Nacka Strands GYMNASIUM

Magisim

Open source resource center for simulations

Jonas Cronholm
Gymnasiearbete 100 poäng
Klass: 21 TE
Teknikprogrammet
Läsåret: 2023-2024
Handledare: Peter Eriksson

Abstract

With serious competition in the modern market it is not trivial to create a product that can match the expectations of professional users and still appeal to hobby engineers, especially at a modest price point. The gap in capabilities between enterprise and open source simulation software is growing bigger due to lack of funding, which often results in limited research and development for free and open source projects. This growing disparity poses a significant challenge for developers aiming to balance advanced features with affordability.

Enterprise simulation software benefits from substantial financial backing, enabling teams to invest in cutting-edge research, continuous enhancements, and dedicated customer support. This translates to robust features, high accuracy, and comprehensive technical support, making them the preferred choice for intricate and mission-critical projects. However, the cost associated with these solutions can be prohibitive for smaller businesses and individual hobbyists. This gap is further exacerbated by the intricate nature of simulation software. Meeting the needs of professionals frequently involves complex algorithms, intricate modeling, and high-performance computing. Striving to offer similar capabilities within a modest price range for hobbyists can be daunting, as the associated costs can quickly spiral upwards.

Addressing this challenge, Magisim, an open-source project, is introduced as a versatile solution, leveraging parallelization to enhance simulation efficiency and accessibility. Designed primarily as an educational tool, Magisim integrates a wide range of simulation algorithms with a node graph interface, simplifying the dataflow programming for users of varying expertise. The cornerstone of Magisim's innovation lies in its implementation of parallel computing techniques. By harnessing the power of parallelization, Magisim significantly reduces computational time and resource requirements, making advanced simulations more accessible to amateurs and cost-effective for professionals. This approach not only democratizes the learning and application of complex simulations but also bridges the gap in performance between high-cost enterprise solutions and open-source software. Consequently, Magisim serves as a testament to the potential of open-source development in narrowing the divide in technological capabilities, fostering an environment where both professionals and hobbyists can explore, learn, and contribute to the evolving field of simulation and data analysis.

Keywords: Computational electromagnetics, CEM, MoM, FEM, parallelization, simulator, CUDA, gradio, Data Analysis, Visualization

Contents

1 Introduction

1.1 Background

The field of radio engineering, particularly within the amateur radio sphere, necessitates a deep understanding of electromagnetic (EM) theory and its practical applications. Traditionally, this domain has been supported by a variety of simulation tools, aiming to provide insights into the complex interactions of electromagnetic fields with their surrounding environments. However, a significant challenge has emerged from the existing landscape of these simulation tools, marked by their prohibitive cost and steep learning curves. This divide presents a substantial barrier to entry for amateur radio enthusiasts and learners, who often seek both affordability and simplicity in educational resources.

1.2 Problem

Historically, professional-grade simulation software in electromagnetics has been tailored to meet the demanding needs of industry experts, incorporating advanced features and comprehensive simulation capabilities. While powerful, these tools come at a price point that is often beyond the reach of hobbyists and educational users. Concurrently, the free or low-cost alternatives available in the market tend to compromise either on the width of features or on user accessibility. The complexity of these tools, coupled with often inadequate documentation, further exacerbates the challenge for those new to the field, impeding practical, hands-on learning.

This gap in the market led to the conceptualization of Magisim, initially envisioned as a user-friendly and cost-effective simulator specifically for electromagnetics. Targeted towards the amateur radio community, the primary goal was to demystify EM theory through interactive simulations, making it more approachable for non-professionals. However, as the project progressed, we realized the broader potential for the underlying technology. The modularity of our initial design, made the software applicable in more fields related to simulations or data analysis. This steered the development of Magisim towards becoming a more versatile platform, transcending its original scope.

1.3 Introduction to Computational Electromagnetics

Computational electromagnetics (CEM) is a field of study that employs numerical methods to solve problems that involve electromagnetic interactions. These computational methods are essential for designing, analyzing, and optimizing electronic and optical devices, such as antennas, radar, waveguides, and optical fibers.

A key technique in CEM is the Finite-Difference Time-Domain (FDTD) method, which simulates electromagnetic field behavior by solving Maxwell's equations across a discretized spatial and temporal grid. This method is prized for its ability to model complex interactions in various materials and structures accurately. FDTD is particularly useful in fields such as antenna design, optical device analysis, and electromagnetic compatibility testing. It allows for a detailed examination of how electromagnetic waves propagate and interact with their environment, making it an essential tool for both researchers and engineers. In educational settings, tools like Magisim utilize FDTD to help students and hobbyists understand and visualize electromagnetic phenomena in a straightforward and accessible manner.

1.4 Related Work

1.4.1 Proprietary Software

1.4.2 Open Source Projects

In the field of electromagnetic simulations, the openEMS tool, described by Liebig et al. (2012), stands out as a noteworthy open-source platform leveraging the Finite-Difference Time-Domain (FDTD) method. This tool, freely available to the public, supports various simulation needs, particularly for high-frequency electromagnetic applications. The openEMS platform uses an equivalent-circuit (EC) FDTD method, which simplifies the implementation of material dispersion, thus enhancing the simulation of complex electromagnetic environments. Although openEMS is a powerful tool, its depth and the required scripting for setup and control can be daunting for amateurs and new users. Our aim with Magisim is to maintain the rigorous simulation capabilities of openEMS while enhancing accessibility and ease of use, thereby making sophisticated electromagnetic simulations more approachable for a broader audience.

In his PhD thesis, Floris Laporte employs a custom Finite-Difference Time-Domain (FDTD) simulator to analyze photonic structures that mimic neural processes for neuromorphic computing applications. The FDTD method is crucial for accurately modeling the propagation and interaction of electromagnetic fields within complex photonic devices. Laporte's work focuses on the simulation aspect, using this method to delve into the dynamic behaviors of photonic cavities and their potential as elements in brain-inspired computing architectures. In developing our simulator, Magisim, we have integrated Laporte's FDTD code as a core component. This integration allows us to leverage the established efficacy of his simulation tools to enhance our ability to model electromagnetic phenomena accurately and efficiently within our open-source platform (Laporte, 2020).

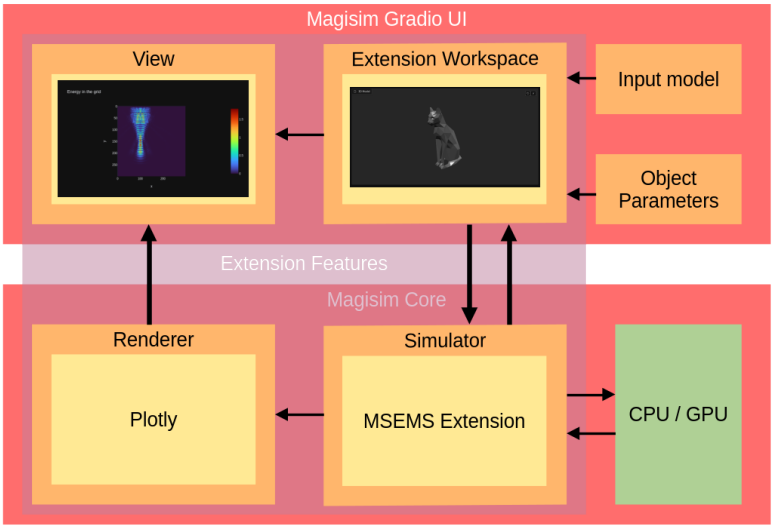
1.5 Method

The initial phase of the project involved comprehensive planning using the Kanban system implemented through Atlassian Trello. This enabled efficient organization and prioritization of envisioned functionalities for the simulator and its interface. The systematic arrangement of tasks on the Kanban board facilitated a clear visualization of the project scope, ensuring effective integration of all functionalities. Subsequently, a Gantt chart was utilized to structure the project's timeline and responsibilities, allowing for a detailed outline of the time plan and division of labor.

1.5.1 User Interface

Following the establishment of the project's framework and timeline, focus shifted to the user interface (UI) design for Magisim. Several UI development toolkits, including QT and GTK, were evaluated to determine the most suitable option based on modularity, integration ease, and development efficiency. Despite the robust features of QT and GTK, challenges were encountered with their steep learning curves and complex integration processes, including licensing constraints.

The search ultimately led to the selection of Gradio, a novel UI library distinguished by its modular design and streamlined development process. Gradio's capability for rapid prototyping and its intuitive user experience made it the preferred choice for the project, facilitating the swift iteration of UI components.



Each extension describes its own user interface by using modules available in the Gradio package. This allows third parties to rapidly develop intuitive and efficient interfaces that integrate seamlessly with the core functionalities of Magisim. The use of Gradio's modules offers developers the flexibility to customize and extend the interface according to specific needs, promoting a high degree of adaptability and user-centric design.

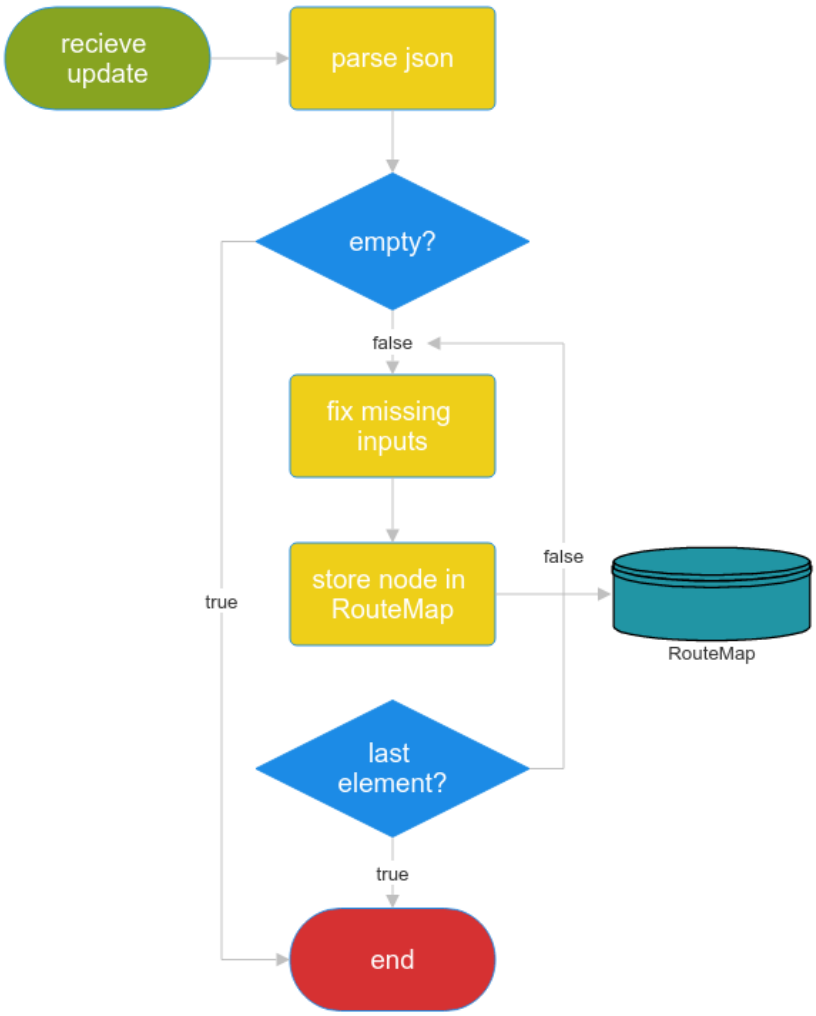
1.5.2 Extension Model and Templates

Aiming for high modularity, the project was structured so that every component was defined as an "Extension". This approach enabled the development of a unified system where all components adhered to a set of predefined rules, including a universal code structure. Such standardization allowed for the components to be loaded by a single function at startup, simplifying the development process and reducing program complexity.

```
# shared extension imports
import gradio as gr
from shared.builtin import Extension
from shared.config import Config

class ExtensionMeta:
name: str = "name"
uuid: str = "some-uuid-v4"
authors: list = ["author"]
version: str = "1.0.0"
license: str = "license"
description: str = """multiline description if needed"""

class ExtensionType:
types: list = [Extension.Simulator, Extension.Workspace]
hasNodes: [(Extension,(list,list))] = [
(Extension.Simulator, # define a simulator node
        (
                [ # inputs
                        ("some-input","some-type")
                ],
                [ # outputs
                        ("some-output1","some-type"),
                        ("some-output2","some-type")
                ]
        )
)]

def load_workspace(app: gr.Blocks):
        with gr.Tab(ExtensionMeta.name, id=ExtensionMeta.uuid):
                gr.Markdown(ExtensionMeta.description)
                # Extension UI code
```

1.5.3 Message Queue and Routing

In defining component interaction and data handling, the initial consideration was to develop an in-house asynchronous communication layer. However, it was soon recognized that many solutions existed for such functionality. A message queue system in combination with a "Router" was chosen to enhance modularity.



The router maps connections between extensions in a Redis database, allowing extensions to remain unaware of the destinations for their outputs. Using Python's Pickle library, the system enables the transmission of diverse data types, including 3D models and tensor arrays.

```
# > is used as a delimiter
connection = f'connection>{extension_id_1}>{extension_id_2}'
Router.redis_client.set(connection, 'connected')
```

1.6 Scope

2 Approach

2.1 Software Structure

2.2 User Interface

The UI (User Interface) for Magisim is built using Gradio, a toolkit originally developed for rapid development of interfaces for Machine Learning projects. We found this

2.3 Application Backbone

2.4 Our FDTD Extension

In the development of our simulator extension, we initially planned to create a custom framework for Finite-Difference Time-Domain (FDTD) simulations. However, it soon became clear that developing our own framework from scratch was both impractical and unnecessary, considering the variety of existing FDTD libraries available in Python.

We built our FDTD-simulation extension upon the library developed by Floris Laporte, which provides a robust framework for conducting Finite-Difference Time-Domain (FDTD) simulations.

2.5 Software Distribution

3 Result

4 Discussion

5 Conclusion

6 References

Liebig, T., Rennings, A., Held, S., & Erni, D. (2012). openEMS – a free and open source equivalent-circuit (EC) FDTD simulation platform supporting cylindrical coordinates suitable for the analysis of traveling wave MRI applications. International Journal of Numerical Modelling (Print), 26(6), 680–696. https://doi.org/10.1002/jnm.1875

Laporte, Floris. (2020). Novel architectures for brain-inspired photonic computers [PhD Thesis]. Ghent University.

Kozlov, M., & Turner, R. (2010). A Comparison of Ansoft HFSS and CST Microwave Studio Simulation Software for Multi-channel Coil Design and SAR Estimation at 7 T MRI. PIERS ONLINE, 6(4), 395.

Written in LaTeX