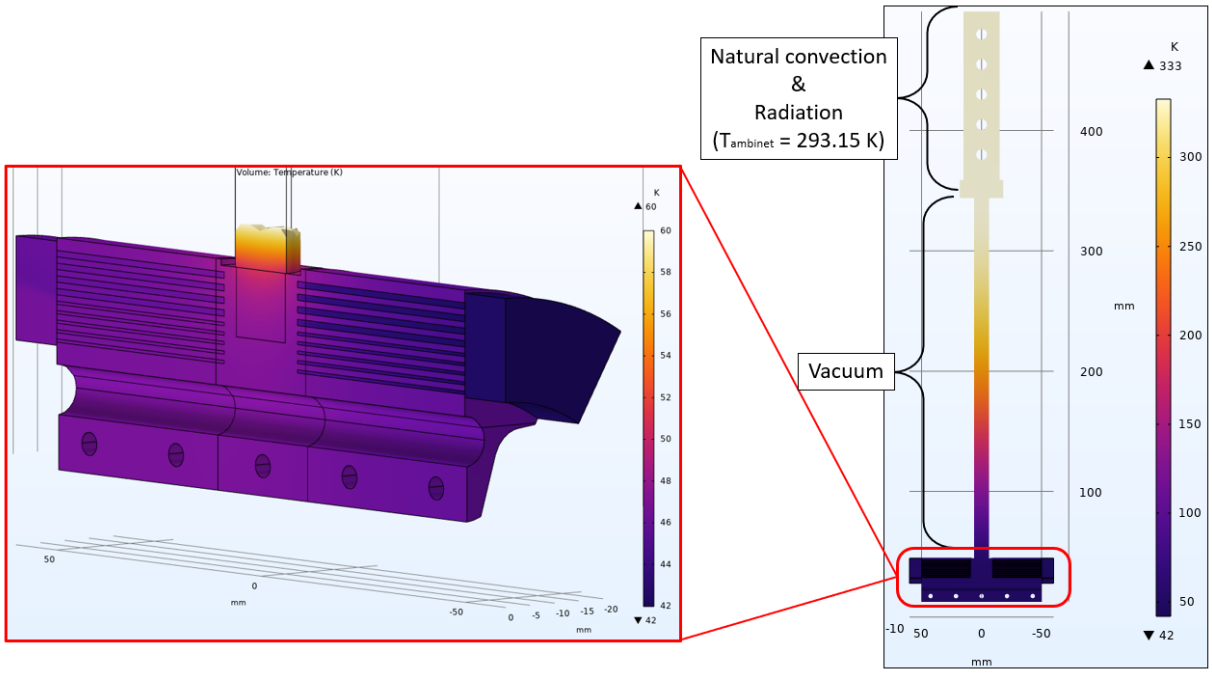


Figure 4.24: Temperature distribution for Sim_convec_1 simulation.

It can be seen here that since a fixed-temperature boundary condition is no longer imposed on the busbar, it rises in temperature upto 333 K. This increase in temperature on the busbar leads to an increase in temperature at the base of the current lead and at the termination where the HTS is connected to the heat sink. This is visible in Figure 4.25. Furthermore, the outlet fluid temperature is also increases to 45.55 K from the previously shown 45.12 K in $Sim\_T\_fix\_1$. The pressure drop across the heat sink however is not significantly affected and stays at 60 Pa. The pressure field across the heat sink for Sim_convec_1 can be seen in Figure 4.26. Results for the rest of the $Sim\_convec\_$**x** simulations are presented in Appendix A.

An important point to note regarding all the simulations presented so far and shown in Table 4.2 & Table 4.3 is that they were performed with perfectly smooth wall surfaces. However, when the heat sink was 3D printed in copper at PROTIQ (Germany), the manufacturer informed that the surface roughness of the fins in the heat sink is between 80 μm and 120 μm. Hence the Sim_convec_1 simulation, being the closest to reality, was rerun on another COMSOL license which had the option of wall roughness. Here the walls in contact with the fluid were set to have a surface roughness of 120 μm. All other boundary conditions were left unchanged. This resulted in a temperature distribution shown in Figure 4.27. The average outlet fluid temperature of helium was at 45.5 K while the new pressure drop across the heat sink was now 90 Pa. The pressure field for this new simulation with a wall roughness of 120 μm is presented in Figure 4.28. Finally, the temperature profile along the length of the heat sink at the three critical locations as seen in Figure 4.17 is presented in Figure 4.29.

Table 4.2: An overview of the fixed temperature and natural convection boundary conditions (Part 1: McFee, CL, and COMSOL)

McFee Optimization						CL Integration Design				COMSOL Simulations
Lower Temp.	Higher Temp.	Design Nominal Current	CL Material	McFee Optimum Power	McFee Optimum L/A	CL Integration Length	CL Cross-section Area	CL Radius	CL Mass	Heat Sink Version
T_L [K]	T_H [K]	I_nom [A]	RRR	Q_L [W]	L/A [cm^-1]	L [m]	A [mm^2]	R [mm]	M_cl [kg]	Name
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_T_fix_1
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_T_fix_2
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_T_fix_3
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_T_fix_4
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_convec_1
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_convec_2
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_convec_3
50	293	2000	CU-RRR-10	90.2	18.34	0.3	163.6	7.216	0.44	Sim_convec_4
50	332.82	1500	CU-RRR-10	77.15	24.46	0.3	122.63	6.248	0.33	Sim_convec_5
50	293	2000	CU-RRR-10	90.2	18.34	0.3	163.6	7.216	0.44	Sim_convec_6
50	293	1500	CU-RRR-10	67.65	24.44	0.3	122.73	6.25	0.33	Sim_convec_7
	Copper desnity	8960	kg/m3		Inlet fluid temp.	42	K			

Table 4.3: An overview of the fixed temperature and natural convection boundary conditions (Part 2: McFee, CL, and COMSOL)

COMSOL Simulations								
Heat Sink Version	Boundary Condition: Hot End	Applied Current	Helium Flow	CL-Heatsink Junction Temp.	Peak Terminal Temp.	Peak CL Temp.	Heat Sink ΔP	Helium Outlet Temp.
Name		I_app [A]	m_he [g/s]	T_cl-heatsink	T_ter [K]	T_peak [K]	Δ P [Pa]	T_outlet
Sim_T_fix_1	Fix Hot Temp.	1500	18	49.87	46.22	293.21	60	45.12
Sim_T_fix_2	Fix Hot Temp.	2000	18	54.9	48.73	348.22	60	46.96
Sim_T_fix_3	Fix Hot Temp.	1500	10	52.58	48.75	293.35	21	47.5
Sim_T_fix_4	Fix Hot Temp.	2000	10	59.7	52.89	351.95	21	50.82
Sim_convec_1	Ambient Convec.	1500	18	50.96	46.8	332.82	60	45.55
Sim_convec_2	Ambient Convec.	1500	10	54.56	49.93	344.15	21	48.45
Sim_convec_3	Ambient Convec.	2000	18	93.96	64.7	1183.6	61	58.39
Sim_convec_4	Ambient Convec.	2000	18	54.65	48.98	363.87	60	47.17
Sim_convec_5	Ambient Convec.	1500	18	51.03	46.82	333.85	60	45.56
Sim_convec_6	Ambient Convec.	1500	18	46.32	44.54	176.56	60	43.89
Sim_convec_7	Ambient Convec.	1000	18	45.27	43.8	176.67	60	43.34

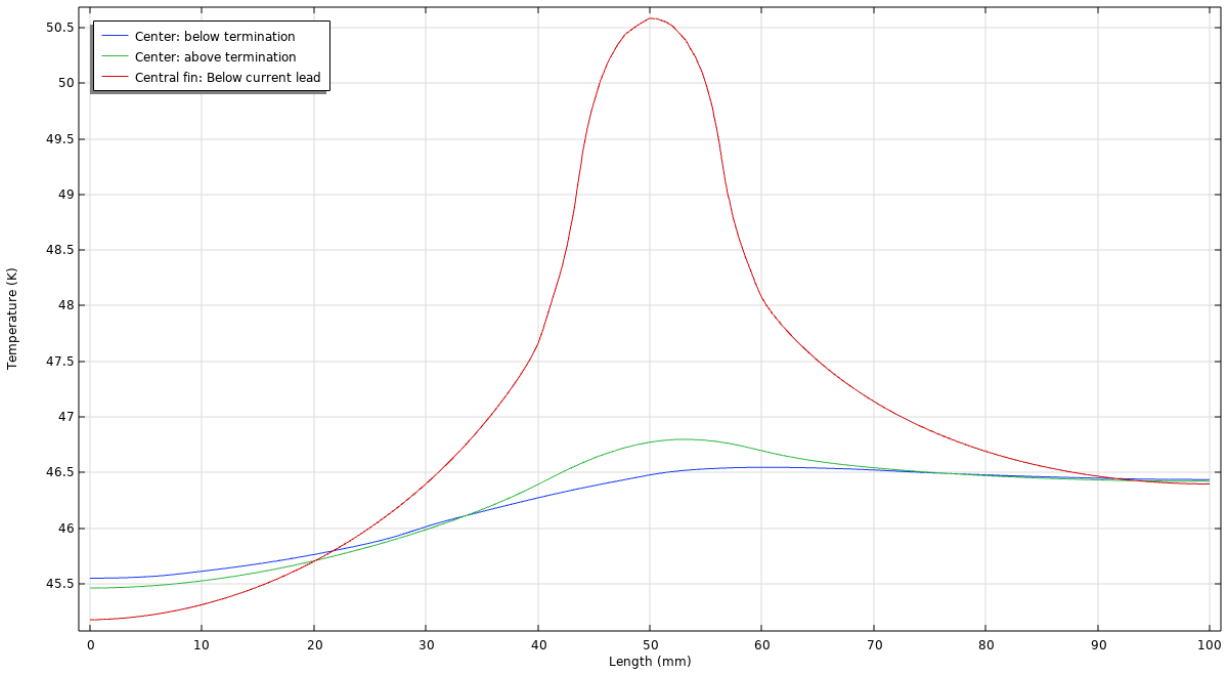


Figure 4.25: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_1 simulation.

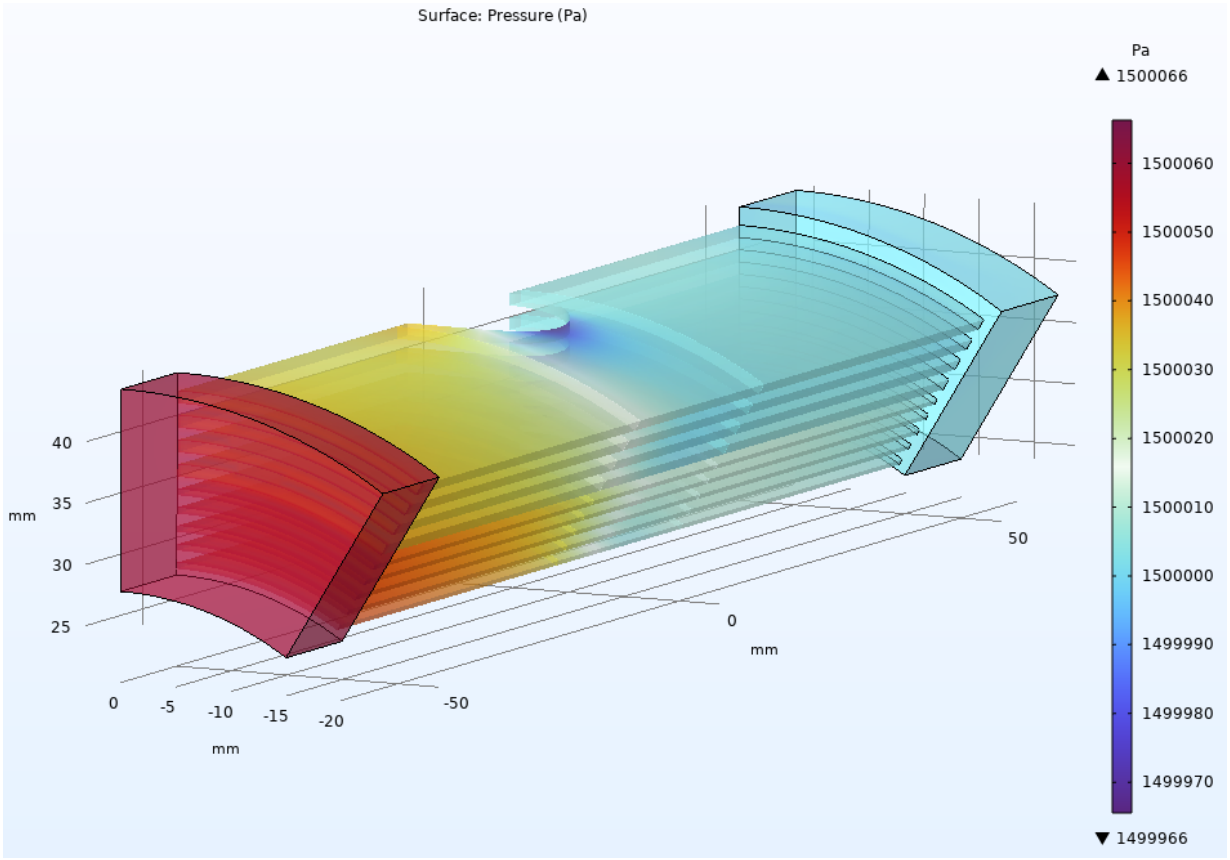


Figure 4.26: The pressure field across the heat sink for Sim_convec_1 simulation.

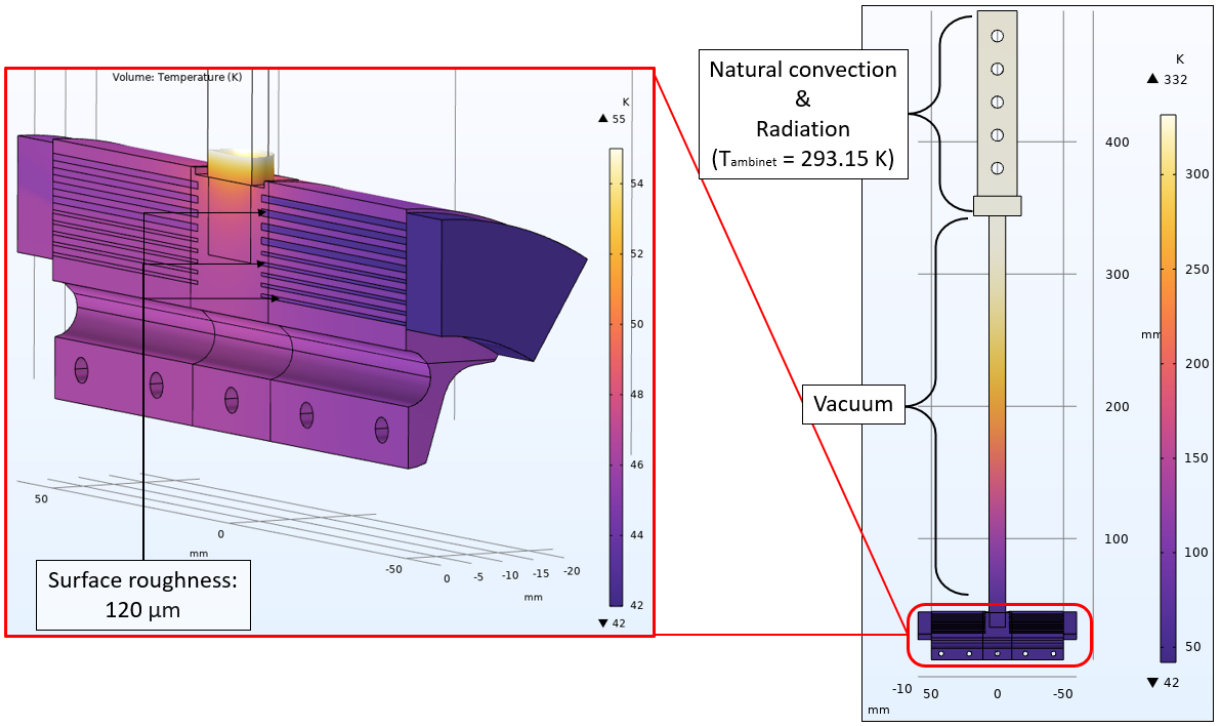


Figure 4.27: Temperature distribution for the rerun of Sim_convec_1 simulation with a surface roughness of 120 μm.

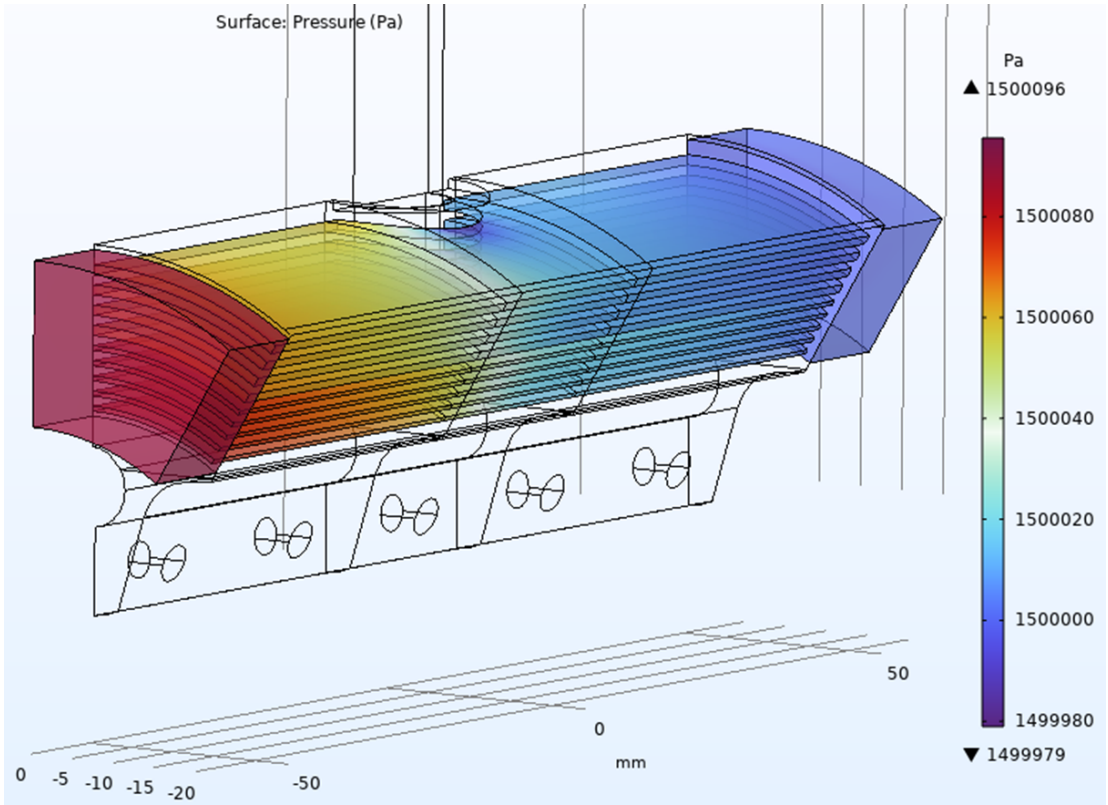


Figure 4.28: The pressure field across the heat sink for the rerun of Sim_convec_1 simulation with a surface roughness of 120 μm.

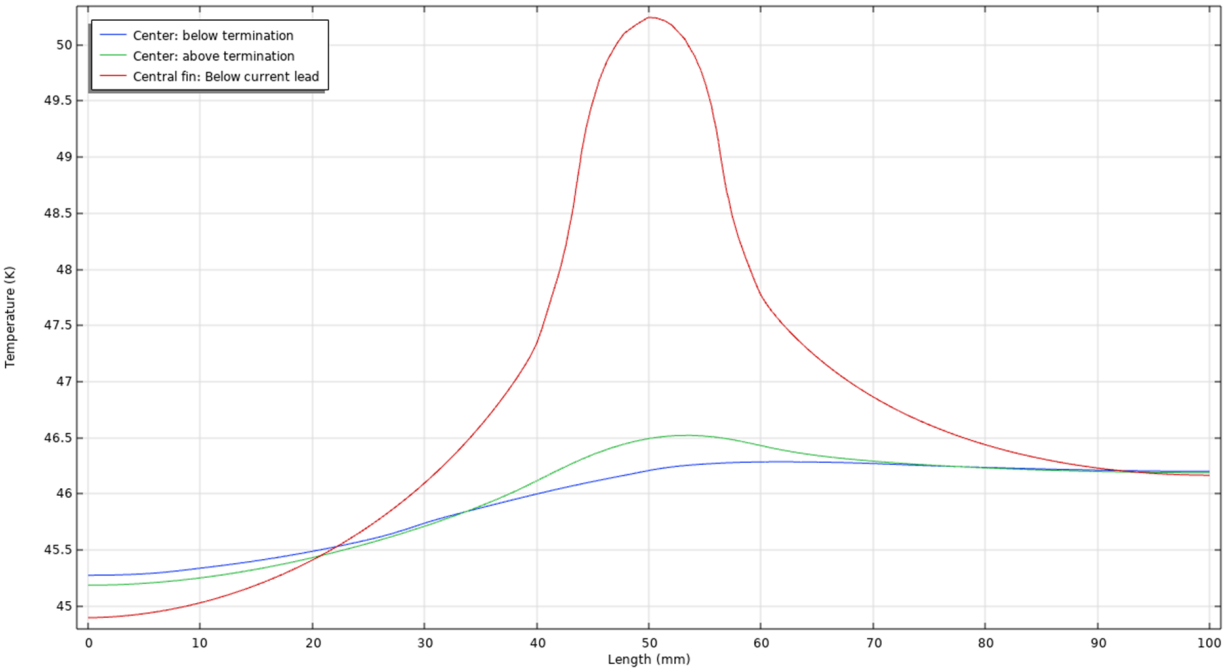


Figure 4.29: Temperature profiles along the length of the concentric fin heat sink for the rerun of Sim_convec_1 simulation with a surface roughness of 120 μm.

Since in the experimental setup, the pressure ports to measure the ΔP across the heat sink were placed before and after the inlet and outlet headers, it is important to identify the pressure drop caused by the expander (inlet header) and by the reducer (outlet header) as it would be contributing to the pressure drop reading of the heat sink. Since the expander and reducer were also 3D printed along with the heat sink at PROTIQ (Germany), a wall roughness of 120 μm was also applied to the headers in the COMSOL model. The pressure drop across the expander and reducer is presented in Table 4.4. The pressure fields in the expander and reducer are shown in Figure 4.30 and Figure 4.31 respectively. As expected, when the flow enters the expander, it has a high velocity, reducing the static pressure at the inlet. As the flow reaches the inlet of the heat sink, the flow cross-sectional area increases, lowering the velocity and raising the static pressure at the outlet of the expander. The exact opposite happens when the flow exits the heat sink into a reducer.

Table 4.4: Pressure drop caused due to the expander and reducer.

Component	**ΔP [Pa]**	**Mass flow rate [g/s]**	**Inlet Temperature [K]**	**Inlet Pressure [Bara]**
Expander (Inlet Header)	10	4.5	42	15
Reducer (Outlet Header)	29	4.5	45	15

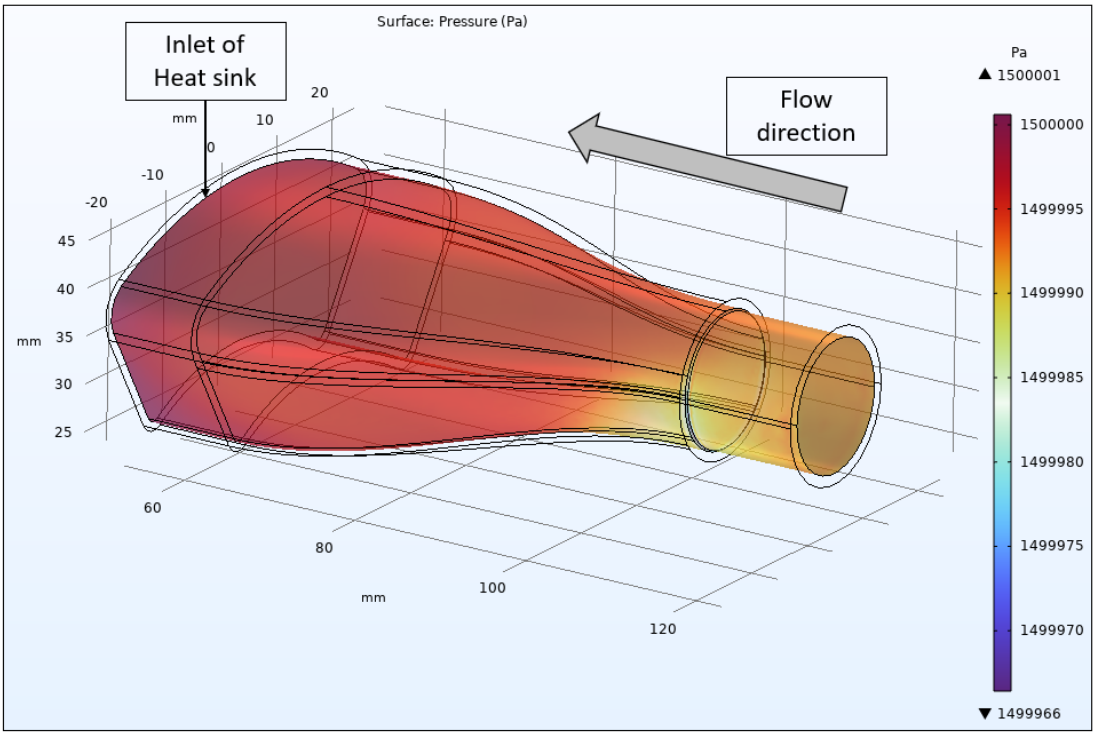


Figure 4.30: The pressure field in the expander at the inlet of heat sink with a surface roughness of 120 μm.

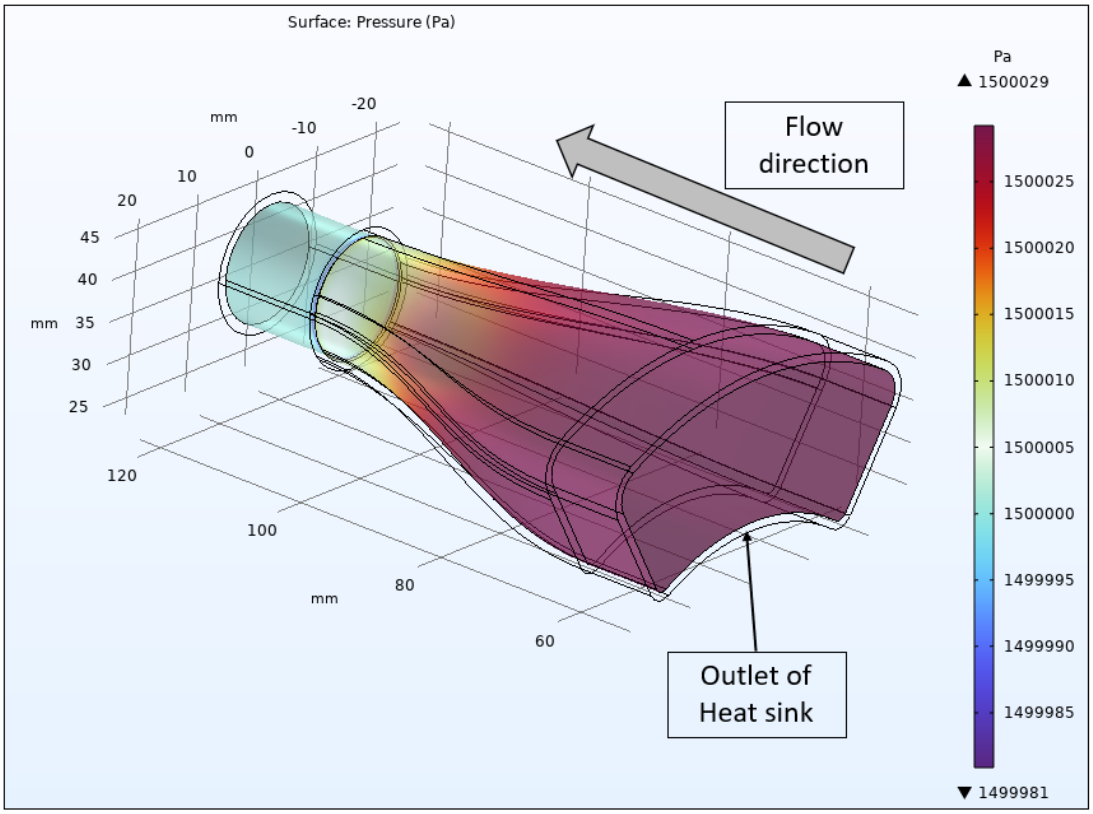


Figure 4.31: The pressure field in the reducer at the outlet of heat sink with a surface roughness of 120 μm.

The progression of the 3D COMSOL model shown in Figure 3.14 provides a quick overview of the design progression. The initial simulations with versions 1 to 3 immediately revealed critical flaws. The placement of the CL base (the primary heat source) blocked central flow channels, creating areas of high thermal resistance and fluid stagnation. Disconnecting the axial fins from the current lead base to allow for a smooth flow through the central channels lead to a longer thermal path between the heat load and the fins. This resulted in the temperature at the CL base (Figure 4.11) exceeding 53 K, and a peak HTS termination temperature of 49 K, providing very little margin for any off-design scenarios. To address this, iterations included the addition of curved copper plates, which successfully lowered the base and termination temperature in Figure 4.16. Since the new pressure drop was still in the budget and the temperature at the current lead base and termination was significantly reduced, a decision was taken to follow this trend and extend this idea of copper curved plates in a concentric fin design.

The shift to a concentric fin design was the most significant design decision. This geometry directly addressed the primary heat source, the CL base, by arranging the cooling surfaces radially from the heat source. This ensured that the heat path from the CL base to the wetted surfaces was short and efficient. The results in Figure 4.19 confirm the success of this topology, where the temperature at the current lead base dropped to 48.7 K, and the peak HTS termination temperature dropped to 46.3 K comfortably meeting the < 50 K requirement. Furthermore, the temperature difference along the HTS termination was well below the 1 K limit, ensuring thermal stability for the superconductor, and finally, a ΔP of 66 Pa was well within the 100 Pa budget. This iterative process demonstrates that for this application, optimizing the path of heat from the source to the fluid was more critical than simply maximizing the fluid-wetted surface area.

The sensitivity study on the heat sink provided in Table 4.2 and Table 4.3 gave important insights into the system's robustness in off-design conditions and its operational boundaries. Comparing $Sim\_T\_fix\_1$ (1.5 kA) to $Sim\_T\_fix\_2$ (2.0 kA) showed a 5 K jump in CL base temperature (49.9 K to 54.9 K). This confirms the design's high thermal sensitivity to current. More importantly, it demonstrates that a 33% over-current event would push the HTS termination temperature to 49 K, almost violating the 50 K requirement. This has significant implications for the aircraft's power electronics, suggesting the need for a fault current limiter (FCL), as mentioned in the literature review (Figure 2.5), to protect this junction.

Similarly, comparing $Sim\_T\_fix\_1$ (18 g/s) to $Sim\_T\_fix\_3$ (10 g/s) resulted in a 2.5 K temperature rise (46.22 K to 48.75 K). This highlights the system's strong dependence on the cryogenic cooling system's performance. A 44% reduction in coolant flow brings the termination temperature to its limit, indicating a strong dependence of the DC link subsystem on the cryogenics teams' designs. The impact of the surface roughness was another key finding of the design. Rerunning Sim_convec_1 with a 120 μm roughness (to match the 3D-printed part) caused the heat sink's ΔP to increase by 50% (60 Pa to 90 Pa). This single manufacturing parameter pushed the design to the very edge of its 100 Pa requirement. This demonstrates that the current state of additive manufacturing, not just the CAD geometry, is a primary performance driver. The roughness also slightly improved heat transfer (peak termination temperature dropped 0.3 K, Figure 4.29), an expected result of increased turbulence, but the penalty in pressure drop was far more significant.

4.3. Experimental results

This section is split into two parts. Section 4.3.1 provides the results for the test(s) conducted on October 22nd of October 2025 for a cryofan speed of 8036 rpm, while Section 4.3.2 gives the experimental results for the second day of testing at cryofan speeds of 13100 and 17600 rpm. The main goal of the tests was to investigate the thermal performance of the heat sink under different heat loads and coolant mass flow rates. This was done to understand how the heat sink would perform under design and off-design conditions.

4.3.1. Heat sink test for a cryofan speed of 8036 rpm

The first test run started with a cool-down of the system with a cryofan speed of 8036 rpm. This was the lowest possible speed setting on the Sterling cryocooler. Figure 4.32 shows the temperature at temperature ports 1, 4, and 5 of the heat sink for the cool down and various heating scenarios. The port numbers correspond to the temperature ports indicated in Figure 3.23, where port 1 is the first port on the heat sink at the inlet and port 5 is the last port on the heat sink at the outlet. The heating scenarios are zoomed in Figure 4.33 to get a clearer view of the temperature at different ports. The measurements for each heat load scenario were stopped once the temperature change in all the ports was less than 0.1 K over the

span of the last 10 minutes (except for the 10 W heat load scenario). In addition to the temperature plots, Table 4.5 provides an overview of the vacuum level at varying time stamps in the cryostat within which the testing setup was placed. It is important to note that there were small leaks in the setup, hence why the cryostat was put under active vacuum pumping. The quantity of leak changed with setup temperature, leading to the fluctuation in the vacuum level of the cryostat observed in Table 4.5. This imperfect vacuum allows for heat leaks into the setup via the collisions and diffusion of the gas molecules present in the cryostat. Even though the cryostat is said to be in high vacuum based on the ISO 3529-1:2019 norm [91], the conduction through gas molecules becomes more important than radiation in the 10^{-3} mbar vacuum range [92]. An estimate for the heat leak due to conduction in gas can be obtained through the Kennard equation given in Equation 4.1 in W/m2 [93].

$$q_{gas} = \alpha_{helium} \cdot \left(\frac{\gamma+1}{\gamma-1}\right) \cdot \sqrt{\left(\frac{R}{8 \cdot \pi \cdot M}\right)} \cdot \frac{\Delta T}{\sqrt{T_g}} \cdot p \tag{4.1}$$

Here, the coefficient α relates the degree of thermal equilibrium between the gas and the wall. Its value ranges from α ≤ 0.5 for helium, to α = 0.78 for argon and α = 0.78 for nitrogen [93]. To be conservative, α = 0.5 was chosen. Next, γ is the ratio of specific heats (c_p/c_v) of the gas, R is the universal gas constant at 8.314 $J/(mol \cdot K)$, M is the molar mass of the gas in kg/mol (0.004 kg/mol for helium), T_g is the temperature of the gas in the vacuum, and P is the vacuum pressure. Since the setup had small leaks and the cryostat was under active vacuum pumping, it is safe to assume that the air inside the cryostat was replaced with the leaking helium gas. The second assumption required for the estimation of the heat leak due to conduction in gas is the temperature of the gas inside the cryostat vacuum. Since no temperature sensors were used to measure the temperature inside the vacuum, it is assumed that the temperature of helium molecules inside the vacuum is the average of the temperature of the copper test setup and the cryostat wall. Even though the temperature across the entire copper setup varies with the location, an average temperature of 45 K is assumed for the sake of this calculation, and the temperature of the cryostat wall is assumed to be the room temperature at 290 K. This results in a temperature of 167.5 K for the helium molecules inside the vacuum. Although the vacuum level ranges from 2 x 10^{-3} to 9 x 10^{-3} mbar across all the tests as seen in Table 4.5, Table 4.7, and Table 4.6, the change in γ for both these vacuum levels is negligible. Therefore, a γ_{helium} of 1.66 at 167.5 K and 9 x 10^{-3} mbar pressure was obtained from NIST [78]. This results in a heat flux of approximately 69 W/m^2K at 2 x 10^{-3} mbar upto 312 W/m^2K at 9 x 10^{-3} mbar. The external surface area of the heat sink test assembly shown in Figure 3.15 along with the venturi meter is equal to 0.067 m^2 (calculated from CAD geometry). This results in a heat load due to an imperfect vacuum of 4.6 W for 2 x 10^{-3} mbar and 21 W for 9 x 10^{-3} mbar. This is a significant percentage of the heat leak onto the setup and is most likely the source of the discrepancy between the simulation and experimentally measured temperature.

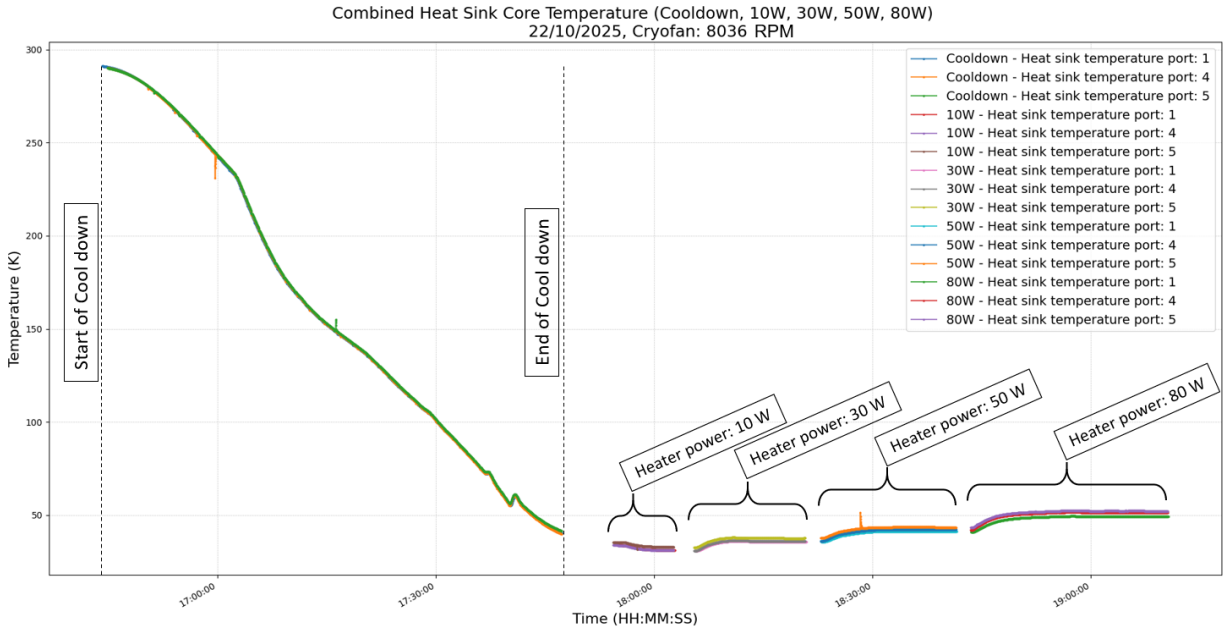


Figure 4.32: Temperature of the heat sink through cooldown and different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

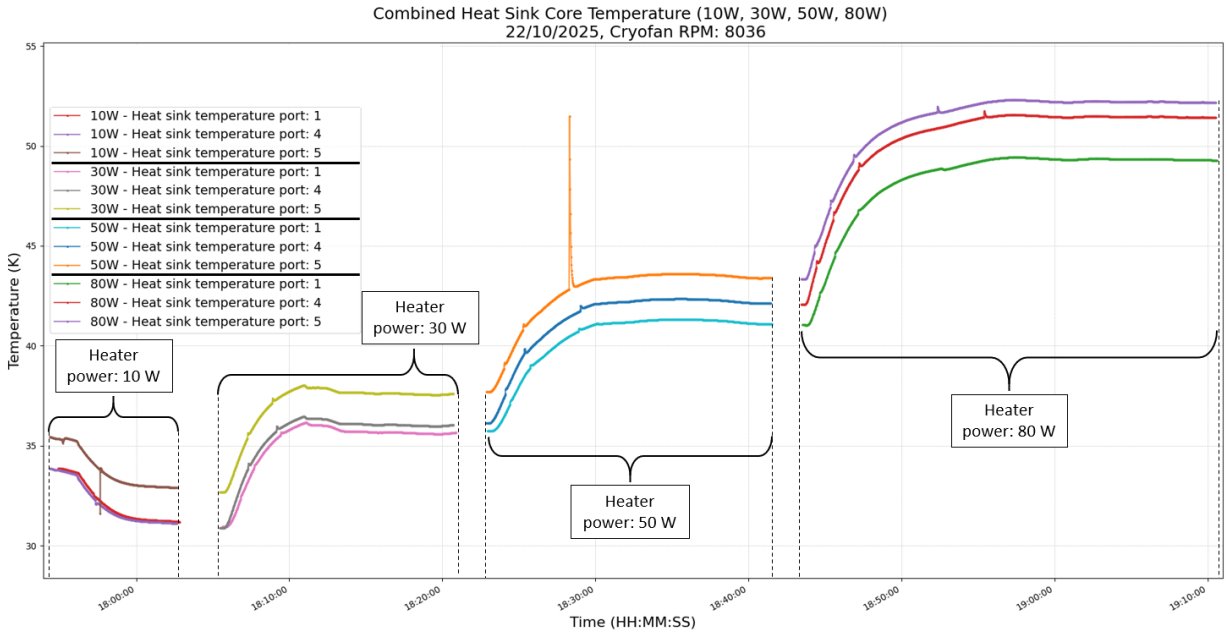


Figure 4.33: Temperature readings from temperature ports 1, 4, and 5, as shown in Figure 3.23, under different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

Table 4.5: Vacuum level in the cryostat over time. [22/10/2025, Cryofan: 8036 rpm]

Time [HH:MM:SS]	Vacuum level [mbar]
17:45:00	5.4 x 10^{-3}
18:06:00	9 x 10^{-3}
18:19:00	7.7 x 10^{-3}
18:23:00	7.7 x 10^{-3}
18:44:00	7.12 x 10^{-3}
19:07:00	5.86 x 10^{-3}

The differential pressure readings were extracted from the raw sensor voltage readings. These raw voltage readings are presented in Appendix A. The pressure drop across the heat sink and the venturi meter is shown in Figure 4.34 along with a ten second moving average of the data in red. The venturi meter was calibrated in house and the coefficient of discharge was found to be between 0.95 & 1 (where it reaches a plateau) for the range of Reynolds from 15,000 to 40,000. For the mass flow rate in the range of 1.5 g/s to 6 g/s, the reynolds number at the throat of the venturimeter is found to be in the range of 35,000 to 150,000. Therefore, a constant value of 0.98 was set as the discharge coefficient for the venturimeter to be on the conservative side. The inner diameter in the throat (D_c) was 7.8 mm, and the inner diameter at the inlet of the venturi meter (D_i) was 13.1 mm. Based on the dimensions of the venturi meter, the properties of the inlet fluid, and the pressure drop across the venturi meter, the volumetric ($\dot{Q}$) and mass flow rate ($\dot{m}$) of helium can be deduced via Equation 4.2 [94]. The inlet temperature of helium is plotted in Figure 4.35 while the mass flow rate over time for the first day of test is shown in Figure 4.36.

$$\dot{Q} = \frac{\dot{m}}{\rho} = C_d \cdot \frac{\pi \cdot D_i^2}{4 \cdot \rho} \cdot \sqrt{\frac{2 \cdot \rho \cdot (\Delta P_{venturi})}{\left(\frac{D_i}{D_c}\right)^4 - 1}} \tag{4.2}$$

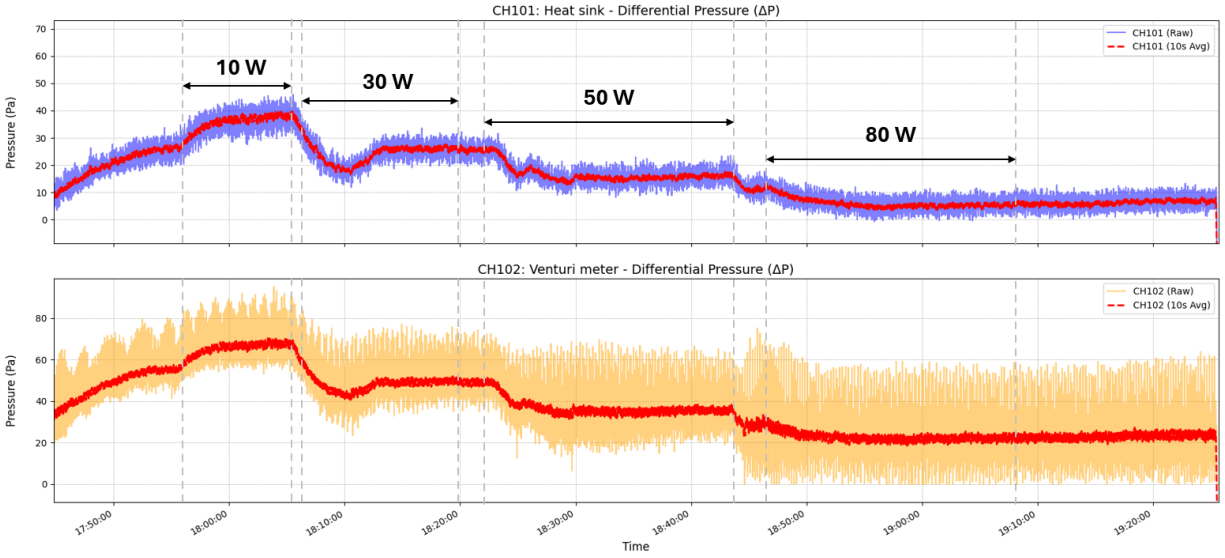


Figure 4.34: Differential pressure readings across the heat sink (CH 101) and venturi meter (CH 102) under different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

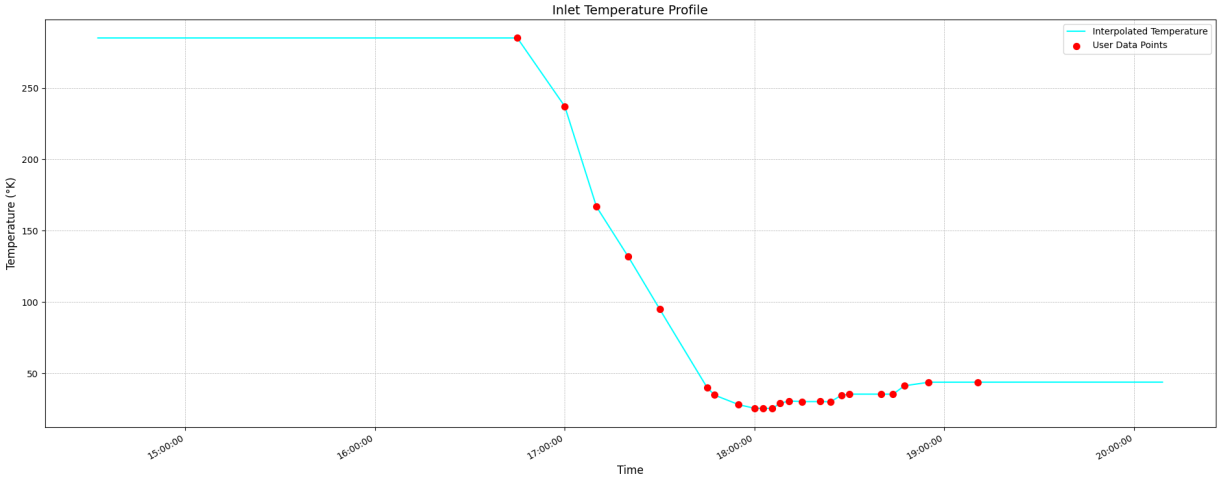


Figure 4.35: The inlet temperature of helium through cooldown and different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

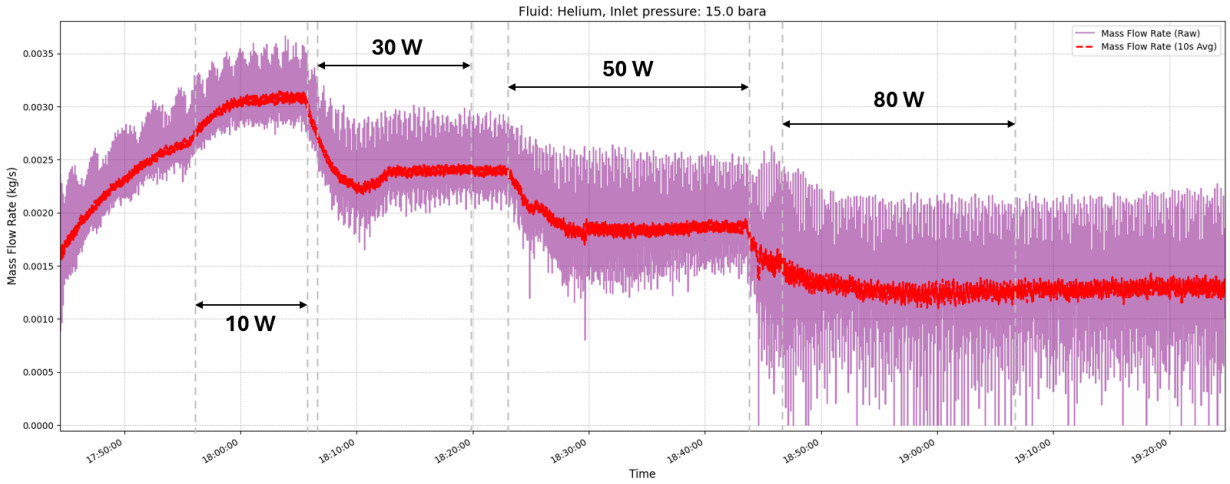


Figure 4.36: The mass flow rate of helium through the setup under different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

4.3.2. Test day 2: 23/10/2025

The second day of testing was split into two test runs. The goal was to map the performance of the heat sink across different mass flow rates. This was once again done to understand the on and off design operation of the component. Therefore, the first run was conducted at a cryofan speed of 17600 rpm, and another at 13100 rpm. The results for both cryofan speeds are provided below.

Cryofan rpm: 17600

On the second day of testing, the same procedure was followed. First, the vacuum pump was initiated, and then the cool-down began early morning. The cool-down was done at the highest speed of the cryofan at approximately 17600 rpm. The fan speed varied by ± 100 rpm over the course of the entire experiment.

The temperature readings for ports 1, 3, and 5 for varying heating scenarios are presented in Figure 4.37. The pressure drops across the heat sink and the venturi meter are also shown in Figure 4.38. The mass flow rate of helium as a function of time at which these temperature readings and pressure drop readings were taken is given in Figure 4.39. Furthermore, the inlet temperature of helium into the setup over time is presented in Figure 4.40 and the vacuum level over the course of this run is given in Table 4.6. Finally, the raw voltage readings from the pressure sensors are provided in Appendix B.

Table 4.6: Vacuum level in the cryostat over time. [23/10/2025, Cryofan: 17600 rpm]

Time [HH:MM:SS]	Vacuum level [mbar]
11:10:00	5.84 x 10^{-3}
12:10:00	2.29 x 10^{-3}
12:20:00	6.56 x 10^{-3}
13:01:00	6.43 x 10^{-3}
14:01:00	5.76 x 10^{-3}
14:20:00	4.15 x 10^{-3}
14:52:00	2.65 x 10^{-3}

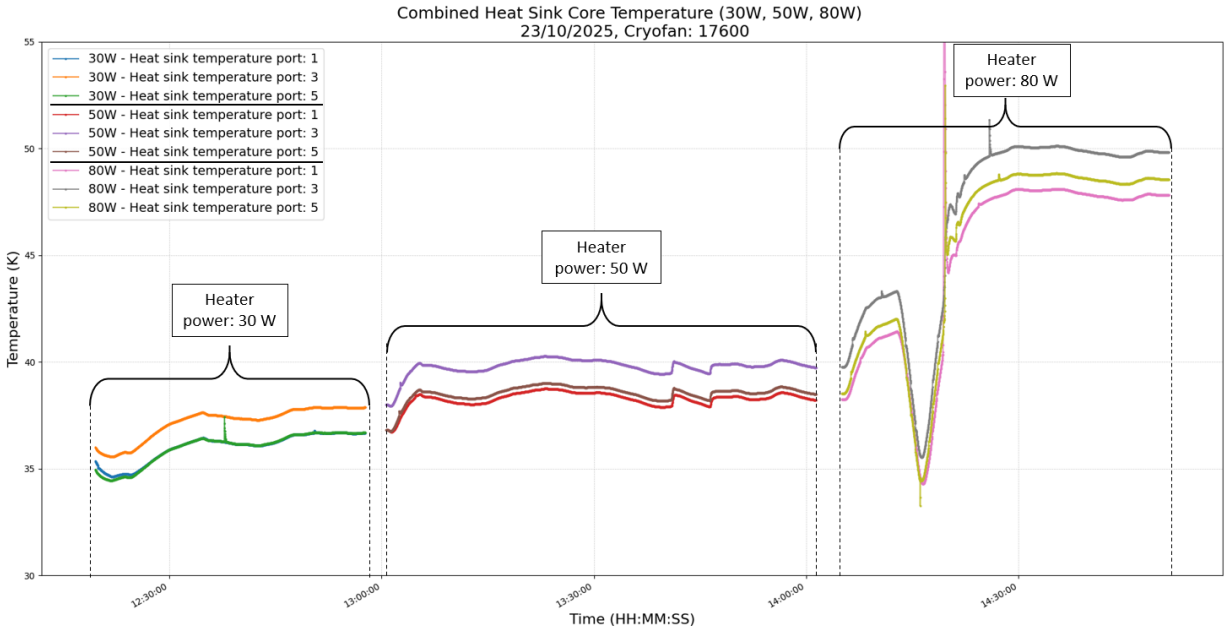


Figure 4.37: Temperature readings from temperature ports 1, 3, and 5 under different heating scenarios. [23/10/2025, Cryofan speed: 17600 rpm]

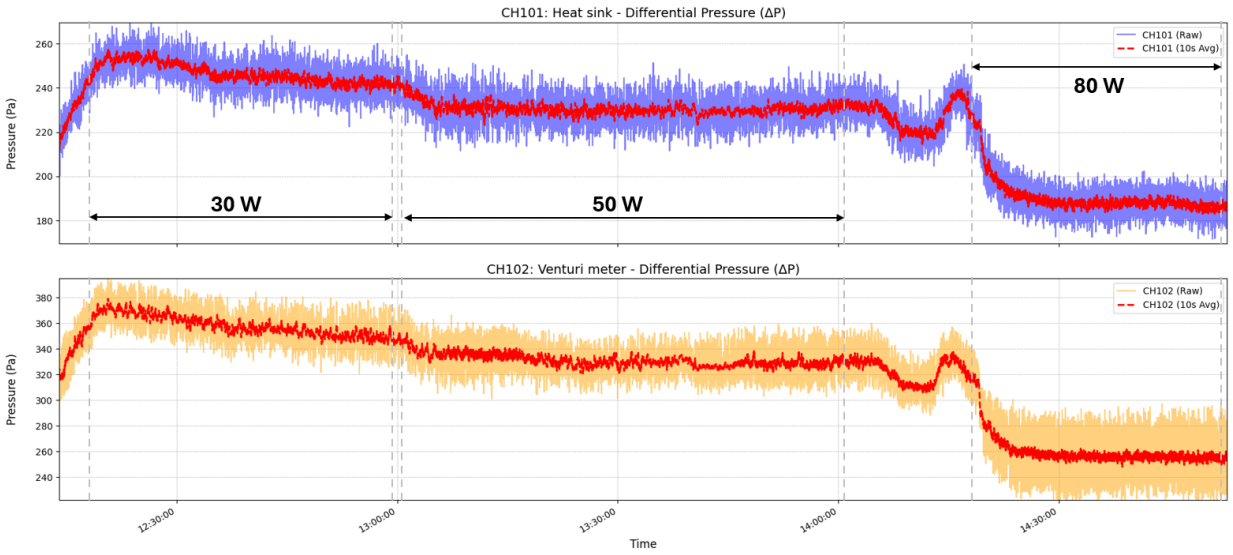


Figure 4.38: Differential pressure readings across the heat sink (CH 101) and venturi meter (CH 102) under different heating scenarios. [23/10/2025, Cryofan speed: 17600 rpm]

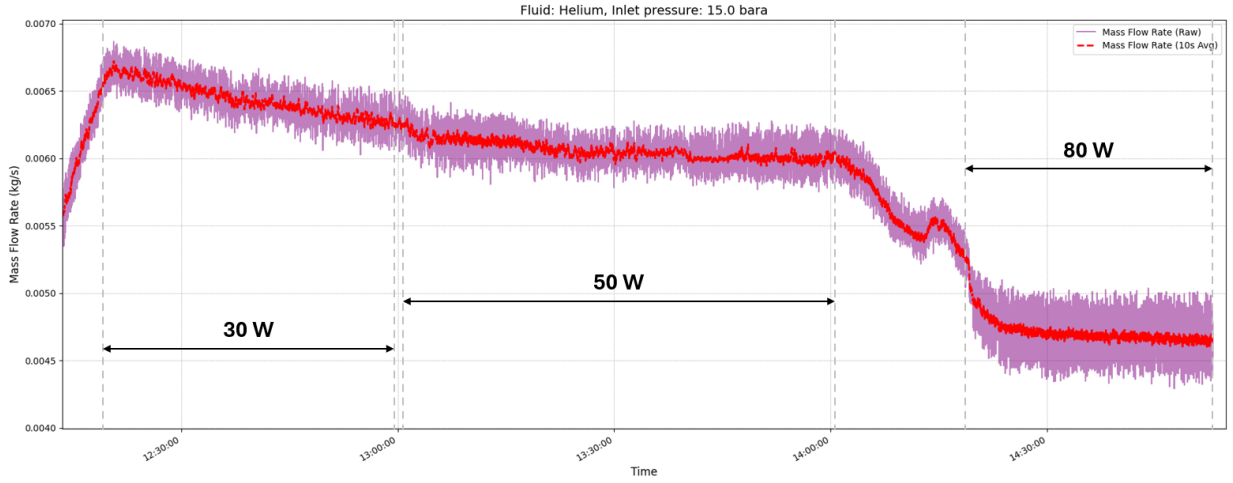


Figure 4.39: The mass flow rate of helium through the setup under different heating scenarios. [23/10/2025, Cryofan speed: 17600 rpm]

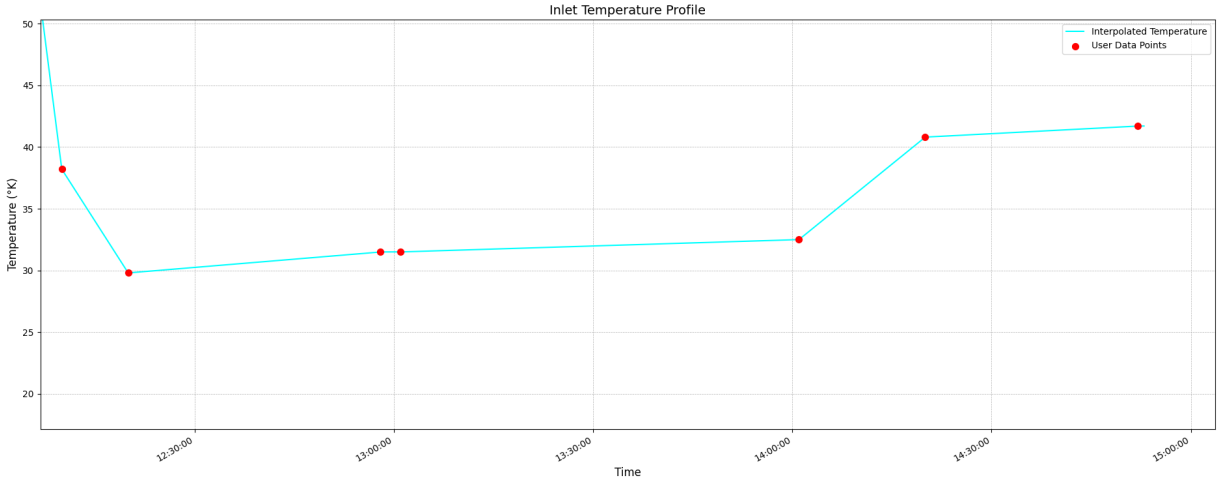


Figure 4.40: The inlet temperature of helium through the setup for different heating scenarios. [23/10/2025, Cryofan speed: 17600 rpm]

Cryofan rpm: 13100

The final run of the test campaign was done at the intermediate cryofan speed of 13100 rpm. As the previous test concluded on 80 Watts run, this test started on the same power setting and proceeded in a descending order until the end. The temperature readings for ports 1, 3, and 5 on the heat sink under different heat loads are presented in Figure 4.44. The pressure drops resulting from the heat load across the heat sink and the venturi meter are presented as a function of time in Figure 4.42. The corresponding mass flow rate of helium over time is shown in Figure 4.43. The plot of helium inlet temperature over time is also given here in Figure 4.44. The vacuum level throughout this entire test run is provided in Table 4.7. Finally, the raw voltage readings from the pressure sensors for this run are provided in Appendix B.

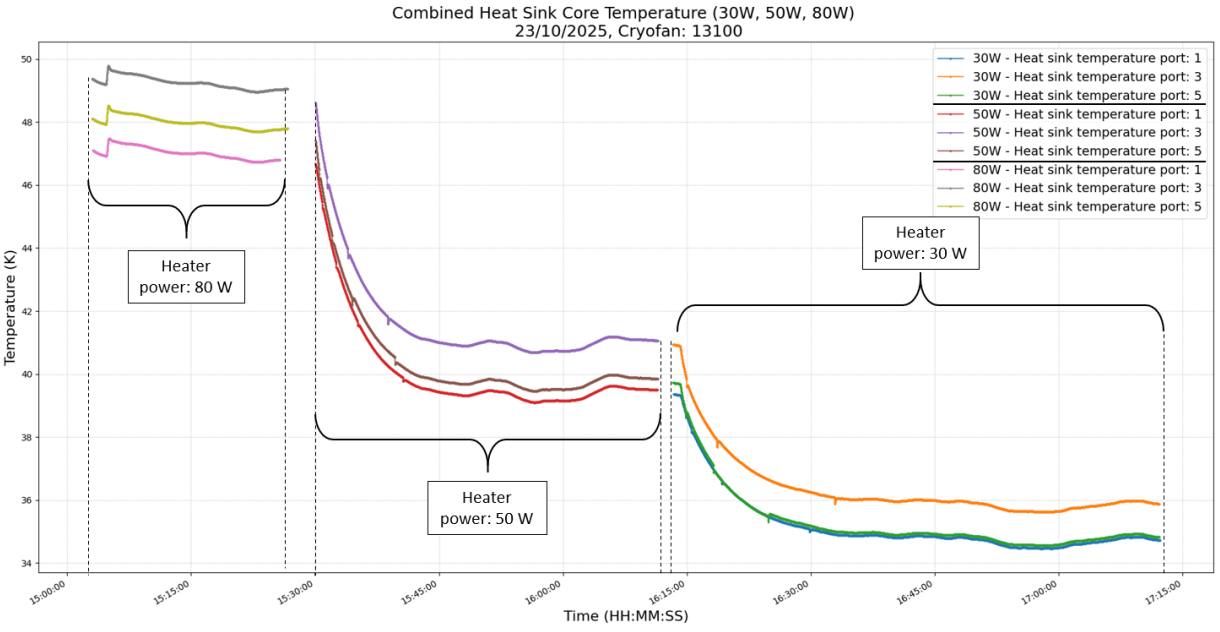


Figure 4.41: Temperature readings from temperature ports 1, 3, and 5 under different heating scenarios. [23/10/2025, Cryofan speed: 13100 rpm]

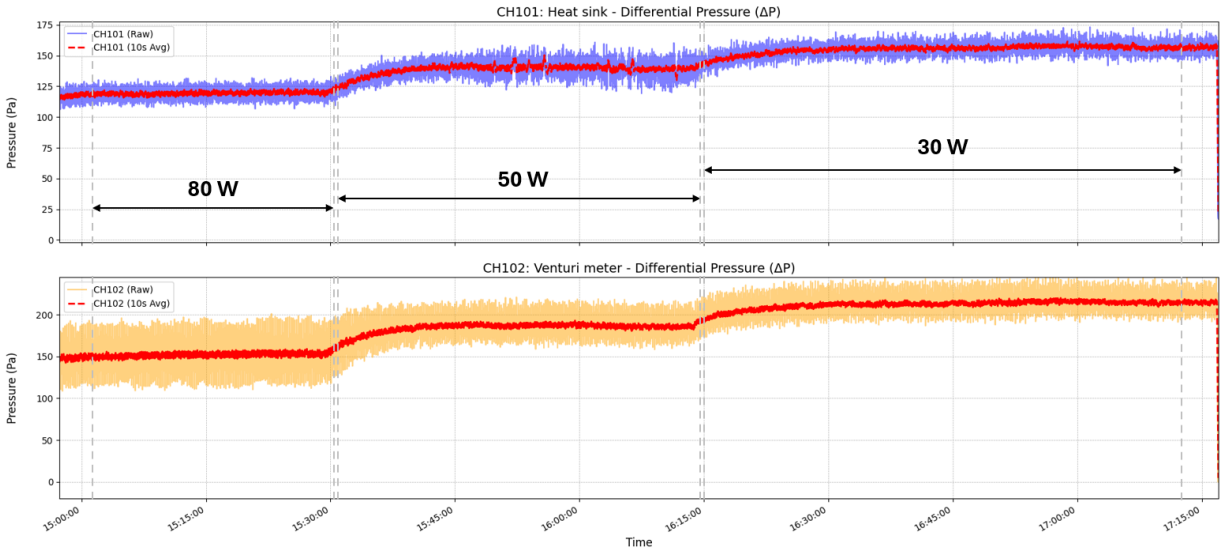


Figure 4.42: Differential pressure readings across the heat sink (CH 101) and venturi meter (CH 102) under different heating scenarios. [23/10/2025, Cryofan speed: 13100 rpm]

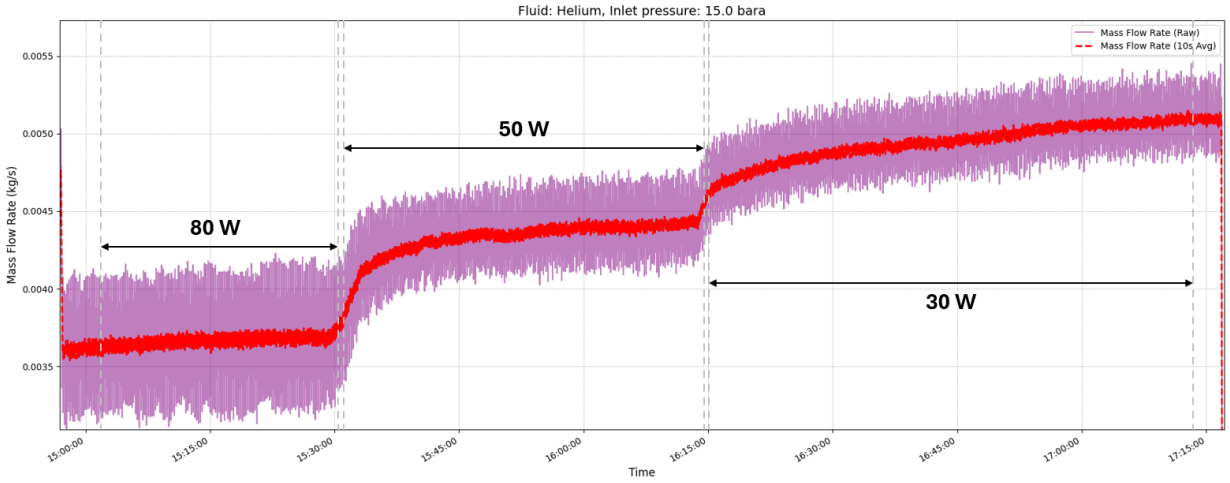


Figure 4.43: The mass flow rate of helium through the setup under different heating scenarios. [23/10/2025, Cryofan speed: 13100 rpm]

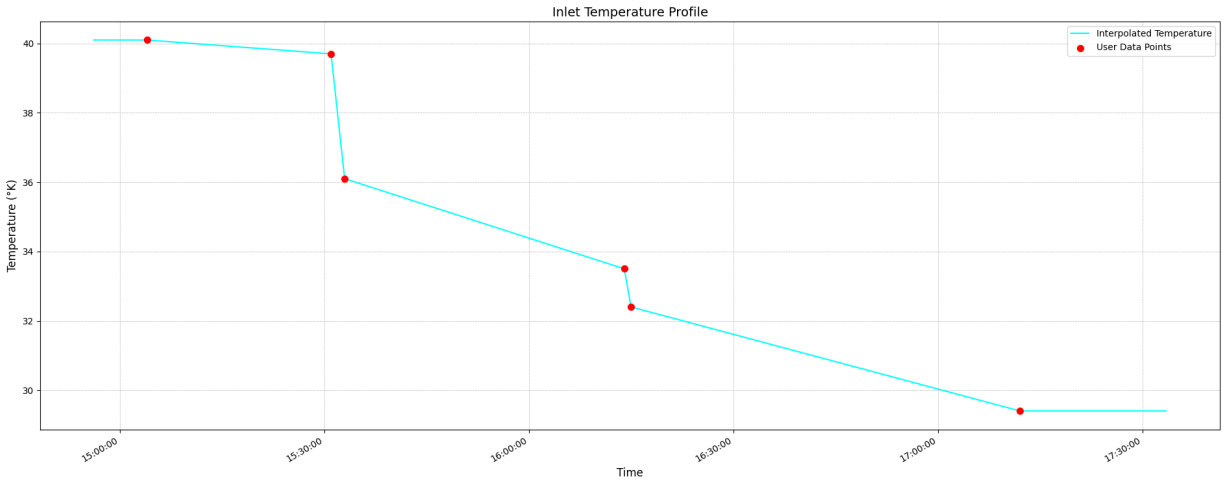


Figure 4.44: The inlet temperature of helium through the setup for different heating scenarios. [23/10/2025, Cryofan speed: 13100 rpm]

Table 4.7: Vacuum level in the cryostat over time. [23/10/2025, Cryofan: 13100 rpm]

Time [HH:MM:SS]	Vacuum level [mbar]
15:04:00	2.67 x 10^-3
15:31:00	2.72 x 10^-3
15:33:00	2.72 x 10^-3
16:14:00	3.79 x 10^-3
16:15:00	3.86 x 10^-3
17:12:00	6.38 x 10^-3

4.3.3. Experimental results: Summary

This section provides a summary of the temperature, pressure, and mass flow rate data gathered from the test campaign. The summary of all the experimental results is presented in Table 4.8.

Table 4.8: The summary of the experimental results under steady state conditions.

Date & Cryofan rpm	Heat load [W]	Inlet helium temp. [K]	Avg. Mass flow rate [g/s]	Temp. Heat sink: Port 1 [K]	Temp. Heat sink: Port 2 [K]	Temp. Heat sink: Port 3 [K]	Temp. Heat sink: Port 4 [K]	Temp. Heat sink: Port 5 [K]	Outlet helium temp. [K]	Pressure drop (Heat sink) [Pa]
22/10/2025 Cryofan speed: 8036 rpm	10	25.2	3.1	31.2	-	-	31.1	32.9	-	38
	30	30.1	2.4	35.6	-	-	36	37.6	-	26
	50	35.3	1.9	41.1	-	-	42.2	43.4	-	15
	80	44.5	1.2	49.3	-	-	51.5	52.3	-	6
23/10/2025 Cryofan speed: 13100 rpm	30	29.4	5	34.5	-	35.6	-	34.6	31.9	120
	50	33.5	4.3	39.2	-	40.7	-	39.5	37.1	137
	80	39.7	3.7	46.7	-	48.9	-	47.7	45.1	160
23/10/2025 Cryofan speed: 17600 rpm	30	31.5	6.3	36.7	-	37.8	-	36.7	33.4	242
	50	32.5	6.1	38.4	-	39.9	-	38.6	34.8	230
	80	41.7	4.7	47.6	-	49.6	-	48.4	45.9	187

4.4. Numerical vs Experimental analysis

The experimental campaign served as validation of the numerical model. A direct comparison is best made between the rerun Sim_convec_1 (with 120 μm roughness) and the experimental run at 17600 rpm (Test Day 2, 80 W), as their operating conditions were the most similar. This comparison is presented in Table 4.9. The temperature at ports 1, 3, and 5 for Sim_convec_1 (roughness: 120 μm) was obtained through the average of the blue and green curves (below and above termination) in Figure 4.29 at 16.7 mm, 50 mm, and 83.3 mm heat sink length, respectively.

Table 4.9: Comparison between Sim_convec_1 (roughness: 120 μm) and experimental results from 23/10/2025 (cryofan rpm: 17600) run.

Simulation/ Experiment	Heat load [W]	Inlet helium temp. [K]	Mass flowrate (per heat sink) [g/s]	Temp. Port 1 [K]	Temp. Port 3 [K]	Temp. Port 5 [K]	Outlet helium temp. [K]	Delta P (including expander, heat sink, reducer) [Pa]
Sim_convec_1 (roughness: 120 μm)	80	42	4.5	45.4	46.35	46.2	45.5	129
23/10/2025 17600 rpm	80	41.7	4.7	47.6	49.6	48.4	45.9	187

The thermal model's estimation is a major success of this research. The simulation predicted a temperature at port 1 of 45.4 K, at port 3 of 46.35 K, and at port 5 of 46.2 K, while the experiment measured 47.6 K, 49.6 K, and 48.4 K at ports 1, 3, and 5, respectively. Here, a maximum discrepancy of just 7% is observed. However, this result must be framed in the context of the design margin. The requirement was a peak HTS termination temperature < 50 K. The simulation (Figure 4.29) predicted a peak HTS termination temperature of 46.3 K, providing a 3.7 K safety margin. The experimental result of 49.6 K at Port 3 consumes 3.3 K (90%) of this safety margin. While the design is still successful (49.6 K < 50 K), this highlights the critical need to accurately account for the minor parasitic effects that were not modeled. This 7% error can be attributed mainly to the heat leak from conduction through gas molecules in the vacuum as shown previously. The experimental vacuum was imperfect and fluctuated over time (Table 4.5 - Table 4.7), allowing for small heat leaks via residual gas conduction and conduction through the bayonets for helium inlet and outlet, which the model (assuming perfect vacuum) did not account for.

The most significant finding of the test campaign is the pressure drop discrepancy. The simulation predicted 129 Pa for the full assembly, while the experiment measured 187 Pa. A 45% under-prediction. This indicates that the fluid-dynamic model, although thermally accurate, is fluid-dynamically incomplete and lacks key sources of pressure loss. The discrepancy is too large to be explained by a single factor. It is most likely a combination of the three factors listed below:

- Flow asymmetry: The 1/8th symmetry model used in COMSOL was a necessary simplification to reduce computation time. However, it carries the major assumption of perfectly uniform flow distribution. The actual experimental setup involves a rapid, complex expansion from a 12mm pipe into the header and vice versa at the heat sink outlet. This will indisputably create a non-uniform flow profile. Furthermore, the majority of the flow through the fins will try to accumulate towards the center to avoid the side wall of the heat sink. This accumulated flow at the center of the inlet of the heat sink is met with a solid blockage at the center of the heat sink due to the presence of the current lead, forcing the flow to squeeze between the current lead (into the heat sink) and the side wall of the heat sink. This squeeze creates a high velocity region inside the heat sink and since pressure drop is proportional to the square of velocity, this "squeeze" would cause a much higher overall ΔP than a uniform flow assumption.
- Underestimated roughness: The 120 μm was a manufacturer's estimate for the additive manufacturing (AM) process. The effective hydraulic roughness of the complex, as-built internal channels could be significantly higher as the manufacturer (PROTIQ) informed that there was no real way for the

company to measure the surface roughness inside the heat sink due to the really tight spacing. The estimate of 120 μm roughness was based purely on the experience of the engineers at PROTIQ. Since the ΔP is highly sensitive to the surface roughness, it would suggest that the surface roughness inside the heat sink is in fact greater than the estimated 120 μm.

- Setup assembly: The final factor to which the ΔP discrepancy can be attributed is the assembly of the heat sink. The model is a ”perfect” CAD geometry. The experimental setup involved silver brazing the headers to the heat sink. This process inevitably creates internal fillets or small burrs that intrude into the flow path, acting as further obstructions (minor losses) that are not present in the simulation.

A 187 Pa drop versus the 129 Pa of the heat sink assembly represents a significant parasitic loss. In the full aircraft, this ΔP must be overcome by the cryofan. This directly translates to higher cryocooler power consumption, which in turn reduces overall aircraft efficiency and adds weight. This discrepancy is not just a numerical error but a direct measurable impact on the system’s key performance metrics.

4.5. Limitations and their impact

The limitations regarding the malfunctioning temperature sensors identified in Chapter 3 became highly relevant during data analysis. The failure of 4 out of 7 Cernox temperature sensors (Figure 3.23) was the most significant experimental limitation. The COMSOL simulations show that the temperature curves have a local maximum around a heat sink length of 50 mm. This is due to the presence of the current lead (primary heat source) at that location. However, since no functioning Cernox sensor was placed in port 3 on day 1 of testing, the peak HTS termination temperature could not be compared with the simulations for the low mass flow rate scenarios. Nevertheless, interesting information can still be extracted from the low mass flow rate runs at a cryofan speed of 8036 rpm. In Sim_convec_2 the average temperature at the HTS termination peaks at a heat sink length of 50 mm, also shown in Figure A.11. This simulation is conducted at roughly half the maximum available mass flow rate for the system at 10 g/s. However, when compared with the experimental results in Figure 4.33, it can be seen that the temperature at port 5 is higher than at port 4 in all the heat load cases. This is despite the fact that port 4 is located closer to the heat source. This implies that the helium flowing through the heat sink has already heated up to or above the temperature recorded at port 5 by the time it reaches that location, as it is not able to extract the heat from the second half of the heat sink, and a higher mass flow rate is necessary to cool the heat sink.

The issue of higher temperature at port 5 versus others was resolved on the second day of testing, where the mass flow rate was increased and the temperature distribution took the expected shape of the activation energy curve for an endothermic reaction in chemistry, as seen in Figure 4.45. Another limitation in the experimental results was the time to achieve a steady state operation. In certain cases, sufficient time was not provided for the setup to achieve a true steady state condition, but close to a steady state. This was mainly due to the time constraints of the laboratory.

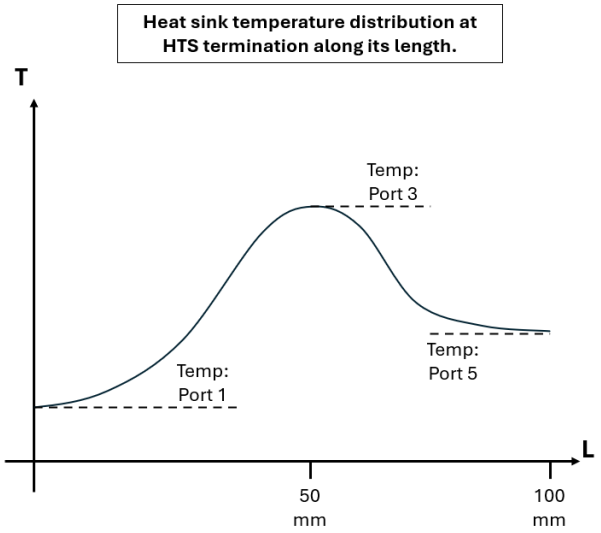


Figure 4.45: A generic representation of the temperature distribution along the length of the heat sink for test day 2.

Finally, the most significant limitation is the scope. The study successfully designed and validated the heat sink (the heat exchanger). It did not test the full termination assembly. The thermal contact resistance (TCR) at the HTS-to-heat-sink joint remains to be tested. A poor mechanical joint, contamination, or thermal-cycling-induced degradation at this interface could add to the existing ΔT, reducing the accuracy of the thermal design. Unfortunately, this could not be tested as the superconducting CORC (Conductor On Round Core) cables being manufactured by CERN and Advanced Conductor Technologies were not ready at the time of testing.

4.6. Answer to the research question

The central research question was: ***"How do the required coolant mass flow rate, system pressure drop, heat sink length, diameter and mass vary with the nominal operating current for a superconducting hydrogen electric aircraft?"*** Based on the results from the simulations and experiments, an answer can be provided pertaining to the nominal operating current, the coolant mass flow rate, the system pressure drop, and finally the heat sink geometry.

The nominal operating current is the primary driver of the heat load. However, the relationship between the current and the heat load is strongly nonlinear. An increase in current from 1.5 kA to 2 kA increased the cooling load from 67 W to over 100 W for the same L/A(as per the McFee model) and raised the peak HTS termination temperature from 46.8 K to 64.7 K (Table 4.3). This shows that any increase of the nominal current must be accompanied by an increase in coolant mass flow and a change in the L/A. This also shows that for severe overcurrent scenarios, the heat sink can not sustain the heat load that originates from the internal resistance of the current lead, highlighting the need for a fault current limiter within the circuit.

The coolant mass flow rate is the primary control variable in the setup upon which the system is highly dependent. Almost halving the mass flow rate from 18 g/s to 10 g/s raised the peak HTS termination temperature by 3.13 K to 49.93 K, bringing it to the limit of 50 K as seen in Sim_convec_1 and Sim_convec_2. ΔP across the heat sink resulting from the mass flow rate is the "cost" of cooling. It was found to be critically sensitive to two factors: surface roughness and the mass flow rate itself. The 3D-printed surface roughness was the single largest factor in the ΔP budget, responsible for increasing the pressure drop across the heat sink from 60 Pa to 90 Pa in the rerun of Sim_convec_1 with a surface roughness of 120 μm and even more during the 80 W run at 17600 rpm. As expected, the ΔP also scales non linearly with mass flow rate as seen in Table 4.8. Even though the inlet temperature is not constant for all the test cases in the experiment, a rough idea of the pressure dependence on the mass flow rate can still be obtained. This was done for the 30 W heating scenario at different mass flow rates, as the inlet fluid temperature

is the most similar in all the runs, shown by Table 4.8. Figure 4.46 shows the exponential trend for the pressure drop as a function of mass flow rate for the 30 W heating scenario, be it with just 3 data points.

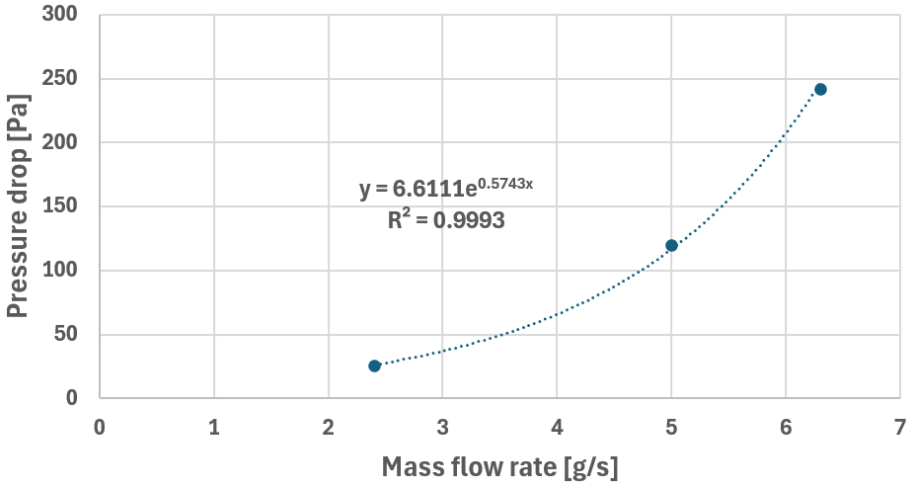


Figure 4.46: Pressure drop as a function of mass flow rate for 30 W heating scenarios.

Finally, the design evolution shown in Figure 3.14 demonstrated that the topology (concentric vs. axial fins) was far more important than simple scaling of length. The final concentric design (V5) fixed the length at 100 mm, optimizing the heat transfer pathways rather than just the dimensions. This optimized topology successfully managed the heat load, where simpler geometries of the same mass and volume failed.

5

Conclusion

This thesis successfully designed and experimentally validated a heat sink for managing the thermal load at the junction between a superconducting (SC) cable and its current lead (CL). This component is essential for enabling high-power superconducting DC links in future hydrogen-electric aircraft.

The research followed a structured methodology, beginning with 1D analytical models in Python to define the heat load (80 W per current lead) and establish a baseline axial-fin design. Next, 3D COMSOL Multiphysics simulations were used to iterate on the design. This iterative process revealed that simple axial fin geometries were insufficient, as they created areas of fluid stagnation at the CL base, the primary heat source. The key design achievement of this work was the development of a novel concentric fin topology. This design, which arranges cooling fins radially outward from the CL base, provided a shorter and more efficient thermal path to the coolant, reducing the thermal resistance between the primary heat source and the coolant. This final design met all project requirements in simulation, keeping the peak HTS termination temperature (46.35 K) below the 50 K limit and the pressure drop (90 Pa) within the 100 Pa budget.

The experimental validation was conducted on a 3D printed copper prototype in a cryogenic GHe loop at the University of Bath where the temperature readings were in line with the simulated thermal model. The experiment measured a peak termination temperature of 49.6 K under the 80 W nominal load (at 17600 rpm cryofan speed). This result successfully meets the critical < 50 K requirement, validating the concentric fin design as a viable solution. The < 7% discrepancy between the simulated (46.35 K) and measured (49.6 K) temperatures is attributed to parasitic heat leaks from an imperfect vacuum and attached bayonets, which were not modeled. A significant discrepancy was however discovered in the pressure drop across the heat sink. The experiment measured 187 Pa, a 45% increase over the 129 Pa predicted by the 120 μm surface roughness simulation. This under-prediction is attributed to real world factors not captured in the ”perfect” CAD model such as the complex, non-uniform flow in the headers, higher than estimated surface roughness ($> 120\mu m$) from the additive manufacturing process, minor losses from the silver brazing in the assembly, and finally the contraction of the heat sink upon cooling.

This work demonstrates that the concentric fin heat sink is a successful and viable component for the Cryoprop DC link. It also concludes that the fluid-dynamic cost (ΔP) of such components is critically sensitive to manufacturing tolerances and assembly methods, which must be a primary consideration in future aerospace-grade hardware development.

6

Recommendations

Although this thesis successfully validated a novel concentric fin heat sink design, confirming its thermal viability for the Cryoprop DC link, the experimental campaign also revealed critical discrepancies and unverified assumptions in the design of the heat sink. This chapter provides a brief overview of the primary recommendations for the future continuation of this research project.

Rec 1: Re-evaluation of the fluid dynamic model
The 45% discrepancy between the simulated (129 Pa) and measured (187 Pa) pressure drop is the most significant finding of this work. This discrepancy, likely caused by flow asymmetry, proves that the 1/8th symmetry model in COMSOL, while being thermally accurate, is fluid dynamically incomplete. Therefore, a full-scale 3D CFD model of the heat sink should be developed. This model must remove the symmetry assumption and include the full geometry of the heat sink, the inlet (expander) and outlet (reducer) headers. Furthermore, instead of using just the electrical, thermal, and fluid dynamic modules to simulate the performance of the heat sink, it is also important to incorporate the mechanical module which accounts for the shrinkage of the heat sink, and therefore the flow channels under cooling. This shrinkage of the flow channels is also a factor in the discrepancy between the pressure drop across the heat sink simulated in COMSOL versus the experiment.

This computationally intensive model is necessary to accurately capture the non-uniform flow distribution, recirculation zones, and the ”squeeze effect” at the CL base and due to channel shrinkage, which are the likely sources of the excess pressure drop. Accurately predicting this ΔP is essential for correctly sizing the aircraft’s cryofan, which has a direct impact on system weight and efficiency.

Rec 2: Optimization for manufacturability and assembly
During the experiment, the pressure drop was found to be critically sensitive to real-world manufacturing and assembly factors such as the surface roughness between the fins of the heat sink and the bumps/fillet caused by the silver brazing. The design must be optimized to mitigate these penalties. An investigation into post-processing techniques for the additively manufactured heat sink including methods such as chemical polishing or abrasive flow machining (AFM) is recommended to improve the surface roughness of the fins inside the heat sink.

This set of recommendations is expected to improve the model accuracy and reduce the discrepancies between the simulation results and the experimental results.

References

[1] ZeroAvia. *Advancing Hydrogen Aviation in 2025 – The 4 Pillars of Success*. Jan. 2025. URL: https://zeroavia.com/blogs/advancing-hydrogen-aviation-in-2025-the-4-pillars-of-success/ (visited on 04/14/2025).

[2] Rodger W. Dyson. *True Zero Emission Electric Aircraft Propulsion Transport Technology*. Tech. rep. E-20119P. Accessed 2025-04-14. Cleveland, OH: NASA Glenn Research Center. URL: https://ntrs.nasa.gov/api/citations/20230006407/downloads/E-20119P.pdf.

[3] Mark D. Guynn et al. *Evaluation of a Hydrogen Fuel Cell Powered Blended-Wing-Body Aircraft Concept for Reduced Noise and Emissions*. Tech. rep. NASA/TM-2004-212989. Accessed 2025-04-14. Hampton, Virginia; Cleveland, Ohio: NASA Langley Research Center; NASA Glenn Research Center, Feb. 2004. URL: https://ntrs.nasa.gov/api/citations/20040033924/downloads/20040033924.pdf.

[4] Stefan Bickert et al. *European Aviation Environmental Report 2025*. Tech. rep. Accessed April 14, 2025. European Union Aviation Safety Agency, 2025. URL: https://www.eurocontrol.int/sites/default/files/2025-01/eurocontrol-easa-eaer-2025.pdf.

[5] Andrew Scott White et al. "Trade-Space Assessment of Liquid Hydrogen Propulsion Systems for Electrified Aircraft". In: *Journal of Aircraft* (2025). Accessed: April 14, 2025, https://arc.aiaa.org/doi/10.2514/1.C037773. DOI: 10.2514/1.C037773.

[6] Patrick A. Hanlon et al. "Modeling and Trade Space for Fuel Cell Powered Aircraft". In: *AIAA Aviation / Electric Aircraft Technologies Symposium (EATS)*. Accessed: 2025-04-14. NASA Technical Reports Server (NTRS), 2024. URL: https://ntrs.nasa.gov/citations/20240008078.

[7] Mohsen Yazdani Asrami et al. "Insulation Materials and Systems for Superconducting Powertrain Devices in Future Cryo-Electrified Aircraft". In: *2022 IEEE Electrical Insulation and Dielectric Phenomena Conference (EIDP)*. Accessed: 2025-04-14. 2022, pp. 61–64. DOI: 10.1109/EIDP56337.2022.9986258. URL: https://strathprints.strath.ac.uk/80268/1/Yazdani_Asrami_etal_IEEE_EIM_2022_Insulation_materials_and_systems_for_superconducting_powertrain_devices.pdf.

[8] Jean Lévêque et al. "Sizing Superconducting motor associated with cryogenics power electronics for disruptive hybrid electric propulsion aircrafts". In: *9th European Conference for Aeronautics and Space Sciences (EUCASS)*. Accessed: 2025-04-14. 2022. DOI: 10.13009/EUCASS2022-7234. URL: https://www.imothep-project.eu/files/EUCASS2022-7234.pdf.

[9] Mark A. Lewis. *Cryocoolers for aircraft superconducting generators and motors*. Tech. rep. Accessed: 2025-04-14. Thermodynamics Research Center, National Institute of Standards and Technology, 2012. URL: https://trc.nist.gov/cryogenics/Papers/Review/2012-Cryocoolers_for_Aircraft_Superconducting_Generators_and_Motors.pdf.

[10] E. Nilsson et al. "Performance of the 500 kW Superconducting DC and AC Links of the ASCEND Demonstrator at Airbus". In: *IEEE Transactions on Applied Superconductivity* 33.8 (2023). Accessed: 2025-04-16, pp. 1–6. DOI: 10.1109/TASC.2023.3346357. URL: https://www.advancedconductor.com/wp-content/uploads/2024/01/Nilsson-E.-IEEE-Trans.-Appl.-Supercond.-33-5401006-2023.pdf.

[11] J. W. Hsu et al. *Current Lead Optimization for Cryogenic Operation at Intermediate Temperatures*. Tech. rep. PSFC/JA-08-9. Accessed: 2025-04-16. Plasma Science and Fusion Center, Massachusetts Institute of Technology, 2008. URL: https://library.psfc.mit.edu/catalog/reports/2000/08ja/08ja009/08ja009_full.pdf.

[12] R. Wesche et al. *Conceptual Design of a 20-kA Current Lead Using Forced-Flow Cooling and Ag-Alloy-Sheathed Bi-2223 High-Temperature Superconductors*. Tech. rep. Accessed: 2025-04-16. Karlsruhe Institute of Technology (KIT), 2011. URL: https://www.osti.gov/servlets/purl/10194806.

[13] A. Zappatore et al. “Quench experiments on sub-size HTS Cable-In-Conduit Conductors for fusion applications: Data analysis and model validation”. In: *Cryogenics* 132 (2023), p. 103695. DOI: https://doi.org/10.1016/j.cryogenics.2023.103695. URL: https://www.sciencedirect.com/science/article/pii/S0011227523000693.

[14] Majid Fard et al. “Aircraft Distributed Electric Propulsion Technologies - A Review”. In: *IEEE Transactions on Transportation Electrification* 8 (Dec. 2022), pp. 1–1. DOI: 10.1109/TTE.2022.3197332.

[15] Maria Coutinho et al. “A Study on Thermal Management Systems for Hybrid–Electric Aircraft”. In: *Aerospace* 10.9 (2023). URL: https://www.mdpi.com/2226-4310/10/9/745.

[16] Francesco Sciatti et al. “An Innovative Cryogenic Heat Exchanger Design for Sustainable Aviation”. In: *Energies* 18.5 (2025). DOI: 10.3390/en18051261. URL: https://www.mdpi.com/1996-1073/18/5/1261.

[17] Moritz Kuhn. *Density of hydrogen under different pressures and temperatures*. https://www.ilkdresden.de/fileadmin/_processed_/2/f/csm_Storage_Density_of_Hydrogen_b71cedc151.jpg. Image courtesy of ILK Dresden, licensed under CC BY-SA 4.0. 2012.

[18] J.E. Hirsch. “Explanation of the Meissner effect and prediction of a spin Meissner effect in low and high Tc superconductors”. In: *Physica C: Superconductivity and its Applications* 470 (2010). Proceedings of the 9th International Conference on Materials and Mechanisms of Superconductivity, S955–S956. DOI: https://doi.org/10.1016/j.physc.2009.10.066. URL: https://www.sciencedirect.com/science/article/pii/S0921453409006248.

[19] Jan Hoffmann et al. “Design Study for a Superconducting High-Power Fan Drive for a Long-Range Aircraft”. In: *Energies* 17.22 (2024). DOI: 10.3390/en17225652. URL: https://www.mdpi.com/1996-1073/17/22/5652.

[20] Lamarcus Hampton et al. *Compact Superconducting Power Systems for Airborne Applications*. Published by Mobility Engineering Technology. Originally published in The WSTIAC Quarterly, v9 n1, pp. 75-77, 2009. 2011. URL: https://www.mobilityengineeringtech.com/component/content/article/9098-afrl-0188 (visited on 04/14/2025).

[21] European Commission, Joint Research Centre. *Superconductors*. Tech. rep. JRC Technical Report. European Commission, Joint Research Centre, July 2018. URL: https://ec.europa.eu/futurium/en/system/files/ged/superconductors.pdf (visited on 04/14/2025).

[22] Muhammad Junaid et al. “Simulation Analysis of DC Fault Interruption Characteristics of Superconducting Electric Aircraft Propulsion”. In: *CSEE Journal of Power and Energy Systems* 99 (Nov. 2023), pp. 1–9. DOI: 10.17775/CSEEJPES.2023.03650.

[23] Mohammad Yazdani Asrami et al. “Selecting a Cryogenic Cooling System for Superconducting Machines: General Considerations for Electric Machine Designers and Engineers”. In: *International Journal of Refrigeration* 140 (May 2022). DOI: 10.1016/j.ijrefrig.2022.05.003.

[24] Runar Mellerud et al. “Design of a Power-Dense Aviation Motor with a Low-Loss Superconducting Slotted Armature”. In: *TechRxiv* (July 2023). Preprint. DOI: 10.36227/techrxiv.23612001.v1.

[25] Sastry Pamidi et al. “High-temperature superconducting (HTS) power cables cooled by helium gas”. In: *Superconductors in the Power Grid: Materials and Applications* (Dec. 2015), pp. 225–260. DOI: 10.1016/B978-1-78242-029-3.00007-8.

[26] JDA Solutions. *EASA and FAA Show 1st Iteration Plans on H2 Certification*. Nov. 2023. URL: https://jdasolutions.aero/blog/easa-and-faa-show-1st-iteration-plans-on-h2-certification/ (visited on 04/18/2025).

[27] Takato Masuda et al. “High-temperature Superconducting Cable Technology and Development Trends”. In: *SEI Technical Review* 59 (Jan. 2005), pp. 8–13.

[28] Gi-Dong Nam et al. “Design and Analysis of Cryogenic Cooling System for Electric Propulsion System Using Liquid Hydrogen”. In: *Energies* 16.1 (2023). DOI: 10.3390/en16010527. URL: https://www.mdpi.com/1996-1073/16/1/527.

[29] Christian Barth et al. “Temperature- and field-dependent characterization of a conductor on round core cable”. In: *Superconductor Science and Technology* 28 (Apr. 2015). DOI: 10.1088/0953-2048/28/6/065007.

[30] Tim Mulder et al. “Optimized and practical electrical joints for CORC type HTS cables”. In: *IOP Conference Series: Materials Science and Engineering* 102 (Dec. 2015), p. 012026. DOI: 10.1088/1757-899X/102/1/012026.

[31] Airbus. *Cryogenics and superconductivity for aircraft, explained*. Accessed: 2025-05-13. Mar. 2021. URL: https://www.airbus.com/en/newsroom/stories/2021-03-cryogenics-and-superconductivity-for-aircraft-explained.

[32] A. Ballarino. *Current Leads, Links and Buses*. Tech. rep. Accessed: 2025-04-14. CERN, 2021. URL: https://uspas.fnal.gov/materials/21onlineSBU/Background/Further%20reading%20-%20Current%20leads.pdf.

[33] Thanh-Dung Le et al. “Thermal Design of a Cryogenics Cooling System for a 10 MW-Class High-Temperature Superconducting Rotating Machine”. In: *IEEE Transactions on Applied Superconductivity* 25 (June 2015), pp. 1–1. DOI: 10.1109/TASC.2015.2395721.

[34] Sergio Calatroni. *Materials & Properties: Thermal & Electrical Characteristics*. https://doi.org/10.48550/arXiv.2006.02842. Preprint, CERN Accelerator School lecture. 2020. arXiv: 2006.02842 [physics.acc-ph].

[35] Eckels Engineering Inc. *CryoComp: Rapid Cryogenic Design and Thermal Analysis Software with Materials Properties Database*. Accessed: 2025-04-22. 2012. URL: http://www.eckelsengineering.com/.

[36] Riaz Muhammad et al. “Fully developed Darcy-Forchheimer mixed convective flow over a curved surface with activation energy and entropy generation”. In: *Computer Methods and Programs in Biomedicine* 188 (2020), p. 105298. DOI: https://doi.org/10.1016/j.cmpb.2019.105298. URL: https://www.sciencedirect.com/science/article/pii/S0169260719316177.

[37] Jr. Ronald G. Ross. “Achieving Cryogenic Temperatures”. In: *Low Temperature Materials and Mechanisms*. Ed. by Yoseph Bar-Cohen. Accessed: 2025-05-15. Boca Raton, FL: CRC Press, 2016. Chap. 6, pp. 109–181.

[38] Randall F. Barron. *Cryogenic Heat Transfer*. Boca Raton, FL: CRC Press, 2016.

[39] Majid Asli et al. “Thermal management challenges in hybrid-electric propulsion aircraft”. In: *Progress in Aerospace Sciences* 144 (Jan. 2024), p. 100967. DOI: 10.1016/j.paerosci.2023.100967.

[40] Peter Bradley et al. *Properties of Selected Materials at Cryogenic Temperatures*. Accessed: 2025-05-19. Boca Raton, FL: CRC Press, 2013. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=913059.

[41] JPE. *Cryogenic (4K…293K) Material Properties*. https://www.jpe-innovations.com/downloads/Cryo-material-properties.pdf. Accessed: 2025-05-19. 2023.

[42] Leslie Bromberg et al. “Efficient and Lightweight Current leads”. In: 1573 (Dec. 2013). DOI: 10.1063/1.4860819.

[43] A. Ballarino. *Conduction-Cooled 60 A Resistive Current Leads for LHC Dipole Correctors*. Tech. rep. LHC Project Report 691. Accessed: 2025-05-19. Geneva, Switzerland: CERN, 2004. URL: https://at-mel-cf.web.cern.ch/resources/60A_leads.pdf.

[44] HTS-110 Ltd. *CryoSaver™ Current Leads*. https://www.hts-110.com/product/cryosaver-current-leads/. Accessed: 2025-05-19. 2021.

[45] Louis J. Salerno et al. *Thermal Contact Conductance*. NASA Technical Memorandum NASA-TM-110429. Accessed: 2025-05-19. Moffett Field, CA: NASA Ames Research Center, 1997. URL: https://ntrs.nasa.gov/api/citations/19970026086/downloads/19970026086.pdf.

[46] A Dillon et al. “Thermal interface material characterization for cryogenic electronic packaging solutions”. In: *IOP Conference Series: Materials Science and Engineering* 278 (Dec. 2017), p. 012054. DOI: 10.1088/1757-899X/278/1/012054.

[47] R.C. Dhuley. “Pressed copper and gold-plated copper contacts at low temperatures – A review of thermal contact resistance”. In: *Cryogenics* 101 (2019), pp. 111–124. DOI: https://doi.org/10.1016/j.cryogenics.2019.06.008. URL: https://www.sciencedirect.com/science/article/pii/S0011227519300943.

[48] H. Fujishiro et al. “Thermal contact resistance between high-Tc superconductor and copper”. In: *Physica C Superconductivity* 357 (Sept. 2001), pp. 785–788. DOI: 10.1016/S0921-4534(01)00364-1.

[49] W. Stautner et al. “Use of Galinstan as a Contact Agent for Additively Manufactured Components in Cryogenic Engineering”. In: *Proceedings of the 27th International Cryogenic Engineering Conference and International Cryogenic Materials Conference (ICEC27-ICMC2018)*. Accessed: 2025-05-19. CERN. Oxford, UK, 2019. URL: https://indico.cern.ch/event/760666/contributions/3391031/attachments/1887598/3112118/Demountable_thermal_interface_material_for_cryogenic_applications_V4.pdf.

[50] Handong Gui et al. “Review of Power Electronics Components at Cryogenic Temperatures”. In: *IEEE Transactions on Power Electronics* 35.5 (2020). Accessed: 2025-05-19, pp. 5144–5156. DOI: 10.1109/TPEL.2019.2944781. URL: https://doi.org/10.1109/TPEL.2019.2944781.

[51] Ho-Myung Chang et al. “Optimization of conduction-cooled current leads with unsteady operating current”. In: *Cryogenics* 49.5 (2009), pp. 210–216. DOI: https://doi.org/10.1016/j.cryogenics.2009.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0011227509000265.

[52] Jack Ekin. *Experimental Techniques for Low-Temperature Measurements: Cryostat Design, Material Properties and Superconductor Critical-Current Testing*. Oxford University Press, Oct. 2006. DOI: 10.1093/acprof:oso/9780198570547.001.0001. URL: https://doi.org/10.1093/acprof:oso/9780198570547.001.0001.

[53] R. Heller et al. “Status of the High Temperature Superconductor Current Lead Development at the Research Centre Karlsruhe”. In: *IEEE/CSC & ESAS European Superconductivity News Forum* 3 (Jan. 2008). Reference No. ST12, Category 6, pp. 1–7. URL: https://www.itek.fzk.de/.

[54] U.S. Navy SBIR Program. *(SBIR) Navy - Advanced Compact Shipboard High Temperature Superconducting (HTS) Cable Terminations*. https://www.navysbir.com/n20_1/N201-035.htm. Topic N201-035. Accessed: April 14, 2025. 2020.

[55] Airbus. *ZEROe: our hydrogen-powered aircraft*. https://www.airbus.com/en/innovation/energy-transition/hydrogen/zeroe-our-hydrogen-powered-aircraft. Accessed: 2025-10-28. 2025.

[56] Gabriel Hajiri et al. “Design and modelling tools for DC HTS cables for the future railway network in France”. In: *Superconductor Science and Technology* 35.2 (2022), p. 024003. DOI: 10.1088/1361-6668/ac43c7.

[57] Quynh Nguyen et al. “Tesla’s fluidic diode and the electronic-hydraulic analogy”. In: *American Journal of Physics* 89 (Apr. 2021), pp. 393–402. DOI: 10.1119/10.0003395.

[58] Ian H. Bell et al. “Pure and Pseudo-pure Fluid Thermophysical Property Evaluation and the Open-Source Thermophysical Property Library CoolProp”. In: *Industrial & Engineering Chemistry Research* 53.6 (2014), pp. 2498–2508. DOI: 10.1021/ie4033999. URL: https://doi.org/10.1021/ie4033999.

[59] Glen W. Howell et al. *Aerospace Fluid Component Designers' Handbook. Volume I, Revision D*. Technical Report AD0874542. TRW Systems Group, Redondo Beach, CA, 1970. URL: https://apps.dtic.mil/sti/pdfs/AD0874542.pdf.

[60] Luiz Eduardo Muzzo et al. "Uncertainty of pipe flow friction factor equations". In: *Mechanics Research Communications* 116 (2021), p. 103764. DOI: https://doi.org/10.1016/j.mechrescom.2021.103764. URL: https://www.sciencedirect.com/science/article/pii/S0093641321001026.

[61] E. R. G. Eckert et al. "Heat-Transfer Studies on Some Stable Organic Fluids in a Review of Current Literature". In: *International Journal of Heat and Mass Transfer* 5.4 (1962), pp. 361–378. DOI: 10.1007/BF02559682.

[62] *Finite Element Mesh Refinement Definition and Techniques*. https://www.comsol.com/multiphysics/mesh-refinement. Accessed: September 22, 2025. COMSOL.

[63] *Choosing a Meshing Sequence Type*. https://doc.comsol.com/5.5/doc/com.comsol.help.comsol/comsol_ref_mesh.15.17.html. Accessed: September 22, 2025. COMSOL.

[64] *Meshing*. https://doc.comsol.com/5.5/doc/com.comsol.help.cfd/cfd_ug_quickstart.05.07.html. Accessed: September 22, 2025. COMSOL.

[65] *Efficiently Mesh Your Model Geometry with Meshing Sequences | COMSOL Blog*. https://www.comsol.com/blogs/efficiently-mesh-your-model-geometry-with-meshing-sequences. Accessed: September 22, 2025. COMSOL.

[66] *Best Practices for Meshing Domains with Different Size Settings | COMSOL Blog*. https://www.comsol.com/blogs/best-practices-for-meshing-domains-with-different-size-settings. Accessed: September 22, 2025. COMSOL.

[67] *Meshing Your Geometry: When to Use the Various Element Types ...* https://www.comsol.com/blogs/meshing-your-geometry-various-element-types. Accessed: September 22, 2025. COMSOL.

[68] Srinivas C. Tadepalli et al. "Comparison of hexahedral and tetrahedral elements in finite element analysis of the foot and footwear". In: *Journal of Biomechanics* 44.12 (2011), pp. 2337–2343. DOI: 10.1016/j.jbiomech.2011.05.006.

[69] S. E. Benzley et al. "A Comparison of All Hexagonal and All Tetrahedral Finite Element Meshes for Elastic and Elasto-plastic Analysis". In: *Proceedings of the 4th International Meshing Roundtable*. 1995, pp. 179–191. DOI: 10.1002/nme.1620381604.

[70] *Hexahedral versus Tetrahedral Finite Elements Comparisons*. https://www.valuedes.co.uk/blog/tet-vs-hex-handbags-at-dawn. Accessed: September 22, 2025. Value Design Ltd.

[71] *Fundamentals of Swept Meshing*. https://www.comsol.com/support/learning-center/article/fundamentals-of-swept-meshing-51861/152. Accessed: September 22, 2025. COMSOL.

[72] *Mesh Element Quality and Size*. https://doc.comsol.com/5.5/doc/com.comsol.help.comsol/comsol_ref_mesh.15.18.html. Accessed: September 22, 2025. COMSOL.

[73] Hanna Gothäll. *How to Inspect Your Mesh in* COMSOL Multiphysics®. Blog post, *COMSOL Blog*. Accessed: 2025-10-29. July 2022. URL: https://www.comsol.com/blogs/how-to-inspect-your-mesh-in-comsol-multiphysics.

[74] *Joule Heating Simulations Tutorial | COMSOL Blog*. https://www.comsol.com/blogs/joule-heating-simulations-tutorial. Accessed: September 23, 2025. COMSOL.

[75] COMSOL AB. *"Single-Phase Flow Interfaces" in COMSOL® Multiphysics® 6.1 Documentation*. https://doc.comsol.com/6.1/doc/com.comsol.help.cfd/cfd_ug_fluidflow_single.06.088.html. Accessed: 2025-10-29. 2025.

[76] *The Heat Transfer in Solids and Fluids Interface*. https://doc.comsol.com/5.5/doc/com.comsol.help.heat/heat_ug_interfaces.08.20.html. Accessed: September 22, 2025. COMSOL.

[77] *Joule Heating in a Fuse on a Circuit Board, Chapter 1*. https://www.comsol.com/video/joule-heating-fuse-circuit-board-chapter-1. Accessed: September 22, 2025. COMSOL.

[78] *The NIST REFPROP Database for Highly Accurate Properties of Industrially Important Fluids*. https://pmc.ncbi.nlm.nih.gov/articles/PMC9619405/. Accessed: September 22, 2025. National Institute of Standards and Technology (NIST).

[79] *Helium — CoolProp 7.1.0 documentation*. https://coolprop.org/fluid_properties/fluids/Helium.html. Accessed: September 22, 2025. CoolProp.

[80] Jack W. Ekin. *Experimental Techniques for Low-Temperature Measurements: Cryostat Design, Material Properties, and Superconductor Critical-Current Testing*. Oxford, UK: Oxford University Press, 2006.

[81] *Properties of Copper and Copper Alloys at Cryogenic Temperatures*. https://ncsx.pppl.gov/NCSX_Engineering/Materials/CopperProperties/PB92172766.pdf. Accessed: September 22, 2025. Princeton Plasma Physics Laboratory.

[82] N. J. Simon et al. *Properties of copper and copper alloys at cryogenic temperatures*. Tech. rep. NIST MONO 177. National Institute of Standards and Technology, 1992.

[83] K. G. Ramanathan et al. “Total hemispherical emissivities of copper, aluminum, and silver”. In: *Applied Optics* 16.11 (1977), pp. 2810–2817. DOI: 10.1364/AO.16.002810.

[84] R. C. Duckworth et al. “Measurement of the emissivity of clean and contaminated silver plated copper surfaces at cryogenic temperatures”. In: *Advances in Cryogenic Engineering: Transactions of the International Cryogenic Materials Conference, ICMC*. Vol. 824. AIP Conference Proceedings. 2006, pp. 61–68. DOI: 10.1063/1.2192334.

[85] J. P. Holman. *Heat Transfer*. 10th. New York, NY: McGraw-Hill, 2010.

[86] *The Low Reynolds Number k-ε Turbulence Model*. https://doc.comsol.com/5.5/doc/com.sol.help.cfd/cfd_ug_fluidflow_single.06.092.html. Accessed: October 8, 2025. COMSOL.

[87] Stirling Cryogenics. *Cryogenic Fans*. Online. Accessed: November 17, 2025. 2025. URL: https://stirlingcryogenics.com/products/cryogenic-fans/.

[88] Lakeshore Cryotronics. *“PackagesCX – Sensor Packaging Options”*. https://www.lakeshore.com/packaging/packagescx. Accessed: 2025-10-29.

[89] KELLER Druckmesstechnik AG. *PD-23X – High-Precision Differential Pressure Transmitters*. https://keller-pressure.com/en/company/news/keller-pd-23x-high-precision-differential-pressure-transmitters. Accessed: 2025-10-29. 2025.

[90] KELLER Druckmesstechnik AG. *Series 21Y – Standard Pressure Transmitters*. https://keller-pressure.com/en/products/pressure-transmitters/standard-pressure-transmitters/series-21y. Accessed: 2025-10-29. 2025.

[91] International Organization for Standardization. *Vacuum technology — Vocabulary — Part 1: General terms*. Standard. Geneva, Switzerland, 2019. URL: https://www.iso.org/standard/71267.html.

[92] M. Chorowski et al. “Optimisation of Multilayer Insulation – An Engineering Approach”. In: *LHC Project Report* 464 (2001), pp. 1–8. URL: https://cds.cern.ch/record/486721.

[93] B. Baudouy. “Heat Transfer and Cooling Techniques at Low Temperature”. In: *CERN Yellow Report CERN-2014-005* (2014). Contribution to the CAS–CERN Accelerator School: Superconductivity for Accelerators, Erice, Italy, 24 April – 4 May 2013, pp. 329–352. DOI: 10.5170/CERN-2014-005.329. arXiv: 1501.07153.

[94] S. Van Sciver et al. “He ii flowmetering”. In: *Cryogenics* 31.2 (1991), pp. 75–86.

Appendix A

This appendix contains the results of all the COMSOL simulations shown in Table 4.2 & Table 4.3 with the exception of $Sim\_T\_fix\_1$ and Sim_convec_1.

A.1. Fixed-Temperature Boundary Condition Simulations

This section contains the results of simulations with a fixed temperature boundary condition on the busbar.

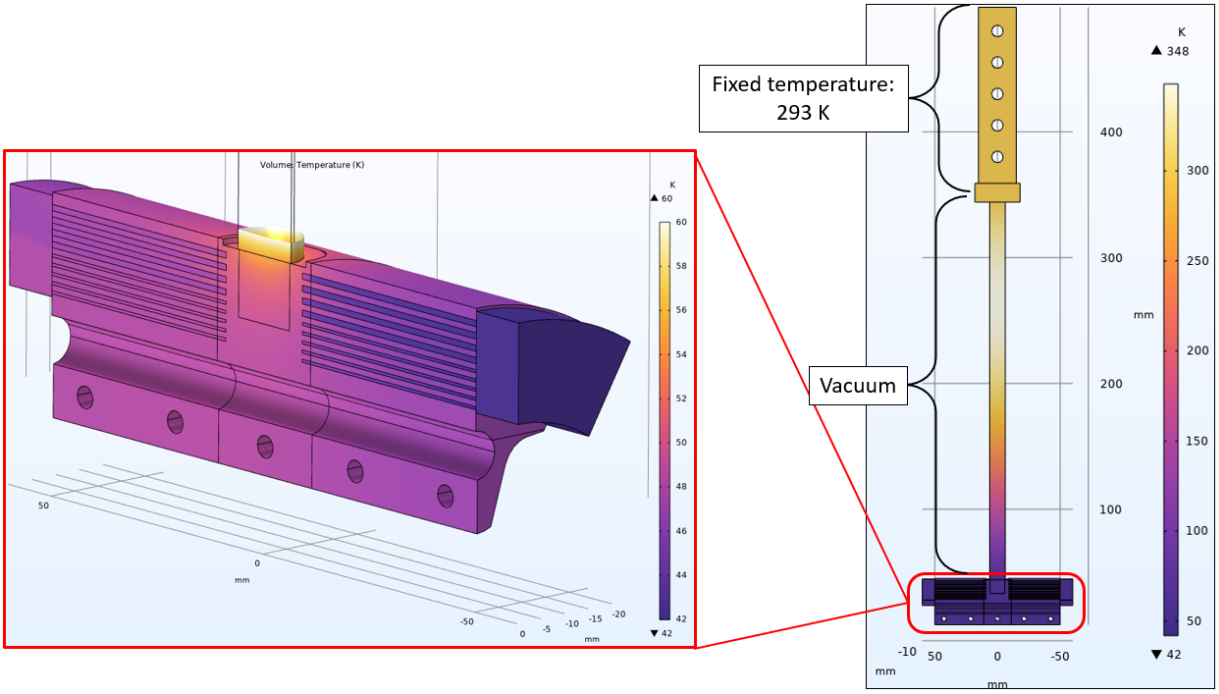


Figure A.1: Temperature distribution for $Sim\_T\_fix\_2$ simulation.

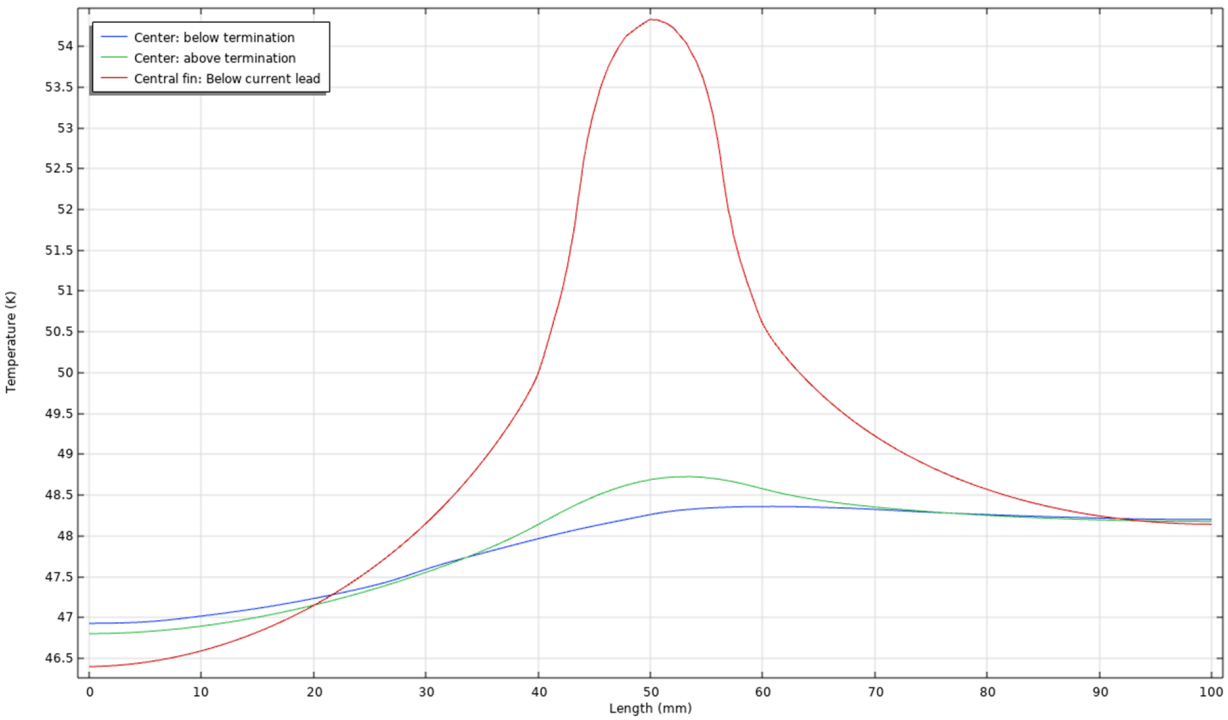


Figure A.2: Temperature profiles along the length of the concentric fin heat sink for $Sim\_T\_fix\_2$ simulation.

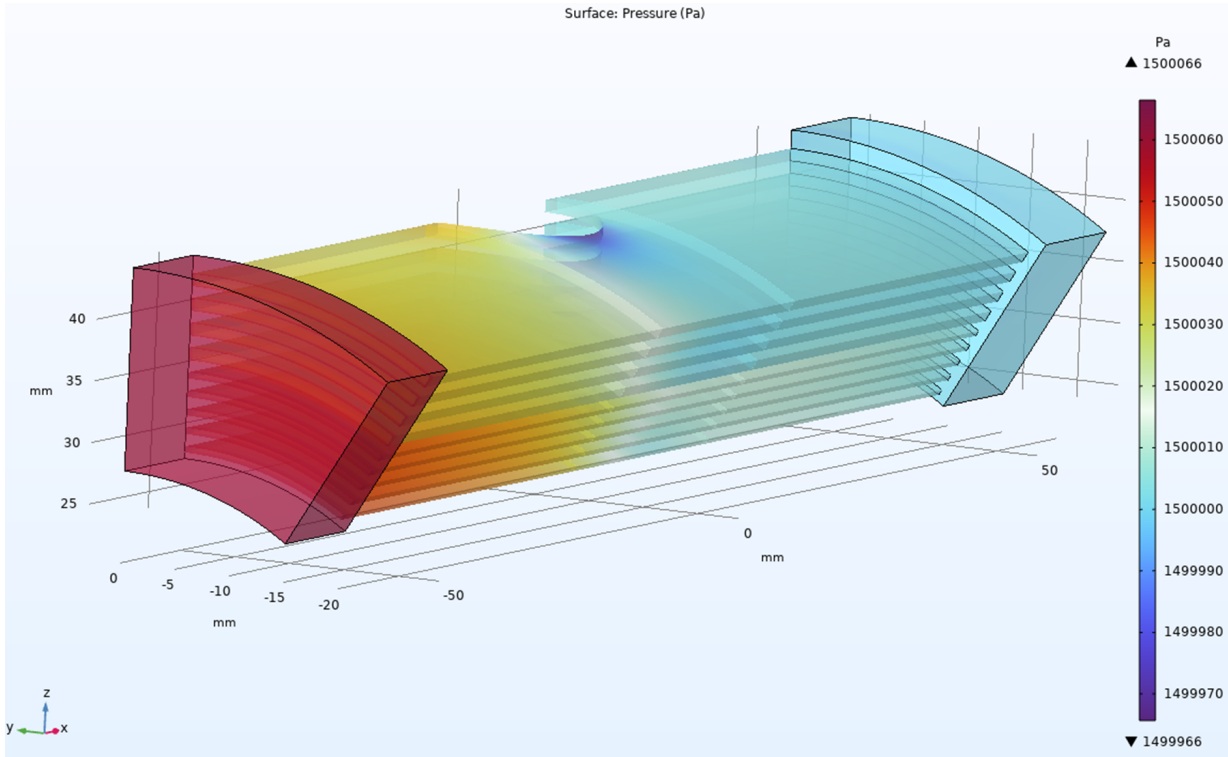


Figure A.3: The pressure field across the heat sink for $Sim\_T\_fix\_2$ simulation.

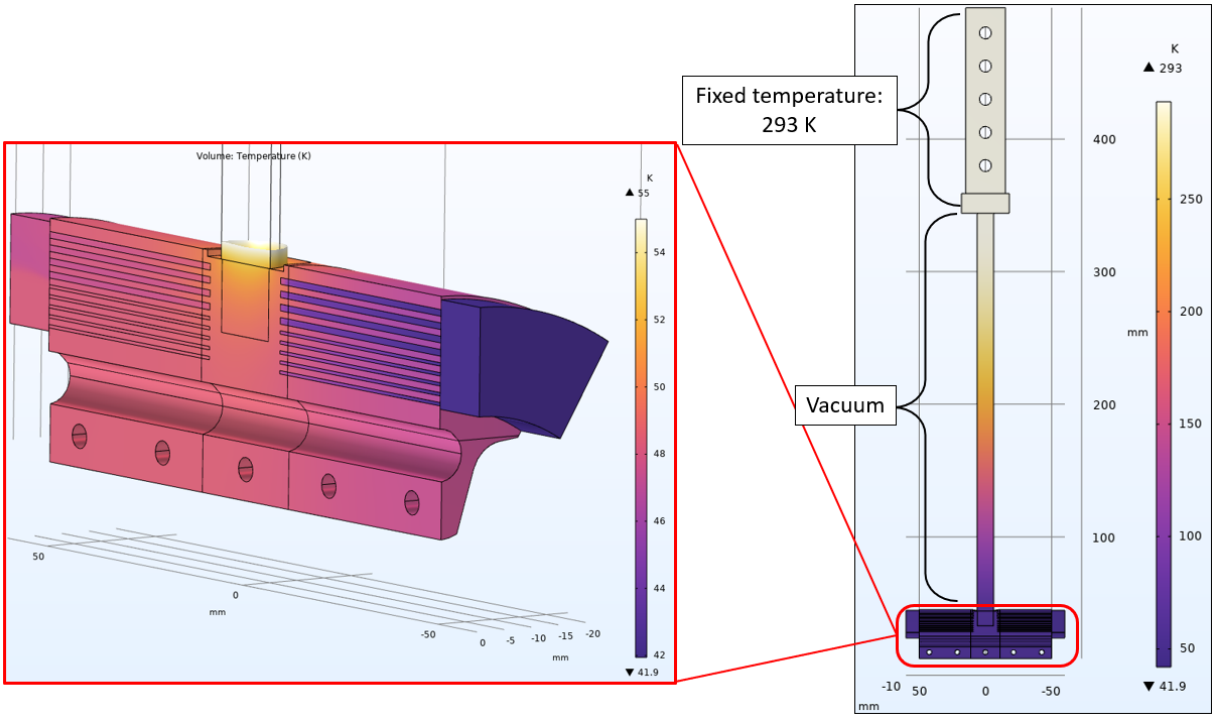


Figure A.4: Temperature distribution for $Sim\_T\_fix\_3$ simulation.

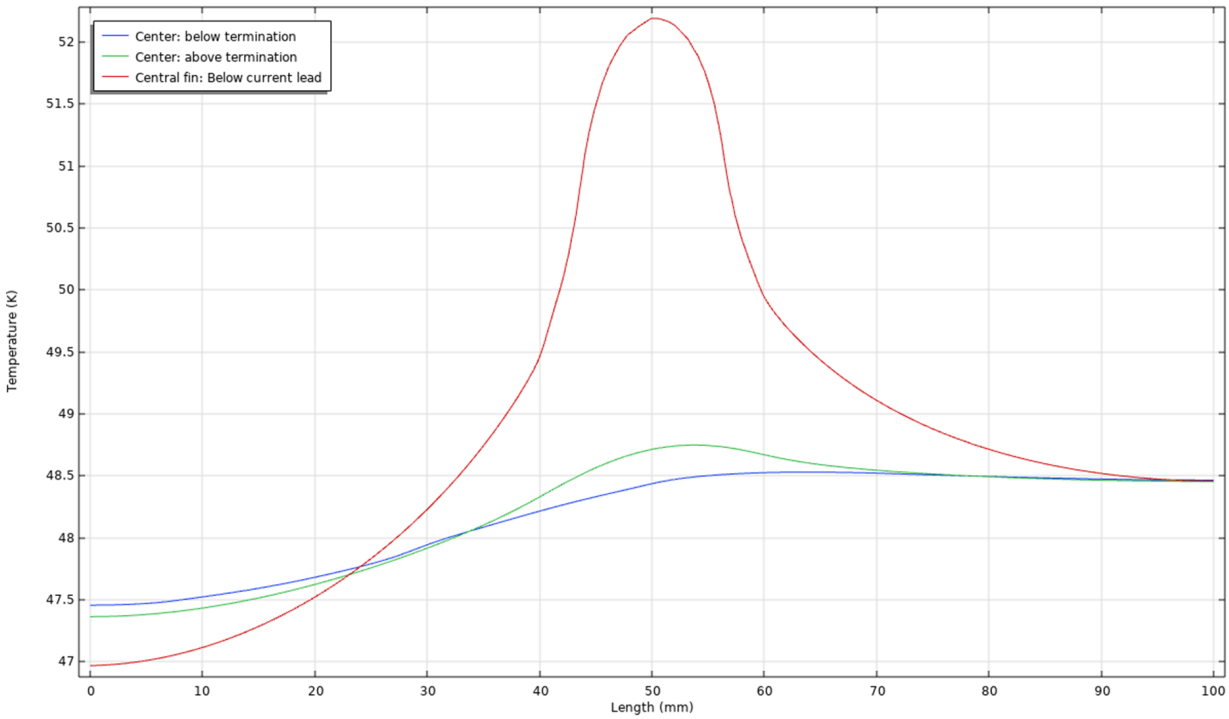


Figure A.5: Temperature profiles along the length of the concentric fin heat sink for $Sim\_T\_fix\_3$ simulation.

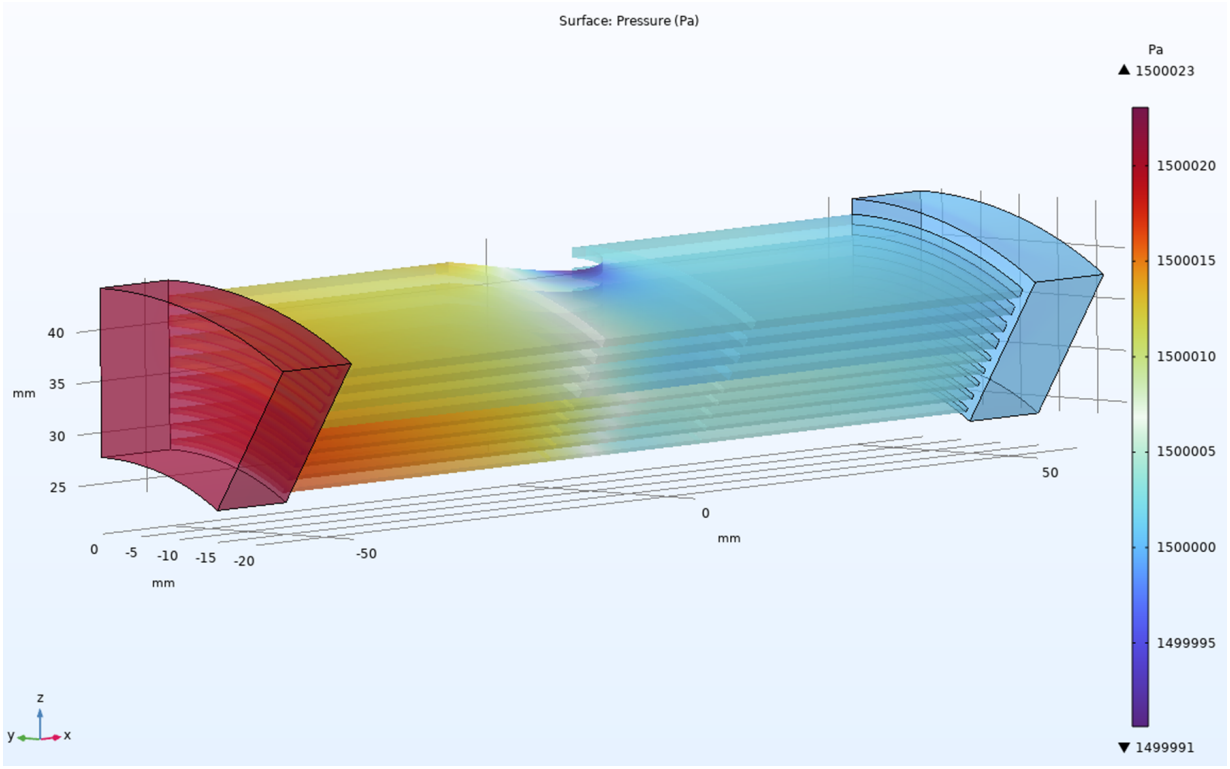


Figure A.6: The pressure field across the heat sink for $Sim\_T\_fix\_3$ simulation.

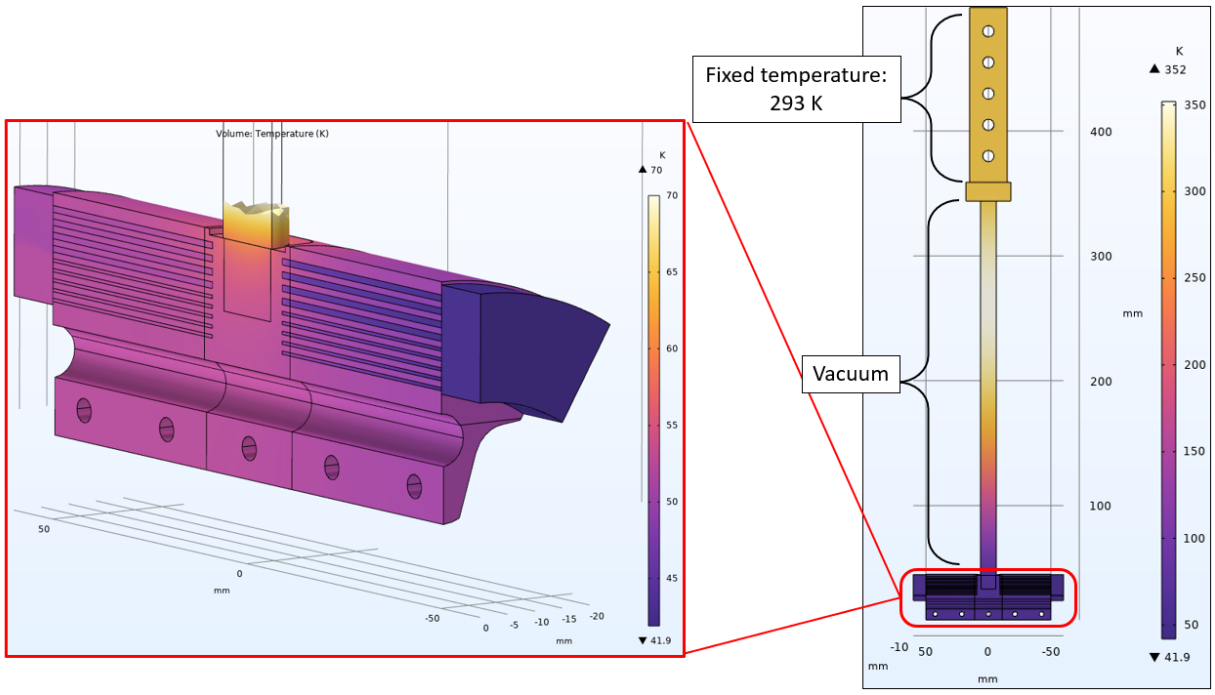


Figure A.7: Temperature distribution for $Sim\_T\_fix\_4$ simulation.

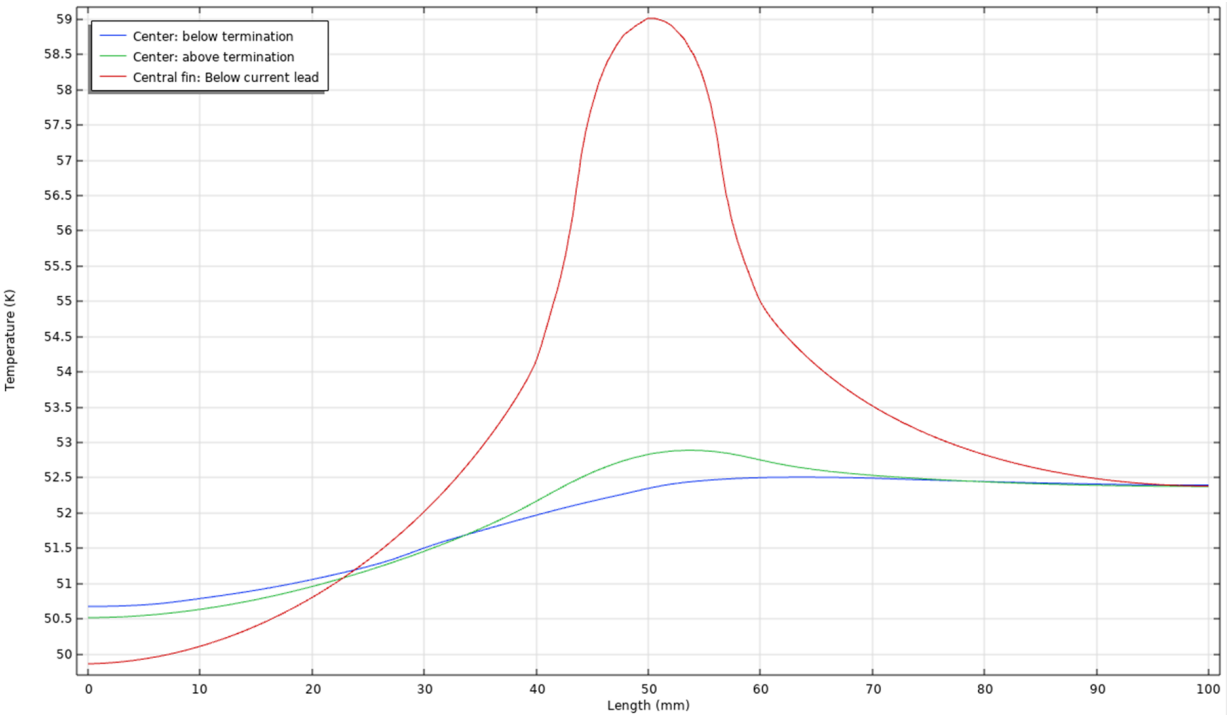


Figure A.8: Temperature profiles along the length of the concentric fin heat sink for $Sim\_T\_fix\_4$ simulation.

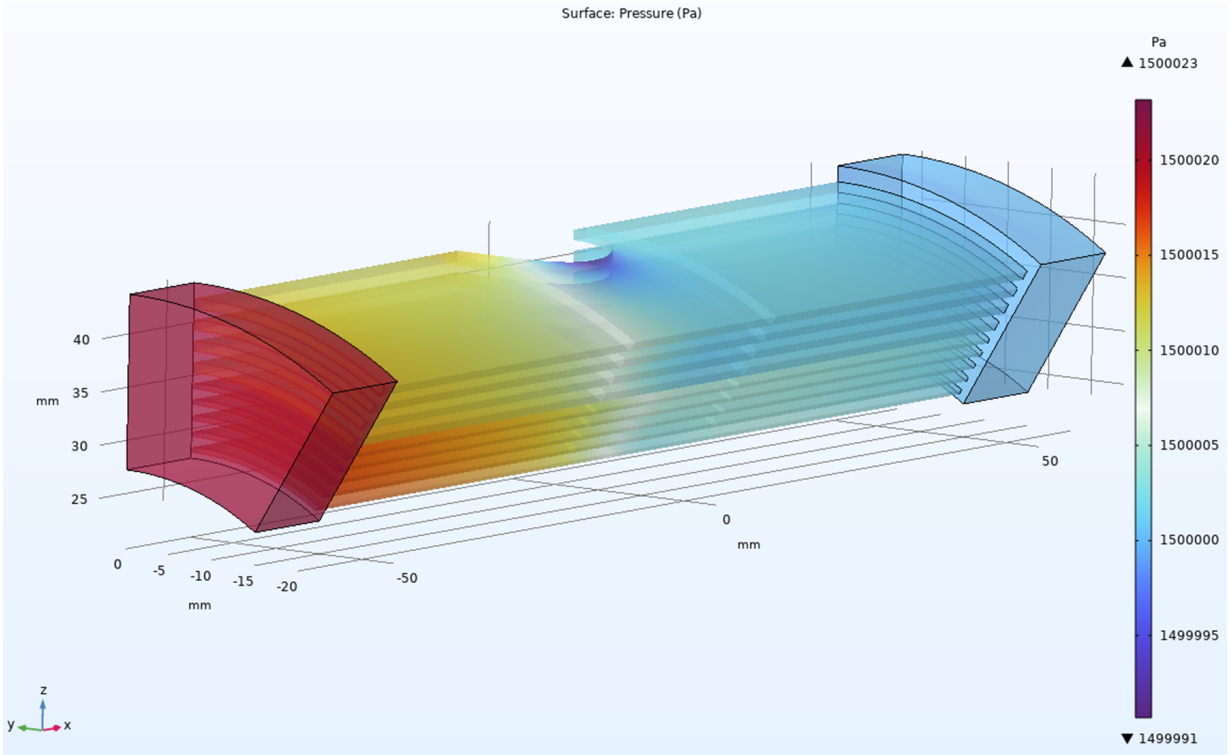


Figure A.9: The pressure field across the heat sink for $Sim\_T\_fix\_4$ simulation.

A.2. Natural Convection & Radiation Boundary Condition Simulations

This section contains the results of simulations with a natural convection and radiation boundary condition on the busbar.

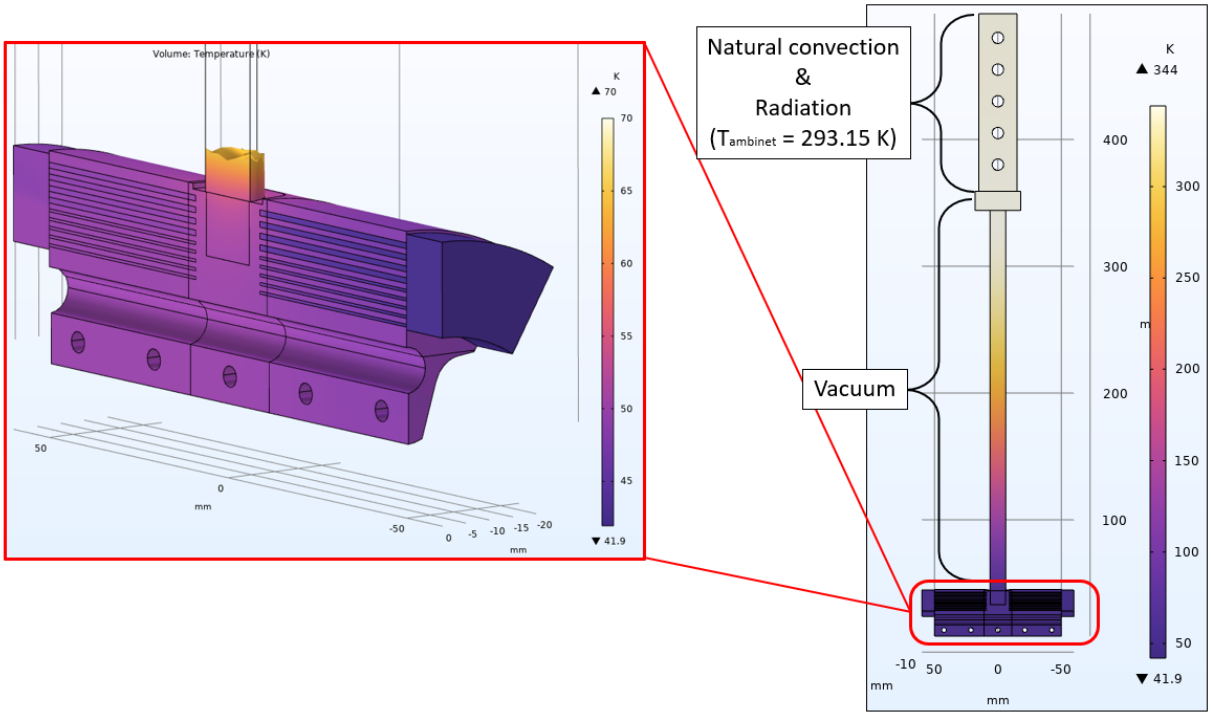


Figure A.10: Temperature distribution for Sim_convec_2 simulation.

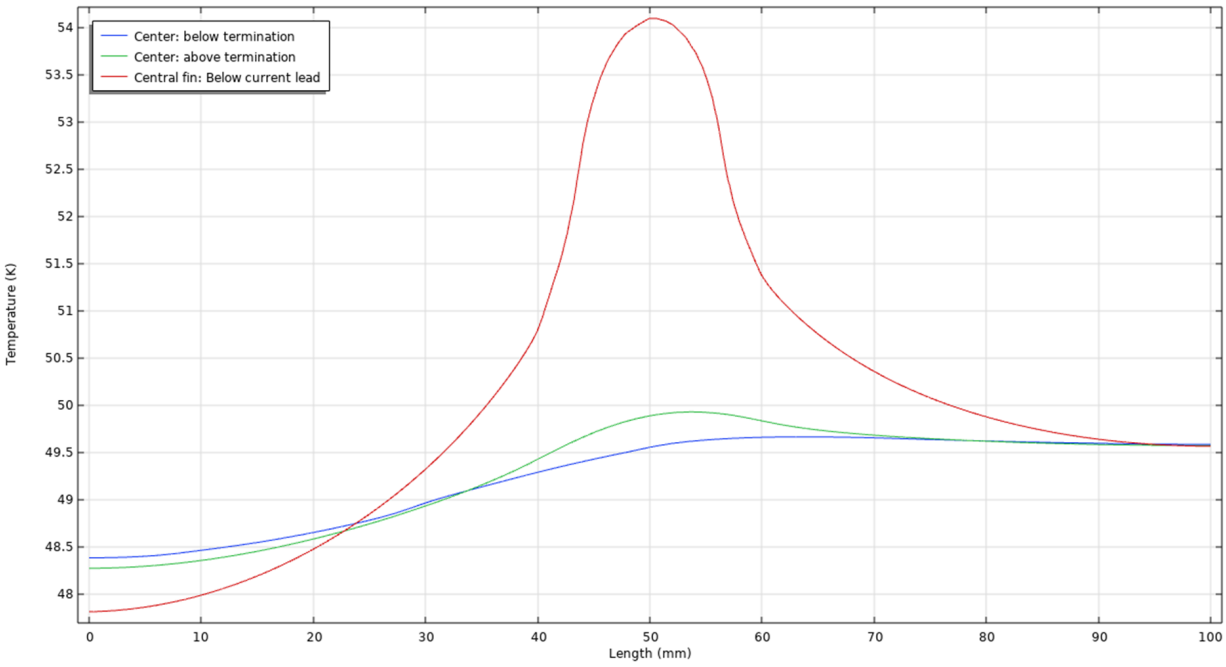


Figure A.11: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_2 simulation.

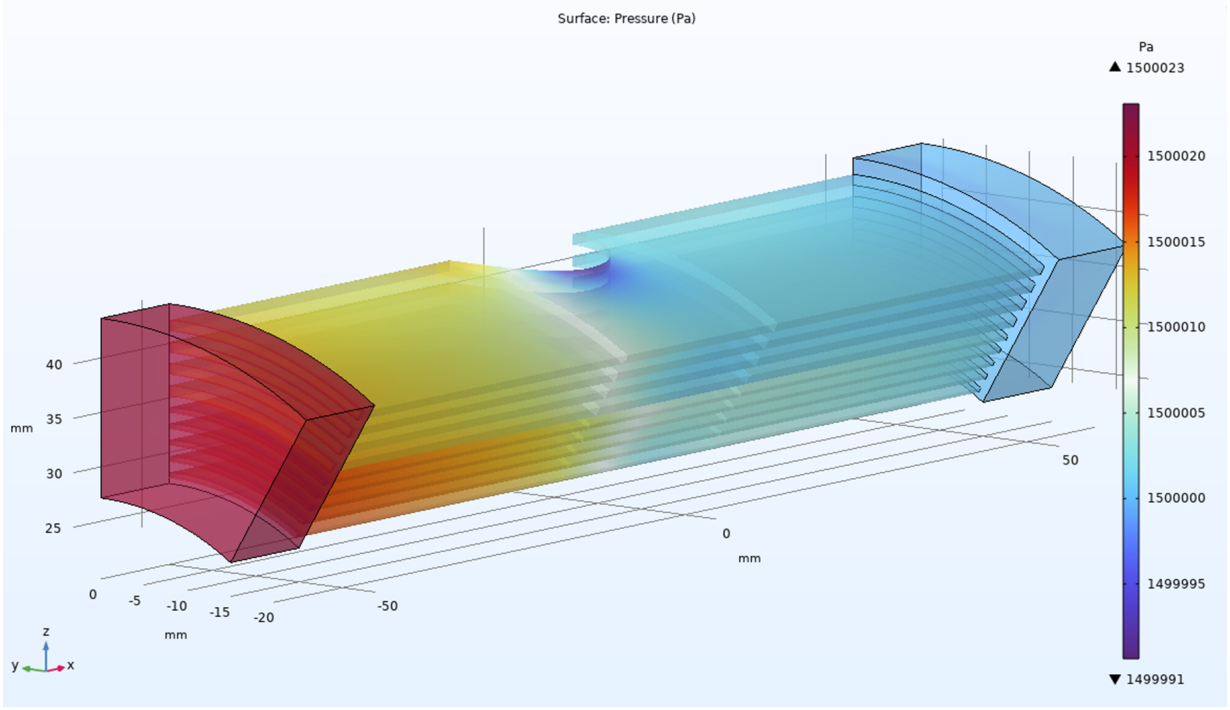


Figure A.12: The pressure field across the heat sink for Sim_convec_2 simulation.

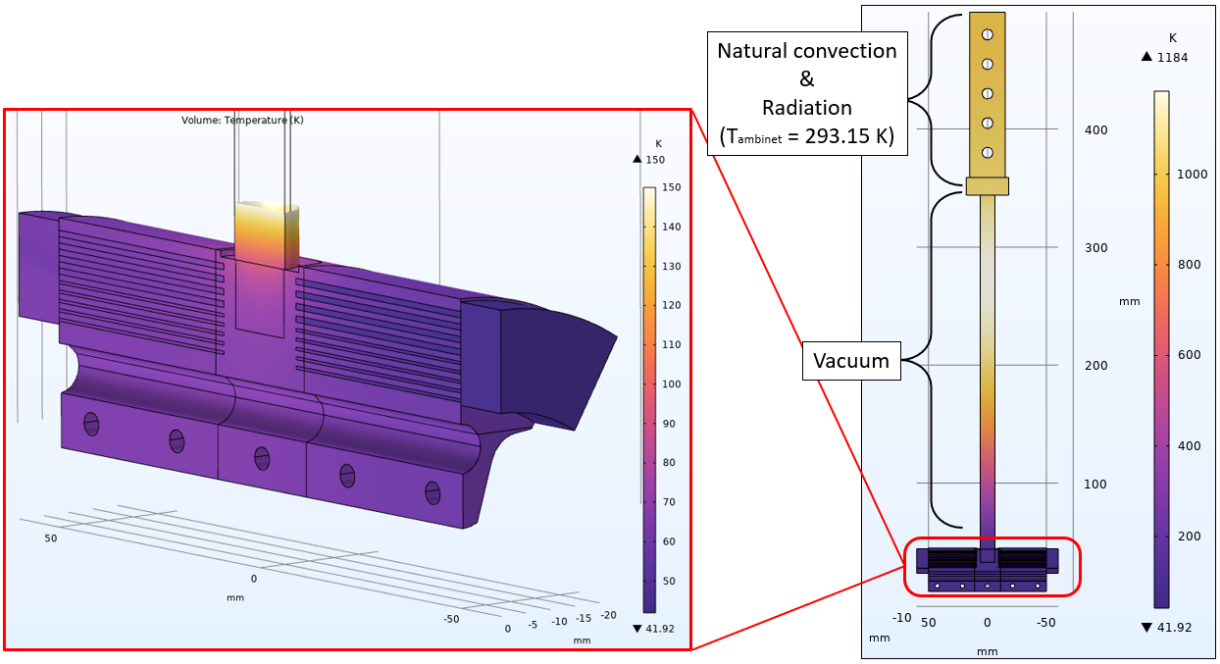


Figure A.13: Temperature distribution for Sim_convec_3 simulation.

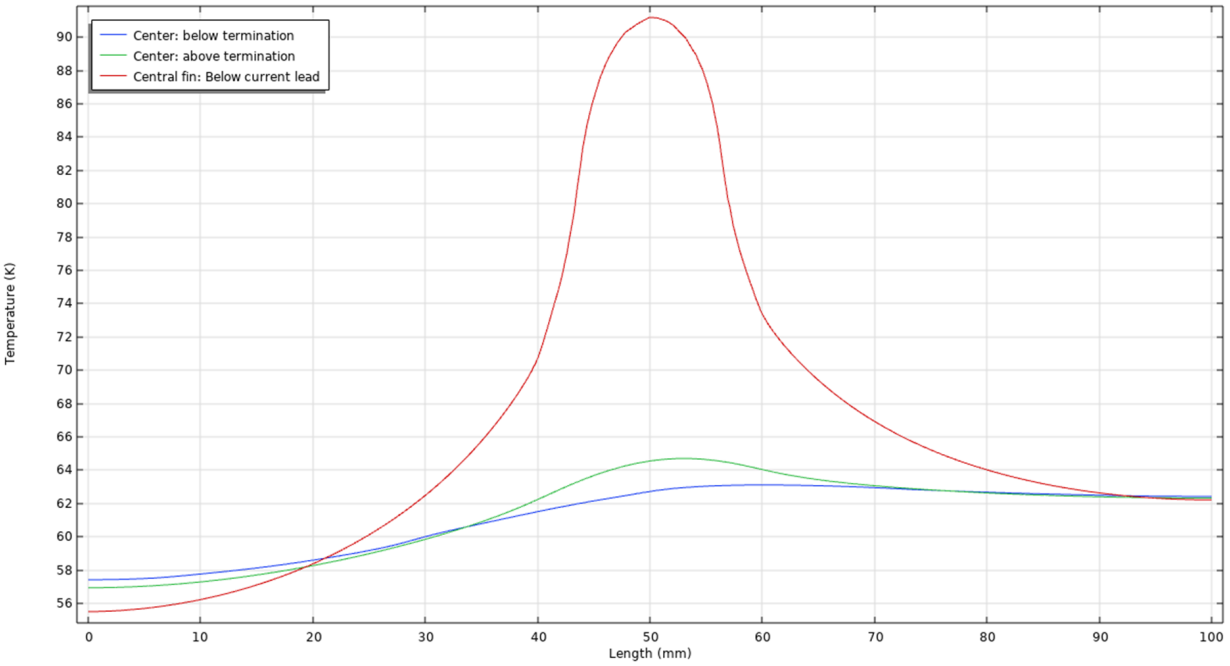


Figure A.14: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_3 simulation.

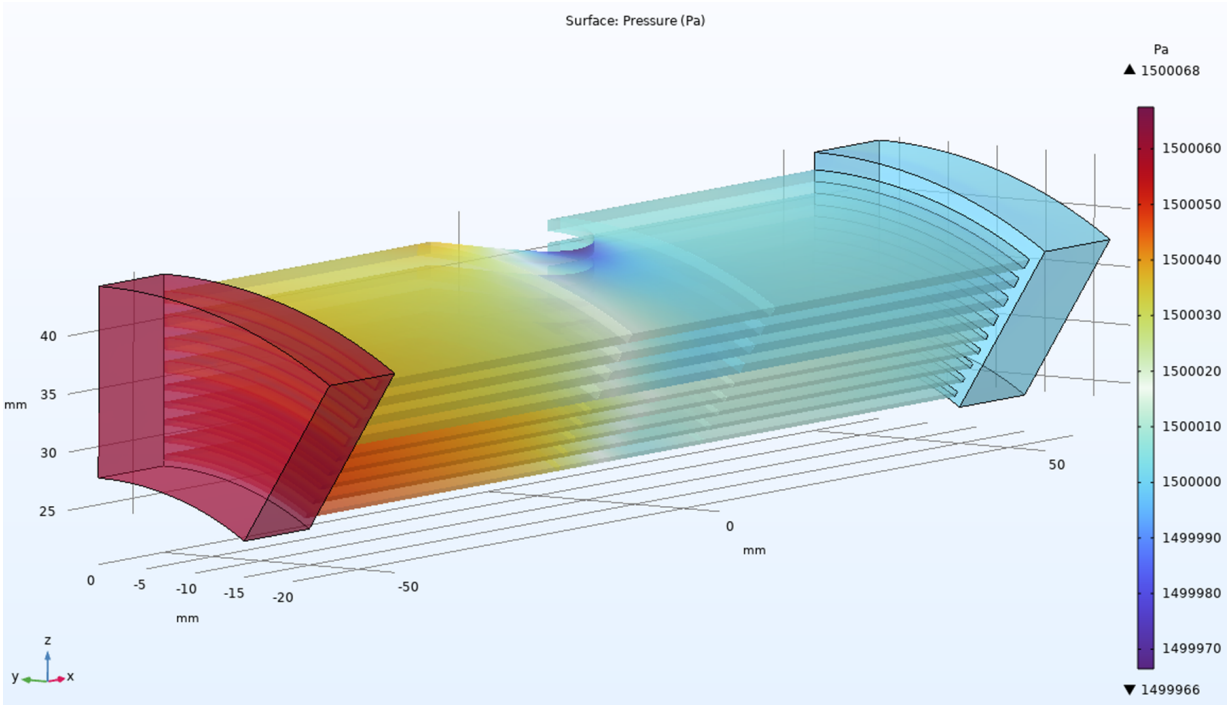


Figure A.15: The pressure field across the heat sink for Sim_convec_3 simulation.

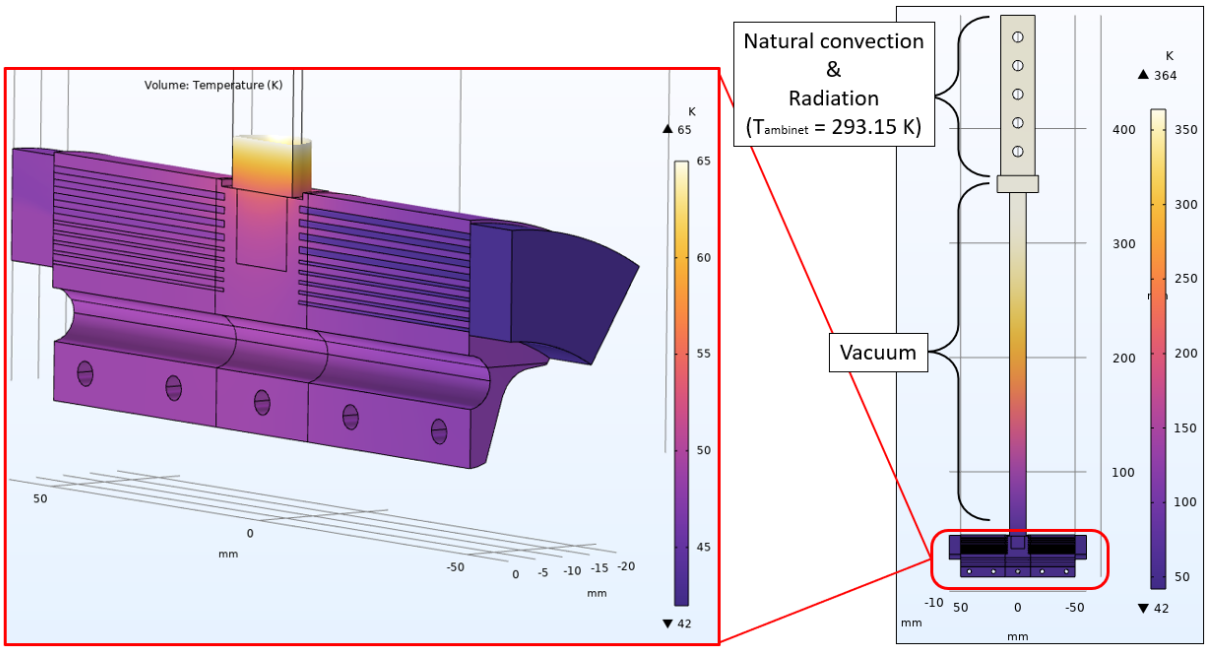


Figure A.16: Temperature distribution for Sim_convec_4 simulation.

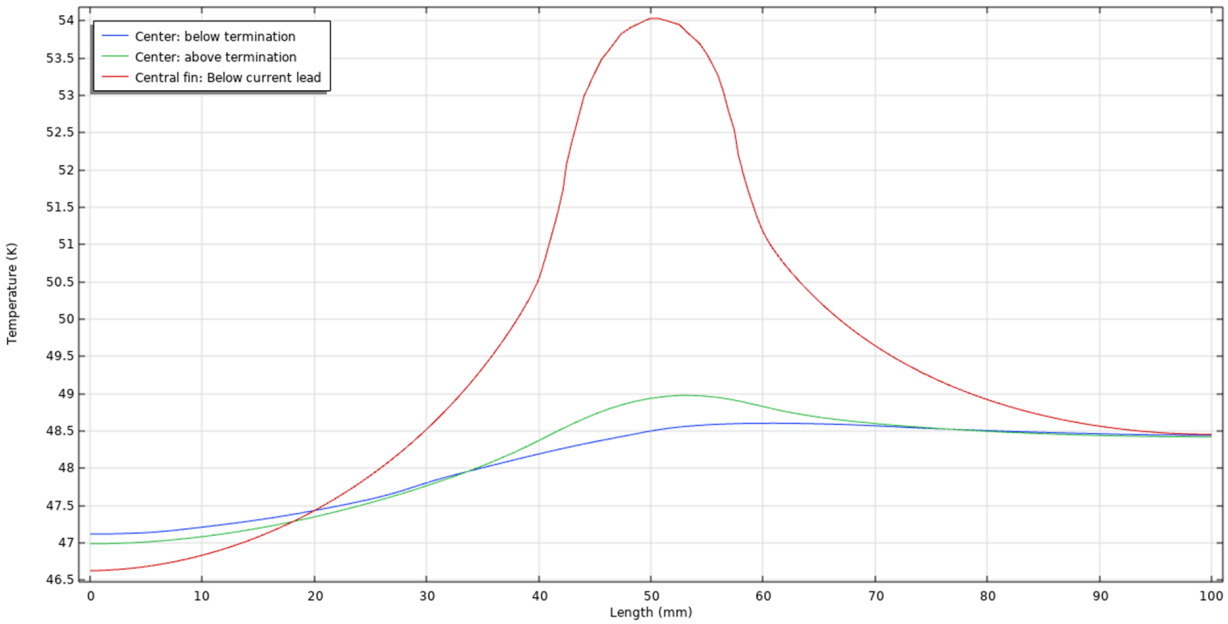


Figure A.17: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_4 simulation.

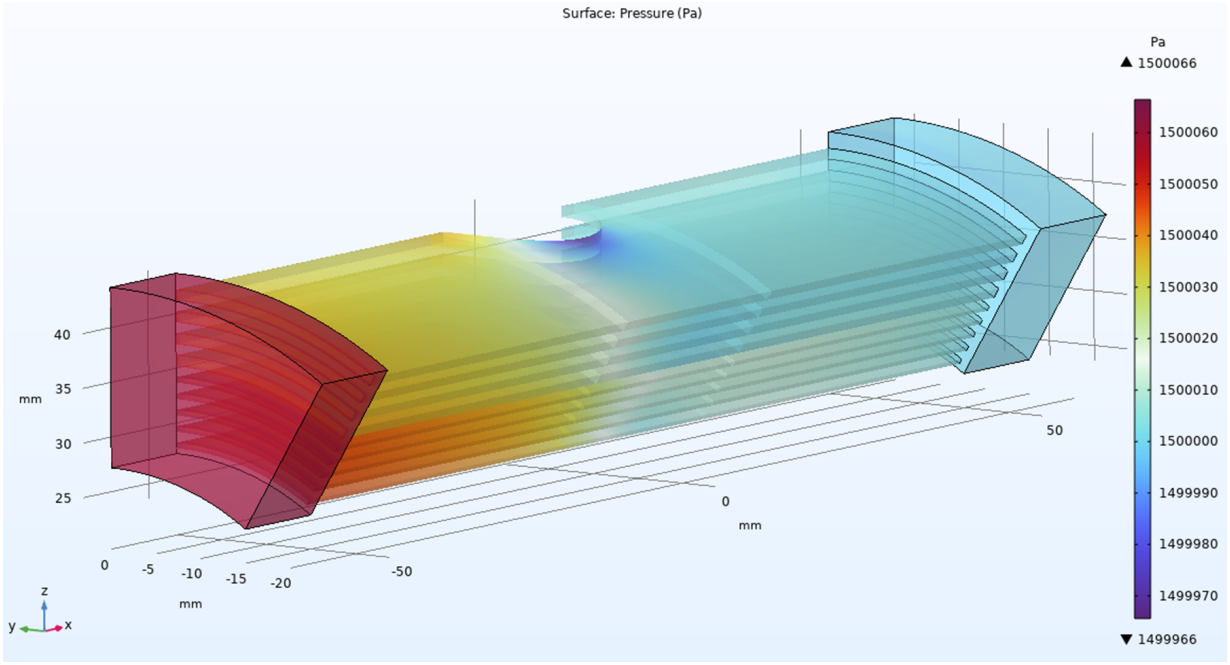


Figure A.18: The pressure field across the heat sink for Sim_convec_4 simulation.

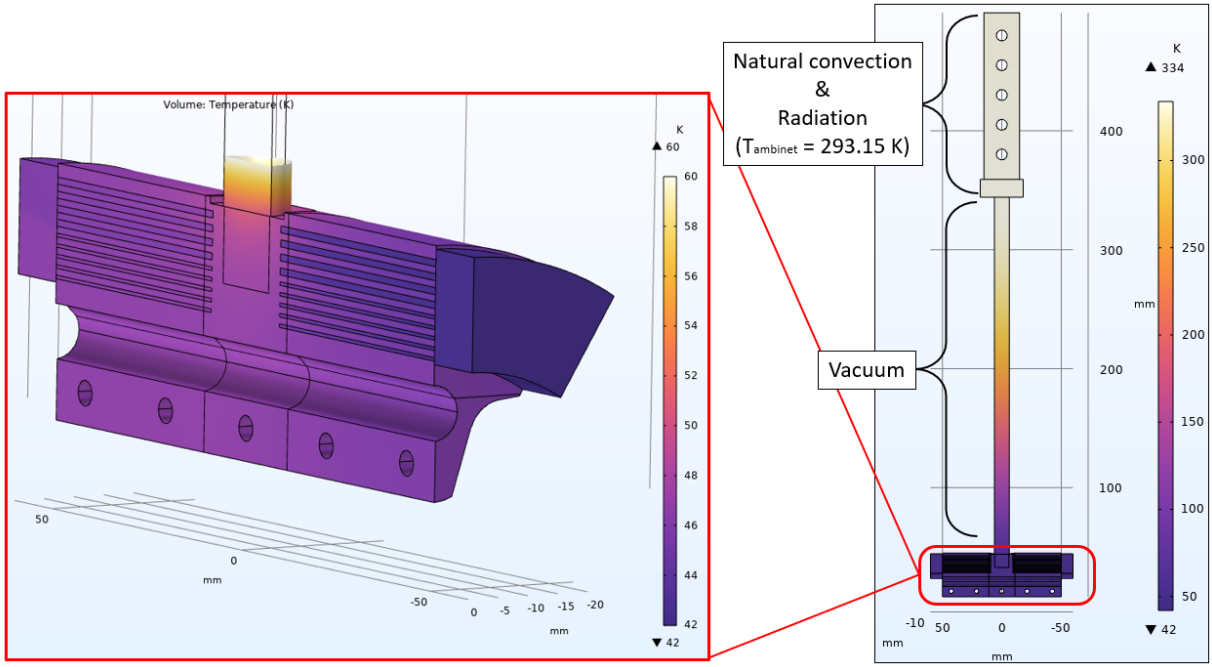


Figure A.19: Temperature distribution for Sim_convec_5 simulation.

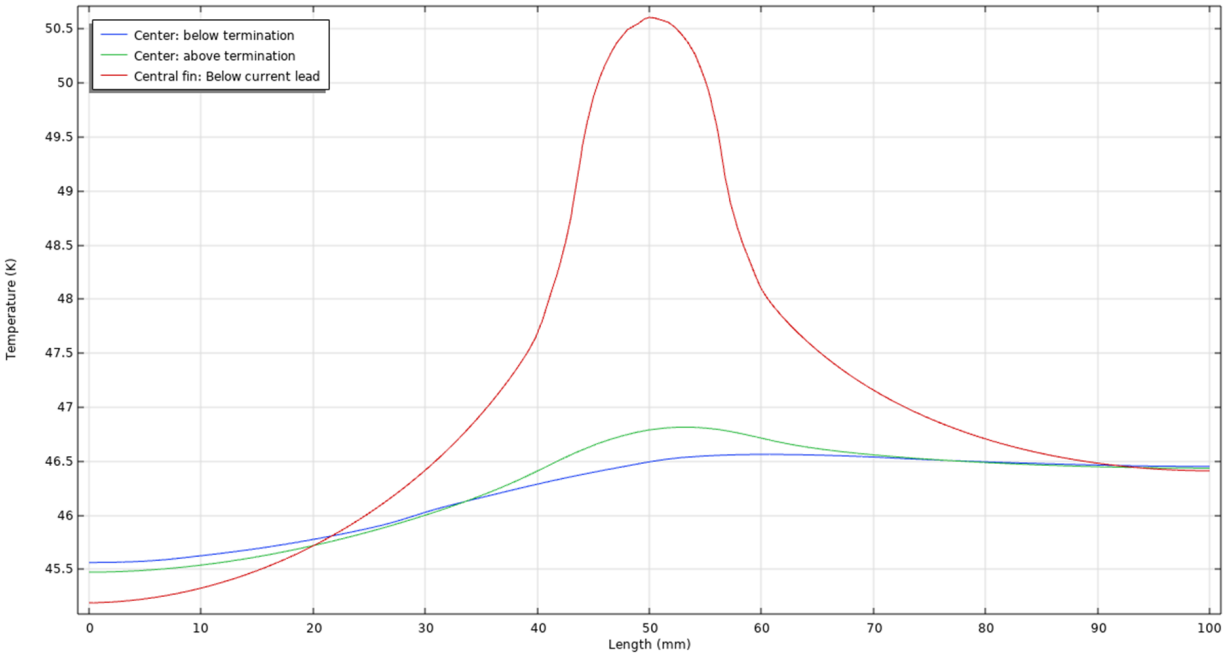


Figure A.20: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_5 simulation.

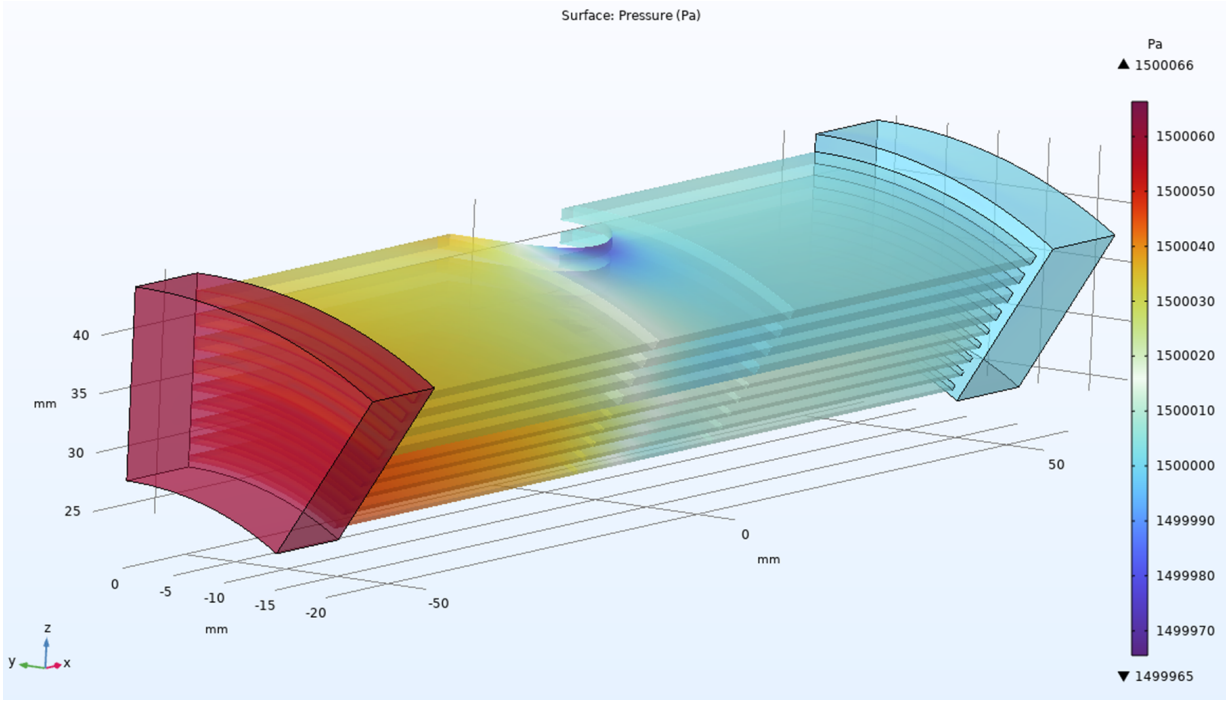


Figure A.21: The pressure field across the heat sink for Sim_convec_5 simulation.

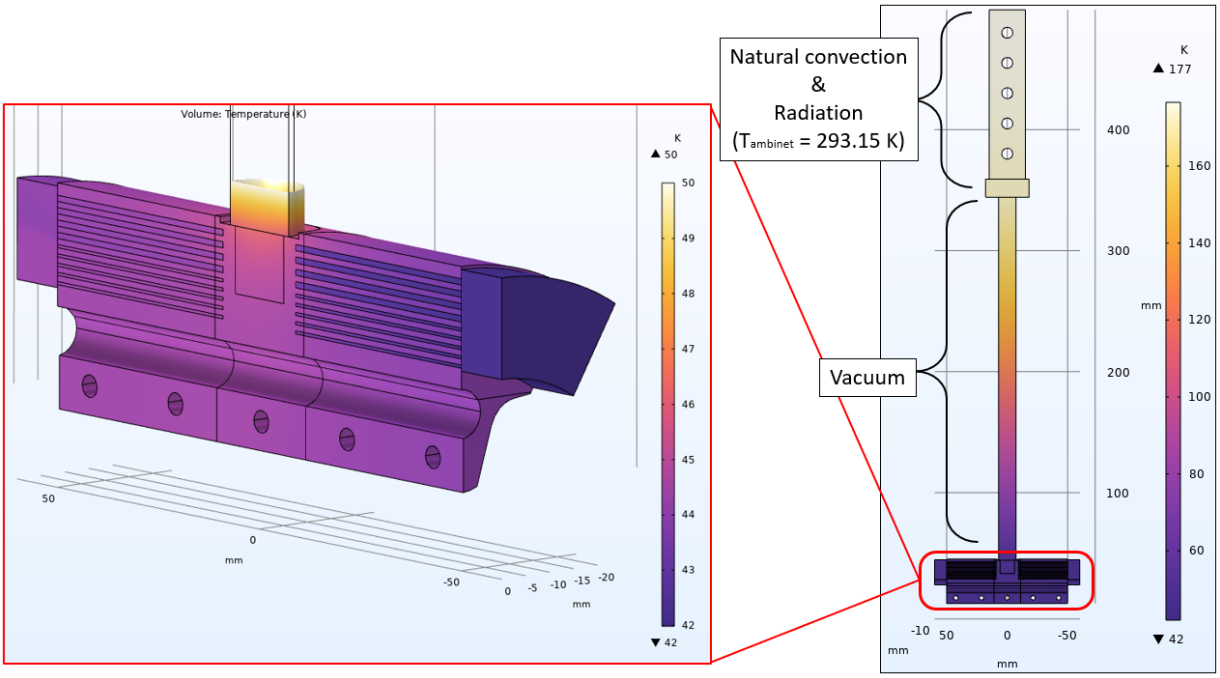


Figure A.22: Temperature distribution for Sim_convec_6 simulation.

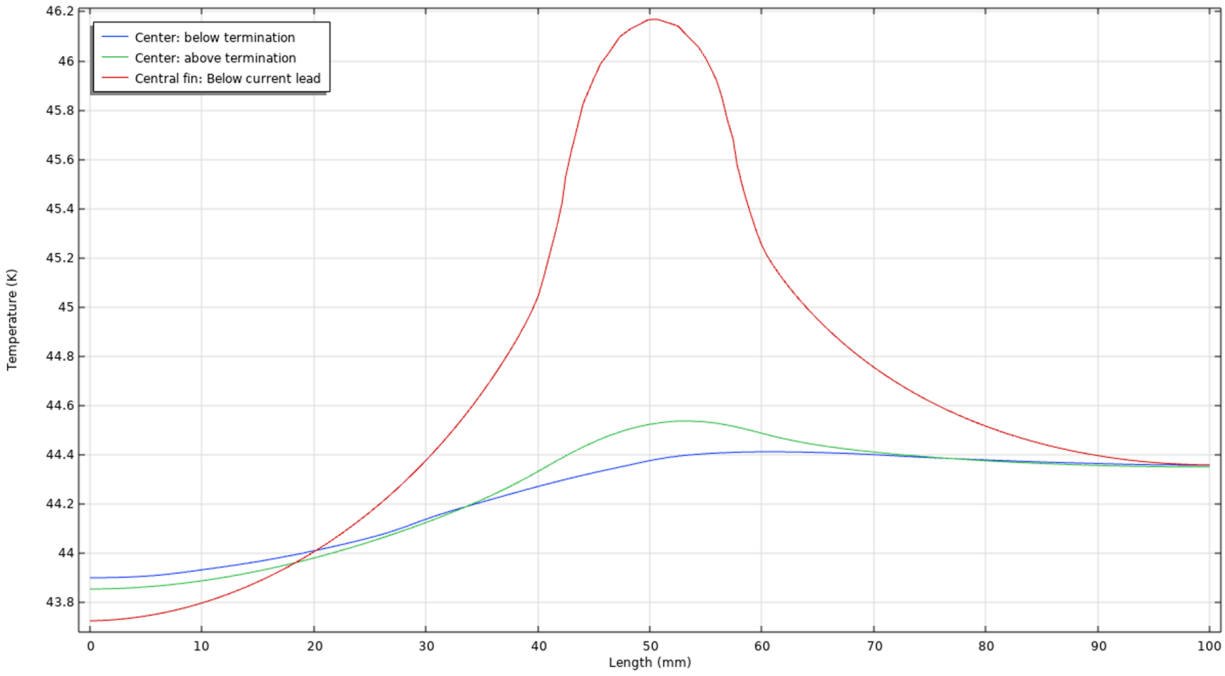


Figure A.23: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_6 simulation.

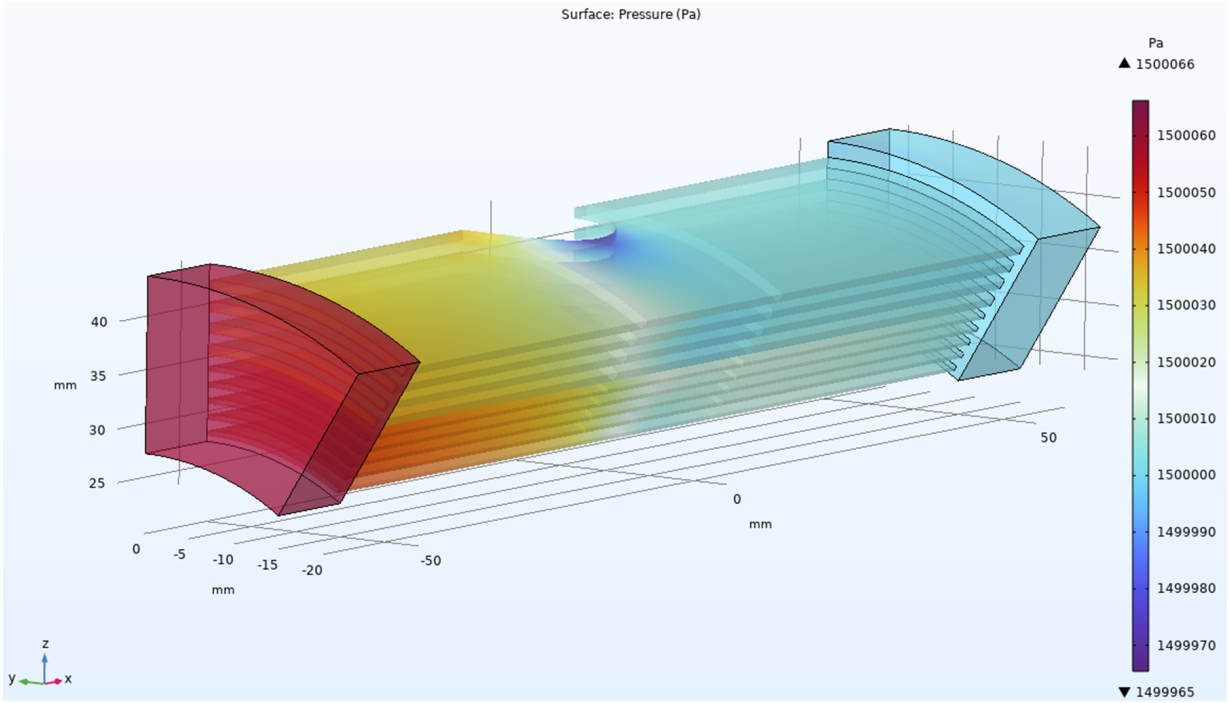


Figure A.24: The pressure field across the heat sink for Sim_convec_6 simulation.

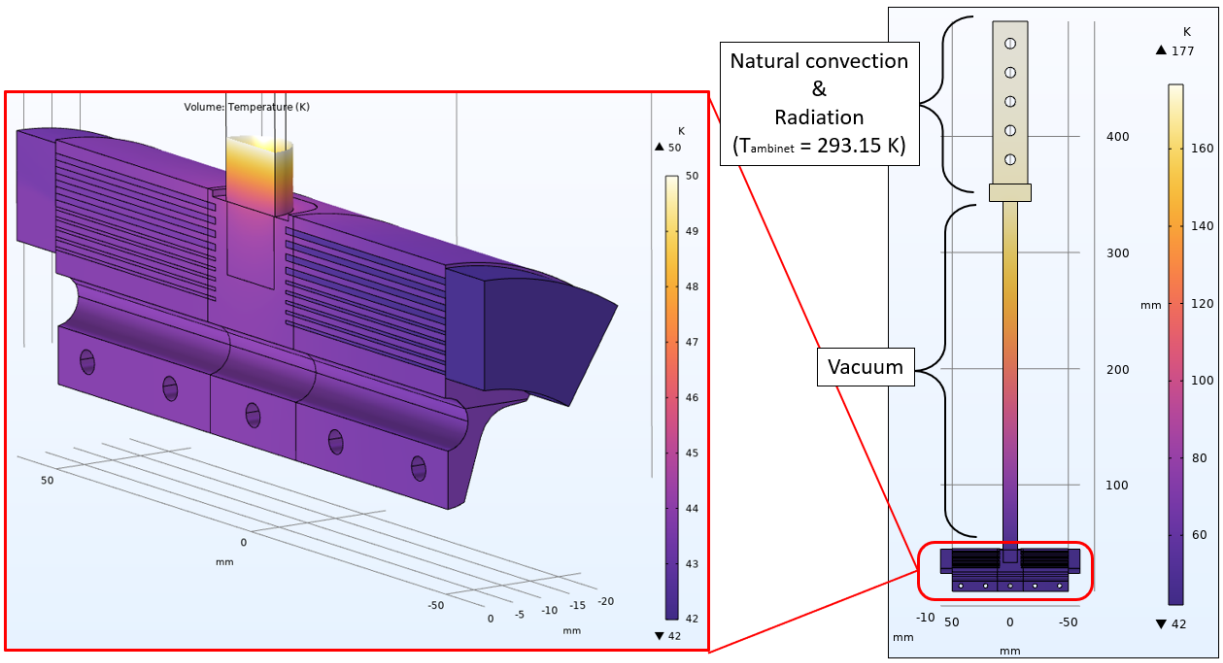


Figure A.25: Temperature distribution for Sim_convec_7 simulation.

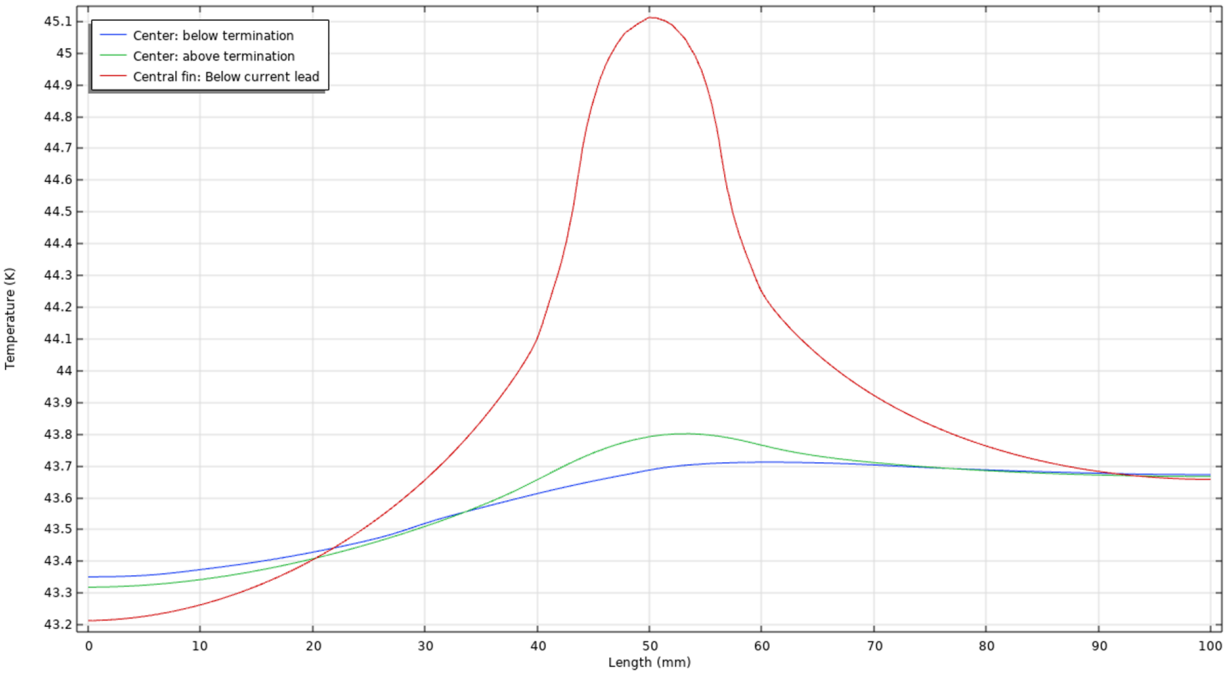


Figure A.26: Temperature profiles along the length of the concentric fin heat sink for Sim_convec_7 simulation.

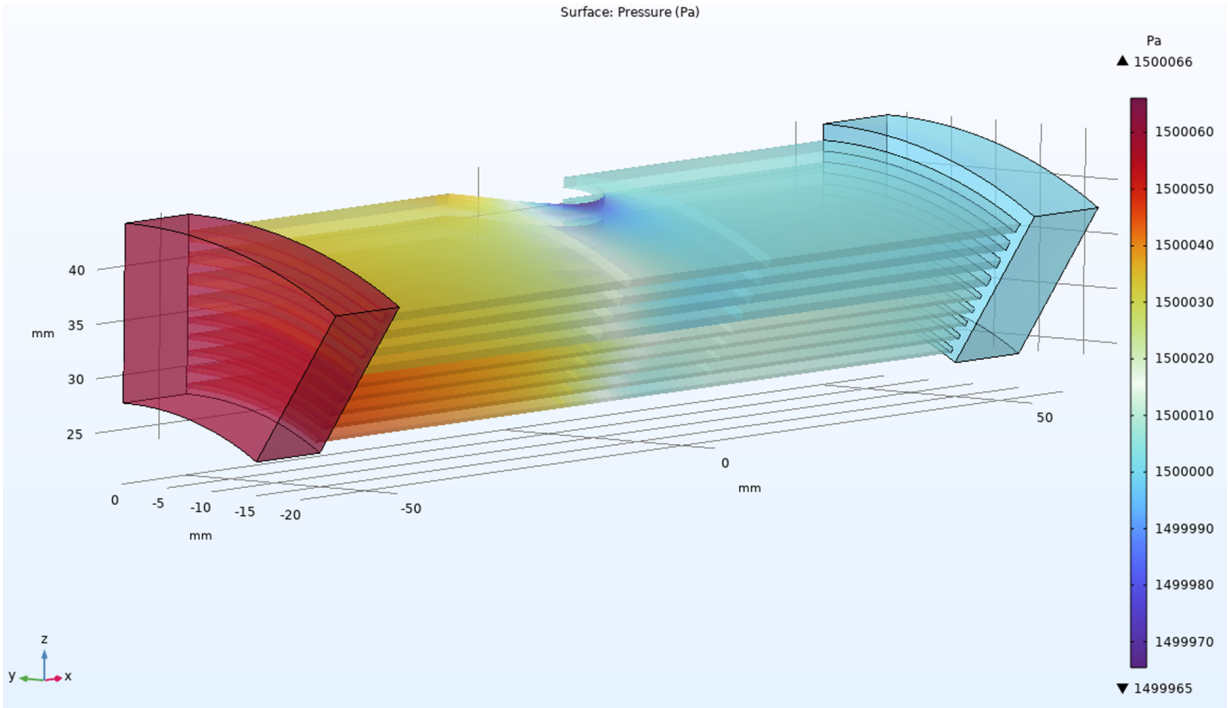


Figure A.27: The pressure field across the heat sink for Sim_convec_7 simulation.

Appendix B

This appendix contains the raw voltage measurements of the pressure sensors used in the heat sink experiment on October 22, 2025, and October 23, 2025.

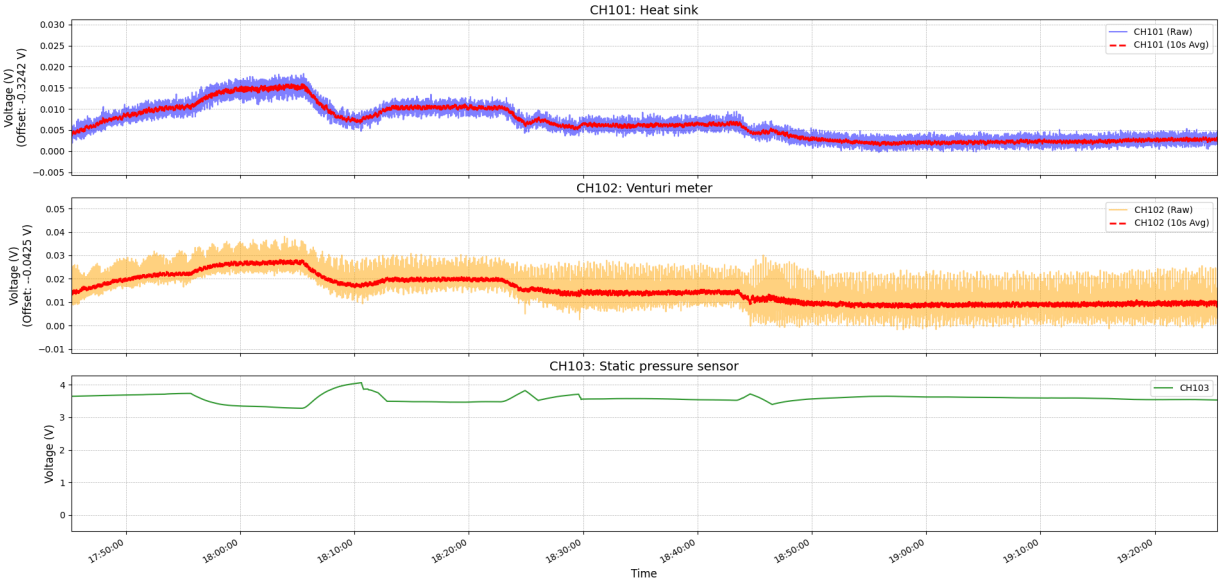


Figure B.1: Voltage readings of pressure sensors across the heat sink (CH 101), venturi meter (CH 102) and of static pressure sensor (CH 103) under different heating scenarios. [22/10/2025, Cryofan speed: 8036 rpm]

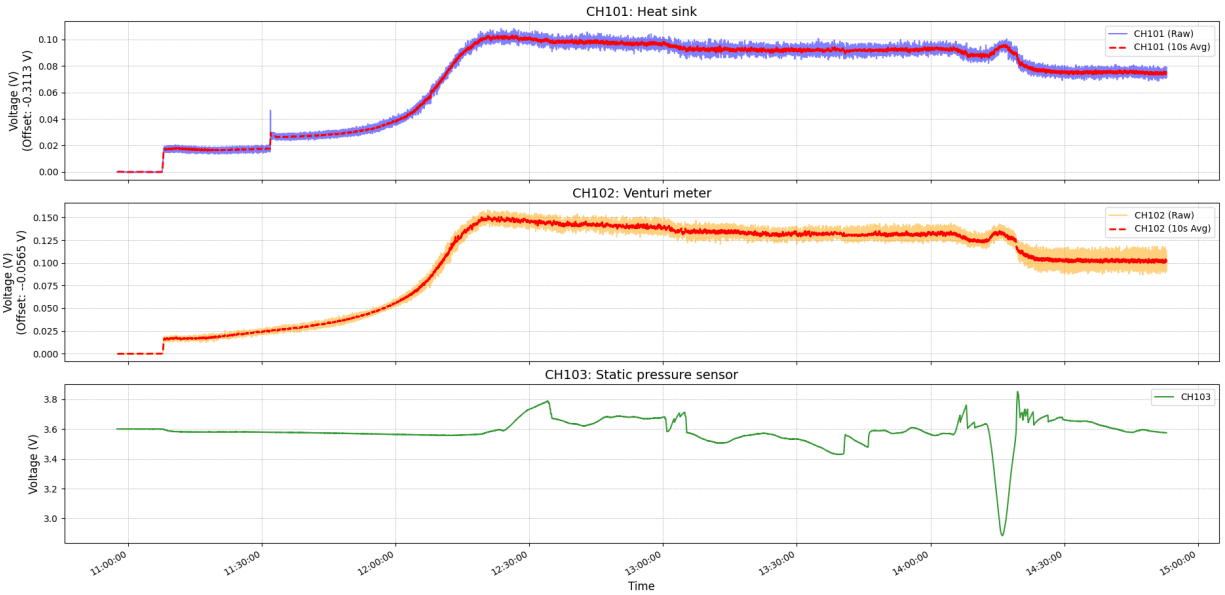


Figure B.2: Voltage readings of pressure sensors across the heat sink (CH 101), venturi meter (CH 102) and of static pressure sensor (CH 103) under different heating scenarios. [23/10/2025, Cryofan speed: 17600 rpm]

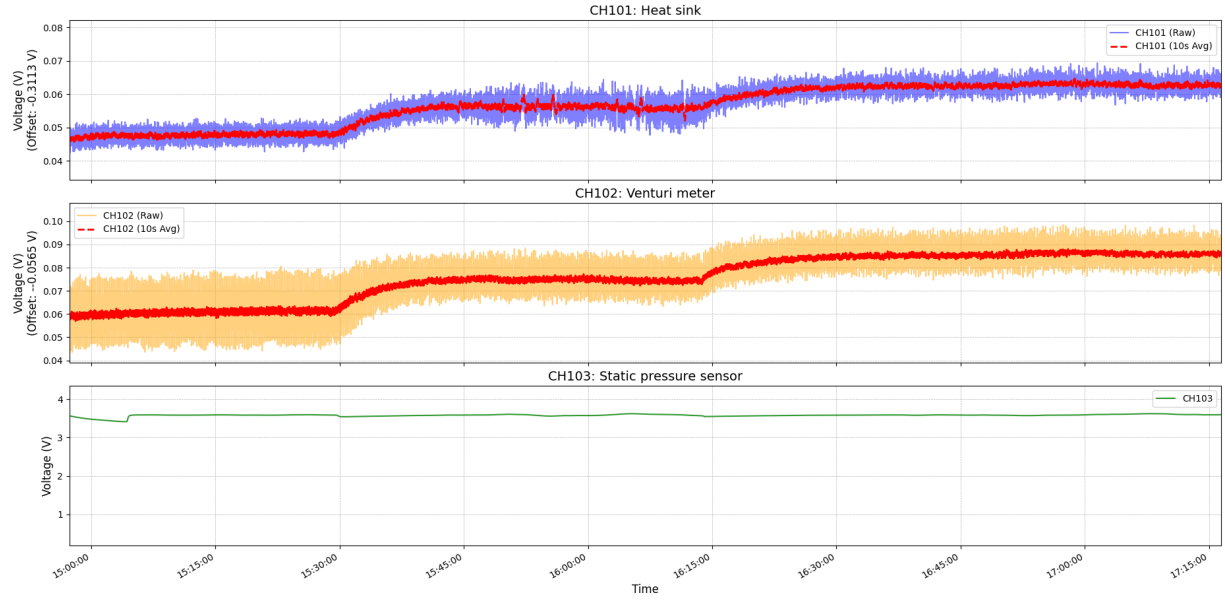


Figure B.3: Voltage readings of pressure sensors across the heat sink (CH 101), venturi meter (CH 102) and of static pressure sensor (CH 103) under different heating scenarios. [23/10/2025, Cryofan speed: 13100 rpm]

C

Appendix C

This appendix contains all the pyhton codes developed for the 1D numerical model of heat sink and the current lead.

C.1. 1D current lead python code

```
#Current lead sizer v5 Kunal G (with Aluminum option)

#MCfee cooling load Matthesian rule with wiedemann law with weight

import matplotlib
matplotlib.use('TkAgg')
import numpy as np
from scipy.integrate import quad, IntegrationWarning, cumulative_trapezoid
from scipy.interpolate import interp1d
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, CheckButtons
import warnings
import time
import tkinter as tk
from tkinter import ttk, messagebox
import threading
import queue
from PIL import Image, ImageTk
import sys
import os

# --- Helper function to find bundled files ---
def resource_path(relative_path):
    """ Get absolute path to resource, works for dev and for PyInstaller """
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = os.path.abspath(".")
    return os.path.join(base_path, relative_path)

# --- Suppress routine integration warnings from quad ---
warnings.filterwarnings("ignore", category=IntegrationWarning)

# --- NEW: Define Material Properties in a Dictionary ---
def rho_intrinsic_poly_copper(T):
    """Polynomial for intrinsic electrical resistivity of Copper in nΩ·m."""
```

```
    return (0.00000000000000010025723664907 5*T**6 -
    ↪  0.00000000000043881458946066700 0*T**5 + 0.00000000073796586204307600000 0*T**4 -
    ↪  0.00000058575885252176900000000 0*T**3 + 0.00022646614963385400000000000 0*T**2 +
    ↪  0.02980300891333570000000000000 0*T -
    ↪  1.20851594355549000000000000000)#(0.00000000000000010025723664907 5*T**6 -
    ↪  0.00000000000043881458946066700 0*T**5 + 0.00000000073796586204307600000 0*T**4 -
    ↪  0.00000058575885252176900000000 0*T**3 + 0.00022646614963385400000000000 0*T**2 +
    ↪  0.02980300891333570000000000000 0*T - 1.20851594355549000000000000000)

def rho_intrinsic_poly_aluminum(T):
    """Polynomial for intrinsic electrical resistivity of Aluminum in nΩ·m."""
    return (0.00000000000001496391367310930 0*T**6 -
    ↪  0.00000000031730226224731800000 0*T**5 + 0.00000025603706132438200000000 0*T**4 -
    ↪  0.00009929236721940880000000000 0*T**3 + 0.00196888451825619000000000000 0*T**2 -
    ↪  0.08155856643932100000000000000 0*T +
    ↪  0.68795365928211500000000000000)#(0.00000000000001496391367310930 0*T**6 -
    ↪  0.00000000031730226224731800000 0*T**5 + 0.00000025603706132438200000000 0*T**4 -
    ↪  0.00009929236721940880000000000 0*T**3 + 0.00196888451825619000000000000 0*T**2 -
    ↪  0.08155856643932100000000000000 0*T + 0.68795365928211500000000000000)

MATERIALS = {
    "Copper": {
        "rho_poly": rho_intrinsic_poly_copper,
        "density": 8960,  # kg/m^3
        "T_melting": 1350,
    },
    "Aluminum": {
        "rho_poly": rho_intrinsic_poly_aluminum,
        "density": 2700,  # kg/m^3
        "T_melting": 660
    }
}

# --- NEW FUNCTION: To print material properties (MODIFIED) ---
def print_material_properties(rrr_val, material):
    """
    Calculates and prints a table of thermal conductivity and electrical
    resistivity for a given RRR value and material from 20 K to 1350 K.
    """
    print("\n" + "="*70)
    print(f"Material Properties for {material} with RRR = {rrr_val}")
    print("="*70)
    print(f"{'Temperature (K)':<20} | {'Thermal Conductivity (W/m·K)':<30} |
    ↪  {'Electrical Resistivity (Ω·m)':<25}")
    print("-"*70)

    # --- Reusing the physical models, now selected by material ---
    rho_intrinsic_poly = MATERIALS[material]["rho_poly"]
    melting_temp = MATERIALS[material]["T_melting"]

    T_RT = 293.15
    rho_intrinsic_RT = rho_intrinsic_poly(T_RT)
    rho_residual = rho_intrinsic_RT / (rrr_val - 1) if rrr_val > 1 else
    ↪  rho_intrinsic_RT


    print("The material is: ",material)
```

```
    T_low = np.arange(20, melting_temp+1, 1)

    rho_total_nano_low = rho_residual + rho_intrinsic_poly(T_low)
    rho_total_low = rho_total_nano_low * 1e-9
    LORENZ_NUMBER = 2.44e-8
    # k_low = (LORENZ_NUMBER * T_low) / rho_total_low

    # k is the thermal resistivity!!!
    k_low = 1/((LORENZ_NUMBER * T_low) / rho_total_low )    # (m² K/W)


    T_extended = T_low
    k_extended = k_low
    rho_extended = rho_total_low

    # Create interpolators to get smooth data at any temperature
    k_interp = interp1d(T_extended, k_extended, kind='linear', bounds_error=False,
    ↪  fill_value="extrapolate")




    rho_interp = interp1d(T_extended, rho_extended, kind='linear', bounds_error=False,
    ↪  fill_value="extrapolate")

    # Define the temperature range for printing
    temp_range_to_print = np.arange(1, 600, 2)

    for T in temp_range_to_print:
        k_val = k_interp(T)
        rho_val = rho_interp(T)
        print(f"{T:<20} | {k_val:<30.4f} | {rho_val:<25.4e}")

    print("="*70 + "\n")

# --- Calculation Functions (MODIFIED) ---
def get_interpolators(rrr_val, material):
    rho_intrinsic_poly = MATERIALS[material]["rho_poly"]
    melting_temp = MATERIALS[material]["T_melting"]
    T_RT = 293.15
    rho_intrinsic_RT = rho_intrinsic_poly(T_RT)
    rho_residual = rho_intrinsic_RT / (rrr_val - 1) if rrr_val > 1 else
    ↪  rho_intrinsic_RT
    T_low = np.arange(20, melting_temp, 1)

    rho_total_nano_low = rho_residual + rho_intrinsic_poly(T_low)
    rho_total_low = rho_total_nano_low * 1e-9
    LORENZ_NUMBER = 2.44e-8
    #k_low = (LORENZ_NUMBER * T_low) / rho_total_low

    #     # k is the thermal resistivity!!!
    k_low = 1/((LORENZ_NUMBER * T_low) / rho_total_low )    # (m² K/W)
    T_extended = T_low
    k_extended = k_low
    rho_extended = rho_total_low
```

```
    k_interp = interp1d(T_extended, k_extended, kind='linear', bounds_error=False,
    ↪  fill_value="extrapolate")
    k_rho_product = k_extended * rho_extended
    C_values = cumulative_trapezoid(k_rho_product, T_extended, initial=0)
    C_interp = interp1d(T_extended, C_values, kind='cubic', bounds_error=False,
    ↪  fill_value='extrapolate')
    def definite_C_interp(T):
        return C_interp(T) - C_interp(20)
    return k_interp, definite_C_interp

def calculate_full_curve(I, T_L, T_H, k_interp, C_interp,material):
    melting_temp = MATERIALS[material]["T_melting"]
    T_p_array = np.logspace(np.log10(T_H + 1e-6), np.log10(melting_temp+1), 75)
    L_A_over, Q_L_over = [], []
    def integrand_over(T, T_p):
        denominator_sq = 2 * I**2 * (C_interp(T_p) - C_interp(T))
        if denominator_sq <= 1e-30: return 0.0
        return k_interp(T) / np.sqrt(denominator_sq)
    for T_p in T_p_array:
        try:
            Q_L_sq = 2 * I**2 * (C_interp(T_p) - C_interp(T_L))
            if Q_L_sq < 0: continue
            Q_L = np.sqrt(Q_L_sq)
            integral_1, _ = quad(integrand_over, T_L, T_p, args=(T_p,))
            integral_2, _ = quad(integrand_over, T_H, T_p, args=(T_p,))
            L_A = (integral_1 + integral_2)
            L_A_over.append(L_A)
            Q_L_over.append(Q_L)
        except (IntegrationWarning, ValueError): continue
    try:
        Q_L_min = np.sqrt(2 * I**2 * (C_interp(T_H) - C_interp(T_L)))
    except ValueError: Q_L_min = 0
    def integrand_under(T, Q_L, current_I):
        denominator_sq = Q_L**2 - 2 * current_I**2 * (C_interp(T) - C_interp(T_L))
        if denominator_sq <= 1e-30: return 0.0
        return k_interp(T) / np.sqrt(denominator_sq)
    Q_L_under_array = np.logspace(np.log10(Q_L_min * 1.001), np.log10(Q_L_min * 10),
    ↪  75) if Q_L_min > 0 else []
    L_A_under, successful_Q_L_under = [], []
    for q_val in Q_L_under_array:
        try:
            L_A, err = quad(integrand_under, T_L, T_H, args=(q_val, I))
            if not np.isnan(L_A) and err / (L_A + 1e-9) < 0.1:
                L_A_under.append(L_A)
                successful_Q_L_under.append(q_val)
        except (IntegrationWarning, ValueError): continue
    full_L_A = np.array(L_A_under + L_A_over[::-1])
    full_Q_L = np.concatenate([successful_Q_L_under, np.array(Q_L_over[::-1])])
    full_L_A_cm = full_L_A / 100.0
    return full_L_A_cm, full_Q_L

def calculate_point_for_Tp(I, T_p, T_L, T_H, k_interp, C_interp):
    def integrand_over(T, peak_T):
        denominator_sq = 2 * I**2 * (C_interp(peak_T) - C_interp(T))
        if denominator_sq <= 1e-30: return 0.0
        return k_interp(T) / np.sqrt(denominator_sq)
```

```
    try:
        Q_L_sq = 2 * I**2 * (C_interp(T_p) - C_interp(T_L))
    except ValueError: return None, None
    if Q_L_sq < 0: return None, None
    Q_L = np.sqrt(Q_L_sq)
    try:
        integral_1, _ = quad(integrand_over, T_L, T_p, args=(T_p,))
        integral_2, _ = quad(integrand_over, T_H, T_p, args=(T_p,))
        L_A = (integral_1 + integral_2)
        L_A_cm = L_A / 100.0
        return L_A_cm, Q_L
    except (ValueError, IntegrationWarning): return None, None

def precompute_all_data(RRR_list, T_L_range, T_H, currents_kA, target_temps, material,
↪  progress_queue):
    storage = {}
    total_calcs = len(RRR_list) * len(T_L_range)
    current_calc = 0
    for rrr in RRR_list:
        storage[rrr] = {}
        # Pass material to get the correct interpolators
        k_interp, C_interp = get_interpolators(rrr, material)
        for tl in T_L_range:
            current_calc += 1
            print(f"Pre-calculating {material}... RRR={rrr}, T_L={tl} K
            ↪  ({current_calc}/{total_calcs})")
            case_data = {'current_curves': [], 'isotherms': [], 'optimal': None,
            ↪  'conduction': None}
            for I_kA in currents_kA:
                I = I_kA * 1000
                l_a, q_l = calculate_full_curve(I, tl, T_H, k_interp,
                ↪  C_interp,material)
                case_data['current_curves'].append({'l_a': l_a, 'q_l': q_l, 'I_kA':
                ↪  I_kA})
            for temp in target_temps:
                isotherm_points = []
                for I_kA in sorted(currents_kA):
                    I = I_kA * 1000
                    l_a, q_l = calculate_point_for_Tp(I, temp, tl, T_H, k_interp,
                    ↪  C_interp)
                    if l_a is not None: isotherm_points.append({'l_a': l_a, 'q_l':
                    ↪  q_l})
                case_data['isotherms'].append({'temp': temp, 'points':
                ↪  isotherm_points})
            try:
                opt_I_range = np.logspace(np.log10(50), np.log10(50000), 100)
                opt_L_A_cm, opt_Q_L = [], []
                def integrand_opt(T, T_H_val):
                    denominator_sq = 2 * (C_interp(T_H_val) - C_interp(T))
                    if denominator_sq <= 1e-30: return 0.0
                    return k_interp(T) / np.sqrt(denominator_sq)
                IL_A_opt, _ = quad(integrand_opt, tl, T_H, args=(T_H,))
                for I_opt in opt_I_range:
                    Q_L_min = I_opt * np.sqrt(2 * (C_interp(T_H) - C_interp(tl)))
                    L_A_opt = IL_A_opt / I_opt
                    opt_L_A_cm.append(L_A_opt / 100.0)
                    opt_Q_L.append(Q_L_min)
```

```
                case_data['optimal'] = {'l_a': opt_L_A_cm, 'q_l': opt_Q_L}
            except (ValueError, IntegrationWarning): pass
            try:
                L_A_cond_cm = np.logspace(np.log10(1), np.log10(500), 100)
                K_integral, _ = quad(k_interp, tl, T_H)
                Q_L_cond = K_integral / (L_A_cond_cm * 100.0)
                case_data['conduction'] = {'l_a': L_A_cond_cm, 'q_l': Q_L_cond}
            except (ValueError, IntegrationWarning): pass
            storage[rrr][tl] = case_data
            progress_queue.put(1)
    progress_queue.put(storage)

# --- PLOTTING FUNCTIONS (MODIFIED) ---

def plot_from_precomputed_data(ax, ax2, data_slice, RRR, T_H, T_L, currents_kA_selected,
↪  material):
    colors = plt.cm.viridis(np.linspace(0, 1, len(currents_kA_selected)))
    target_temps = [item['temp'] for item in data_slice['isotherms']]
    isotherm_colors = plt.cm.magma(np.linspace(0.2, 0.9, len(target_temps)))
    color_map = {I_kA: colors[i] for i, I_kA in enumerate(currents_kA_selected)}
    current_handles, line_handles, isotherm_lines, isotherm_texts = [], [], [], []

    for curve_data in data_slice['current_curves']:
        I_kA = curve_data['I_kA']
        if I_kA in currents_kA_selected:
            handle = ax.scatter(curve_data['l_a'], curve_data['q_l'], s=15,
            ↪  color=color_map[I_kA], label=f'{I_kA} kA', alpha=0.8)
            current_handles.append(handle)

    for idx, isotherm_data in enumerate(data_slice['isotherms']):
        points = isotherm_data['points']
        if len(points) > 1:
            l_a_vals = [p['l_a'] for p in points]
            q_l_vals = [p['q_l'] for p in points]
            line, = ax.plot(l_a_vals, q_l_vals, color=isotherm_colors[idx],
            ↪  linestyle=':', lw=1.2)
            isotherm_lines.append(line)
            if l_a_vals:
                text_handle = ax.text(l_a_vals[-1] * 1.03, q_l_vals[-1],
                ↪  f"{isotherm_data['temp']} K",
                                      ha='left', va='center', fontsize=7,
                                      ↪  color=isotherm_colors[idx],
                                      bbox=dict(facecolor='white', alpha=0.5,
                                      ↪  edgecolor='none', boxstyle='round,pad=0.1'))
                isotherm_texts.append(text_handle)

    if data_slice.get('optimal'):
        handle, = ax.plot(data_slice['optimal']['l_a'], data_slice['optimal']['q_l'],
        ↪  color='black', label='Optimal Operation', lw=2)
        line_handles.append(handle)
    if data_slice.get('conduction'):
        handle, = ax.plot(data_slice['conduction']['l_a'],
        ↪  data_slice['conduction']['q_l'], color='grey', linestyle='--',
        ↪  label='Thermal Conduction Only (I=0)')
        line_handles.append(handle)

    ax.set_xscale('log')
```

```
ax.set_yscale('log')
ax.set_xlim(1, 500)
all_y_data = []
if data_slice.get('optimal'): all_y_data.extend(data_slice['optimal']['q_l'])
if data_slice.get('conduction'): all_y_data.extend(data_slice['conduction']['q_l'])
for curve in data_slice['current_curves']:
    if curve['I_kA'] in currents_kA_selected: all_y_data.extend(curve['q_l'])
if all_y_data and max(all_y_data) > 0: ax.set_ylim(bottom=1, top=max(all_y_data) *
↪  1.5)
else: ax.set_ylim(bottom=1, top=10000)

ax.set_ylabel(r'Cooling Load $\dot{Q}_L$ [W]', fontsize=16)
ax.set_xlabel(r'Length/Area $L/A$ [cm$^{-1}$]', fontsize=16)
# Update title to include material name
ax.set_title(f'Material: {material} | RRR = {RRR}\n$T_H = {T_H}$ K, $T_L =
↪  {T_L:.0f}$ K', fontsize=20)

ax.grid(True, which='both', linestyle='--', linewidth=0.5)

if line_handles:
    leg1 = ax.legend(handles=line_handles, loc='upper right', title="Condition",
    ↪  fontsize=14)
    ax.add_artist(leg1)
if current_handles:
    ax.legend(handles=current_handles, loc='lower left', title="Current (kA)",
    ↪  fontsize=14)

ax2.set_yscale('log')
ax2.set_ylabel("Length (m)", color='tab:green', fontsize=16)
ax2.tick_params(axis='y', labelcolor='tab:green', which='both', right=True,
↪  labelright=True)
ax2.yaxis.set_label_position("right")
# ax2.set_ylim(0.1, 10)

# Use density based on selected material
density = MATERIALS[material]["density"]
iso_weights_kg = [0.1, 0.5, 1.0, 2.0,3, 5.0,7,8,10,15,20]
x_range_cm = np.logspace(np.log10(ax.get_xlim()[0]), np.log10(ax.get_xlim()[1]),
↪  200)
x_range_si = x_range_cm * 100

iso_weight_lines = []
for weight in iso_weights_kg:
    length_m = np.sqrt((weight * x_range_si) / density)
    line, = ax2.plot(x_range_cm, length_m, color='tab:green', linestyle='-.',
    ↪  alpha=0.7)
    iso_weight_lines.append(line)
    label_x_pos = x_range_cm[len(x_range_cm) // 2]
    label_y_pos = length_m[len(length_m) // 2]
    p1 = ax2.transData.transform((x_range_cm[10], length_m[10]))
    p2 = ax2.transData.transform((x_range_cm[-10], length_m[-10]))
    angle = np.degrees(np.arctan2(p2[1] - p1[1], p2[0] - p1[0]))
    ax2.text(label_x_pos, label_y_pos, f'{weight} kg', color='darkgreen',
    ↪  ha='center', va='center',
             rotation=angle, bbox=dict(facecolor='white', alpha=0.8,
             ↪  edgecolor='none', pad=1))
```

```
    return iso_weight_lines, isotherm_lines, isotherm_texts

def launch_interactive_plots(all_results, T_L_values, T_H, RRR_list_selected,
↪  currents_kA_selected, material):
    figures_and_widgets = []
    for rrr in RRR_list_selected:
        fig, ax = plt.subplots(figsize=(10, 8.5))
        # Update window title to include material
        fig.canvas.manager.set_window_title(f'Plot for {material} RRR = {rrr}')
        fig.subplots_adjust(bottom=0.25, right=0.85)
        ax2 = ax.twinx()
        plot_state = {
            'fig': fig, 'ax': ax, 'ax2': ax2, 'rrr': rrr,
            'toggleable_elements': {'checkboxes': {}}
        }

        def make_update_func(state):
            def update(val):
                state['ax'].cla()
                state['ax2'].cla()
                idx = np.abs(np.array(T_L_values) - val).argmin()
                snapped_val = T_L_values[idx]

                iso_lines, isotherms, isotherm_texts = plot_from_precomputed_data(
                    state['ax'], state['ax2'], all_results[state['rrr']][snapped_val],
                    state['rrr'], T_H, snapped_val, currents_kA_selected, material
                )

                t_elems = state['toggleable_elements']
                t_elems['iso_weight_lines'] = iso_lines
                t_elems['iso_weight_axis'] = state['ax2']
                t_elems['isotherm_lines'] = isotherms
                t_elems['isotherm_texts'] = isotherm_texts

                right_grid_lines = []
                for y_tick_loc in state['ax2'].get_yticks(minor=True):
                    right_grid_lines.append(state['ax2'].axhline(y_tick_loc,
                    ↪  color='green', linestyle=':', linewidth=0.8, alpha=0.9,
                    ↪  zorder=-1))
                for y_tick_loc in state['ax2'].get_yticks():
                    right_grid_lines.append(state['ax2'].axhline(y_tick_loc,
                    ↪  color='green', linestyle=':', linewidth=1, alpha=1, zorder=-1))
                t_elems['right_grid_lines'] = right_grid_lines

                for key, checkbox in t_elems['checkboxes'].items():
                    is_visible = checkbox.get_status()[0]
                    if key == 'isotherm_lines':
                        for item in t_elems.get('isotherm_lines', []):
                        ↪  item.set_visible(is_visible)
                        for item in t_elems.get('isotherm_texts', []):
                        ↪  item.set_visible(is_visible)
                    elif key == 'iso_weight_axis':
                        t_elems.get(key).set_visible(is_visible)
                        for item in t_elems.get('iso_weight_lines', []):
                        ↪  item.set_visible(is_visible)
                    else:
                        for item in t_elems.get(key, []): item.set_visible(is_visible)
```

```
                state['fig'].canvas.draw_idle()
            return update

        def make_toggle_func(state, key):
            def toggle(label):
                is_visible =
                ↪  state['toggleable_elements']['checkboxes'][key].get_status()[0]
                if key == 'isotherm_lines':
                    for item in state['toggleable_elements'].get('isotherm_lines', []):
                    ↪  item.set_visible(is_visible)
                    for item in state['toggleable_elements'].get('isotherm_texts', []):
                    ↪  item.set_visible(is_visible)
                elif key == 'iso_weight_axis':
                    state['toggleable_elements'].get(key).set_visible(is_visible)
                    for item in state['toggleable_elements'].get('iso_weight_lines',
                    ↪  []): item.set_visible(is_visible)
                else:
                    for item in state['toggleable_elements'].get(key, []):
                    ↪  item.set_visible(is_visible)
                state['fig'].canvas.draw_idle()
            return toggle

        ax_slider = fig.add_axes([0.25, 0.1, 0.5, 0.04])
        step = T_L_values[1] - T_L_values[0] if len(T_L_values) > 1 else 1
        tl_slider = Slider(ax=ax_slider, label='Cold End Temperature $T_L$ [K]',
        ↪  valmin=min(T_L_values), valmax=max(T_L_values), valinit=min(T_L_values),
        ↪  valstep=step, color='#A9CCE3')
        update_function = make_update_func(plot_state)
        tl_slider.on_changed(update_function)

        checkbox_definitions = {
            'iso_weight_axis': ([0.02, 0.88, 0.15, 0.05], ' Iso-Weight'),
            'right_grid_lines': ([0.02, 0.83, 0.15, 0.05], ' Right Grid'),
            'isotherm_lines': ([0.02, 0.78, 0.15, 0.05], ' Isotherms'),
        }

        checks = []
        for key, (rect, label) in checkbox_definitions.items():
            ax_check = fig.add_axes(rect, frame_on=False)
            check = CheckButtons(ax_check, (label,), (True,))
            plot_state['toggleable_elements']['checkboxes'][key] = check
            check.on_clicked(make_toggle_func(plot_state, key))
            checks.append(check)

        update_function(min(T_L_values))
        figures_and_widgets.append({'fig': fig, 'slider': tl_slider, 'checks': checks})

    plt.show()


# --- GUI Application Class (MODIFIED) ---
class App:
    def __init__(self, master):
        self.master = master
        master.title("Current Lead Sizing Tool")
        self.frame = ttk.Frame(master, padding="15")
```

```
        self.frame.pack(fill="both", expand=True)

        # --- NEW: Material Selection Frame ---
        material_frame = ttk.LabelFrame(self.frame, text="Material", padding="10")
        material_frame.pack(fill="x", expand=True, pady=(0,10))
        self.material_var = tk.StringVar()
        self.material_combo = ttk.Combobox(material_frame,
        ↪  textvariable=self.material_var)
        self.material_combo['values'] = list(MATERIALS.keys())
        self.material_combo.set("Copper") # Default value
        self.material_combo.pack(fill="x", expand=True)

        params_frame = ttk.LabelFrame(self.frame, text="Parameters", padding="10")
        params_frame.pack(fill="x", expand=True)
        params_frame.columnconfigure(1, weight=1)
        ttk.Label(params_frame, text="Hot Temperature T_H (K):").grid(row=0, column=0,
        ↪  sticky="w", pady=5)
        self.th_entry = ttk.Entry(params_frame, width=30)
        self.th_entry.insert(0, "290")
        self.th_entry.grid(row=0, column=1, padx=5)
        ttk.Label(params_frame, text="Cold Temperatures T_L (K):").grid(row=1, column=0,
        ↪  sticky="w", pady=5)
        self.tl_entry = ttk.Entry(params_frame, width=30)
        self.tl_entry.insert(0, "25, 30, 35, 40, 45, 50")
        self.tl_entry.grid(row=1, column=1, padx=5)
        ttk.Label(params_frame, text="(comma-separated list)").grid(row=2, column=1,
        ↪  sticky="w", padx=5)
        try:
            image_path = resource_path("gui image v2.PNG")
            original_image = Image.open(image_path)
            resized_image = original_image.resize((80, 120), Image.Resampling.LANCZOS)
            self.photo_image = ImageTk.PhotoImage(resized_image)
            image_label = ttk.Label(params_frame, image=self.photo_image)
            image_label.grid(row=0, column=2, rowspan=3, sticky="e", padx=(20, 0))
        except Exception as e:
            print(f"Warning: Could not load image. {e}")
        rrr_frame = ttk.LabelFrame(self.frame, text="Enter RRR Values", padding="10")
        rrr_frame.pack(fill="x", expand=True, pady=10)
        self.rrr_entry = ttk.Entry(rrr_frame)
        self.rrr_entry.insert(0, "50, 100, 150, 300")
        self.rrr_entry.pack(fill="x", expand=True)
        ttk.Label(rrr_frame, text="(comma-separated list)").pack(anchor="w")
        current_frame = ttk.LabelFrame(self.frame, text="Enter Currents (kA)",
        ↪  padding="10")
        current_frame.pack(fill="x", expand=True)
        self.current_entry = ttk.Entry(current_frame)
        self.current_entry.insert(0, "0.5, 1.0, 2.0, 4.0")
        self.current_entry.pack(fill="x", expand=True)
        ttk.Label(current_frame, text="(comma-separated list)").pack(anchor="w")
        control_frame = ttk.Frame(self.frame, padding="10")
        control_frame.pack(fill="x", expand=True)
        self.start_button = ttk.Button(control_frame, text="Start Calculation",
        ↪  command=self.start_calculation)
        self.start_button.pack(pady=10)
        self.progress_bar = ttk.Progressbar(control_frame, orient="horizontal",
        ↪  length=300, mode="determinate")
        self.progress_bar.pack(pady=5)
```

```
        self.status_label = ttk.Label(control_frame, text="Status: Ready.")
        self.status_label.pack(pady=5)
        self.calculation_thread = None
        self.results_queue = queue.Queue()

    def start_calculation(self):
        try:
            # Get selected material from the new combobox
            self.selected_material = self.material_var.get()
            if self.selected_material not in MATERIALS:
                raise ValueError("Please select a valid material.")

            self.T_H_main = int(self.th_entry.get())
            tl_string = self.tl_entry.get()
            if not tl_string: raise ValueError("T_L list is empty.")
            self.T_L_values = sorted([int(x.strip()) for x in tl_string.split(',')])
            if not self.T_L_values: raise ValueError("T_L list is empty after
            ↪  parsing.")
            rrr_string = self.rrr_entry.get()
            if not rrr_string: raise ValueError("RRR list is empty.")
            self.selected_rrrs = sorted([int(x.strip()) for x in
            ↪  rrr_string.split(',')])
            if not self.selected_rrrs: raise ValueError("RRR list is empty after
            ↪  parsing.")
            current_string = self.current_entry.get()
            if not current_string: raise ValueError("Currents list is empty.")
            self.selected_currents = sorted([float(x.strip()) for x in
            ↪  current_string.split(',')])
            if not self.selected_currents: raise ValueError("Currents list is empty
            ↪  after parsing.")
        except ValueError as e:
            messagebox.showerror("Invalid Input", f"Please check your input.\nError:
            ↪  {e}")
            return

        # Print properties for the selected material
        for rrr in self.selected_rrrs:
            print_material_properties(rrr, self.selected_material)

        self.start_button.config(state="disabled")
        total_steps = len(self.selected_rrrs) * len(self.T_L_values)
        self.progress_bar["maximum"] = total_steps
        self.progress_bar["value"] = 0
        self.status_label.config(text=f"Status: Calculating for
        ↪  {self.selected_material}...")
        self.master.update_idletasks()
        if self.selected_material == "Copper":
            target_temps = np.arange(1350, 300, -100)
        if self.selected_material == "Aluminum":
            target_temps = np.arange(675, 300, -100)
        self.calculation_thread = threading.Thread(
            target=precompute_all_data,
            # Pass selected material to the calculation thread
            args=(self.selected_rrrs, self.T_L_values, self.T_H_main,
            ↪  self.selected_currents, target_temps, self.selected_material,
            ↪  self.results_queue)
        )
```

```
        self.calculation_thread.start()
        self.master.after(100, self.check_queue)

    def check_queue(self):
        try:
            message = self.results_queue.get_nowait()
            if isinstance(message, int):
                self.progress_bar.step(message)
            elif isinstance(message, dict):
                self.progress_bar["value"] = self.progress_bar["maximum"]
                self.status_label.config(text="Status: Complete! Launching plot(s)...")
                self.master.update_idletasks()
                time.sleep(1)
                self.master.destroy()
                # Pass selected material to the plotting function
                launch_interactive_plots(message, self.T_L_values, self.T_H_main,
                ↪  self.selected_rrrs, self.selected_currents, self.selected_material)
                return
        except queue.Empty: pass
        if self.calculation_thread and self.calculation_thread.is_alive():
            self.master.after(100, self.check_queue)
        elif not self.results_queue.empty():
            self.master.after(100, self.check_queue)


if __name__ == '__main__':
    root = tk.Tk()
    app = App(root)
    root.mainloop()
```

C.2. 1D heat sink python code

```
#heat sink model v10 optimization_cleanup

#Heat sink model v10 optimization

# Heat Sink model v9 (including poles)
# 19/03/2025

from scipy.interpolate import RegularGridInterpolator
import numpy as np
import matplotlib.pyplot as plt
import tkinter as tk
from PIL import Image, ImageTk
from tkinter import ttk
from tkinter import messagebox
from tkinter import font
from CoolProp import AbstractState
from CoolProp.CoolProp import PhaseSI, PropsSI, get_global_param_string
import CoolProp.CoolProp as CoolProp
from CoolProp.HumidAirProp import HAPropsSI
from time import *
from math import *
```

```
# Initialize m_dot with the script value
lh2_mass_at_90_percent = 850  # kg also the maximum fill lvl
# m_dot_default = (lh2_mass_at_90_percent * 6 / 9) / (15 * 60)  # m_dot_kg_hr/(3600)
↪  #kg/s
m_dot_default = 14.4  # kg/hr
m_dot = m_dot_default / 3600  # kg/s
original_time_to_fill = lh2_mass_at_90_percent / m_dot  # sec
# print("Original time to fill the tank to 90 % at a mass flow rate of "+str(m_dot)+"
↪  kg/s is: ",original_time_to_fill/60, " mins.")

tank_mass = 2000  # kg (Aluminium or steel)
Cp_AL = 903  # J/kg*K
Cp_SS304 = 502  # J/kg*K
Cp_SS316 = 490  # J/kg*K

latent_heat_lh2 = 461 * 10 ** 3  # J/kg


def friction_coefficient_6mm(reynolds):
    if reynolds <= 32000:
        return 0.1
    elif 32000 < reynolds <= 125000:
        return 0.00014 * reynolds ** 0.628
    else:
        return 0.22


def friction_coefficient_12mm(reynolds):
    if reynolds <= 64000:
        return 0.08
    elif 64000 < reynolds <= 205000:
        return 0.000145 * reynolds ** 0.57
    else:
        return 0.15


def friction_coefficient_25mm(reynolds):
    if reynolds <= 95000:
        return 0.065
    elif 95000 < reynolds <= 360000:
        return 0.000325 * reynolds ** 0.465
    else:
        return 0.13


def friction_coefficient_100mm(reynolds):
    if reynolds <= 125000:
        return 0.06
    elif 125000 < reynolds <= 500000:
        return 0.00031 * reynolds ** 0.445
    else:
        return 0.115


def friction_coefficient_150mm(reynolds):
    if reynolds <= 125000:
```

```
        return 0.054
    elif 125000 < reynolds <= 500000:
        return 0.000255 * reynolds ** 0.454
    else:
        return 0.099


# Mapping diameters to their corresponding functions
diameter_functions = {
    6: friction_coefficient_6mm,
    12: friction_coefficient_12mm,
    25: friction_coefficient_25mm,
    100: friction_coefficient_100mm,
    150: friction_coefficient_150mm
}


def calculate_friction_coefficient_with_warning(diameter, reynolds):
    sorted_diameters = sorted(diameter_functions.keys())
    if diameter in diameter_functions:
        # Use the corresponding function for the exact diameter
        return diameter_functions[diameter](reynolds)
    elif diameter < min(sorted_diameters):
        # If diameter is lower than the lowest possible, use the lowest possible and
        ↪  give a warning
        print(
            f"Warning: Diameter {diameter}mm is lower than the lowest predefined
            ↪  diameter. Using {min(sorted_diameters)}mm instead.")
        return diameter_functions[min(sorted_diameters)](reynolds)
    elif diameter > max(sorted_diameters):
        # If diameter is higher than the highest possible, use the highest possible and
        ↪  give a warning
        print(
            f"Warning: Diameter {diameter}mm is higher than the highest predefined
            ↪  diameter. Using {max(sorted_diameters)}mm instead.")
        return diameter_functions[max(sorted_diameters)](reynolds)
    else:
        # Interpolate between the closest diameters
        lower_diameter = max(filter(lambda x: x < diameter, sorted_diameters))
        upper_diameter = min(filter(lambda x: x > diameter, sorted_diameters))

        lower_friction = diameter_functions[lower_diameter](reynolds)
        upper_friction = diameter_functions[upper_diameter](reynolds)
        interpolated_friction = ((diameter - lower_diameter) / (upper_diameter -
        ↪  lower_diameter)) * upper_friction + ((upper_diameter - diameter) /
        ↪  (upper_diameter - lower_diameter)) * lower_friction
        return interpolated_friction


# Create a function to perform calculations and plot the results

def calculate_pressure_drop():
    global m_dot
    H2 = "hydrogen"
    p_bar = float(entry_inlet_pressure.get())
    p_inlet = p_bar * (10 ** 5)
    # quality = 0  # 0 For complete liquid & 1 for complete gas
```

```
    inlet_temp = float(entry_inlet_temp.get())  # [K]

    # pipe_roughness = 3 * 10 ** (-5)  # m
    pipe_roughness = 12 * 10 ** (-5)  # m
    m_dot_user_min = entry_m_dot.get()
    m_dot_user_max = entry_m_dot_max.get()

    no_90_bends = entry_90_elbow.get()
    no_T_straight = entry_T_straight.get()
    no_T_branch = entry_T_branch.get()
    no_180_bends = entry_180_bend.get()

    Q_in_total = float(entry_external_heatload.get())

    selected_pipe_type_option = pipe_type_combobox.get()
    selected_fluid_type_option = fluid_type_combobox.get()

    selected_radius_type_option = radius_type_combobox.get()

    selected_radius_T_type_option = radius_T_type_combobox.get()
    selected_radius_T_branch_type_option = radius_T_branch_type_combobox.get()
    selected_radius_180_bend_type_option = radius_180_bend_type_combobox.get()
    # inlet_temp = RP.REFPROPdll(selected_fluid_type_option, "PQ", "T", MASS_SI, 0, 0,
    ↪  p_inlet, quality, [1.0]).Output[
    #     0]  # [K]
    # print("Inlet temperature: ", inlet_temp)
    selected_element_option = element_combobox.get()

    if selected_element_option == "Pipe/Tube":
        no_channels = float(entry_no_channels.get())
        if m_dot_user: # Not in use currently
            try:
                m_dot_custom = float(m_dot_user_min) # Using m_dot_user_min
                if m_dot_custom >= 0:
                    m_dot = (m_dot_custom) / no_channels # IN kg/s per channel
                else:
                    messagebox.showerror("Input Error", "Mass flow rate must be a
                    ↪  non-negative value.")
                    return
            except ValueError:
                messagebox.showerror("Input Error", "Invalid input for mass flow rate.
                ↪  Please enter a number.")
                return

        pipe_inner_diameter_min = float(entry_min_inner_diameter.get()) # [mm]
        pipe_inner_diameter_max = float(entry_max_inner_diameter.get()) # [mm]
        pipe_inner_diameter = np.arange(pipe_inner_diameter_min,
        ↪  pipe_inner_diameter_max + 1, 1)  # [mm]

        D_elec = 0  # 20  # mm
        D_test = (pipe_inner_diameter_min ** 2 - 2 * D_elec ** 2) / (2 * D_elec +
        ↪  pipe_inner_diameter_min) # mm
        print("For minimal diameter given, this is the hydraulic diameter", D_test,
        ↪  "mm")
        pipe_length_str = entry_pipe_length.get()
        pipe_length = [float(val.strip()) for val in pipe_length_str.split(",")]  #
        ↪  Convert comma-separated values to a list
```

```
delta_p_lst = [[] for _ in range(len(pipe_length))]

density_1 = PropsSI("D", "T", inlet_temp, "P", p_inlet,
↪  selected_fluid_type_option)#H2)  # [kg/m^3]
specific_volume_1 = 1/density_1

viscosity_1 = PropsSI("V", "T", inlet_temp, "P", p_inlet,
↪  selected_fluid_type_option)#H2)  # [Pa*s]
print("Viscosity: ", viscosity_1)

E_in = 0  # 1 # W/m

Q_in_per_channel = Q_in_total / no_channels #W

total_delta_p_for_each_diameter_and_length = []  # List to store total pressure
↪  drops for each diameter and pipe length
total_velocity_for_each_diameter_and_length = []  # List to store total
↪  velocities for each diameter and pipe length
total_reynolds_for_each_diameter_and_length = []  # List to store total
↪  reynolds for each diameter and pipe length
total_specific_volume_for_each_diameter_and_length = []  # List to store total
↪  specific volume for each diameter and pipe length
num_segments = 100  #

Q_in_per_channel_segment = Q_in_per_channel/num_segments #W

for i in range(len(pipe_length)):

    segment_length = pipe_length[i] / num_segments
    delta_p_for_current_length = []  # List to store delta_p for each diameter
    ↪  for the current pipe length
    velocity_for_current_length = []
    reynolds_for_current_length = []
    specific_volume_for_current_length = []
    for D in pipe_inner_diameter:
        p_segment = p_inlet  # Reset pressure for each diameter
        T_segment = inlet_temp  # Reset temperature for each diameter
        total_delta_p = 0  # Reset total pressure drop for each diameter

        for segment in range(num_segments):
            # Recalculate fluid properties at the beginning of each segment
            density = PropsSI("D", "T", T_segment, "P", p_segment,
            ↪  selected_fluid_type_option)  # [kg/m^3]
            specific_volume = 1/density  #
            ↪  RP.REFPROPdll(selected_fluid_type_option, "PT", "V", MASS_SI, 0,
            ↪  0, p_segment, T_segment, [1.0]).Output[0]
            viscosity = PropsSI("V", "T", T_segment, "P", p_segment,
            ↪  selected_fluid_type_option)  # [Pa*s]
            print("Viscosity: ", viscosity)
            prandtl = PropsSI("PRANDTL", "T", T_segment, "P", p_segment,
            ↪  selected_fluid_type_option)
            fluid_conductivity = PropsSI("L", "T", T_segment, "P", p_segment,
            ↪  selected_fluid_type_option)  # [W/m.K]
            specific_heat = PropsSI("C", "T", T_segment, "P", p_segment,
            ↪  selected_fluid_type_option)  # J/kgK
```

```
enthalpy_segment_inlet = PropsSI("H", "T", T_segment, "P",
↪  p_segment, selected_fluid_type_option) #J/kg
print("Enthalpy of fluid in segment: ", enthalpy_segment_inlet)
enthalpy_segment_outlet = (Q_in_per_channel_segment/m_dot) +
↪  enthalpy_segment_inlet #  J/kg

hydraulic_diameter = (D ** 2 - 2 * D_elec ** 2) / (2 * D_elec + D)
↪  # mm

flow_area = np.pi * ((0.001*hydraulic_diameter/2) ** 2) #m2
volumetric_flow_rate = (specific_volume * m_dot)
# print("m_dot = ",m_dot)
velocity = volumetric_flow_rate / flow_area  # m/s
# print("Velocity: ", velocity)
reynolds_no = (4 * m_dot) / (np.pi * viscosity *
↪  (hydraulic_diameter*0.001))

if selected_pipe_type_option == "Rigid":
    # friction_factor = 1 / 1.8 ** 2 / ((np.log10(
    #     (6.9 / reynolds_no) + (pipe_roughness / (3.7 * (D * 10 **
    ↪  (-3)))) ** 1.11)) ** 2)  # liquid
    friction_factor_haaland = 1 / 1.8 ** 2 / ((np.log10(
        (6.9 / reynolds_no) + (pipe_roughness / (3.7 * (D*0.001)))
        ↪  ** 1.11)) ** 2)  # liquid
    while True:
        friction_factor_darcy_init = (
        ↪  1/(-2*log10((pipe_roughness/(3.7*(D*0.001)))+(2.51/(reynolds_no*np.sq
        ↪  )**2
        friction_factor_darcy_new = (
        ↪  1/(-2*log10((pipe_roughness/(3.7*(D*0.001)))+(2.51/(reynolds_no*np.sq
        ↪  )**2
        print("iterative friction factor = ",
        ↪  friction_factor_darcy_new, "For pipe diameter of ", D,
        ↪  "mm")
        if abs(friction_factor_darcy_new -
        ↪  friction_factor_darcy_init) < 0.000001:
            friction_factor = friction_factor_darcy_new
            break
        else:
            friction_factor_haaland = friction_factor_darcy_new

    print("friction factor = ", friction_factor, "For pipe diameter
    ↪  of ", D, "mm")
    if selected_radius_type_option == "Standard radius (R/D = 1)":
        K1 = 800
        K_infinity = 0.091
        K_d = 10.6  # mm^0.3
        K = K1 / reynolds_no + K_infinity * (1 + (K_d / (
                (hydraulic_diameter) ** 0.3)))  # hydraulic
                ↪  diameter is in mm as it should be for this
                ↪  formula
        dp_bend = 0.5 * K * density * (velocity ** 2) *
        ↪  float(no_90_bends)  # in Pa
        # print("pressure drop due to bend: ", dp_bend)
    if selected_radius_type_option == "Standard radius (R/D = 1.5)":
    ↪  # Interpolated !
        K1 = 800
```

```
            K_infinity = (0.091 + 0.056) / 2
            K_d = (10.6 + 10.3) / 2  # mm^0.3
            K = K1 / reynolds_no + K_infinity * (1 + (K_d / (
                    (hydraulic_diameter) ** 0.3)))  # hydraulic
                    ↪ diameter is in mm as it should be for this
                    ↪ formula
            dp_bend = 0.5 * K * density * (velocity ** 2) *
            ↪ float(no_90_bends)  # in Pa
            # print("pressure drop due to bend: ", dp_bend)
        if selected_radius_type_option == "Long radius (R/D = 2)":
            K1 = 800
            K_infinity = 0.056
            K_d = 10.3  # mm^0.3
            K = K1 / reynolds_no + K_infinity * (1 + (K_d / (
                    (hydraulic_diameter) ** 0.3)))  # hydraulic
                    ↪ diameter is in mm as it should be for this
                    ↪ formula
            dp_bend = 0.5 * K * density * (velocity ** 2) *
            ↪ float(no_90_bends)  # in Pa
            # print("pressure drop due to bend: ", dp_bend)

        if selected_radius_T_type_option == "Flanged (R/D = 1)":
            K1 = 150
            K_infinity = 0.05
            K_d = 10.6
            K = K1 / reynolds_no + K_infinity * (1 + (K_d / (
                    (hydraulic_diameter) ** 0.3)))  # hydraulic
                    ↪ diameter is in mm as it should be for this
                    ↪ formula
            dp_bend_T = 0.5 * K * density * (velocity ** 2) *
            ↪ float(no_T_straight)  # in Pa
            # print("pressure drop due to bend: ", dp_bend_T)

        if selected_radius_T_type_option == "Stub-in branch":
            messagebox.showerror("Error",
                                 "Stub-in branch option is not
                                 ↪ currently supported \n for T -
                                 ↪ straight! .")
            return

        if selected_radius_T_branch_type_option == "Standard radius
        ↪ (R/D = 1.5)":
            K1 = 800
            K_infinity = 0.14
            K_d = 10.6
            K = K1 / reynolds_no + K_infinity * (1 + (K_d /
            ↪ ((hydraulic_diameter) ** 0.3)))
            dp_bend_T_branch = 0.5 * K * density * (velocity ** 2) *
            ↪ float(no_T_branch)  # in Pa
        if selected_radius_T_branch_type_option == "Stub-in branch":
            K1 = 1000
            K_infinity = 0.34
            K_d = 10.6
            K = K1 / reynolds_no + K_infinity * (1 + (K_d /
            ↪ ((hydraulic_diameter) ** 0.3)))
            dp_bend_T_branch = 0.5 * K * density * (velocity ** 2) *
            ↪ float(no_T_branch)
```

```
                if selected_radius_180_bend_type_option == "Flanged (R/D = 1)":
                    K1 = 1000
                    K_infinity = 0.12
                    K_d = 10.6
                    K = K1 / reynolds_no + K_infinity * (1 + (K_d /
                    ↪  ((hydraulic_diameter) ** 0.3)))
                    dp_bend_180 = 0.5 * K * density * (velocity ** 2) *
                    ↪  float(no_180_bends)
                if selected_radius_180_bend_type_option == "All types (R/D =
                ↪  1.5)":
                    K1 = 100
                    K_infinity = 0.1
                    K_d = 10.6
                    K = K1 / reynolds_no + K_infinity * (1 + (K_d /
                    ↪  ((hydraulic_diameter) ** 0.3)))
                    dp_bend_180 = 0.5 * K * density * (velocity ** 2) *
                    ↪  float(no_180_bends)

            if selected_pipe_type_option == "Flexible":
                friction_factor = calculate_friction_coefficient_with_warning(D,
                ↪  reynolds_no)
                if D == 25 and segment == 1:
                    print("friction factor = ", friction_factor) #Kunal Test
                dp_bend, dp_bend_T, dp_bend_T_branch, dp_bend_180 = 0,0,0,0
                ↪  #KUNAL needed definition in case the option flexible was
                ↪  selected

            # Calculate pressure drop for the segment
            delta_p_segment = ((friction_factor * (segment_length /
            ↪  (hydraulic_diameter*0.001)) * (
                    density * (
                        velocity ** 2) / 2)) ) / 100  # in MilliBar

            # Update total pressure drop for the diameter
            total_delta_p += delta_p_segment

            if total_delta_p > p_inlet / 100:
                print(
                    f"Pressure drop for pipe length {pipe_length[i]}m and
                    ↪  diameter {D} mm is greater than inlet pressure.")
                return
            # Update pressure and temperature for the next segment
            p_segment -= delta_p_segment * 100  # Convert from MilliBar to
            ↪  Pascal for next iteration
            T_segment = PropsSI("T", "H", enthalpy_segment_outlet, "P",
            ↪  p_segment, selected_fluid_type_option)#E_in * segment_length /
            ↪  (m_dot * specific_heat)

        total_delta_p += (dp_bend + dp_bend_T + dp_bend_T_branch +
        ↪  dp_bend_180)/100
        delta_p_for_current_length.append(total_delta_p)
        velocity_for_current_length.append(velocity)
        reynolds_for_current_length.append(reynolds_no)
        specific_volume_for_current_length.append(specific_volume)
        print("Length of velocity list", len(velocity_for_current_length))
```

```
        ↪ total_delta_p_for_each_diameter_and_length.append(delta_p_for_current_length)

        ↪ total_velocity_for_each_diameter_and_length.append(velocity_for_current_length)

        ↪ total_reynolds_for_each_diameter_and_length.append(reynolds_for_current_length)

        ↪ total_specific_volume_for_each_diameter_and_length.append(specific_volume_for_current

plt.figure(figsize=(10, 6))
for i, delta_p_for_current_length in
↪  enumerate(total_delta_p_for_each_diameter_and_length):
    plt.plot(pipe_inner_diameter, delta_p_for_current_length,
    ↪  label=f'{pipe_length[i]} m')

plt.xlabel("Pipe ID [mm]", fontsize=18)
plt.ylabel("Pressure drop [millibar]", fontsize=18)
plt.xticks(size=17)
plt.yticks(size=17)
plt.legend(loc="upper right", fontsize=16)
plt.title("Pressure drop vs Pipe ID ", fontsize=20)
plt.grid()
# plt.show()

plt.figure(figsize=(10, 6))
for i, velocity_for_current_length in
↪  enumerate(total_velocity_for_each_diameter_and_length):
    plt.plot(pipe_inner_diameter, velocity_for_current_length,
    ↪  label=f'{pipe_length[i]} m')

plt.xlabel("Pipe ID [mm]", fontsize=18)
plt.ylabel("Velocity [m/s]", fontsize=18)
plt.xticks(size=17)
plt.yticks(size=17)
plt.legend(loc="upper right", fontsize=16)
plt.title("Velocity vs Pipe ID ", fontsize=20)
plt.grid()

plt.figure(figsize=(10, 6))
for i, reynolds_for_current_length in
↪  enumerate(total_reynolds_for_each_diameter_and_length):
    plt.plot(pipe_inner_diameter, reynolds_for_current_length,
    ↪  label=f'{pipe_length[i]} m')

plt.xlabel("Pipe ID [mm]", fontsize=18)
plt.ylabel("Reynolds number", fontsize=18)
plt.xticks(size=17)
plt.yticks(size=17)
plt.legend(loc="upper right", fontsize=16)
plt.title("Reynolds number vs Pipe ID ", fontsize=20)
plt.grid()

plt.figure(figsize=(10, 6))
for i, specific_volume_for_current_length in
↪  enumerate(total_specific_volume_for_each_diameter_and_length):
    plt.plot(pipe_inner_diameter, specific_volume_for_current_length,
    ↪  label=f'{pipe_length[i]} m')
```

```
        plt.xlabel("Pipe ID [mm]", fontsize=18)
        plt.ylabel("Specific volume [m^3/kg]", fontsize=18)
        plt.xticks(size=17)
        plt.yticks(size=17)
        plt.legend(loc="upper right", fontsize=16)
        plt.title("Specific volume vs Pipe ID ", fontsize=20)
        plt.grid()

        plt.show()

    if selected_element_option == "Heat Sink (Trapezoidal Channels)":  # No
    ↪  discritization in length, so no heat absorbed and no temperature increase
        #Define geometry
        no_of_poles = int(entry_no_poles.get())
        diameter_of_poles = float(entry_heat_sink_poles_OD.get())/1000 #m
        c_spacing = float(entry_c_space.get())/1000 #spacing between 2 adjacent poles
        ↪  [m]
        h_spacing = float(entry_h_space.get())/1000 #edge clearance [m]

        if no_of_poles == 2:
            hex_diameter = ((c_spacing/2 + diameter_of_poles + h_spacing)*2) #m
        if no_of_poles > 2:
            sector_angle = 360/no_of_poles #in degs
            sector_acute_angle = (180-sector_angle)/2 #in degs
            hypotenus = (c_spacing/2 +
            ↪  diameter_of_poles/2)/np.cos(np.deg2rad(sector_acute_angle)) # [m]
            hex_diameter = ((hypotenus + diameter_of_poles/2 + h_spacing)*2) # [m]
            print("The hex diameter v2 is: ",hex_diameter)



        thickness_sink = 2/1000 #m : Heat sink thickness

        fin_thickness = float(entry_fin_thickness.get())/1000 # m
        # max_fin_thickness = float(entry_max_fin_thickness.get())/1000 #m : Maximum
        ↪  fin thickness
        num_fins = int(entry_no_fins.get()) # USER CHOICE
        max_num_fins = int(entry_max_no_fins.get()) # USER CHOICE

        fin_height = float(entry_fin_height.get())/1000 #m : Height of fin

        selected_material_option = materials_combobox.get()
        if selected_material_option == "Aluminium":
            rho_material = 2700 #kg/m3
        if selected_material_option == "Copper":
            rho_material = 8960 #kg/m3

        if fin_thickness < 0:
            print("Error: Fin thickness is negative. Please check the input values.")
            exit()
        else:
            # Print the calculated number of fins
            print(f"Number of fins: {num_fins}")
            print(f"Fin thickness: {fin_thickness:.4f} m")
```

```
no_of_fins_range = np.arange(num_fins, max_num_fins + 1, 1)  #
# fins_thickness_range = np.arange(fin_thickness, max_fin_thickness + 0.1/1000,
↪  0.1/1000)  # [m]
dm = 0.0001 #kg/s
mass_flow_rate_range =
↪  np.arange(float(m_dot_user_min),float(m_dot_user_max)+dm,dm)

heat_sink_length = float(entry_heat_sink_length.get())/1000  # [m]

pipe_length = heat_sink_length # [m]

# delta_p_lst = [[] for _ in range(len(pipe_length))]

density_1 = PropsSI("D", "T", inlet_temp, "P", p_inlet,
↪  selected_fluid_type_option)#H2)  # [kg/m^3]
specific_volume_1 = 1/density_1

viscosity_1 = PropsSI("V", "T", inlet_temp, "P", p_inlet,
↪  selected_fluid_type_option)#H2)  # [Pa*s]
# print("Viscosity: ", viscosity_1)

E_in = 0  # 1 # W/m

total_delta_p_for_each_no_fins_and_spacing = []  # List
total_velocity_for_each_no_fins_and_spacing = []  # List
total_reynolds_for_each_no_fins_and_spacing = []  # List
total_specific_volume_for_each_no_fins_and_spacing = []  # List

num_segments = 1  # Always keep 1 (Not discritized in length)

# for i in range(len(pipe_length)):

segment_length = pipe_length/num_segments #m
delta_p_for_current_spacing = []  # List to store delta_p for each diameter for
↪  the current pipe length
velocity_for_current_spacing = []
reynolds_for_current_spacing = []
specific_volume_for_current_spacing = []
hydraulic_diameter_lst = []

dp_for_fin_space_lst = []

fin_thick = fin_thickness #m #FIN_THICK[i, j]

# Create meshgrid for all combinations
NUM_FIN, MASSFLOW = np.meshgrid(no_of_fins_range, mass_flow_rate_range)

# Initialize result arrays
HYD_DIAM = np.zeros_like(NUM_FIN, dtype=float)
DELTA_P = np.zeros_like(NUM_FIN, dtype=float)
DELTA_T = np.zeros_like(NUM_FIN, dtype=float)
DELTA_T_WALL = np.zeros_like(NUM_FIN, dtype=float)
VELOCITY = np.zeros_like(NUM_FIN, dtype=float)
REYNOLDS = np.zeros_like(NUM_FIN, dtype=float)
MASS_HEAT_SINK = np.zeros_like(NUM_FIN, dtype=float)
FLUID_DENSITY = np.zeros_like(NUM_FIN, dtype=float)
HEAT_TRANSFER_COEF = np.zeros_like(NUM_FIN, dtype=float)
```

```
# Compute values for all meshgrid points
for i in range(NUM_FIN.shape[0]):  # Iterate over rows
    for j in range(NUM_FIN.shape[1]):  # Iterate over columns
        num_fin = NUM_FIN[i, j]
        mass_heatsink = (num_fin*((fin_thick)*heat_sink_length*(fin_height)) +
        ↪  (((np.pi*(hex_diameter)**2 / 4) -
        ↪  (no_of_poles*np.pi*(diameter_of_poles)**2 /
        ↪  4)))*heat_sink_length)*rho_material
        MASS_HEAT_SINK[i, j] = mass_heatsink  # kg
        no_channels = num_fin - 1
        Q_in_per_channel = Q_in_total/no_channels
        Q_in_per_channel_segment = Q_in_per_channel/num_segments
        m_dot = MASSFLOW[i,j]/no_channels

        theta = 360/num_fin # sector angle in degrees

        sector_vol = (np.pi*((hex_diameter+2*fin_height)/2)**2 -
        ↪  np.pi*((hex_diameter)/2)**2)*(segment_length)*(theta/360) -
        ↪  2*(fin_height*(fin_thick/2)*(segment_length))#m3
        wetted_surface =
        ↪  ((theta/360)*2*np.pi*(hex_diameter/2))*(segment_length)+(((theta/360)*2*np.pi*((h
        ↪  #m2
        # hydraulic_diameter =
        ↪  (2*((fin_space_base+fin_space_top)/2)*fin_height)/(((fin_space_base+fin_space_top
        ↪  #4*sector_vol/wetted_surface # [mm]     #(y_value*(fin_space_base +
        ↪  y_value*(1/np.tan(alpha_angle))))/(fin_space_base+((2*y_value)/np.sin(alpha_angle
        hydraulic_diameter = 4*sector_vol/wetted_surface # [m]
        ↪  #(y_value*(fin_space_base +
        ↪  y_value*(1/np.tan(alpha_angle))))/(fin_space_base+((2*y_value)/np.sin(alpha_angle

        HYD_DIAM[i, j] = hydraulic_diameter  # Store in array

        # Fluid properties (example using CoolProp)
        density = PropsSI("D", "T", inlet_temp, "P", p_inlet,
        ↪  selected_fluid_type_option)
        # print("Density: ", density, "kg/m3")
        viscosity = PropsSI("V", "T", inlet_temp, "P", p_inlet,
        ↪  selected_fluid_type_option)
        enthalpy_inlet = PropsSI("H", "T", inlet_temp, "P", p_inlet,
        ↪  selected_fluid_type_option) #J/kg
        cp_inlet = PropsSI("C", "T", inlet_temp, "P", p_inlet,
        ↪  selected_fluid_type_option) #J/kgK

        # Flow calculations
        flow_area_per_channel = (np.pi*((hex_diameter+2*fin_height)/2)**2 -
        ↪  np.pi*((hex_diameter)/2)**2)*(theta/360) -
        ↪  2*(fin_height*(fin_thick/2))#m2
        # print("Flow area per channel: ",flow_area_per_channel)
        velocity = (m_dot / density) / flow_area_per_channel  # m/s
        VELOCITY[i, j] = velocity  # Store    <------------------------ PROBLEM
        ↪  PLACE

        reynolds_no = (4 * m_dot) / (np.pi * viscosity * (hydraulic_diameter))
        REYNOLDS[i, j] = reynolds_no  # Store

        # Friction factor (Haaland Equation)
```

```
            friction_factor_haaland = 1 / (1.8 * np.log10((6.9 / reynolds_no) +
            ↪  (pipe_roughness / (3.7 * (hydraulic_diameter))))) ** 2
            while True:
                friction_factor_darcy_init = (
                ↪  1/(-2*log10((pipe_roughness/(3.7*(hydraulic_diameter)))+(2.51/(reynolds_no*np
                ↪  )**2
                friction_factor_darcy_new = (
                ↪  1/(-2*log10((pipe_roughness/(3.7*(hydraulic_diameter)))+(2.51/(reynolds_no*np
                ↪  )**2
                # print("iterative friction factor = ", friction_factor_darcy_new,
                ↪  "For pipe diameter of ", D, "mm")
                if abs(friction_factor_darcy_new - friction_factor_darcy_init) <
                ↪  0.000001:
                    friction_factor = friction_factor_darcy_new
                    break
                else:
                    friction_factor_haaland = friction_factor_darcy_new
            # Pressure drop
            delta_p_segment = ((friction_factor * (segment_length /
            ↪  (hydraulic_diameter )) * (density * (velocity ** 2) / 2))) # / 100
            ↪  # in mBar
            # print("Delta P :",delta_p_segment)
            enthalpy_outlet = (Q_in_per_channel_segment/m_dot) + enthalpy_inlet #
            ↪  J/kg
            delta_T_due_tofriction =
            ↪  segment_length*(friction_factor*((velocity**2)/(2*hydraulic_diameter*cp_inlet)))
            Temp_fluid_outlet = PropsSI("T", "H", enthalpy_outlet, "P", (p_inlet -
            ↪  delta_p_segment), selected_fluid_type_option)+
            ↪  delta_T_due_tofriction
            mean_fluid_temp = (inlet_temp + Temp_fluid_outlet)/2

            prandtl = PropsSI("PRANDTL", "T", mean_fluid_temp, "P",
            ↪  (p_inlet+((p_inlet - delta_p_segment)))/2,
            ↪  selected_fluid_type_option)
            fluid_conductivity = PropsSI("L", "T", mean_fluid_temp, "P",
            ↪  (p_inlet+((p_inlet - delta_p_segment)))/2,
            ↪  selected_fluid_type_option)  # [W/m.K]
            nusselt_no =
            ↪  ((friction_factor/8)*(reynolds_no-1000)*prandtl)/(1+12.7*(friction_factor/8)**0.5
            ↪  * (prandtl**(2/3) - 1))#0.023 * prandtl**(1/3) * reynolds_no
            h_c_in = (fluid_conductivity*nusselt_no)/(hydraulic_diameter) #W/m2K
            print("H_c_in : ",h_c_in)
            HEAT_TRANSFER_COEF[i, j] = h_c_in
            heta_transfer_area = 2*np.pi*(hydraulic_diameter/2)*segment_length #m2

            mean_temp_wall = Q_in_per_channel_segment/(h_c_in*heta_transfer_area) +
            ↪  mean_fluid_temp

            DELTA_P[i, j] = delta_p_segment  # Store in pascals
            DELTA_T[i, j] = Temp_fluid_outlet # Store
            DELTA_T_WALL[i, j] = mean_temp_wall # Store

print("\nThe diameter of the HEX: ,",hex_diameter," m.")

fig = plt.figure(figsize=(12, 6))

ax2 = fig.add_subplot(111, projection='3d')
```

```
surf = ax2.plot_surface(NUM_FIN, MASSFLOW, DELTA_P, cmap='viridis',
↪  edgecolor='k', alpha=0.8)
ax2.set_xlabel('Number of Fins')
ax2.set_ylabel('Mass flow rate (kg/s)')
# ax2.set_zscale('log')
ax2.set_zlabel('Pressure Drop (Pa)')
ax2.set_title('3D Surface Plot of Pressure Drop')

fig.colorbar(surf, ax=ax2, shrink=0.5)

#For wall temperature
fig_2 = plt.figure(figsize=(10, 8))
ax_3 = fig_2.add_subplot(111, projection='3d')

z_plane = 46  # Define the z-plane value
x_vals, y_vals = np.meshgrid(NUM_FIN[0], MASSFLOW[:, 0])  # Create a grid for x
↪  and y
z_vals = np.full_like(x_vals, z_plane)  # Create a constant z-plane

# Surface plot
surf = ax_3.plot_surface(NUM_FIN, MASSFLOW, DELTA_T_WALL, cmap='viridis',
↪  edgecolor='k', alpha=0.8)
#also plot a z plane at z = 45 on the same 3D graph

ax_3.plot_surface(x_vals, y_vals, z_vals, color='red', alpha=0.5, label='z =
↪  46')  # Plot the z-plane

# Find intersection points between z-plane and DELTA_T_WALL curve
intersection_points = []
for i in range(NUM_FIN.shape[0]):
    for j in range(NUM_FIN.shape[1] - 1):
        if (DELTA_T_WALL[i, j] - z_plane) * (DELTA_T_WALL[i, j + 1] - z_plane)
        ↪  < 0:
        # Linear interpolation to find the intersection point
            t = (z_plane - DELTA_T_WALL[i, j]) / (DELTA_T_WALL[i, j + 1] -
            ↪  DELTA_T_WALL[i, j])
            num_fin_intersect = NUM_FIN[i, j] + t * (NUM_FIN[i, j + 1] -
            ↪  NUM_FIN[i, j])
            massflow_intersect = MASSFLOW[i, j] + t * (MASSFLOW[i, j + 1] -
            ↪  MASSFLOW[i, j])
            intersection_points.append((num_fin_intersect, massflow_intersect,
            ↪  z_plane))

# Plot intersection points as black dots
for point in intersection_points:
    ax_3.scatter(point[0], point[1], point[2], color='black', s=50) # if
    ↪  'Intersection' not in [t.get_text() for t in ax_3.get_legend().texts]
    ↪  else "")

# Labels & Titles
ax_3.set_xlabel('Number of Fins')
ax_3.set_ylabel('Mass flow rate (kg/s)')
ax_3.set_zlabel('Mean wall temp. (K)')
ax_3.set_zlim(40, 80)  # Set Z-axis limits
ax_3.set_title('3D Surface Plot of Wall temp.')
```

```
# Create a vertical plane along the intersection points that passes through the
↪  pressure drop curve
if intersection_points:
    # Extract x (Number of Fins) and y (Mass Flow Rate) coordinates of
    ↪  intersection points
    x_intersect = [point[0] for point in intersection_points]
    y_intersect = [point[1] for point in intersection_points]

    # Create a meshgrid for the vertical plane
    z_plane_2 = np.linspace(ax2.get_zlim()[0], ax2.get_zlim()[1], 50)  # Z-axis
    ↪  range
    x_plane, z_plane_2 = np.meshgrid(x_intersect, z_plane_2)  # Meshgrid for x
    ↪  and z
    y_plane = np.tile(y_intersect, (z_plane_2.shape[0], 1))  # Repeat y values
    ↪  for the plane

    # Plot the vertical plane on the DELTA_P 3D plot
    ax2.plot_surface(x_plane, y_plane, z_plane_2, color='red', alpha=0.5,
    ↪  label='Vertical Intersection Plane')

    # Create an interpolator for the DELTA_P surface
    interp_DELTA_P = RegularGridInterpolator((MASSFLOW[:, 0], NUM_FIN[0]),
    ↪  DELTA_P)

    # Store intersection points for DELTA_P surface
    delta_p_intersections = []

    print("Intersection points (x = Number of Fins, y = Mass Flow Rate (kg/s),
    ↪  z = Pressure Drop (Pa)):")
    for x, y in zip(x_intersect, y_intersect):
        try:
            z = interp_DELTA_P((y, x))  # MASSFLOW first, then NUM_FIN
            ax2.scatter(x, y, z, color='black', s=50)
            delta_p_intersections.append((x, y, float(z)))
            print(f"  x = {x:.3f}, y = {y:.4f}, z = {float(z):.3f}")
        except ValueError:
            pass  # Skip out-of-bound points

fig.colorbar(surf, ax=ax_3, shrink=0.6)

fig_4 = plt.figure(figsize=(10, 8))
ax_4 = fig_4.add_subplot(111)

# 2D plot
for i, massflow in enumerate(MASSFLOW[:, 0]):
    ax_4.plot(NUM_FIN[0], MASS_HEAT_SINK[i, :], label=f'Mass flow rate =
    ↪  {massflow:.3f} kg/s')

# Labels & Titles
ax_4.set_xlabel('Number of Fins')
ax_4.set_ylabel('Mass Heat Sink (Kg)')
ax_4.set_title('2D Plot of Heat Sink Mass vs Number of Fins')
# ax_4.legend()
ax_4.grid()
```

```
        fig_5 = plt.figure(figsize=(10, 8))
        ax_5 = fig_5.add_subplot(111, projection='3d')

        # Surface plot
        surf_5 = ax_5.plot_surface(NUM_FIN, MASSFLOW, DELTA_T, cmap='viridis',
        ↪  edgecolor='k', alpha=0.8)

        # Labels & Titles
        ax_5.set_xlabel('Number of Fins')
        ax_5.set_ylabel('Mass flow rate (kg/s)')
        ax_5.set_zlabel('Outlet fluid temp (K)')
        ax_5.set_title('3D Surface Plot of Outlet fluid temp')


        # Colorbar
        fig.colorbar(surf_5, ax=ax_5, shrink=0.6)

        fig_6 = plt.figure(figsize=(10, 8))
        ax_6 = fig_6.add_subplot(111, projection='3d')

        # Surface plot
        surf_6 = ax_6.plot_surface(NUM_FIN, MASSFLOW, VELOCITY, cmap='viridis',
        ↪  edgecolor='k', alpha=0.8)

        # Labels & Titles
        ax_6.set_xlabel('Number of Fins')
        ax_6.set_ylabel('Mass flow rate (kg/s)')
        ax_6.set_zlabel('Fluid velocity (m/s)')
        ax_6.set_title('3D Surface Plot of fluid velocity')

        # Colorbar
        fig.colorbar(surf_6, ax=ax_6, shrink=0.6)

        fig_7 = plt.figure(figsize=(10, 8))
        ax_7 = fig_7.add_subplot(111, projection='3d')

        # Surface plot
        surf_7 = ax_7.plot_surface(NUM_FIN, MASSFLOW, HEAT_TRANSFER_COEF,
        ↪  cmap='viridis', edgecolor='k', alpha=0.8)

        # Labels & Titles
        ax_7.set_xlabel('Number of Fins')
        ax_7.set_ylabel('Mass flow rate (kg/s)')
        ax_7.set_zlabel('Heat transfer coef (W/m2*K)')
        ax_7.set_title('3D Surface Plot of Heat transfer coef.')

        # Colorbar
        fig.colorbar(surf_7, ax=ax_7, shrink=0.6)

        plt.show()

# Creating a GUI for the code
import tkinter as tk
from tkinter import ttk
from tkinter import font
from PIL import Image, ImageTk
```

```
# Create a GUI window
root = tk.Tk()
root.title("Pressure Drop Calculator")

# Create and arrange widgets in the GUI
frame_inputs = ttk.Frame(root)
frame_inputs.grid(row=0, column=0, padx=20, pady=10)

frame_plot = ttk.Frame(root)
frame_plot.grid(row=1, column=0, padx=10, pady=10)

# Element Options Combobox
def update_visibility(event):
    selected_option = element_combobox.get()
    if selected_option == "Pipe/Tube":
        for widget in heat_sink_trapezoid_widgets:
            widget.grid_remove()
        for widget in pipe_widgets:
            widget.grid()
    if selected_option == "Heat Sink (Trapezoidal Channels)":
        for widget in pipe_widgets:
            widget.grid_remove()
        for widget in heat_sink_trapezoid_widgets:
            widget.grid()

label_element = ttk.Label(frame_inputs, text="Element Option:")
label_element.grid(row=0, column=0)
element_options = ["Pipe/Tube", "Heat Sink (Trapezoidal Channels)"] # Removed
↪  Rectangular
element_combobox = ttk.Combobox(frame_inputs, values=element_options)
element_combobox.grid(row=0, column=1)
element_combobox.set(element_options[0])  # Set default value to Pipe/Tube
element_combobox.bind("<<ComboboxSelected>>", update_visibility)

# Common Inputs
label_external_heatload = ttk.Label(frame_inputs, text="External Heat Load (Watts):")
label_external_heatload.grid(row=0, column=2)
entry_external_heatload = ttk.Entry(frame_inputs)
entry_external_heatload.grid(row=0, column=3)
entry_external_heatload.insert(0, "70")

label_inlet_pressure = ttk.Label(frame_inputs, text="Inlet Pressure (bara):")
label_inlet_pressure.grid(row=1, column=0)
entry_inlet_pressure = ttk.Entry(frame_inputs)
entry_inlet_pressure.grid(row=1, column=1)
entry_inlet_pressure.insert(0, "15")

label_inlet_temp = ttk.Label(frame_inputs, text="Inlet Temperature (K):")
label_inlet_temp.grid(row=1, column=2)
entry_inlet_temp = ttk.Entry(frame_inputs)
entry_inlet_temp.grid(row=1, column=3)
entry_inlet_temp.insert(0, "40")

label_m_dot = ttk.Label(frame_inputs, text="Min. Mass Flow Rate (kg/s):")
label_m_dot.grid(row=2, column=0)
entry_m_dot = ttk.Entry(frame_inputs)
entry_m_dot.grid(row=2, column=1)
```

```
entry_m_dot.insert(0, "0.016")

label_m_dot_max = ttk.Label(frame_inputs, text="Max. Mass Flow Rate (kg/s):")
label_m_dot_max.grid(row=2, column=2)
entry_m_dot_max = ttk.Entry(frame_inputs)
entry_m_dot_max.grid(row=2, column=3)
entry_m_dot_max.insert(0, "0.080")

label_fluid_type = ttk.Label(frame_inputs, text="Fluid:")
label_fluid_type.grid(row=3, column=2)
fluid_type_options = ["hydrogen", "helium", "nitrogen"]
fluid_type_combobox = ttk.Combobox(frame_inputs, values=fluid_type_options)
fluid_type_combobox.grid(row=3, column=3)
fluid_type_combobox.set(fluid_type_options[0])

# Pipe/Tube Specific Inputs
# DISPLAY THIS PART OF THE CODE ON THE GUI FOR PIPE/TUBE OPTION ONLY
↪ ----------------------------------------------

pipe_widgets = []

label_tube = ttk.Label(frame_inputs, text="For PIPE/TUBE Only:", font=("Helvetica", 14,
↪  "bold"))
label_tube.grid(row=4, column=0)
pipe_widgets.append(label_tube)

#Pipe diameter range
label_inner_diameter_range = ttk.Label(frame_inputs, text="Inner Diameter Range (mm):")
label_inner_diameter_range.grid(row=5, column=0)
pipe_widgets.append(label_inner_diameter_range)

entry_min_inner_diameter = ttk.Entry(frame_inputs)
entry_min_inner_diameter.grid(row=5, column=1)
entry_min_inner_diameter.insert(0, "25")
pipe_widgets.append(entry_min_inner_diameter)

entry_max_inner_diameter = ttk.Entry(frame_inputs)
entry_max_inner_diameter.grid(row=5, column=2)
entry_max_inner_diameter.insert(0, "100")
pipe_widgets.append(entry_max_inner_diameter)

# Pipe Lengths
label_pipe_length = ttk.Label(frame_inputs, text="Pipe Lengths (comma-separated, e.g.,
↪  1,3,5,10):")
label_pipe_length.grid(row=6, column=0)
pipe_widgets.append(label_pipe_length)
entry_pipe_length = ttk.Entry(frame_inputs)
entry_pipe_length.grid(row=6, column=1)
entry_pipe_length.insert(0, "1,3,5,10")
pipe_widgets.append(entry_pipe_length)

# Number of channels
label_no_channels = ttk.Label(frame_inputs, text="Number of channels")
label_no_channels.grid(row=6, column=2)
pipe_widgets.append(label_no_channels)
entry_no_channels = ttk.Entry(frame_inputs)
entry_no_channels.grid(row=6, column=3)
```

```
entry_no_channels.insert(0, "1")
pipe_widgets.append(entry_no_channels)

# Pipe type
label_pipe_type = ttk.Label(frame_inputs, text="Pipe type:")
label_pipe_type.grid(row=7, column=0)
pipe_widgets.append(label_pipe_type)
pipe_type_options = ["Rigid", "Flexible"]
pipe_type_combobox = ttk.Combobox(frame_inputs, values=pipe_type_options)
pipe_type_combobox.grid(row=7, column=1)
pipe_type_combobox.set(pipe_type_options[0])  # Set default value
pipe_widgets.append(pipe_type_combobox)

# No. of 90 deg elbow welded
label_90_elbow = ttk.Label(frame_inputs, text="No. of 90 deg elbows (welded):")
label_90_elbow.grid(row=8, column=0)
pipe_widgets.append(label_90_elbow)
entry_90_elbow = ttk.Entry(frame_inputs)
entry_90_elbow.grid(row=8, column=1)
entry_90_elbow.insert(0, "0")
pipe_widgets.append(entry_90_elbow)

# Standard or long radius
label_radius_type = ttk.Label(frame_inputs, text="Radius type:")
label_radius_type.grid(row=8, column=2)
pipe_widgets.append(label_radius_type)
radius_type_options = ["Standard radius (R/D = 1)", "Standard radius (R/D = 1.5)",
↪  "Long radius (R/D = 2)"]
radius_type_combobox = ttk.Combobox(frame_inputs, values=radius_type_options)
radius_type_combobox.grid(row=8, column=3)
radius_type_combobox.set(radius_type_options[0])  # Set default value
pipe_widgets.append(radius_type_combobox)

# No. of Tee straight
label_T_straight = ttk.Label(frame_inputs, text="No. of T straight:")
label_T_straight.grid(row=9, column=0)
pipe_widgets.append(label_T_straight)
entry_T_straight = ttk.Entry(frame_inputs)
entry_T_straight.grid(row=9, column=1)
entry_T_straight.insert(0, "0")
pipe_widgets.append(entry_T_straight)

# Standard or long radius
label_radius_T_type = ttk.Label(frame_inputs, text="Radius type:")
label_radius_T_type.grid(row=9, column=2)
pipe_widgets.append(label_radius_T_type)
radius_T_type_options = ["Flanged (R/D = 1)", "Stub-in branch"]
radius_T_type_combobox = ttk.Combobox(frame_inputs, values=radius_T_type_options)
radius_T_type_combobox.grid(row=9, column=3)
radius_T_type_combobox.set(radius_T_type_options[0])  # Set default value
pipe_widgets.append(radius_T_type_combobox)

# No. of Tee branch
label_T_branch = ttk.Label(frame_inputs, text="No. of T branch:")
label_T_branch.grid(row=10, column=0)
pipe_widgets.append(label_T_branch)
entry_T_branch = ttk.Entry(frame_inputs)
```

```
entry_T_branch.grid(row=10, column=1)
entry_T_branch.insert(0, "0")
pipe_widgets.append(entry_T_branch)

# Standard or long radius(T branch)
label_radius_T_branch_type = ttk.Label(frame_inputs, text="Radius type:")
label_radius_T_branch_type.grid(row=10, column=2)
pipe_widgets.append(label_radius_T_branch_type)
radius_T_branch_type_options = ["Standard radius (R/D = 1.5)", "Stub-in branch"]
radius_T_branch_type_combobox = ttk.Combobox(frame_inputs,
↪  values=radius_T_branch_type_options)
radius_T_branch_type_combobox.grid(row=10, column=3)
radius_T_branch_type_combobox.set(radius_T_branch_type_options[0])  # Set default value
pipe_widgets.append(radius_T_branch_type_combobox)

# No. of 180 bend
label_180_bend = ttk.Label(frame_inputs, text="No. of 180 deg bends:")
label_180_bend.grid(row=11, column=0)
pipe_widgets.append(label_180_bend)
entry_180_bend = ttk.Entry(frame_inputs)
entry_180_bend.grid(row=11, column=1)
entry_180_bend.insert(0, "0")
pipe_widgets.append(entry_180_bend)

# Standard or long radius(180 bend)
label_radius_180_bend_type = ttk.Label(frame_inputs, text="Radius type:")
label_radius_180_bend_type.grid(row=11, column=2)
pipe_widgets.append(label_radius_180_bend_type)
radius_180_bend_type_options = ["Flanged (R/D = 1)", "All types (R/D = 1.5)"]
radius_180_bend_type_combobox = ttk.Combobox(frame_inputs,
↪  values=radius_180_bend_type_options)
radius_180_bend_type_combobox.grid(row=11, column=3)
radius_180_bend_type_combobox.set(radius_180_bend_type_options[0])  # Set default value
pipe_widgets.append(radius_180_bend_type_combobox)

# --- Rectangular Heat Sink Widgets Removed ---

heat_sink_trapezoid_widgets = []
# DISPLAY THIS PART OF THE CODE ON THE GUI FOR HEAT SINK OPTION ONLY
↪  ------------------------------------------------
x_space = 0
y_space = 0
new_width_trapezoid = 700
new_height_trapezoid = 350
# Open and resize the image TRAPEZOID
# *** NOTE: You must update this path to match your file location ***
try:
    image_trapezoid = Image.open('C:/Users/KGUPJHAC/Desktop/Python_Scripts/Heat sink
    ↪  thermo-fluid dynamic scripts/trapezoidal_fin.PNG')
    image_trapezoid = image_trapezoid.resize((new_width_trapezoid,
    ↪  new_height_trapezoid), Image.Resampling.LANCZOS)
    # Convert to PhotoImage
    image_trapezoid = ImageTk.PhotoImage(image_trapezoid)
    # Create and place the label
    label_image_trapezoid = ttk.Label(frame_inputs, image=image_trapezoid)
    label_image_trapezoid.grid(row=5, column=0)
    # Keep a reference to the image to prevent garbage collection
```

```
    label_image_trapezoid.image = image_trapezoid
    heat_sink_trapezoid_widgets.append(label_image_trapezoid)
except FileNotFoundError:
    print("Warning: Trapezoidal fin image not found. GUI will not display the image.")
    label_image_not_found = ttk.Label(frame_inputs, text="[Trapezoidal Fin Image Not
    ↪  Found]")
    label_image_not_found.grid(row=5, column=0)
    heat_sink_trapezoid_widgets.append(label_image_not_found)


label_emptyspace_2 = ttk.Label(frame_inputs, text="                          ")
label_emptyspace_2.grid(row=6, column=0)
heat_sink_trapezoid_widgets.append(label_emptyspace_2)

label_heat_sink = ttk.Label(frame_inputs, text="For HEAT SINK Only:", font=("Helvetica",
↪  14, "bold"))
label_heat_sink.grid(row=4, column=0)
heat_sink_trapezoid_widgets.append(label_heat_sink)

# Diameters of poles
label_heat_sink_poles_OD = ttk.Label(frame_inputs, text="Poles diameter (mm):")
label_heat_sink_poles_OD.grid(row=6-y_space, column=2+x_space)
heat_sink_trapezoid_widgets.append(label_heat_sink_poles_OD)

entry_heat_sink_poles_OD = ttk.Entry(frame_inputs)
entry_heat_sink_poles_OD.grid(row=6-y_space, column=3+x_space)
entry_heat_sink_poles_OD.insert(0, "5")
heat_sink_trapezoid_widgets.append(entry_heat_sink_poles_OD)

# Number of poles
label_no_poles = ttk.Label(frame_inputs, text="Number of poles:")
label_no_poles.grid(row=7-y_space, column=2+x_space)
heat_sink_trapezoid_widgets.append(label_no_poles)

entry_no_poles = ttk.Entry(frame_inputs)
entry_no_poles.grid(row=7-y_space, column=3+x_space)
entry_no_poles.insert(0, "4")
heat_sink_trapezoid_widgets.append(entry_no_poles)

#Heat sink length and width range
label_heat_sink_length = ttk.Label(frame_inputs, text="Heat Sink length (mm):")
label_heat_sink_length.grid(row=6-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_heat_sink_length)
entry_heat_sink_length = ttk.Entry(frame_inputs)
entry_heat_sink_length.grid(row=6-y_space, column=1+x_space)
entry_heat_sink_length.insert(0, "100")
heat_sink_trapezoid_widgets.append(entry_heat_sink_length)

#Heat sink fin height, thickness
label_fin_height = ttk.Label(frame_inputs, text="Fin height (mm):")
label_fin_height.grid(row=7-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_fin_height)
entry_fin_height = ttk.Entry(frame_inputs)
entry_fin_height.grid(row=7-y_space, column=1+x_space)
entry_fin_height.insert(0, "10")
heat_sink_trapezoid_widgets.append(entry_fin_height)
```

```
label_c_space = ttk.Label(frame_inputs, text="Spacing between 2 adjacent poles [mm]:")
label_c_space.grid(row=8-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_c_space)
entry_c_space = ttk.Entry(frame_inputs)
entry_c_space.grid(row=8-y_space, column=1+x_space)
entry_c_space.insert(0, "10")
heat_sink_trapezoid_widgets.append(entry_c_space)

label_h_space = ttk.Label(frame_inputs, text="Edge clearance [mm]:")
label_h_space.grid(row=8-y_space, column=2+x_space)
heat_sink_trapezoid_widgets.append(label_h_space)
entry_h_space = ttk.Entry(frame_inputs)
entry_h_space.grid(row=8-y_space, column=3+x_space)
entry_h_space.insert(0, "2")
heat_sink_trapezoid_widgets.append(entry_h_space)

#Number of Fins
label_no_fins = ttk.Label(frame_inputs, text="Min. No. of fins:")
label_no_fins.grid(row=9-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_no_fins)
entry_no_fins = ttk.Entry(frame_inputs)
entry_no_fins.grid(row=9-y_space, column=1+x_space)
entry_no_fins.insert(0, "20")
heat_sink_trapezoid_widgets.append(entry_no_fins)

label_max_no_fins = ttk.Label(frame_inputs, text="Max No. of fins:")
label_max_no_fins.grid(row=9-y_space, column=2+x_space)
heat_sink_trapezoid_widgets.append(label_max_no_fins)
entry_max_no_fins = ttk.Entry(frame_inputs)
entry_max_no_fins.grid(row=9-y_space, column=3+x_space)
entry_max_no_fins.insert(0, "30")
heat_sink_trapezoid_widgets.append(entry_max_no_fins)

#Heat sink fin thickness
label_fin_thickness = ttk.Label(frame_inputs, text="Fin Thickness (mm):")
label_fin_thickness.grid(row=10-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_fin_thickness)
entry_fin_thickness = ttk.Entry(frame_inputs)
entry_fin_thickness.grid(row=10-y_space, column=1+x_space)
entry_fin_thickness.insert(0, "1")
heat_sink_trapezoid_widgets.append(entry_fin_thickness)

# label_max_fin_thickness = ttk.Label(frame_inputs, text="Max Fin Thickness (mm):")
# label_max_fin_thickness.grid(row=10-y_space, column=2+x_space)
# heat_sink_trapezoid_widgets.append(label_max_fin_thickness)
# entry_max_fin_thickness = ttk.Entry(frame_inputs)
# entry_max_fin_thickness.grid(row=10-y_space, column=3+x_space)
# entry_max_fin_thickness.insert(0, "2")
# heat_sink_trapezoid_widgets.append(entry_max_fin_thickness)

#HEAT SINK Material
label_sink_material = ttk.Label(frame_inputs, text="Heat Sink material:")
label_sink_material.grid(row=11-y_space, column=0+x_space)
heat_sink_trapezoid_widgets.append(label_sink_material)
sink_material_options = ["Aluminium", "Copper"]
materials_combobox = ttk.Combobox(frame_inputs, values=sink_material_options)
materials_combobox.grid(row=11-y_space, column=1+x_space)
```

```
materials_combobox.set(sink_material_options[0])  # Set default value
heat_sink_trapezoid_widgets.append(materials_combobox)

# Set initial visibility
update_visibility(None)

# Calculate Button
calculate_button = ttk.Button(frame_inputs, text="Calculate",
↪  command=calculate_pressure_drop)
calculate_button.grid(row=14, column=2, columnspan=3)

root.mainloop()
```