An Open-Source Finite Element Quench Simulation Tool for Superconducting Magnets

Andrea Vitrano, Mariusz Wozniak, Erik Schnaubelt, Tim Mulder, Emmanuele Ravaioli, and Arjan Verweij

Abstract—**An open-source Finite Element Quench Simulator (FiQuS) is being developed as part of the STEAM framework following CERN's open science policy (CERN, 2022). The tool is based solely on open-source software and uses Python to generate geometries and meshes with Gmsh and compute solutions with GetDP. FiQuS scripts have a modular structure to accommodate a broad range of geometries and simulation requirements, focusing mainly on superconducting accelerator magnets. At its advanced stage, the tool will be capable of 1D, 2D, and 3D geometry generation of superconducting elements such as bus bars, multi-pole, solenoid, and canted-cos-theta (CCT) magnets. It already has the capability for parametrized mesh control and subsequent model generation of 2D multi-pole and 3D CCT magnets. It will be possible to perform either electromagnetic (EM), thermal (TH), or coupled EM-TH simulations for static or transient analysis. The focus is on aspects related to the powering and quench transients, enabling parametric analyses and co-simulations to support comprehensive quench protection studies. In this contribution, we lay the foundation of FiQuS by presenting its structure and three specific capabilities that represent the basis upon which the future modules will be built, mainly: enabling cooperative simulations and the Single Source Of Truth (SSOT) practice; seamlessly integrating magnet design details around a set of input files; enabling parametric analysis and multi-objective optimization with Dakota software developed by Sandia National Laboratories. These capabilities showcase the integration of FiQuS with software developed at CERN, at other national laboratories, and within the STEAM framework.**

***Index Terms*—Finite element method, electromagnetic, superconducting magnets, quench simulations, FiQuS.**

I. Introduction

QUENCH phenomena in superconducting magnets are transient events associated with a local or distributed transition to the electrically resistive state of the coils [2]. Such events cause temperature increases, thermal and electromagnetic stresses, and voltages within the magnet, which ultimately lead to damages if not dealt with in time. Systems for quench detection and protection must be designed and used to avoid magnet failure or degradation. Software tools capable of aiding the design by simulating such phenomena are essential.

Within this context, a **Fi**nite Element (FE) **Qu**ench **S**imulator, FiQuS [3], is herein introduced along with a detailed description of its structure and main capabilities. FiQuS is essentially a free and open-source Python-based toolset that makes use of the computer-aided design (CAD), meshing and post-processing capabilities of Gmsh [4] to automatically generate parametrized geometries and meshes of superconducting elements such as bus-bars, multi-pole, solenoid, and canted-cos-theta (CCT) magnets. Additional post-processing capabilities are developed in Python, for example export of magnetic field map or self-mutual inductance matrix. It computes the numerical solutions through GetDP [5], a general FE solver for the treatment of discrete problems. Both Gmsh and GetDP belong to the open-source ONELAB bundle [6], which is coded in C++.

Simulating a magnet quench is an arduous task from both the physics and numerics standpoint. It requires multi-physics modeling at very different space and time scales [7]. More specifically, it involves both electromagnetic (EM) and thermal (TH) modeling of abrupt events in complicated geometries with differently sized components that may also require 3D treatment. Due to this level of complexity, FiQuS' wide range of capabilities, currently being developed within the STEAM team [7] at CERN, will be presented separately herein as well as in future publications. The focus of this contribution is specifically on magnetostatic simulations of 2D multi-pole magnets. FiQuS is also already able to model 3D CCT magnets in magnetostatic simulations [8]. Successive publications will showcase FiQuS capability of generating bus-bar and solenoid geometries, and simulating transient TH and coupled EM-TH problems.

II. FiQuS Structure

A. Modules and Workflow

FiQuS is organized into multiple Python scripts represented by the scheme displayed in Fig. 1. The code is constituted by a mix of built-in functions, third-party libraries, and Gmsh API commands and command-line interface (CLI) calls to GetDP. Its structure is modular. As such, each group of methods having a distinct task forms a module that is independent of the others. This feature facilitates code readability, reusability, and testing. Modularity also enables non-consecutive workflows, where only the necessary modules are involved in the process.

Manuscript received 11 November 2022; revised 27 January 2023 and 7 March 2023; accepted 16 March 2023. Date of publication 20 March 2023; date of current version 30 March 2023. *(Corresponding author: Andrea Vitrano.)*

Andrea Vitrano, Mariusz Wozniak, Tim Mulder, Emmanuele Ravaioli, and Arjan Verweij are with CERN, 1211 Meyrin, Switzerland (e-mail: andrea.vitrano@cern.ch).

Erik Schnaubelt is with CERN, 1211 Meyrin, Switzerland, and also with the Technical University of Darmstadt, 64289 Darmstadt, Germany.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TASC.2023.3259332.

Digital Object Identifier 10.1109/TASC.2023.3259332

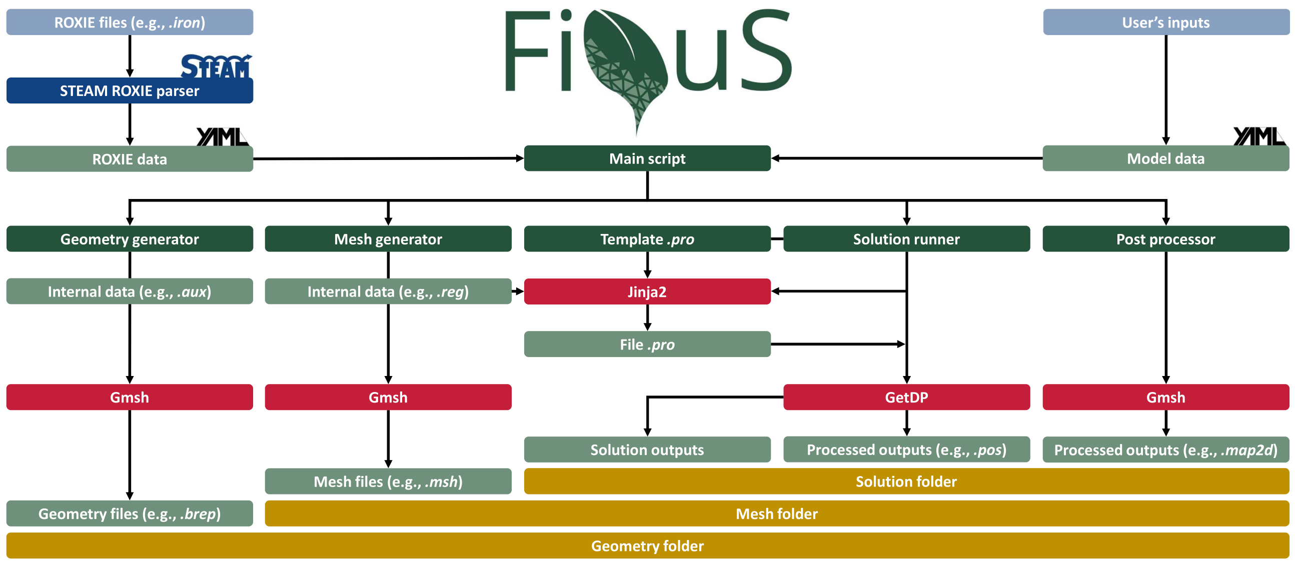

Fig. 1. FiQuS modules structure and workflow.

Depending on the type of model that needs to be built, the number of initial (top-level) input files to FiQuS varies between one and three. For instance, models of multi-pole magnets, whose 2D geometry was designed via ROXIE [9], need an additional file comprising geometrical data. A YAML file [10] is always necessary as input; it is a data-oriented, ASCII text file containing model data and user inputs describing the type of simulation, folder organization, mesh settings, and desired solution and post-processing settings.

The main script parses the input files, prepares the working folders, and selects the workflow based on the information contained in the inputs. It also performs a data check to assess the validity and feasibility of the user instructions based on both the available and provided files. The validation and parsing of the input data are performed using Pydantic dataclasses [11], which enforce type hints at run-time based on Python type annotations. The complete workflow that involves all the modules in FiQuS consists of the geometry generation, mesh generation, assembling of the GetDP-readable file with extension *.pro*, computing the solution, and post-processing. Nevertheless, modularity allows for flexibility in the choice of the type of FiQuS run. For instance, starting a simulation from the solution step is possible if a previously generated mesh is at the user's disposal. This is enabled by allowing each module to communicate with the main script only. Although this approach increases the number of interactions with respect to monolithic codes, the advantages of modular programming outweigh the drawbacks as code maintainability greatly increases over time, especially for continuously growing software such as FiQuS.

Since the various superconducting elements need to be treated differently according to their geometrical characteristics and relevant physical phenomena to be modeled, each module is constituted by multiple scripts with a common task name but different procedures. The selection of the correct procedure is automatically addressed without the user's instructions owing to the standardization of classes and methods across the various scripts.

The aforementioned *.pro* file is the typical GetDP input file, where the simulation problem is defined along with the computational domains, numerical schemes, and initial and boundary conditions [5]. Its commands follow the ONELAB syntax, which is based on C++. Getting acquainted with this type of file might represent a time-consuming challenge to inexperienced users. One of the main advantages of FiQuS consists of the automatized compilation of the *.pro* file based solely on the FiQuS text input files. This feature relieves the user from learning new software syntax and settings while focusing more on the simulated phenomena. The automatized assembling of the *.pro* file is performed through Jinja [12], a Python-compatible templating engine, and a set of *.pro* templates, customized for each solution type and distributed together with the FiQuS code base.

B. Directories and Outputs

The files generated by each module are stored in three nested levels of directories that correspond to the typical computer-aided engineering (CAE) steps for a numerical simulation: 1) Geometry folder, which includes all files related to the geometry of the model, relevant identification numbers of geometrical objects (*.aux* file), and cartesian coordinates of significant locations for post-processing purposes; 2) Mesh folder, which includes all files related to the mesh of the model, relevant identification numbers of meshed regions (*.reg* file), and physical groups of regions; 3) Solution folder, which includes all the outputs of the simulation solution, GetDP auxiliary files, and post-processed data files and figures.

FiQuS is capable of two ways of working with the (CAD) geometry files. It can either import a geometry from an externally generated file (e.g., ISO 10303 i.e. *.STEP* file), or generate internally a Boundary Representation (*.BRep*) format file with

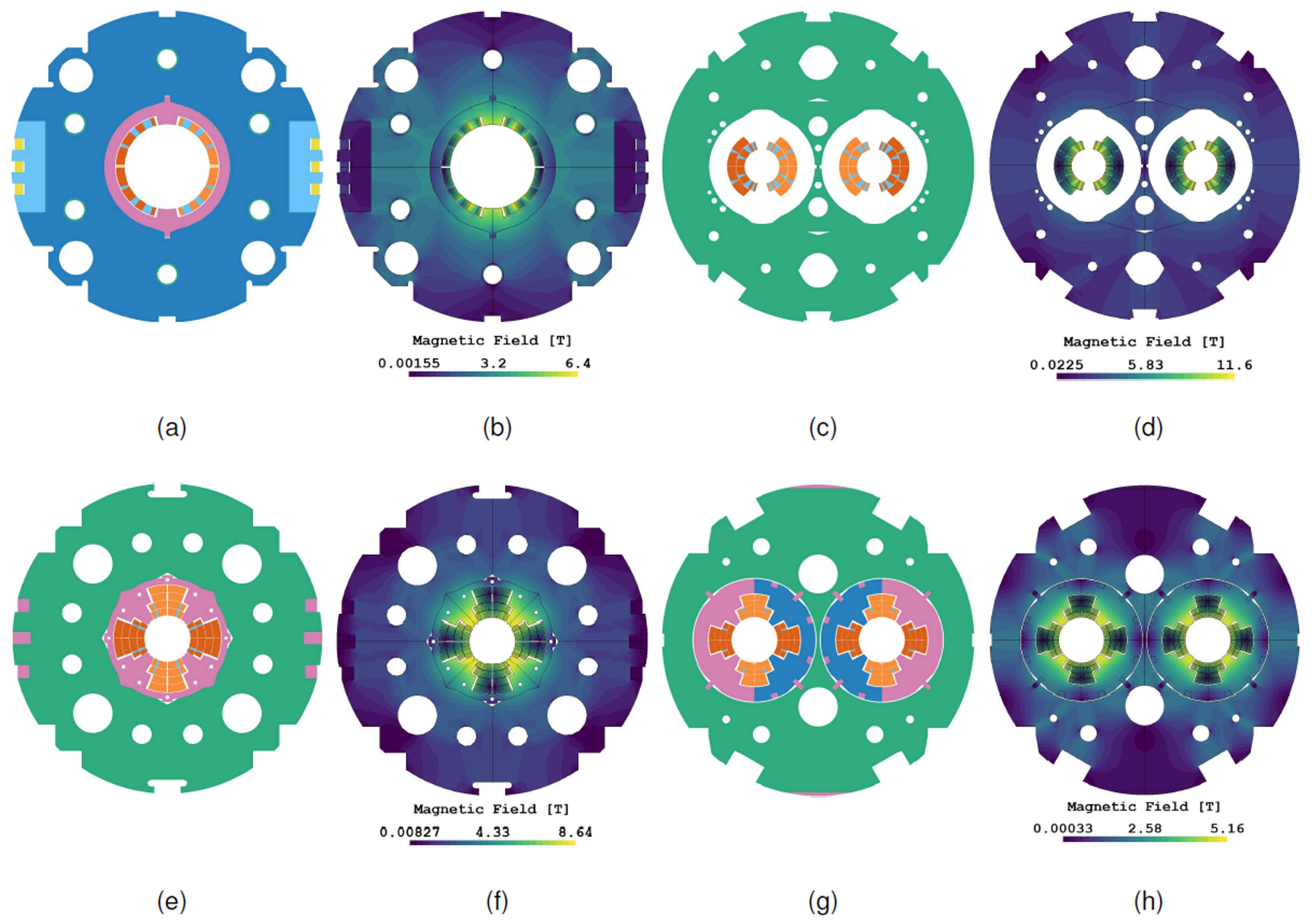

Fig. 2. 2D cross-sections of four LHC magnets with corresponding magnetic field maps showing the norm of the magnetic flux density at the nominal current: beam separation dipole MBXF [17] (a), (b); double-aperture bending dipole MBH [18] (c), (d); low-beta quadrupole MQXA [19] (e), (f); wide-aperture matching quadrupole MQY [20] (g), (h).

the Open CASCADE Technology library [13]. Neither of these file formats can store all the necessary geometry information. Therefore, additional files need to be generated to store entirely the state of the geometry (e.g., order of geometrical object tags). The same limitation applies to the model mesh *.msh* file, which does not store the mesh physical groups. The latter need to be accessible and synchronized with the *.pro* file regions numbering used in GedDP. This is particularly relevant in FiQuS: as each module is called independently from the others and objects are not passed directly between successive modules, module inputs must retain all the information that characterized the outputs of the previous step. In FiQuS, this is ensured by saving additional information from each module's process in the form of either YAML or JSON [14] dictionaries.

III. Capabilities

A. STEAM Coupling

Another advantage of FiQuS is the possibility to benefit from the functionalities offered by the STEAM framework [7]. FiQuS is part of STEAM and fully coupled with all its components. As such, FiQuS is being developed following STEAM's values: specialization, trust, consistency, repeatability, and sustainability [15]. Among others, this coupling enables consecutive and cooperative simulations with the other tools of the framework, and the Single Source Of Truth (SSOT) practice [16] through the centralization of model and material data.

In the context of theata.quench simulations of particle accelerator superconducting magnets and circuits, the scope of the STEAM framework is broad and aims to simulate diverse transients (e.g., energy extraction, quench heaters and CLIQ [Coupling-Loss Induced Quench] [21] induced quenches) for different circuit and magnet types (e.g., cos-theta, CCT, solenoids, pancake coils) at different levels of detail (e.g., circuit, magnet, conductor). Since no single tool can accommodate all this, STEAM incorporates several tools that are specialized in specific tasks and hence complement each other. Within STEAM, FiQuS plays the role of a reliable and accurate FE simulator, which is capable of supporting agile tools that rely on more approximate numerical methods. For instance, FiQuS can enhance the capabilities of STEAM-LEDET [22] by calculating magnetostatic field maps for an arbitrary current distribution

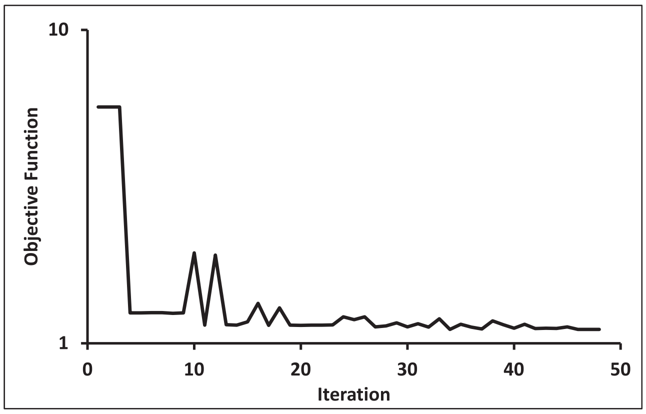

Fig. 3. Evolution of the objective function f with respect to the iteration number of Dakota's multi-objective optimization.

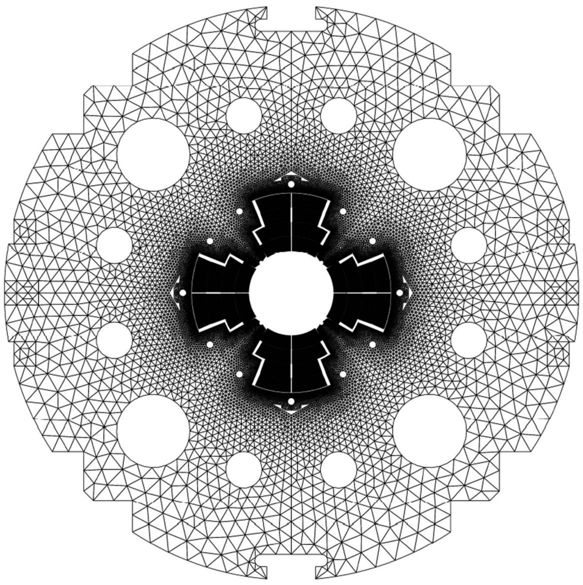

Fig. 4. Resulting mesh for fast, yet low-error magnetic simulations.

at specific time steps of a transient phenomenon. In such a co-simulation, the accuracy of the solution is improved by taking into account the non-linear effect of the iron yoke saturation more accurately through FE computation instead of simply interpolating between field maps available at specific currents.

Complex analyses often require multiple steps and tools. The higher the complexity, the harder it is to keep track of all the steps required to reproduce a specific analysis. STEAM solves this problem with AnalysisSTEAM [23], a Python-based unit that ensures traceability and reproducibility by handling information from beginning to end among user settings, data libraries, parsers, drivers, and numerical tools. AnalysisSTEAM operates based on a YAML input file that carries information about tool settings and versions, data file paths, input changes, and type of simulations. A sequence of instructive steps allows to both automatize multi-step analyses and store their executed steps simultaneously. A typical step sequence would involve setting

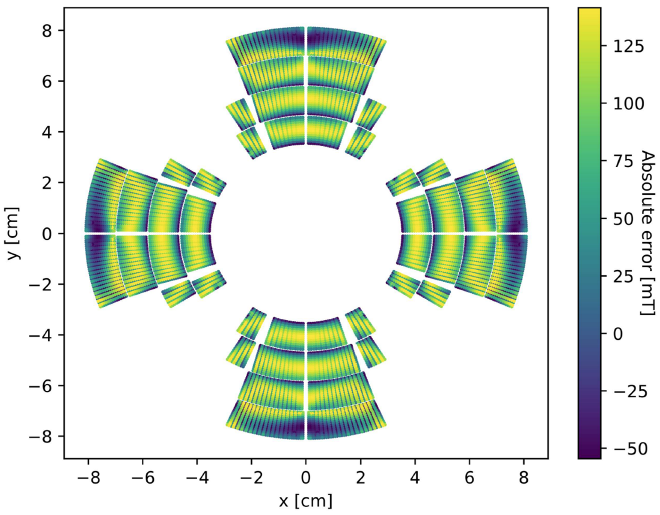

Fig. 5. Absolute error in magnetic flux density between FiQuS and ROXIE.

up the working folder, loading a reference model, changing one or more reference parameters, and running multiple simulations associated with each change. For instance, FiQuS can be instructed through AnalysisSTEAM to build a magnet geometry and mesh, and then compute multiple magnetostatic solutions at different initial currents by changing the inputs of a reference file stored in the library.

Also the database plays an important role in the consistency and reliability of results. STEAM features a library of data on magnet design and operational parameters, benchmark measurements, and material properties. The methods to fetch this information have been recently standardized across all STEAM tools and adapted to a common source in order to guarantee data normalization. This structure establishes the so-called SSOT practice within STEAM, reducing data redundancy, eliminating inconsistencies, simplifying modifications and revisions, ultimately enhancing productivity. The data is stored as YAML text files as dictionaries. As such, the library benefits from the advantages of text files: versatility, longevity, and independence from software changes.

B. *ROXIE Parser*

The vast majority of LHC and HL-LHC magnets were designed using the electromagnetic simulation and design optimization software ROXIE [9]. As such, almost all superconducting magnets at CERN are associated with a set of ROXIE files that define the geometrical characteristics, design features, and materials of their components. Shortening the path between numerical simulations of a magnet model and its design data from ROXIE is essential to enhance productivity and efficiency. With this aim, a ROXIE parser has been developed within STEAM in order to parse all types of ROXIE files. The new version of the ROXIE parser is capable of turning information contained in *.data*, *.cadata*, *.iron*, *.mod*, and *.bhdata* files into dictionaries that can also be exported as YAML text files (see Fig. 1).

These dictionaries include processed data about the conductor positions, cable design details (e.g., insulation specifics), iron yoke and collar geometry, and B-H curves. FiQuS is integrated with the ROXIE parser and, based on its output, can generate the geometry of any magnet characterized by ROXIE files. As an example, Fig. 2 shows a sample of 2D geometries of four LHC magnets generated by FiQuS using the ROXIE parser dictionaries. The figure also shows the magnetic flux density map from magnetostatic solutions computed by FiQuS for each magnet at the corresponding nominal current.

C. Parametric Analyses

As explained in the introduction, numerical simulations are essential to comprehend the conditions of a magnet during quench. However, such phenomena can arise under several circumstances depending on different combinations of operational conditions, design parameters, and material specifications. Covering all possible cases is an arduous task even with the usage of tools that facilitate multi-step analyses such as AnalysisSTEAM (see Section III-A). A systematic and reliable set of procedures is needed to explore probabilistic forecasts and statistical scenarios based on design constraints and tolerances.

For this purpose, STEAM has recently adopted the open-source parametric analysis software Dakota developed by Sandia National Laboratories [24]. Dakota offers a wide range of algorithms for various analysis types such as parametric sweep, uncertainty quantification, and multi-objective optimization. The full parametrization of all simulation inputs of FiQuS allows parallelizing several runs using Dakota. The coupling with Dakota is quite straightforward as Dakota treats the linked tool like a black box that accepts inputs and returns outputs to be processed or stored. Nevertheless, this coupling has been tailored to the needs and structure of STEAM, which has been interfaced with Dakota through a Python package provided by the Dakota developers. This extends previous work where Dakota has been coupled to LEDET [25]. Dakota is equipped with a text file interpreter, which can be advantageously used for the coupling with FiQuS' YAML input files. Although Dakota offers a standardized syntax to modify iteratively the input parameters of text files, an ad-hoc Python script has been developed within STEAM to change specifically the inputs of multi-step AnalysisSTEAM YAML files. Moreover, Dakota's syntax is not applicable to nested dictionaries, as the communication between Dakota's input *.in* files and its interpretation of nested keys would be unfeasible. The Python script thus helps circumvent this obstacle, while allowing the coupling with more complex YAML structures. A FiQuS user is not required to learn how to set up *.in* files, as STEAM includes building methods that prepare them based on simplified inputs and pre-compiled Jinja templates.

IV. Demonstrative Simulation

As a demonstration of some of the capabilities described above, a computational analysis is reported for the LHC quadrupole magnet MQXA (see Fig. 2(e)). Such analysis involves the coupling with the STEAM framework, the ROXIE parser, and Dakota. FiQuS builds the geometry based on the data dictionary constructed by the ROXIE parser starting from ROXIE files stored in the STEAM models library. Then, a multi-step AnalysisSTEAM input file is run iteratively based on Dakota's instructions on FiQuS input parameters. For the solution of the electromagnetic problem, a 2D magnetostatic simulation with current density along the invariant direction was carried out using a formulation based on the magnetic vector potential [26] with a cable current of 7149 A.

The goal of the analysis is to find an optimal mesh of the magnet 2D cross-section through a multi-objective bound-constrained optimization that aims to minimize an objective function f. This function f is a weighted sum of two simulation outputs and can be written as $f = (1 - w)b + w\Delta t$, where b is the 2-norm of the absolute errors in magnetic flux density between FiQuS and ROXIE at the strands locations, Δt is the computation time required to obtain the solution, and w is a weighting factor corresponding to 10^{-3}. The optimization was set up with two constrained design variables, that is, the minimum element size of both coil and iron yoke, which were varied within different feasibility regions by a Dakota multi-start algorithm. The chosen algorithm is based on the Fletcher-Reeves conjugate gradient optimization method [27]. Fig. 3 shows the evolution of the objective function throughout the optimization, whose final result yields the mesh displayed in Fig. 4. The solution time associated with this mesh is about 1 minute.

Fig. 5 shows the absolute error in magnetic flux density between FiQuS and ROXIE obtained using the mesh of Fig. 4. The purpose of this validation is to obtain within reasonable computational time a magnetic flux density with an error that is acceptable for quench simulations.

V. Summary

The Finite Element Quench Simulator, FiQuS and its integration with ROXIE input files and STEAM framework was introduced. FiQuS is currently being developed as part of the STEAM framework and will be capable, in its advanced stage, of 1D, 2D and 3D modeling of various superconducting devices. This work presented the fundamental structure of FiQuS together with its main capabilities as well as a demonstration of magnetostatic simulations of 2D multi-pole magnet models, which represent a necessary step in the path towards advanced quench simulations. The advantages of FiQuS include: 1) free-to-use; 2) open-source; 3) fast C++ solver; 4) fully parametrized inputs; 5) simple text-based input files; 6) integration with ROXIE files via parsed files; 7) built-in templating of third-party input files (e.g., GetDP's *.pro*); 8) integration with the STEAM framework. Thanks to the latter, FiQuS is: i) capable of consecutive and cooperative simulations; ii) integrated with a library of magnet design parameters and material specifications; iii) integrated with Dakota for parametric analyses with templated and automatically generated Dakota input file (*.in*). For advanced users and developers who would like to extend FiQuS capabilities, other advantages can be highlighted: a) modern scripting Python code; b) easy-to-extend modular structure.

References

[1] "CERN open science policy," CERN, Geneva, Switzerland, Tech. Rep. CERN-OPEN-2022-013, 2022. [Online]. Available: https://cds.cern.ch/record/2835057

[2] M. Wilson, *Superconducting Magnets* (Series Monographs on Cryogenics). Oxford, U.K.: Clarendon Press, 1987.

[3] FiQuS 2023.1.0., "Finite element quench simulation tool," 2022. [Online]. Available: cern.ch/fiqus

[4] C. Geuzaine and J.-F. Remacle, "Gmsh: A 3-D finite element mesh generator with built-in pre-and post-processing facilities," *Int. J. Numer. Methods Eng.*, vol. 79, no. 11, pp. 1309–1331, 2009.

[5] P. Dular, C. Geuzaine, F. Henrotte, and W. Legros, "A general environment for the treatment of discrete problems and its application to the finite element method," *IEEE Trans. Magn.*, vol. 34, no. 5, pp. 3395–3398, Sep. 1998.

[6] C. Geuzaine, F. Henrotte, J.-F. Remacle, E. Marchandise, and R. Sabariego, "ONELAB: Open numerical engineering laboratory," in *Proc. 11e Colloque Nat. en Calcul des Structures*, 2013.

[7] STEAM, "Simulation of transient effects in accelerator magnets," 2022. [Online]. Available: https://espace.cern.ch/steam

[8] M. Wozniak, E. Ravaioli, and A. Verweij, "Co-simulation of quench behaviour of HL-LHC dipole canted cos-theta orbit corrector prototypes," *IEEE Trans. Appl. Supercond.*, vol. 33, no. 5, pp. 1–5, 2023.

[9] S. Russenschuck, "ROXIE: Routine for the optimization of magnet X-sections, inverse field calculation and coil end design," CERN, Geneva, Switzerland, Tech. Rep. CERN-99-01, 1999.

[10] O. Ben-Kiki, C. Evans, and B. Ingerson, "YAML ain't markup language (YAML) version 1.1," Working Draft 2008-5-11, 2009.

[11] "Pydantic," 2022. [Online]. Available: https://pydantic-docs.helpmanual.io/

[12] Jinja2, "A fast, expressive, extensible templating engine," 2022. [Online]. Available: https://jinja.palletsprojects.com/en/3.1.x/

[13] "Open CASCADE technology," 2022. [Online]. Available: https://dev.opencascade.org/

[14] D. Crockford, "The application/JSON media type for Javascript object notation (JSON)," RFC 4627, 2006.

[15] C. C. Venters et al., "Software sustainability: Research and practice from a software architecture viewpoint," *J. Syst. Softw.*, vol. 138, pp. 174–188, 2018.

[16] C. Pang and D. Szafron, "Single source of truth (SSOT) for service oriented architecture (SOA)," in *Proc. Int. Conf. Serv.- Oriented Comput.*, Paris, France, 2014, pp. 575–589.

[17] M. Sugano et al., "Development of 2-m model magnet of the beam separation dipole with new iron cross section for the high-luminosity LHC upgrade," *IEEE Trans. Appl. Supercond.*, vol. 29, no. 5, Aug. 2019, Art. no. 4003607.

[18] L. Fiscarelli, M. Giovannozzi, P. D. Hermes, S. I. Bermudez, S. Russenschuck, and F. Savary, "Field quality of MBH 11-T dipoles for HL-LHC and impact on beam dynamic aperture," *IEEE Trans. Appl. Supercond.*, vol. 28, no. 3, Apr. 2018, Art. no. 4004005.

[19] Y. Ajima et al., "The MQXA quadrupoles for the LHC low-beta insertions," *Nucl. Instrum. Methods Phys. Res. Sect. A: Accel., Spectrometers, Detect. Assoc. Equip.*, vol. 550, no. 3, pp. 499–513, 2005.

[20] R. Ostojic, "The LHC insertion magnets," *IEEE Trans. Appl. Supercond.*, vol. 12, no. 1, pp. 196–201, Mar. 2002.

[21] E. Ravaioli, "CLIQ. A new quench protection technology for superconducting magnets," Ph.D. dissertation, University of Twente, Enschede, The Netherlands, Jun. 2015. [Online]. Available: https://cds.cern.ch/record/2031159

[22] E. Ravaioli, B. Auchmann, M. Maciejewski, H. T. Kate, and A. Verweij, "Lumped-element dynamic electro-thermal model of a superconducting magnet," *Cryogenics*, vol. 80, pp. 346–356, 2016.

[23] L. Bender, "Simulation of quench behaviour of the recombination dipole magnet for the LHC high luminosity upgrade," Master's Thesis, Hochschule Karlsruhe, Karlsruhe, Germany, 2022. [Online]. Available: https://cds.cern.ch/record/2838807

[24] B. Adams et al., "Dakota, A multilevel parallel object-oriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 6.13 user's manual," Sandia National Lab (SNL-NM), Albuquerque, NM, USA, Tech. Rep. SAND-2020-4987, 2020.

[25] M. Wozniak et al., "Quench protection of the HL-LHC hollow electron lens superconducting solenoid magnets," *IEEE Trans. Appl. Supercond.*, vol. 32, no. 6, Sep. 2022, Art. no. 4701205.

[26] P. Lombard and G. Meunier, "A general method for electric and magnetic coupled problem in 2D and magnetodynamic domain," *IEEE Trans. Magn.*, vol. 28, no. 2, pp. 1291–1294, Mar. 1992.

[27] G. N. Vanderplaats, "CONMIN: A Fortran program for constrained function minimization: User's manual," NASA, Washington, DC, USA, Tech. Memorandum NASA-TM-X-62282, 1973.