TECHNICAL UNIVERSITY OF LODZ

Faculty of Electrical, Electronic, Computer and Control Engineering

MASTER OF SCIENCE THESIS

Automated, Object-Oriented Simulation Framework for Modelling of Superconducting Magnets at CERN

Michał Maciejewski

178089

CERN Supervisor

Emmanuele Ravaioli

TUL Supervisor

prof. Andrzej Bartoszewicz

Geneva, August, 2014

Contents

Acknowledgements

I would like to thank my supervisors, colleagues from TE-MPE-PE group at CERN, family and friends who supported me during simulation framework development and thesis writing.

Firstly, many thanks to my supervisor Emmanuele Ravaioli for his continuous support, guidance and patience during design and development of the Quench Simulation Framework as well as many constructive comments throughout the thesis creation.

Secondly, thanks to Arjan Verweij and Mateusz Koza who thoroughly reviewed some chapters of the thesis and professor Andrzej Bartoszewicz who monitored my thesis progress from Łódź.

Finally, I would like to express my gratitude to my brothers Adam and Marcin, parents Małgorzata and Krzysztof, grandparents Wiesława and Józef, my best friend Julian and girlfriend Marta who accompanied me during this tough period and kept their fingers crossed for me to finish the thesis in time.

1 Introduction

Conseil Européen pour la Recherche Nucléaire (CERN) is currently one of the largest particle physics laboratories in the world. CERN was founded in 1954 in Geneva and is located near France-Swiss border between Lake Geneva and Jura Mountains. Nowadays, the European Organization for Particle Physics has 21 member states and employs scientists and engineers from round the world. The CERN mission is the fundamental research and study of the basic elements of matter – fundamental particles. The experiments in high energy physics (HEP) are intended to answer questions about the origins of the Universe: *Why there is more matter than antimatter? What is the source of gravity force? What is dark matter made of?* [57]

For that purpose at CERN the complex of accelerators has been built over decades as presented in Figure 1.1.

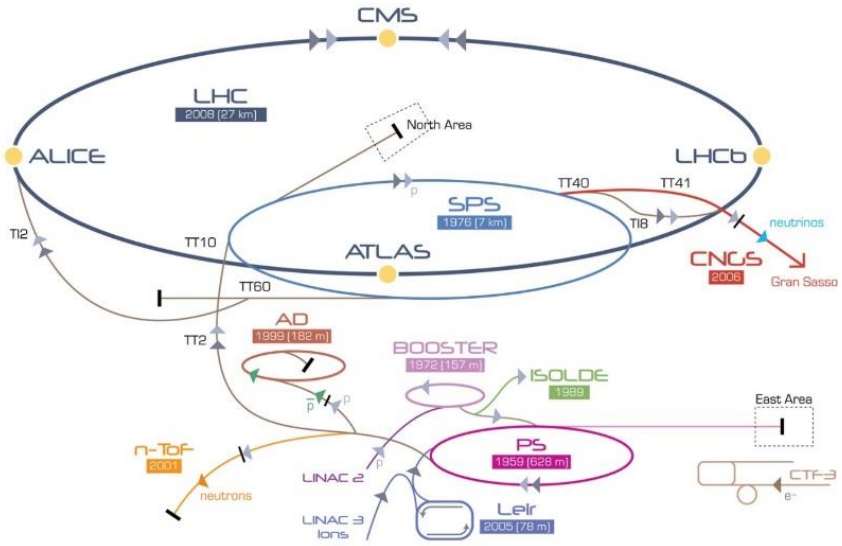


Figure 1.1. CERN Accelerator Complex [26]

In order to find answers to aforementioned questions, the particles like protons or lead ions are accelerated in the chain of accelerators to velocities close to the speed of light and then are made to collide. The protons start their journey from a bottle of hydrogen gas and they are fed into the linear accelerator Linac2 that uses a strong electric field to extract electrons from the hydrogen atoms and increase their energy. Second stage is the Proton Synchrotron Booster (PSB) that further increases the energy of beams of protons from 50 MeV to 1.4 GeV. When the particles are fed to the Proton Synchrotron (PS) they are twenty-five times heavier than at rest. The PS is followed by the Super Proton Synchrotron (SPS), a ring of 7 kilometres in circumference that further increases the energy up to 450 GeV (at this point the velocity of beams is reaching the limit of speed of light, hence the mass of particles increases). [26]

Finally, the beams of particles are injected to the Large Hadron Collider (LHC). LHC is a circle of 27 kilometres and is built at a mean depth of 100 meters. The LHC is composed of 8 sectors with straight section and the arc (see Figure 1.2). The arc is full of superconducting magnets that bend the beam and adjust its shape, and straight sections where the experiments and accelerating structures to boost the beam energy are located. Inside the accelerator, two high-energy particle beams travel in opposite directions in separate pipes. The two tubes require an ultrahigh vacuum in order to avoid spurious collisions. The magnets with turns made of a special materials, in order to become superconducting must be cooled down to 1.9 K – a temperature colder than outer space [27].

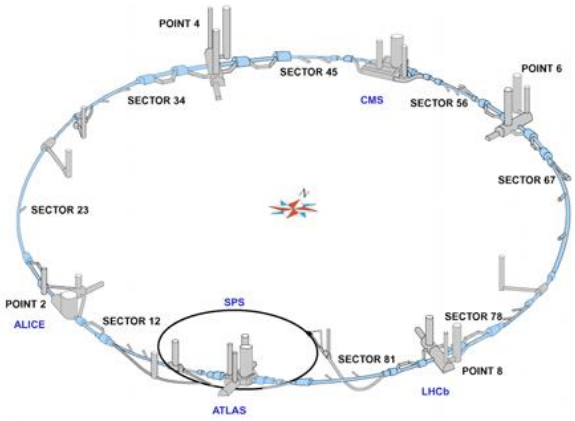


Figure 1.2. LHC schematic with main sectors [29]

After reaching the desired energy, the two beams of particles are made to collide. The energy of particles obtained in the accelerator depends on its circumference and magnetic field produced by the magnets needed to keep the particles inside the beam pipe. The results of particles collisions are studied in 4 gigantic detectors (so called experiments) [27]:

1. ALICE (A Large Ion Collider Experiment) is a detector designed to study results of lead-ion collisions that should provide insights on the quark-gluon plasma. Such a state of matter is supposed to exist after the Big Bang.
2. ATLAS (A Toroidal LHC Aparatus) is an experiment detecting wide range of particles resulting from collisions in the LHC. It is the largest-volume detector that has been ever constructed.
3. CMS (Compact Muon Solenoid) is a detector with similar purpose as ATLAS but different technical solution.
4. LHCb (Large Hadron Collider beauty) focuses on the study of the asymmetry between matter and antimatter in interactions of *B*-particles that contain *b* quark.

The ATLAS and CMS observed a new particle in the mass region around 126 GeV on 4 July 2012. The particle that is consistent with the Higgs boson proposed within the Standard Model. For that discovery François Englert and Peter Higgs received the Nobel prize in physics in 2013. [57]

Superconducting magnets play an important role in both accelerators and detectors. They are employed to control trajectory and shape of the beam of particles. The dipole magnets bend the beam in arc sections whereas quadrupole magnets squeeze the particles in order to increase the probability of head-on collisions in the detectors. There are also sextupole, octapole and decapole superconducting magnets, that correct the beams of particles. The superconducting magnets in the detectors are applied in order to bend and separate the particles with positive charge from the ones with negative charge.

1.1. Thesis Scope

During the current LHC operation its upgrade is already in design phase. In order to reach higher luminosities new generation of superconducting magnets is being developed. The protection of high-field Nb_3Sn accelerator magnets presents considerable technical challenge due to the much higher stored energy per unit volume. Additionally, more energy is needed in order to provoke and propagate the transition from the superconducting to normal state (quench) in such magnets. Hence, the new method to protect the magnets – Coupling Loss Induced Quench (CLIQ) has been recently developed at CERN. In order to analyze in depth all consequences of the CLIQ application for stand-alone magnets as well as more importantly chains of magnets, series of simulations along with experiments must be performed. The CLIQ system is capable of transferring large portions of the magnet to the normal zone. Hence, two dimensional model is a good approximation as in such a case longitudinal properties of the model are homogeneous. Currently, the superconducting magnets at CERN are modelled in the ROXIE program (employing finite elements method in three dimensions) as well as the OrCAD PSpice environment (applying lumped-element two dimensional dynamic electro-thermal model). The former does not include the dynamic feedback of inter-filament and inter-strand coupling currents on transport current in the magnet, which are the key ingredients of the CLIQ quench initiation mechanism. Subsequently, the later implements a coupling between filaments, strands and magnet turns, however, the model can be only developed manually. In case of the models composed of thousands of components such approach is extremely time consuming and prone to errors.

In order to study new quench protection methods, as well as existing ones in more convenient way there was a need to develop a new quench simulation framework on the basis of experience and knowledge already implemented in aforementioned software. The framework that on the one hand should implement dynamic effects of inter-filament and inter-strand coupling currents in the magnet and on the other hand allows conveniently performing various simulation scenarios.

1.2. Thesis Aim

Firstly, the thesis aims at design and development of a simulation environment that implements a full 2D electro-thermal dynamic model of superconducting magnet. Secondly, the simulation framework should allow automatically building models on the basis of an input file. Thirdly, the new framework should be easy to extend and modify. In order to further improve the modeller's experience, the framework should be equipped with a GUI, parametric sweep, and parallel computing modules. In other words a new simulation environment should be scalable, flexible and maintainable so that the user can only focus on model structure and parameters definition and analysis of the obtained results after simulation. Such features are not present by default in any simulation software available on the market. Furthermore, majority of those programs (OrCAD PSpice, PSIM, etc.) focus on simulation of models created manually and do not support automatic model development. However the application of a sophisticated software architecture along with a powerful network solver with a library of reusable components seems to be a promising solution. For that purpose MATLAB (a numerical analysis environment) along with its companion Simulink (simulation environment) were employed. The MATLAB language includes an ability to employ Object Oriented Programming (that is a natural solution while developing complex software projects) as well as contains functions to build Simulink models automatically.

1.3. Thesis Content

The remainder of the thesis is organized as follows. Chapter 2 briefly describes the superconductivity and superconducting magnets. Chapter 3 contains the description of lumped-element dynamic electro-thermal model of superconducting magnets. It presents all assumptions as well as analytical formulas applied in the model. Chapter 4 provides a detailed list of the framework requirements followed by explanation of framework architecture and implementation. Chapter 5 illustrates the development of the graphical user interface that creates a closed, user-friendly interface to conveniently perform simulations. Chapter 6 presents first results of the simulations performed by the automated, object oriented framework. Chapter 7 concludes the thesis, provides a summary of results as well as plans for the future.

2 Superconductivity

2.1. History of Superconductivity

The effect of the resistance decrease of conductors (such as copper, aluminium) with decrease of the temperature is well understood as the atoms in a material structure oscillate with lower energy so electrons can move with more freedom. In 1911 Dutch physicist Heike Kamerling-Onnes while carrying out experiments in very low temperatures with certain metals (such as mercury, lead, and tin) discovered a completely different behaviour [38]. Below a certain temperature value the resistance of those metals sharply dropped to zero. However, the mercury subjected to an external field magnetic field of about 1/20 T transfers to the normal state [32]. Nevertheless, the experiments performed by Kamerling-Onnes created a new field of physics – superconductivity. In fact, the superconductivity is an interesting phenomenon, whereby particular materials have no electrical resistance (or a value that is non-measurable with available sensors). Over 50 years later new significant discoveries came up in USA. Matthias and Kunzler obtained a new generation of high-field superconducting materials. Their research focused on alloys that were able to remain in the superconducting state up to very high fields and at very high current densities [32].

Various superconducting materials along with their critical temperature are presented in Table 2.1.

Material	Critical temperature T_c [K]	Comments
Hg	4.2	First superconductor discovered in 1911.
Nb	9.3	The highest T_c among all elements under normal pressure.
Nb_3Sn	18	New superconductor type to be used in future accelerator magnets.
Nb-Ti	9.2	Currently most widely used superconducting alloy in accelerator magnets.
$YBa_2Cu_3O_7$	92	The first discovered superconductor with T_c > 77 K (hence can operate in liquid nitrogen).
MgB_2	39	Superconductivity discovered in 2001.
ReBCO	90	Operates in Tc > 77 K and for temperatures in range 4-20 K can withstand high magnetic field (up to 50 T)
H	300	Expected value for the metallic phase under a pressure greater than a pressure inside the Earth core.

Table 2.1. Description of key superconducting materials in the superconductivity history [38]

2.2. Conditions for Superconductivity

For a given superconductor three parameters describing conditions for superconductivity are defined:

1. critical magnetic field (at zero current and zero temperature) B_{c0},
2. critical temperature (at zero current and zero magnetic field) T_{c0},
3. critical current density (at zero temperature and zero magnetic field) J_{c0}.

However, third parameter depends on the superconducting cable production process (the ratio between superconducting material and non-superconducting stabilizer such as copper, aluminium) and may vary from one solution to another.

The superconducting material is in the superconducting state when the following three conditions are satisfied:

1. The temperature T of the superconductor must stay below critical temperature,
2. The magnetic field B in the superconductor must be less then critical magnetic field,
3. The current density J in the superconductor cannot exceed critical current density.

The critical surface BJT combines together all three properties and has different shape for particular material (see Figure 2.1). [32]

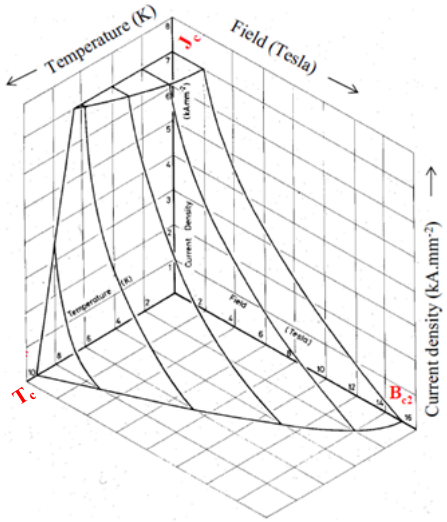


Figure 2.1. Critical surface for Nb-Ti adopted from [32]

The critical surface defines a boundary between the superconducting and the normal state, i.e. the material has no resistance below the surface and becomes resistive outside of the surface. Additionally, note that an increase of one critical parameter results in decrease of the other two (the critical curves are strictly monotonic).

2.3. Superconducting Magnets Applications

Nowadays, the superconducting materials are commonly used in superconducting magnets producing high magnetic fields. They are applied in new generation of means of transport, medical devices, electrical energy storage and transfer, and accelerator magnets.

The superconducting magnets currently allow producing high magnetic field at reasonable cost. Such a feature made it possible to design a new generation of transport vehicles (for example trains, trams) that are capable of levitating over the railroad. Hence, it was possible to obtain higher velocities as compared to conventional means of transport, achieving at the same time higher robustness (no moving parts, reduced friction, reduced noise). [32]

Another real-life application of superconducting magnets is magnetic resonance imaging (MRI) technique. MRI allows investigating the anatomy and physiology of body to support diagnostics. MRI scanners employ strong magnetic fields along with radio waves to create an image of tissues inside the human body. Moreover, this way of imaging does not expose patients to a dangerous ionizing radiation as in a case of X-ray based scanners. [56]

The superconductivity has a big potential to be applied in power energy applications as it is capable of significantly reducing energy losses. Hence it can be used to store electrical energy as well as to transport energy. Recently at CERN a high-temperature MgB_2 superconducting cable has been successfully applied in a superconducting link [55].

Another interesting application of superconducting materials is a fault current limiter [43]. A superconducting fault limiter allows limiting the current as soon as it increases above a certain threshold. In normal operation, such a device presents a negligible impedance, whereas above a given current introduces a high impedance in the protected circuit. Moreover, when the current recovers to the safe limit, the impedance automatically disappears.

This thesis describes a simulation framework for superconducting magnets modelling. In order to reduce the model complexity the magnet cable is in the superconducting state if the conditions defined above are met, otherwise the magnet is treated as a conventional conductor. Furthermore, no attempt will be made to explain superconductivity in detail as this topic has been already widely described in the literature [38].

2.4. Superconducting Accelerator Magnet Types

A coil made of low-temperature superconducting material (such as Nb-Ti or Nb_3Sn) is capable of carrying high current density and producing high magnetic field. In order to obtain such desirable features in many applications extremely low temperatures (less than 10 K) are required; typical

operating temperatures are in the range of the helium boiling point (4.2 K) or even less. On the contrary, conventional magnets can operate at room temperature. However, due to the iron saturation, the operating range of magnetic field without additional cooling for conventional magnets is limited to about 2 T. Figure 2.2 illustrates the operating range of standard electromagnets and superconducting magnets (critical curves at a constant temperature of 4.2 K).

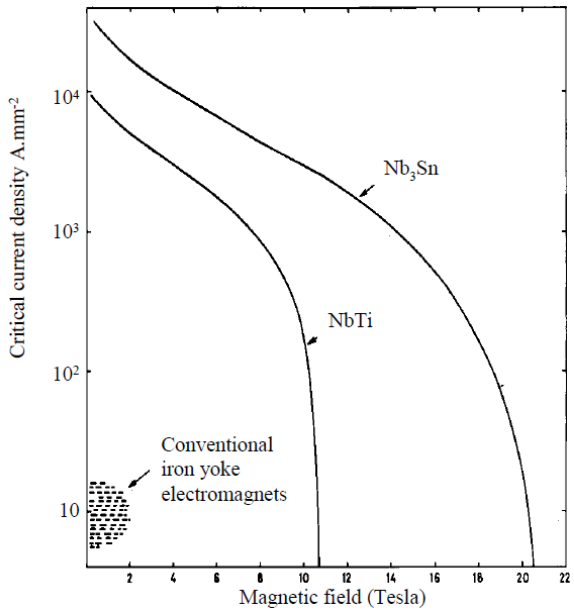


Figure 2.2. Comparison between the operating range of conventional iron-coated electromagnets and low-temperature superconducting magnets at a temperature of 4.2 K [32]

As one can see in Figure 2.2 the superconducting cables allow obtaining higher magnetic fields. To date niobium-titanium is the most common superconducting alloy applied in accelerator magnets, whereas Nb_3Sn featuring higher critical magnetic field, temperature and density is currently considered as a very promising, next generation material for superconducting magnets. Table 2.2 summarizes the comparison between conventional electromagnets and superconducting magnets.

Feature	Conventional electromagnets	Superconducting magnets
Yoke	iron	soft iron
Magnetic field	low (limited)	High (even > 10 T)
Power dissipation	yes	no
Amperturns cost	high	low
Refrigeration	not required	required

Table 2.2. Comparison between electromagnets and superconducting magnets [32]

The current flow in the superconducting cable is initiated by a power converter. Then only power required to maintain the current flow in a superconducting magnet is the refrigeration power needed to keep it at low temperature.

Superconducting magnets are a key ingredient for high-energy particle accelerators. They are used for controlling the trajectory of particle beams and controlling their shape. They are composed

of multiple cable turns. Figure 2.3 presents a cross-section of the superconducting main dipole magnet used in the LHC.

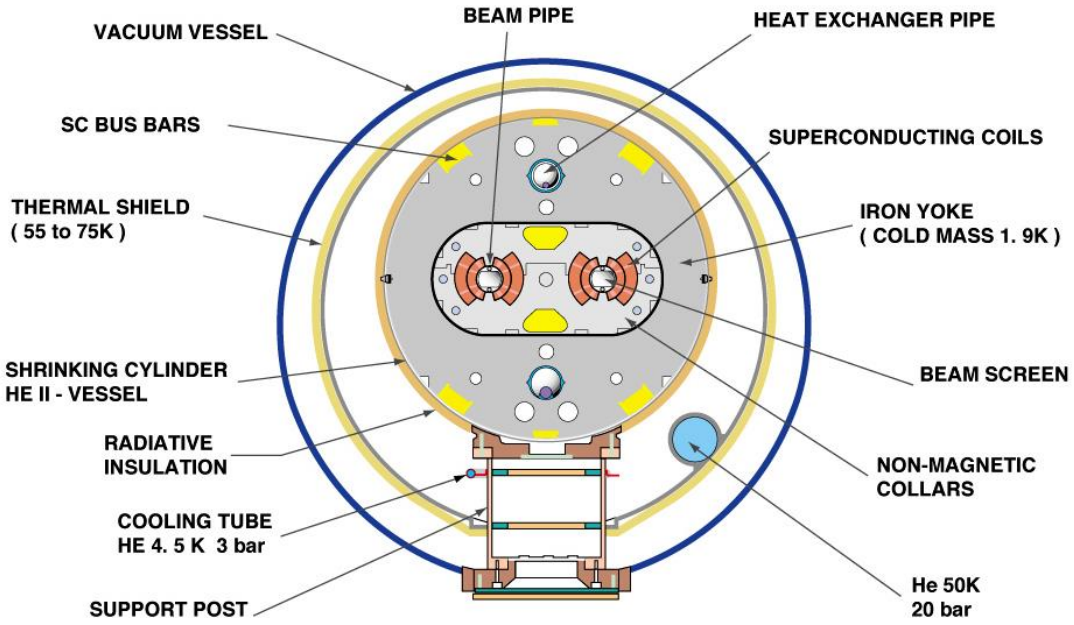


Figure 2.3. Cross-section of the LHC main dipole [44]

The LHC magnet system is composed of 1232 superconducting dipoles and 386 main quadrupoles along with about 20 different types of additional magnets for beam insertions and correction (quadrupoles, sextupoles, octapoles, decapoles). Since in the LHC there are two pipes with beams of particles travelling in the opposite directions twin-aperture magnets are often employed (Figure 2.4). [37]

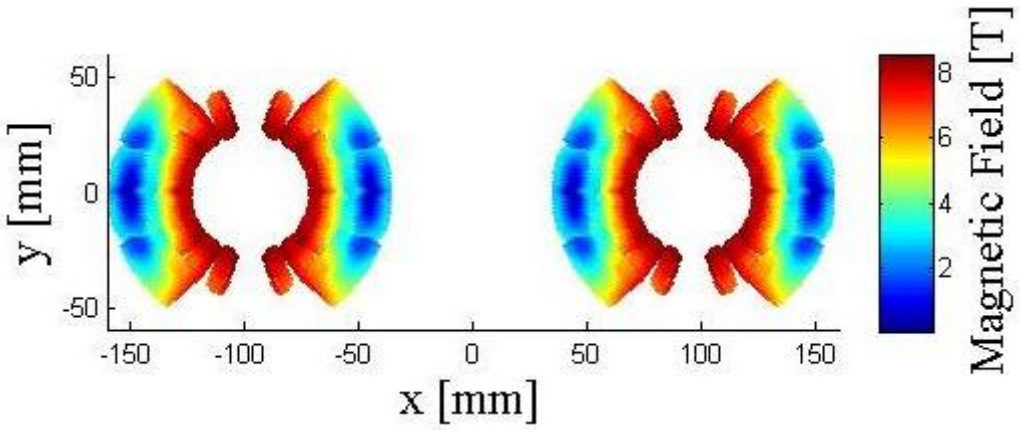


Figure 2.4. Magnetic field in the LHC twin-aperture main dipole magnet simulated with ROXIE

The working principle of superconducting magnets in particles accelerators is the Lorentz force [25].

2.4.1. Dipole Magnets

The dipole magnets are composed of a single pair of magnetic poles (N-S) in order to bend the beam of particles (Figure 2.5). Thus, in the LHC they are placed in the arc sections. Following the

right hand rule and assuming that field lines **B** in a magnet go from N to S and the positively charged particles q travel with velocity **v** out of the page, the Lorentz force **F** is to the right

$$\mathbf{F} = q\mathbf{v} \times \mathbf{B}\,. \qquad [\mathrm{N}] \quad (2.1)$$

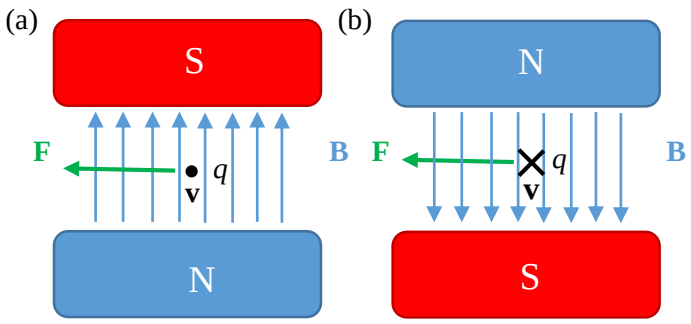


Figure 2.5. Schematic of a twin-aperture dipole magnet and Lorentz force acting on the particles traveling (a) into and (b) out of the page, respectively.

2.4.2. Quadrupole Magnets

The quadrupole magnets are composed of four magnetic poles as presented in Figure 2.6. The magnetic field in the middle of a magnet is equal to zero and grows as the radial distance increases. Hence, they are used in accelerator magnets to squeeze the particles in order to obtain higher beam luminosity, which in turn increases the probability of particles collisions. In fact, as one can notice from Figure 2.6 the quadrupole magnet is focusing beam of particles in one plane and defocusing them in the other. In other words, the quadrupole accelerator magnet works as a lens for particles that simultaneously operates in two planes.

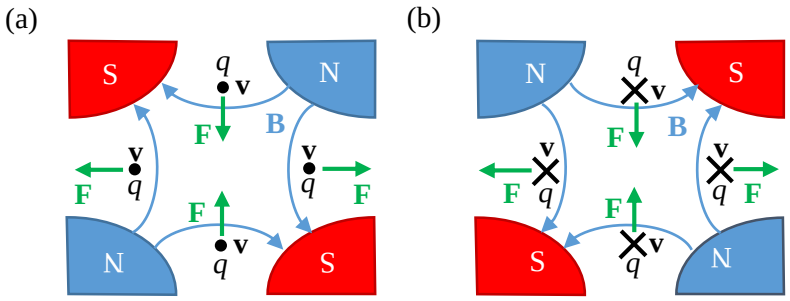


Figure 2.6. Schematic representation of a twin-aperture quadrupole magnet and Lorentz force acting on the particles traveling (a) into and (b) out of the page, respectively.

2.4.3. Other Magnets

Another important class of magnets used in the LHC are corrector magnets composed of more than 4 poles such as sextupoles (MCS), octupoles (MCO), decapoles (MCD) compensate dipole magnetic field imperfections. [40, 37]. Additionally, there are also kicker magnets that are used to

inject the accelerated beam of particles from one accelerator to another [37]. Detailed description of various magnet types applied in the LHC can be found in [37].

2.5. Superconducting Cables and Wires

The Nb-Ti superconductor cable is created in a complicated industrial process including many stages. Firstly, the filaments are created by twisting a large amount of superconductor wires. Secondly, twisted filaments are covered by a copper stabilizer (Figure 2.7).

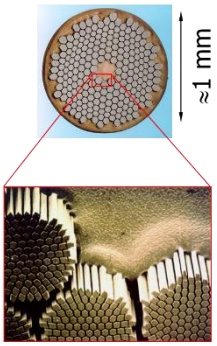


Figure 2.7. Structure of superconducting strands and filaments

An important parameter of the resulting strands is the filament twist-pitch defined as the axial length in which a filament firstly returns to its original relative position in a twisted conductor. Finally, the strands are used to create superconducting cables that are directly used to wind the magnet coils.

Since the cables generate a non-negligible self-magnetic field that reduces the main magnetic field, it is necessary to change the position of a strand in order to compensate that field. This allows reducing field errors and AC losses. There are three main ways (see Figure 2.8) of twisting strands in the superconducting cables to limit magnetic field deterioration:

1. Rope,
2. Braid,
3. Rutherford.

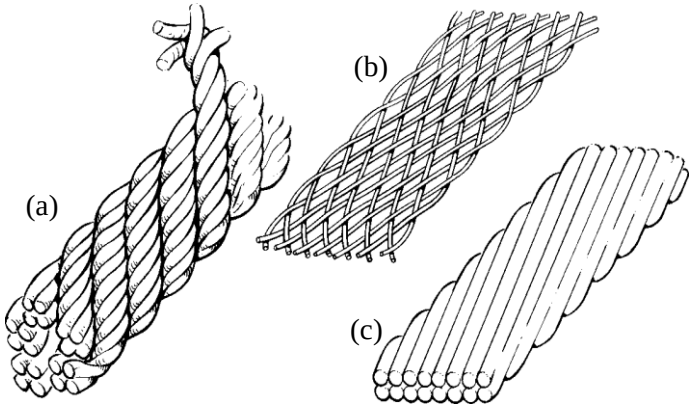


Figure 2.8. Most common superconducting cable types used in the accelerator magnets (a) rope, (b) braid, (c) Rutherford

The Rutherford cable is currently the most common superconducting cable type for accelerator magnets. The main reason for this is that it can be compacted to a high density without damaging the wires. Moreover, the dimensions of the Rutherford cable can be precisely controlled. One of the parameters representing the Rutherford cable is the strand twist-pitch defined as the minimum distance covered by a strand to firstly return to its original relative position (see Figure 2.9). Figure 2.10 depicts the cross-section of a Rutherford cable composed of 36 twisted strands. Finally, the cables are insulated with a few layers of the insulating material (Kapton, G10).

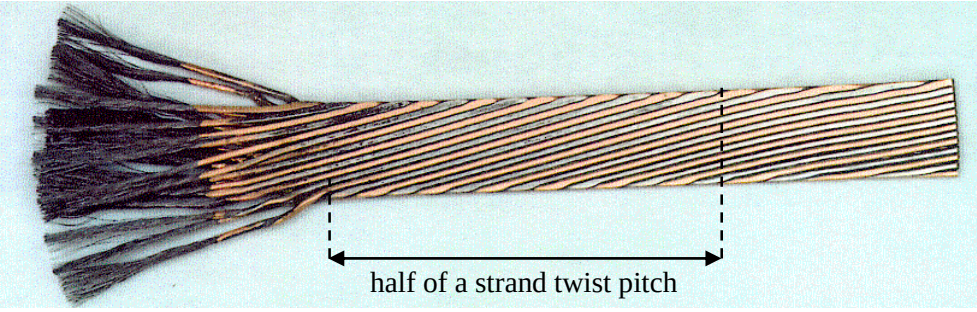


Figure 2.9. Example of the Rutherford cable [51]

Figure 2.10. Cross-section of a Rutherford cable [51]

2.6. Quench and Quench Protection

A quench is defined as an instantaneous transition from the superconducting to the normal state. This is part of the normal life of a superconducting magnet. Figure 2.11 shows the main sources of a quench in a magnet such as cooling problems, power converter problems, particle showers, mechanical stress. The integral of energy densities (a heat deposited in the coil winding pack) presented in the figure can increase the temperature in a region of the coil so that locally the superconductor characteristic point crosses the critical surface. The largest heat is usually deposited at the point where the quench starts (so called hot-spot). If the heat is conducted out of the quenched zone faster than it is generated the normal zone will shrink. In the opposite case, the quench will start to propagate in the superconducting cable.

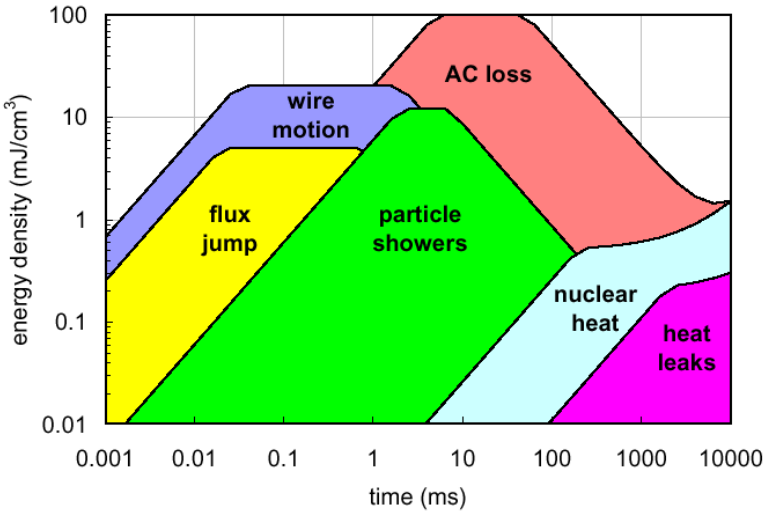


Figure 2.11. Energy density and time window of most common sources of a quench

The energy stored in the coil winding pack of a superconducting magnet is typically sufficient to melt kilos of material. For instance, each LHC main dipole magnet has inductance of 98.7 mH and nominal current of 11850 A. Thus, the stored magnetic energy is equal to about 7 MJ and enough to melt 500 kg of copper. In the case of a quench, a protection system is required in order to avoid irreversible damages due to overheating in magnets and connections between them (so called bus-bars). Figure 2.12 shows an example of the damage caused to a magnet coil by a not well protected quench.

Figure 2.12. Example of damage in the magnet after a not well protected quench

In general, all quench protection methods are designed to keep the temperature in the magnet hot-spot at a safe level by homogeneously distributing the magnet energy in the coil and quickly discharging the magnet current. The quench protection relies on the fast and reliable detection of a quench, and on fast and reliable discharge of energy stored in the magnet [37]. There are two main protection strategies. The former strategy employs devices such as by-pass diode [37, 40], parallel resistor [40], and extraction resistor [37, 40] that are routing around as much current as possible. The latter strategy relies on quench propagation in the magnet in order to homogeneously distribute the energy over the entire coil volume. Moreover, the so called quench resistance can be sufficient to quickly discharge the current in the magnet and keep the temperature in acceptable range of values.

For that purpose additional energy is needed to initiate quench in the remainder of the magnet. To date, most common approach is heating up the coil by means of Quench Heaters in order to reach temperature current sharing [39]. Furthermore, recently at CERN a new, very promising Coupling-Loss Induced Quench (CLIQ) system has been developed [13-18].

Quench protection systems can also be subdivided into the following two categories:

1. Passive systems: Passive elements which do not require active actions to start operation. Typical examples are by-pass diode or resistors in parallel to the magnet.
2. Active systems: Methods that require detecting a quench electronically, and then fire a protection system, such as energy extraction, quench heaters, or CLIQ.

The redundancy of the quench protection methods is twofold. On the one hand, particular system must be equipped in redundant elements to increase its reliability (such as multiple switches in the Energy-Extraction system). On the other hand, in some cases magnets are protected by both cold-mass diodes and quench heaters, whereas there are multiple Quench Heaters attached to the coil winding pack.

2.6.1. Parallel Resistor

Accelerator magnet can also be protected by parallel resistors (Figure 2.14). The corrector magnets operate typically at current levels significantly less than critical current, hence they do not require such sophisticated protection methods as energy extraction and quench heaters. During current ramp in the magnet, the inductive voltage across the magnet forces a current to flow through the resistor. Then, in the normal operation entire current flows through the coil. On the contrary after a quench the current flows through the parallel resistance, which value for protection should be small. However, the value of parallel resistor should not be too small due to the leakage current that during a current ramp may affect the beam operation. [40]

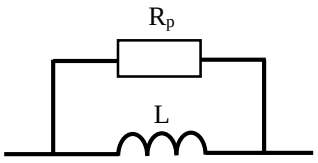


Figure 2.14. Magnet protected by a parallel resistor

2.6.2. By-pass Diode

Superconducting magnet can by-passed by diodes operating at cryogenic temperatures (so called cold diodes). For instance cold diodes are used for the protection of the LHC main dipole magnets [37]. The advantage of the diodes over parallel resistors is that during current increase they do not conduct current (up to certain ramp rate). In other words, during current ramp and normal

operation, the diodes forward voltage must be higher than inductive voltage. However, when the coil winding pack becomes resistive after the transition to the normal state, the voltage drop across the magnet builds up with the temperature increase. Thus, the diode starts conducting and the circuit current by-passes the magnet. In order to transfer out as much of current as possible, the diode resistance should be small [37]. Additionally, in some cases (for instance LHC main dipole circuit), by-pass diodes are part of a circuit with large inductance (tens of henries in the LHC). As a consequence energy dissipation in the diodes can take a very long time. Hence, in order to protect the diodes an Energy-Extraction system is applied in the magnet quench protection.

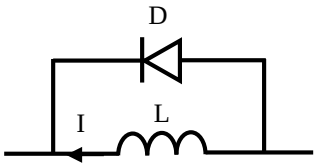


Figure 2.13. Magnet protected by a parallel diode

2.6.3. Energy Extraction

The easiest way of discharging the magnet current is to introduce a certain extraction resistance R_{EE} in series to the quenched magnet. The resulting time constant is equal to $N_{mag} \cdot L_{mag}/R_{EE}$, where N_{mag} is a number of protected magnets, L_{mag} is inductance of particular magnet [H], and R_{EE} is extraction resistance [Ω] (self-resistance of a magnet is negligible). Hence, one should select the highest economically feasible resistance value so that, the energy stored in the magnet would be discharged as quickly as possible. Nevertheless, the value of R_{EE} is limited by the safety voltage ($U_{EE}=R_{EE} \cdot I$) in the circuit.

In normal operation the extraction resistor R_{EE} is by-passed by a electromechanical switch SW_{EE} (Figure 2.15). Moreover, often a thyristor is also installed in series to the switch for obtaining a smoother opening of the switch. Once a quench is detected in one magnet in a chain or there is a problem related to the Power Converter (PC), the switch is opened and the current flows through R_{EE}. In some applications, a snubber capacitor C_{SNEE} is installed across the switch to smooth transient voltage oscillations following the switch opening.

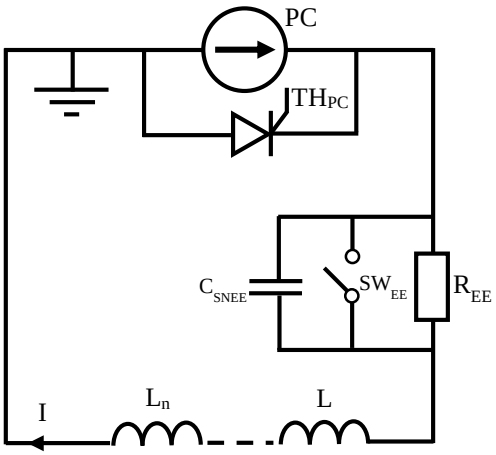


Figure 2.15. Energy extraction system diagram protecting a chain of *n* magnets

2.6.4. Quench Heaters

A quench heater (QH) system is composed of strips of stainless steel (partially plated with copper) with a polyimide foils insulation and a power supply with a normally charged capacitor bank (see Figure 2.16) [37, 39]. The copper addition to some portions of the strip allows reducing the electrical resistance of quench heater strips which can be connected in series. The quench heaters can be attached to the outer or inner layers of a magnet or to both. A triggered thyristor discharges the electrical energy stored in a capacitor bank, forcing a current through the quench heater strips. In turn, Ohmic loss is generated in the QH strips and heat diffuses from them to the magnet cables through insulation layers. Temperature increase in the cables to which quench heaters are attached eventually results in a quench. When a part of a coil becomes resistive, it also produces Joule heating that further increases the quench resistance by propagating the normal zone to the surrounding coil windings. This in turn quickly discharges the magnet current, hence limiting the hot-spot temperature in the quenching coils.

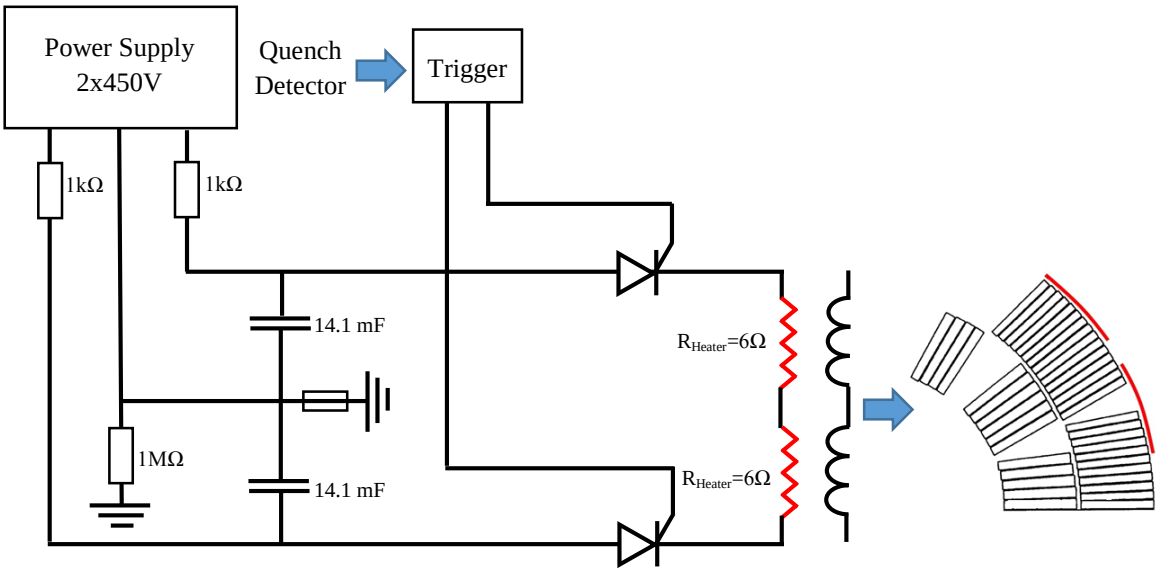


Figure 2.16. Quench heater power supply diagram, adopted from [37].

However, for some applications the quench heaters may be too slow as they rely on the heat transfer through insulation layers. Moreover, the heater pulsed from a capacitor bank has a high voltage, and creates a conflict between good thermal contact (thin insulation layer) and good electrical insulation (thick insulation layer). Thus, they are prone to electrical break-down. Another disadvantage is that they are difficult (when attached to the outer layers) or impossible (when attached between inner and outer layers) to repair.

2.6.5. CLIQ

Next generation of Nb_3Sn based high-field superconducting magnets for future particle accelerators presents a considerable technical challenge in terms of coil winding pack protection after a quench. As a consequence high energy stored in the magnet requires quick energy spreading over large portions of the magnet in a short time to limit the hot-spot temperature. The answer to those challenges can be a new quench protection mechanism called CLIQ [13-18].

Figure 2.17 illustrates the schematic of a CLIQ unit attached in the middle of a magnet. The CLIQ unit consists of a capacitor bank *C*, a floating voltage source *S*, a thyristor *TH*, and a reverse diode *D*. This diode provides a path for current oscillations during capacitive discharge. The CLIQ is connected to the magnet by two resistive current leads R_{CL1}, R_{CL2}. The capacitor bank is charged by *S* with a voltage U_0. When a quench in the magnet is detected, the thyristor is activated introducing a current I_C to be discharged through R_{CL1} and R_{CL2} in the middle of the coil. Additionally, after quench detection, the PC is switched off and by-passed by a thyristor and a diode that together allow current to flow in both directions. A detailed analytical description of the CLIQ quench mechanism and experiments results is presented in [13-18].

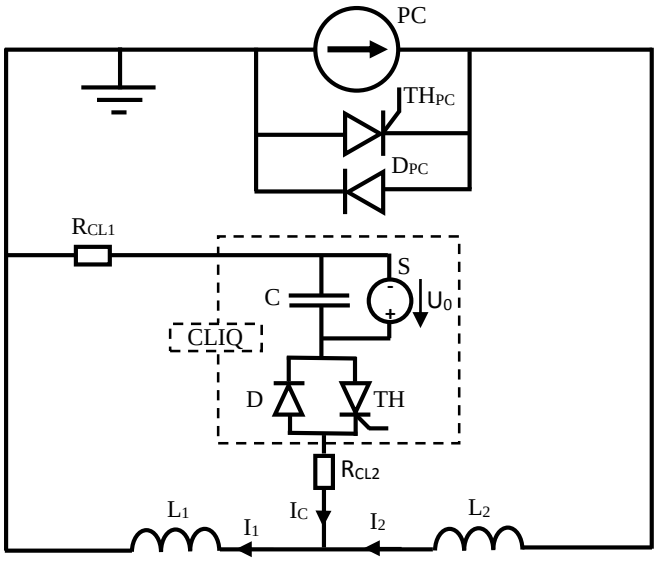


Figure 2.17. Schematic of the CLIQ system adopted from [14]

Introduced oscillating current in the coil winding pack results in a fast change of the magnetic field in the conductor. As a result, inter-filament and inter-strand coupling losses are generated in the copper matrix of the superconductor which heat up the coil and quickly provoke a quench due to enhanced temperature. Hence, the CLIQ does not rely on thermal diffusion as in a case of quench heaters but deposits energy directly in the magnet windings. As a consequence, the CLIQ system is capable of quenching large portions of the magnet in relatively short time (as compared to Quench Heaters). Sampled results of CLIQ tests and simulation results are presented in Chapter 6.

In order to obtain a safe operation of the superconducting magnets, quench protection systems must be thoroughly analysed by means of computer simulations.

3 Superconducting Magnets Modelling

3.1. Motivation

The analysis of the protection of superconducting magnets after a transition to the normal state (quench) is a key ingredient of the design and operation of superconducting magnets. Hence, the simulation of electro-thermal transients including dynamic effects in the superconducting circuits is needed in order to

- study new quench protection methods,
- design electrical circuits with superconducting magnets,
- assess the performance of existing ones.

3.2. Challenges

A convenient simulation environment is required to simulate complex electro-thermal transients accurately and in a relatively limited time. However, the simulations of superconducting magnets provide various challenges.

Firstly, the models typically comprise elements characterised by very different spatial scales, ranging from chains of magnets (kilometres), to magnetic elements (couple of meters), cables (tens of millimetres), strands (usually a fraction of millimetre), and superconducting filaments (microns). The combined effects occurring at these different scales usually need to be taken into consideration simultaneously in order to correctly reproduce electrical, magnetic, and thermal transients. Hence, a high level of flexibility is needed and the simulation environment should allow defining different magnet configurations, protection schemes, and spatial resolution.

Secondly, transient effects relevant from the quench protection perspective occur in a wide range of time constants, ranging from milliseconds (for instance inter-filament coupling current time-constant), to hundreds of milliseconds (thermal diffusion, inter-strand coupling current time-constant), to tens of seconds (for instance discharge time-constant of large circuits).

Thirdly, the model of a superconducting magnet features different physical sub-systems including electrical (electrical transients, Joule losses), magnetic, thermal (heat propagation, cooling) as well as dynamic effects (inter-filament and inter-strand coupling loss) and superconducting effects (quench).

Finally, due to wide range of available magnet types, protection schemes, and resulting configurations, the simulations must return results in a relatively short time. Namely, the model development should be accomplished in 1-2 days and simulation runs including automatic generation

of the model should be completed in less than 1 hour. Furthermore, it should be possible to run simulation sets automatically.

3.3. Modelling Approach

Finite-element modelling (FEM) is an advanced and powerful computing method of calculating sets of differential equations on the basis of domain discretization into finite number of elements. The FEM simulation software allows building both two dimensional (2-D) and three dimensional (3-D) models. For instance the COMSOL and the ANSYS simulation environments allow building a full-scale 3D model of a superconducting magnet including all aforementioned physical domains. Moreover, 3D models offer a good accuracy and are applied whenever longitudinal propagation in the magnet has a significant impact on its behaviour. However, in order to obtain a good prediction of propagation velocities, a fine spatial resolution is required. High complexity of the model and large scale of the magnets often make the FEM application inefficient due to extremely long simulation time. Moreover, FEM methods apart from good knowledge in physics require some numerical analysis skills in order to define the mesh of the components to obtain meaningful results.

Nevertheless, a full 3-D FEM model of the superconducting magnet is not the only accurate solution. In some cases, even a simple lumped inductor can be a reasonable superconducting magnet approximation. As presented in [28] a good agreement between measurements and simulation for a chain of superconducting magnets was achieved by simple lumped-element network.

Since the FEM methods in their standard form are not feasible to apply, in the thesis an alternative 2-D lumped-element modelling (LEM) approach has been employed. This approach is very efficient in terms of time required to solve the problem and it provides an acceptable accuracy for a wide range of problems including quench protection mechanisms such as passive by-pass devices, quench heater, energy extraction, and CLIQ. Lumped-element modelling, however, requires deep understanding of the physics representing the model behaviour. It requires deriving equations that describe the entire, complex problem and enables to quickly derive conclusions how the circuit performs, what are the most important parameters, how it can be adjusted in order to obtain better results. Furthermore it also requires finding convenient ways to reproduce complex behaviours with a limited number of differential-algebraic equations (DAEs). Reproducing complex problems with simple circuits of lumped elements relies on existing solvers (PSpice, Simulink, Simplorer, etc.) to couple together and solve equations from different domains. In other words, the lumped-element modelling approach provides a bigger control over how the physics is implemented, hence it can be controlled and modified.

On the contrary to 3-D models, 2-D simulations due to reduced complexity offer faster execution times with a comparable accuracy in the case of the simulation of magnets that are homogenous in the longitudinal direction. The 2-D models, however, are not suited to provide an estimate on the initial voltage increase up to a threshold resulting in the quench detection.

3.4. Lumped-Element Dynamic Electro-Thermal Model

The adopted lumped-element dynamic electro-thermal model of a superconducting magnet consists of electrical, dynamic and thermal sub-networks solved simultaneously [7,13-19]. However, if the complete thermal modelling of the magnet is not required, it is possible to define simpler components composed of conventional self and mutual inductor or non-linear inductors [28]. Hence, the electrical sub-network is always present in the model, whereas the remaining two sub-networks can be added to reproduce additional phenomena occurring in the magnet. Figure 3.1 shows a diagram representing the connections existing between the three sub-networks of the model.

The electrical sub-network is composed of power sources, inductances, resistors, and other standard electrical elements. The resistors representing the electrical resistance of a magnet are set to zero if it is in the superconducting state. The dynamic feedback from inter-filament coupling currents and inter-strand coupling currents on the transport current in the system is modelled through mutual electrical coupling. Furthermore, the magnetic field generated by the transport currents in the system is calculated. The resulting change of the local magnetic field introduces inter-filament coupling loss (IFCL) and the inter-strand coupling loss (ISCL). The heat produced by IFCL and ISCL increases the temperature of the coil modelled as a thermal mass. If the temperature in a portion of the coil is greater than the local current sharing temperature, that portion of coil transits from the superconducting to the normal state. As a result of the coil becoming resistive, an electrical resistance builds up which influences the main electrical network. The local resistance is affected by the local temperature and magnetic field due to the magneto-resistivity effect. Moreover, when the coil resistance becomes non-zero Ohmic loss is generated which further increases the temperature. The resulting temperature rise has an influence on material properties of the coil. Moreover, the heat propagates due to conductive heat transfer from turn to turn of the magnet. Furthermore, some portion of heat is dissipated to the helium bath surrounding the magnet.

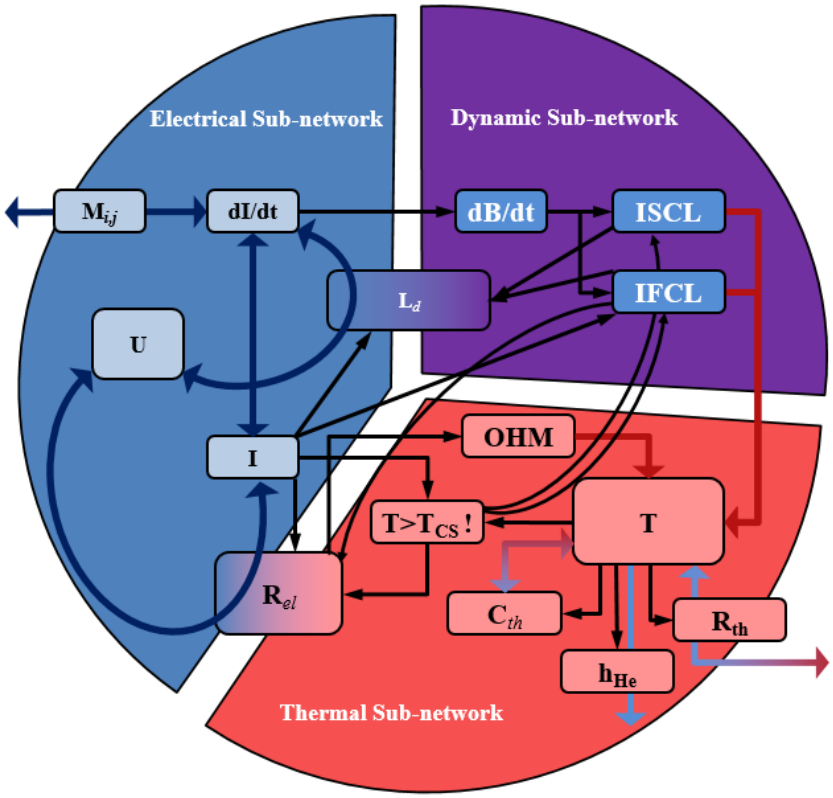


Figure 3.1. Diagram representing the three physical sub-networks constituting the adopted model of a superconducting magnet and the flow of signals between them

Let N_E denote the number of electrical parts in the model. In general, N_E can be equal to number of poles in the magnet, however it can also be multiple of and aliquot of number of poles. Moreover the order of connection between electrical parts may vary and is represented by the variable $\mathbf{c}_{order}$ which is a 1xN_E vector composed of any combination of figures from 1 to N_E. Such a feature in the model can be useful if the transport current in different parts of a magnet is not the same.

Furthermore, the thermal model of the superconducting magnet is discretized in two dimensions (x, y) assuming that within each block all physical and magnetic properties are homogeneous along the longitudinal direction. All properties are averaged over the block volume which typically consists of one cable turn or a small group of cable turns.

Each block belongs to the electrical part where it is physically located. In general, each pole may have different number of blocks. Hence, vector $\boldsymbol{n}_B$ defines the number of blocks in each electrical part.

$$\mathbf{n}_B = [n_{B,1}, n_{B,2}, ..., n_{B,Ne}], \tag{3.1}$$

where, $n_{B,e}$ in general may differ from $n_{B,j}$ for any e=1,…,N_E, and $e \neq j$. Then, the model of the magnet is composed of N_B blocks

$$N_B = \sum_{e=1}^{N_E} n_{B,e} \,. \tag{3.2}$$

Let $\mathbf{m}_B$ be a 1xN_B vector that maps the blocks to the corresponding electrical parts defined as

$$\mathbf{m}_B = \left\lfloor 1 \cdot \mathbf{I}_{n_{B,1}} \quad 2 \cdot \mathbf{I}_{n_{B,2}} \quad \cdots \quad N_E \cdot \mathbf{I}_{n_{B,N_E}} \right\rfloor. \tag{3.3}$$

As an example, Figure 3.2 shows the electrical parts and thermal blocks representing the model of a quadrupole magnet divided in 4 electrical parts (N_E=4), and whose each pole is equally divided in 36 blocks. Hence, the total number of blocks equals 144 (N_B=4·36=144). A block is a fundamental element of the model and can be represented in both electrical and thermal domains by means of the lumped elements.

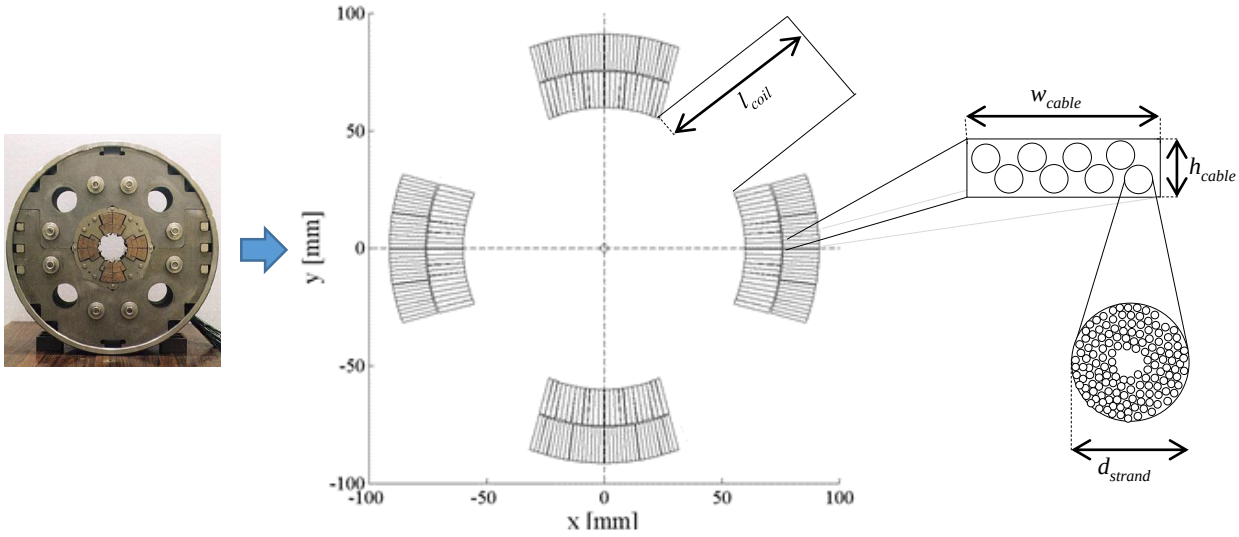


Figure 3.2. Example of two dimensional representation of a quadrupole magnet

Additionally, dynamic effects in the magnets are also taken into account by means of additional lumped elements electrically coupled to the electrical parts (Figure 3.3).

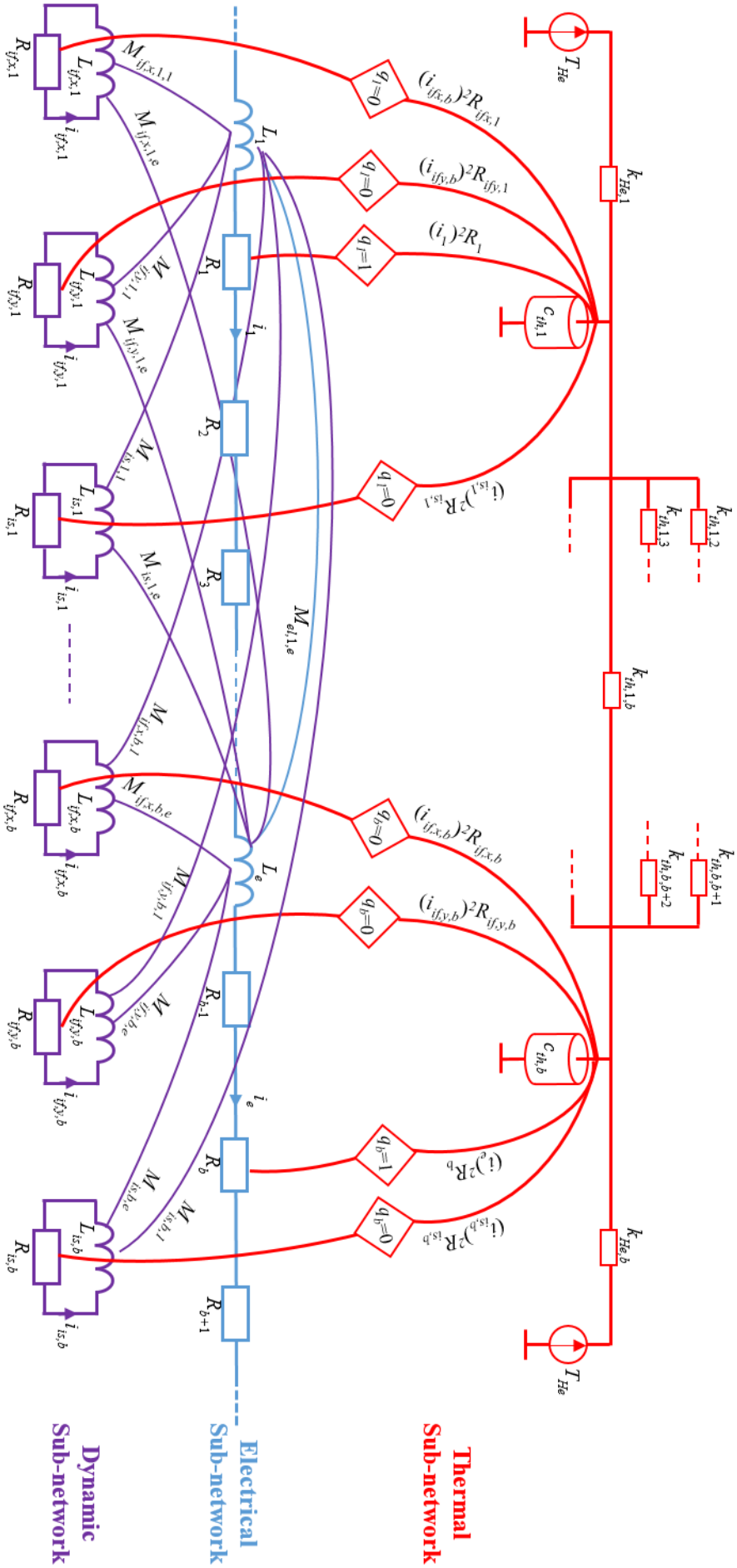


Figure 3.3. Electrical, thermal, and dynamic coupling between two arbitrarily selected sub-systems 1 and *b*.

The physical properties of the different materials of each block, namely the electrical resistivity, thermal capacity, and thermal conductivity vary with temperature, transport current, and magnetic field in the block. Furthermore, the magnetic field in each block depends on the transport current in all N_E electrical parts. The formulae used to calculate the material properties embedded in the model components are reported in the annex.

3.4.1. The Electrical Sub-network

The electrical sub-network contains conventional electrical lumped-elements such as voltage and current sources, self and mutual inductors, resistors, capacitors, diodes, thyristors, switches, ground connections, and other standard electrical elements.

The modeled superconducting magnet is represented as a series of N_E electrical parts connected according to the order defined by the $\mathbf{c}_{order}$ vector. Each electrical part e is composed of a self-inductor L_e and $n_{B,e}$ series resistors R_b which are non-zero only in the case of a transition to the normal state.

3.4.1.1. *Quench Resistor*

The quench resistor elements model the electrical resistance of the thermal blocks. In the superconducting state, the block has no resistance (0 Ω). In the normal state the average resistivity of copper $\rho_{Cu,b}$ of the b-th quench resistor is calculated based on the instantaneous average block temperature $T_b(t)$ and absolute magnetic field $B_{t,b}(t)$

$$\rho_{Cu,b}\left(T_b, B_{t,b}\right) = \begin{cases} 0, & for\ q_b = 0 \\ \left(\dfrac{c_0}{RRR_b} + \dfrac{1}{(c_1 / T_b^{\,5} + c_2 / T_b^{\,3} + c_3 / T_b)} \right) \cdot 10^{-8} + \rho_{mr,b} \cdot B_{t,b} \cdot 10^{-10}, & for\ q_b = 1 \end{cases}, \qquad [\Omega\cdot\mathrm{m}] \quad (3.4)$$

where c_0=1.7 Ω·m, c_1=2.33·10^9 K^5, c_2=9.57·10^5 K^3, c_3=163 K are the parameters of copper resistivity approximation function, $\rho_{mr,b}$ is the magneto-resistivity of copper [Ω·m/T], and RRR_b (residual resistance ratio) is the ratio between the resistivity of a material at room temperature (290 K) and just above the critical temperature (practically defined as 4 K). The quench indicator q_b can only assume the value 0 or 1. Furthermore, the model calculates the variable $t_{q,b}$ which defines the instant when block b transits to the normal state. In general superconductors are poor conductors in the normal state; hence, once a block is quenched only the resistance of the copper stabilizer is considered (3.4).

The electrical resistance R_b and the generated Ohmic loss $Q_{Ohm,b}$ in block b are calculated as follows

$$R_b = \frac{\rho_{Cu,b} \cdot l_b}{cs_b \cdot f_{Cu,b}}, \qquad [\Omega] \quad (3.5)$$

$$Q_{Ohm,b} = i_{e,b}^2 \cdot R_b, \qquad [\mathrm{W}] \quad (3.6)$$

where $l_b=l_{coil}\cdot n_{turns,b}$ is the total length of the conductor [m], l_{coil} is the length of the magnet coil [m], $n_{turns,b}$ the number of turns, cs_b the cross-section of the cable [m^2], $f_{Cu,b}$ the fraction of copper in block b, and $i_{e,b}$ is the transport current flowing in the electrical part where block b is located [A].

The superconductor transits to the normal state if its characteristic point moves outside of the critical surface defined by three variables: temperature, current density, and magnetic field (see Section 2.4). From the modelling perspective it is convenient to express this condition as one variable in terms of the remaining two. The transition between superconducting and normal state is assumed to be instantaneous.

The value of the quench indicator q_b used in (3.4) is determined by the instantaneous conditions in block b. For Nb-Ti superconductors the transition to the normal state occurs when the following condition is met

$$q_b = \begin{cases} 1, \; for\, T_b > T_{cs,b} \\ 0, \; for\, T_b \le T_{cs,b} \end{cases}, \tag{3.7}$$

where $T_{cs,b}$ is the current-sharing temperature in block b [K]. The calculation of quench condition for Nb-Ti requires three steps (3.8-3.10). Firstly, the critical temperature in block b is calculated as

$$T_{C,b} = T_{C0}\cdot(1-B_{t,b}/B_{C2})^{0.59}, \quad \text{[K]} \tag{3.8}$$

where B_{c2}=14.5 T is the critical magnetic field of Nb-Ti at zero temperature and current density and T_{c0}=9.2 K is its critical temperature at zero magnetic field and current density [30]. Secondly, the critical current in block b is calculated as

$$I_{C,b} = c_{1,Ic,b} + c_{2,Ic,b}\cdot B_{t,b}, \quad \text{[A]} \tag{3.9}$$

where $c_{1,Ic,b}$ and $c_{2,Ic,b}$ are coefficients approximating the critical current at zero temperature in block b expressed in [A] and [A/T], respectively. Finally, the current-sharing temperature in block b is calculated as

$$T_{cs,b} = T_{C,b}\cdot(1-I/I_{C,b}). \quad \text{[K]} \tag{3.10}$$

For Nb_3Sn superconductors, the transition to the normal state in block b occurs when

$$q_b = \begin{cases} 1, \; for\, I_b > I_{cs,b} \\ 0, \; for\, I_b \le I_{cs,b} \end{cases}. \tag{3.11}$$

The critical current of block b is calculated as follows

$$I_{cs,b} = J_{c,SC,b}\cdot cs_{block,b}\cdot f_{SC,b}, \quad \text{[A]} \tag{3.12}$$

where $f_{SC,b}$ is the fraction of superconductor in block b and $J_{c,SC,b}$ is the critical current density in block b [A/m^2] defined as

$$J_{c,SC,b} = \frac{J_{c,Nb_3Sn0}}{\sqrt{\max\left(10^{-3}, B_{t,b}\right)}} \cdot \left[1 - \min\left(1, \frac{B_{t,b}}{\max(10^{-3}, B_{c2})}\right)\right]^2 \cdot \left[1 - \min\left(1, \frac{T_b}{T_{c0}}\right)\right]^2, \quad [\text{A/m}^2] \ (3.13)$$

where $J_{c,Nb3Sn0}$ is the critical current density of the particular Nb_3Sn superconducting cable composing block b and T_{c0}=18 K and B_{c20}=29 T are the critical temperature and magnetic field of Nb_3Sn, respectively [31]. Finally, the critical magnetic field B_{c2} used in (3.12) is derived in the following way

$$B_{c2} = B_{c20} \cdot \left[1 - \left(T_b / T_{c0}\right)^2\right] \cdot \left[1 - 0.31 \cdot \left(T_b / T_{c0}\right)^2 \left(1 - 1.77 \cdot \log(T_b / T_{c0})\right)\right]. \quad [\text{T}] \quad (3.14)$$

In particular applications, it may be interesting to impose the transition to the normal state of some of all blocks. For example, the performance of an ideal system transferring the whole magnet to the normal state instantaneously may be useful to simulate. Thus, the model allows the user to artificially change the value of each q_b at desired time.

3.4.1.2. *Magnetic Field Calculation*

In order to calculate quench margins as well as the copper magneto-resistivity, the absolute magnetic field in each block b must be determined. In the adopted two-dimension approximation, the absolute value of the total magnetic field in block b is

$$B_{t,b} = \sqrt{B_{tx,b}^2 + B_{ty,b}^2}, \quad [\text{T}] \quad (3.15)$$

where $B_{tx,b}$ and $B_{ty,b}$ are the total magnetic field in the x and y direction, respectively, averaged over the volume of block b [T]. In turn, each component of $\mathbf{B}_{t,b}$ is calculated as the sum of an average applied magnetic field $\mathbf{B}_{a,b}$, only determined by the transport current in the N_E electoral parts, and an average induced magnetic field $\mathbf{B}_{i,b}$, opposing to the applied field, generated by the presence of inter-filament coupling currents [19, 32]:

$$B_{tx,b} = B_{ax,b} + B_{ifx,b}, \quad [\text{T}] \quad (3.16)$$

$$B_{ty,b} = B_{ay,b} + B_{ify,b}. \quad [\text{T}] \quad (3.17)$$

The induced field produced by inter-strand coupling currents is here neglected. Let $\mathbf{i}_{el}=[i_1, i_2, \ldots, i_{NE}]^T$ be the vector of the transport currents in each electrical part [A]. Then, the applied magnetic field in the x and y direction, is calculated as follows

$$B_{ax,b} = \mathbf{f}_{mx,b} \cdot \mathbf{i}_{el}, \quad [\text{T}] \quad (3.18)$$

$$B_{ay,b} = \mathbf{f}_{my,b} \cdot \mathbf{i}_{el}, \quad [\text{T}] \quad (3.19)$$

where $\mathbf{f}_{mx,b}$ and $\mathbf{f}_{my,b}$ are the b-th rows of N_BxN_E $\mathbf{F}_{\mathrm{m,x}}$ and $\mathbf{F}_{\mathrm{m,y}}$ magnetic-field transfer function matrices in the x and y direction, respectively [T/A]. Each element (b,e) of these matrices is defined as the ratio between the average magnetic field generated in the strands of block b by a current flowing in the electrical part e. The magnetic field is calculated using dedicated simulation software (ROXIE, Soleno, Opera, etc.). Moreover, the quench margins are calculated using the maximum magnetic field in each block instead of its average value; hence, the magnetic field calculated in equations (3.8, 3.9, and 3.13) used two $\mathbf{F}_{mx,marg}$ and $\mathbf{F}_{my,marg}$ matrices different from the ones used in (3.18-3.19).

The procedure for calculating the average induced magnetic field $B_{ix,b}$ and $B_{iy,b}$ is described in the next section detailing the dynamic sub-network of the model.

3.4.2. The Dynamic Sub-network

The complex interaction between the macroscopic magnetic elements in the circuit and the coupling currents generated between superconducting filaments and strands are reproduced with a dedicated sub-networks of RL loops mutually electrically coupled with the self-inductors of the electric sub-network [7].

Thus, in addition to the self and mutual inductance between the N_E electrical parts in the electrical sub-network, 3 sets of N_B loops are present in the model representing dynamic effects due to inter-filament coupling currents in the x and y direction and due to inter-strand coupling currents in the direction perpendicular to the cable broad face, respectively. The current flowing in each RL loop represents an equivalent coupling current in block b. The power dissipated in the loop resistors represents the coupling loss generated in the corresponding blocks. The ratio between the inductors and resistors of the loops corresponds to the characteristic time constants associated to the development of inter-filament and inter-strand coupling currents [19, 32]. The mutual coupling between the loops and the self-inductors in the electrical sub-network models the dynamic interaction between the magnetic elements in the circuit and the coupling-current effects in their coils. Finally, the currents flowing in the dynamic loops are proportional to the induced magnetic field introduced in (3.15-3.16), thus they also influence the resistivity and the quench margins in each block.

3.4.2.1. *Coupling Between Electrical and Dynamic Sub-network*

The coupling between the electrical and dynamic sub-networks of the model is described by

$$\mathbf{u}(t) = \mathbf{M} \cdot \frac{d\mathbf{i}(t)}{dt} + \mathbf{R} \cdot \mathbf{i}, \qquad \text{[V]} \quad (3.20)$$

where $\mathbf{u}(t)$ is a voltage vector of (N_E+3N_B)x1 elements [V], $\mathbf{i}(t)$ is a current vector of (N_E+3N_B)x1 elements, [A], $\mathbf{M}$ is a symmetric, mutual coupling inductance matrix of the system of (N_E+3N_B)^2

elements [H], and **R** is a block, diagonal, equivalent resistance matrix of $(N_E+3N_B)^2$ elements [Ω]. The vector $\mathbf{u}(t)$ is composed of the following parts

$$\mathbf{u}(t)=\left[\mathbf{u}_{el}(t)\quad \mathbf{u}_{if,x}(t)\quad \mathbf{u}_{if,y}(t)\quad \mathbf{u}_{is}(t)\right]^T, \qquad \text{[V]} \quad (3.21)$$

where the $1xN_E$ elements of $\mathbf{u}_{el}(t)=[u_1(t), u_2(t), \ldots, u_{NE}(t)]^T$ represent the voltage drops across the self-inductors and quench resistors of the N_E electrical parts and the $1xN_B$ elements of $\mathbf{u}_{if,x}(t)=[u_{if,x,1}(t), u_{if,x,2}(t), \ldots, u_{if,x,NB}(t)]^T$, $\mathbf{u}_{if,y}(t)=[u_{if,y,1}(t), u_{if,y,2}(t), \ldots, u_{if,y,NB}(t)]^T$, and $\mathbf{u}_{is}(t) = [u_{is,1}(t), u_{is,2}(t), \ldots, u_{is,NB}(t)]^T$ represent the voltage drops across the $3N_B$ loops modelling the inter-filament effects in the *x* and *y* direction and the inter-strand effects, respectively.

The current vector in the system is composed of the N_E transport currents ($\mathbf{i}_{el}$) and of the $3N_B$ equivalent inter-filament and inter-strand coupling currents ($\mathbf{i}_{if,x}$, $\mathbf{i}_{if,y}$, and $\mathbf{i}_{is}$),

$$\mathbf{i}(t)=\left[\mathbf{i}_{el}(t)\quad \mathbf{i}_{if,x}(t)\quad \mathbf{i}_{if,y}(t)\quad \mathbf{i}_{is}(t)\right]^T. \qquad \text{[A]} \quad (3.22)$$

Finally, the mutual inductance matrix **M** is defined as

$$\mathbf{M}=\begin{bmatrix} \mathbf{M}_{el} & \mathbf{M}_{if,x}^T & \mathbf{M}_{if,y}^T & \mathbf{M}_{is}^T \\ \mathbf{M}_{if,x} & \mathbf{L}_{if,x} & \mathbf{0} & \mathbf{0} \\ \mathbf{M}_{if,y} & \mathbf{0} & \mathbf{L}_{if,x} & \mathbf{0} \\ \mathbf{M}_{is} & \mathbf{0} & \mathbf{0} & \mathbf{L}_{is} \end{bmatrix}. \qquad \text{[H]} \quad (3.23)$$

The various sections of the **M** matrix are described separately. $\mathbf{M}_{el}$ is the N_ExN_E matrix of self and mutual inductance of the electrical parts in the circuit,

$$\mathbf{M}_{el}=\begin{bmatrix} L_1 & M_{1,2} & \cdots & M_{1,NE} \\ M_{2,1} & L_2 & \cdots & M_{2,NE} \\ \vdots & \vdots & \ddots & \vdots \\ M_{NE,1} & M_{NE,2} & \cdots & L_{NE} \end{bmatrix}\cdot l_{magnetic}, \qquad \text{[H]} \quad (3.24)$$

where $l_{magnetic}$ is the magnetic length of the coil [m].

$\mathbf{M}_{if,x}$, $\mathbf{M}_{if,y}$, and $\mathbf{M}_{is}$ are N_BxN_E mutual coupling inductance matrices between the elements of the electrical sub-network and inter-filament loops in the *x* and *y* direction and inter-strand loops, respectively [H]. Finally, $\mathbf{L}_{if,x}$, $\mathbf{L}_{if,y}$, $\mathbf{L}_{is}$ are N_BxN_B diagonal, self-inductance matrices of the inter-filament and inter-strand loops [H]. The analytical expressions applied to calculate elements of $\mathbf{M}_{if,x}$, $\mathbf{M}_{if,y}$, $\mathbf{M}_{is}$, $\mathbf{L}_{if,x}$, $\mathbf{L}_{if,y}$, $\mathbf{L}_{is}$ have been derived from [7].

Finally, **R** is a diagonal matrix defined as

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{el} & 0 & 0 & 0 \\ 0 & \mathbf{R}_{if,x} & 0 & 0 \\ 0 & 0 & \mathbf{R}_{if,y} & 0 \\ 0 & 0 & 0 & \mathbf{R}_{is} \end{bmatrix}. \quad [\Omega] \quad (3.25)$$

Each element of the N_Ex1 vector $\mathbf{R}_{el}$ represents the sum of the electrical resistance in all blocks in the resistive state located in the corresponding electrical part:

$$R_{el,e} = \sum_{b:\mathbf{m}_B = e} R_b \text{ , for } e=1,\ldots, N_E. \quad [\Omega] \quad (3.26)$$

$\mathbf{R}_{if,x}$, $\mathbf{R}_{if,y}$, and $\mathbf{R}_{is}$ are N_Bx1 vectors of inter-filament equivalent resistance in the x and y direction and inter-strand equivalent resistance, respectively. The elements of those vectors are calculated according to appropriate expressions reproducing the dynamic behaviour of the superconducting magnet [7].

3.4.2.2. *Equivalent Inter-Filament Resistance*

The inter-filament coupling loss generated in the x and y directions in block b is calculated as

$$Q_{if,x,b} = i_{if,x,b}^2 \cdot R_{if,x,b} \cdot q_b \,, \quad [\text{W}] \quad (3.27)$$

$$Q_{if,y,b} = i_{if,y,b}^2 \cdot R_{if,y,b} \cdot q_b \,, \quad [\text{W}] \quad (3.28)$$

As one can notice in equations 3.27 and 3.28 the coupling losses are generated as long as the block b is in the superconducting state. However, those equations can be modified by removing the q_b term and account for the fact that decaying inter-filament coupling currents can still generate losses.

The average magnetic field induced in block b by inter-filament coupling currents in the x and y direction is defined as

$$B_{if,x,b} = -\frac{\mu_0}{2d_b^*} \cdot i_{if,x,b} \,, \quad [\text{T}] \quad (3.29)$$

$$B_{if,y,b} = -\frac{\mu_0}{2d_b^*} \cdot i_{if,y,b} \,, \quad [\text{T}] \quad (3.30)$$

with

$$d_b^* = d_{strand,b} \cdot f_{ds,b} \,, \quad [\text{m}] \quad (3.31)$$

where $d_{strand,b}$ is the diameter of the strands in the conductor of block b [m], $f_{ds,b}$ is an effective strand diameter factor, and $\mu_0 = 4\pi \cdot 10^{-7}$ H/m is the magnetic permeability of vacuum. These induced magnetic fields are used for the calculation of the total magnetic field in equations (3.16, 3.17).

Finally, the inter-filament coupling currents disappear once the superconductor is transferred to the normal state. Hence, if a thermal block *b* quenches its corresponding values of $\mathbf{R}_{if,x}$ and $\mathbf{R}_{if,y}$ are increased exponentially up to a certain resistance R_{max} which assures $i_{if,x,b} \approx 0$ and $i_{if,y,b} \approx 0$:

$$R_{if,x,b} = R_{if,y,b} = \begin{cases} R_{if,x,b}, & for\, t \le t_{q,b} \\ \min\left(R_{if,x,b}(t = t_{q,b}) \cdot e^{-\frac{(t - t_{q,b})}{\tau_{if,q}}}, R_{\max} \right), & for\, t > t_{q,b} \end{cases}, \quad [\Omega] \quad (3.32)$$

where $\tau_{if,q}$ is a time constant representing time when the current between superconducting filaments decays after quench and only transport current flows through the copper stabiliser [s].

3.4.2.3. *Equivalent Inter-Strand Resistance*

Analogously, the inter-strand coupling loss in block *b* is calculated as

$$Q_{is,b} = i_{is,b}^2 \cdot R_{is,b} \cdot q_b . \quad [W] \quad (3.33)$$

Furthermore, the inter-strand coupling currents disappear once the superconductor is transferred to the normal state, hence the resistance $R_{is,b}$ of thermal blocks in the normal state is increased exponentially:

$$R_{is,b} = \begin{cases} R_{is,b}, & for\, t \le t_{q,b} \\ \min\left(R_{is,b}\left(t = t_{q,b}\right) \cdot e^{-\frac{(t - t_{q,b})}{\tau_{is,q}}}, R_{\max} \right), & for\, t > t_{q,b} \end{cases}, \quad [\Omega] \quad (3.34)$$

where $\tau_{is,q}$ is a time constant representing time when the current between strands decays after quench and only transport current flows through the copper stabiliser [s].

3.4.2.4. *Mutual Inductor Stability Analysis*

While modelling high-order dynamic components, the important aspect to be considered is the stability analysis. The stability should be analysed in order to verify whether the simulation including the mutual inductor **M** will converge or not. In fact, the Simulink (as other network solvers) does not analyse the stability of the model before starting the simulation. Control theory, however, provides many methods for assessing the stability of linear, time-invariant systems [33].

The mutual inductor is stable if the magnetic energy storage is non-negative for any set of currents. In order to define an analytical formula for mutual inductor stability analysis, the coupling coefficient matrix must be introduced. Let **M** be a *n*x*n* symmetric, mutual inductance matrix and **K** be *n*x*n* symmetric, coupling coefficient matrix. The elements $K_{i,j}$ are calculated as follows

$$K_{i,j} = \frac{M_{i,j}}{\sqrt{M_{i,i}M_{j,j}}}, \text{ for any } i=1,\ldots,n, \text{ and } j=1, \ldots,n. \quad (3.35)$$

The first condition to be fulfilled for the mutual inductor to be stable is the following relation $|K_{i,j}|\leq 1$, for any $i=1,\ldots, n$, and $j=1, \ldots, n$. However, this relation is only a necessary condition for the stability of the mutual inductor (it can be shown that for dim(**K**)=2x2 it is also a sufficient condition). Therefore, the necessary and sufficient conditions that any inductance matrix of any order is positive or semi-definite, hence stable, can be formulated in terms of its eigenvalues [37]. The eigenvalues of matrix **K** are calculated as the roots of its characteristic polynomial. Thus,

$$P(\lambda) = \det(\mathbf{K} - \lambda \mathbf{I}_n) = (-1)^n(\lambda - \lambda_1)(\lambda - \lambda_2)\ldots(\lambda - \lambda_n) \quad (3.36)$$

where $\mathbf{I}_n$ – is a unit matrix of order n, λ_i for any $i = 1, \ldots, n$ are eigenvalues of **K**.
Finally, the following theorem holds [34]:

If and only if **K** *is real, symmetric, and positive definite, then the eigenvalues* λ_i *are real and* $\lambda_i > 0$.

Hence, this theorem provides a tool for assessing the physical meaning of a mutual inductance matrix, and may be effectively applied in order to provide information about simulation convergence. Moreover, most popular mathematical computing environments are equipped with the procedures to calculate eigenvalues. However, the proposed algorithm does not provide any information about which element, or elements, of **M** lead to instability.

3.4.3. The Thermal Sub-network

The thermal sub-network of the model contains both passive (thermal capacitances, conductive heat transfer blocks) and active devices (heat sources, temperature sources). It is composed of elements referring to the N_B thermal blocks and to additional blocks modelling quench heaters attached to the coil.

The thermal balance of the system composed of N_B thermal blocks reads

$$\mathbf{Q}_{CL} + \mathbf{Q}_{Ohm} + \mathbf{Q}_{ex} + \mathbf{Q}_{He} = \frac{d\left[\mathbf{C}(\mathbf{T})\cdot(\mathbf{T} - T_{He})\right]}{dt}, \quad \text{[W]} \quad (3.37)$$

where $\mathbf{Q}_{CL}=\mathbf{Q}_{if,x}+\mathbf{Q}_{if,y}+\mathbf{Q}_{is}$ is the generated coupling loss vector of N_Bx1 elements defined in equations 3.27, 3.28, and 3.33 [W], $\mathbf{Q}_{Ohm,b}$ is the generated Ohmic loss vector defined in equation (3.5) [W], $\mathbf{Q}_{ex}$ is the heat exchanged with other blocks vector defined in equation (3.38) [W], $\mathbf{Q}_{He}$ is a vector of heat flows from blocks to the helium bath defined in equation (3.39) [W], **C** is the vector of the total thermal capacitances of the blocks defined in (3.40) [J/K], **T** is the average temperature in the block of N_Bx1 elements [K], and T_{He} is the temperature of the helium bath [K], assumed to be time-invariant and constant all around the magnet.

By assuming that the heat diffusion occurs in one direction and that one layer of insulation is present, the elements of vector $\mathbf{Q}_{ex}$ can be approximated as

$$Q_{ex,b} = -\sum_{h=1}^{N_{bh}} \frac{k_b(T_b)\cdot k_h(T_h)}{k_b(T_b)\cdot s_h + k_h(T_h)\cdot s_b} \cdot A_{b,h} \cdot (T_b - T_h), \quad \text{[W]} \quad (3.38)$$

where N_{bh} is the number of blocks that exchange heat with b, $A_{b,h}$ is the contact area between b and h [m^2], s_b and s_h are the insulation thickness of the conductor in block b and h [m], k_b and k_h are the thermal conductivity of the insulation material in block b and h [W/(m·K)], and T_b and T_h is the average temperature in block b and h, respectively [K]. The thermal barrier due to the conductor material is neglected because it is much smaller than that due to the insulation layer. Moreover, some of the N_{bh} blocks present in equation 3.36 can be blocks modelling quench heaters physically adjacent to block b.

The elements of the vector $\mathbf{Q}_{He}$ describing heat dissipated to the helium bath is calculated as

$$Q_{He,b} = -k_{b,He}(T_b, T_{He})\cdot A_{b,He} \cdot (T_b - T_{He}), \quad \text{[W]} \quad (3.39)$$

where $k_{b,He}$ is the heat transfer coefficient to helium in block b [W/(m^2·K)] and $A_{b,He}$ is the area of block b adjacent to the helium bath [m^2].

Let $N_{m,b}$ be the number of materials composing block b. The elements of the thermal capacitance vector $\mathbf{C}$ can be calculated as the sum of the thermal capacitance of each material composing the conductor of each block, weighted over their respective volume. Thus,

$$C_b(T_b) = V_b \cdot \sum_{m=1}^{N_{m,b}} \left[c_{m,b}(T_b) \cdot f_{m.b} \right], \quad \text{[J/K]} \quad (3.40)$$

where $V_b = l_b \cdot w_b \cdot h_b$ is the volume of block b [m^3], w_b the cable width [m], h_b the cable height [m], $c_{m,b}$ the volumetric heat capacity of material m [J/m^3/K], and $f_{m,b}$ is the fraction of material m in block b. Typically, the materials included in the calculation of the block heat capacity are the superconductor, the copper stabilizer, the insulation material, and liquid helium in the case the modelled coil is helium-impregnated.

The coupling between the thermal and electrical sub-networks is twofold. Firstly, the Ohmic loss $Q_{Ohm,b}$ in block b depends on the transport current $i_{e,b}$ flowing in the electrical part e where block b is located (see equation 3.5). Secondly, the resistance of each electrical part e is the sum of all the $n_{B,e}$ blocks contained in e (see equation 3.26).

3.5. Choice of Simulation Software

Different dimensions of applied lumped-element blocks make it difficult to develop a simulation framework and require a unified, formal description of the problem. So far, there were many attempts to build simulation environments for superconducting magnets. For that purpose the modellers have developed their own simulation software as well as used commercial environments

to build models of superconducting magnets. Some of the most recent simulation software applied to model superconducting circuits are listed below:

- ROXIE (Routine for the Optimization of coil X-sections, Inverse calculations and Endspacer design) was developed by Stephan Russenschuck at CERN in order to perform electromagnetic simulation and optimization of accelerator magnets [8]. The ROXIE combines powerful geometry macros with the numerical accuracy of FEM coupling, hundreds of design variables and objective parameters, powerful optimization algorithms, and CAD/CAM interfaces [10]. Furthermore, a 2-D thermal model weakly coupled with an electric network and an electromagnetic solver which includes coupling are part of the software. However, the coupling current module does not include a direct feedback from inter-filament and inter-strand equivalent currents on the main electrical part. The ROXIE is capable of performing quench simulation in superconducting magnets as well [9]. Additionally, the ROXIE allows performing 2-D+1 simulations and calculating magnetic field transfer functions that are required for the lumped-element models.
- OrCAD PSpice is a very powerful tool in terms of available components in the library and wealth of simulation scenarios (time and frequency domain, parametric sweep, worst-case scenario, sensitivity analysis, Monte Carlo analysis, etc.). Furthermore, OrCAD PSpice developed by Cadence Company is also capable of designing Printed Circuit Boards (PCB), hence it offers a simulation environment to model, test and design electrical circuit [11]. The thermal networks can be represented on the basis of electro-thermal analogies. Furthermore, the software allows defining analog-behaviour components whose electrical behaviour depends on user-defined functions. However, the limitation of this software when including a large number of analog-behaviour components is the lack of ability to create and modify models automatically. Hence, every new model introduces a need to copy and paste lots of subcomponents and modify their parameters and/or equations. Naturally, such “hand-made” approach can be applied to some extent as long as it is economically efficient. Nevertheless, the models created this way are prone to errors and difficult to maintain and reuse as each modeller has its own way of model development. Additionally, in the case of OrCAD PSpice the parametric sweep functionality is limited only to a few variables and cannot be executed within a large set of simulations.
- MATLAB&Simulink is a sophisticated numerical analysis environment that enables the user performing complex scientific calculations and simulations. Moreover, MATLAB and the graphical companion tool, Simulink, are seamlessly connected [5]. In other words, simulations executed in Simulink can exchange variables with MATLAB and MATLAB is capable of controlling Simulink simulations [5,6]. Although the thermal effects can be efficiently modelled by means of analog electrical elements, the Simscape library of the Simulink makes it possible

to use separately electrical and thermal domains [12]. The Simscape library offers the ability to model a large variety of physical domains. In addition, it automatically handles a domain check, i.e. it warns the user if the schematic has unphysical connections between different domains. The biggest advantage of Simscape is the ability to automatically write code describing the component behaviour. It allows including non-linear material properties, including non-linear time-variant differential equations. In general, apart from extending existing domains and components it is also possible to create new domains only if particular DAEs are available. Moreover, it is also possible to create and modify Simulink components by executing specific MATLAB functions, thus automatizing model design. Hence, one of the key OrCAD PSpice limitations may be efficiently resolved. Finally, MATLAB offers an extensive amount of built-in numerical analysis functions that can be used to post-process the obtained simulation results.

4 Quench Simulation Framework Architecture

The simulation software should enable modellers to create low- and high-scale models with relatively similar effort. Additionally, it should be easy to modify the model structure depending on current test scenario. Easiness of use should be comparable for an experienced engineer, as well as new users. In other words the simulation environment should be convenient in use so that the user may focus only on parameters and structure of the model definition and analysis of the obtained results. The internal mechanism dealing with actual model design and calculation in the network solver should be hidden and treated as a so called "black-box". As a consequence, developed framework should not require any special skills from the modeller (such as knowledge of advanced programming techniques, sound experience in simulation software, etc.). Once those limitations were reached the TE-MPE-PE section at CERN decided to develop a new Quench Simulation Framework (QSF) in MATLAB R2013a and Simulink 8.1 (R2013a). Nevertheless, a strong foundation for the new framework was the existing knowledge already implemented and validated in the PSpice models.

4.1. Requirements

The requirements facing the new Quench Simulation Framework were primarily grouped into two categories: network solver requirements (implementation of the electro-thermal physical equations) and framework requirements (software related).

- The initial solver-related requirement was to ensure full agreement between the same superconducting magnet models developed in the existing and in the new simulation environments. Furthermore, another important aspect to take into consideration was easy development of new components.
- The simulation framework requirements included a list of high level features that were missing in the PSpice models and can greatly improve user experience. As a consequence of successful application of framework requirements the user must be able to build models and analyse results easily and efficiently.

The first requirement has been satisfied by building models of the MQXC2 superconducting magnet in Simulink using components written in the Simscape language. The Simscape language provides syntax to efficiently describe lumped-element components behaviour, i.e. define physical domains, list of parameters as well as explicitly implement nonlinear, time-variant differential equations in the code. Furthermore, the Simscape environment automatically handles physical unit conversion and validates whether physical units in the equation commensurate. Additionally, the list of available physical domains in Simscape is far wider than the one in OrCAD PSpice. The modeller can select among electrical, thermal, magnetic, mechanic and hydraulic domains as well as define

new ones. The new domain has to define variables needed to exchange energy and data with other blocks (so called through and across variables). For instance in the electrical domain the current is a "through" variable, whereas the voltage is an "across" variable. To sum up, the thermal equations are implemented explicitly without the need to apply the analogy between electrical and thermal domains. Another important feature of the Simscape object-oriented language is that it is based on the MATLAB programming language [12].

Taking into account all aforementioned assumptions resulting from the applied modelling approach as well as prior modelling experience, the following key features of the QSF have been derived:

1. Library of reusable components – most fundamental element of each simulation framework is a set of available building blocks. In case of simulations based on lumped-elements usually one component is reused many times in the model. The only difference between copies is the list of block parameters. Hence, it is profitable to develop a library of components as it will increase model robustness to various errors. Once tested, each library component may be reused many times in different simulations. Additionally, any modification applied to the component will automatically update among all copies. As a consequence this approach makes it much faster to fix errors as well as implement new concepts. Another important aspect is the ability to share libraries as each component has defined list of parameters, inputs and outputs.
2. Modularity – modularity should be obtained both in terms of the library of modelling blocks and the main application structure. In general each module should perform only one specified functionality, i.e. be self-sufficient and have interface describing required inputs and expected outputs. Moreover, each library component should have modular and hierarchical structure. The electro-thermal equations already provide guidance on how to build coherent blocks. A good example of components modularity at superconducting magnet model is a quench resistor (analysed in depth in the remainder of this chapter) that consists of components calculating magnetic field based on currents in the system, magnetic field map, and induced magnetic field connected to resistor modelling superconducting cable. The magnetic field block and quench resistance are not able to work separately, but together they are capable of calculating quench limits and resistance in the superconducting cable. However, the same magnetic field block can be reapplied in the hot-spot temperature calculation block. Finally, software should have also a modular structure as it simplifies code development, simplifies application testing and opens an easy way to extend the application. Well-designed architecture with decentralized modules provides a high level of flexibility on how the elements can interact with each other.

3. Scalability – proposed modelling approach assumes that the magnet is divided into N_E electrical parts, whereas each electrical part is subdivided into N_B blocks each with homogeneous physical properties. Since N_E and N_B are *a priori* unknown and may vary from magnet to magnet as well as from one simulation scenario to another, the QSF should be capable of building the model in every case. Furthermore, at an early stage of the design process simple models are necessary to make some major conceptual design decisions. Afterwards selected parts of the model can be more detailed in order to further investigate certain phenomena (for example a part where quench starts modelled with a finer spatial resolution). When creating lumped-element models it is important to choose a reasonable spatial resolution [35]. In fact, on the one hand, a too coarse spatial resolution may affect the accuracy of the simulation; on the other hand, at some point further blocks division does not improve obtained results but only increases computational costs and memory consumption. Although often small models with coarse spatial resolution do not provide meaningful results they can be employed in the testing routines, where reduced complexity and faster execution times are essential. Since the models presented in this thesis are assumed to have only two dimensions, it should be also possible to extend them in the longitudinal direction, i.e. by splitting the coil into some blocks in the third direction. The modelling environment should be created in a structure that makes easy a future extension of the model in this direction. Finally, in terms of superconducting magnets modelling it should be achievable to simulate a stand-alone magnet, a twin aperture magnets as well the chain of magnets (for example the RB circuit of the LHC section). As a consequence of the increasing number of magnets in the circuit, the ability to reduce their complexity (decreasing accuracy) becomes important.

4. Convenient User Interface – the user should be able to use the framework without understanding how all features have been internally implemented. In order to ensure easiness of framework use, it should be equipped with simple, and self-explaining input/output file types, and especially an intuitive Graphical User Interface (GUI). The GUI should also provide guidance in model design in order to ensure its correctness before the start of the simulation.

5. Automatic Model Design – in order to simulate various magnet protection schemes with varying parameters all blocks should be placed, parameterised and connected on the schematic automatically based on one input file. This feature opens a new way of performing parametric sweep – once implemented it may be used to simulate different model structures, for example changing the electrical order of the magnet electrical parts, CLIQ connections, etc., and/or varying parameters in a single simulation set. Hence, the parametric sweep must not be restricted to modification of parameters of model with fixed structure, but allow for both dynamically changing model structure and parameters. In R&D projects, where simulations are

a major tool to validate concepts such extended parametric sweep allows assessing which parameters have the highest impact on the system and achieving the best compromise among performance, complexity, and cost [35].

6. Maintainability –Well-structured and designed project already becomes maintainable as it is divided into many decentralized modules with low coupling. In turn, any modification in one module does not affect the remaining parts of the system. Test Driven Development (TTD) approach provides a set of rules to develop maintainable code. According to that software development principle each module should be covered by unit-testing procedures that assess its performance. Next, integration tests should be defined in order to validate the interaction between modules. Finally, the tests of the entire system should be performed (acceptance tests). As a consequence, the project developed following a TTD approach can be easily extended and adapted to new requirements. [2, 21]

4.2. Architecture Definition

In order to obtain all aforementioned requirements, the following hierarchical architecture, composed of three main layers, has been designed as depicted in Figure 4.1.

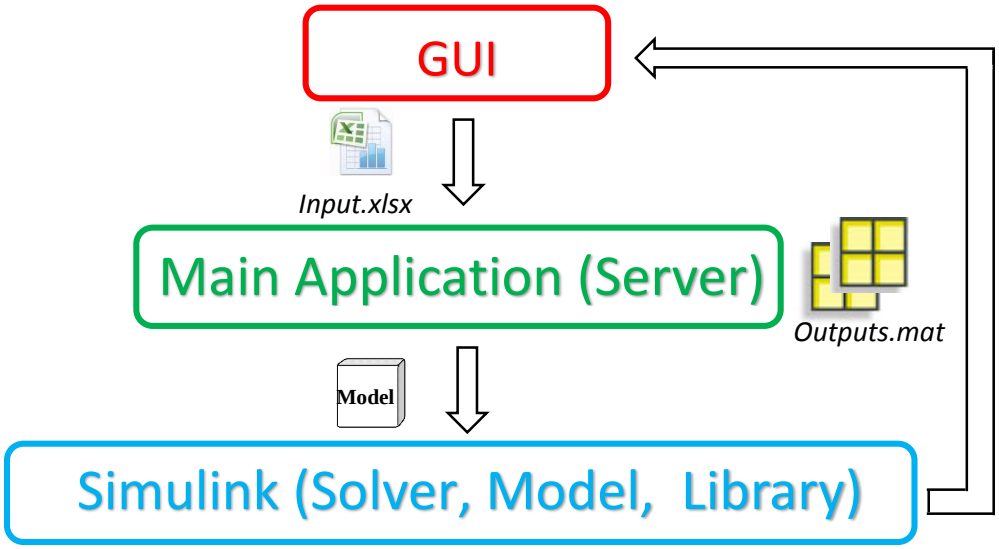


Figure 4.1. Three-layered structure of the QSF and communication channels between layers

The first layer is Simulink with its sophisticated network solver and library of reusable components written in the Simscape language. On top of that, there is a main application developed in MATLAB programming language that handles all tasks needed to build a model and also to post-process simulation results. Finally, the user communicates with the QSF by the intuitive GUI, which enables sketching electrical circuit of the model, defining blocks parameters, running either single simulation or a set of simulations with varying parameters and finally analyse obtained results. The QSF employs spreadsheet files (Excel-based in particular) to store all information required to

build the model of the superconducting magnet. Spreadsheet is a widely accepted file type and can be conveniently used to store all relevant parameters in the tables. Additionally, employed file type simplifies model description storage and sharing with other users. Furthermore, the input model file can be edited outside of the MATLAB environment and then directly fed into the QSF framework. Hence, MATLAB license is not necessarily needed to modify input files.

The single **Model** class instance entirely describes the model of the superconducting magnet and serves as a link between main MATLAB application and low-level Simulink model and its network solver. In other words the **Model** object stores information about model structure, block parameters, and is equipped with methods to open Simulink schematic, create model, run simulation and lastly close the Simulink window.

Although Simulink has built-in feature to save Simscape physical systems simulation results into a single MATLAB workspace variable (simlog), the results of the superconducting magnets simulations are stored in many *.mat* files (one file with vector of signals per each model block). The reason for this is twofold. Firstly, due to large scale of standard simulations the single simlog variable was consuming an excessive amount of memory. Secondly, even if the simlog variable was saved, access to one particular signal was more time consuming than in case of distributed *.mat* files. The GUI provides also a dedicated tab to plot the obtained results using both simple time-series plots and custom user-defined functions (2-D profiles, temperature-series, etc.).

Nevertheless, the user is most interested in the definition of structure, components, parameters, and circuit, and in the analysis of results. Furthermore, the duties of the software developer are also strictly defined – the developer is responsible for implementation and maintenance of the framework that enables the users to run simulation in a very convenient way. The QSF in its closed form does not require the user to either write or modify existing code to execute simulations. Most importantly, the QSF dynamically adapts to input simulation definition and creates the model by calling appropriate software modules.

Each of the three main QSF layers has been implemented applying the Object-Oriented Programming (OOP) paradigm in the MATLAB and Simscape programming languages.

Since the framework deals with a great number of lumped-element library components from two physical domains, the natural representation is to use OOP in order to represent them in the code. The OOP paradigm provides techniques to represent a problem in a hierarchical, structured way. The OOP enables creating new data types to represent the object in a coherent way, i.e. the class is the most fundamental building block of the code. In other words a class is a new data type that can be used in the code along with base data types such as logical, numeric and string. The class stores information describing a particular object (for example resistance of an energy extraction system) and

actions that the object can perform (for example object knows how to be added to the schematic). Due to data encapsulation, the access to the stored data in objects is limited and can be done only by appropriate set and get methods. Hence, the code is protected against accidental parameters modification by overwriting variables. Additionally, the mechanism of inheritance greatly supports problem decomposition by means of generic base classes and more specific derived classes. The base class stores information and functions common to some groups of objects, whereas each derived class has more detailed information about particular object. Thus, the properties (for example CLIQ operating parameters as well as position on the schematic) and methods (get parameters, get port handles, add block to the schematic, set block parameters, connect blocks) are expressed in a unified, coherent way. In other words single class instance possesses all information about the Simulink library component and all operation it can perform. What is more, the OOP encourages building a hierarchical and well-structured code. Hence, it makes it easy to reproduce the model structure in the code and at the same time supports scalability, modularity and maintainability. The OOP provides techniques to define rules on how the code should be created by means of abstract classes, and interfaces. On the one hand it reduces the problem complexity by creating base blocks that can be smartly connected together, and on the other defines steps to be taken in order to enhance the framework capabilities (add new library component, new plotting function, etc.). Moreover, many useful design patterns have been derived for the object-oriented languages, hence it further simplifies development of clean, self-documenting code [1]. OOP paradigm has been applied to the code of library components by applying the Simscape language along with the main application and GUI developed in the MATLAB programming language.

As a consequence of hierarchical, modular structure, the designed architecture is decentralised. As pointed out in [20] the higher the autonomy of particular system parts, the lower the probability that changes applied to one or a few modules will affect remaining modules in the project. Application of the OOP techniques results in a simple software structure, even though it implements a complex problem. The layered structure simplifies software validation assessment in terms of conditional correctness [20]. Each layer is assessed to be correct if performs all assumed tasks and all previous layers are working properly as well.

4.3. The QSF Library

The QSF was designed and developed maintaining the library of component as its one of the most fundamental element. This was a natural choice as each library element implements some portion of the entire superconducting magnet model (see Chapter 3). Additionally, the process of building the models requires deep understanding of physical phenomena governing them.

Furthermore, the bottom part of the system is most likely to be modified or extended on the basis of increasingly validated components and experience.

All equations described in the Chapter 3 are implemented in the Simscape object-oriented language. It allows to define base, fundamental building blocks that can be used to build more specific ones. Furthermore, it is profitable to connect together various Simscape blocks that represent coherent part of the magnet model and standard Simulink components to perform simulation control tasks (for example stop of the simulation). A good example is the calculation of the temperature in the magnet hot-spot (the highest temperature in the magnet) composed of a magnetic field calculation component, a controlled heat source, and a thermal mass. Moreover, in order to stop the simulation when the temperature is greater than a certain threshold simple logic was employed by means of functions from *Sinks and Logic and Bit Operations*. If the temperature threshold during the simulation was reached, the magnet was not protected and further simulation results are meaningless. Additionally, the hot-spot block during simulation saves temperature and heat flux.

In order to account for various modelling requirements hybrid Simscape-Simulink library of components has been designed, developed, and tested. As a result, the Simscape and Simulink elements are wrapped up together in subcomponents and are represented by an icon and mask[1] with a list of parameters [6]. Such approach has many advantages. Firstly, all blocks in the Simscape-Simulink library have exactly the same interface with input and output ports and list of parameters. Secondly any modification of internal block structure does not affect other framework modules (as long as the list of ports and parameters remains the same). In turn, any modification of Simscape components (for example modification of governing equations) or addition/removal of signals saved during simulations can be done very quickly. Thirdly, such an approach provides a strict definition of new components. Finally, the icon and mask are used to automatically generate a prototype of a class. There is always one single dedicated class per each Simscape-Simulink library component that represents the model in the main application.

Moreover, the QSF is capable of generating the Simscape code based on input parameters (see 4.3.4) as well as automatically create complex electro-thermal dynamic models of the superconducting magnets.

The library of available components of the QSF is summarised in Figure 4.2. All Simscape components are listed inside blue and green boxes. The Simscape library consists of both thermal (denoted by red font) and electrical (denoted by light blue font) components. Additionally, the Mutual Inductor block, even though is developed in Simscape language is distinguished by violet font. The

[1] Mask is a custom parameter dialog box of the subcomponent.

reason for this is that the code of Mutual Inductor library components is generated automatically based on the **M** matrix (eq. 3.19). Next, the Simscape blocks together with Simulink standard library elements create hybrid components (denoted by boxes with green border) and are directly used in the appropriate superconducting magnet model. Finally, there are also three components created automatically during superconducting magnet model development (denoted by boxes with blue border), namely an electro-thermal superconducting magnet model, a chain of superconducting magnets model, and a Quench Heater model. Their structure is pre-defined, however the number of subcomponents is varying and defined in the input file.

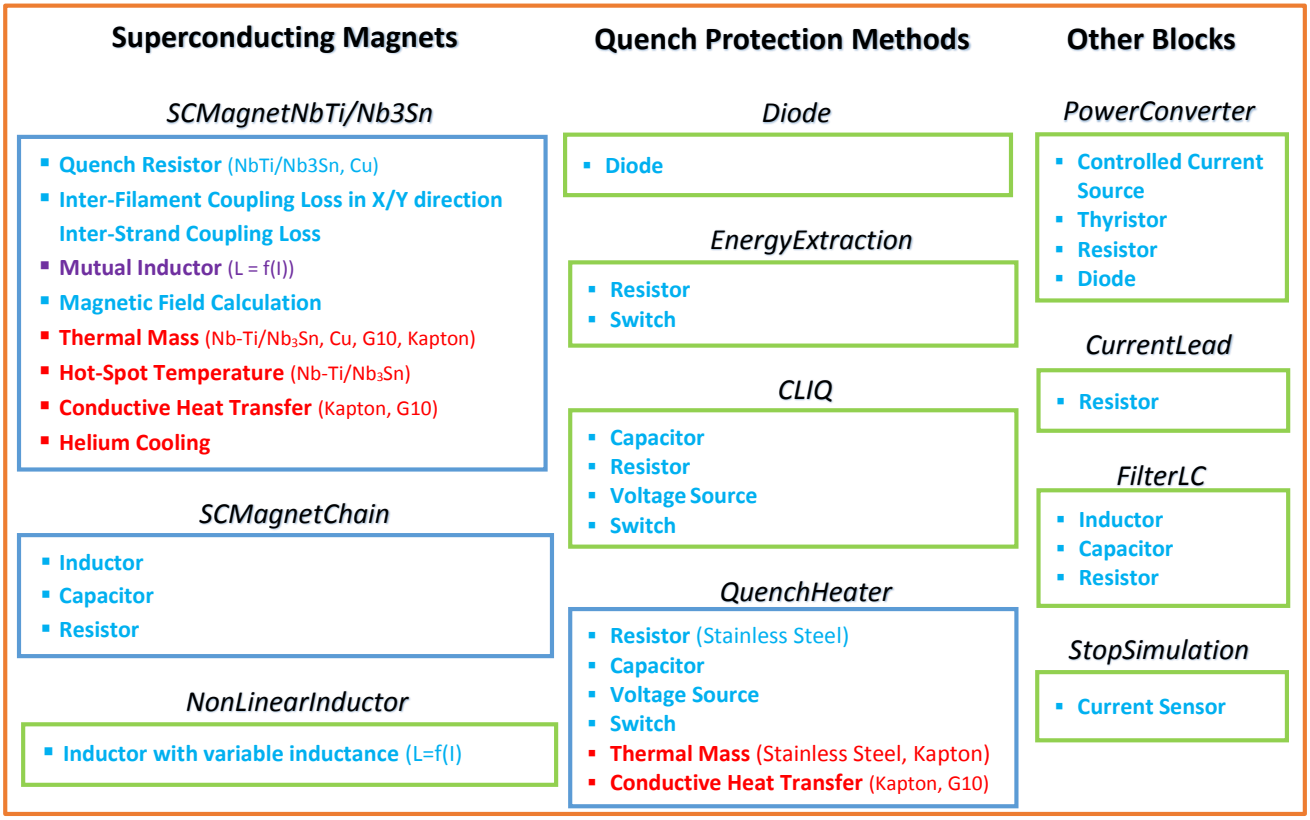


Figure 4.2. The QSF library structure

From another, purpose-based perspective, the QSF library can be divided into three main categories:

- **Superconducting Magnets** category consisting of full electro-thermal model, nonlinear inductor implementing relation between inductance and current and simplified lumped magnet model. The decision on which magnet model should be used depends on the requested level of detail, i.e. if quench mechanism can be neglected a simple nonlinear inductor is a good magnet approximation during standard operation. Moreover, the model of superconducting magnet may be composed of both simple and complex magnet models connected together.
- **Quench Protection Methods** category composed of standard and prototype magnet protection devices. Namely, the modeller can select from a by-pass diode, an energy extraction, quench heaters and CLIQ units.

- **Other Blocks** category containing all remaining blocks such as Power Converter models (controlled current-source), current leads, and stop simulation block. The latter can be used to stop the simulation when the main current in the system drops below a certain value as a consequence of current discharge. In fact, at the end of the simulation, when the most relevant transient effects disappeared high resolution is not required. For large scale models (composed of over 1000 lumped blocks) when current drops to zero, the zero crossing error may occur, which results in tightening time step in attempt to accurately calculate the moment of reaching zero.

Summary of relations between electro-thermal Simscape library (see Figure 4.2), standard Simulink library, library of dynamically generated blocks, library of mutual inductors whose Simscape code is generated automatically and resulting QSF library is schematised in Figure 4.3.

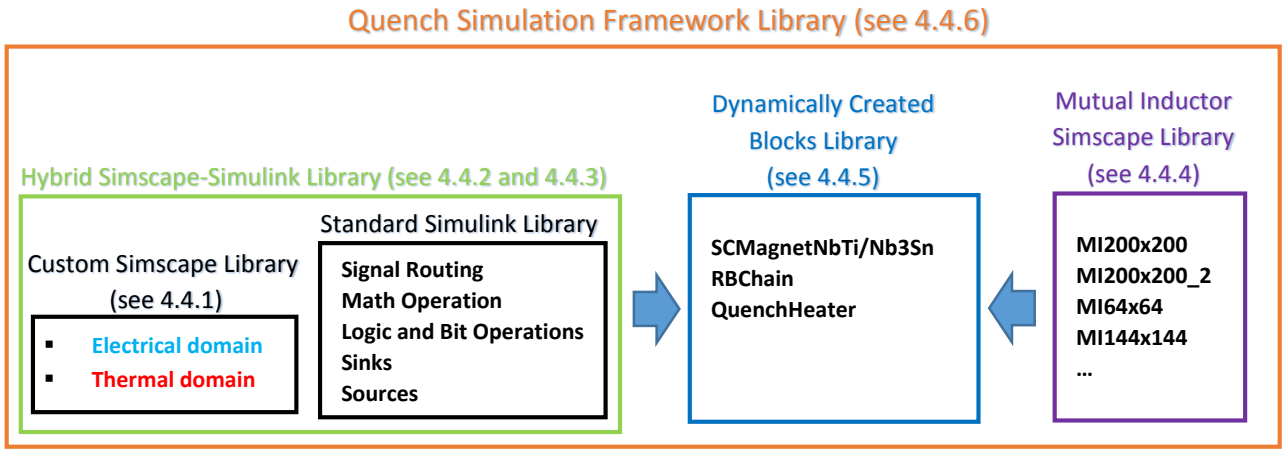


Figure 4.3. Relations between the QSF library and sub-libraries

As one can notice, the QSF library consists of three types of sub-libraries. Each type has completely different characteristics, due to the required different design procedure. Some of them are static, whereas some of them are created dynamically. Hence, it is important to develop a unified interface to access each element from the QSF library in the same way. Proposed solutions, based on OOP techniques, are described in the following sections.

4.3.1. Custom Simscape Library – Quench Resistor Simulink Implementation

The first stage in adding a new component to the framework is the implementation of DAEs characterizing its behaviour in the Simscape object-oriented language. Furthermore, in the case of superconducting magnets DAEs are usually nonlinear with variable coefficients (mostly the coefficients are a function of temperature, magnetic field and/or current). Moreover, the modelled superconducting circuit is subjected to sharp changes, for example when a transition to the normal state occurs in a block. The Simscape language allows implementing material properties explicitly or implicitly as a Look-Up Table (LUT). Furthermore, the Simscape technology allows creating a physical network, hence, the applied approach is different from Simulink, where blocks represent mathematical operations. As a consequence all physical connections are non-directional, and connecting Simscape blocks is equivalent to connecting two objects like a voltage source and resistor in a conventional circuit modelling program. Thus, the modeller is directly implementing the problem in the simulation environment. Before the Simulink starts simulating the modelled system, the Simscape solver compiles together all equations that characterize the behaviour of the entire morel. Subsequently, all DAEs characterizing the system behaviour are solved simultaneously at each simulation step [12]. Thus, common simulation problem with the algebraic loops, where one signal influences the other, whereas the former has also an impact on the later, is therefore resolved. A good example is the model of a quench resistor and a thermal mass of the superconducting magnet. After transition from the superconducting to the normal state Ohmic losses are generated in a conductor following the Joule's law. The produced heat increases a temperature, which in turn determines the local value of physical properties such as electrical resistivity, heat capacity, and thermal conductivity.

For every physical domain in Simscape one can define generic base component. Such a component can be used to build more specific ones by inheriting signals and variables. A good example of this concept is a variable resistor modelling the resistance of a superconducting cable. In the electrical domain, the most fundamental component is an electrical branch simply defining the current flowing through the component and the voltage drop across it (see Listing 4.1). Listing 4.1 presents the Simscape syntax and code structure. Each file starts with the keyword *component* followed by a name defining component (line 1). Lines 2-4 include comments that will be displayed as the component description in the context help. The *nodes* section (lines 5-8) describes electrical ports and the comments after line 6 and 7 defining name (+ and -) and node position on the icon (left and right, respectively). Next, current and voltage variables together with corresponding units are specified (lines 9-12). Finally, last section defines the current flow and the voltage drop (lines 13-16).

```
1   component branch
2   % Electrical Branch
3   % Defines an electrical branch with positive and negative external nodes.
```

```
4   % Also defines associated through and across variables.
5     nodes
6       p = foundation.electrical.electrical; % +:left
7       n = foundation.electrical.electrical; % -:right
8     end
9     variables
10      i = { 0, 'A' };
11      v = { 0, 'V' };
12    end
13    function setup
14      through( i, p.i, n.i );
15      across( v, p.v, n.v );
16    end
17  end
```

Listing 4.1. Simscape code for electrical branch

Such a generic component can then be used as a foundation to create new electrical elements. In terms of OOP each electrical component with two nodes like resistor, inductor or capacitor in Simcape inherits from an electrical branch (see Figure 4.4).

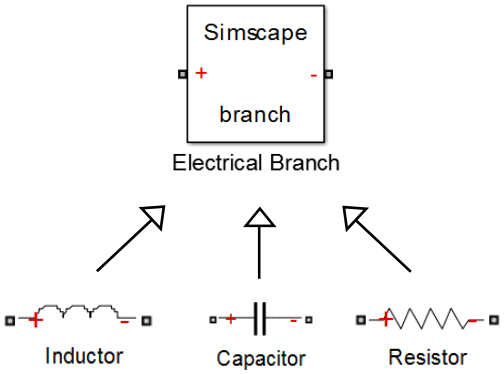


Figure 4.4. Inheritance mechanism of electrical components

What is more, by inheriting from base electrical element the new component is automatically added to the electrical domain, and therefore the Simscape solver will check whether each electrical node is connected to the electrical network of the system. Hence, this mechanism helps reducing errors.

The Simscape object-oriented language has been applied to implement all equations characterising the behaviour of electro-thermal dynamic lumped-element blocks described in Chapter 3. Once the code is written it is compiled by an external C++ compiler (for example Microsoft Visual Studio). The resulting compiled block is added to the Simscape library and can be subsequently used in future Simulink simulations.

In order to better explain how physical equations are represented in the Simscape code, the implementation of the block modelling resistance of a cable made of copper and Nb-Ti is presented in the Listing 4.2. First of all, the quench resistor inherits from already described electrical branch (line 1). In turn current and voltage are already defined (in the electrical branch component), and modeller can focus on defining more specific properties and equations related to quench margins (lines 40-54), copper resistivity (line 52) as a function of average temperature and magnetic field, or a flag signal indicating whether a transition from the superconducting to the normal state occurred.

Next, the single thermal node is defined so that produced Ohmic loss can be connected to thermal network of the model (lines 4-6). Lines 7-18 define input and output physical signals. The component during simulation reads information about total magnetic field used to calculate copper resistance (line 8) and total magnetic field margin used to calculate quench margin (line 9). Signals saved during simulation include the Ohmic loss (line 12), transport current of the block (line 13), the quench flag (line 14), the copper resistivity (line 15), the temperature current-sharing (line 16), and copper resistance (line 17). Note in Figure 4.5 that on the icon input and output ports are represented by a triangle pointing into the icon, and out of the icon, respectively. Nodes, are represented by squares since they are non-directional. It is necessary to stress that input and output signals are applied to read information to block or write information to block. In other words, they serve as information ports, i.e. each physical signal has a single source, is not correlated with any physical domain and does not change its value even if connected to many other blocks. Hence, they behave differently than the nodes (so-called physical conserving ports) for which the law of conservation of energy holds.

Afterwards, the set of parameters is determined (lines 19-30). Each parameter is characterised by a name, physical units and a description (in a comment after percentage sign) that appears in the block mask. In the *variables* and *function setup* sections the generated Ohmic loss (Q) and the temperature (T) are defined as well as already defined current and voltage.

The *equations* section implements equations governing block behaviour. Lines 41 and 42 calculate critical current and critical temperature, respectively (hence implementing equations 3.8 and 3.7). Then, they are used to calculate temperature current sharing in line 45. Moreover, the copper resistivity is calculated in line 51 by implementing equation 3.3. Note that this resistance is set to value of R_{min} parameter (0 Ω by default) if the quench flag is zero. Furthermore, the voltage (line 55) and the Ohmic loss (line 56) associated with v and Q variables are calculated. Finally output signals: the Ohmic loss (line 57), transport current (line 58), quench flag (line 59), copper resistivity (line 60), temperature current-sharing (line 61), and copper resistivity (line 62) are calculated.

The *let in end* statement is applied to represent equations in more structured way. Namely, between the *let* and the *in* keywords variable can be defined as a value or equation and later used in the statement between the *in* and the *end* keywords (lines 44-64).

```
1   component resistorQuenchNbTi < foundation.electrical.branch
2       % Quench Resistor NbTi
3       % The block simulates quench resistance of superconducting NbTi cable
4       nodes
5           P = foundation.thermal.thermal;       % OHM: right
6       end
7       inputs
8           Bt = { 1, 'T' };            % Bt: left
9           BtMarg = { 1, 'T' };        % BtMarg: left
10      end
11      outputs
12          QOhmOut = { 1, 'J/s' };     % QOhm: right
13          iQROut = { 1, 'A' };        % iQR: right
14          flagQOut = { 1, '1'};       % flagQ: right
```

```
        rhoOut = {1, 'Ohm*m'}        % rho: right
        TcsOut = { 1, 'K'};          % Tcs: right
        ROut = {1, 'Ohm'}            % R: right
    end
    parameters
        t_quench = { 0, 's' };       % Artificial Quench Trigger
        RRR = {0, '1' };             % Residual resistance ratio
        rho_mag = { 0, 'Ohm*m/T' }; % Magneto-resistivity factor
        R_min = { 0, 'Ohm*m/T' };    % Splice resistance
        l_group = { 1, 'm' };        % Length of the coil
        cs_cable_Cu = { 1, 'm^2' }; % Area of Cu
        Bc2 = { 1, 'T' };            % Critical Magnetic Field
        Tc0 = { 1, 'K' };            % Critical Temperature
        c1_Ic = { 1, 'A' };          % Coefficient c1_Ic
        c2_Ic = { 1, 'A/T' };        % Coefficient c2_Ic
    end
    variables
        Q = { 0, 'J/s' };
        T = { 0, 'K' };
    end
    function setup
       through( Q, [], P.Q );
       across( T, P.T, []);
    end
    equations
        let
            Ic = c1_Ic + c2_Ic*BtMarg;
            Tc = Tc0*( {1, '1'} - BtMarg/Bc2)^0.59;
        in
            let
                Tcs = Tc*( {1, '1'} - i/Ic);
            in
                let
                    c0 = { 1.7, 'Ohm*m' };
                    c1 = { 2.33*1e9, 'K^5' };
                    c2 = { 9.57*1e5, 'K^3' };
                    c3 = { 163, 'K' };
                    rho = (c0/RRR+{1, 'Ohm*m'}/(c1/P.T^5+c2/P.T^3+c3/P.T))*1e-8+rho_mag*Bt);
                    flag = ( (T > Tcs) || (time > t_quench) );
                in
                    v == i * rho * l_group / cs_cable_Cu * flag + i*(1-flag)*R_min;
                    Q == i^2 * rho * l_group / cs_cable_Cu * flag + i*(1-flag)*R_min;
                    QOhmOut == i^2 * rho * l_group / cs_cable_Cu * flag + i*(1-flag)*R_min;
                    iQROut == i;
                    flagQOut == flag;
                    rhoOut == rho;
                    TcsOut == Tcs;
                    ROut == rho * l_group / cs_cable_Cu * flag + i*(1-flag)*R_min;
                end
            end
        end
    end
end
```

Listing 4.2. Nb-Ti quench resistor Simscape code

4.3.2. Hybrid Simscape-Simulink Library – Quench Resistor Code Representation

After successful component compilation (depending on size of a component and size of the library, the compilation may take from few seconds to couple of minutes) and verification, the Nb-Ti Quench Resistor component has been added to a bigger schematic composed of components from both Simscape (*Quench Resistor NbTi*, *Calculate Bt*, *Calculate BtMarg*, *Physical Signal-Simulink Conversion*, *Inport* blocks) and standard Simulink (*Mux*, *Outport* blocks) libraries as depicted in Figure 4.5. *Calculate Bt* and *Calculate BtMarg* components calculate absolute magnetic field according to equation (3.14) and pass results to the *Quench Resistor NbTi* implementing equations (3.3, 3.7, 3.8). During simulation 10 signals are saved to *.mat* files, hence each of them is firstly converted from Physical Signal data type to Simulink data type by *Physical Signal-Simulink* conversion function. Next, all converted signals are merged together by the *Mux* block and the

resulting vector of signals is passed outside of the schematic through *Outport* component. All aforementioned blocks create a sub-component, and their parameters are now accessible through the mask. Moreover, the sub-component also obtains an icon and appears as an atomic element on the model schematic. In other words, the Simulink sub-component is an equivalent of a function or a procedure in text-based languages (C/C++, Java, MATLAB) with defined input, output and list of additional parameters.

It is important to build clean sub-component schematics with meaningful block names, port names and signal names, as it becomes self-documented (see Figure 4.5). Furthermore, the icon appearance can be adjusted in order to better reflect the block characteristic. For instance, in Figure 4.5 the *Quench Resistor NbTi* block has a green background (meaning that it belongs to the Simscape-Simulink library) and blue foreground (meaning it belongs to the electrical domain).

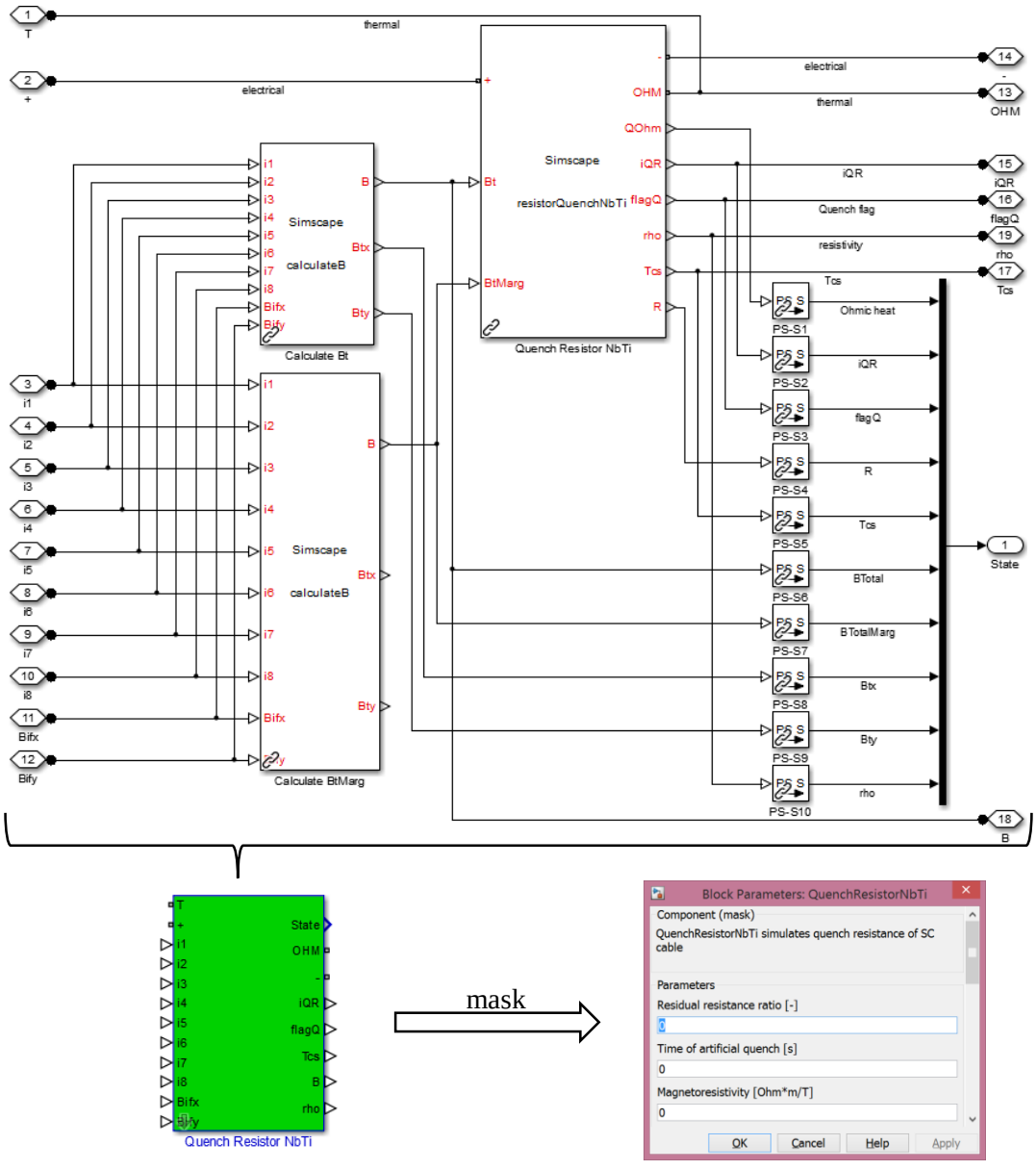


Figure 4.5 Simscape-Simulink library component development procedure

Component testing was accomplished by creating a Simulink test model containing the new component and running testing scenarios so that material properties, input/output mapping, and overall behaviour are validated. Once a component was verified in a test routine, all changes were automatically incorporated into all models containing that component by means of the library [35].

A description of physical equations implementation in the Simscape object-oriented language and addition to the hybrid Simscape-Simulink library concludes this section. The described procedure applies to every element presented in Figure 4.2 and denoted by a green box. Furthermore, the mask and the icon in the Simscape-Simulink library will be used as a pattern in the next stage and will serve as a link between that library and corresponding MATLAB classes.

4.3.3. Hybrid Simscape-Simulink Library – Class Representation

Development of a reusable library of components was the first step towards developing an automated simulation framework. Furthermore, the QSF library is a key element of the automated, object-oriented simulation framework as each component implements either some physical phenomena of superconducting magnets or quench protection mechanism and additional blocks (see Figure 4.2). Since Simulink is seamlessly coupled with MATLAB environment, and especially there is a set of MATLAB functions devoted to create simulations automatically, main application has been developed. First of all, the main MATLAB application has to provide an interface to efficiently connect with the QSF library made of blocks implemented in different manner. Next, dedicated class hierarchy must be defined in order to reproduce all sub-systems in the model and relations between them. Finally, the application should be capable of handling the execution of simulation sets, data visualization, and report generation.

In order to develop a well-structured, hierarchical software architecture, OOP capabilities of the MATLAB programming language have been applied. A simplified Unified Modelling Language (UML) standard notation has been employed to represent class hierarchy of the main application [22] in the remainder of the chapter. The UML is a widely accepted industrial standard of graphs and charts notation, however its capabilities are not only limited to represent class diagrams. In brief, classes are represented by a box with a name on the top, and fields and methods inside the box (see Figure 4.6). The class fields are described by their accessibility, name and type. Private, public and protected fields are denoted by ‘-‘, ‘+’, ‘#’, respectively. The class name written in italics represents the interface (together with <<interface>> word). Finally, the abstract class representation on the UML diagram has a name written in italics. In order to conserve space it is assumed that for each private class field corresponding accessor and mutator methods exist, but it is not presented on the diagrams. Additionally, each presented class as every class in MATLAB inherits from a **handle** class and it is not included in the schematic.

The first stage of the QSF middle layer development was to design a base class composed of a list of properties and methods common to every block of the hybrid Simscape-Simulink library (depicted in green in Figures 4.2 and 4.3). In general, every block placed in the Simulink schematic can be described by the list of properties depicted in Figure 4.6.

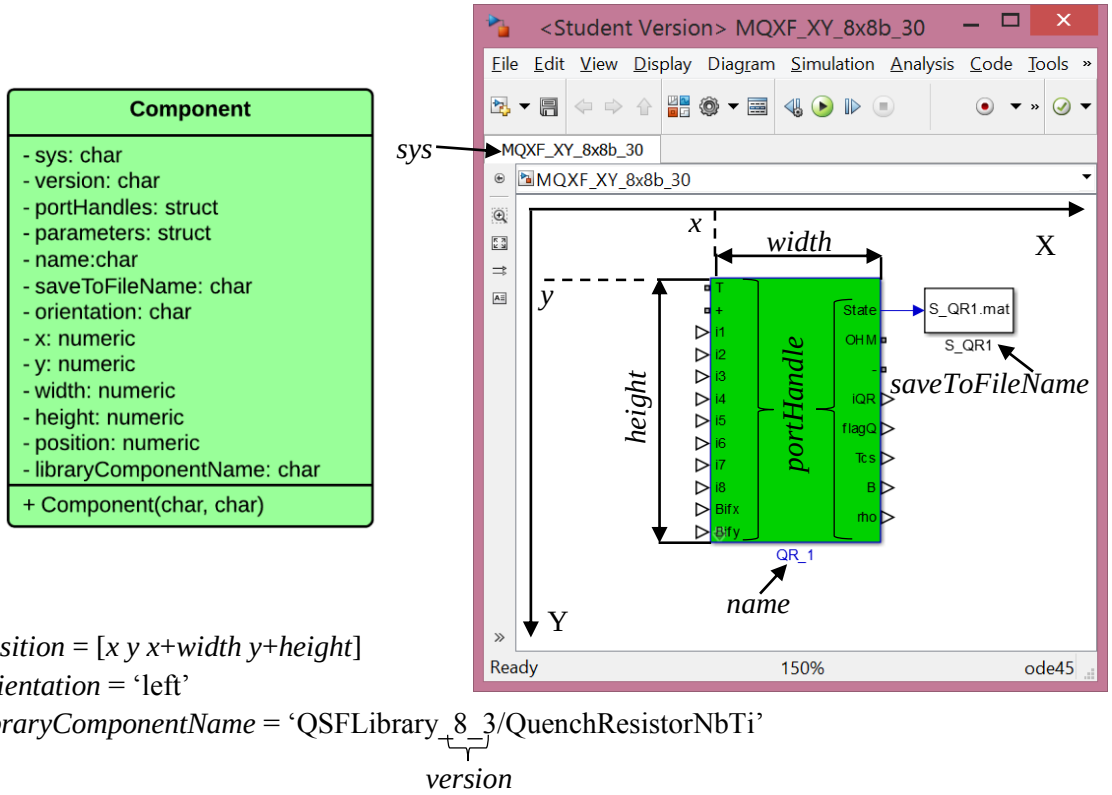


Figure 4.6. Component Class and properties description

Component is a single class including parameters describing position and orientation in the Simulink schematic together with name of the model, block and library component. Furthermore, the list of parameters is necessary to automatically build a model by calling an appropriate MATLAB function. In order to add a block[2] to the Simulink the following list of parameters must be provided: name of the model (*sys*), name of the block (*name*), name of the library component (*libraryComponentName*), position (*position* as a 4-element vector [*x y x+width y+height*]), and orientation of the block (*orientation*). If needed, a "save to file" block can also be added (*saveToFileName*). In Simscape-based simulations, signals travel from one block to another through wires. To connect[3] two blocks together port handles (*portHandles*) and their names are necessary. For that purpose the **Component** class has a dedicated method to access each port handle by name. Finally, names of the block parameters (*parameters*) are required to set parameters[4].

[2] add_block(*nameLibraryComponent, [sys '/' name], 'Position', position, 'Orientation', orientation*)
[3] add_line(*sys, outputPortHandle, inputPortHandle, 'Autorouting', 'on'/'off'*)
[4] set_param(*[sys '/' name], param1, value1, ..., paramN, valueN*)

The proposed description, however, is not limited to the QSF library but applies to every component from the standard Simulink library as well as components from libraries of additional toolboxes. In this sense, the middle layer of the QSF is generic and can be reapplied to any simulation task implemented in MATLAB&Simulink.

All blocks from the QSF library inherit from the **Component** class, hence it is guaranteed that each of them has the same set of parameters required to build a model. However, at this stage more detailed information on particular library component is required. As it has been already stated each component from the Simscape-Simulink library is represented by an icon and a mask. Figure 4.7 and Listing 4.3 present an example on how the parameters of the *QuenchResistor NbTi* component are accessible in the mask and reflected in the corresponding class body.

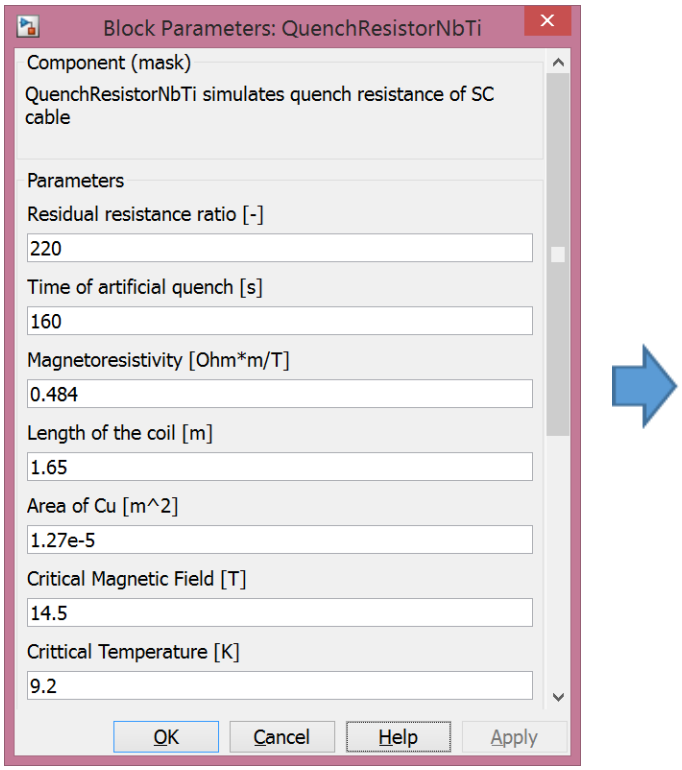


Figure 4.7. Quench Resistor NbTi block mask

```
classdef QuenchRResistorNbTi < Component &
Automatable
%QuenchResistorNbTi simulates quench
% resistance of superconducting NbTi cable
properties
    % Residual resistance ratio [-]
    RRR
    % Time of artificial quench [s]
    t_quench
    % Magnetoresistivity [Ohm*m/T]
    rho_mag
    % Splice resistance [Ohm]
    R_min
    % Length of the coil [m]
    l_group
    % Area of Cu [m^2]
    cs_cable_Cu
    % Critical Magnetic Field [T]
    Bc2
    % Critical Temperature [K]
    Tc0
    % Coefficient c1_Ic [A]
    c1_Ic
    % Coefficient c2_Ic [A/T]
    c2_Ic
    % Magnetic field TF X [T/A]
    fMagX
    % Magnetic field TF Y [T/A]
    fMagY
    % Magnetic field Margin TF X [T/A]
    fMagMargX
    % Magnetic field Margin TF Y [T/A]
    fMagMargY
end
methods
    function obj = QuenchResistorNb3Sn(sys,
    version, name, md, index )
    function obj = placeBlocks(obj)
    function [] = setParameters(obj)
end
end
```

Listing 4.3. QuenchResistorNbTi class body

Once the library component is developed the appropriate class prototype is automatically generated based on the component description in the library. This way the representation of the block in the corresponding class is created, i.e. a list of parameters is copied from the Simulink mask to the class body.

In principle, each element of the Simscape-Simulink library has a different set of parameters. In order to reduce errors occurrence and save time spent on writing class body for each corresponding

library element, the class prototype is automatically generated based on the component mask and icon. MATLAB provides functions to get properties from the block mask (needed to initialise actual class prototype) and read number of ports from the block icon (needed to initialise **Component** class). The actual names of the internal variables used to pass values from mask to blocks inside the sub-component are names of class properties (for example RRR, see line 7 in Listing 4.3). These names are invisible in the mask; their corresponding prompt text is defined in the class body by means of comments introduced by the sign "%" (for example, see line 6 in Listing 4.3).

In some cases it might be necessary to modify code to include additional features. For this purpose some classes are equipped with a parameter range check in the class prototype. For instance, all geometrical dimensions have to be positive. The Simscape provides tools to validate parameters during model compilation. However, in order to speed up simulation execution it is profitable to validate parameters during object creation in the MATLAB code, not on the Simulink schematic. Moreover, in case of errors, more detailed description from the QSF is offered.

It is worth noticing that the described procedure applies to every element of the hybrid Simscape-Simulink library and by executing code generator all component class prototypes have been created. In result such mechanism greatly simplifies extension of the hybrid library by adding new components. Namely, the modeller has to implement physical equations in the Simscape component, add Simulink components if needed, build sub-component with an icon and mask, and finally generate class prototype.

A common feature shared by all elements from the hybrid Simscape-Simulink library is that they can be added to the schematic which is represented by an *Automatable* interface. It requires from other classes to implement *placeComponent()* method, that adds a component to the Simulink diagram, and *setParameters()* method that sets the parameters in the mask of the corresponding component.

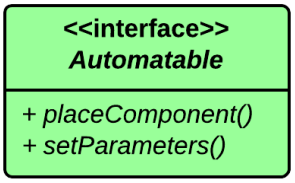


Figure 4.8. *Automatable* interface diagram

To sum up, each class representing components created from both Simscape and Simulink blocks inherits from the base **Component** class and implements the interface *Automatable*. Hence, all classes have a list of common properties and an individual implementation of methods requested by the interface. However, the particular implementation may vary from one class to another, the only differences between blocks being the list of parameters and port handles, and the methods for implementing of *placeComponent()* and *setParameters()*.

4.3.4. Mutual Inductor Simscape Library – Automatic Code Generation

The model of the Mutual Inductor is a hub, which couples together overall effects of multi-scale elements, i.e. superconducting cable, strands of superconducting cable, and finally the filaments. Since the number of components used in the model with fine spatial resolution is usually large (over 5 thousand lumped components) and can be subjected to modifications, the Simscape code for the **MutualInductor** block must dynamically adapt to those variations. Profiting from the well-structured syntax of the Simscape language, it was possible to write methods inside the **MutualInductor** class that automatically generate and compile the component code, based on the **M** matrix and on a look-up table accounting for the dependence of the magnet inductance on the transport current. Since the **M** matrix is sparse[5] the Simscape code directly implements only non-zero elements. As a result, the Simscape solver calculates only the relevant current derivatives (see equations 3.19 and 3.22). Furthermore, in order to reduce compilation time, the coefficients of equations 3.19 are embedded in the code, which means that they cannot be accessed through the block mask.

To sum up, such an approach provides significant time savings. For instance, the Simscape **MutualInductor** component of a superconducting magnet composed of 8 electrical parts (N_E=8) and 64 blocks (N_B=64) and dim(**M**)=440x440, requires over 4000 lines of code which are created in a repeatable manner in less than 15 minutes (including both code generation and component compilation). In order to further save time the compiled components are stored in the library with information about **M** matrix and current-inductance LUT.

Figure 4.9 shows the block diagram of the generation of the **MutualInductor** component. For each new simulation, firstly the Mutual Inductor Simscape library is checked to establish whether the requested component already exists. If the component is found, the new simulation starts without the need to recompile a new **MutualInductor** component. Otherwise, the matrix **K** is calculated in order to check if all its elements, i.e. coupling coefficients are smaller or equal to 1. If this condition is fulfilled the stability of **M** matrix is verified; the procedure generates Simscape code and compiles new component. On the contrary, if at least one element of **K** is greater than 1 (eq. 3.35) or if **M** is unstable (eq. 3.36) the procedure generates an appropriate error and ends the current run without starting a new simulation.

[5] For N_B=64, **M** has only 4% non-zero elements. In general **M** has $N_E \cdot N_E + (2 \cdot 3 N_E \cdot N_B + 3 \cdot N_B)$ non-zero elements.

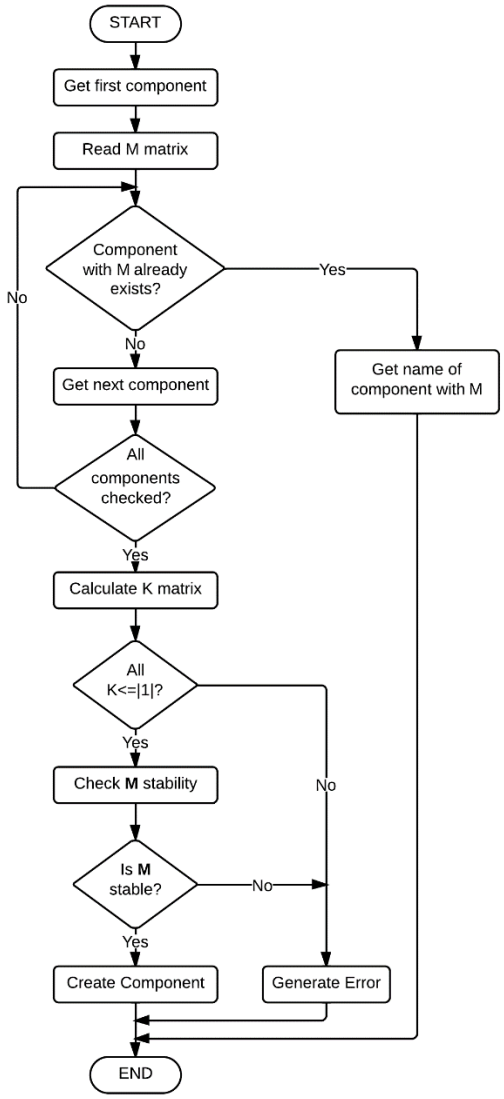


Figure 4.9. Block diagram of Mutual Inductor component generation

4.3.5. Dynamically-Generated Component Library

Components with fixed structure, but variable number of blocks represent the most complex part of the QSF library. Due to unknown *a priori* size, it was not possible to create static components for complex models of superconducting magnets or Quench Heaters as in the case of CLIQ or Energy Extraction models. Hence, aforementioned components are created dynamically during the model design. For that purpose, a dedicated interface (see Figure 4.10) has been designed, composed of functions required to build a component dynamically.

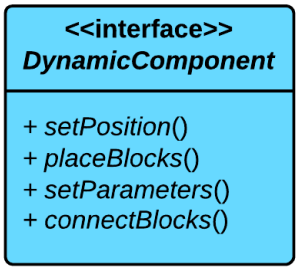


Figure 4.10. *DynamicComponent* interface diagram

First of all, each dynamic component has to calculate the position of all sub-elements. For maintainability and debugging simplicity sake, such a method should be scalable and produce a well-structured network, so that by reading block coordinates it is possible to quickly locate it on the schematic. Then, each compositional block has to be added to the schematic. Finally, each class describing dynamic component must be equipped with a method that sets parameters and connects together all sub-components.

One of the most important tasks of the QSF was to implement a lumped-element dynamic model of superconducting accelerator magnet. The model combines together components from the hybrid Simscape-Simulink library and Mutual Inductor Simscape library (see Figure 4.3). Those blocks inherit from **Component** class and implement interface *Automatable*. The *Automatable* interface defines methods needed to add a component to the schematic automatically. Additionally, note that size of corresponding objects reflects definitions from Chapter 3.

Full, Lumped-Element Dynamic Electro-Thermal Model of Superconducting Magnet made of Nb_3Sn cable is depicted in Figure 4.11 (a diagram for Nb-Ti has the same structure, but different components). The full model is composed of Simscape-Simulink library components (green), and Mutual Inductor Simscape Library (violet). Moreover, since each magnet model is described among other parameters by number of electrical parts N_E, number of blocks N_B, and order of electrical parts connection, a base class **Magnet** has been derived. Interface *Netlistable* is required for the elements of the QSF library to be directly added to the schematic and is described in the following section. Furthermore, the class for chains of superconducting magnets has been developed and is depicted in Figure 4.12. The **SCMagnetChain** class can be composed of the cell of **MainDipole** objects from hybrid Simscape-Simulink library, extends the **Magnet** base class and implements the interfaces *Netlistable* and *DynamicComponent*.

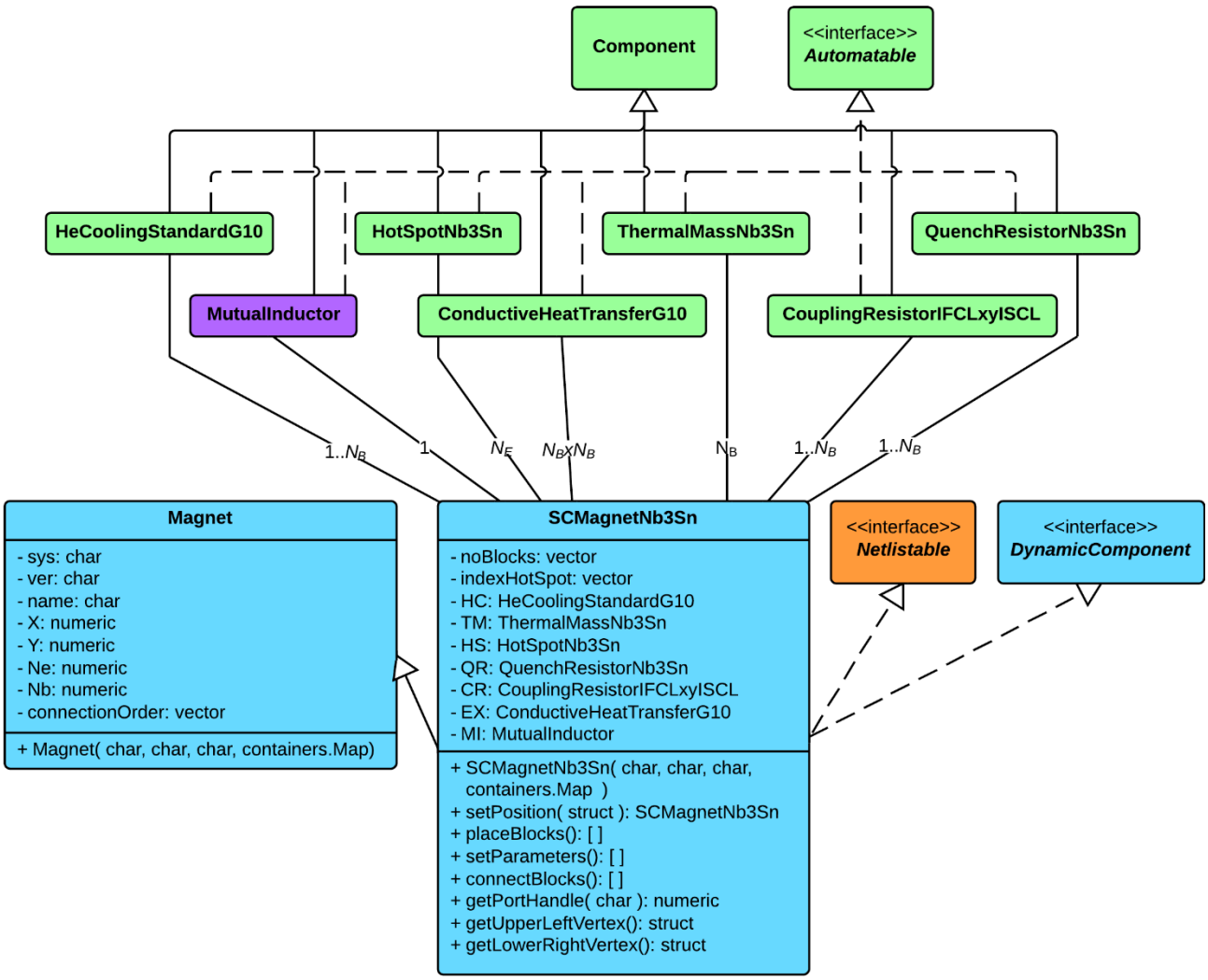


Figure 4.11. UML Class Diagram of Full Model of a Nb3Sn Superconducting Magnet

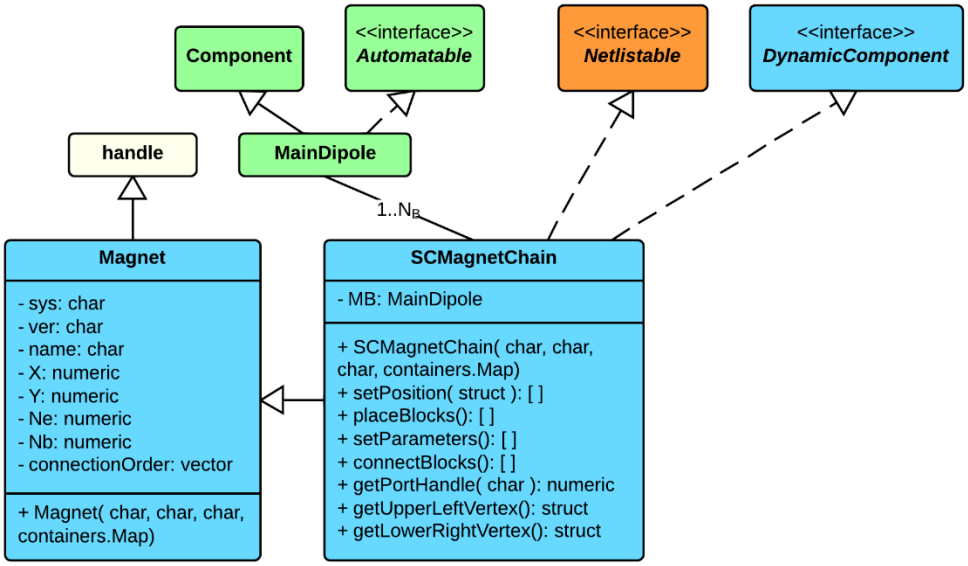


Figure 4.12. UML Class Diagram of **SCMagnetChain**

The last element of dynamically created component library is a quench heater unit (see Figure 4.13). The quench heater unit is composed of a **ThermalMassQH** block modelling thermal mass and resistance of the stainless steel, voltage source, capacitors, and helium cooling. For each

ThermalMassQH object there is at least one **ConductiveHeatTransfer** block representing heat transfer from stainless steel to an appropriate thermal block through the insulation layer of the Quench Heater and that of the block.

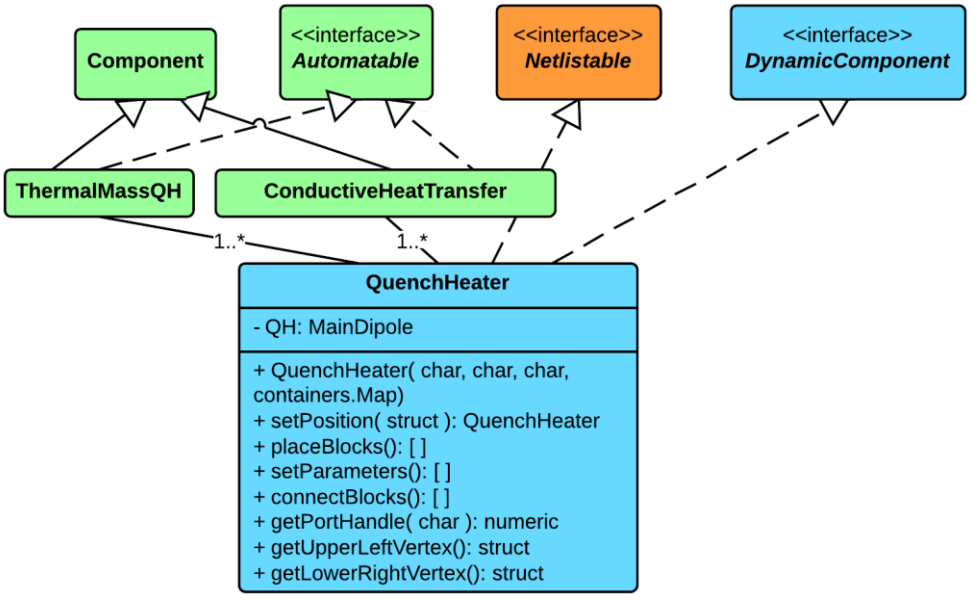


Figure 4.13. UML Class Diagram of a quench heater unit

4.3.6. Model Class Diagram

Finally, the model is composed of three class layers that represent three sub-libraries of the QSF library (see Figure 4.3). The netlist approach has been applied in order to define model structure and connection between sub-components in generic and repeatable manner. Therefore, each block from the QSF library that can be directly added to the model has to implement the interface *Netlistable* (see Figure 4.14). The interface *Netlistable* defines three methods required to build the model automatically by applying the netlist approach. Its working principle is presented in the remainder of the chapter.

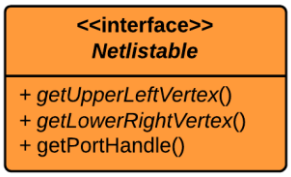


Figure 4.14. Interface Netlistable

To sum up the class hierarchy for the QSF library components is depicted in Figure 4.15. The model may be composed of three groups of components: magnetic elements, magnet protection system, and other components. The figure below depicts how the libraries presented in Figures 4.2 and 4.3 are mapped into the class hierarchy.

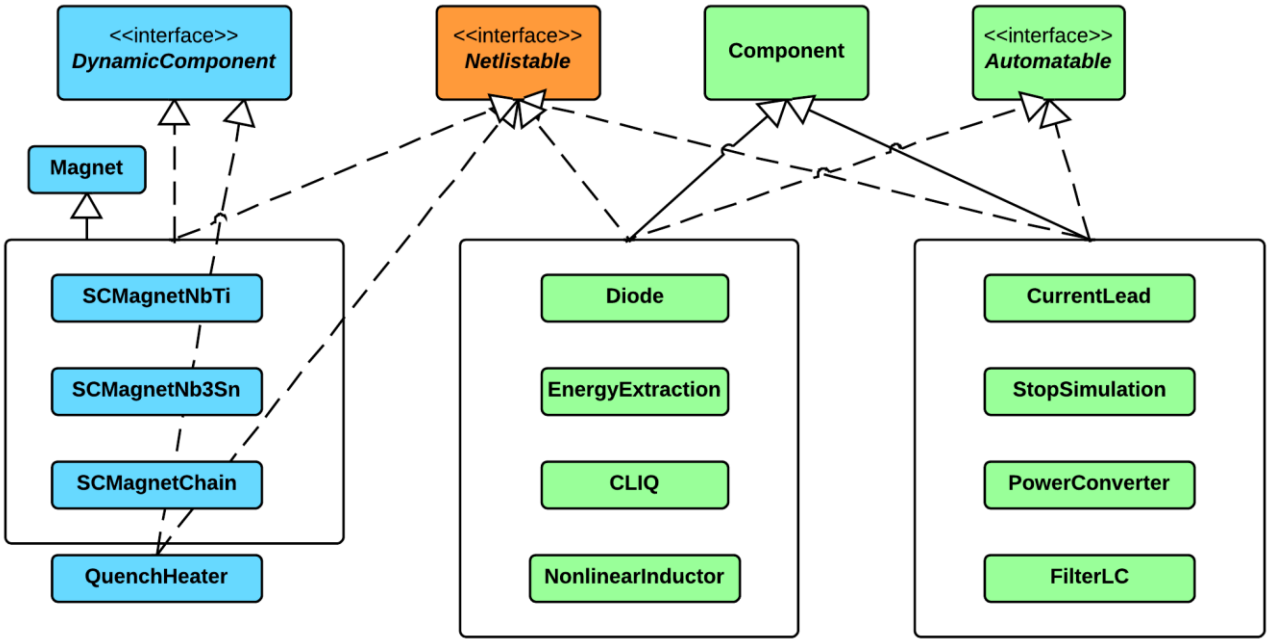


Figure 4.15. Relation diagram of the QSF library classes

Components from the hybrid Simscape-Simulink library are denoted by green and each of them extends **Component** class and implement interface *Automatable*. However, some blocks like **Diode** or **CLIQ** are specific enough and can be added to the model directly, hence their corresponding classes also implement interface *Netlistable*. Furthermore, dynamically generated components (denoted by blue boxes) implement interfaces *DynamicComponent* (see Figure 4.10) and *Netlistable*.

Finally, the **Model** class merges together all aforementioned classes describing the QSF library elements. Namely, as shown in Figure 4.16, it has the following properties:

- *sys* – the name of the Simulink model file,
- *config* – information about the QSF configuration,
- *md* – magnet description stored as a dictionary,
- *configuration* – description of blocks used in particular model,
- *connections* – information on how the blocks in the model are connected,
- *blocksMap* – the dictionary of objects representing blocks used in the model,
- *solver* – an object storing Simulink DAEs solver parameters.

Additionally, the relations between the **Model** class and classes representing components from the QSF library are presented in Figure 4.17. **Model** is a single class required to build the model. In other words its fields contain all information about the model, whereas its methods are capable of building the model automatically.

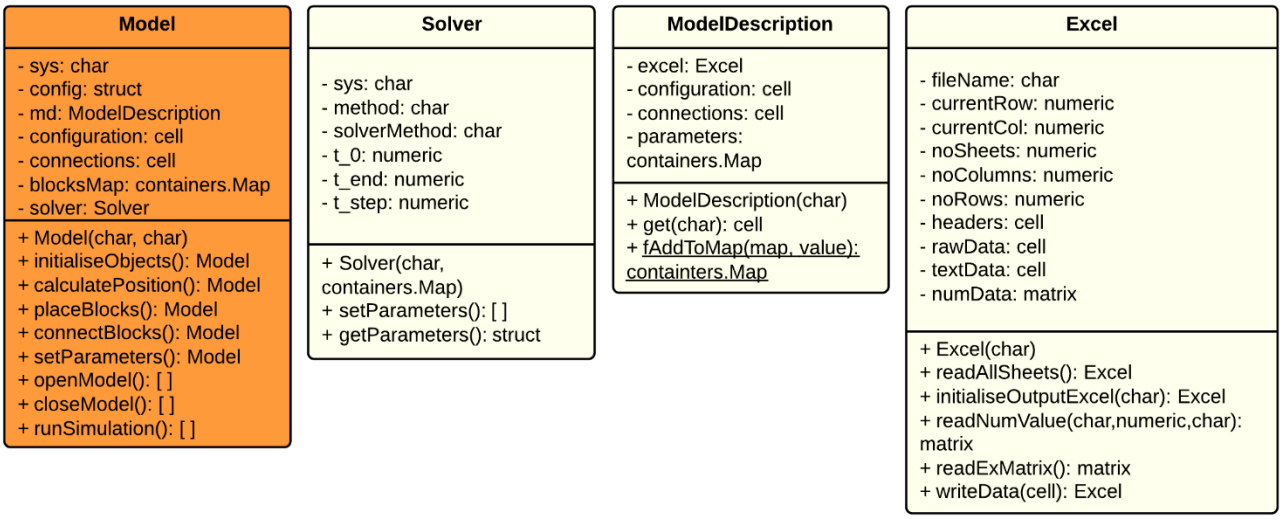


Figure 4.16. UML Class diagram of the **Model** class

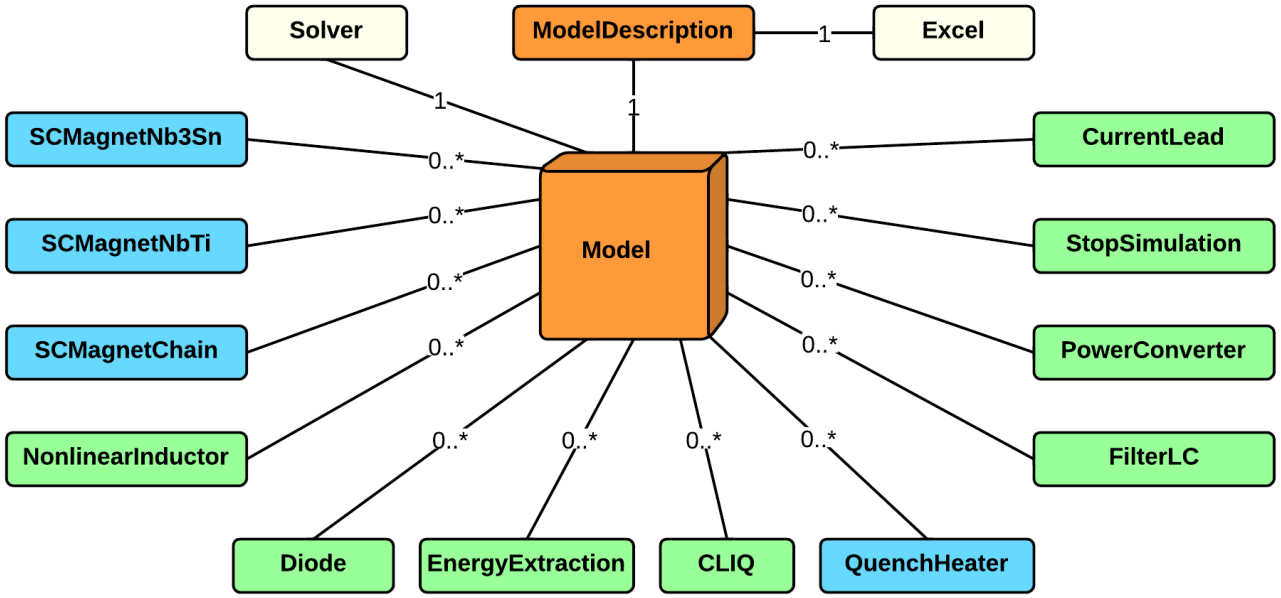


Figure 4.17. **Model** class components relation diagram

This concludes the description of the mechanism applied to treat all building blocks from the QSF library in the identical way even though they have different characteristics. As a result, the modelling problem has been decomposed in a convenient way with clearly defined rules on how to add a new component and how it is described in the code.

4.4. Main Application

The class hierarchy applied to represent the electro-thermal QSF library in the code is only a part of the Simulation Control Module in the Main Application. The class hierarchy serves as a link between bottom and middle layer of the QSF. The QSF main application is composed of five software modules: simulation control, report generation, parallel computing, executable model, and parametric sweep (see Figure 4.18).

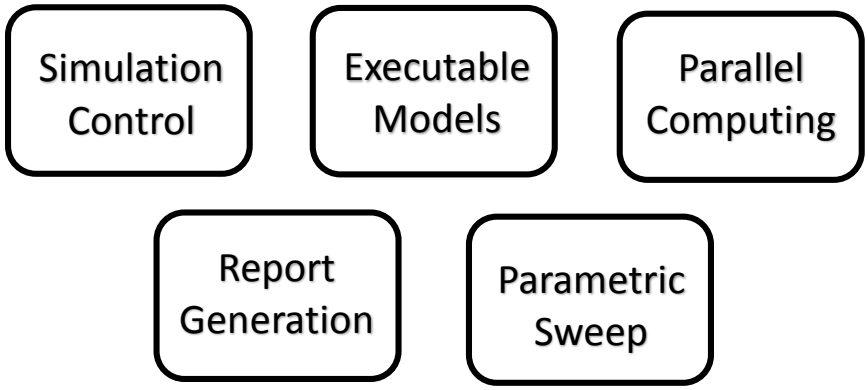


Figure 4.18 Modules of the Main Application

Each of the modules can be used separately, for instance to run a single simulation, create a single executable model or generate summary of simulations as a set of pictures and/or a text document. Alternatively, they can create a sequence of actions. The user can firstly create a set of simulations with varying structures and/or parameters (Parametric Sweep module), then either run a simulation (Simulation Control module) or build a model (Executable Models module) in parallel (Parallel Computing module), and finally observe the results of simulation runs as figures (Report Generation module).

4.4.1. Simulation Control Module

The main purpose of the middle QSF layer, however, is to build the superconducting magnet model automatically. The core of the QSF main application, i.e. the place where all already defined building blocks are connected together, is the Simulation Control Module (SCM). The SCM is responsible for handling all simulation-related tasks including two execution modes: single and multiple simulation. The simplified block diagram of the SCM is represented in Figure 4.19.

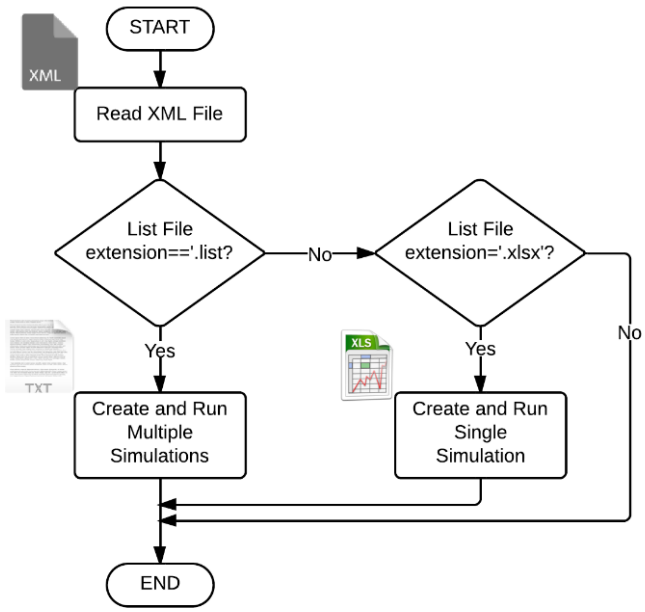


Figure 4.19. Block Diagram of the Simulation Control Module

All information required to run the SCM are stored in the single Extensible Markup Language (XML) configuration file as presented in Listing 4.4. Namely, the configuration XML file defines the current framework version (*version* tag), the locations of the QSF library (*library_path* tag), the QSF MATLAB code (*script_path* tag), the directory with input Excel files (*input_path* tag), the directory for simulation output files (*output_path* tag), the network folder for parallel computing module (*network_path* tag), the file with the list of simulations to run (*list_file* tag), and the Excel file where a summary of the simulations is written (*summary_file* tag). Conveniently, MATLAB has built-in functions to read to and write from XML files.

```
<?xml version="1.0"?>
<simulation_framework>
    <version id="8_3">
        <library_path>G:/Projects/lhccm/Simulink Framework/Ver 8_3/Libraries</library_path>
        <script_path>G:/Projects/lhccm/Simulink Framework/Ver 8_3/Scripts</script_path>
        <input_path>G:/Projects/lhccm/MQXF/Input</input_path>
        <output_path>G:/Projects/lhccm/MQXF/Output</output_path>
        <network_path>G:/Projects/lhccm/MQXF/Output</network_path>
        <list_file>G:/Projects/lhccm/MQXF/Input List Files/files_MB_P13_P24_8.list</list_file>
        <summary_file>G:/Projects/lhccm/MQXF/MQXF_P13_P24_1.xlsx</summary_file>
    </version>
</ simulation_framework>
```

Listing 4.4. Sample framework configuration XML file

The input of the SCM is the value associated with the *list_file* tag in the configuration XML file. The extension of the SCM input file is determining the mode in which the SCM operates:

- single simulation mode, if *list_file* in the configuration XML file has *.xls* extension,
- multiple simulations mode, if *list_file* in the configuration XML file has *.list* extension.

Any other file type is neglected and terminates the simulation control module.

4.4.2. Single Simulation Module

In order to build and run a single simulation the following steps must be executed (see Figure 4.20). Firstly, the **ModelDescription** object within the **Model** class (see Figures 4.16 and 4.17) is initialised on the basis of the input Excel file. Secondly, the objects are created by using Factory Design Pattern. Once the positions of all blocks are calculated, they are added to the Simulink schematic. Next, connections between blocks on the model are created and parameters of each block are set to appropriate values. Finally, the resulting model is simulated and after successful completion it is saved and closed, so that it can be reused in the future. Note that while calling model design methods (calculate position, place blocks, connect blocks, set parameters) the **Model** object refers to particular methods implemented by the classes. Since all derived classes depending on their library type implement common interfaces required to be added to the schematic automatically, it is guaranteed that Model object methods work properly. Light green background in Figure 4.20 depicts steps that are skipped when the model is being updated. Model update (assuming that the model

schematic is identical) only requires to open an existing model, set again parameters with new values, run the simulation, and close the model.

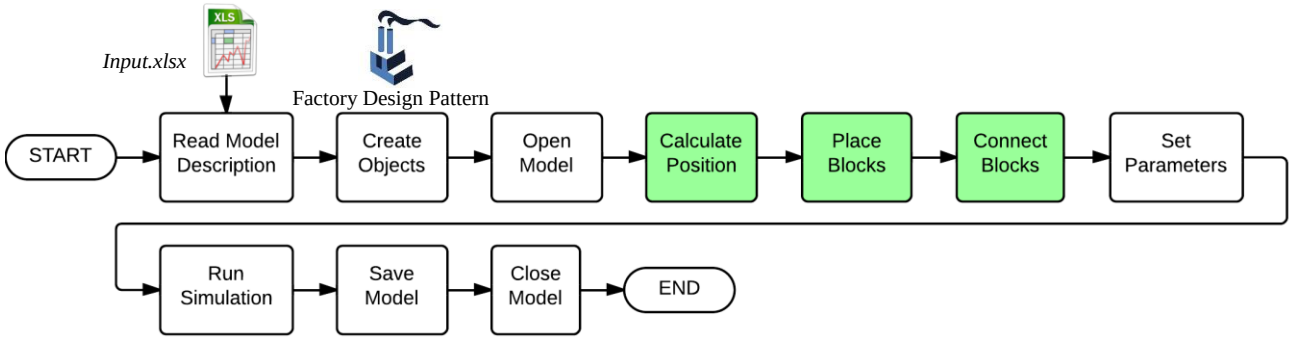


Figure 4.20. Create Model and Run Simulation Procedure

4.4.3. Input Excel File

Spreadsheet file type provides a convenient way of storing tabular data. Hence, this extension was selected for the input file with the entire configuration of the model. As a result, one single file is required to build a model composed of elements from various libraries. The information contained in the file is organised into multiple sheets. The first and second sheets always consist of configuration netlist and connections netlist, respectively. The remaining sheets store parameters of the models of the magnetic elements, magnet protection system, and other components. The configuration of the model – name, position and library component of the blocks used in the model as well as connections between the blocks are represented in terms of netlist approach [36].

4.4.4. The Netlist Notation - Configuration

Methods defined in the interface *Netlistable* are necessary to apply the netlist notation to describe elements used in the model as well as connections between them. Since high flexibility of the framework and ability to automatically adapt to various simulating scenarios are required, the model design procedure must include information on how the blocks are to be named, modelled, and positioned on the schematic. The concept applied in the QSF will be explained with the aid of the sample superconducting circuit schematized in Figure 4.21.

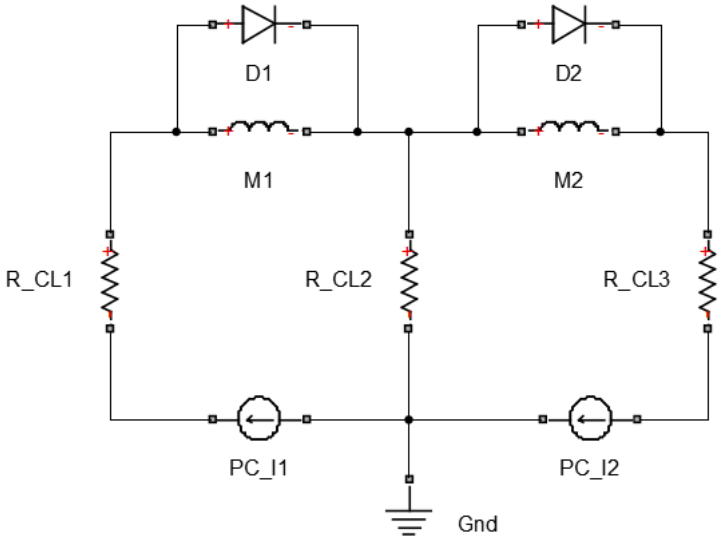


Figure 4.21. Sample schematic of a superconducting circuit

The following algorithm has been applied to save information about blocks position in the spreadsheet in a unified way applicable to every element of the QSF library (Table 4.1). The first block in the list always has fixed coordinates on the schematic, predefined as the centre of the schematic editor, whereas a reference block and a cardinal direction must be specified to calculate the position of each following block. In other words, the position of each consecutive block is calculated based on the position of existing blocks and shifted as indicated by the chosen direction (NEWS). The proposed two-element notation represents the smallest set of information required to add a new block to the schematic.

Name	Library	Relative To	Direction
M1	SCMagnetNb3Sn	none	none
R_CL1	CurrentLead	M1	SW
PC_I1	PowerConverter_I	M1	S
D1	Diode	M1	N
R_CL2	CurrentLead	M1	SE
M2	SCMagnetNb3Sn	M1	E
R_CL3	CurrentLead	M2	SE
PC_I2	PowerConverter_I	M2	S
D2	Diode	M2	N
Gnd	Electrical Reference	PCI_1	SE

Table 4.1. Configuration Netlist of the sample superconducting circuit shown in Figure 4.21

The full electro-thermal model with fine spatial resolution consumes a large area on the Simulink model schematic, thus it may be difficult to locate a particular element in case of need. However, it is possible to locate any block based on the component netlist if model investigation is required. Let *dx*, *dy*, be position increase in the *x* and *y* direction, respectively. Those increases are calculated as a width and height added to a fixed value. Additionally, x_{ul}, y_{ul}, x_{lr}, y_{lr}, are the *x* coordinate of upper left vertex of a reference block, the *y* coordinate of upper left vertex of a reference block, the *x* and *y* coordinate of lower right vertex of a reference block, respectively. Moreover, all those parameters can be deduced from the **Component** class (see Figure 4.6). Then, actual

mechanism to calculate the position of a component in the *x* and *y* direction based on reference block position (depicted in violet) and cardinal directions (NEWS) is presented in Figure 4.22.

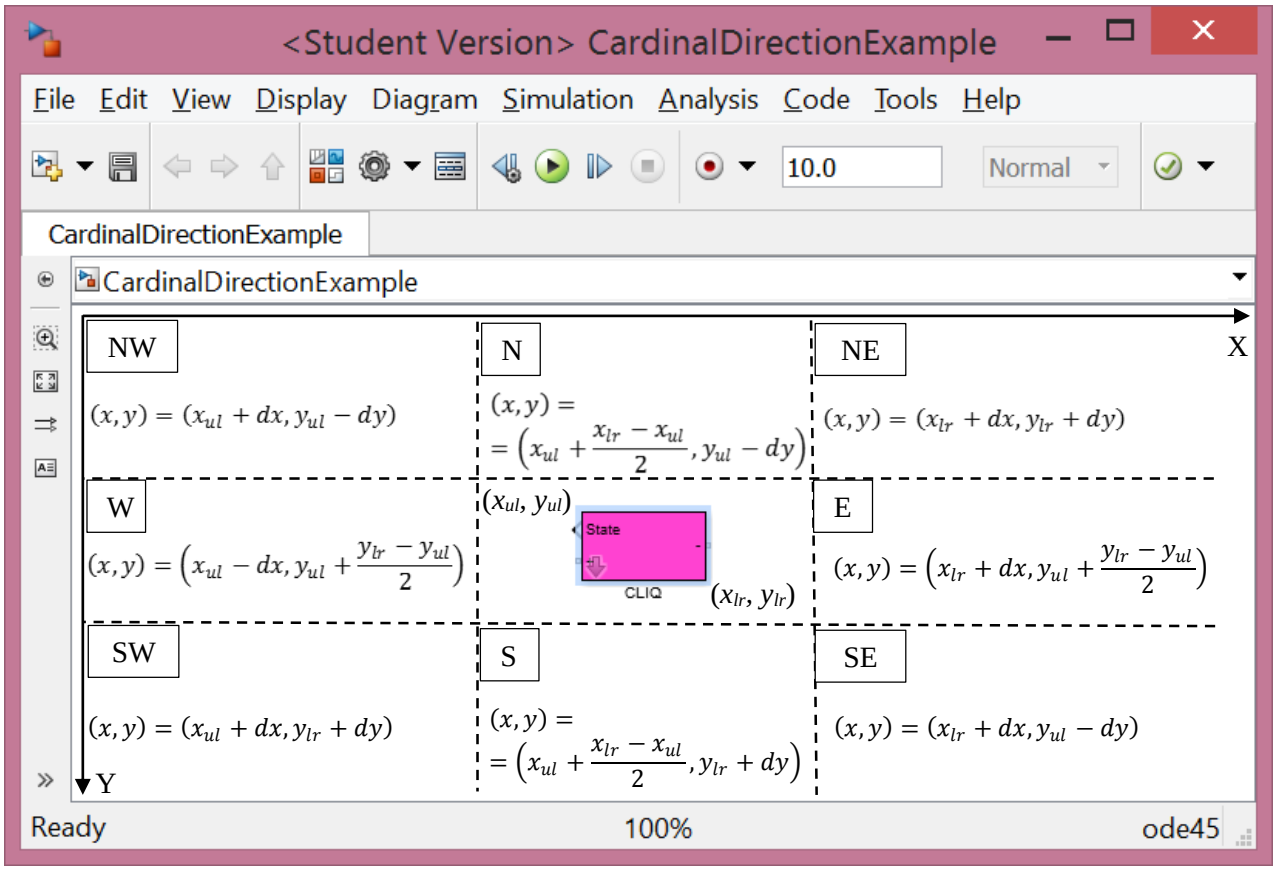


Figure 4.22. Calculation of block coordinates applying cardinal directions.

4.4.5. The Netlist Notation - Connections

The second stage of the netlist configuration consists of connections definition between existing blocks. This step only requires the definition of four parameters, namely from-block name, from-port name, to-block name, and to-port name. This concept is widely known in simulation software (for example Spice) as a netlist description. Listing 4.5 presents the netlist corresponding to the sample circuit shown in Figure 4.21, composed of two superconducting magnet elements, each by-passed by a diode, two power converters, three current leads, and a ground connection.

From the electrical point of view, every block can be considered as a box with positive and negative nodes. Such an observation applies to simple resistors or voltage source as well as to complex, electro-thermal superconducting magnet model. In fact, even complex components only have two floating electrical nodes (positive and negative).

```
1  connect(PC_I1/pos,R_CL1/neg)
2  connect(R_CL1/pos,M1/pos)
3  connect(M1/neg,M2/pos)
4  connect(M2/neg,R_CL3/pos)
5  connect(R_CL3/neg,PC_I2/neg)
6  connect(PC_I2/pos,PC_I1/neg)
7  connect(PC_I2/pos,R_CL2/neg)
8  connect(R_CL2/pos,M2/pos)
9  connect(D1/pos,M1/pos)
```

```
10  connect(D1/neg,M1/neg)
11  connect(D2/pos,M2/pos)
12  connect(D2/neg,M2/neg)
13  connect(PC_I1,Gnd)
```

Listing 4.5. Connections netlist

4.4.6. Factory Design Pattern

Information obtained from the configuration sheet is fed to the method implementing Factory Design Pattern that instantiates all objects present in the netlist [1]. As one can notice from Figure 4.17, there is a large amount of possible superconducting circuit model configurations. Moreover, the exact model structure is not known *a priori*. Hence, the **Model** class must be equipped with a mechanism to instantiate only the objects needed for a particular simulation. Since all elements of the QSF library are represented by the classes sharing the same interfaces (*Netlistable* in particular) in the main MATLAB application, the Factory Design Pattern is a natural solution for this problem. Factory Design Pattern provides a mechanism to efficiently create variable-structure models as the user may define model configuration composed of different protection schemes. The working principle of the Factory Design Pattern is fairly straightforward. It includes looping through all elements from the configuration netlist and adds an appropriate key and value to the dictionary data type (in MATLAB as in Java it is a Map data type - *containers.Map*). The key is the name of the component and the value is an object of the class associated with that name. Next, the object is initialized with all block properties. During the creation of the objects, each class constructor checks whether its list of parameters agrees with the list of properties in the mask of the component.

Since all blocks from the QSF library implement the same interface (*Netlistable*), they have the same list of identical functions. As a result, all blocks are treated in the same way and this reduces the possibilities of errors during the operation of the SCM. At the same time, the application of the Factory Design Pattern guarantees high flexibility of the framework.

4.4.7. Creating Model Procedure

Once the objects have been successfully initialized by applying the Factory Design Pattern, the actual model schematic can be created in the Simulink following the block diagram shown in Figure 4.20. Hence, the first step is to open a template Simulink model with properly set environment parameters and save it under a new name (it is assumed that the model name is identical to the name of the input Excel file). Next, the **Model** class method to calculate the positions of each block is executed. Then, each block in the dictionary is equipped with methods to calculate its position in its own way. The *calculatePositions()* method loops through all objects stored in a dictionary and calculate positions based on the proposed configuration approach (cardinal directions). For that purpose each block class must be equipped with a proper implementation of the functions *getUpperLeftVertex()* and *getLowerRightVertex()*. In fact, each class for the QSF library objects

implements the interface *Netlistable*. Hence, it is guaranteed that the *calculatePositions()* method always performs as expected independently of the blocks included in the schematic.

After calculating the positions of all blocks, they can be placed on the Simulink schematic by calling the *addBlock()* method. This method may also include connecting compositional blocks together in the case of components created automatically.

The next stage includes adding connection between all stand-alone blocks in the model, i.e. the blocks from the configuration netlist. Here another important method from the interface *Netlistable* is used. Namely, in order to connect two blocks together handles to particular ports in the block are required. Besides, for a given port name an associated handle needs to be returned.

To sum up, each element from the Simscape-Simulink library and Mutual Inductor Simscape library can be added to the schematic by inheriting from **Component** class and implementing the *Automatable* interface. Then it can either become a part of a dynamically created component or it can be directly added to the schematic. In the former case, a dedicated class implements a *DynamicComponent* interface, hence it implements methods that build a model out of hybrid components. These methods call functions from hybrid components required by the *Automatable* interface. Thus, both stand-alone blocks from the Simscape-Simulink library and dynamically created components implement the *Netlistable* interface with functions required to build a model from the netlist.

In conclusion, the creating model procedure may include both dynamic component development (low-level operations) and actual model development (high-level operations). The set of low-level operations is strictly defined for each dynamic component and the number of its sub-elements may change. On the other hand, for the high-level operations neither configuration nor the connections between standalone blocks are predefined. However, due to the implemented hierarchical class design with base classes and interfaces the SCM is capable of handling either case.

For instance the result of the create model procedure in case of MQXC2 (explained in detail in Section 6.1) magnet model composed of 8 electrical parts equally subdivided into 25 blocks per each part (N_E=8, $\mathbf{n}_B$=[25 25 25 25 25 25 25 25], N_B=144) is a simulation composed of over 3000 sub-components from the QSF library described by more than 20000 parameters.

4.4.8. Multiple Simulation Mode

The developed well-structured model design procedure opened an easy way to implement a functionality of automatically executing sets of simulation in a sequential order. The working principle of the multiple simulation mode of the SCM is schematised in Figure 4.24.

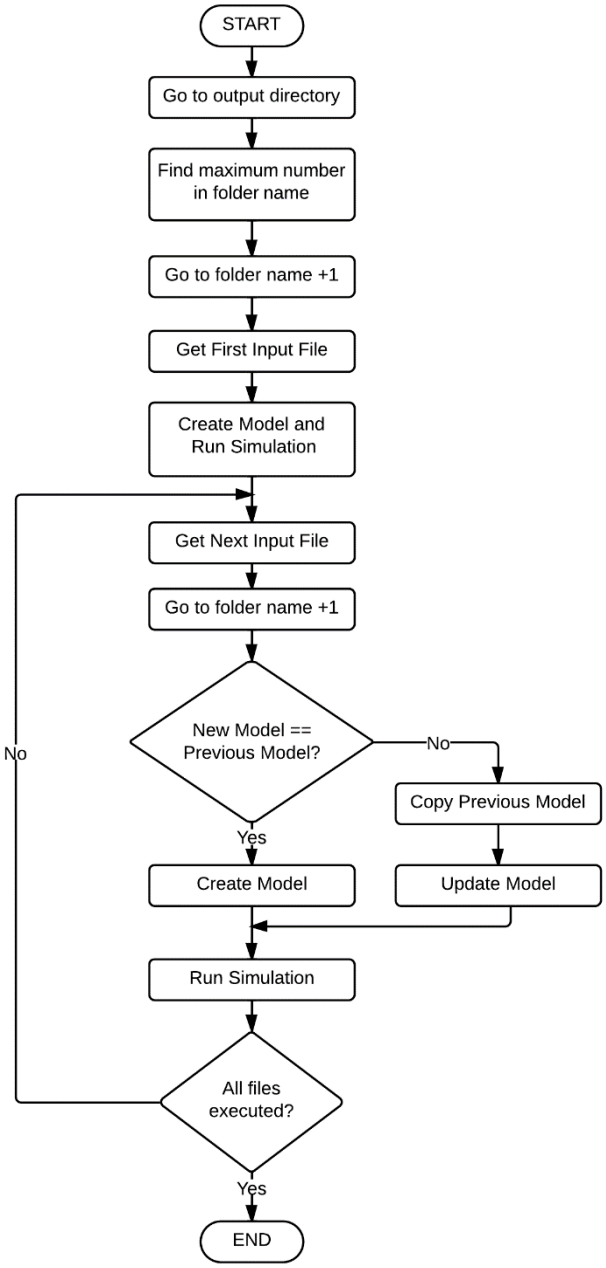


Figure 4.24. Block diagram of the multiple simulation mode of the Simulation Control Module

If the property *input_file* in the configuration XML file is set to a file name with *.list* extension, the QSF is executed in multiple simulation mode. The input of the multiple simulation mode is a text file with a list of simulations to be executed, as shown in Listing 4.6.

```
1  nameModel_1.xlsx
2  nameModel_2.xlsx
3  nameModel_3.xlsx
4  nameModel_4.xlsx
5  nameModel_5.xlsx
6  nameModel_6.xlsx
7  …
```

Listing 4.6. Sample Simulation List File

The multiple simulation mode always begins with building and executing the first file from the list. In order to build a model, the function firstly goes to the output directory defined in the configuration XML file. Then, it loops through all file names and finds the highest number in the

name. It is assumed that the results are saved in folders with numerical names. As a result, the output folder is well organized as each simulation has a separate location with a unique name. Furthermore, for each new simulation, once the highest number is found, it is incremented by 1 and becomes the name of a new folder. Next, a copy of the input Excel file is stored in the newly-created directory for the current simulation and the model is created (see Figure 4.20) and run.

If the list has more than one input file, the SCM may either create again a new model, or reuse an existing model from previous simulation runs, simply updating its parameters. For this purpose, the equality operator for **Model** class has been overloaded, hence it can be used to compare two models together to decide if the model can be updated. Therefore, if either model configuration netlist or connections netlist has changed, the overloaded equality operation should return value false. As a result, model in the list of simulations can be reused many times as long as a previous file is the same. Hence, the SCM has been equipped with a simple, two stage pipeline. The first stage of the pipeline is creating the model, whereas second stage is setting model parameters. Updating a model may provide significant time saving; as an example, the time required for the development and simulation of a model containing a 144-block superconducting magnet can be reduced by about 20%. On the contrary, every change in the model in the list structure results in cleaning the pipeline and rebuilding model again.

4.4.9. Other Modules

The QSF, apart from handling automatic simulation design and execution, is also equipped with modules supporting those tasks.

1. Report Generation Module

   Since the superconducting magnet model is composed of many sub-systems characterized by different parameters and behaviour, graphical representation of the obtained output is of a great help while analysing simulation results. First of all, dedicated MATLAB scripts were developed to visualise time-series plots and 2-D profile of the most relevant signals in the various sub-components of the system, such as current, voltage, electrical resistance, temperature, Ohmic and coupling losses. Next, all resulting graphs are saved into separate output directory. While plotting signals in 2-D, the output is also saved as an animated GIF file, hence the user can observe the time evolution of particular signals (for example the temperature in the different parts of a magnet). Moreover, parametric sweep study allows to check system performance under varying operating conditions. Hence the report generation module is also equipped with a procedure dedicated to compare two simulations signal after signal in order to identify differences.

Furthermore, during experimental campaigns testing quench protection systems it became important to compare together simulation and measurement results. Additionally, the measured data is saved by a LabVIEW application in a text format, hence specific procedures for loading and managing large data sets have been developed.

Another important aspect is to provide a tool to automatically summarise simulation and verify correctness of obtained results. For this purpose, detailed text document is generated for each simulation, containing system parameters and plots of the signals. Moreover, the report generation script recalculates dependant signals like absolute magnetic field and material properties from base signals like current and temperature. Once those signals are derived, they are compared with actual signals obtained from the simulation.

However, in some test scenarios only quench protection performance measures are required. Then, the QSF stores information about the quench load, the hot-spot temperature, and other key simulation results.

Finally, there is also a sub-module responsible for informing by e-mail the user when the simulation set has executed and all data is available in the output directory.

2. Executable Models

   The Simulink has a Real-Time Workshop (RTW) extension that allows translating a Simulink model into executable code [5,6]. The RTW can be set to execute the code either on the computer that is running the Simulink or on the embedded controller. In the former case, simulations should perform faster as wealth of functionalities provided by the Simulink schematic window is disabled, and simulations are executed from the MATLAB command line. However, in the latter case it is possible to perform hardware in the loop studies (HIL). Once the model is compiled into executable file, it communicates with the user through *.mat* files. In fact, the parameters of all used blocks are passed to a *.mat* file when the simulation starts and the results are returned in the same format once the simulation is completed. Furthermore, such a mechanism only applies to elements from the standard Simulink library and is not supported for the Simscape components [12]. However, this limitation can be overcome by passing those parameters as inputs to the Simscape blocks. Moreover, such a solution has been efficiently applied by extending only a *addComponent()* and *setParameters()* methods in classes describing the hybrid Simscape-Simulink components library (see Figure 4.8).

3. Parametric Sweep Module

   The SCM capability of automatically executing sets of simulations in a sequence allowed implementing the parametric sweep functionality. For this purpose, the parametric sweep module has been developed and allows generating input Excel files based on user's selection of parameters and corresponding values. The parametric sweep analysis allows the user to

assess how the performance of the system is affected by the one of its parameters, such as the RRR of the copper stabilizer of a superconducting cable. Using this feature, the user can achieve an enhanced understanding of the dynamics and overall performance of the modelled system.

Furthermore, it is also possible to perform a frequency sweep analysis in order to calculate the frequency Bode plots of the system as well as Monte Carlo analysis (especially with executable models [6]). Parametric sweep study may be also extended and include tests of various superconducting model configurations (protection schemes and superconducting magnets circuits). To sum up, the framework offers wider range of modifications to the model under the term "parametric sweep" than other software.

4. Parallel Computing Module

The last module of the main application allows executing simulation sets in parallel. Nowadays, the computing power of computers increases and especially new processor architectures allows executing tasks in parallel (multi-core processors). The Parallel Computing Module handles all tasks related to multi-core, multi-machine simulation execution. First of all, the QSF makes it possible to run multiple simulations in parallel using the same framework version. Furthermore, simple single master-multi slave architecture has been implemented. The communication between master and slave is resolved by writing to and reading from the common folder. The master is a MATLAB session running the main application of the QSF; each of the slave sessions runs a dedicated MATLAB script that creates a folder whose name is the slave identification number (ID) and sends to the master session information on the slave availability. The master is capable of periodically checking the shared folder to update the list of available slaves and sending simulation commands to selected ones through the shared network folder (see Figure 4.25). The communication between master and slaves has three types of messages that are sent through dedicated files:

- *config_i.xml* – message from master to slave with information on simulation set to be executed; once received, the slave executes the SCM and the input configuration file is *config.xml*;
- *simulation_i.running* – message from slave to master on the current state of a simulation. The file is updated after every completed simulation, hence provides the master with information on the progress of the simulation set;
- *simulation_i.finished* – message from slave to master confirming that the simulation set has executed and slave is ready to accept a new configuration file.

The PCM allows designating more than one simulation set on the selected slave. In other words the user can define more than one simulation set for a slave that will execute one after

another. For that reason, an *i* suffix in the name of each file described above accounts for additional simulation sets to be executed on the selected slave, and provides information on the history of that particular slave. In order to avoid errors when either master or slave is writing information to the file, each message file is confirmed by the same name as an edited file with the .*saved* extension.

The proposed architecture does not distinguish between the slave that is running on a remote machine or on the computer running the main application. In other words, if the machine is not connected to the network it is possible to run several other QSF sessions on the local machine, whereas for network computers slave can be run on any PC.

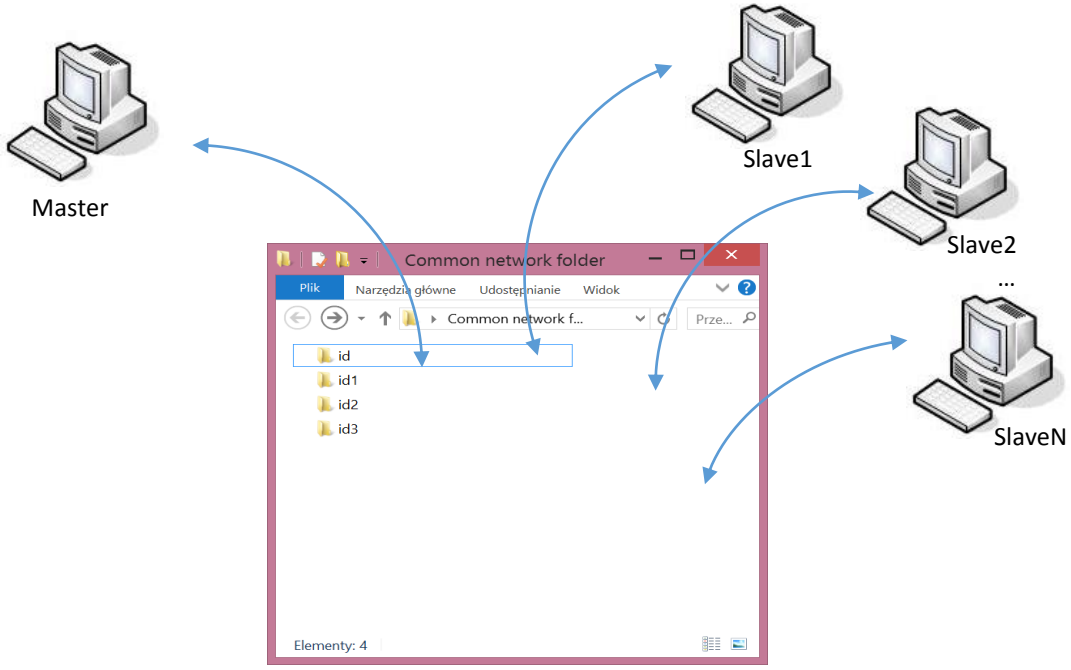


Figure 4.25. Parallel computing module working principle

As a result, the equivalent simulation execution time greatly decreases with increase of running MATLAB slave QSF sessions. Currently, the PCM supports only single-master multiple-slave operation mode, however it can be easily extended to multiple-master multiple-slave operation.

5 Graphical User Interface

The ability to build models automatically with great ease is indeed an interesting and convenient feature. However, without the GUI, the QSF would become difficult to use and maintain. The GUI not only calls existing modules and functions but also creates a closed form of the entire simulation framework. Although the input spreadsheet file can be created manually in any available editor, it is a tedious and prone to error task. Hence, the first and most fundamental goal of the GUI is to develop an intuitive, informative and aesthetically pleasant interface between the modeler and the QSF. This interface must replicate all operations performed in the spreadsheet file editor and modules of the QSF main application, such as defining model structure and properties. Moreover, preparation of simulation list file (parametric sweep), simulation execution and control as well as output data visualization must be supported by the GUI. Naturally, it is possible to perform each of the tasks separately, for example the modeler can load existing file and run simulation (without the need to redefine configuration and parameters), or plot signals from previously executed models. Furthermore, the GUI provides a guidance and a feedback during the model design (such as parameter range check, electrical schematic visualization and structure analysis), hence it further improves user experience with the simulation software.

MATLAB language, as most of high-level programming languages (Java, C#, etc.) is provided with a set of event-driven graphical elements. In general, those elements are subdivided into two main classes:

- *uicontrols* (user interface controls) – a group of controls with the purpose of either performing an action or presenting information. This group consists of the following controls: check box, editable text field, pop-up list, list box, push button, radio button, toggle button, and slider. Additionally, a frame to group several controls together is also available [23].
- *uimenus* (user interface menus) – it allows to add to the GUI a custom, user-defined pull down menu such as the familiar menu bar at the top of application windows in Windows and Mac-OS operating systems [23].

Moreover, the *uicontrols* and *uimenus* can be combined with other graphical elements in MATLAB like plots in order to build a more sophisticated GUI. Additionally, MATLAB, by default, allows the user to employ graphical Java elements from the Swing library, hence it greatly extends the list of available GUI building components and enables reusing existing code written in Java [24]. To sum up, a GUI can be developed by applying two, fundamentally different approaches:

- low-level approach (in literature it is also known as a bottom-up approach) which assumes designing the window layout as well as implementing functions associated with the controls by writing appropriate scripts or classes in MATLAB. Such an approach guarantees complete control over the created GUI.
- high-level approach (top-down approach) which employs a Graphical User Interface Development Environment (GUIDE) editor. The GUIDE enables creating a GUI layout in a convenient window by dragging desired objects from the controls palette and dropping them in the desired position on a layout. Next, the developer only writes the callback functions associated with elements present in the GUI.

The front-end user interface of the QSF has been developed by writing appropriate classes in the MATLAB object-oriented language and is composed of both standard *uicontrols* and *uimenus* along with Java objects (low-level approach). However, the GUIDE has been applied in the first phase of the GUI development in order to quickly assess and validate various concepts (high-level approach).

5.1. GUI Architecture Definition

In order to conserve space on the window, the GUI employs tabs to display different information to the user at the same time. Furthermore, each module (see Figure 4.18) is associated with a separate tab. Therefore, the majority of the window is devoted to the content of each tab, and the bottom part is occupied with buttons to select the active tab as presented in Figure 5.1. Status field displays to the user all relevant messages about the performed actions from the GUI level. As a result, the user is not exposed to too much information but can focus on a single task. The tabs allow efficiently reusing the window space by displaying appropriate content according to the selected tab. The GUI of the developed simulation framework is composed of six main modules:

1. Home
2. Schematic Designer
3. Parameter Editor
4. Parametric Sweep
5. Simulation Control
6. Signal Viewer.

Each tab is responsible for a separate task and the order of the tabs (from left to right) represents the natural flow of data and information while creating and running simulations. The user can, however, skip any of the tabs according to current simulation needs. The GUI should be clean, easy to understand, and intuitive to use. For this purpose, the designed GUI selectively replicates good

practices from existing simulation software. In other words, the GUI (as well as the entire framework) is tailored to satisfy the needs of superconducting magnet modelling.

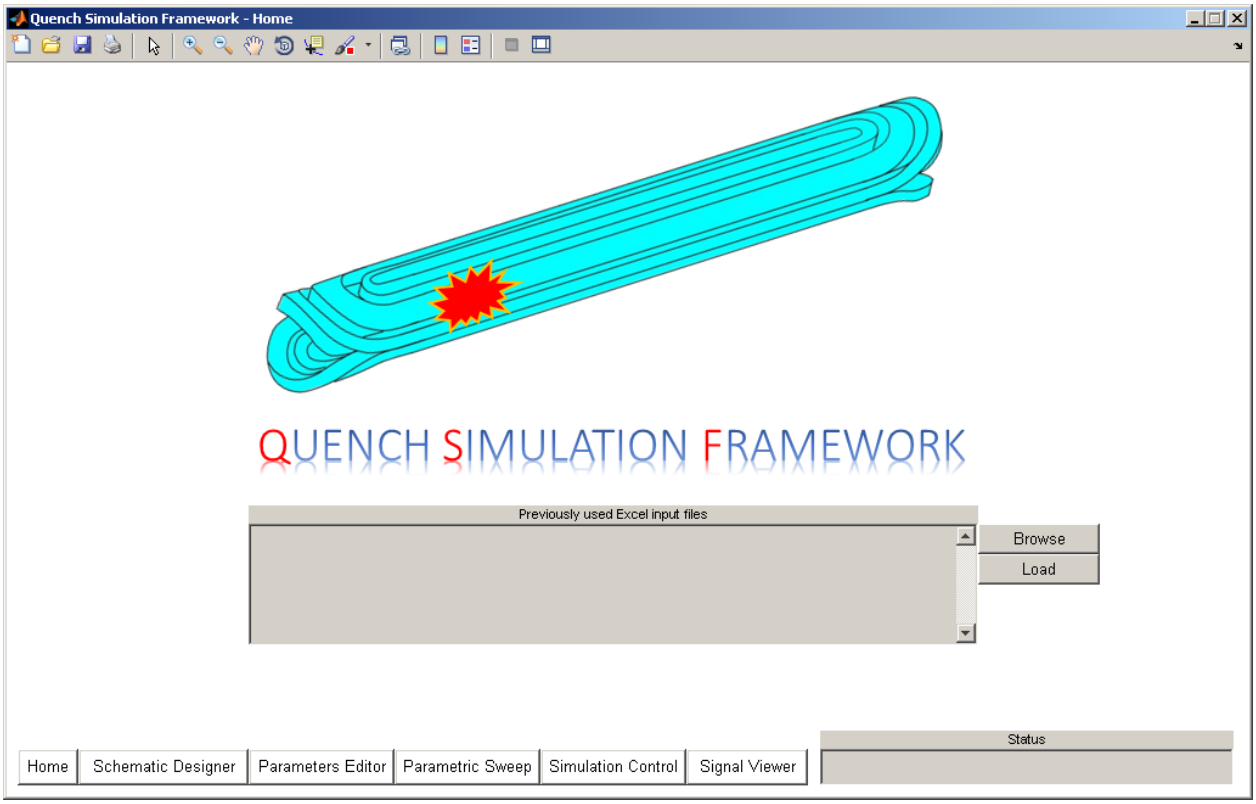


Figure 5.1. The GUI of the Quennch Simulation Framework with active Home tab

For each element of the GUI a dedicated data type (a class) has been designed and implemented (see Figure 5.2). As a consequence, each tab can be developed separately. Next, the **MainGUI** class links together all classes including both elements of the GUI and classes needed to store information about the model designed by the user. Finally, the singleton design pattern ensures that from the current MATLAB session only single GUI window can be opened [1].

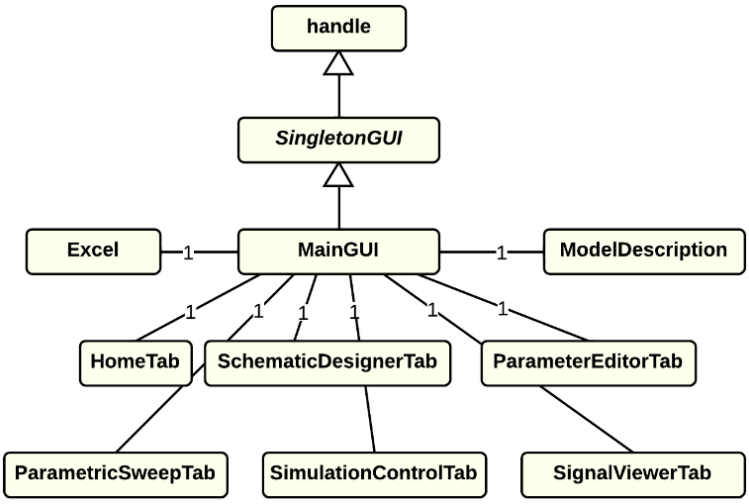


Figure 5.2. UML Class Diagram of the GUI

5.1.1. Singleton Design Pattern

In order to guarantee that there are no file access errors during the model development, the model input files must be created, modified, and saved from a single GUI instance. Hence, it is desirable to guarantee that the **MainGUI** class has only one instance. For this purpose, the singleton design pattern has been implemented. The singleton pattern provides a global point of access that allows checking if the class has been instantiated already. As a result, the GUI can be instantiated only once, and every following constructor call returns that instance.

5.1.2. UML Sequence Diagram

The UML sequence diagrams is employed to represent possible user actions and how the GUI responds, along with the flow of data between the elements of the user interface (more detailed information on UML diagrams can be found in [22]). In general, the diagram is organised as follows: the horizontal axis consists of all relevant objects in a particular case, whereas the vertical axis displays actions happening in time and corresponding messages passing from one object to another – so called life-cycle axis. Usually, the objects are grouped into the four stereotype categories presented in Table 5.1.

Icon	Stereotype Description
 Actor	Actor is an external object to the following objects. In the description of the GUI, the user is an actor.
 Boundary	Boundary object represents an interface between internal and external objects. Hence, here the GUI serves as a boundary stereotype.
 Controller	Controller object is dealing with all tasks related to the management of one or many classes.
 Entity	Entity depicts all data to be stored in the system (for example a database, a text file, a spreadsheet).

Table 5.1. List of icons of the most common objects used in UML sequence diagrams [22]

Since MATLAB does not support tabs in the graphical user interfaces, the following solution has been applied: the classes representing tabs are equipped with a method changing their status from visible to invisible, and vice versa. The life-cycle of tab management procedure is presented as a sequence diagram in Figure 5.2. Once a particular tab is selected, the **M_GUI** class switches off the visibility of all other tabs and turns on the visibility of the selected one. Additionally, while setting the visibility to on, a content of the selected tab is reloaded.

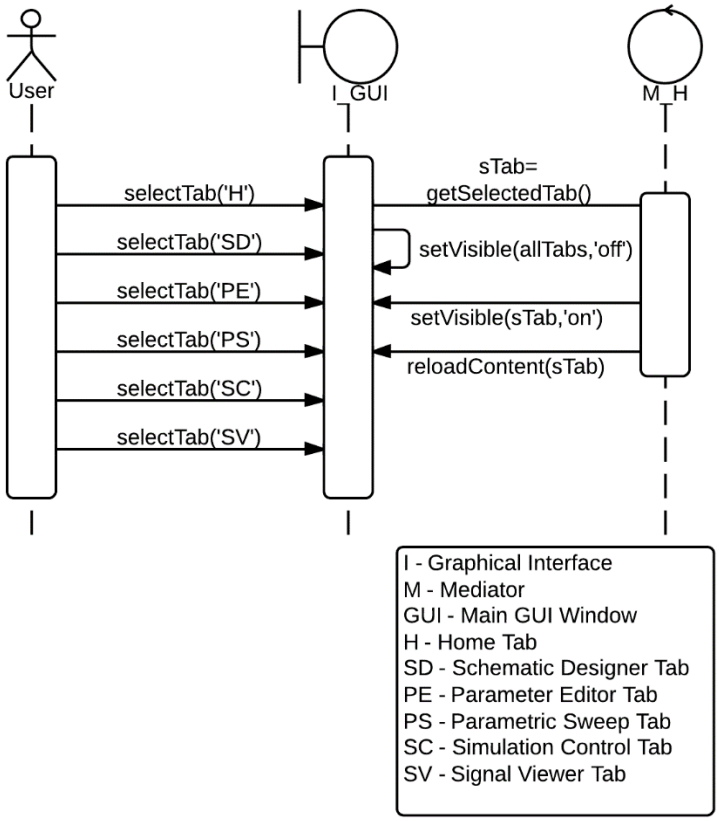


Figure 5.3. Sequence diagram of the main GUI handling selection of an active tab

5.1.3. Mediator Design Pattern

From Figure 5.3 one can clearly see that even handling tab management requires some exchange of data between separate classes representing all tabs in the GUI. Moreover, each tab also has to manage the communication between the internally applied graphical controls. Hence, mediator design pattern has been applied to efficiently handle this problem [1]. Thanks to this solution, instead of updating the state of each relevant control while a callback[6] is called, each control registers the state change event in a **Mediator** object. As a consequence, the **Mediator** object receives all information about the state of the GUI elements and distributes tasks (by calling appropriate callbacks) to all controls that require update. In other words, the **Mediator** class serves as a hub around which all controls revolve. Furthermore, such an approach greatly reduces the code complexity as there is only one class to be modified while any change is required, and only one class that handles the communication between all GUI controls.

To sum up, each class representing a single tab in the GUI consists of a graphical interface object (denoted by I on the sequence diagrams) and a mediator object (denoted by M on the sequence diagrams).

[6] A callback in MATLAB is a function executed after the state of a control has changed

5.2. Home Tab

At the opening of the GUI the user is welcomed by a so called "splash screen" (see Figure 5.1). The Home tab is composed of the QSF logo and a list of twenty previously opened input files. The list is stored in a separate text file and is loaded while the splash screen becomes active. The window allows the modeler to either select an existing input spreadsheet file, or start working with a blank file. Furthermore, it is possible to select one of the recent input files presented in the recent files list (*Previously used Excel input files*), or load an existing one as schematized in Figure 5.4. The user can browse for an input spreadsheet file or select and load an existing one from the recent files list by clicking the *Browse* or *Load* buttons, respectively. In the case of missing file from the list on the disk an appropriate message is displayed. In turn browsing and selecting a new file updates the list of the recent input files. If neither an existing file is loaded nor a previous one selected, the QSF is initialized with a blank file.

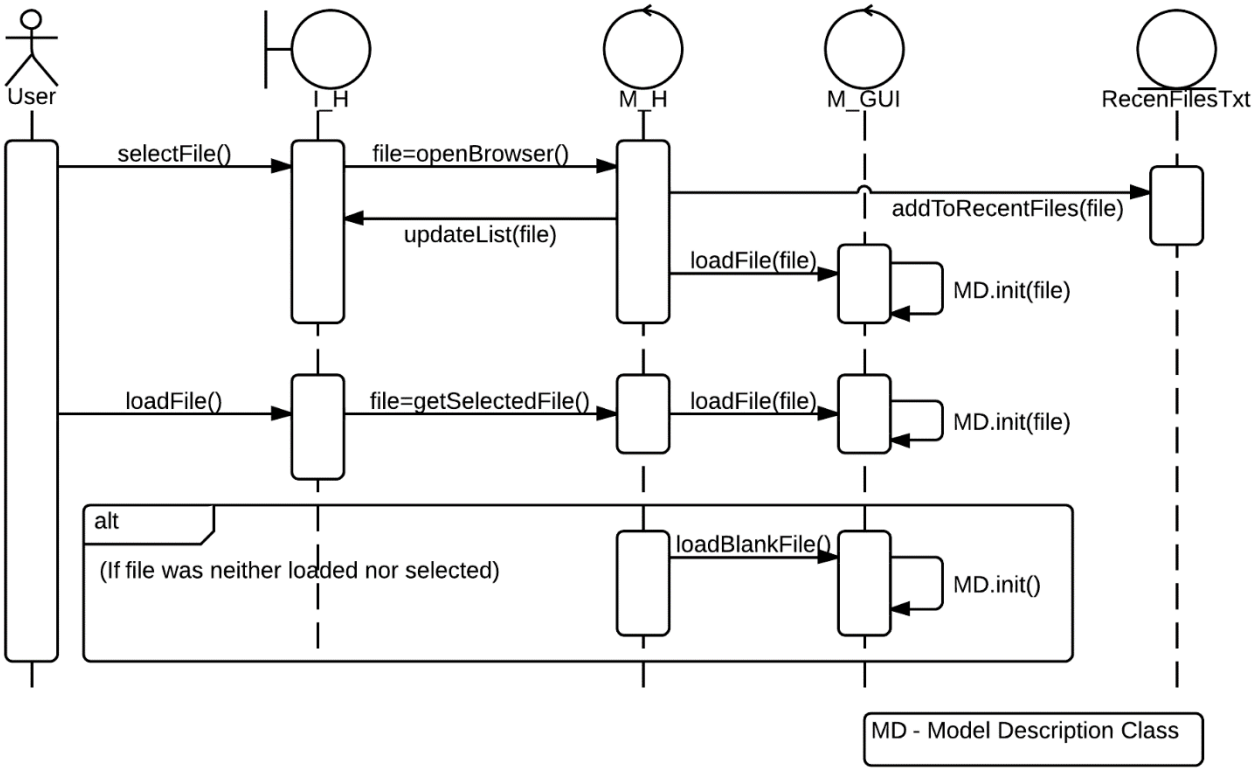


Figure 5.4. Sequence diagram of the Home tab

5.3. Schematic Designer Tab

The second tab of the GUI serves as an editor of the electrical schematic for superconducting magnet circuits. The main part of the window is covered by a plot displaying the components added to the schematic and the connections between them (Figure 5.5). Hence, the Schematic Editor tab is a convenient way of defining the model structure by applying the proposed netlist approach for model configuration and connections (see Section 4.5.4). Every modification of the two netlists is

automatically reflected on the schematic, hence the modeler is able to analyze the model configuration in a convenient way, in a similar manner as while drawing a schematic on the paper. Moreover, the GUI employs built-in MATLAB figures to represent graphs, hence the modeler can further adjust the schematic by using available functions from the toolbar at the top of the window. Thus, the Schematic Designer provides capabilities in terms of graphical model representation comparable to the simulation software available on the market. On the contrary, the schematic designer does not allow directly drawing a network on the schematic, but instead defines components and connections by using appropriate controls. Since the QSF library is composed of tens of components it is an efficient and acceptable solution.

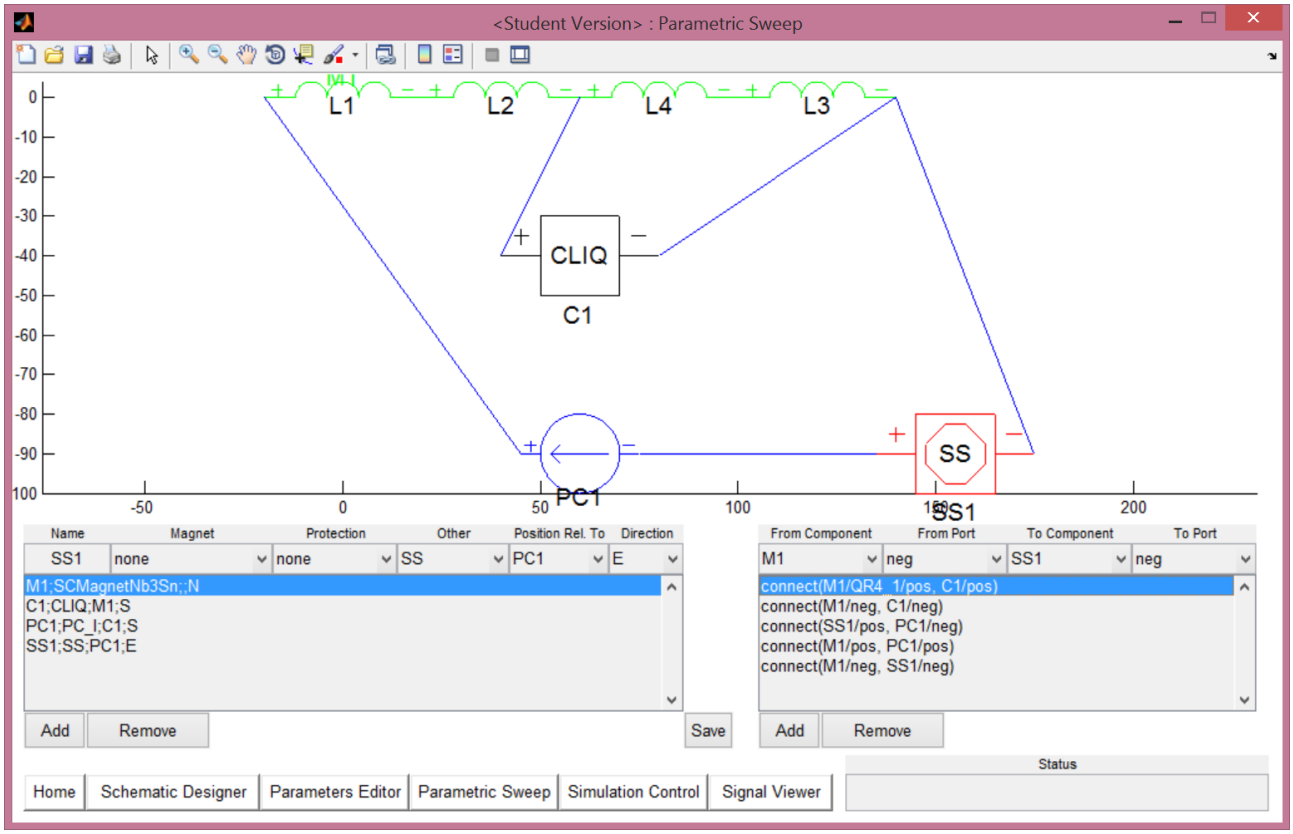


Figure 5.5. Schematic Designer tab with a stand-alone magnet protected by one CLIQ unit

The UML sequence diagrams in Figures 5.6 and 5.7 present the life-cycle of the Schematic Designer window. The components used in the model are listed the bottom left side of the GUI. New elements can be added by choosing an entry from the appropriate drop-down library (magnet, quench protection, or other components) and specifying a unique name. Note that those three component libraries are exactly the same introduced in Figure 4.2. Moreover, the controls used to select the library component are mutually exclusive, thus the modeller can selected only one of them at time. Additionally, if needed, a dialog window appears to offer more details about added component (for example the number of electrical parts in a magnet model). After clicking the *Add* button below the components list, the component is added to that list and its icon shows up on the schematic. Furthermore, the name of the new component is added to the drop-down controls *From Component*

and *To Component* controls used to add connections between components. Finally, any component added to the schematic can be removed as well by means of the *Remove* button.

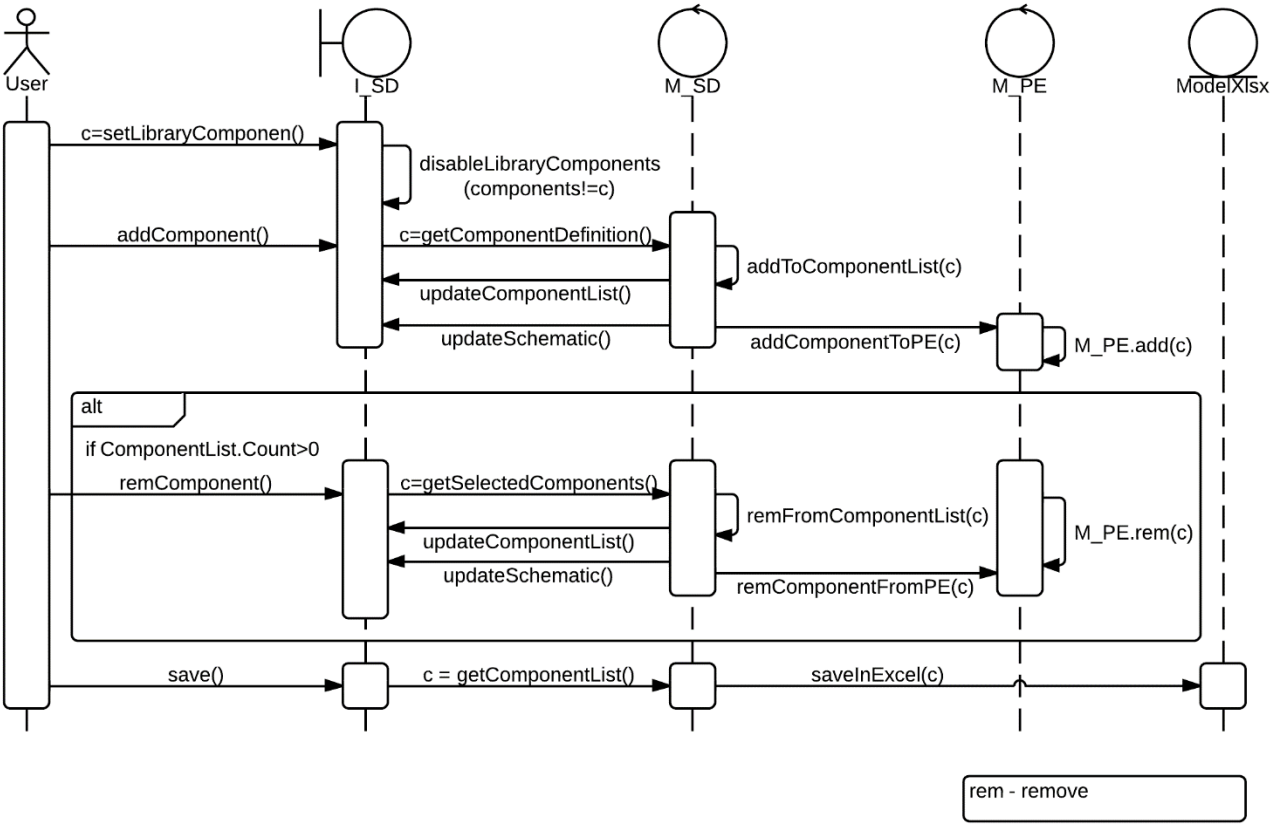


Figure 5.6. Sequence diagram of the procedure responsible for adding components to the schematic

When at least two components appear on the schematic it is possible to connect them together as shown in the sequence diagram in Figure 5.7. In fact, after selecting the *From Component* and *To Component* controls, they are updated with a list of corresponding ports. At this stage, the user's choice is constrained to components and ports already existing in the schematic. Moreover, some connections are not allowed (for instance if a connection already exists or if the modeller wants to connect together the same port of a component) and such situation results in displaying a warning message.

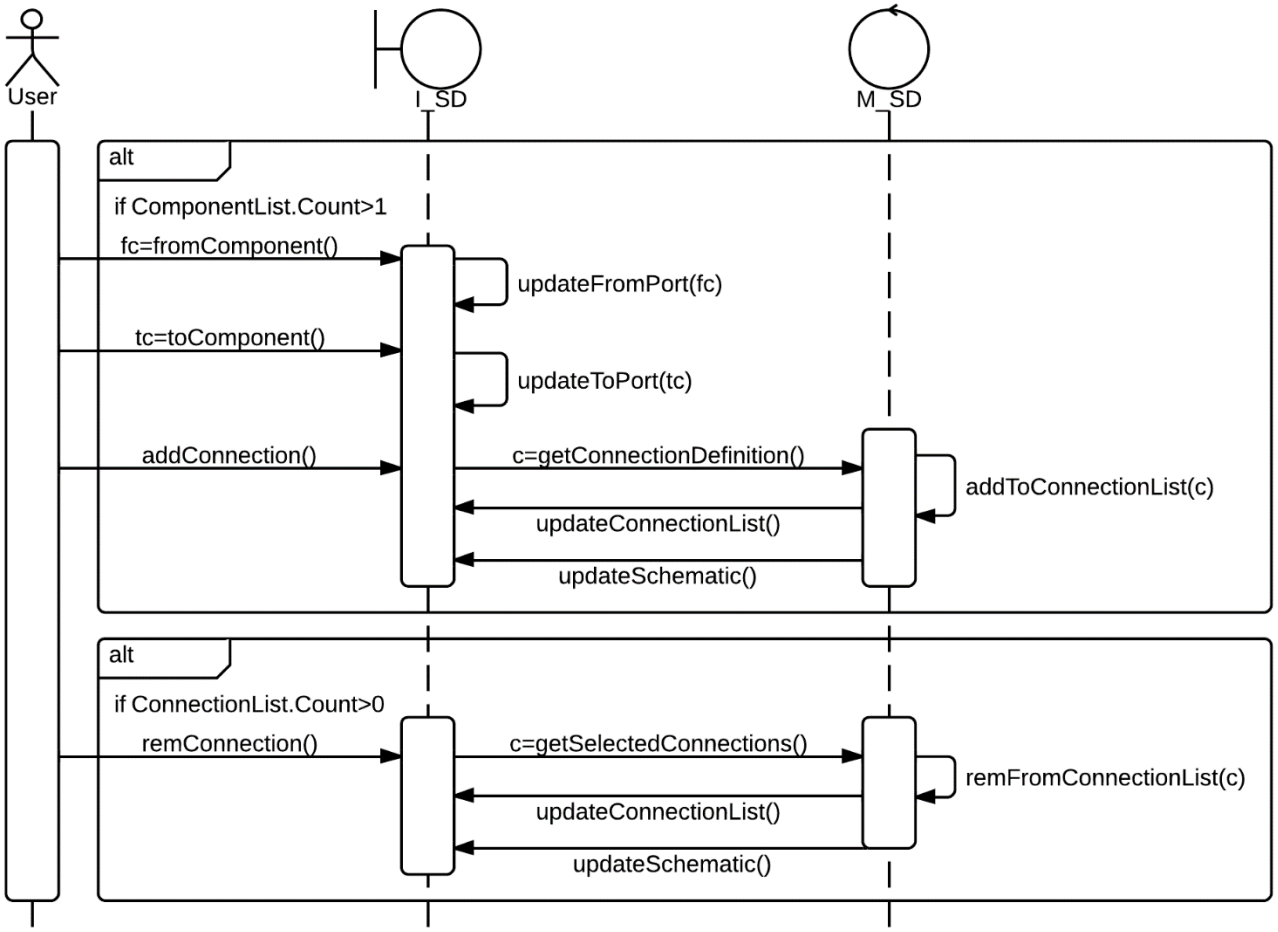


Figure 5.7. Sequence diagram of the procedure responsible for adding connections between components

5.4. Parameter Editor Tab

The next stage after creating the main electrical schematic is to define parameters of each component. All components added to the schematic, either by creating a new model or by loading configuration netlist from an existing input spreadsheet file, automatically populate the *Selected Component* drop-down list in the top of the Parameter Editor tab (see Figure 5.9).

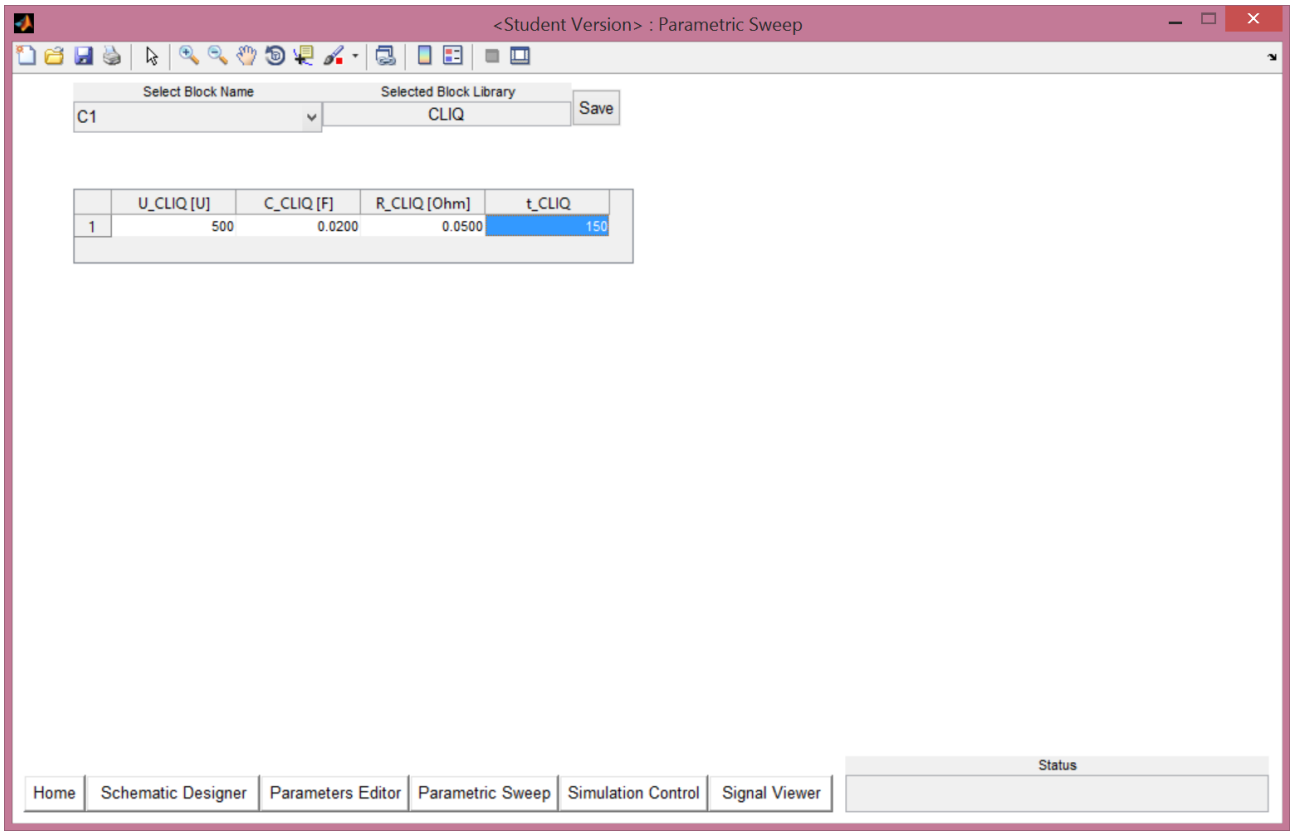


Figure 5.9. Parameter Editor tab with active CLIQ component

Figure 5.10 depicts the life-cycle of the Parameter Editor window. As already stated, the list of available components to be parameterized refers to the list of components added in the Schematic Designer tab. The appearance of the Parameter Editor tab depends on the specific component selected from the *Selected Component* list. The parameters are initialised with standard values corresponding to each library element. Additionally, the parameters can be saved as a spreadsheet file for later use in other models.

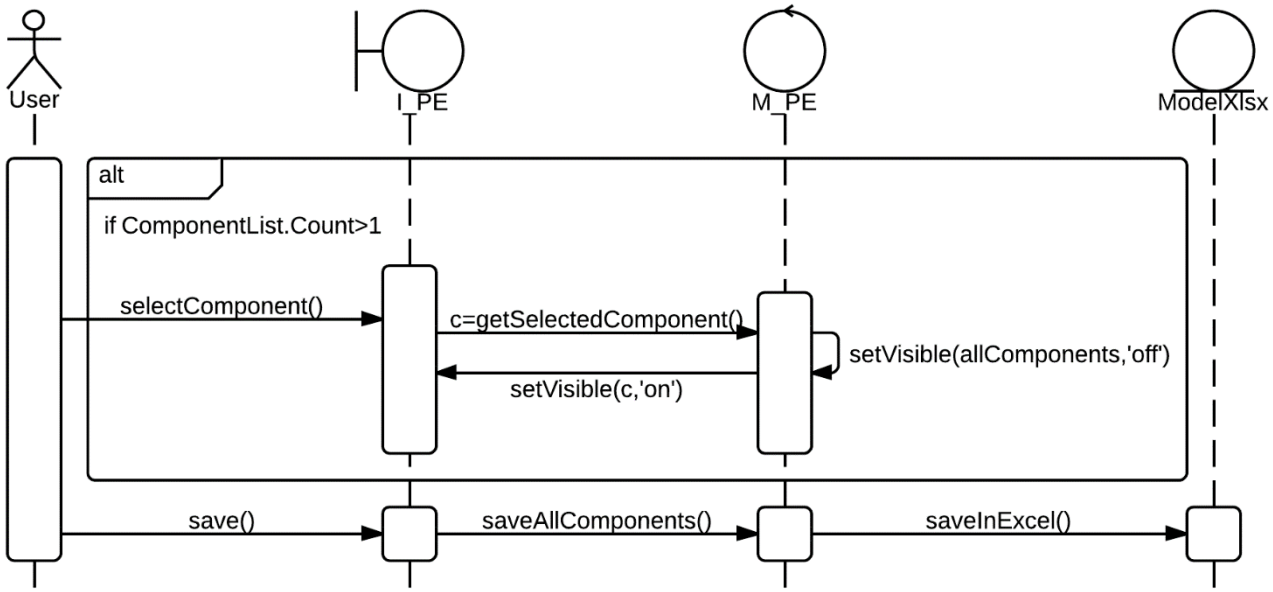


Figure 5.10. Diagram of actions performed in the Parameter Editor tab

5.5. Parametric Sweep Tab

Once the second and third tabs are filled with circuit description and corresponding component properties the model is fully defined and ready to be executed. It is worth noticing that by dividing model design procedure into two stages the user can focus firstly on the overall high-level electrical circuit definition and afterwards on the adjustment of the parameters of each component included in the model. Furthermore, at each stage there is the possibility to save temporary results. However, an optional step is to extend the list of simulations. Namely, the already created input/output files can serve as a model description pattern for the following input files with varying parameters (so called parametric sweep study). The Parametric Sweep window is divided into the following three parts:

1. *Input List* – a list on the left-hand side including all input files in the selected directory. The user can either copy selected files directly into the Parametric Sweep list or update their parameters and add to the parametric sweep list.
2. *Parameters Table* - a table in the central part of the window is dedicated to parameter definition for the parametric sweep study.
3. *Parametric Sweep List* – a list composed of the defined input files for the parametric sweep study. The list can also be loaded, edited, and saved.

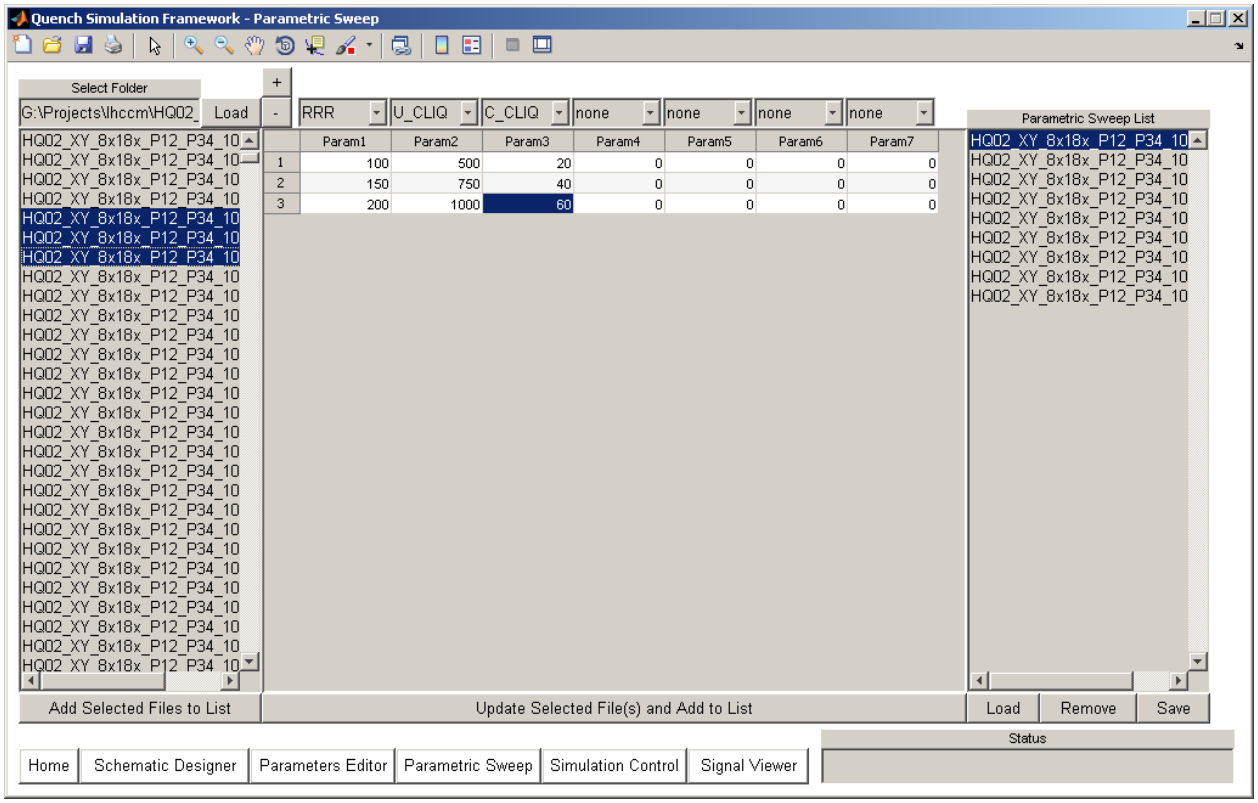


Figure 5.11. Parametric Sweep tab

The UML sequence diagram in Figure 5.12 presents the life-cycle of the Parametric Sweep window. Let *n* denote the number of selected files on the input list, *p* denote the amount of parameter

values defined by the user, and v=1, 2,…, 7 denote the number of selected parameters (the user can select up to 7 parameters from the lists above the table – see Figure 5.11). In order to enable control on the Parametric Sweep window, the number of selected parameters to be updated must be greater than or equal to 1. The first step is to select a folder with input files and load them into the Input List. Then, there are two possible operating modes. Firstly, if n>1 and the user presses the button *Add Selected Files to List*, the control class adds n files to the *Parametric Sweep List*. Secondly, if n>1 and p≥1 the user can execute the Parametric Sweep module input generator. After clicking the button *Update Selected Input File(s) and Add to List*, the framework finds the largest number out of all suffixes included in the input file names (the number increased by one becomes a reference suffix for the generated files) and creates $n \cdot p$ copies of the selected file by updating v parameters p times and saving resulting files under a new name with increasing suffix. The parametric sweep list automatically updates once the process is completed.

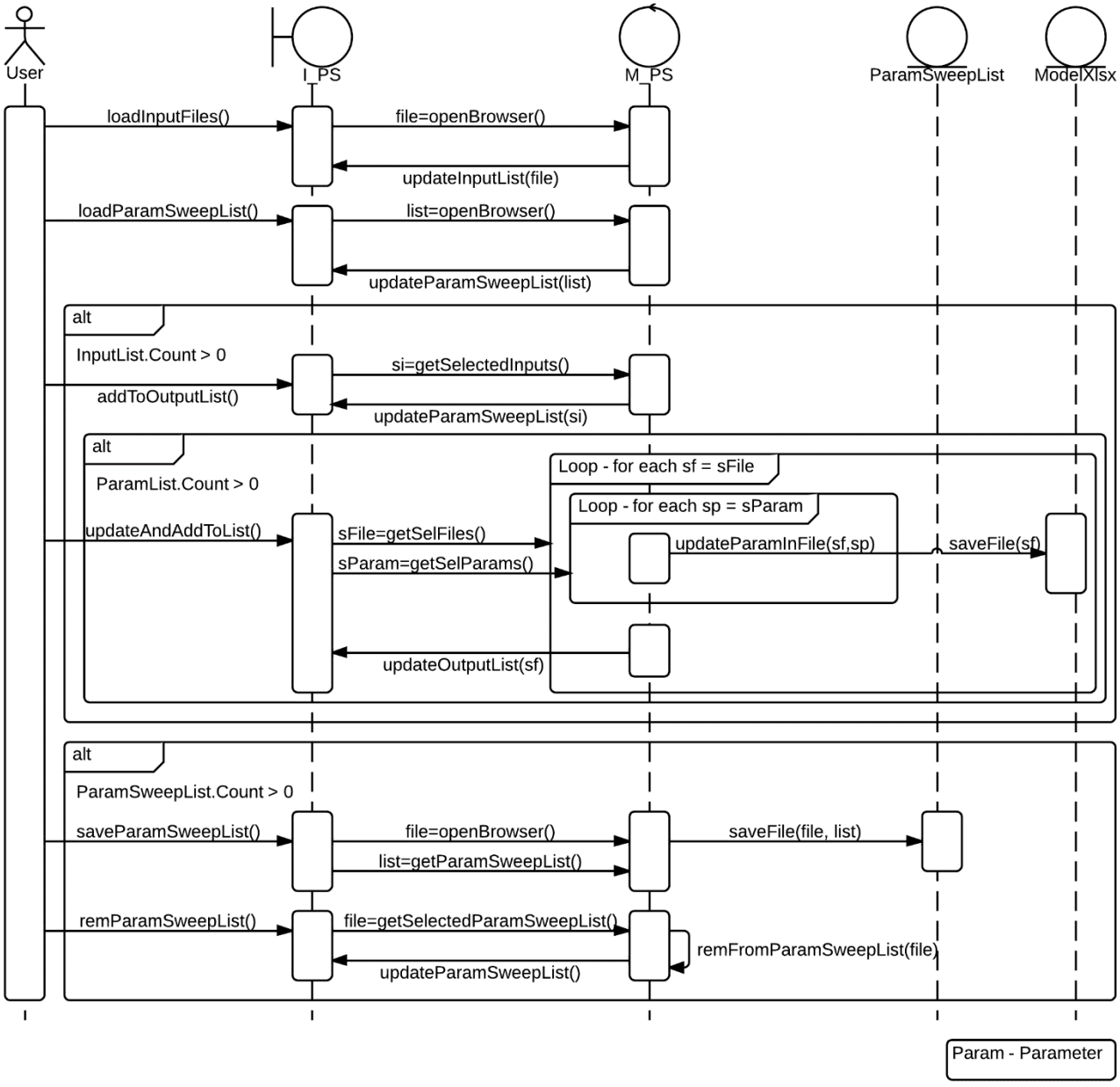


Figure 5.12. Sequence diagram depicting the work-flow in the Parametric Sweep tab

5.6. Simulation Control Tab

The Simulation Control module is directly linked to the Simulation Control tab of the GUI (see Figure 5.13). The window is organized as follows: the top part of the window is occupied with controls used to obtain information about absolute paths to directories with input and output files; and the bottom part contains list of available slave sessions of the QSF on both local and remote machines.

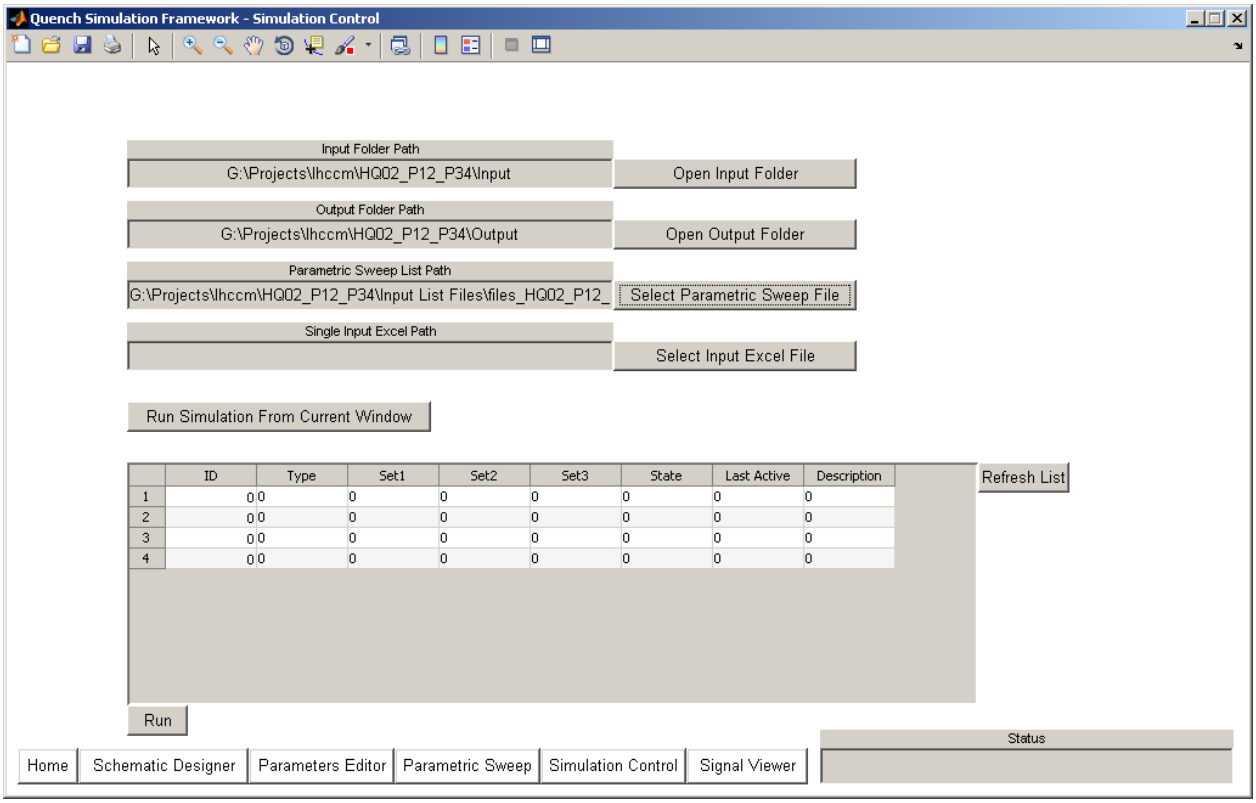


Figure 5.13. Simulation Control tab

The life-cycle of the Simulation Control tab is schematized in Figure 5.14. Firstly, the SCM requires providing information about path of input folder, output folder, and network folder. Secondly, the user must select either a path to a single spreadsheet file or a list file with parametric sweep files. Depending on this choice, the SCM will operate in either single simulation mode or multiple simulation mode, respectively. Moreover, these controls are mutually exclusive, hence the user can select only one option. Finally, the user can execute simulations in one of two possible modes:

1. Execute simulations from the current MATLAB session by clicking *Run Simulation From Current Window*. However, this will freeze the current MATLAB session as long as the single simulation or set of simulations is being executed.
2. Select the slave MATLAB session from the list of available slave sessions including both local and remote machines and execute the Parallel Computing Module. After selecting a desired slave and clicking on the *Run* button the master session (GUI) saves appropriate files in order to communicate with the slave sessions through the network folder.

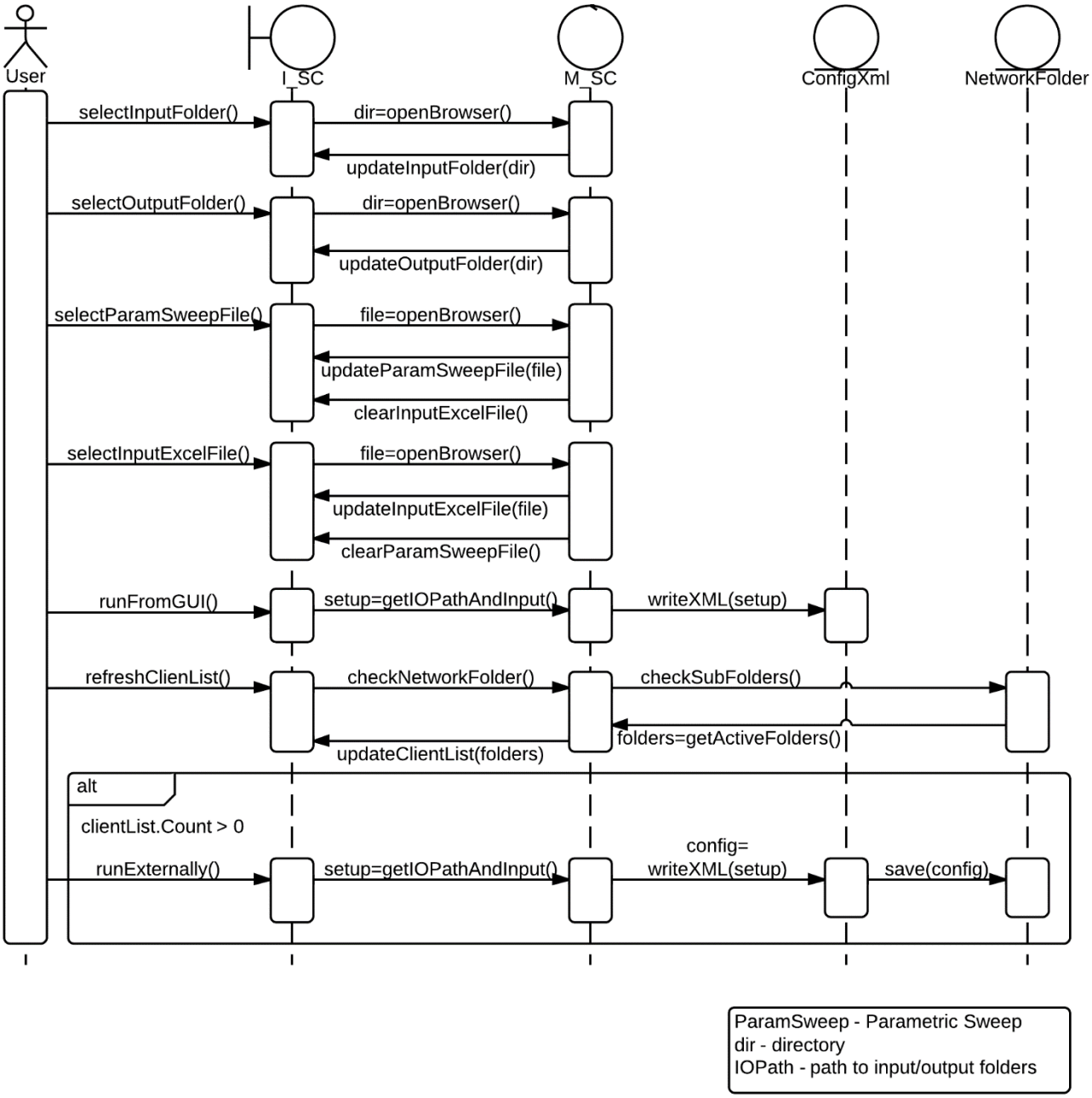


Figure 5.14. Life-cycle of the Simulation Control tab

5.7. Signal Viewer Tab

Finally, the last tab of the GUI is devoted to the analysis of the simulation results by calling routines included in the Generate Report Module. The majority of the window is covered by a plot displaying the results obtained from the simulation. The tree on the left-hand side is dynamically populated depending on the elements in the folder with simulation results. In other words, for models composed of different number of blocks and testing different protection scenarios, the tree will have a different structure. Information on how to read data from *.mat* files is stored in an XML file. Due to the lack of a graphical object to display the data as a tree in MATLAB, a *JTree* object from the Java Swing library has been employed. Furthermore, the tree structure created this way enables selecting and simultaneously plotting many signals on the same plot.

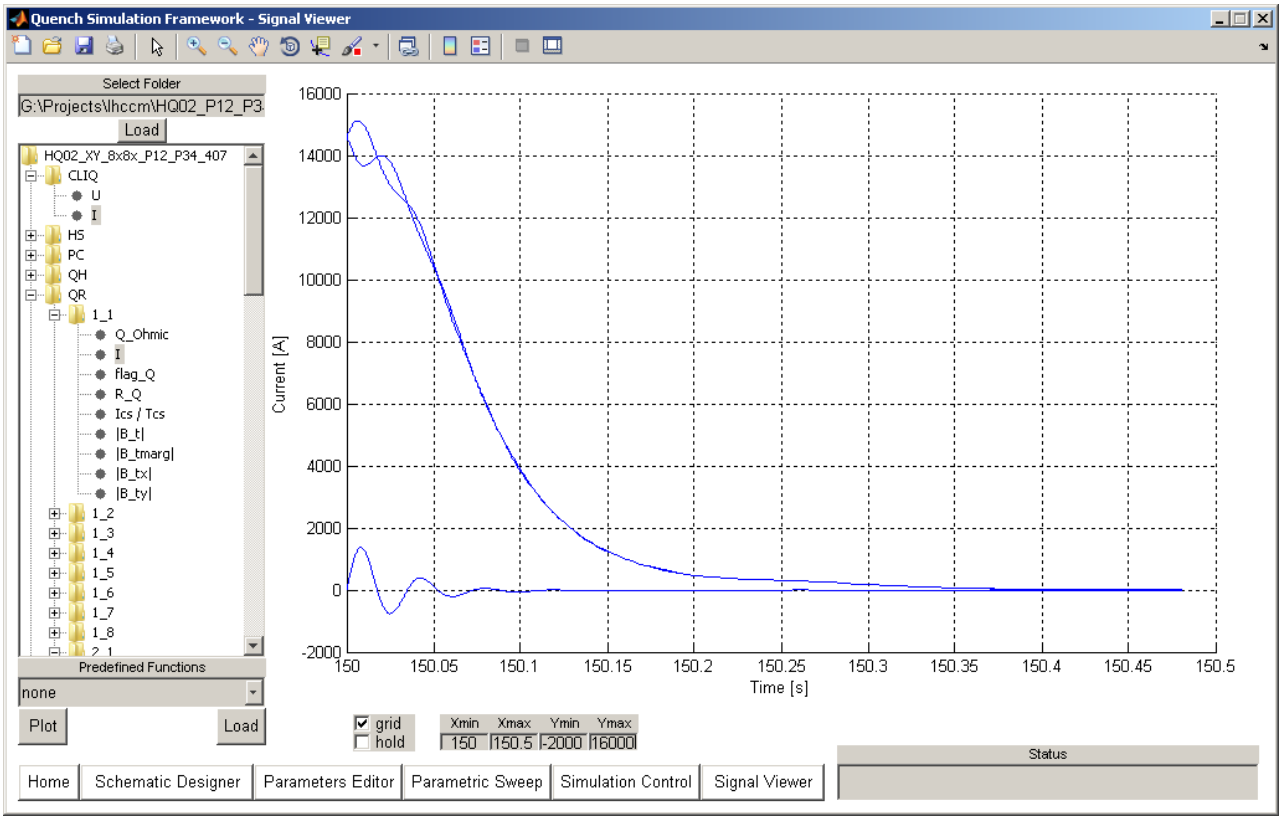


Figure 5.15. Signal Viewer tab with a plot of the simulated currents in various components of the system

The key actions performed by the user are depicted as an UML sequence diagram in Figure 5.16. Firstly, in order to avoid mismatches between the folder structure defined by a model and its actual *.mat* files (in the case some output files are removed, renamed, etc.), the Signal Viewer module reads the saved files from the loaded output directory and populates the tree structure. Secondly, the user can plot the signals selected in the tree as well as load previously saved plots with *.fig* extension. Thirdly, in order to better examine the influence of one component on the another the framework also support user-defined functions that better perform some additional post-processing tasks, such as calculation of quench resistance of an entire magnet and visualization of 2-D temperature distribution. Finally, the properties of the visualized plot, such as limits of the axes or presence of the grid, can be adjusted as standard MATLAB plots. Moreover, the figure can be further edited as the Signal Viewer tab employs built-in functions to zoom/in out, move, rotate, and save a plot.

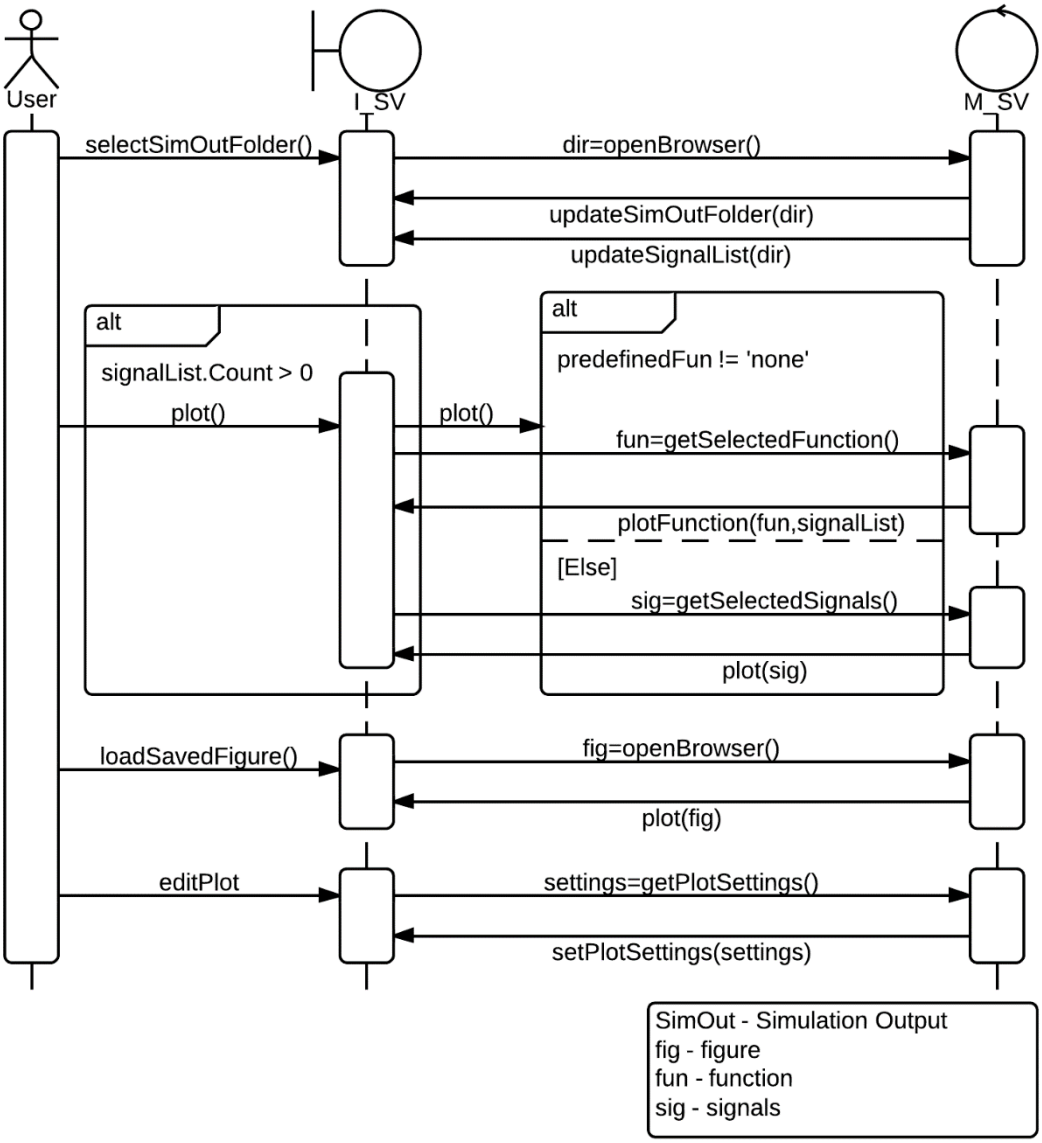


Figure 5.16. Sequence Diagram of the Signal Viewer tab

6 First Experience Using the Quench Simulation Framework

The most important aspect of the simulation framework development is validation of an adopted model against the measured data from experiments. It allows defining the conditions under which the model accurately reproduces complex transients occurring in the magnet. The simulations must provide an acceptable agreement for wide range of operating parameters. As a consequence of successful validation, the model can provide insights to real magnet performance at intermediate current levels as well as to predict what can happen at higher/lower current levels not covered during test campaigns. Additionally, as presented in Chapter 3, the model includes non-measurable signals such as temperature or heat flow that can be displayed in a convenient way and studied in order to better understand the magnet behaviour. However, it requires performing various experiments with different magnets. Finally, simulations provide useful information about the key parameters of the model and its spatial resolution.

Already during its development phase, the QSF was used to simulate the behaviour of various types of superconducting magnet circuits. The following sections present a selection of the obtained results in order to better illustrate the features of the framework and validate it against measurement results:

1. MQXC2 – Stand-alone Nb-Ti quadrupole magnet protected by the quench heaters and energy extraction systems. Simulation and comparison with measurements are provided.
2. HQ02 – Stand-alone Nb_3Sn quadrupole magnet protected by CLIQ and energy extraction system. Validation against the experimental data and detailed explanation of the electro-thermal transients simulated by the model.
3. MQXF – Full-scale version of the HQ02 magnet, to be manufactured in the following years. Parametric sweep study aimed at finding optimum configuration of the CLIQ system as well as magnet parameters.
4. MB – Full-scale LHC main dipole magnet. Analysis of a possible quench protection backup solution based on CLIQ.
5. RB – Circuit composed of 154 MB magnets. Presents model capabilities in terms of simulating long chains of magnets including simplified dynamic effects.

In order to perform simulations composed of full electro-thermal model of superconducting magnets (sections 6.1-6.4) the magnetic field transfer function must be provided. For that purpose the ROXIE has been employed. The remaining parameters come from geometrical dimensions of the magnet and properties of materials used. The simulations were executed using a machine with Intel Core i7 vPro 3.2 GHz processor and 32 GB of RAM memory. For instance, the average computation time for a model of MQXF magnet composed of 200 blocks comprising over 8000 fundamental

Simulink and Simscape components with over 20000 parameters was equal to 55 minutes. The contribution of model design procedures to the overall single simulation time is presented in Figure 6.1. Simulations executed in the multiple simulation mode feature time savings of about 20 % as there is no need to build a model again.

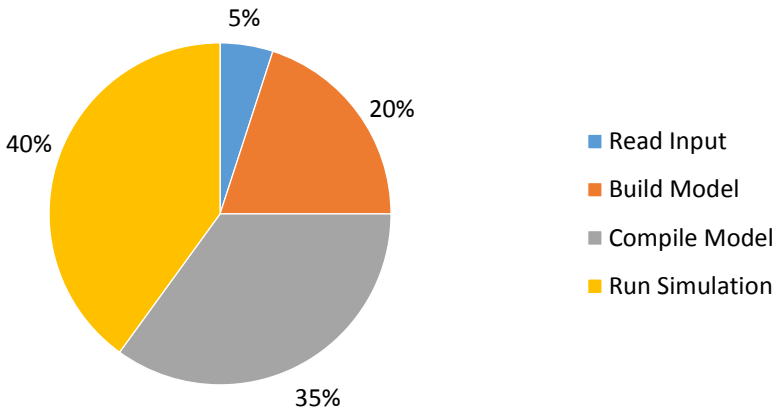


Figure 6.1. Simulation execution time.

6.1. MQXC2

The MQXC2 magnet is a Nb-Ti, 1.6 meter quadrupole model magnet for the LHC high luminosity upgrade [46]. The parameters of the magnet are summarized in Table 6.1 and the circuit schematic including the stand-alone magnet together with power converter, Quench Heaters, and energy-extraction system is depicted in Figure 6.1. The magnetic field distribution in half a pole of the tested magnet used to calculate quench margins, magneto-resistivity, and dynamic effects in the model is presented in Figure 6.3. The magnet is equipped with 8 quench heater stainless steel strips, each protecting half a pole and attached between the outer and inner layers of the magnet (see Figures 6.2 and 6.3a). Each quench heater circuit has a capacitor bank of 12 mF charged to 900 V that after triggering provides a peak current of 62 A flowing through the strips. The capacitor bank is discharged with a time constant equal to 90 ms. The original design foresaw four separate QH circuits, each composed of two strips connected in series. However, one strip attached to pole 4 suffered electrical break-down so only one strip was connected to the fourth QH circuit. As a result, the peak current in the fourth circuit is equal to 90 A. The quench heater system has been tested in order to assess its performance and provide useful information for the model validation (mainly thermal sub-network) [13, 46]. Several tests were repeated under different operating conditions. During each test, the magnet voltage and current were measured directly, and the effective coil resistance was obtained by subtracting from the measured voltage its inductive component.

Parameter	Value	Unit
Nominal current, I_{nom}	12800	kA
Operating temperature	1.9	K
Self-inductance at I_{nom}	8.4	mH/m
Magnetic length	1.655	m
Number of turns per pole	15(in)/25(out)	
Number of strands	28(in)/36(out)	
Strand diameter	1.065/0.825	mm
Bare cable width	15.10	mm
Bare cable thickness	1.9	mm
Insulation thickness	0.15	mm
Copper/NbTi ratio	1.65(in)/1.95(out) ±0.05	
Filament twist pitch	15	mm
RRR of the copper matrix	228/209	

Table 6.1. Table of the MQXC2 magnet parameters

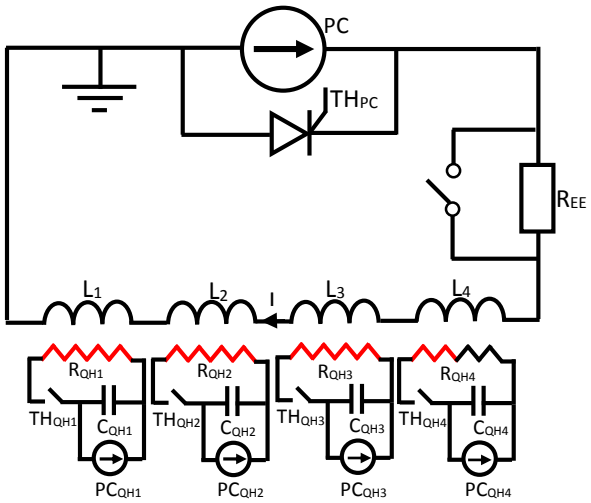


Figure 6.2. Schematic of 4 quench heater units in the MQXC2 magnet test.

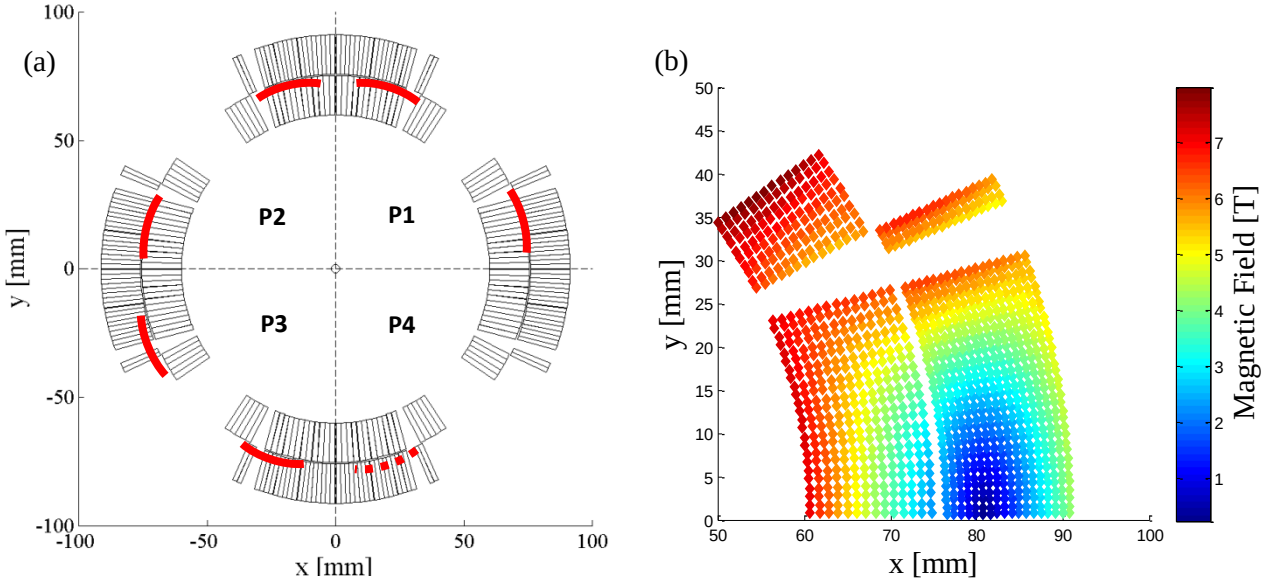


Figure 6.3. (a) Cross-section of the MQXC2 magnet showing attached quench heaters. Note that the real thickness of the quench heater strips is about hundred times thinner than in thee figure. (b) Magnetic field distribution in half a pole of the MQXC2 magnet, simulated using ROXIE.

A key step in the model development is to select a reasonable spatial resolution as well as determine the parameters that have the largest influence on the model performance. The developed QSF is scalable in a sense that the modeler can run models with variable resolution, i.e. with blocks as small as the volume of a single magnet turn or as big as the volume of an entire pole. Hence, it was possible to efficiently analyse how the spatial resolution affects the obtained results and simulation execution time.

The performed experiments aimed at analysis of the Quench Heaters performance under different operation conditions. Each quench heater circuit is triggered manually at t=0 s. After triggering, a current is discharged through Quench Heater strips and generates Ohmic heat in so called heat stations distributed along the magnet. Resulting increase of the temperature at those points forces

the surrounding magnet windings to quench. In turn, the normal zone propagates longitudinally and builds up a quench resistance that discharges the magnet transport current. Figure 6.4 presents a comparison between experimental quench resistance and quench resistance simulated with two MQXC2 models composed of 8 electrical parts equally divided into 9 (low-resolution case) or 18 (high-resolution case) blocks. The resistances are obtained with a Quench Heater discharge at a nominal current of 12.8 kA. At this stage, both models feature a set of nominal parameters of the MQXC2 magnet.

As one can notice from Figure 6.3b the magnetic field can vary significantly within one block (a turn or multiple turns). For this reason, such blocks should be represented in the model with the highest possible accuracy (see windings in the inner layer in the mid-plane in Figure 6.3b). Otherwise, if the blocks are composed of turns with high magnetic field gradient and turns with homogenous magnetic field, the quench initiation is less accurate. The reason for this result is that blocks with high magnetic field has lower quench margin, and are easier to quench. Hence, averaging process can lead to inaccuracies in estimation of quench initiation. On top of that after the quench of a block the Ohmic loss calculated by the model provides additional heat and increases temperature faster. This result in a certain heat to diffuse to adjacent coil windings, a process also called turn to turn propagation. As a result, after one model block quenches, its neighbours are transferred to the normal state due to the superposition of Ohmic loss produced in that block and in the heat station. On the contrary, in the case of the blocks where magnetic field is averaged over a larger volume (low-resolution case) the resulting quench margin is larger; thus it takes more time for such blocks to cross the critical surface. Moreover, the low-resolution model has less thermal connections between the blocks. Hence, the higher the spatial resolution of the model, the more accurately turn to turn propagation is reproduced.

The simulated 2-D temperature profile in the magnet cross-section reported in Figure 6.5 shows the difference between low- and high-resolution models at t=20, 50, 200 ms (depicted by red dashed lines in Figure 6.4). In fact, in the former case (Figure 6.5a, b, c), the calculated peak magnetic field is averaged over a large number of coil windings and the simulated quench initiation is less accurate. In the latter case (Figure 6.5 d, e, f) the magnetic field in the blocks is calculated with higher accuracy and better reproduces the transition to the normal zone. More importantly, the incorrect reproduction of the quench initiation of the low-resolution model causes a difference of more than 100 K in the calculated hot-spot temperature; in fact, the current discharge is significantly lower due to the delayed development of the quench resistance.

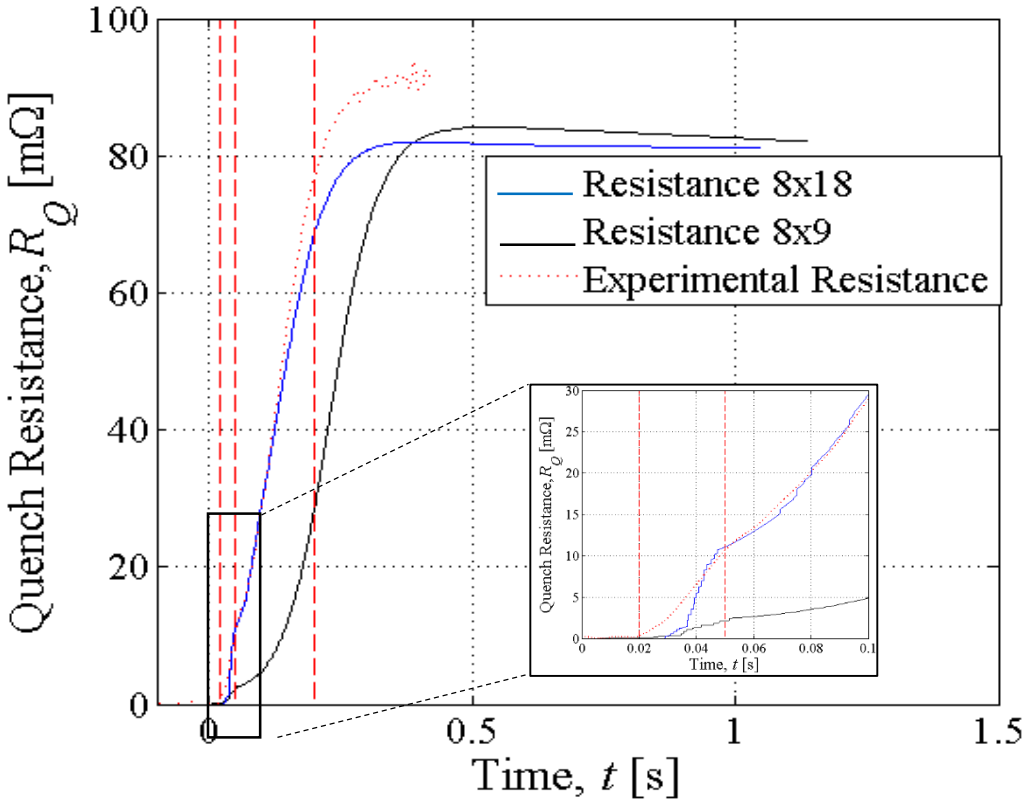


Figure 6.4. Quench resistance after quench heaters triggering at I_0=12.8 kA for 8x9 and 8x18 models compared to experimental resistance

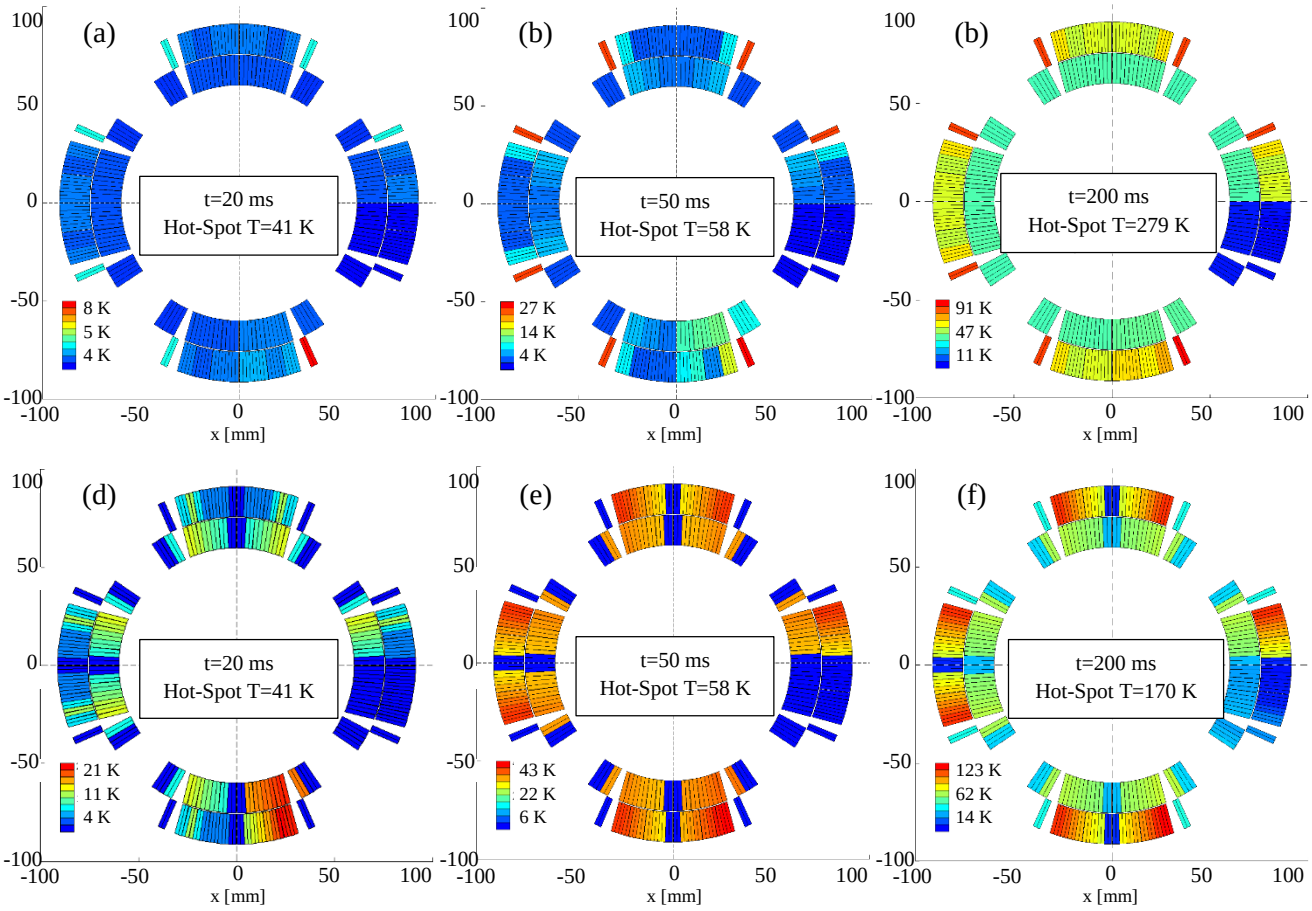


Figure 6.5. Simulated magnet temperature distributions in the quadrupole coil windings cross section featuring 4 blocks of windings in two coil layers, at 20, 50, and 200 ms after triggering Quench Heater for 8x9 (*a*, *b*, *c*) and 8x18 (*d*, *e*, *f*) models.

Thus, the high-resolution magnet model was adopted composed of 8x18=144 blocks. The simulation results are compared with measurements performed during tests. As one can notice from Figure 6.4 there is about 10 ms difference between the measured and simulated quench initiation for high-resolution model. For that purpose, in order to obtain a better agreement, the conductive heat transfer coefficient of the kapton insulation layer of the quench heaters has been multiplied by a factor 2 in the model. This modification enhances the thermal conductivity; hence, the model predicts a faster quench in the blocks adjacent to the quench heaters and reproduces more accurately the moment of the quench initiation in the coil (see Figures 6.6 and 6.7).

The model has been validated for nominal current (12.8 kA) and intermediate current (6 kA). For both current levels the quench heaters were able to protect the magnet (see Figures 6.6 and 6.7). Furthermore, the model predicts correctly the time instant when the quench starts in the magnet which is a key parameter from the quench heater design point of view.

Additionally, during the 6 kA test an energy extraction system was triggered at t=950 ms, hence introducing additional 40 mΩ resistance in series to the magnet. The inhomogeneous energy deposition resulting from the presence of a damaged quench heater strip (see Figure 6.2a) is included in the model to reduce the sources of inaccuracies between measurements and simulations. The magnet at nominal current is transferred to the normal state faster as compared to the 6 kA case. The reason for this is that at higher current level the magnet requires less energy to initiate and propagate a quench as its current density and magnetic field are higher. Moreover, the higher the current the higher the Ohmic loss generated in the coil winding pack.

However, the simulated resistance after the current discharge in the magnet at both current levels is lower than measured one. This discrepancy is due to the longitudinal heat propagation between the two halves of pole 4 which is not included in the model. In the real case, this heat transfer causes a propagation of the quench also in the half pole which has no quench heater strip, thus resulting in a larger quench resistance.

Also, the agreement between simulated and experimental resistance at 6 kA is less good for t<200 ms. The reason for this result is the slower longitudinal quench propagation velocity at 6 kA. The applied modelling approach in the QSF assumes that the block quenches its entire volume, and does not account for the longitudinal quench propagation.

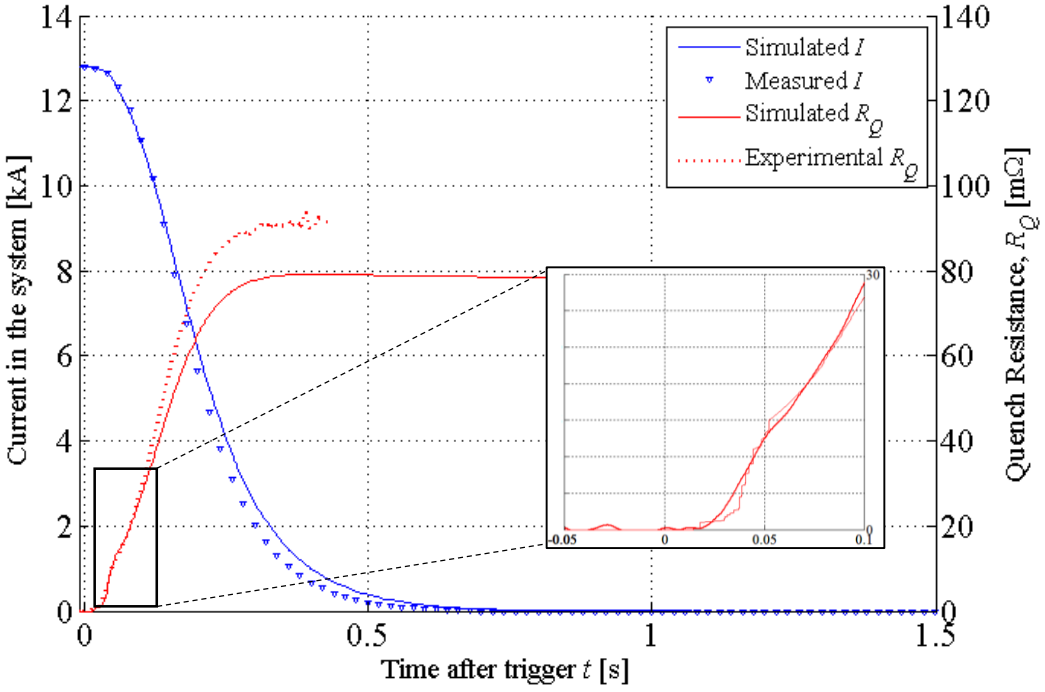


Figure 6.6. Comparison between measured and simulated magnet current in and resistance versus time, for the magnet discharged by the Quench Heater system at I_0=12.8 kA.

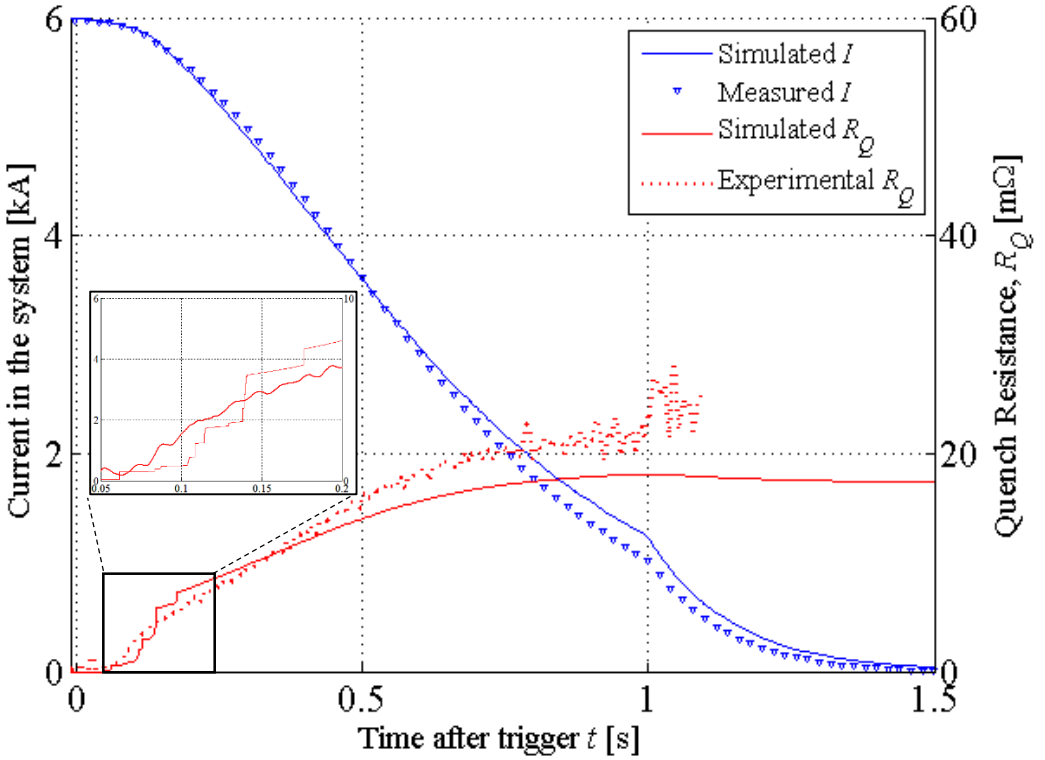


Figure 6.7. Comparison between currents and resistances versus time for the magnet discharged by the Quench Heater system at I_0=6 kA.

It is worth noticing that, since the model is capable of simulating the thermal behaviour of the different portions of the magnet, the analysis of the obtained temperature profile is of great help in understanding the source of inaccuracies (Figure 6.8).

As stated before, the higher current level results in a faster quench of the magnet induced by the heat flowing from the quench heater strips through insulation layers to the coil winding pack. At

lower current levels, the magnet starts to quench about 50 ms after the quench heater discharge (Figure 6.8) in the half of pole 4 where higher quench heater power is discharged. Apart from pole 4, the temperature of the magnet is distributed evenly. However, the broken quench heater may result in performance differences between poles of the magnet in time as it is subject to a different thermal impact.

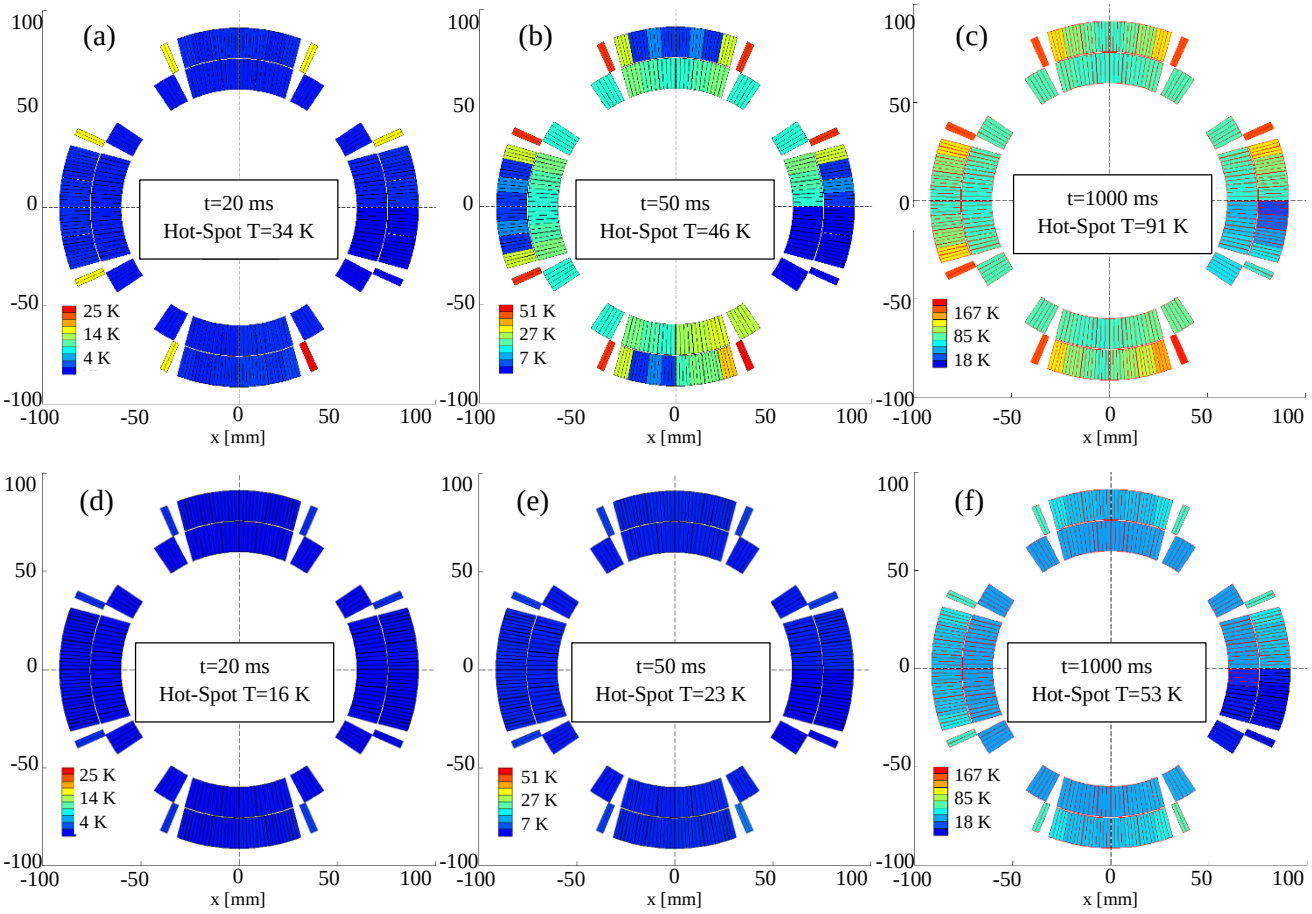


Figure 6.8. Simulated magnet temperature distribution in the MQXC2 cross section at 20, 50, and 1000 ms after triggering Quench Heaters at I_0=12.8 kA (*a*, *b*, *c*) and I_0=6 kA (*d*, *e*, *f*).

6.2. HQ02b

The 0.8 meter long, Nb_3Sn quadrupole model magnet for the LHC high luminosity upgrade (called HQ02b) has been tested in the CERN magnet test facility. A CLIQ unit (28.2 mF, 500V) was connected between the middle of the magnet and the negative side of the magnet (see Figure 6.9). The CLIQ is triggered at t=0 s by manually generating a signal from the quench detection system. Additionally, the magnet is protected by a 10 mΩ energy extraction fired at t=3 ms. The main HQ02 magnet and cable parameters are presented in Table 6.2 [18]. The magnet cross-section and magnetic field distribution are illustrated in Figure 6.10.

Parameter	Value	Unit
Nominal current, I_{nom}	14600	kA
Operating temperature	1.9	K
Self-inductance at I_{nom}	7.0	mH/m
Magnetic length	0.8	m
Number of turns per pole	46	
Number of strands	35	
Strand diameter	0.778	mm
Bare cable width	15.15	mm
Bare cable thickness	1.437	mm
Insulation thickness	0.1	mm
Copper/Nb3Sn ratio	1.23	
Filament twist pitch	13.55	mm
RRR of the copper matrix	75-140	

Table 6.2. Magnet Parameters

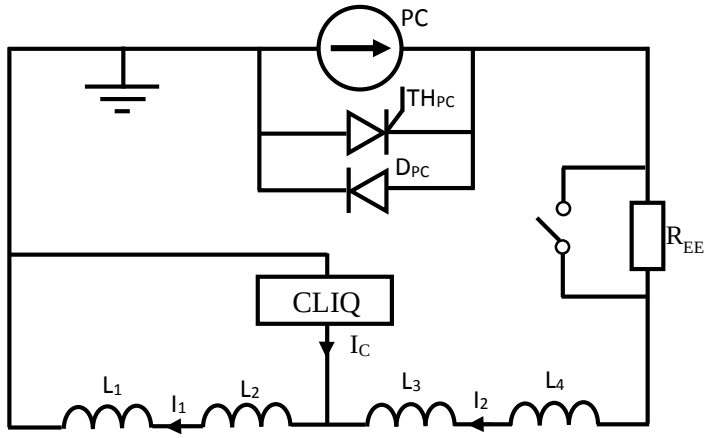


Figure 6.9. Schematic of the tested HQ02b magnet with single CLIQ unit

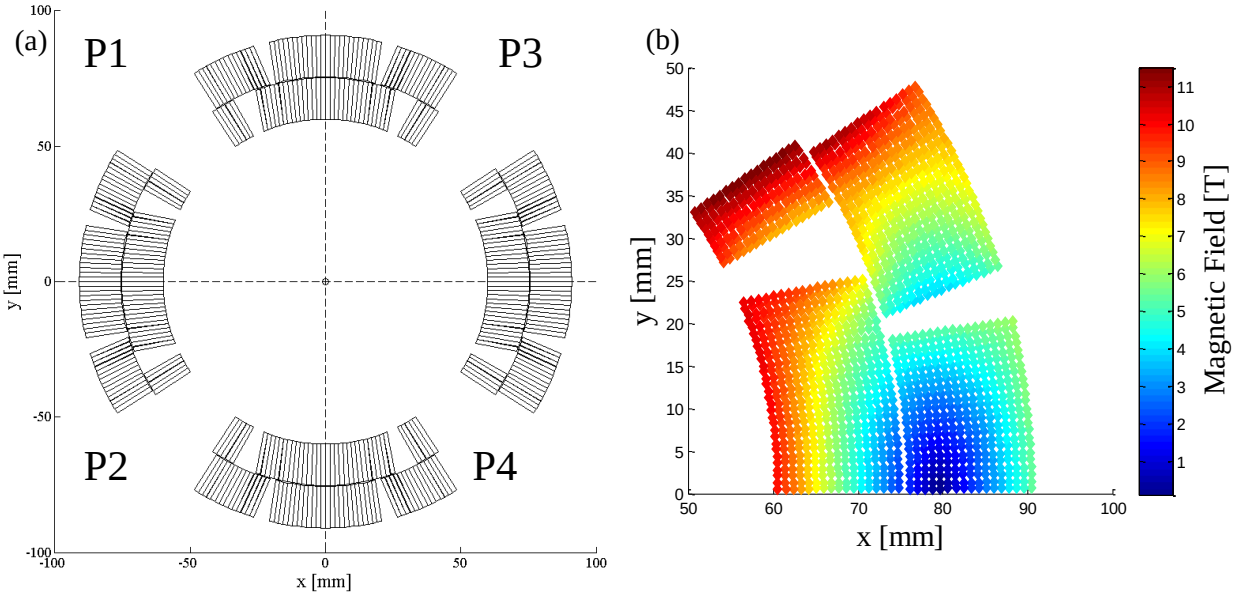


Figure 6.10. (a) Conductor distribution in the HQ02b magnet. (b) Magnetic field distribution in half a pole of the HQ02b magnet, simulated using ROXIE.

Figure 6.11 shows the current discharge after triggering CLIQ at a nominal current of 14.6 kA, together with the quench resistance developed during the transient. The model reproduced the complex electro-thermal transients with very good accuracy. Additionally, the model predicts well the frequency of oscillations along with the peak current. Experimental quench resistance is not measured directly, but calculated by subtracting the inductive component from the voltage across the sides of the magnet with different transport current. Hence, the obtained value is subject to measurement inaccuracies of current and voltage, estimation of dynamic effects in the magnet and signal filtering.

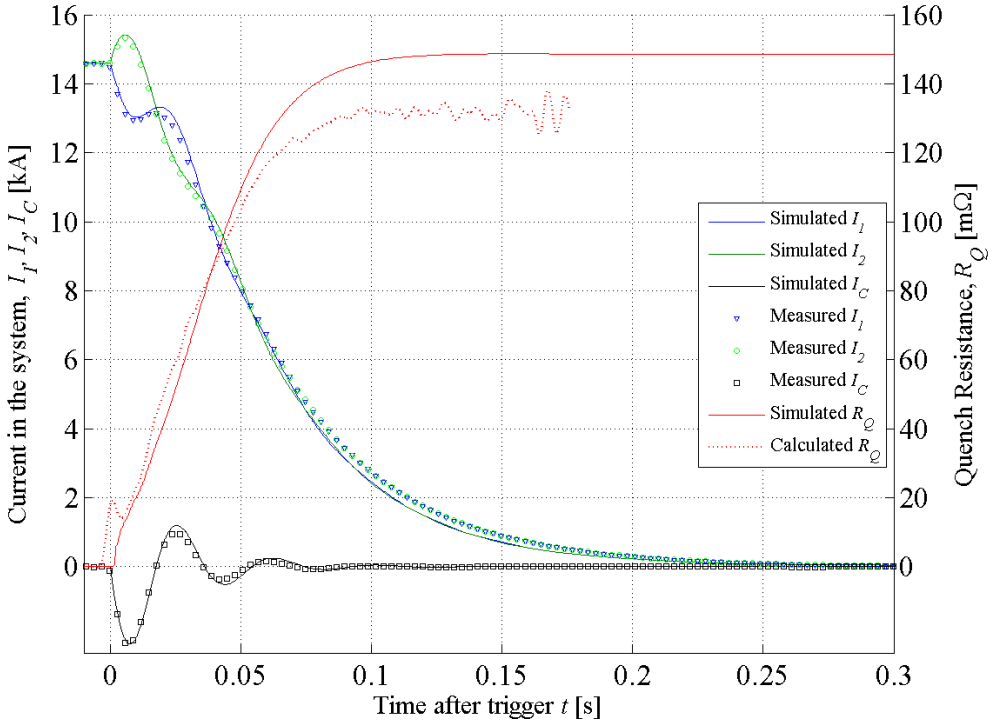


Figure 6.11. Magnet discharged by CLIQ (U_0=500V, C=28.2 mF). Measured currents I_C and I_2, calculated current $I_1 = I_2$-I_C, versus time after the triggering CLIQ at 14.6 kA. Effective coil resistance R_Q, versus time. Simulated I_1, I_2, I_C, and R_Q.

In order to better understand the transients occurring in the magnet and the initiation of a quench, the modelled current flowing in a selected block $I_{1\text{-}11}$ (thermal block 11 in electrical part 1) is shown in Figure 6.12a, together with its corresponding critical current $I_{cs,1\text{-}11}$; furthermore, the simulated quench resistance, developing at the instant when $I_{1\text{-}11} > I_{cs,1\text{-}11}$, is plotted. The critical current decreases quickly after the CLIQ discharge by effect of the increase in the local temperature.

In fact, the oscillations provoked by the CLIQ discharge introduce a fast change of the local magnetic field change, which in turn introduces inter-strand and inter-filament coupling losses. The produced losses increase the temperature in the cable until its characteristic point crosses the critical surface. From Figure 6.12b one can observe that triggering CLIQ produces enough losses to quickly decrease critical current; only a few milliseconds after the CLIQ triggering a quench is initiated. As a consequence, the block starts increasing its resistance and Ohmic losses are generated in the block. Figure 6.12b also shows that for this particular magnet CLIQ mostly relies on inter-filament coupling loss rather than inter-strand coupling loss to deposit heat in the cable. In fact, the time constant of the inter-strand coupling loss (green) is larger than the time constant of the inter-filament coupling loss (blue), hence their contribution is limited.

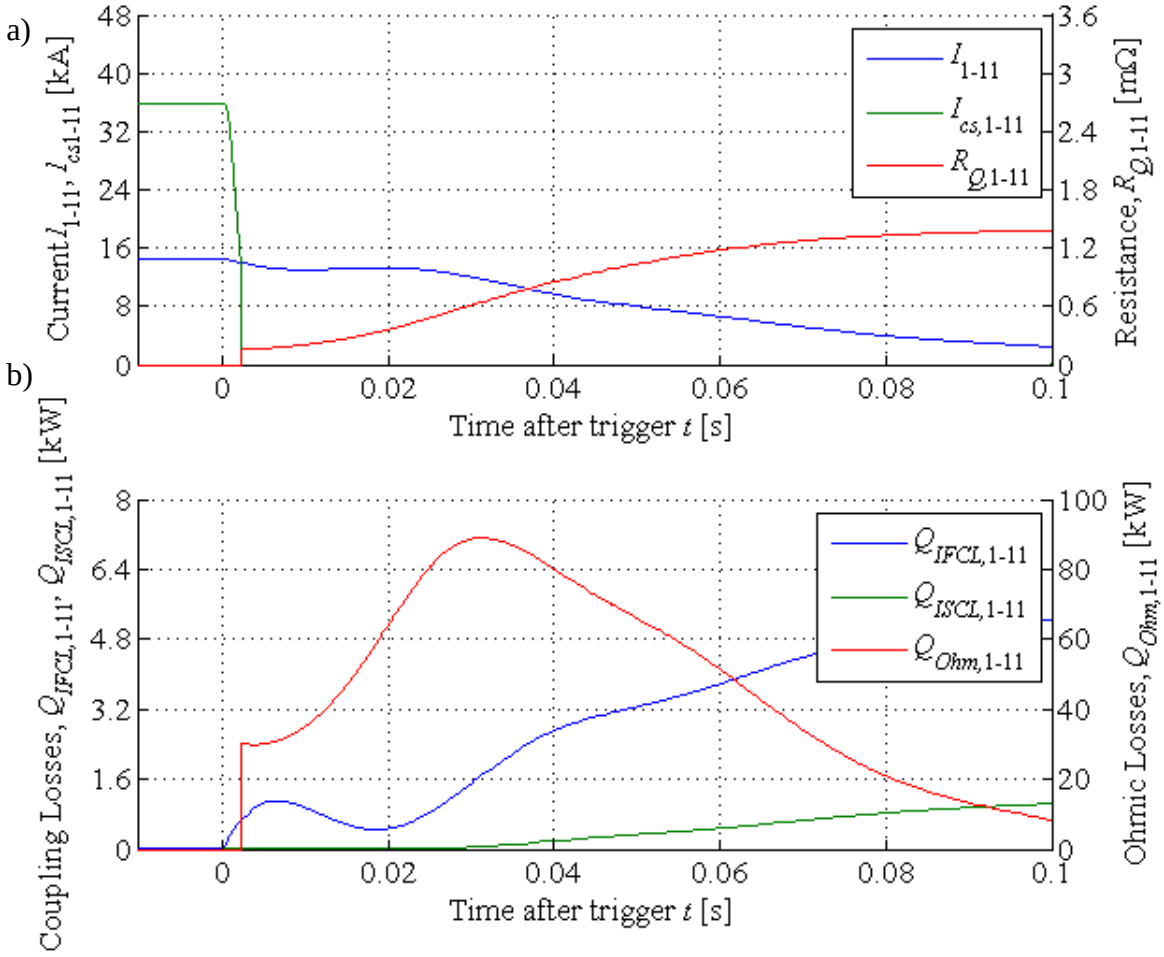


Figure 6.12. (a) Quench mechanism in a selected block of the modelled magnet (thermal block 11 in electrical part 1). (b) Inter-filament and inter-strand coupling loss introduced by CLIQ and Ohmic loss.

Figure 6.13 and 6.14 present the simulated distribution of the inter-filament and inter-strand coupling loss per unit volume deposited in the coil winding pack, respectively. IFCL losses are higher in the inner layer of the magnet where the introduced magnetic-field change is higher. Another important aspect is difference in RRR between the poles – poles 1 and 3 have higher RRR than poles 2 and 4 (see Figure 6.10a). In turn, higher RRR results in smaller copper resistivity and higher IFCL amplitude (see Figure 6.13). After triggering the CLIQ unit, the current oscillations are too fast for the ISCL to significantly develop. Hence, they have a much lesser effect on the quench initiation than the IFCL. However, in other transients, such as slow current ramps, ISCL may be the dominant contribution to AC losses, and it is therefore important to include them in the model.

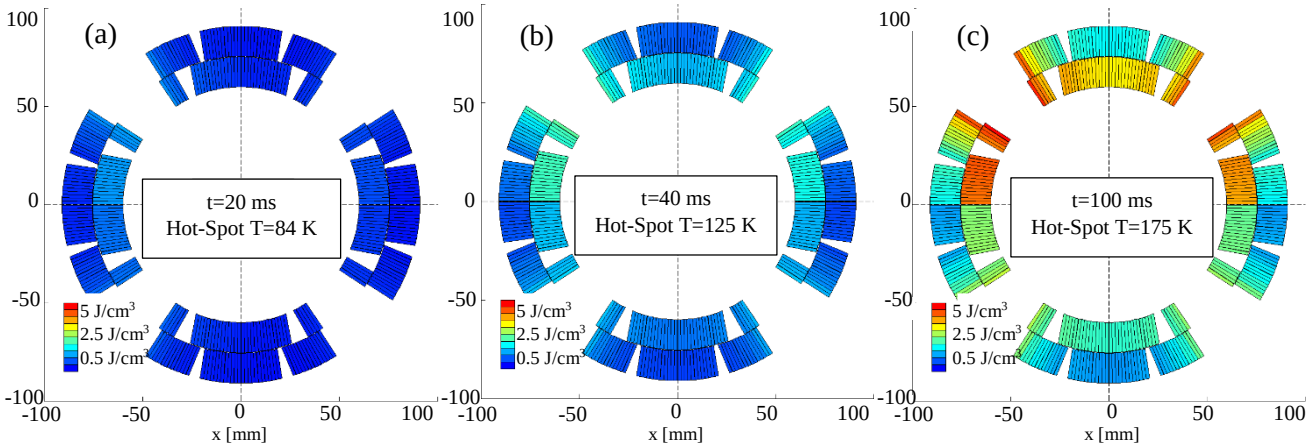


Figure 6.13. Simulated distribution of IFCL per unit volume generated in the quadrupole coil windings cross section at (a) 10, (b) 40, and (c) 100 ms after triggering CLIQ.

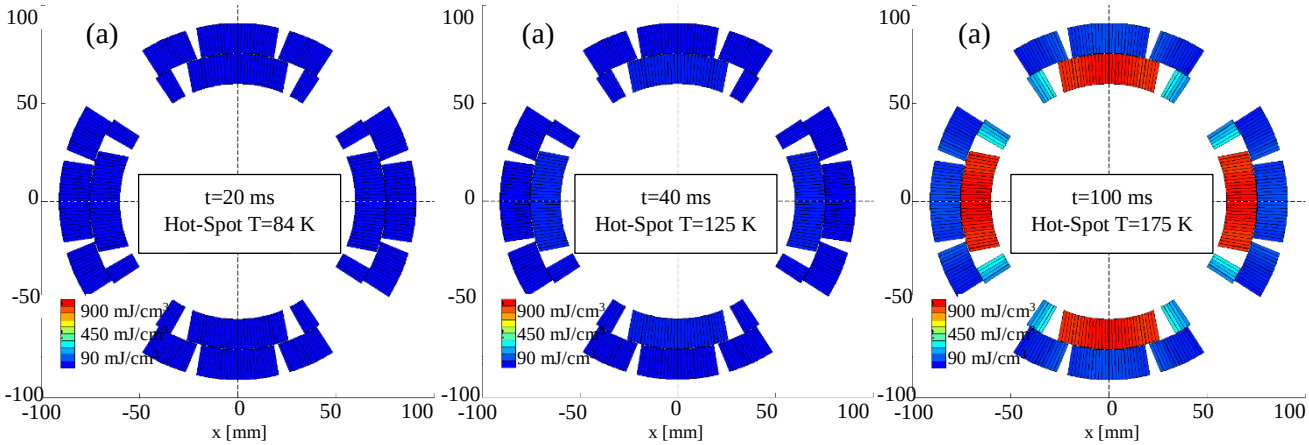


Figure 6.14. Simulated distribution of ISCL per unit volume generated in the quadrupole coil windings cross section at (a) 20, (b) 40, and (c) 100 ms after triggering CLIQ.

After transition from the superconducting to the normal state the Ohmic loss is the main driver for temperature increase in the magnet. In other words, the inter-filament and inter-strand coupling loss are needed to provoke a quench in the coil winding pack, whereas the losses produced according to the Joule's law provide enough heat to increase the overall quench resistance and protect the magnet. Higher losses are present in the inner layer since it has higher magnetic field, hence higher magneto-resistivity. Additionally, the poles that obtain first positive peak of the CLIQ current discharge produce higher Ohmic loss.

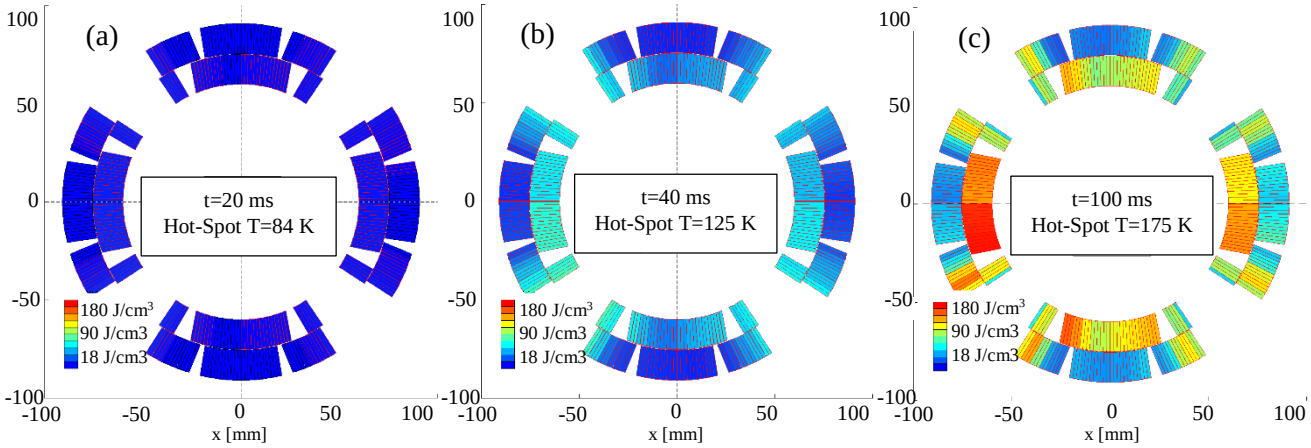


Figure 6.15. Simulated Ohmic loss per unit volume generated in the quadrupole coil windings cross section at (a) 20, (b) 40, and (c) 100 ms after triggering CLIQ.

The temperature distribution in the magnet windings is mainly dependent on the local deposited Ohmic loss, as it can be observed by comparing Figure 6.16 and 6.15. Namely, the blocks that are subject to higher Ohmic loss are getting higher temperatures. The subsequent rise of R_Q is due to the increase in the coil temperature through Ohmic losses.

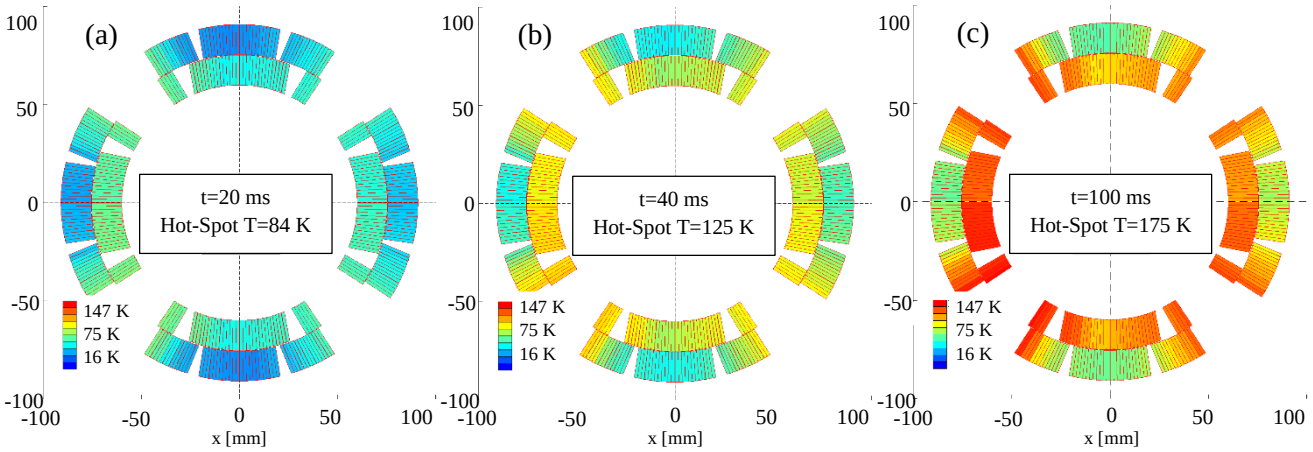


Figure 6.16. Simulated temperature distributions in the HQ02b magnet cross section, at (a) 20, (b) 40, and (c) 100 ms after triggering CLIQ.

Performed tests included measurements with initial current in the range 6 to 12.8 kA. The comparison between measured and simulated magnet currents is presented in Figures 6.17-6.19. The simulation and test at nominal current provided better agreement, than in the case at lower currents. In fact, at higher current the quench limits are lower, so the quench is initiated and propagated faster. As one can notice from Figure 6.10b, the magnetic field within a turn may vary significantly (for instance turns in the inner layer close to the mid-plane of the magnet). Furthermore, in the model a turn is represented by the average magnetic field and it is assumed that after a quench entire block volume becomes resistive. However, in reality, at lower current level CLIQ can quench immediately only high-field strands and then the remainder of a turn is being transferred to the normal state due to the heat propagation. The heat propagation velocity varies significantly with current level and for lower current the aforementioned assumption is less accurate. In fact, for current levels in the range

6 to 9 kA (see Figures 6.18 and 6.19), the difference between measurement and simulation is larger as compared to high current levels (12.8 and 14.6 kA). In other words, at low current where the quench propagation is lower, the energy stored in the CLIQ capacitor bank may not be sufficient to start a simultaneous transition to the normal state in the entire cross-section of the coil winding pack as it occurs in the model.

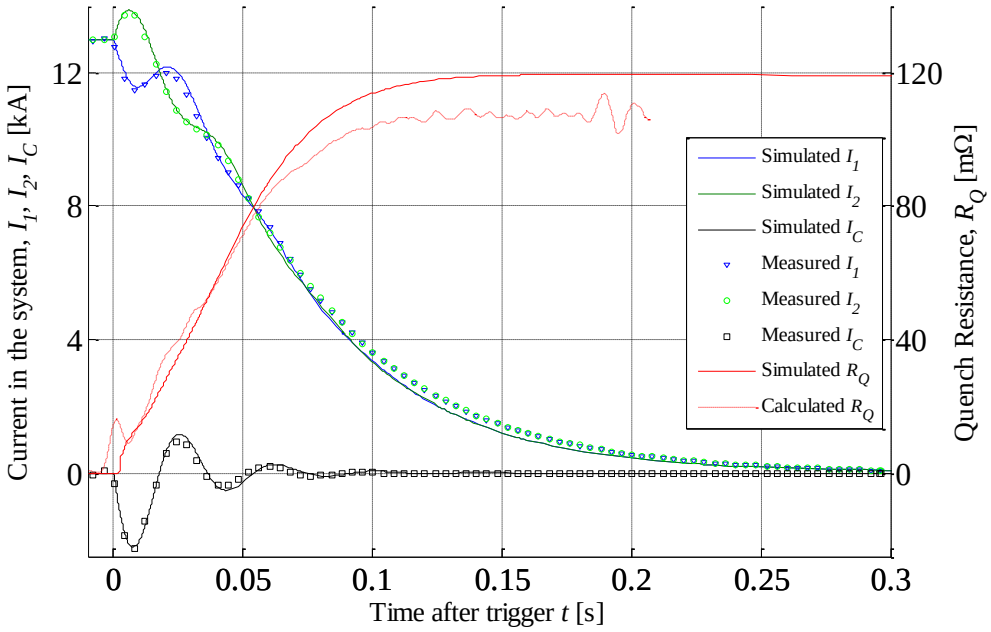


Figure 6.17. Magnet discharged by CLIQ after the triggering CLIQ at 12.8 kA. Measured currents I_C and I_2, calculated current I_1=I_2-I_C, versus time. Effective coil resistance R_Q, versus time. Simulated I_1, I_2, I_C, and R_Q, versus time.

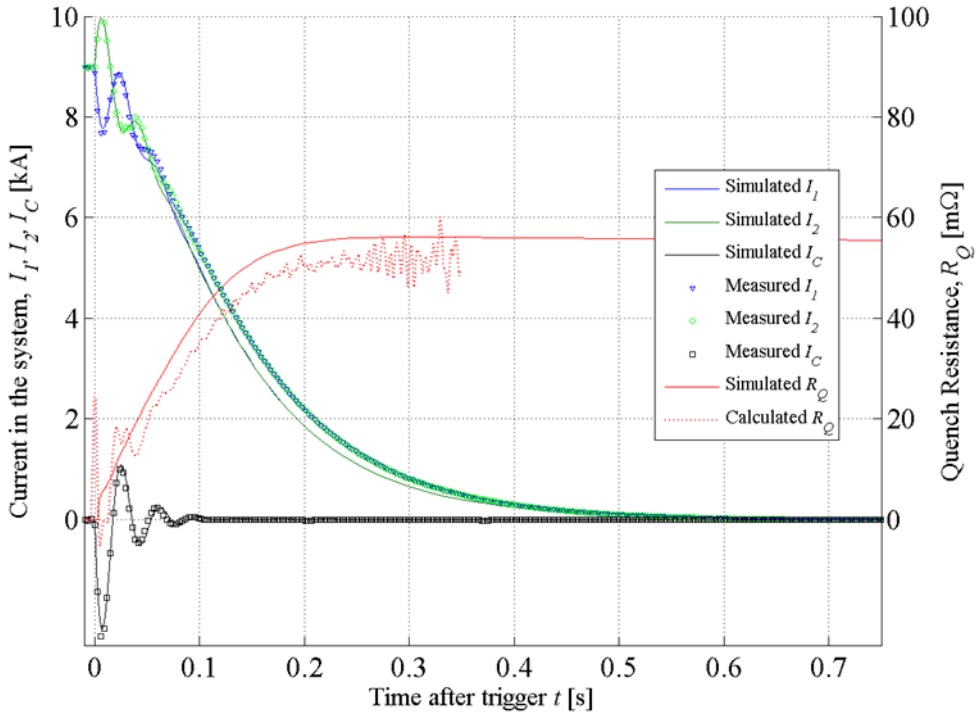


Figure 6.18. Magnet discharged by CLIQ after the triggering CLIQ at 9 kA. Measured currents I_C and I_2, calculated current I_1=I_2-I_C, versus time. Effective coil resistance R_Q, versus time. Simulated I_1, I_2, I_C, and R_Q, versus time.

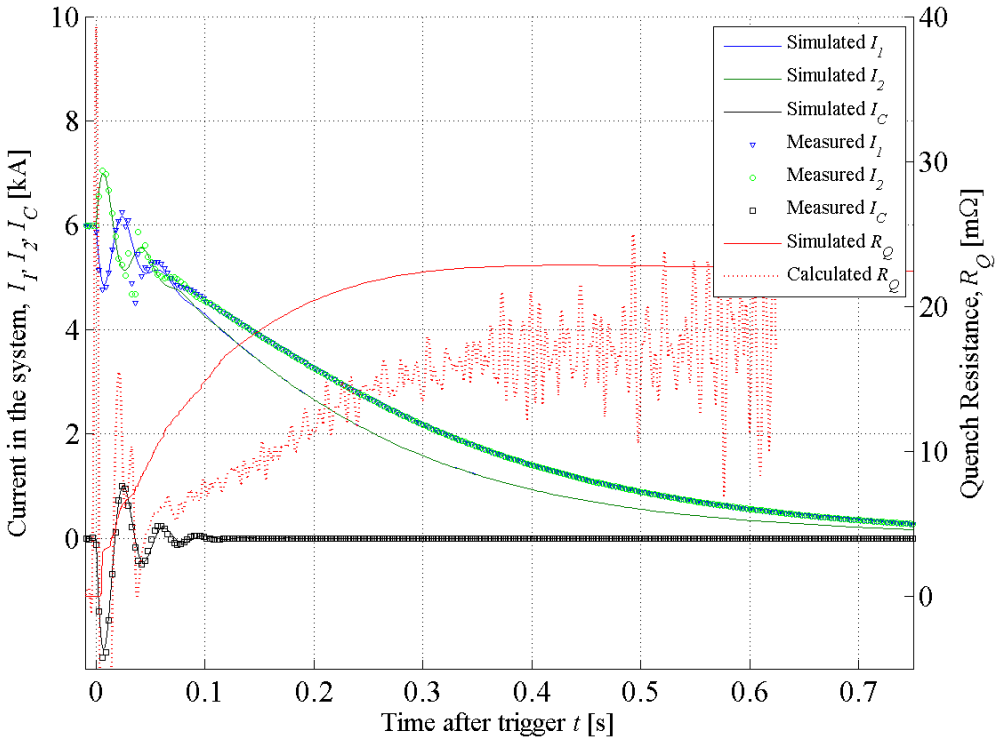


Figure 6.19. Magnet discharged by CLIQ after the triggering CLIQ at 6 kA. Measured currents I_C and I_2, calculated current I_1=I_2-I_C, versus time. Effective coil resistance R_Q, versus time. Simulated I_1, I_2, I_C, and R_Q, versus time.

A good figure of merit of the performance of a quench protection system is the quench load, calculated as the time integral of the square of the magnet current $\int I_1^2 dt$. The quench load is proportional to the energy deposited in the magnet hot-spot, however only the former can be practically measured during tests. Figure 6.20 shows a comparison between the measured and simulated quench load for the above-mentioned CLIQ tests. Additionally, since the model predicts the magnet behaviour with good accuracy, it is possible to apply the same model parameters and simulate the ideal case of a protection system capable to quench the entire coil winding pack at t=0 s. Such an ideal system transfers the whole magnet to the normal state instantaneously. It is worth noticing, that the presented CLIQ system is not far from the ideal case as the tested unit was sufficient to quickly initiate voluminous quench in the magnet.

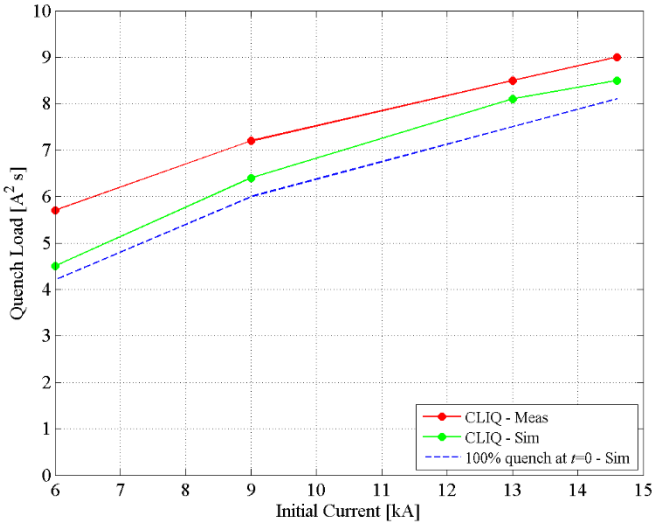


Figure 6.20. Comparison of quench load between measurements, simulations and an ideal quench protection system.

From Figure 6.20 it can be seen that the simulations underestimate the quench load. In fact, the transport currents calculated during the simulation always stays below the measured ones (see Figures 6.11 and 6.17-6.19).

6.3. MQXF

The HQ02b magnet is the model for a new class of quadrupole magnets made of Nb_3Sn superconductor. A good agreement between measured and simulated results allowed analysing the performance of a full-scale magnet with a magnetic length of 4 or 6.8 m (called MQXF, see Table 6.3). Figure 6.23 illustrates the MQXF magnet cross-section and magnetic field map of a half of a pole. A series of simulations allowed assessing the performance of various protection systems based the 1-CLIQ or 2-CLIQ configurations as presented in Figure 6.21.

Parameter	Value	Unit
Nominal current, I_{nom}	17320	kA
Operating temperature	1.9	K
Self-inductance at I_{nom}	8.2	mH/m
Magnetic length	4 or 6.8	m
Number of turns per pole	50	
Number of strands	40	
Strand diameter	0.85	mm
Bare cable width	18.1	mm
Bare cable thickness	1.5	mm
Insulation thickness	0.15	mm
Copper/Nb3Sn ratio	1.13	
Filament twist pitch	19	mm
RRR of the copper matrix	140	

Table 6.3. MQXF Magnet Parameters

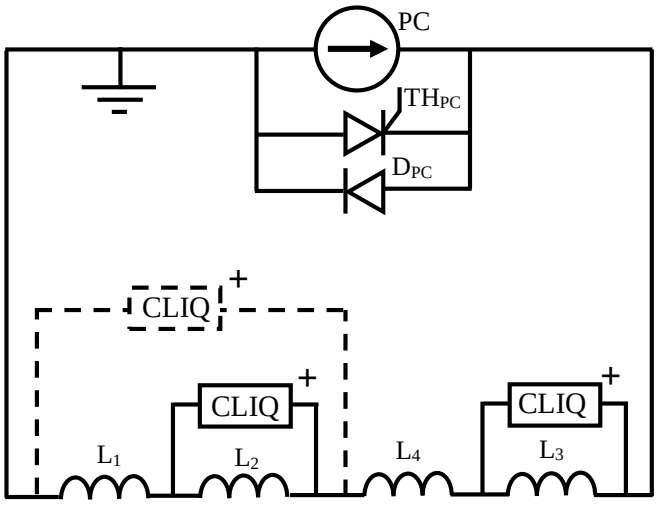


Figure 6.21. Schematic of the simulated MQXF magnet with one CLIQ unit (dashed) and two CLIQ units (continuous)

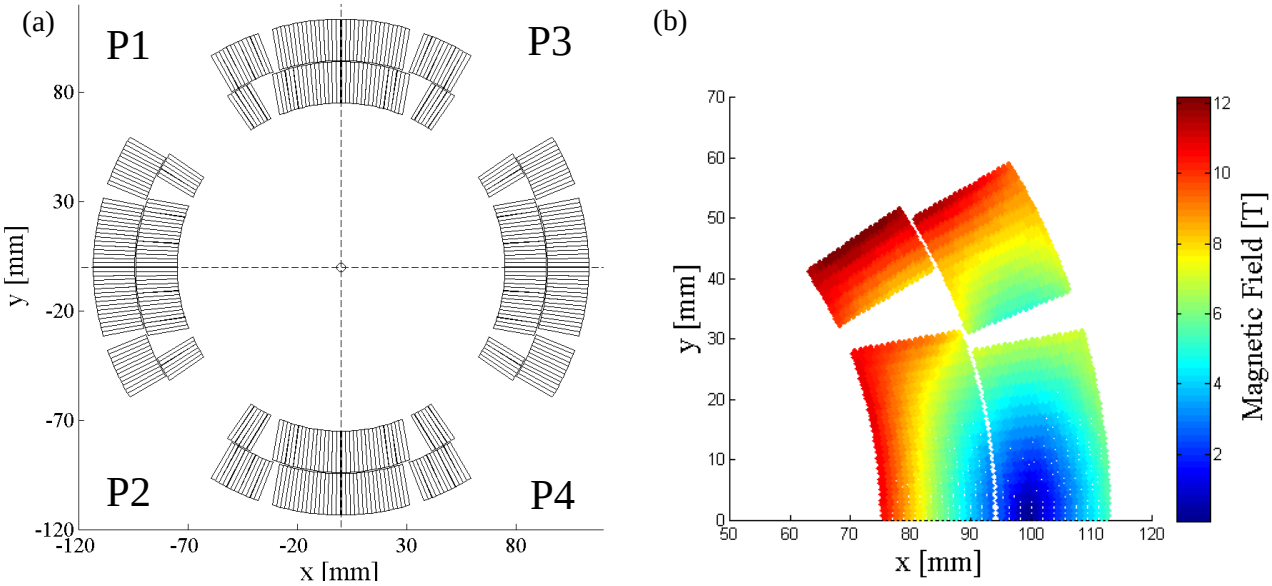


Figure 6.22. (a) Cross-section of the magnet, (b) magnetic field distribution in half a pole of the MQXF magnet, simulated using ROXIE

The model of the simulated 6.8 meter long MQXF magnet is divided equally into 8 electrical parts N_E=8, each layer of the magnet constituting a separate part. Since the number of windings in the outer and inner layer is different, different electrical parts are modelled with a different number of thermal blocks, namely 28 blocks and 22 blocks, respectively. Thus, $\mathbf{n}_B$=[28 22 28 22 28 22 28 22]^T and N_B=200 blocks.

Since the MQXF magnet is still in the design phase the simulations can provide useful information on magnet design as well as the performance of quench protection system. The temperature reached in the magnet hot-post is a key design parameter and was employed to compare the performance of various simulation results. For that purpose, two groups of simulations have been executed:

1. Optimization of the CLIQ configuration

   The maximum voltage to ground in the circuit is an important parameter in the design of a protection system; usually, it is of interest to keep its value to the range 500 to 1000 V. Hence, two configurations were tested in the first place: one single CLIQ unit charged to 1 kV and two CLIQ units charged to 500 V. Figure 6.23a presents the simulated hot-spot temperature obtained in the magnet after a quench at a nominal current of 17.3 kA protected by either of the two above-mentioned CLIQ configurations, as a function of the capacitance of each CLIQ unit in the range 1 mF to 120 mF. It can be observed that for each CLIQ configuration a minimum value of capacitance exists that allows protecting the magnet maintaining the hot-post temperature below the target limit of 350 K. On the other hand, it can be noticed that at this current level increasing the CLIQ capacitance over 50 mF does not improve the system performance.

Furthermore, the performance of the two configurations were analysed at different current levels fixing the capacitance of each CLIQ unit to 60 mF. Thus, the 1-CLIQ-1-kV system has twice as much stored energy as the 2-CLIQ-500-V system. The simulated hot-spot temperature is presented in Figure 6.23b. The 2-CLIQ-500V configuration is capable of protecting the magnet for currents greater or equal to 7 kA. For current levels from 7 kA to 17 kA, the hot-spot temperature is around 200 K. The reason for this result is that for higher current levels the discharge is faster as compared to the lowered current levels, thus the time-integral of the current remains almost constant in various simulations. On the other hand, one single CLIQ unit charged to 1 kV is sufficient to protect the magnet at every current level. For currents greater than 17 kA both presented systems perform in similar way as the energy stored in capacitor bank is sufficient to quench large portions of the magnet in first few milliseconds after CLIQ activation.

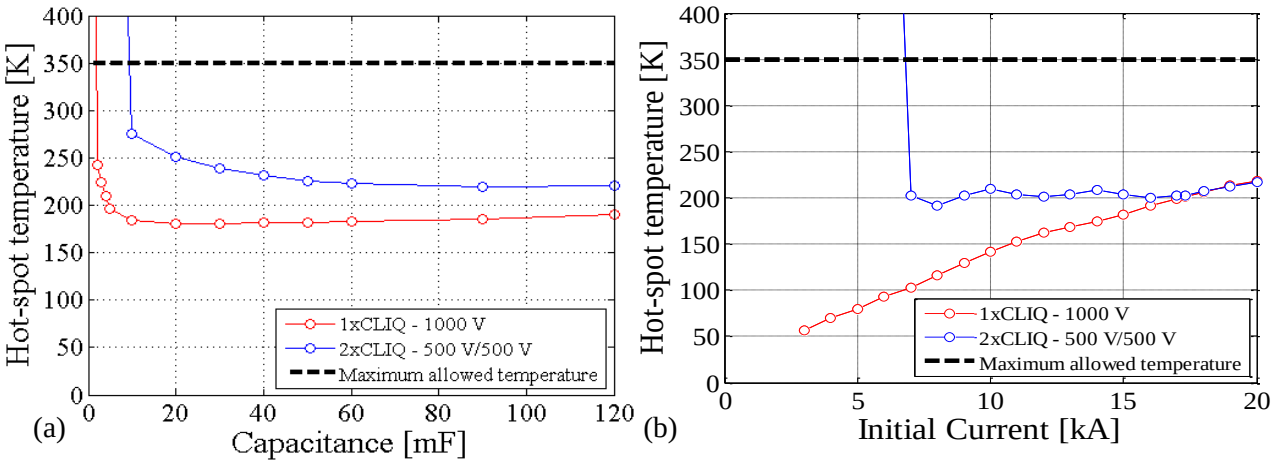


Figure 6.23. Simulated hot-spot temperature versus capacitance (a) and initial current (b) after triggering single CLIQ unit (1000 V) and two CLIQ units (500 V each).

2. Optimization of the magnet parameters for the highest performing CLIQ system

Analysis of simulation results obtained for CLIQ system optimization was an input for the next stage of parametric sweep study. Namely, 1-CLIQ-1-kV as the highest performing configuration was adopted as the baseline for a set of simulations with varying magnet design parameters such as RRR and filament twist pitch.

Figure 6.24a illustrates results of the parametric sweep study for the value of the RRR of the copper matrix of the superconductor. The RRR has an impact on the amplitude and time-constant of the inter-filament coupling loss, on the copper resistivity, and on temperature reached in the hot-spot. In fact, lower value of RRR results in higher copper resistivity, smaller IFCL amplitude, and smaller IFCL time-constant; and vice versa. To sum up, for higher RRR less IFCL are produced, but they develop faster. The two effects (on IFCL amplitude and time constant) may or may not compensate each other. Thus, due to high complexity of the model it is hard to predict even qualitatively results and case by case simulations are mandatory.

Furthermore, the effect of the RRR on the hot-spot temperature is twofold. Firstly, lower RRR translates to higher resistivity of copper stabiliser, hence more Ohmic loss is generated for the same current; thus, for the exact same current profile a strand with lower RRR will reach a higher temperature. Secondly, a conductor with lower RRR has a higher resistance per unit meter; hence, the increased resistance of the normal-conducting portions of the magnet may force a faster decay of the magnet current. Again, the two effects may or may not compensate each other. As a result, the magnet performance cannot be predicted *a priori* and parametric sweep of RRR greatly improves model analysis and provides insights on the optimal value of this parameter.

As explained in [14], both the amplitude and the time-constant (τ_{if}) of inter-filament coupling loss constant are proportional to the square of the filament twist-pitch. Hence, the influence of the filament twist-pitch on the IFCL deposited during a CLIQ discharge is twofold. On the one hand, if a filament twist-pitch is too small the amplitude of induced loss is also small. On the other hand, too large filament twist-pitch may lead to high IFCL time-constant as compared to the CLIQ oscillation period ($\tau_{if}>>1/f_{CLIQ}$) and coupling loss may not effectively develop during fast oscillations. Once again, the two effects may or may not compensate each other and the simulations are very important to assess the optimised value of filament twist-pitch. Figure 6.24b provides information on the optimal value of filament twist-pitch (about 20 mm) minimizing the hot-post temperature for the given CLIQ configuration and operating conditions. On the contrary, for values of filament twist-pitch smaller than 10 mm or larger than 50 mm, the magnet is not protected. However, the design value of the MQXF filament twist-pitch proves to be already optimised (see Table 6.3).

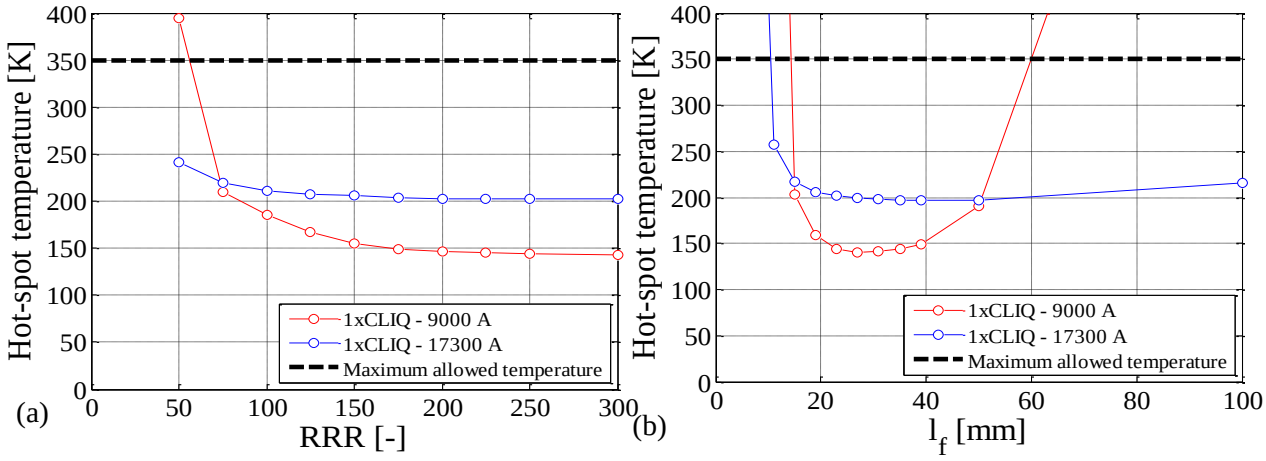


Figure 6.24. Simulated hot-spot temperature versus RRR (a) and filament twist-pitch (b) after triggering single CLIQ unit at two current levels (9kA and 17.3 kA).

6.4. MB

The magnets presented so far are models for magnets to be installed in the future LHC upgrades. However, during the LHC operations in current configuration it may be required to provide a backup

solution for existing magnets with failing quench protection systems. For instance, the quench heaters due to their low thickness are prone to mechanical stress and might get damaged as in the case of MQXC2 magnet (see Figure 6.3). For that purpose tens of simulations of a CLIQ-based protection system to be applied to the twin aperture main LHC dipole magnet (MB, see Figure 6.26) were performed for various operating conditions. Table 6.4 presents the main parameters of the magnet [37]. The schematic depicted in Figure 6.25 obviously differs from the complete LHC main dipole circuit and illustrates an experimental setup for testing CLIQ on a stand-alone MB magnet.

Parameter	Value	Unit
Nominal current, I_{nom}	11850	kA
Operating temperature	1.9	K
Self-inductance at I_{nom}	6.8	mH/m
Magnetic length	14.312	m
Number of turns per pole	15(in)/25(out)	
Number of strands	28(in)/36(out)	
Strand diameter	1.065(in)/ 0.825(out)	mm
Bare cable width	15.10	mm
Bare cable thickness	1.9/1.48	mm
Insulation thickness	0.15	mm
Copper/NbTi ratio	1.65/1.95 ±0.05	
Filament twist pitch	15	mm
RRR of the copper matrix	150	

Table 6.4. Magnet parameters

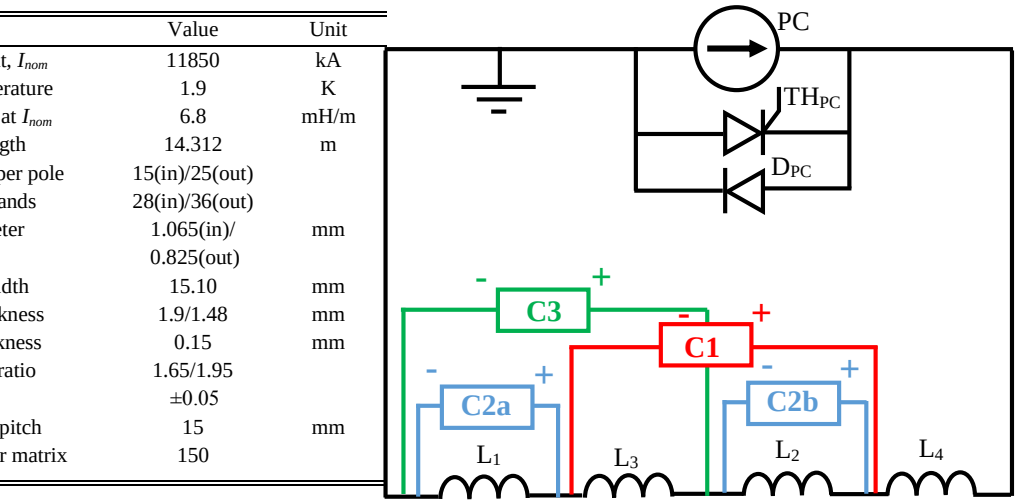


Figure 6.25. Simulated superconducting circuit

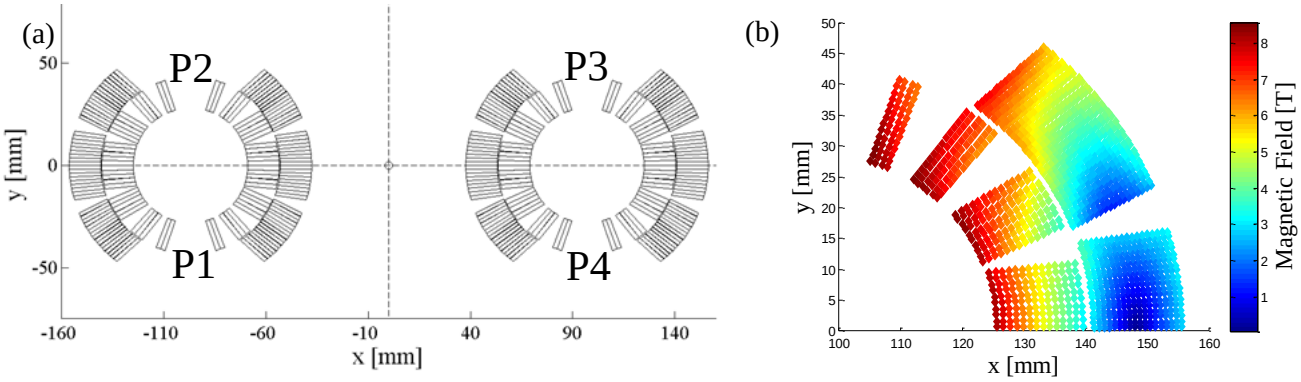


Figure 6.26. (a) Cross-section of the twin-aperture MB dipole magnet. (b) Magnetic field distribution in half a pole of the MB magnet, simulated with ROXIE.

Protection of long magnets poses a considerable challenge as CLIQ performance decreases with the magnet length. Paper [17] presents a thorough discussion on CLIQ optimization techniques. First of all, the power per unit volume deposited by CLIQ is roughly proportional to the square of the magnetic-field change generated in the superconducting cable. In turn, the magnetic-field change in first approximation is proportional to the current change introduced by CLIQ. As presented in [18] the peak introduced current change is directly proportional to the CLIQ charging voltage and inversely proportional to the equivalent circuit inductance. The former is limited by a maximum safe

voltage, whereas the latter depends on coil geometry, positioning of CLIQ connections, and electrical order of the magnet poles. It can be shown that the equivalent inductance can be greatly reduced by introducing opposite current change in poles that are physically adjacent. In many practical cases such an optimized configuration can be easily implemented with no effect on stationary magnet performance. Furthermore, a magnet can be protected by multiple CLIQ units (so called Multi-CLIQ). Such a configuration can further reduce the equivalent inductance.

The adopted model of MB magnet is composed N_E=4 electrical parts, each composed of 36 blocks, i.e. $\mathbf{n}_B$=[36 36 36 36]^T. The simulations are performed at nominal current of 11.85 kA. The magnet is protected only by the CLIQ system, and the three different CLIQ configurations summarised in Table 6.5 are tested.

#	Capacitance [mF]	Voltage [V]	Resistance [mΩ]	Trigger [s]	Positive Node	Negative Node	Description
1	50	1000	20	150	0	2	1xCLIQ, non-optimal configuration
2	50	1000	20	150	1	3	1xCLIQ, optimal configuration
3	50/50	500/500	20/20	150/150	0/2	1/3	2xCLIQ, optimal configuration

Table 6.5. Operating parameters of simulations

The simulation results of all three CLIQ configurations are presented together in Figure 6.27. In the first configuration one CLIQ unit is attached in parallel to one aperture of the simulated dipole magnet (see Figure 6.25, in green). Since there is a negligible magnetic coupling between the two magnet apertures such a configuration is non-optimised. The second configuration introduces opposite current changes in adjacent poles as both terminals of the CLIQ unit are attached in the middle points of the two magnets (see Figure 6.25 in red). Hence, the equivalent magnet inductance decreases and oscillation frequency and peak current are higher than in previous case. As a result, magnet is quenched more quickly and the current is discharged faster. The third configuration includes two CLIQ units charged at half voltage (see Figure 6.25, in blue); thus the total energy stored in their capacitor banks half of the other two configurations. However, such a configuration is optimal as adjacent poles see opposite current derivative. To sum up, Configuration 2 performs much better as compared to configuration 1. Moreover, Configuration 3 with half stored energy discharges current faster than Configuration 1 and almost as fast as Configuration 2.

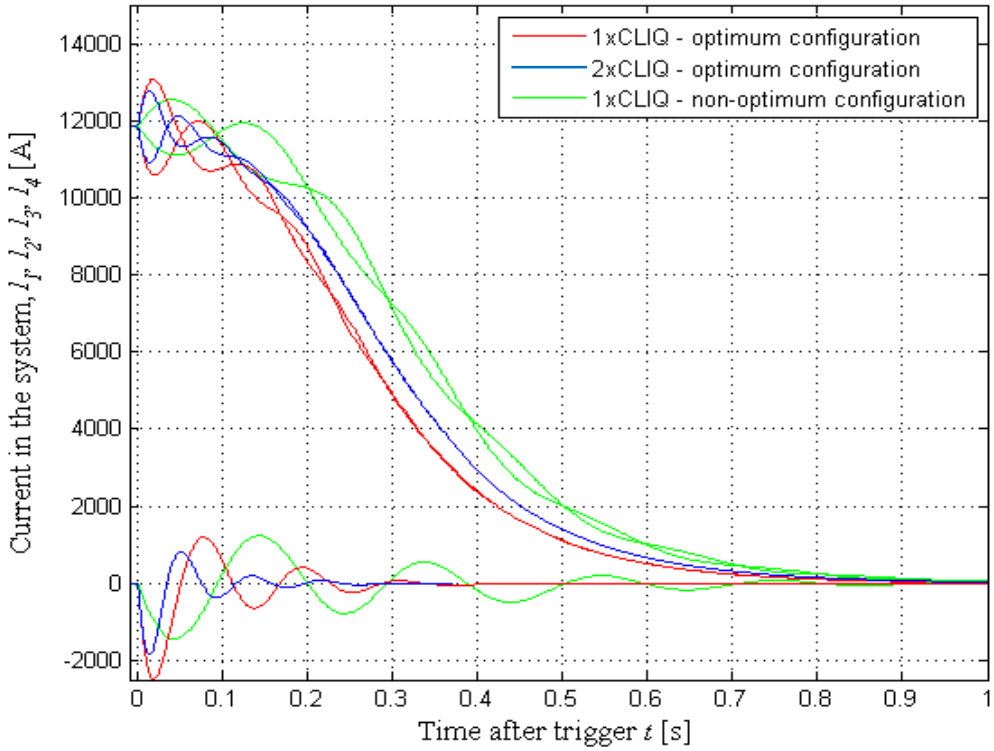


Figure 6.27. Simulated discharge current for MB magnet

6.5. RB circuit

Each LHC sector (see Figure 1.1) contains one main dipole circuit (RB), each composed of N_{mag}=154 superconducting dipole magnets denoted by M001-M154. Due to the high complexity of the system it is necessary to simulate transients occurring in the chain of magnets in order to assess its performance under different operating conditions. Moreover, analytical calculations are not sufficient for properly analysing the behaviour of superconducting magnets due to the presence of magnetization effects, eddy currents, and coil-to-ground parasitic capacitances. In the case of a quench in a magnet of the chain or a problem with the Power Converter (PC), a Fast Power Abort (FPA) is fired in order to protect the magnets and the PC. After the FPA voltage waves occur in the RB circuit. They must be analysed in depth as they can lead to spurious triggering of the quench detection system [28, 29]. Furthermore, for long chains of magnets the application of a full electro-thermal model might be computationally expensive. Hence, as the number of magnets in a chain increases, the complexity of a model must be reduced accordingly. It was shown in the past that the effects with the most important impact on the performance of the circuit and its detection system [46-50] can be reproduced with a limited number of equivalent lumped elements. Such an approach was applied in [29] and resulted in a very good agreement between measurements and simulations.

A schematic representation of the RB circuit is shown in Figure 6.28. The PC is modelled as a single controlled current-source. In the case of the FPA, the PC is by-passed by a crowbar (a thyristor $TH_{Crowbar}$ becomes conductive). At the output of the PC a low-pass LC filter is present, composed of an inductor L_{Filter}=285 μH, a capacitor C_{Filter}=100 mF, and a resistor R_{Filter}=10.1 mΩ).

The LC filter has a theoretical resonance frequency of $f_{Filter}=1/(2\pi \cdot sqrt(L_{Filter} \cdot C_{Filter}))\approx 31.8$ Hz. The current flowing through the PC during the initial phase of the current ramp should have continuous first derivative to avoid exciting the LC filter and the chain of superconducting magnet. Thus, the power converters applied in the LHC employ parabolic-exponential-linear-parabolic (PELP) function generator.

Each dipole magnet is protected by quench heaters and by a by-pass diode. Additionally, the stored energy in the chain is too large to absorbed by one of the by-pass diodes which are designed to sustain the circuit current only for a limited time. For this reason, the chain of magnets is discharged with two energy-extraction units. The former unit is located in the middle of the chain (between magnets M077 and M078), and the latter is connected at the end of the chain (between magnet M154 and negative side of the power converter). Each energy extraction system is composed of a resistor ($R_{EE1}=R_{EE2}=148$ mΩ), an electromechanical switch (SW_1, SW_2) and snubber capacitors to reduce the voltage spikes at the switch opening ($C_{SN1}=C_{SN2}=53$mF). The two energy extraction units are triggered with a delay of 350 and 600 ms after a FPA, respectively, to avoid superposition of the voltage waves generated at the PC switching-off and switch openings. The value of extraction resistor depends on the amount of energy stored in the chain. During the first run of the LHC (energy of beam was equal to 3.5-4 TeV) the dipole magnets were working at half of the nominal current (6 kA), hence 148 mΩ was sufficient to protect the RB circuit. However, at the restart of the LHC the main dipoles are supposed to work at the nominal current (11.85 kA) and the beam energy should reach 6.5-7 TeV, thus extraction resistance will be increased to 300 mΩ.

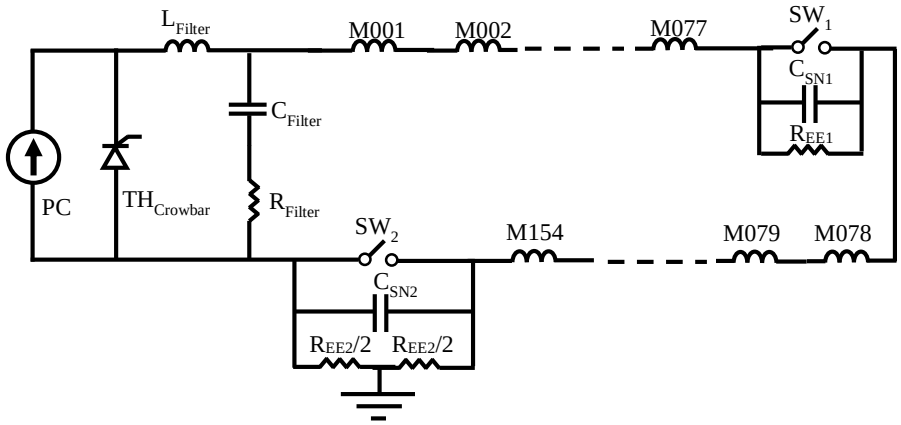


Figure 6.28. Schematic of the LHC main dipole circuit. [28]

Each main dipole magnet is composed of two apertures connected in series with a total self-inductance $L_{mag}=98.7$ mH. Hence, one entire RB circuit has a self-inductance of about 15.2 H and a stored energy of about 1.1 GJ at the nominal current of 11850 A. Each magnet is by-passed by a resistor R_p for smoothing fast voltage oscillations.

Each magnet of the chain is modelled with the lumped-element circuit shown in Figure 6.29. A fraction k=0.75 of the self-inductance L=49 mH of each aperture is by-passed by a virtual resistor (R_1, R_2) which accounts for the eddy currents induced in the coils. The values of these parallel resistors are free parameters in the model that can be estimated from experimental results and modified to better represent the overall circuit behaviour and the inhomogeneous dynamic characteristic of particular apertures. According to [29] a good agreement between simulation and measurements is obtained for $7<R_{1,i}<10$, $7<R_{2,i}<10$, for $i=1,\ldots,N_{mag}$. Finally, the capacitors C=150 nF model coil-to-ground parasitic capacitances.

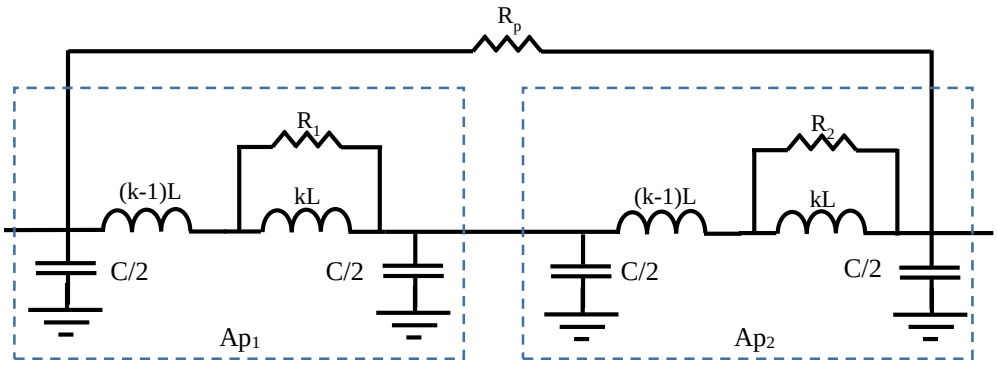


Figure 6.29. Lumped-element, simplified LHC main dipole magnet model [29].

Hence, the presented model does not account for thermal transients occurring in the superconducting magnets and only electro-dynamic effects are represented in a simplified form by networks of self-inductors, resistors, and capacitors. This network is relatively simple and simulations of a FPA transient can be computed in a very limited time, typically a few minutes.

The sample simulation presented here comprises a FPA at 2 kA at nominal dI_0/dt=10 A/s. In the model a parabolic-linear ramp is implemented; for the first 50 seconds the current increases according to a parabolic reference function, and then it increases linearly with a desired slope. Finally, at t=225 s, when the current is equal to 2 kA, a FPA is triggered and the voltage waves occurring in the RB circuit at the PC switching-off and at the switch openings are analysed. Nevertheless, the main purpose of FPA is current discharge in the circuit in order to protect the magnets and the PC. The current is discharged by switching off the PC and then sequentially activating both energy extraction units. As a direct consequence, the transport current is discharged with a time constant calculated as $N_{mag} \cdot L_{mag}/(2R_{EE})$ and equal to about 50 s (see Figure 6.30).

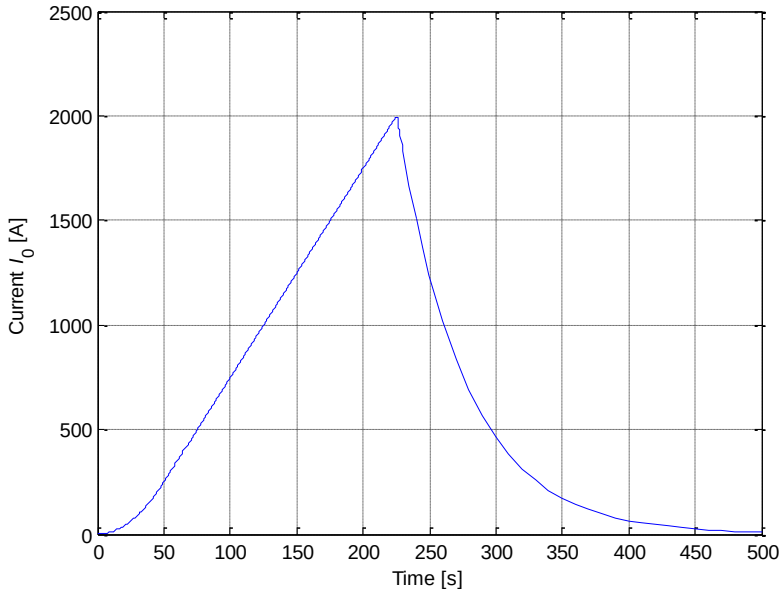


Figure 6.30. Fast Power Abort in the LHC main dipole circuit: Current discharge after firing two energy extraction systems

Figure 6.31 illustrates the voltage at the output the power converter after triggering a FPA at t=0. Before the FPA, this voltage is a sum of the inductive voltage across the magnet chain and the resistive voltage built across the warm elements of the circuit. Hence, $U_{PC,0}=R_{circuit}\cdot I_0+N_{mag}\cdot L_{mag}\cdot dI_0/dt\approx 160$ V, where $R_{circuit}\approx 1$ mΩ is the resistance of the non-superconducting components of the circuit. When the fast power abort is triggered, the voltage oscillates at a frequency equal to the resonance frequency of the LC filter.

The voltage transient across the magnet M001 (U_{mag}) following a FPA at 2 kA during current ramp (see Figure 6.30) is presented in Figure 6.32. The initial voltage at t=0 is $L_{mag}\cdot dI_0/dt\approx 1$ V. After the PC switching-off, the voltage oscillates at a frequency f_{Filter} due to the wave generated at the PC output and travelling through the magnet chain. After 350 ms, the first energy-extraction switch SW_1 is opened and the voltage decreases to $U_{mag}=-R_{EE}\cdot I_0/N_{mag}\approx -2$ V. Finally, the second switch SW_2 is opened and the voltage further decreases to about -4 V, which is the voltage drop across two energy extraction units divided by the number of magnets in the chain $U_{mag}=-2R_{EE}\cdot I_0/N_{mag}$. The snubber capacitors in parallel to each extraction switches smoothens the voltage transients following their opening. Quench detection is based on the comparison of the voltages across the two apertures of the magnet. If this difference is greater than a certain threshold for more than a certain validation time, the quench is detected and quench heaters are fired [28, 29]. Hence, a difference in the AC behaviour of the two apertures when they are excited by a voltage wave may lead to the spurious triggering of the quench protection systems. In turn, such accidents provoke unnecessary firing of the quench heater units which may result in additional stress to their electrical insulation.

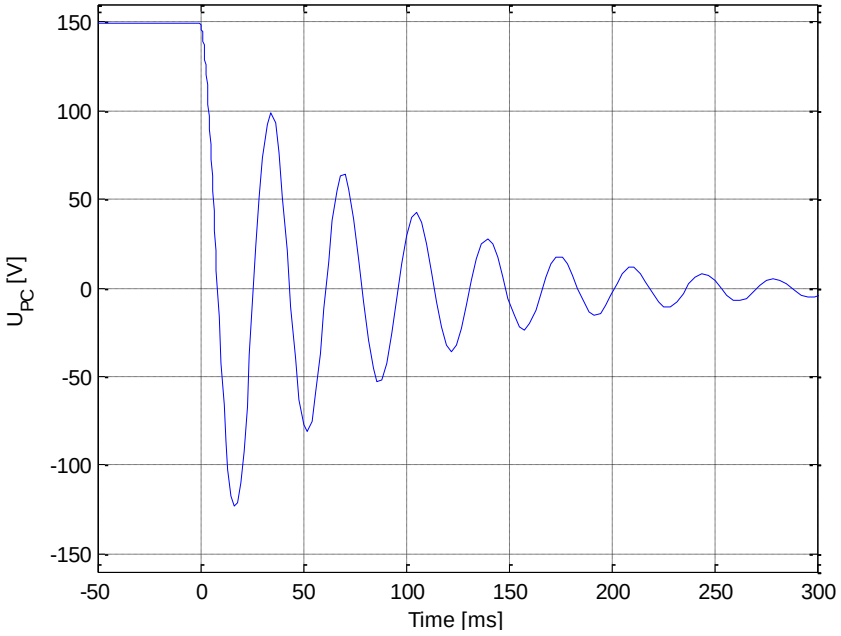


Figure 6.31. Simulated voltage across PC during FPA

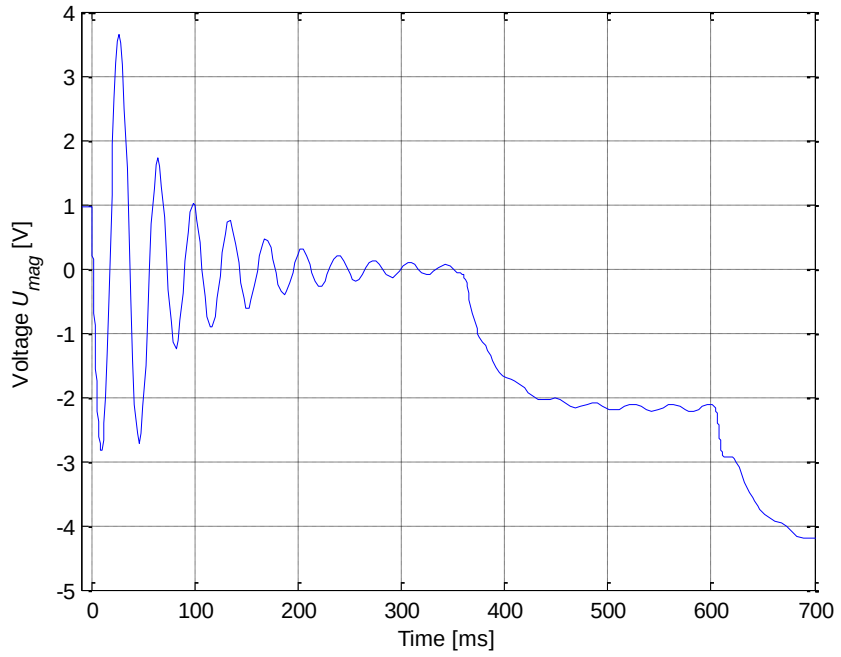


Figure 6.32. Simulated voltage across M001 during FPA

Figure 6.33 presents the maximum and minimum values of the voltage across each of the 154 magnets of the RB circuit after switching off the power converter (0 ms<t<350 ms). One can observe a wave propagation traveling through the chain of magnets.

Furthermore, the peak-peak amplitude of the oscillations of the voltage difference between the two apertures ΔU_{ap} of each magnet after the PC switching-off is presented in Figure 6.34. During the experiments performed on the RB circuit an unbalanced AC behaviour was observed in the two aperture composing various dipole magnets [53]. However, the model is capable of correctly reproducing such phenomena thanks to the appropriate choice of the virtual resistors R_1 and R_2 of each main dipole magnet model.

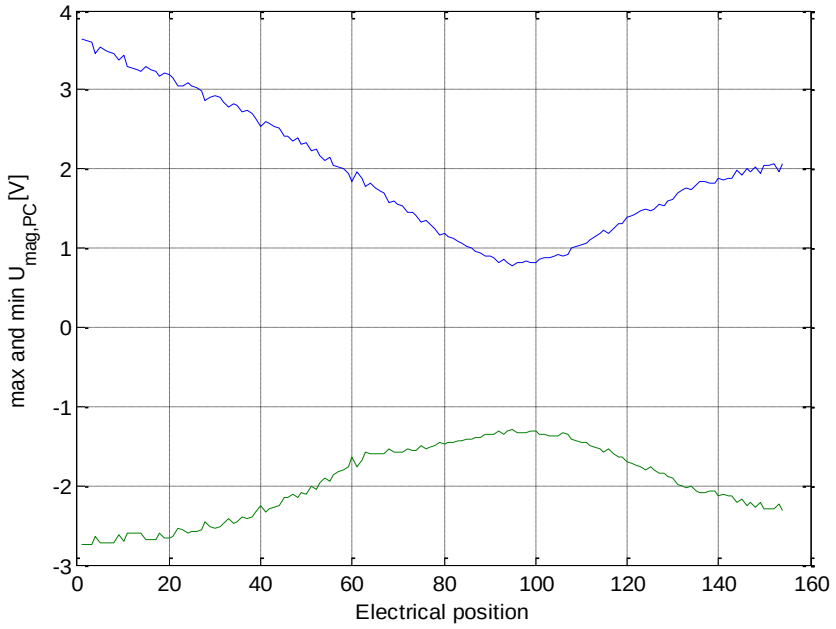


Figure 6.33. Maximum and minimum $U_{mag,PC}$ (200 ms<t<350 ms) at I_0 = 2 kA and dI_0/dt = 10 A/s

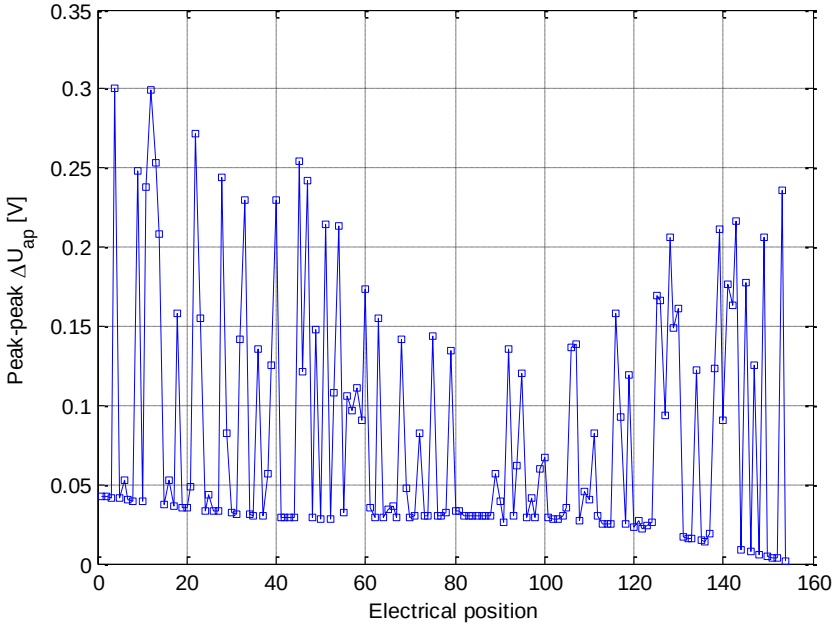


Figure 6.34. Peak-peak ΔU_{ap} of each dipole magnet after a FPA (200 ms<t<350 ms) at I_0 = 2 kA and dI_0/dt = 10 A/s

In conclusion, a simple model provided very good approximation of the behaviour of a chain of superconducting magnets. Detailed description as well as model validation against measurements can be found in [29].

7 Conclusion

The analysis of the protection of superconducting magnets after a quench is a key ingredient in the design and operation of superconducting magnets. A convenient simulation program needs to simulate complex electro-thermal transients accurately and in a relatively limited time. The simulations of superconducting magnets at CERN are performed in ROXIE (three dimensional, finite element method) and PSpice (two dimensional, lumped-element method). However, both environments have some drawbacks. On the one hand, the former does not implement a feedback of inter-filament and inter-strand coupling currents on the transport current in the magnet and is time consuming. On the other hand, the latter executes faster and implements aforementioned coupling mechanism, but supports only manual model development and has limited parametric sweep capabilities.

The presence of multi-domain problems, time-variant physical properties, and non-linear behaviours require developing a scalable, flexible, and maintainable simulation environment. The Quench Simulation Framework (QSF) employs a lumped-element dynamic electro-thermal model of superconducting magnets along with lumped-element models of quench protection methods currently used in accelerator magnets (by-pass diodes, energy extraction systems, quench heaters) as well as innovative ones (CLIQ). However, convenient methods need to be implemented that create lumped-element components reproducing the overall behaviour of relatively large portions of the system (typically 1-2% of the total volume) based on analytical equations describing effects occurring at a much smaller scale (typically by several orders of magnitude). In order to improve user experience while performing simulation, the QSF should possess the following key features:

1. Library of reusable components
2. Modularity
3. Scalability
4. Convenient, intuitive Graphical User Interface (GUI)
5. Automatic model development
6. Maintainability.

The developed QSF provides a unified approach that is easy to learn and enables quickly implementing physical equation describing modelled problem as well as re-using existing knowledge. The developed software architecture dynamically adapts to various modelling scenarios based on the defined input files. Furthermore, the proposed framework architecture allows the simulation environment to evolve on the basis of an increasingly validated library of components. As a result, the QSF is tailored to the needs of superconducting magnet modelling. The QSF provides a unique approach, with intuitive GUI and software architecture on top of the widely applied mathematical and

simulation environment MATLAB/Simulink. The framework architecture is divided into three main layers, namely the library of elements, the main application-server, and the graphical user interface (GUI). The developed software includes a dedicated GUI featuring schematic editor, parameter editor, parametric sweep, simulation control, and signal viewer. Thus, the user is able to effectively create a new model or a new set of simulations from the GUI level.

The first step for the creation of an automated, object-oriented simulation framework was the successful implementation of the same modelling features present in the existing OrCAD PSpice models developed and used at CERN. For that purpose, the Simscape object-oriented programming language has been employed. Subsequently, a thorough validation of the model results against equivalent PSpice simulations was performed. The required flexibility of the QSF was achieved by creating a library of reusable components. This QSF library is composed of both static and dynamic components in order to obtain an optimal representation of the simulated electrical, thermal, and magnetic objects. However, due to application of Object Oriented Programming (OOP) techniques (inheritance, encapsulation) the resulting class hierarchy describing the model allows treating all sub-components in the same way. Hence, the QSF library representation is well-structured with defined relations between all elements. As a result, a strict way of extending the library has been obtained and the user has to only create a component from the blocks available in the standard Simulink library or in custom Simscape ones. Eventually, the dedicated procedure generates a class prototype that can either be directly applied in the model design procedure or adjusted to account for additional conditions.

The QSF is composed of three layers, namely a Simulink solver and the QSF library (bottom layer), the main application (middle layer), the GUI (top layer). The main application architecture is decentralized, hence there is a loose coupling between its modules. Following this approach the project is easier to manage, extend, and maintain. Each of the described modules of the framework can be effectively re-used. In other words, by default, the framework is a black box with defined inputs and outputs; however there is also a transparency of each layer, thus the user can access a particular module of the QSF and use it according to current needs. On the one hand, the hybrid Simscape-Simulink library along with mutual inductor library can be easily extracted from the QSF structure and applied to create standard models. On the other hand, middle and top layer can be reapplied and adopted to model another physical problem by replacing the superconducting magnet library with a new one. This makes it possible to extend the framework with relatively small effort at each level.

The QSF allows modelling superconducting magnets with adaptable level of detail. Due to the formal representation of the model description it is possible to build models composed of a variable

number of sub-components. What is more, the scalability in terms of circuit modelling means the ability to simulate stand-alone magnets as well as chains of magnets (for example the RB circuit of the LHC section). As a consequence of the increasing number of magnets in the circuit, the ability to reduce their complexity (decreasing accuracy) becomes important.

In order to obtain maintainability, coherent naming conventions for physical variables names (Hungarian notation) and names of classes, fields, and methods (Camel Case notation, Pascal Case notation) were adopted. Moreover, the code and library components are controlled by a version control system (Git). The QSF is equipped with unit tests (for classes and library components) and acceptance tests in order to validate its performance and reduce the occurrence of errors. The QSF is also equipped with procedures performing parameter range check, stability analysis of the matrix describing the system dynamics, and validation of electrical circuits.

The goal of the Graphical User Interface of the QSF is improving the simulation development process by executing appropriate software modules in a very convenient way. Application of the MATLAB object-oriented language along with the design patterns (mediator and singleton) resulted in scalable and maintainable software architecture. The design of the GUI was further simplified by the modular structure of the main application. For the same reason, the QSF can be conveniently operated both through the GUI and from the MATLAB command line. The GUI enables the user to sketch the model configuration in a similar fashion as in the case of available simulation software on the market. Nevertheless, the GUI does not provide ability to draw a model with the mouse, but it requires selecting block parameters and then adding them to the components list and schematic. However, it is more robust to errors as the available options are limited and validated. For this purpose the QSF employs two netlists to define model structure, namely a list of the components used in the model (configuration netlist) and a list of connections between them (connection netlist). In order to locate a block on the two dimensional plane of the model only two parameters are required: the reference block and the direction, whereas the connections are defined as pairs of from-to ports. Hence, the user can create a model schematic without any constraint on its structure. However, since the resulting model schematic is *a priori* unknown, the factory design pattern and appropriate interfaces have been employed in order to accomplish this task. As a result, the GUI is a useful and reliable tool for executing the complex task of handling simulation design and execution.

A key feature of the QSF is the automatic simulation development. The framework is capable of automatically building the schematic based on one single input Excel file containing the model configuration. As a result, the modeler can focus only on high level model definition and analysis of the simulation results without the need to acquire knowledge of the programming language and of

the simulation solver. With this approach, runs of multiple simulations and parametric analysis can be effectively performed.

Another interesting feature of the QSF is its ability to perform a parametric sweep study on a new level as compared to default capabilities of most popular simulation environments (MATLAB/Simulink, PSpice, PSIM, MultiSim, etc.). In fact, the parametric sweep module and the simulation control module allow running sets of simulations with varying model structure and/or varying parameters. This flexibility of dynamically changing the model components during a parametric analysis represents a definite advantage over comparable simulation programs.

The QSF is able to perform simulations with acceptable accuracy in the required time frame. However, the parallel computing module allows further decreasing the effective simulation time. The proposed, simple master-slave architecture enables performing multi-core, multi-machine simulations without the need to purchase another MATLAB toolbox dedicated to simulation and application parallelization (Parallel Computing Toolbox, Distributed Computing Toolbox).

Furthermore, the QSF is equipped with a module responsible for generating stand-alone, executable models. This module is based on Real-Time Workshop, the Simulink extension that allows creating an executable code from a Simulink model. On the one hand, the simulations are executed faster as the wealth of functionalities provided by Simulink schematic window is disabled. On the other hand, the compiled model encapsulates the knowledge applied in the Simulink model.

As a consequence of the successful validation and application of the framework requirements, the QSF has become a valuable alternative to OrCAD PSpice. Next, additional components were added to the QSF in a well-structured and self-documenting way, thus further extending the model capabilities and improving its flexibility. The features of the framework made it the natural choice for the prediction of magnet behaviour and performance analysis during various test campaigns comprising quench protection studies performed in the CERN magnet test facility (MQXC2 and HQ20b magnets). The developed model successfully reproduced the complex electro-thermal transients following a quench in a superconducting magnet protected by an Energy Extraction system, Quench Heaters, and/or CLIQ); the simulated results were in very good agreements with the measured data over a wide range of operating parameters. Finally, due to highly automatized character, the QSF has been also applied to assess the performance of optimized configuration of the CLIQ system and new superconducting magnet design.

During the first 8 months of operation the QSF has executed over 5000 simulations, producing over 3 TB of simulation data. At the same time the framework provides a support in managing the resulting outputs in well-structured directories with a dedicated summary file per each output folder. The time required to create the model of a new superconducting magnet was reduced from 1-2 weeks

to 1-2 days. Once developed, each model can be reused and modified in a very convenient way. The simulation time of an existing magnet model is typically less than 1 hour. The framework provides a flexible interface which uses the Simulink only to solve a set of differential-algebraic equations characterising the model behaviour. To sum up, the QSF does not require sound experience in MATLAB, Simulink, and Simscape in order to build complex models of superconducting circuits.

Future work includes removing ROXIE/SOLENO dependency to calculate magnetic field maps. Once such a module is implemented, the QSF can become even more convenient to use. Automatic simulation development along with parallel computing module opened an easy way to implement parameters optimization, i.e. tuning the free parameters of the model in order to minimize error between measurements and simulations. An interesting and relatively easy QSF extension is a three-dimensional block calculating the initial development of a natural quench in the magnet and predicting the time required to detect it.

Finally, the proposed architecture of both library of components and main application is very generic, hence it can be conveniently reapplied and adopted to any multi-physics modelling problem employing a lumped-element approach in MATLAB/Simulink.

Streszczenie

Celem pracy jest zaprojektowanie elastycznego, rozszerzalnego, przyjaznego użytkownikowi środowiska do modelowania stanów nieustalonych zachodzących w nadprzewodzących magnesach. Symulacje są podstawowym narzędziem do oceny osiągów magnesów oraz metod ich ochrony przed zjawiskiem quench'u. Środowisko wykorzystuje elementy skupione modelujące zjawiska elektryczno-cieplne oraz dynamiczne zachodzące w nadprzewodzącym magnesie wraz z elementami skupionymi reprezentującymi metody ochrony podczas quench'u stosowane obecnie w magnesach używanych w akceleratorach (diody bocznikujące, systemy ekstrakcji energii, specjalne grzałki – quanch heater) oraz innowacyjne metody (CLIQ). Obwód cieplny nadprzewodzącego magnesu jest przedstawiony w dwóch wymiarach. Zatem, wszystkie parametry fizyczne i magnetyczne są jednorodne w osi wzdłużnej. Reasumując, stworzone środowisko rozszerza możliwości istniejących narzędzi symulacyjnych.

Środowisko symulacji quench'u (ŚSQ) utworzono stosując skalowalną i modułową architekturę bazującą na paradygmacie programowania obiektowego (PPO) oraz wzorcach projektowych. Uzyskana trójmodułowa aplikacja (silnik obliczeniowy Simulink oraz biblioteka, główna aplikacja MATLAB, GUI) może być łatwo rozszerzana w przyszłości. Co więcej, każda z symulacji składająca się z tysięcy elementów jest tworzona automatycznie. Dodatkowo użytkowni z poziomu intuicyjnego GUI może uruchomić symulacje ze zmiennymi parametrami oraz strukturą modelu, uruchamiać symulacje równolegle (tryb wielo-procesorowy oraz wielo-maszynowy) i przedstawiać wyniki w dogodny sposób (porównanie z wynikami eksperymentów, wykresy specjalne, itd.).

Środowisko zostało zastosowane do przewidywań zachowania magnesów i analizy osiągów podczas różnorodnych kampanii testowych obejmujących analizę ochrony przed quench'em wykonane w laboratoriach magnesów w CERN. Stworzony model z sukcesem odtworzył złożone elektro-cieplne przebiegi wywołane w skutek quench'u w nadprzewodzącym magnesie chronionym przez system ekstrakcji energii, quench heater, i/lub CLIQ. Otrzymane wyniki symulacyjne bardzo dobrze odzwierciedlały dane pomiarowe w szerokim zakresie parametrów pracy. Ze względu na wysoce automatyczny charakter ŚSQ zastosowano również do oceny osiągów optymalnych konfiguracji systemu CLIQ oraz projektowania nowych nadprzewodzących magnesów.

Ostatecznie, zaproponowana metodologia oraz architektura oparta na PPO może być łatwo dostosowana do dowolnego problemu modelowania złożonych zjawisk fizycznych przy użyciu elementów skupionych jak również przeniesiona do innego języka programowania oraz silnika symulacyjnego takich jak odpowiednio Java lub Spice.

Abstract

The thesis aims at designing a flexible, extensible, user-friendly interface to model transients occurring in superconducting magnets. Simulations are a fundamental tool for assessing the performance of a magnet and its protection system against the effects of a quench. The framework employs a lumped-element dynamic electro-thermal model of superconducting magnets along with lumped element models of quench protection methods currently used in accelerator magnets (by pass diodes, energy extraction systems, quench heaters) as well as innovative ones (CLIQ). The thermal sub-network of the superconducting magnet model is represented in two dimensions. Thus, all physical and magnetic properties are homogeneous along the longitudinal direction. To sum up, the developed framework extends modelling capabilities of existing simulation tools.

The Quench Simulation Framework (QSF) is created using scalable and modular architecture based on object-oriented programming paradigm and design patterns. Resulting three-layer application (Simulink solver and library, main MATLAB application, GUI) opens an easy way for future extensions. What is more, each simulation composed of thousands of blocks is automatically created. Additionally, the user from the intuitive GUI is able to run sets of simulations with varying parameters and model structure (parametric and structure sweep), execute simulations in parallel (in multi-core and multi-machine mode), and present results in a convenient way (comparison with experimental data, custom plots, etc.).

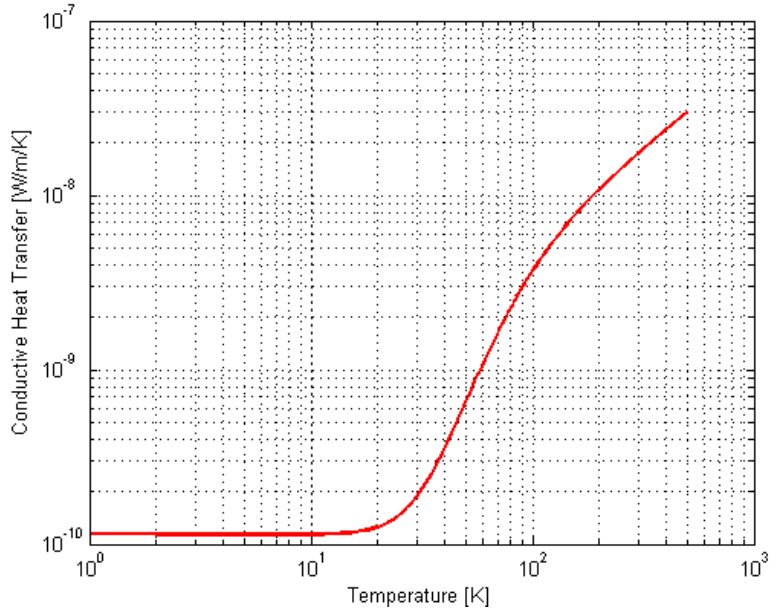
The framework has been applied to predict magnet behaviour and performance analysis during various test campaigns comprising quench protection studies performed in the CERN magnet test facility. The developed model successfully reproduced the complex electro-thermal transients following a quench in a superconducting magnet protected by an Energy Extraction system, Quench Heaters, and/or CLIQ). The obtained simulation results were in very good agreements with the measured data over a wide range of operating parameters. Due to highly automatized character, the QSF has been also applied to assess the performance of optimized configuration of the CLIQ system and new superconducting magnet design.

Finally, the proposed methodology and architecture based on OOP techniques can be adopted to any multi-physics modelling problem employing lumped-element approach as well as easily migrated to other programming languages and network solvers such as Java and Spice, respectively.

Annex

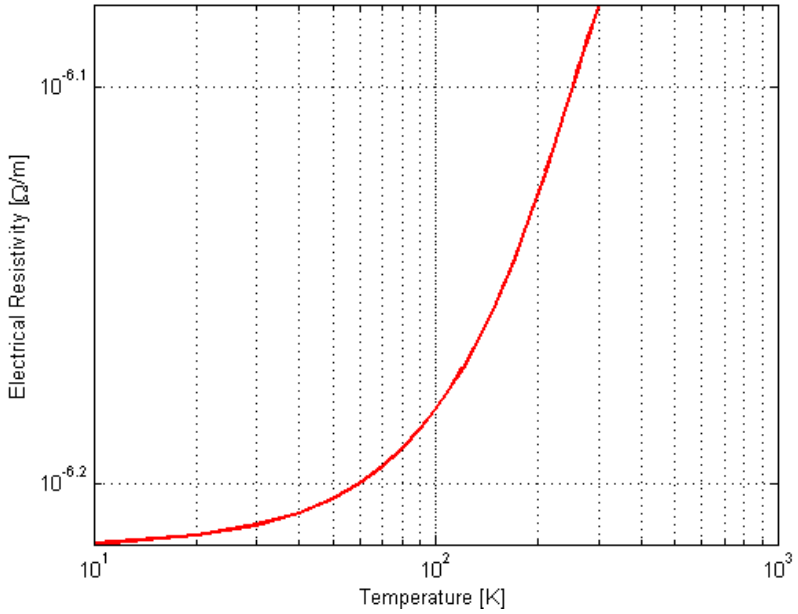
Relations adopted in the calculation of the electrical and thermal properties.

1. Resistivity of copper (without magneto-resistivity)



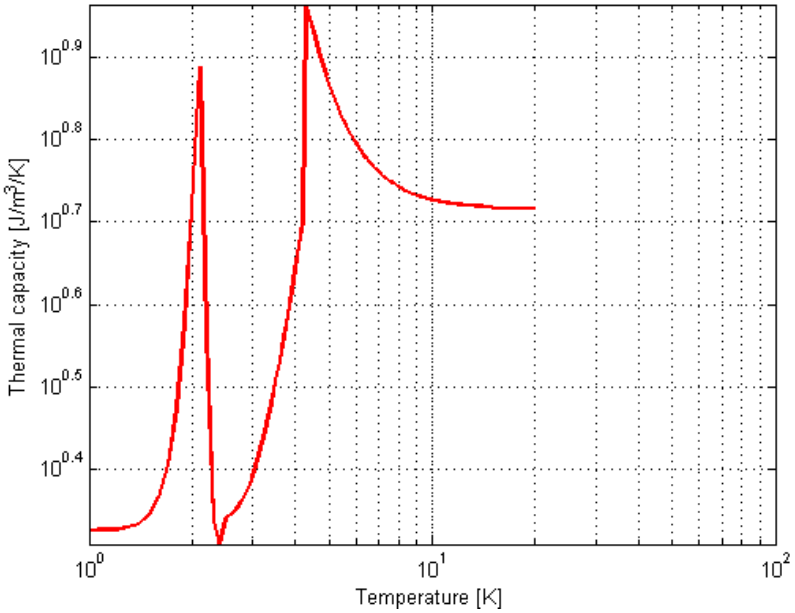
2. Resistivity of Stainless Steel

$$\rho_{SS}(T) = \left(-6.88\cdot10^{-7}T^3 + 3.94\cdot10^{-4}T^2 + 1.92\cdot10^{-2}T + 6.07\right)\cdot10^{-8} \qquad [\Omega\cdot\mathrm{m}]$$



3. Thermal capacity of He

$$cp_{He}(T)=\begin{cases}2.12+0.000678T^{12.159}, & for\ T<2.17K\\ -274.45T^3+1961.6T^2-4673.2T+3712.9, & for\ T<2.5K\\ 0.9163T^2-4.484T+7.68, & for\ T<4.3K\\ 5.2+1489.4T^{-4.06}, & for\ T<15K\\ 5.2, & for\ T\geq 15K\end{cases} \quad [J/m^3/K]$$

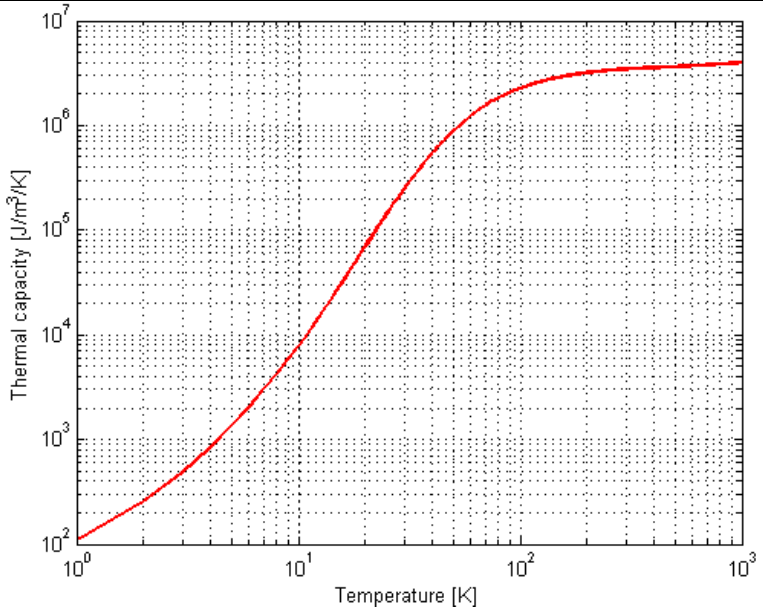


4. Thermal capacity of Cu

$$cp_{Cu}(T)=aT^4+bT^3+cT^2+dT+e, \quad [J/m^3/K]$$

where

T [K]	a	b	c	d	e
T<10	$-3.08\cdot10^{-2}$	7.23	-2.129	101.9	2.563
10<T<40	-0.3045	29.87	-455.6	3470	-8250
40<T<125	$4.19\cdot10^{-2}$	-14.02	1509	-31600	$1.784\cdot10^{5}$
125<T<300	$-8.48\cdot10^{-4}$	0.8419	-325.5	60590	$-1.29\cdot10^{6}$
300<T<500	$-4.8\cdot10^{-5}$	$9.173\cdot10^{-2}$	-64.12	20360	$1.03\cdot10^{6}$
T>500	0	$1.2\cdot10^{-5}$	-0.2149	1004	$3.18\cdot10^{6}$

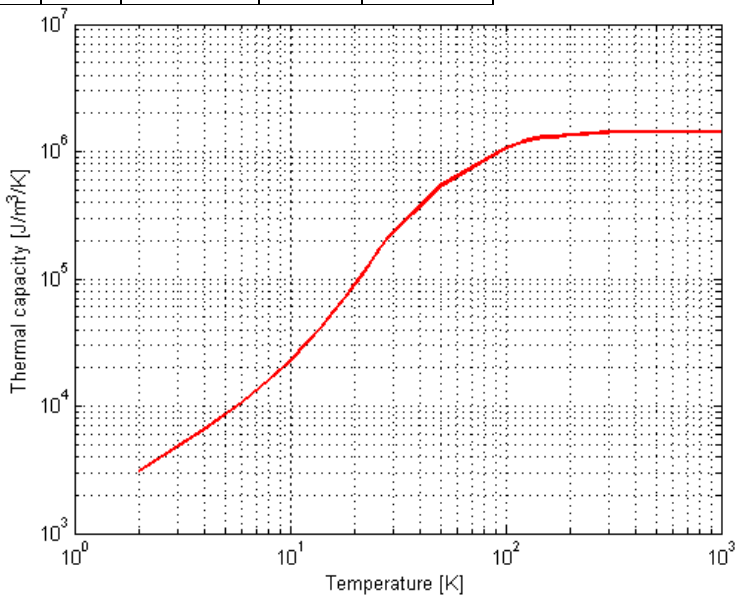


5. Thermal capacity of Nb_3Sn

$$cp_{Nb_3Sn}(T) = aT^3 + bT^2 + cT + d\,, \qquad [J/m^3/K]$$

where

T [K]	a	b	c	d
$T \leq 26.113$	7.42	0	1522	0
$26.113 < T \leq 169.416$	0	-61.635	19902	-305807
$169.416 < T \leq 300$	0	-7.4636	4411	763801
$T > 300$	0	0	0	1415377

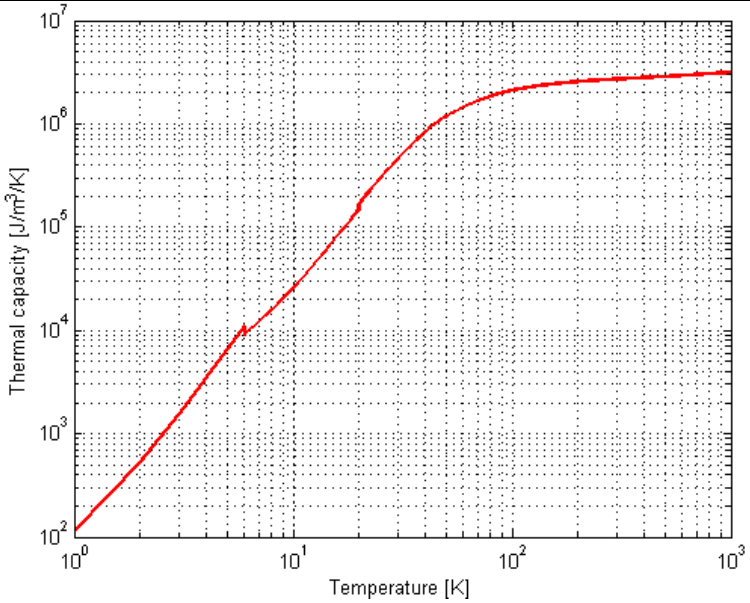


6. Thermal capacity of Nb-Ti

$$cp_{NbTi}(T) = aT^4 + bT^3 + cT^2 + dT + e\,, \qquad [J/m^3/K]$$

where

T [K]	a	b	c	d	e
$T \leq T_{cs}$	0	49.1	0	64	0
$T_{cs} < T \leq 20$	0	16.24	0	928	0
$20 < T \leq 50$	-0.2177	11.9838	553.71	-7846.1	41383
$50 < T \leq 175$	$-4.82 \cdot 10^{-3}$	2.976	-716.3	83022	$-1.53 \cdot 10^{6}$
$175 < T \leq 500$	$-6.29 \cdot 10^{-5}$	$9.296 \cdot 10^{-2}$	-51.66	13706	$1.24 \cdot 10^{6}$
$T > 500$	0	0	-0.257	955.5	$2.45 \cdot 10^{6}$

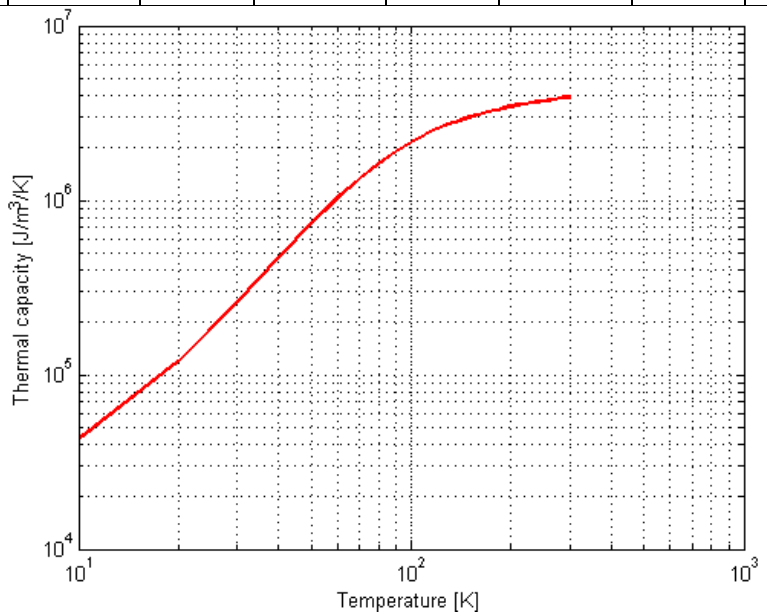


7. Thermal capacity of stainless steel

$$cp_{Nb_3Sn}(T) = \begin{cases} (aT^8 + bT^7 + cT^6 + dT^5 + eT^4 + fT^3 - gT^2 + hT + i)\cdot 8000, \; for T < 300K \\ 488.6 \cdot 8000, \; for T \geq 300K \end{cases}, \quad [\mathrm{J/m^3/K}]$$

where

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
-1.4477 $\cdot 10^{-15}$	1.7843 $\cdot 10^{-12}$	-8.9192 $\cdot 10^{-10}$	2.2916 $\cdot 10^{-7}$	-3.1054 $\cdot 10^{-5}$	1.8923 $\cdot 10^{-3}$	-1.2052 $\cdot 10^{-2}$	3.8364 $\cdot 10^{-1}$	1.1148

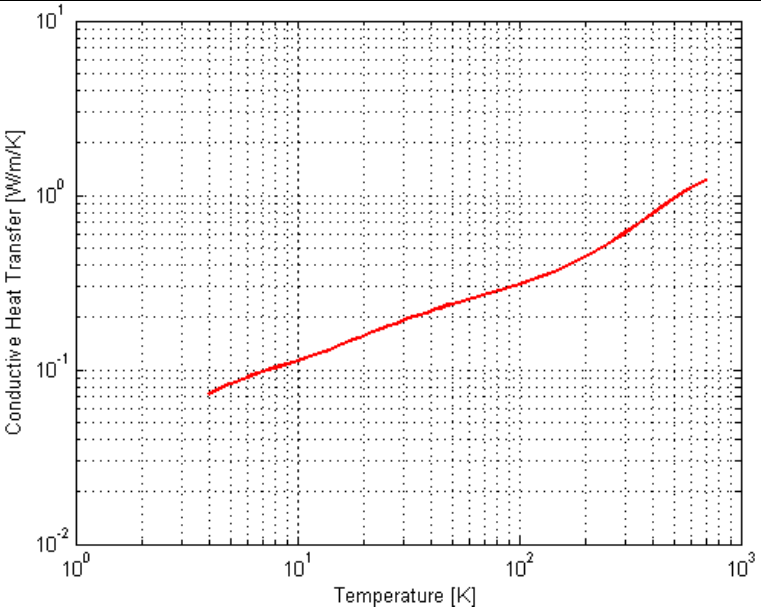


8. Conductive heat transfer G10

$$k_{G10}(T) = 10^{(a+b\log(T_1)+c\log(T_1)^2+d\log(T_1)^3+e\log(T_1)^4+f\log(T_1)^5+g\log(T_1)^6+h\log(T_1)^7)} \cdot 1.9 \cdot 10^3, \quad [\mathrm{W/m/K}]$$

where

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
-4.1236	13.788	-26.068	26.272	-14.663	4.4954	-0.6905	0.0397

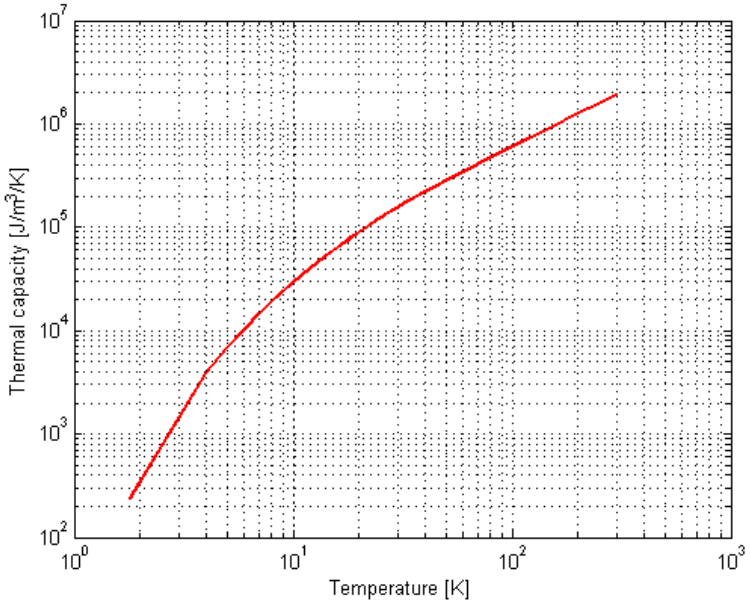


9. Thermal capacity of G10

$$k_{G10}(T)=10^{(a+b\log(T_1)+c\log(T_1)^2+d\log(T_1)^3+e\log(T_1)^4+f\log(T_1)^5+g\log(T_1)^6+h\log(T_1)^7)}\cdot 1.9\cdot 10^3, \quad [\mathrm{J/m^3/K}]$$

where

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
-2.4083	7.6006	-8.2982	7.3301	-4.2386	1.4294	-0.24396	0.015236

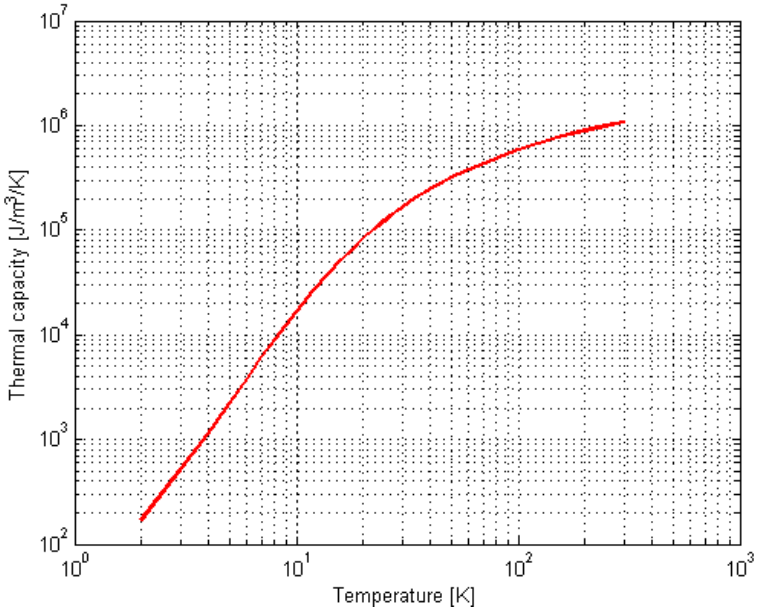


10. Thermal capacity of Kapton

$$c_{Kapton}(T)=10^{(a+b\log(T_1)+c\log(T_1)^2+d\log(T_1)^3+e\log(T_1)^4+f\log(T_1)^5+g\log(T_1)^6+h\log(T_1)^7)}\cdot 1420, \quad [\mathrm{J/m^3/K}]$$

where

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
-1.3684	0.65892	2.8719	0.42651	-3.0088	1.9558	-0.51998	0.051574

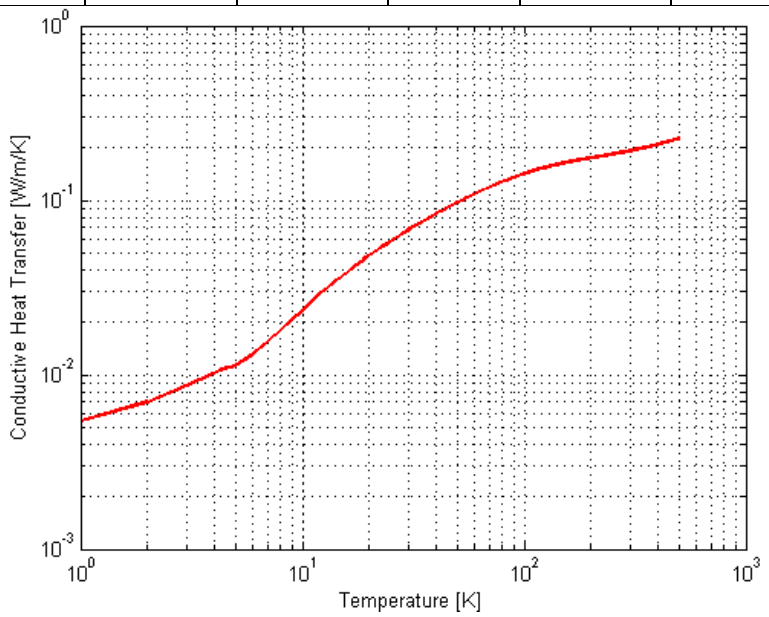


11. Conductive heat transfer of Kapton

$$k_{Kapton}(T)=\begin{cases}0.010703-0.00161(4.3-T), & for\ T\le 4.3K\\ 10^{(a+b\log(T)+c\log(T)^2+d\log(T)^3+e\log(T)^4+f\log(T)^5+g\log(T)^6+h\log(T)^7)}, & for\ T\ge 4.3K\end{cases}, \quad [\mathrm{W/m/K}]$$

where

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
5.73101	-39.5199	79.9313	-83.8572	50.9157	-17.9835	3.42413	-0.27133



Bibliography

[1] J. W. Cooper, *Java design patterns. A Tutorial*, Addison Wesley Longman Inc., 2000.
[2] R. C. Martin, *The Clean Coder*, Pearson Education Inc., 2011.
[3] W. Gajda, *Git - rozproszony system kontroli wersji*, Helion, Poland 2013.
[4] B. Hahn, D. T. Valentine, *Essential MATLAB for Engineers and Scientists*, Elsevier, Italy, 2007.
[5] A. K. Tyagi, *MATLAB and Simulink for Engineers*, Oxford University Press, 2012.
[6] J. B. Dabney, T. L. Harman, *Mastering Simulink*, Pearson Education, 2004.
[7] E. Ravaioli, Private communications.
[8] S. Russenschuck, *Field Computation for Accelerator Magnets: Analytical and Numerical Methods for Electromagnetic Design and Optimization*, Wiley-VCH, Germany, 2010.
[9] N. Schwerg, B. Auchmann, and S. Russenschuck, *Quench simulation in an integrated design environment for superconducting magnets*, IEEE Trans. on Magnetics, 44(6), pp. 934–937, June 2008.
[10] http://cern.ch/roxie, last access Aug 2014.
[11] D. Fitzpatrick, *Analog Design and Simulation using OrCAD Capture and PSpice*, Elsevier, 2007.
[12] *Simscape Guide User Guide*, MathWorks, USA, 2014.
[13] E. Ravaioli et al., *New, Coupling Loss Induced, Quench Protection System for Superconducting Accelerator Magnets*, Applied Superconductivity, IEEE Trans. on, 24(3), pp. 1-5, 2014.
[14] E. Ravaioli et al., *First Experience with the New Coupling-Loss Induced Quench System*, Cryogenics, 60, pp. 33-43, 2014.
[15] E. Ravaioli et al., *A New Hybrid Protection System for High-Field Superconducting Magnets*, Superconductor Science and Technology, 27(4), 2014.
[16] *AC-Current Induced Quench Protection System*, application has been filed with the European Patent Office on June 28, 2013 under the application number EP13174323.9.
[17] E. Ravaioli et al., *Towards an optimized Coupling-Loss-Induced Quench protection system (CLIQ) for quadrupole magnets*, Physics Procedia, to be published.
[18] E. Ravaioli et al., *Protecting a Full-Scale Nb_3Sn Magnet with CLIQ, the New Coupling-Loss Induced Quench System*, Applied Superconductivity, IEEE Trans. on, 25, submitted for publication, 2014.
[19] A. Verweij, *Electrodynamics of superconducting cables in accelerator magnets*, PhD thesis University of Twente, The Netherlands (chapter 4), 1995.
[20] B. Meyer, *Object-Oriented Software Construction (2nd Edition)*, ISE Inc., USA, 2005.
[21] S. L. Eddins, *Automated Software Testing for MATLAB*, Computing in Science and Engineering, 11(6), pp. 48-54, 2009.
[22] S. Wryczka, B. Marcinkowski, J. Maślankowski, *UML 2.x – Ćwiczenia zaawansowane*, Helion, Poland, 2012.
[23] S. T. Smith, *MATLAB Advanced GUI Development*, Dog Ear Publishing, USA, 2006.
[24] Y. Altman, *Undocumented Secrets of MATLAB-Java Programming*, CRC Press, USA, 2012.
[25] D. Halliday, R. Resnick, J. Walker, *Fundamentals of Physics, 6th Edition*, Wiley, 2001.
[26] http://home.web.cern.ch/about/accelerators, last access Aug 2014.
[27] *LHC: the guide*, CERN, 2009.
[28] E.Ravaioli et al., *Impact of the voltage transients after a fast power abort on the quench detection system in the LHC main dipole chain*, MT22, Marseille, 2011.
[29] E. Ravaioli et al., Modelling of the Voltage Waves in the LHC Main Dipole Circuits, Applied Superconductivity, IEEE Trans. on, 22(3), pp. 9002704-9002704, 2012.
[30] M. S. Lubell, *Empirical scaling formulas for critical current and critical field for commercial NbTi*, Magnetics, IEEE Trans. on, 19, pp. 754-757, 1983.
[31] L.T. Summers, et al., *A Model for the Prediction of Nb3Sn Critical Current as a Function of Field, Temperature, Strain, and Radiation Damage*, Magnetics, IEEE Trans. on, 27, pp. 2041-2044, 1991.
[32] M. N. Wilson, *Superconducting Magnets*, Oxford University Press, USA, 1990.
[33] A. Dębowski, *Automatyka – Podstawy teorii*, WNT, Poland, 2007.
[34] Y. Tokad, M. B. Reed, *Criteria and Tests for Realizability of the Inductance Matrix*, AIEE Trans., 78 pp. 924-926, 1960.
[35] D. M. Christhilf, B. J. Bacon, *Simulink-based Simulation Architecture for Evaluating Controls for Aerospace Vehicles (SAREC-ASV)*, Report, American Institute of Aeronautics and Astronautics, 2006.
[36] http://www.en.wikipedia.org/wiki/Netlist, last access Aug 2014.
[37] *LHC Design Report, The LHC Main Ring*, 1, European Organization for Nuclear Research, 2004.

[38] A.Szewczyk, et al., *Magnetyzm i nadprzewodnictwo*, PWN, Poland, 2012.
[39] F. Rodriguez-Mateos, F. Sonnemann, *Quench Heater Studies for the LHC Magnets*, PAC, Chicago, 2001.
[40] R. Schmidt et al., *Protection of the Superconducting Corrector Magnets for the LHC*, EPAC, Vien, 2000.
[41] K. Dahlerup Pettersen et al., *The protection system for the superconducting elements of the Large Hadron Collider at CERN*, PAC, New York, 1999.
[42] L. Rossi, et al., *Superconductivity leads the way to high luminosity*, CERN Courier, 53(1), pp. 28-31, 2014.
[43] http://www.eccoflow.org/index.php?option=com_content&view=article&id=51&Itemid=55, last access Aug 2014.
[44] http://cds.cern.ch/record/841539/files/, last access Aug 2014.
[45] http://lhc-machine-outreach.web.cern.ch/lhc-machine-outreach/components/cable.htm, last access Aug 2014.
[46] G. Kirby et al., *LHC IR Upgrade Nb–Ti, 120-mm Aperture Model Quadrupole Test Results at 1.8 K*, Applied Superconductivity, IEEE Trans. on, 24(3), pp. 1-5, 2014.
[47] R. E. Shafer, *Eddy Currents, Dispersion Relations, and Transient Effects in Superconducting Magnets*, Fermilab, TM-991, 1980.
[48] K. M. Smedley and R. E. Shafer, *Experimental Determination of Electrical Characteristics and Circuit Models of Superconducting Dipole Magnets*, Magnetics, IEEE Trans. on, 30(5), pp. 2708-2712, 1994.
[49] K. Dahlerup-Petersen, F. Schmidt, *Impedance Measurements and Modelling of the Ten Meter Prototype LHC Dipole Magnet*, LHC Project Note 11, 1995.
[50] F. Bourgeois, K. Dahlerup-Petersen, *Methods and results of modelling and transmission line calculations of the superconducting dipole chains of CERN's LHC collider,* PPPS, Las Vegas, 2001.
[51] R. Denz, F. Rodriguez-Mateos, *Electronic Systems for the Protection of Superconducting Elements in the LHC*, Applied Superconductivity, IEEE Trans. on, 16(2), pp. 1725-1728, 2006.
[52] R. Denz, et al., *Upgrade of the protection system for superconducting circuits in the LHC*, PAC, Vancouver, 2009.
[53] E. Ravaioli et al., *Unbalanced Impedance of the Aperture Coils of Some LHC Main Dipole Magnets*, Applied Superconductivity, IEEE Trans. on, 23(3), pp. 4000104-4000104, 2013.
[54] http://te-epc-lpc.web.cern.ch/te-epc-lpc/machines/lhc/general.stm, last access Aug 2014.
[55] http://public.web.cern.ch/public/features-archive/0909-0912.html, last access Aug 2014.
[56] http://en.wikipedia.org/wiki/Magnetic_resonance_imaging, last access Aug 2014.
[57] http://cern.ch, last access Aug 2014.

Łódź, dnia…….....2014 roku

Michał Maciejewski
(IMIĘ I NAZWISKO STUDENTA)

178089
(NR ALBUMU)

Elektrotechniki, Elektroniki, Informatyki i Automatyki
(WYDZIAŁ)

Automatyka i Robotyka
(KIERUNEK STUDIÓW)

Stacjonarne 1I-go stopnia
(RODZAJ I FORMA STUDIÓW)

OŚWIADCZENIE

Świadomy odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedkładana praca magisterska / inżynierska / licencjacka (*) na temat:

Automated, Object-Oriented Simulation Framework for Modelling of Superconducting Magnets at CERN

została napisana przeze mnie samodzielnie.
Jednocześnie oświadczam, że ww. praca:
- nie narusza praw autorskich w rozumieniu ustawy z dnia 4 lutego 1994 roku o prawie autorskim i prawach pokrewnych (j.t. Dz.U. z 2006 r. Nr 90, poz. 631, z późn. zm.) oraz dóbr osobistych chronionych prawem cywilnym, a także nie zawiera danych i informacji, które uzyskałem w sposób niedozwolony,
- nie była wcześniej podstawą żadnej innej urzędowej procedury związanej z nadawaniem dyplomów wyższej uczelni lub tytułów zawodowych.

…….
(podpis studenta)

(*) – niepotrzebne skreślić

Łódź, dnia…….....2014 roku

Michał Maciejewski
(IMIĘ I NAZWISKO STUDENTA)

178089
(NR ALBUMU)

Elektrotechniki, Elektroniki, Informatyki i Automatyki
(WYDZIAŁ)

Automatyka i Robotyka
(KIERUNEK STUDIÓW)

Stacjonarne 1I-go stopnia
(RODZAJ I FORMA STUDIÓW)

OŚWIADCZENIE

Świadomy/a odpowiedzialności karnej za składanie fałszywych zeznań oświadczam, że przedkładana na nośniku elektronicznym praca magisterska/inżynierska (*) na temat:

Automated, Object-Oriented Simulation Framework for Modelling of Superconducting Magnets at CERN

Zawiera te same treści, co oceniany przez promotora i recenzenta wydruk komputerowy.

Jednocześnie oświadczam, że jest mi znany przepis art. 233 kk określający odpowiedzialność za składanie fałszywych zeznań.

…….

(podpis studenta)

(*) – niepotrzebne skreślić