This document is the accepted manuscript version of the following article:
Maciejewski, M., Auchmann, B., Martins Araujo, D., Vallone, G., Leuthold, J., & Smajic, J. (2023). Model-based system engineering framework for superconducting accelerator magnet design. IEEE Transactions on Applied Superconductivity (5 pp.). https://doi.org/10.1109/TASC.2023.3249647

Model-Based System Engineering Framework for Superconducting Accelerator Magnet Design

Michał Maciejewski, Bernhard Auchmann, Douglas Martins Araujo, Giorgio Vallone, Juerg Leuthold, *Fellow, IEEE,* and Jasmin Smajic, *Senior Member, IEEE*

***Abstract*—The design specification of future high-field accelerator magnets require innovation in and integration of multiple disciplines. Numerical models underpin each step of the design. In this paper, we present a methodology for collaborative modeling based on relevant concepts of model-based systems engineering. The methodology is composed of three pillars: encapsulated computing environments with service query interface, model notebooks with auto-generated model views, and model query interface with results caching. The methodology aims at a decentralized approach to multi-model and multi-scale collaboration that maintains the cornerstones of traceability and reproducibility. It is demonstrated with a multi-model coil design optimization of a high-field superconducting magnet.**

***Index Terms*—Superconducting magnets, Magnet design and analysis techniques, Systems engineering.**

I. Introduction

The lifetime of superconducting accelerator magnet projects may span over several decades from design optimization, through fabrication and testing of prototypes to series production, commissioning, and operation. The overall process is typically carried out by several cross-continent teams of experts who use models to support the design. It is, therefore, of high importance to keep track of models used for design and analysis and of their evolution over time. A motivating example is the design of the LHC successor, which involves the exploration of various superconducting magnet geometries and circuit topologies, e.g., [2], [3], [4].

The main contribution of this work is a methodology for collaborative modeling to tackle superconducting-magnet design challenges. Firstly, we incorporate Model-Based Systems Engineering (MBSE) concepts [5] to establish traceable design workflows. Secondly, we present a set of optimization algorithms suited for computationally efficient multi-physics optimization of field problems. The model-based design workflow is illustrated with an integrated magneto-thermo-mechanical geometry optimization of a coil for a high-field superconducting magnet.

The paper is organized as follows. Section II introduces the MBSE methodology and outlines three core pillars. In Section III we discuss the versioning of design variants and collaboration across multiple laboratories and institutes. Section IV presents a multi-model, multi-objective optimization problem implemented with the MBSE framework. Finally, section V concludes the paper.

M. Maciejewski, J. Leuthold, and J. Smajic are with Institute of Electromagnetic Fields, ETH Zürich, Switzerland. Corresponding author e-mail: mmaciejewski@ethz.ch.

B.Auchmann is with TE-MSC, CERN, Geneva, Switzerland, and with Paul Scherrer Institute, Villigen PSI, Switzerland.

D. Martins Araujo is with Paul Scherrer Institute, Villigen PSI, Switzerland.

G. Vallone is with Lawrence Berkeley National Laboratory, Berkeley, USA.

This work was performed under the auspices and with support from the Swiss Accelerator Research and Technology (CHART) program (www.chart.ch).

II. MBSE Framework

From Systems Engineering (SE) perspective a system design is accompanied with a set of documents such as conceptual design reports (CDRs), technical design reports (TDRs), etc. MBSE complements SE and introduces a paradigm shift from a document-centric protocol for sharing of information to a model-based sharing of information. In this setup, models are treated as repositories of data queried whenever a result is needed. Following this approach a system design is represented with human-readable view of a model, i.e., an auto-generated, interactive report. Previously, MBSE has been employed for life cycle management of large-scale systems, e.g., in automotive and aerospace engineering [6], [7].

A. Computing Environment Encapsulation

In order to ensure reproducibility of modelling results we introduce operating system (OS)-level encapsulation for numerical solvers or other services. We rely on Docker containers to provide a unified and encapsulated modelling environment across popular operating systems [10]. Unlike a virtual machine, a Docker container does not require the installation of an operating system on a host OS. Instead, the Docker engine is handling information exchange between a container and a host OS. As a result, containers are lightweight and portable. A Docker image stores all dependencies needed to execute a simulation. Docker images are versioned and stored in an image registry.

Since containers do not have external dependencies and are versioned, they ensure the fullest possible repeatability of numerical input/output behavior. The solver query communication protocol, e.g., HTTP, is added to a container by a generic multi-stage build operation, without additional work for the solver developer. The solver query interface is used through a simple Python API which provides four functions:

```
process_id = service.init(name)
service.upload(process_id, input_file(s))
service.run(process_id)
service.download(process_id, artefact(s))
```

The `name` variable is replaced by an IP address of the container. By means of the containerization of services and the HTTP-based micro-service architecture, it is easy for users to install and use a multitude of services in their workflows without loosing the cornerstones of traceability and repeatability. A multi-container environment is composed by running several instances together with built-in network and file system sharing. We are aware of no other implementation that achieves a similar result, other than a monolithic solution such as that by Siemens [11].

Containers exist for ANSYS as a command-line interface (provided by the company, [12]), ROXIE (including GUI), Opera 3D, as well as for the model cache and for work with notebook-based models; cf. the remainder of this Section. In Section IV we use a custom Docker container for multi-model, multi-objective optimization.

B. Model Notebooks as Repositories of Data with Model View

In order to provide a uniform modelling interface, we encapsulate numerical models in notebooks. A notebook combines high-level function calls with results of their execution along with documentation in the form of text, input files, plots, equations, tables, and output files. A model is implemented as a notebook, interacting with a solver through the solver query interface. A notebook may also incorporate measurement database queries for model verification and validation.

Furthermore, a notebook can be exported and persisted as a report. A report contains information about its execution date, used versions of software, input files as well as input files, interactive plots and other relevant results. We chose python Jupyter notebooks due to abundance of scientific libraries and relatively large community of users, [8], [9].

C. Model Query Mechanism

By implementing the MBSE methodology, model notebooks become repositories of data, and therefore, need to provide a query mechanism. Following this approach, models can query both figures of merit and artefact files from one another; see Figure 1. In a naïve approach this may lead to a redundant execution of models. To this end, we introduce a model registry composed of model configuration and cache database.

The configuration file contains information about model location, its inputs, along with dependencies to other models, i.e., each physics model may depend on a geometry model. The model dependency is conveniently represented as a Design Structure Matrix [5]

$$\begin{bmatrix} & M_1 & M_2 & M_3 \\ M_1 & & \mathrm{x} & \mathrm{x} \\ M_2 & \mathrm{x} & & \\ M_3 & & \mathrm{x} & \end{bmatrix}.$$

A symmetrical entry indicates an infinite loop in a model query. In the example above, model M_1 calls M_2 and vice-versa. In this case, the two models may be: a) strongly coupled and require a third (e.g., fixed-point iteration, waveform relaxation) model that iteratively executes both queries until convergence is reached (the third model then provides both consistent results as artefacts); or b), one model can be divided into two or more models which can then be executed sequentially[1]. In both cases, the symmetric entries and the associated loop disappear. A valid execution tree forms a directed acyclic graph.

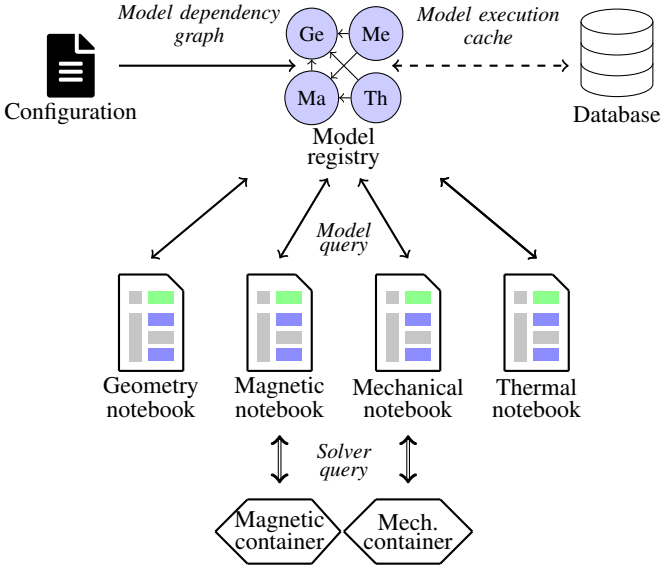

Fig. 1. Automated multi-physics design optimization workflow. Bidirectional arrows represent the flow of model queries. Double lines denote bidirectional solver queries.

When a model is queried multiple times, the model would be evaluated multiple times with the same inputs, thus causing unnecessary computational cost. A dedicated database keeps track of model execution and serves as storage for a cache mechanism. To avoid redundant model execution, the query mechanism first checks the dependency tree and a cache database. If a matching model exists in the cache, the corresponding outputs are returned without executing the model.

To this end, a cache database stores model snapshots. A model snapshot consists of a model hash, model inputs, figures of merit, artefacts, and hashes of all dependent models. A model hash is computed from model content, input parameters, and input file contents. Thus, a model hash change is detected when either a model is changed or any of its dependencies changed.

The caching database not only saves time for superfluous model executions, but also keeps track of each execution of the model. As such, the cache database enables advanced analytics on model results, e.g., after an optimization, all evaluations, along with their inputs, can be retrieved, and their figures of merit and artefacts displayed and analyzed.

III. Versioning of Design Variants

Models and inputs are integrated with a code repository enabling versioning and branching of design variants, e.g., reference, as-built, etc.. Relevant design decisions can be labelled with a tag to establish what is the current valid baseline.

[1]Example: if a margin calculation requires input from a mechanical model, and the mechanical model requires Lorentz-force input from the electromagnetic model, then the margin calculation must be represented as an independent model, separate from the electromagnetic calculation, to resolve the loop.

In addition, for each input version, a model can be queried for artefacts and report; this in turn executes a model on a dedicated machine. A model can also be executed at regular intervals to check for potential regressions, due to e.g., software version changes, OS updates.

The model registry and automated model execution are implemented as a distributed GitLab repository enabling collaborative work of the teams of experts. This concept naturally extends to a multi-project setup for multiple systems designed by separate teams. In this setup, each team maintains a separate repository which versions relevant model variants. The model query mechanism supports a distributed mode through the GitLab API. The distributed model query, accessing external model repositories, provides an elegant way to query models generated by other groups and institutes, and include their results in a design report, all-the-while retaining full traceability and repeatability.

IV. Systems Engineering Trade Study

The Docker containers with solvers, model notebooks, the model query mechanism, and a versioning platform are the cornerstones of an open and light-weight MBSE methodology. The presented methodology allows to perform trade studies such as parameter space exploration, verification and validation, and trade studies to name a few. Those studies aim at improving the figures of merit and finding a best trade value for a model. In case of superconducting magnets, a coil geometry optimization is a prime example of a trade study carried out at an early design stage.

A. Coil Geometry Optimization

The main objective of a superconducting magnet design is to reach the target magnetic field, minimize its unwanted imperfections (expressed as unwanted magnetic field components), and obtain a desired operational margin (expressed in terms of a position on a superconductor load line). Furthermore, the use of a brittle superconductor such as Nb_3Sn or Bi2212, while offering higher level of critical current at elevated fields, introduces a limit on the peak stress in the coil to avoid irreversible performance degradation. Lastly, a magnet has to be protected from excessive thermal load and voltages to ground in an event of a quench either in a standalone setup or in a larger circuit.

The geometry of a cos-θ coil is typically optimized by adjusting the number of conductors and the angular position of each block as depicted in Figure 2.

We perform the coil geometry optimization with bio-inspired algorithms which are suited for combinatorial optimization problems with discrete design variables. Furthermore, we chose the stochastic optimizers due to their robustness in terms of the computational error associated with chosen field simulators [13]. In particular, the micro-genetic algorithm has been applied for optimization of field problems [13].

We note that the bio-inspired algorithms are typically suited for efficiently scanning a large parameter space. Thus, we also implemented Rosenbrock method [14] for fine tuning of obtained design in the vicinity of the genetic optimization result. The Rosenbrock method does not rely on gradient calculation and, therefore, matches the problem setup.

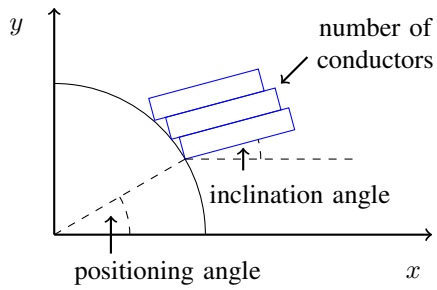

Fig. 2. 2D cross-section of a single block of an innermost layer in a cos-θ magnet with indication of design variables.

V. High-Field Magnet Optimization

As a proof of concept, the model-based design workflow is illustrated with an optimization of a high-field, four-layer cos-theta dipole magnet. Due to the magnet's symmetry, only a single quadrant is considered. In addition to optimizing each block, the total number of blocks per layer is also adjusted by the algorithm. Note that the optimization study serves only as a demonstration of the framework; for an actual application to multi-model coil geometry optimization with the presented MBSE methodology please refer to [17]. An up-to-date user documentation is available at [18].

Due to a wide range of design variables aimed at covering a wide parameter trade space, a randomly selected set of parameters may lead to an inconsistent geometry and/or divergent numerical results. Thus, the initialization is carried out until a desired number of consistent individuals is reached. In case any model failure occurs during the optimization loop, it is assigned a penalty trade value.

The schematic of the multi-objective geometry optimization is shown in Figure 3. The value model computes a trade value for a particular set of parameters prescribed by the optimization algorithm. The optimization notebook iteratively queries the value model for the trade value. The trade value is a, generally non-linear, combination of relevant figures of merit that are queried from the respective physics models; cf. Figure 3. Furthermore, the design constraints are added in a similar manner to the trade value.

The optimization algorithm updates a single parametric geometry definition. Afterwards, the geometry definition is queried in each model notebook and supplied to model generators to build updated model inputs. In case a geometry definition is inconsistent, e.g., blocks fall outside of the first quadrant, wedge tip is too sharp, the input is discarded and the trade value assigned a penalty value. For valid geometries, the trade value is computed from the figures of merit as follows

$$\text{tv} = 10^{-1}|b_3| + 10^{-1}|b_5| + |m| + 10^{-9}\sigma_{\text{vm}} + 10^{-3}T_{\text{hs}},$$

where b_3 and b_5 are sextupole and decapole field components queried from the magnetic model, m is the superconductor margin queried from the mechanical model, σ_{vm} is the Von

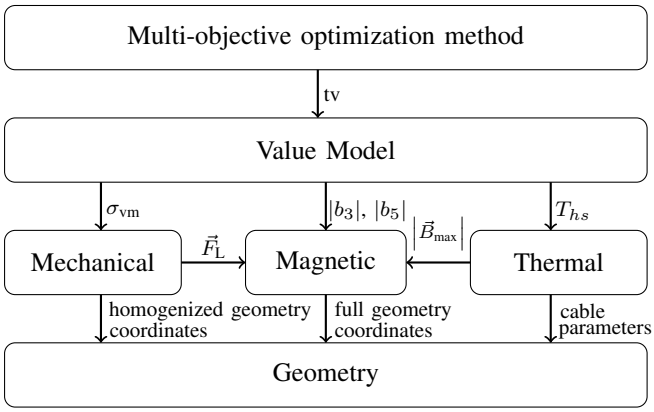

Fig. 3. Automated multi-physics design optimization workflow. Arrows represent the flow of model queries; they start at the model that request a figure of merit or artefact and end at the model that provides the requested data.

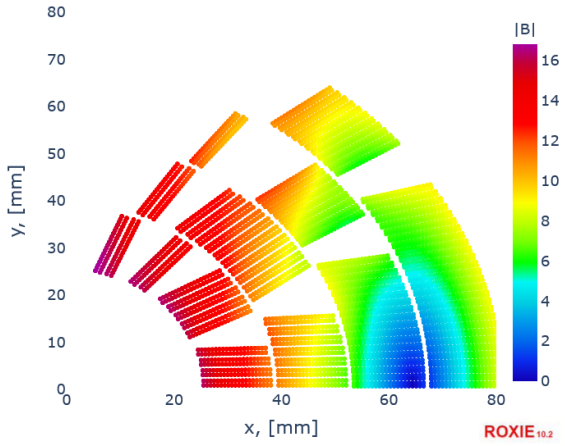

Fig. 4. Magnetic flux density distribution in superconducting strands of a magnet quadrant.

Mises stress, and T_{hs} is the hot-spot temperature queried from the thermal model.

A static electromagnetic simulation with ROXIE [15] is performed to obtain the peak magnetic field, B_{max}, Lorentz force, $\vec{F}_L$, field harmonics, and the superconductor margin. The model is constructed from full magnet geometry, i.e., coordinates of each conductor are considered. To reduce computation time, the model does not include the non-linear iron yoke contribution. The magnetic field is computed by means of Biot-Savart method. Figure 4 shows the distribution of the magnetic flux density for the best individual.

A static mechanical ANSYS APDL [16] model is constructed from homogenized geometry, i.e., conductors in a block are merged into one. The model is loaded with Lorentz force, that is queried from the electromagnetic model, and the peak stress in the coil is evaluated. We consider linear material properties and relevant contact interfaces between coil layers. The obtained von Mises stress distribution of the best individual is depicted in Figure 5.

The third model is an adiabatic estimation of the maximum temperature in the coil for the conductor with the peak magnetic field, $|\vec{B}_{max}|$. This approach leads to a worst-case scenario by assuming absence of the heat transfer and cooling mechanisms. The magnet is assumed to be protected by an energy-extraction resistor.

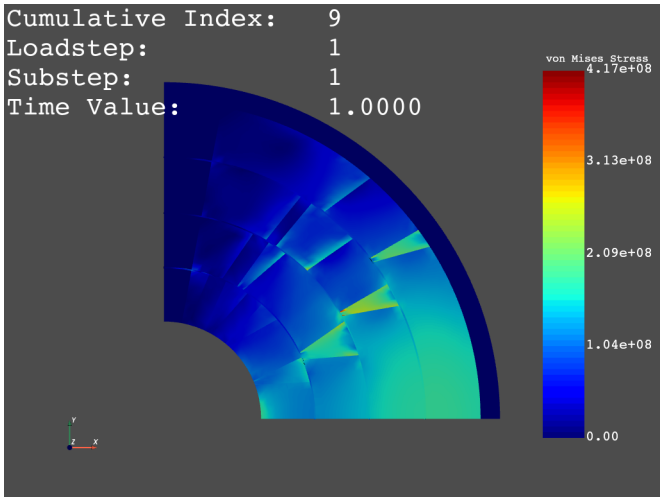

Fig. 5. Von Mises stress distribution (in Pa) in superconducting cables and structural elements in a cross-section of a magnet quadrant.

The genetic algorithm controls a population of 20 candidate solutions over 100 generations. At the end of each generation, the trade value is evaluated and genetic operators are applied. Specifically, tournament selection of size 3 is used to choose individuals for mating. Single-point crossover is performed at a randomly selected position to obtain new individuals. Lastly, a random bit-flip mutation is carried out with uniform probability and a threshold value of 0.95.

Models snapshots for each model are stored to the cache database with each iteration of the optimization algorithm. The snapshot of the value model contains the values of the figure of merit, trade values, and an artefact with geometry coordinates corresponding to each individual. During an optimization run, the pieces of information from the snapshot are displayed in an optimization cockpit. Furthermore, the model snapshots can be conveniently compared across several optimization runs. The evolution of the minimum and mean trade values per generation are depicted in Figure 6.

Note that due to the application of elitism, i.e., propagation of the best two individuals from generation to generation, the trade value is non-increasing. Furthermore, the mean of the trade value is gradually decreasing indicating that the entire population improves from generation to generation.

VI. Conclusion

In this paper, we presented an MBSE approach to collaborative modeling for accelerator magnets and design optimization. The methodology is based on standard, open-source technologies. We begin with the application of containers for encapsulation of computational environments. Numerical solvers are embedded into containers and provide an encapsulated and reproducible environment, exposing a language-agnostic service API. Each modelling task is implemented in a

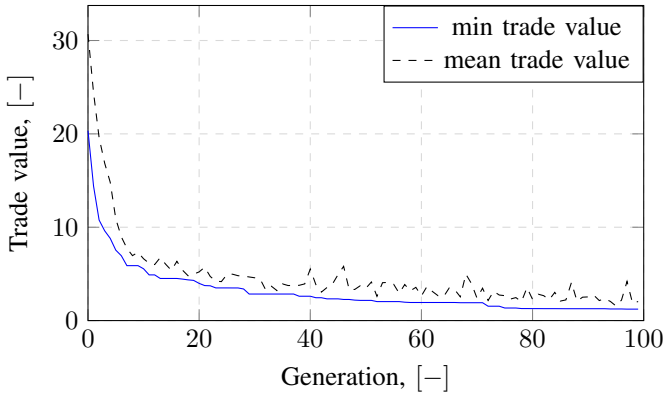

Fig. 6. Minimum and population mean trade value for genetic optimization with elitism. The population mean is computed by excluding individuals with penalized scores.

notebook, controlling model creation, execution, and query of results. A model registry enables efficient queries of models.

It is important to note that, while the sum of all methods will give the best result, the implementation of only a subset, i.e., of one or two of the methods, will still provide substantial benefits. Notebooks, if used correctly, can provide traceability better than most other methods. Model queries with pyMBSE create a powerful and consistent link between models, even in the absence of notebooks or encapsulated solvers in containers. Containerization and encapsulation through HTTP makes for a more portable maintainable integrated deployment process.

The MBSE methodology shifts the focus from documents to models in system design: (i) models provide an abstraction of a system; (ii) models generate human-readable views with relevant data and keep track of versions; (iii) models are queried for data by other models; (iv) models and their inputs are versioned for the sake of life-cycle management, consistency and traceability.

The notebooks are combined into a multi-physics coil geometry optimization workflow which is executed programmatically. The result is documented in an auto-generated report [19], illustratory example. Nonetheless, the presented MBSE methodology itself is applicable beyond a magnet design. This approach naturally extends to other components of a particle accelerator such as beam dynamics, cryogenic systems, vacuum installation, power converters, etc.

VII. Acknowledgement

The authors would like to thank the TE-MSC-TM section at CERN for a fruitful collaboration on ROXIE containers. The authors would like to also thank Alex Kaszynski for support with PyANSYS python package and ANSYS containers.

References

[1] M. N. Wilson, "Superconducting magnets," *Clarendon Press* Oxford, 1983.

[2] B. Auchmann, L. Brouwer, Caspi,J. Gao,G. Montenero,M. Negrazus,G. Rolando,S. Sanfilippo, "Electromechanical Design of a 16-T CCT Twin-Aperture Dipole for FCC", *IEEE Transactions on Applied Superconductivity*, 28(3), 2018.

[3] M. Prioli, T. Salmi, B. Auchmann, L. Bortot, M. Maciejewski, A. Verweij, B. Caiffi, S. Farinon, C. Lorin, M. Segreti, A.M. Fernandez, J. Munilla, "The CLIQ Quench Protection System Applied to the 16 T FCC-hh Dipole Magnets", *IEEE Transactions on Applied Superconductivity*, 29(8), 2019.

[4] M. Prioli, B. Auchmann, L. Bortot, M. Maciejewski, T. Salmi, A. Verweij, "Conceptual Design of the FCC-hh Dipole Circuits With Integrated CLIQ Protection System", *IEEE Transactions on Applied Superconductivity*, 29(8), 2019

[5] E. Crawley, B. Cameron, and D. Selva, "System Architecture: Strategy and Product Development for Complex Systems," *Prentice Hall Press*, New Jersey, April 2015.

[6] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and MBSE opportunities for automotive system design," *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2075-2080.

[7] J. Crane and L. Brownlow, "Optimization of multi-satellite systems using integrated Model Based system Engineering (MBSE) techniques," *2015 Annual IEEE Systems Conference Proceedings*, 2015, pp. 206-211.

[8] T. Kluyver, et al. "Jupyter Notebooks – a publishing format for reproducible computational workflows," *In Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016, pp. 87–90.

[9] F. Pedregosa, et al. "Scikit-learn: Machine Learning in Python,", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[10] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, 2014

[11] MBSE integrated with Teamcenter ties the entire cross-product lifecycle together, Siemens, https://www.plm.automation.siemens.com/global/en/products/collaboration/mbse-model-based-systems-engineering.html, Access date: 09.08.2022.

[12] A. Kaszynski, "pyansys: Python Interface to MAPDL and Associated Binary and ASCII Files, " *Zenodo*, 2020, [Online]. Available: https://github.com/pyansys/pymapdl

[13] C. Hafner, C. Xudong, J. Smajic, R. Vahldieck, "Efficient procedures for the optimization of defects in photonic crystal structures," *Journal of the Optical Society of America A*, vol. 24, 2007, pp. 1177-1188.

[14] H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," *The Computer Journal*, vol. 3, 1960, pp. 175-184

[15] S. Russenschuck and B. Auchmann. ROXIE 10.2., [Online]. Available: http://cern.ch/roxie.

[16] ANSYS©, Mechanical ANSYS Parametric Design Language, Release 2020R1

[17] G. Vallone, B. Auchmann, M. Maciejewski, J. Smajic, "Magneto-Mechanical Optimization of Cross-Sections for cos(θ) Accelerator Magnets," *IEEE Transactions on Applied Superconductivity*, 2022, in preparation

[18] PyMBSE User Documentation, https://chart-magnum.github.io/pymbse-docs, Access date: 25.01.2023

[19] Example of a CDR report, https://chart-magnum.github.io/compumag-cdr-example/intro.html, Access date: 09.02.2022