MRI Sequence Design and Simulation as a Service in a Free and Open-Source Web Platform

Pablo Villacorta-Aylagas[1*], Manuel Rodríguez-Cayetano[1,2], Carlos Castillo-Passi[3,4,5,6], Pablo Irarrázaval-Mena[3,4,5], Federico Simmross-Wattenberg[1,2], Carlos Alberola-López[1,2]

[1*] Laboratorio de Procesado de Imagen, Universidad de Valladolid, Paseo Belén 15, Valladolid, 47011, Spain.
[2] IBioVall, Institute for Health Research, C. Rondilla Sta. Teresa, s/n, Valladolid, 47010, Spain.
[3] Institute for Biological and Medical Engineering, Pontificia Universidad Católica de Chile, Santiago, Chile.
[4] Millennium Institute for Intelligent Healthcare Engineering (iHEALTH), Pontificia Universidad Católica de Chile, Santiago, Chile.
[5] School of Biomedical Engineering and Imaging Sciences, Kings College, London, UK.
[6] Department of Radiology, Stanford University, California, USA.

*Corresponding author(s). E-mail(s): pablo.villacorta@uva.es;
Contributing authors: manuel.rodriguez@uva.es; cncastillo@uc.cl; pim@uc.cl; fedsim@uva.es; carlos.alberola@uva.es;

Abstract

We present MRSeqStudio, a new *all-in-one* web-based tool for MRI sequence development and simulation, with the physics-based simulator KomaMRI running at the back-end and our own sequence designer at the front-end. It combines accessibility, interactivity and technical flexibility, within an environment suitable for both education and research. Our tool provides MR sequence design and simulation *as a service*, with no local installation needed by the user; alternatively, the code is publicly available on GitHub, for users who wish to deploy the application on their own server.

Keywords: MRI, simulation, sequence design, web service.

1 Introduction

Pulse sequence design enables the development of novel magnetic resonance imaging (MRI) acquisition protocols —as well as the optimization of those already existing— for reducing scan time, improving or achieving new image contrasts or eliminating artifacts. Complementarily, simulation provides the ground to test these sequences without the need to access a physical scanner; it also provides the capability to analyze specific phenomena in the acquisition process (e.g., off-resonance effects) and to synthesize images for further usage, such as training of learning models or creation of signal dictionaries [1]. Moreover, simulation serves as a powerful resource for education and training [2]. The coordination of both activities, namely, sequence design and simulation, is essential to fully harnessing the potential of MRI technology.

Originally, sequence design relied on vendor-specific frameworks, which were only available to scanner manufacturers [3, 4] or to associated research institutions. In response to this, open-source initiatives for sequence design [3–9] have emerged in recent years. Some of them also include a web interface that provides easier access, and has led to coining the term *sequence as a service* [4]. As for sequence exchange, Pulseq [10, 11] has become the *de facto* standard in the field.

A number of MRI simulators have been released, either with simple simulation engines based on evaluating the analytical expressions of the most common sequences [2, 12], or with more involved physics-based engines. About the latter, open-source tools such as JEMRIS [13], MRiLab [14, 15], KomaMRI [1], and CMRsim [16] are well-known. Proprietary options have also been reported, such as BlochSolver [17] or Corsmed, which is an evolution of MRISIMUL [18, 19] and coreMRI [20]. About the open-source simulators, JEMRIS, CMRsim and KomaMRI provide support for complex motion, cardiovascular MR (CMR), and MR angiography (MRA) simulations. KomaMRI stands out in terms of performance [21]; this is due to the fact that CMRsim is written in Python and the original JEMRIS lacks GPU support; KomaMRI, however, is written in Julia and benefits from its vendor-agnostic GPU support. Moreover, a recent GPU-compatible extension of JEMRIS does not seem to be competitive with KomaMRI [22].

As previously stated, the coordination of sequence design and simulation seems mandatory. However, software tools in which these two activities are integrated is scarce. Actually, most of the pulse design tools enumerated above do not have direct interfaces to MRI simulators but rely on Pulseq as an intermediate step. An exception to this seems to be CAMRIE [23, 24], presented as *a comprehensive cloud-compatible simulation pipeline*. However, it is not publicly available, since it uses a modified non-public version of KomaMRI and, as described in [24], the project remains under development. Complementarily, [9] also reports a connection with a simulation engine but the simulator itself is not clearly identified.

This paper introduces a new web-based tool, referred to as MRSeqStudio, that bridges the gap between sequence design and MRI simulation, with KomaMRI running at the back-end and our own sequence designer at the front-end. It aims to combine accessibility and interactivity with technical flexibility, offering an environment suitable for both education and research. Graphical capabilities of KomaMRI running in the back-end are displayed in the front-end with no loss of interactivity.

Sequence design is accompanied with different viewers (sequence, slice selection and simulation results). Global variables can be arbitrarily defined by the user and then referenced within the configuration of individual sequence blocks, allowing high-level parameters such as TE, TR, or FOV to be defined once and automatically propagate to all dependent blocks in the sequence. Overall, our tool provides MR sequence design and simulation *as a service*: it requires no local installation, and offers an end-to-end workflow in which pulse sequences can be designed in the browser, simulated in the server —i.e., the *cloud*—, and stored together with their corresponding results.

2 Design and Implementation

2.1 Design objectives

Our main objective is to build an MRI sequence editor with an integrated simulator that is easy to use for prospective users and achieves high physical accuracy and competitive simulation times with respect to the state of the art in the open source. Specifically, our aim is to provide the community with an application with the following characteristics:

Block-by-block sequence design: blocks are defined as RF pulses, gradients, delays or readout windows. Each block has its own parameter set according to its class. By freely arranging these blocks, the user is able to build complex MRI sequences bottom-up. Blocks can be added, modified, moved and deleted by using the keyboard or the mouse. Blocks can also be grouped into composite blocks which behave like regular blocks, thus achieving an arbitrary grade of complexity while maintaining the ease of use that blocks provide.
Parseable user-defined global variables: Users may define global variables, which can subsequently be referenced and manipulated throughout the editor. For example, one could set variables $A = 45$ and $B = 30$ and then set the flip angle of an RF pulse to the value $A + B$.
Interactive diagnostics panels: as the user composes arbitrarily complex sequences by combining different blocks at will, the resulting sequence is displayed (by pressing a button) in a time sequence diagram visualizer and a 3D slice viewer. The selected —i. e., simulated— slice is displayed in a third view pane as well. The user can select between an immediate *slice* visualization and a realistic *volume* rendering via the KomaMRI simulator at the back-end.

2.2 Functional Design

The general operation of the application is illustrated in Fig. 1 and is based on a client-server architecture, specifically, a REST architecture. The front-end runs in the browser, where user interactions generate HTTP requests that are sent to the server. This server, acting as an intermediary between the client and back-end resources —such as the MRI simulator, database, or front-end files— uses a REST API. It follows the typical request-response model: clients send requests, and the server processes them and returns appropriate responses.

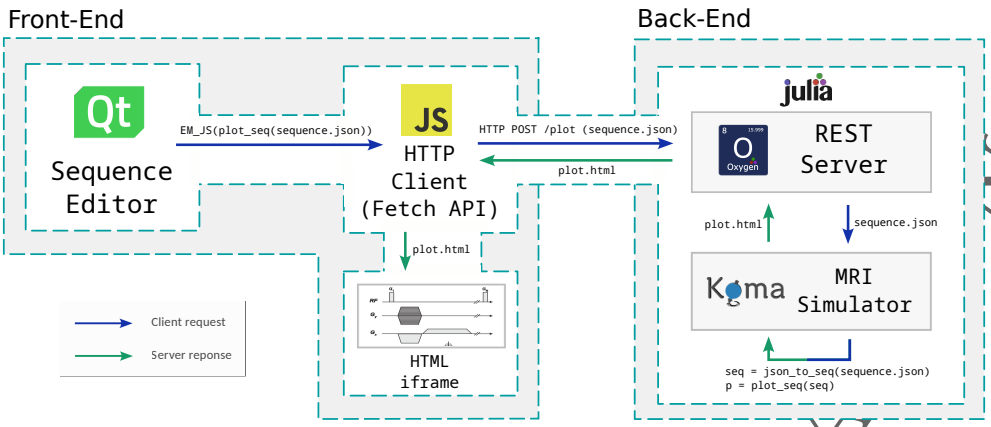


Fig. 1: General operation of the web application. The web browser is responsible for both rendering the front-end and acting as an HTTP client to communicate with the server. The HTTP server, in turn, provides a REST API to the web browser for accessing the back-end functionalities. Blue arrows represent client-side requests; green arrows represent server responses.

Graphical Sequence Editor

The editor contains drop-down menus for file handling —creation, loading, and saving—, as well as for plotting functionalities, and panels providing a) a schematic overview of the sequence, b) predefined blocks that can be freely added to the sequence, c) block groups management, d) selected block parameters, e) scanner parameters, f) user-defined global variables, g) an editable text field where the user can input a description for the sequence, and h) phantom selection and simulation launcher. Qt has been chosen as the core framework for this part of the work for its ability to create cross-platform GUI with a native appearance and usability. Compiling to WebAssembly enables the execution of Qt (C++) applications within a web browser [25].

Sequence Diagram Viewer

This module consists of a single panel displaying the sequence diagram generated by KomaMRI. On the client side, a JSON file containing the complete sequence data from the GUI is generated and sent to the REST server via a POST request. These request flows are represented with blue arrows in Fig. 1. On the server side, KomaMRI processes the request and generates an interactive HTML plot using PlotlyJS.jl. This HTML file is sent back as the server's response, which the front-end embeds into the sequence diagram panel. This response flow is depicted with green arrows in Fig. 1, completing the communication cycle and ensuring seamless integration of the sequence visualization.

3D Slice Viewer

The 3D visualization module consists of a single panel displaying the phantom volume loaded from a NIfTI file, along with the slice selected by the MRI sequence, which is also part of that volume. The visualization is achieved using three orthogonal slice planes[1], as this reduces the computational load on the client side by avoiding rendering the full volume. Additionally, the slice selected by the MRI sequence is shown as another slice plane, which results in four planes being displayed in total: three orthogonal planes plus the selected one. The implementation of this module is based on the vtk.js library, which provides efficient in-browser rendering.

Simulation Result Viewer

Fig. 1 can also be used as a reference for understanding this part of the system, as both the sequence viewer and the simulation results viewer are entirely analogous. The only difference is that, in this case, the server must first receive not only the sequence information but also the phantom and the scanner data, enabling it to perform the simulation and return the corresponding results. These results are provided as an HTML file which contains the plot generated by KomaMRIPlots.jl.

2.3 Back-end Module and front-end integration

The back-end of the application consists of an HTTP server and additional processes responsible for generating the necessary data. The HTTP server acts as an intermediary between the front-end and the rest of the back-end. It receives requests from the client, forwards them to the appropriate internal components, and returns the generated results. Notably, the entire back-end module has been implemented in Julia, and its source files can be found in the `/backend` directory of the repository[2]. The application is designed so that all repository files, including both back-end and front-end components, reside on the server machine. The client, i.e., the end user, only requires a web browser to access and use all functionalities.

For the design of the HTTP server, a REST API has been defined to handle incoming HTTP requests from clients. In this architecture, the REST API and the HTTP server are unified in the same process. Once the HTTP request reaches the API, it directly calls Julia functions to interact with the back-end resources, bypassing HTTP for internal processing. This approach ensures that communication with the back-end occurs efficiently within the server process, without needing additional HTTP calls. The implemented methods are documented[3] and categorized into three sections: web content rendering, simulation, and plotting.

The client needs to process the response content, ensuring appropriate actions are taken based on the data received. When necessary, the information should be displayed on the screen in the correct format. For example, when a simulation is ongoing, progress updates should be displayed, and upon completion, the result should be shown in a meaningful way to the user.

---

[1] As outlined in the viewer developed by [2], this approach follows a similar methodology.
[2] See https://github.com/pvillacorta/MRSeqStudio/tree/master/backend.
[3] API documentation available in: https://petstore.swagger.io/?url=https://raw.githubusercontent.com/pvillacorta/MRSeqStudio/refs/heads/master/docs/api.yaml.

3 Results

3.1 Application overview. GUI and Features

Fig. 2 shows the main interface of the application[4], which occupies the entire available screen space and adapts to the browser window size. Thanks to this responsive design, the application can be executed seamlessly on mobile device browsers, as illustrated in Fig. 3. The layout is organized into multiple panels that ensure a clear separation of functionalities. Panels A–G are dedicated to sequence design. In panel A, the sequence is constructed my means of blocks, which are placed sequentially and can be freely rearranged through *drag-and-drop* operations. These blocks can be also wrapped into groups that allow repetitions and enable the user to emulate, for instance, the set of sequence events occurring during a TR. Panel B provides a menu with buttons for different block types; clicking one inserts the corresponding block into panel A. Panel C stores the groups that, once created, can be replicated and instantiated within the sequence. Panel D displays the configuration menu and adjustable parameters which correspond to the block selected in Panel A. Panel E is used to define the parameters of the MRI scanner on which the sequence is to be executed; these include, among others, the main magnetic field strength B_0 and the maximum values for the RF pulse amplitude, gradient strength, and slew rate supported by the system. Panel F provides a lightweight programming environment where global variables can be defined as either numerical values or mathematical expressions. Such variables can be used to control block parameters and ensure that modifications to high-level settings —such as TE, TR or FOV— automatically propagate without the need to manually adjust individual blocks. Panel G allows the user to define a sequence description in plain text format. Panel H is used to select the anatomical model —i.e., the phantom— for visualization and simulation. The "Simulate" button can then be used to launch the back-end simulation using the sequence currently being developed in the interface and the chosen phantom. Panels I–K serve as interactive visualization modules, and display, respectively, the sequence temporal diagram, the phantom, and the MRI simulation output. Specifically, the phantom viewer offers two visualization modes: a lightweight representation based on three orthogonal slices, and a volume-rendering mode in which the individual spins composing the phantom can be displayed. The former mode also allows the user to observe the specific slice selected by the sequence (see Fig. 2), and the latter enables the visualization of phantom motion or blood flow if present. Panel L is a menu bar that provides file handling for loading and saving sequence files, as well as a menu to display the sequence diagram in Panel I. Finally, button M opens the user panel, which allows managing the current session, logging out, or accessing user information, stored sequences, and results from previous simulations.

In addition to the main interface, the application includes dedicated panels for complementary tasks. A results panel (Fig. 4) allows each user to access and review previously generated simulations. For administrators, a dedicated panel (Fig. 5) provides tools for user management, usage statistics, and inspection of stored sequences and results, organized across three tabs.

[4] Available at https://mrseqstudio.lpi.tel.uva.es.

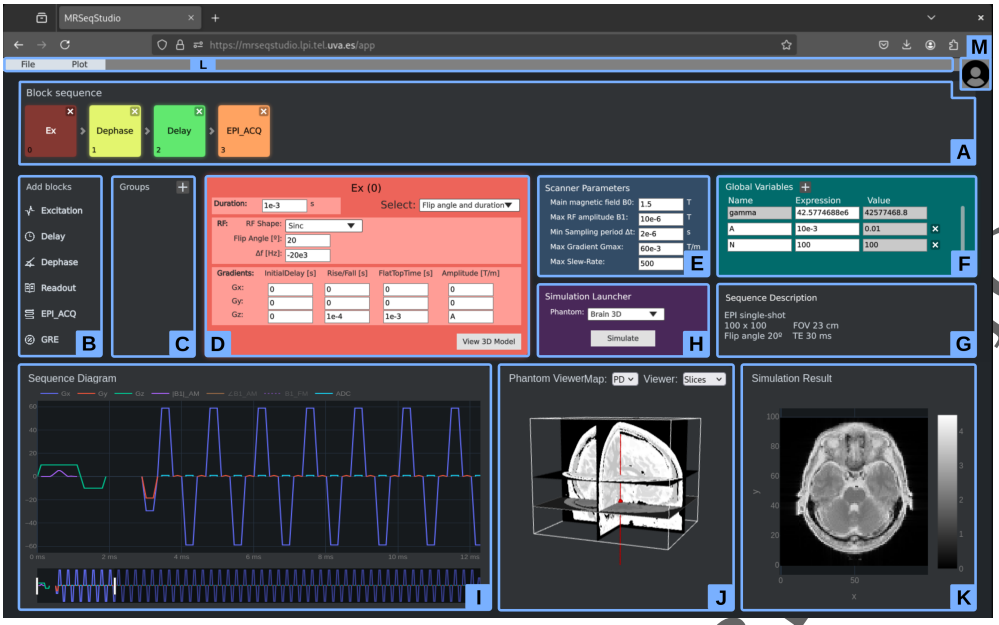


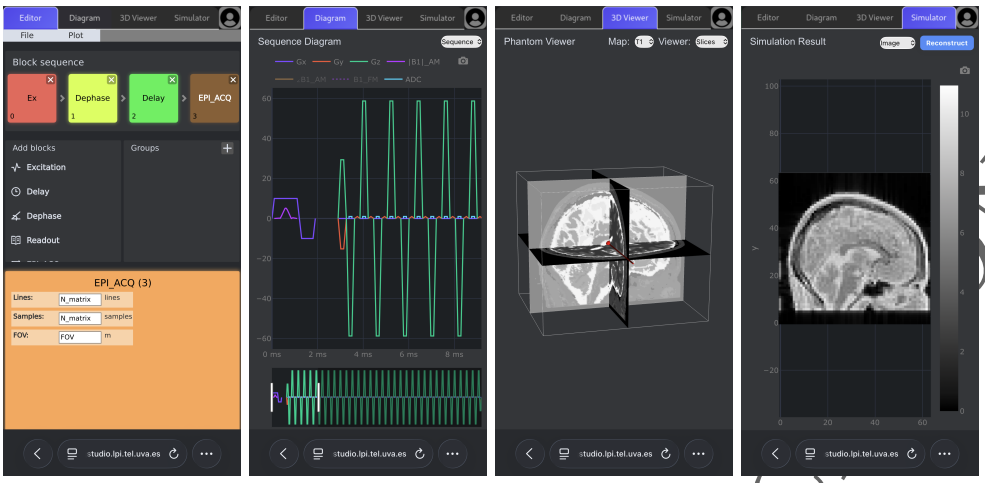
Fig. 2: Application main layout, which is divided into panels that ensure a clear separation of functionalities. The example shown corresponds to the design and simulation of a GE-EPI sequence on a 3D brain phantom.

3.2 User Workflow and Examples

The application is intended to accommodate different types of users, ranging from radiographers to sequence programmers and MRI researchers. Depending on the user profile, the workflow slightly differs. For radiographers, the typical approach consists of loading pre-defined sequences and adjusting global parameters (such as TE, TR, or FOV) before running a simulation. All of the examples presented in this paper fall under this usage profile. By contrast, researchers and sequence programmers can design sequences from scratch by building them block by block, defining custom variables, and creating reusable groups.

EPI

One of the simplest use cases is the loading and adjustment of a single-shot GE-EPI sequence. The interface shown in Fig. 2 illustrates this process through the concatenation of four blocks: the `Ex` block implements a sinc-shaped RF pulse with a frequency offset of −20 kHz and a simultaneous slice-select gradient in z. This offset shifts the excited slice downward along the z-axis, so that its center is no longer located at the origin; the `Dephase` block adds a negative z gradient immediately after the RF pulse (i.e., a post-excitation dephasing [26]); the `Delay` block introduces a 1 ms delay; and the predefined `EPI_ACQ` block performs a 100×100 point EPI acquisition, fully covering k-space. In the global variables panel, in addition to the default variable `gamma`, the variables `A` and `N` are defined. These variables control, respectively, the amplitude



(a) (b) (c) (d)

Fig. 3: Screenshots of the application running in a mobile browser. The example corresponds to the design and simulation of a GRE-EPI sequence. The slice-selection gradient applied along the x-axis produces a sagittal slice on the 3D brain phantom. (a) Sequence editor. (b) Sequence diagram visualizer. (c) 3D Phantom visualizer. (d) Simulation result.

of the z gradients in both the `Ex` and `Dephase` blocks, and the number of k-space lines and points per line which are sampled within the `EPI_ACQ` block. Simulation is then carried out over a 3D brain phantom. The Supplementary Video S1 further demonstrates this use case.

Spin Echo

Another use case is the loading and adjustment of a spin echo sequence. Fig. 6 shows the application GUI with the corresponding block arrangement. The first key aspect to highlight is the presence of a block group called `TR`, which contains all blocks that repeat every TR. The number of repetitions of this block has been set to `N_matrix` = 100, a value that has also been assigned to the `Samples` field of the `Readout` block. This results in a 100×100 k-space matrix. The two `Ex` blocks represent the 90^{o} and 180^{o} RF pulses. The first `Dephase` block applies gradients in x and y to position the readout pointer at the top-right corner of k-space, and in z to compensate for the effect of the slice-selection gradient. The second and third `Dephase` blocks act as a crusher gradient pair straddling the 180^{o} refocusing pulse [27]. The `Readout` block acquires a single k-space line of 100 points while simultaneously applying a frequency-encoding gradient along the x direction. Finally, the `Delay` blocks introduce timing delays to ensure compliance with the predefined TE and TR values. In this case, multiple global variables have been defined and referenced within each block's configuration. The right panel of Fig. 6 shows simulation results of the spin-echo sequence for three experiments with different TE and TR values. These variations

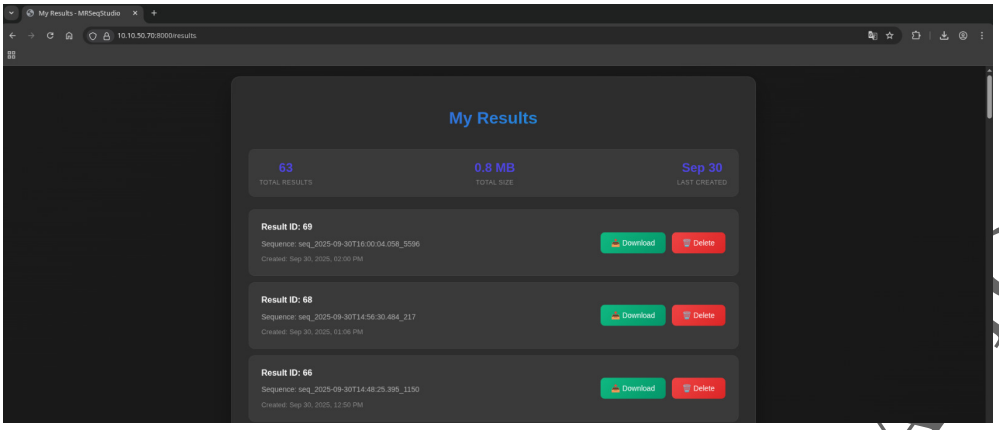


Fig. 4: Simulation results interface. The panel shows the history of simulations conducted by an specific user. These simulation results can be downloaded or deleted.

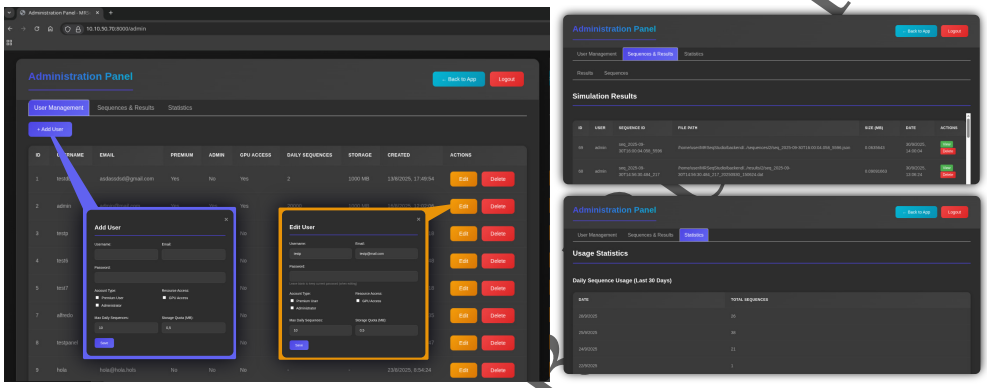


Fig. 5: Administrator panel of the application, which provides tools for user and resource management organized across three tabs. The first tab (left) enables CRUD operations ("Create, Read, Update, Delete") for users. The second tab (top right) allows inspection of sequences and simulation results generated by all users. The third tab (bottom right) displays usage statistics, such as the number of sequences used per day, simulations performed, and related metrics.

alter the image contrast, and enable the acquisition of T1, T2, or PD-weighted images. This example is illustrated in Supplementary Video S2.

3.3 Motion, CMR and MRA Simulation

It is possible to perform simulations over dynamic phantoms, thanks to recent improvements in KomaMRI [21]. This functionality enables the design and testing of motion-related sequences —such as phase contrast and time of flight—, as well as the assessment of motion-induced artifacts when conventional sequences are applied.

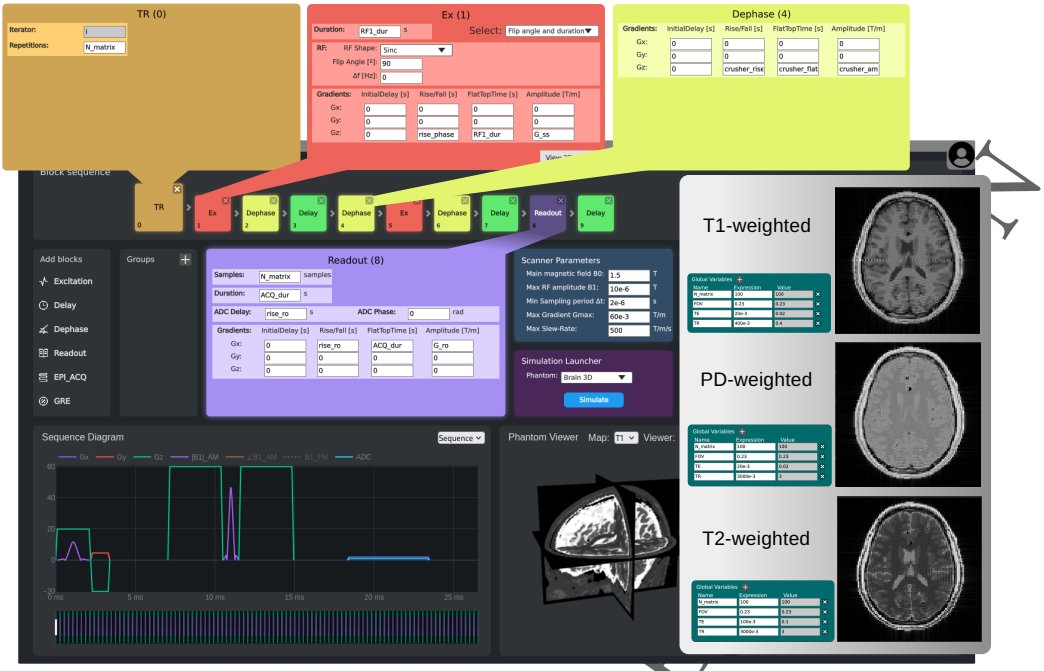
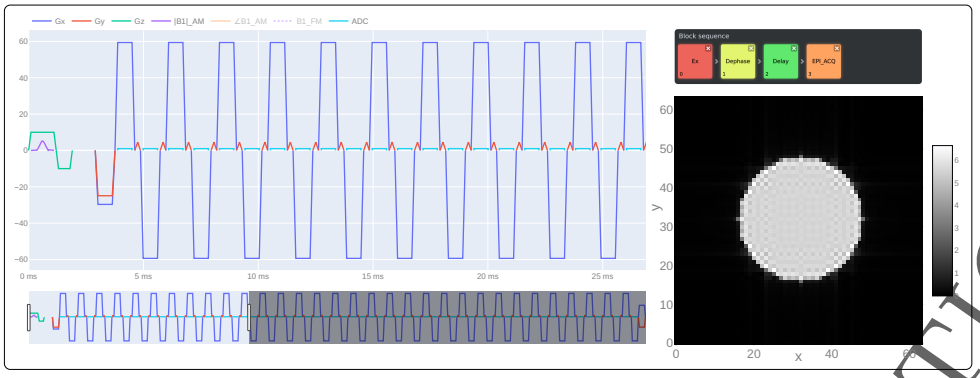


Fig. 6: Application GUI displaying the design of a spin-echo sequence. The sequence is organized into a block group that repeats every TR, along with RF, dephasing, crusher, readout, and delay blocks. Four block configuration panels are shown. The right panel presents the results of three simulation experiments in which TE and TR values were varied.

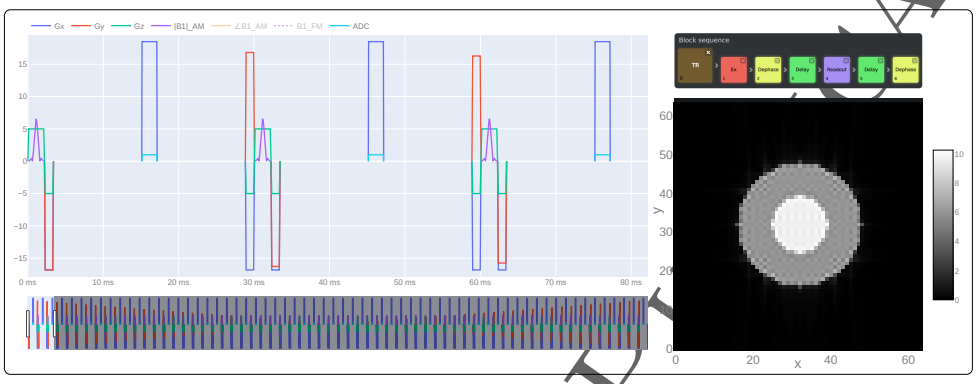
To illustrate this capability, we consider a cylindrical phantom with fluid flowing inside. Both the cylinder wall and the flowing interior were assigned identical T1, T2, and PD, so no intrinsic tissue contrast is present. Then, two different sequences were designed and simulated within the application, selecting the exact same axial slice. The first was an EPI acquisition (Fig. 7a), which produced images where the static cylinder wall and the lumen appeared with equal intensities. The second was a bSSFP acquisition (Fig. 7b), in which the lumen appeared brighter than the wall due to the time-of-flight effect [28]. The illustrative Video S3 accompanies this example.

3.4 Performance

Benchmarks were conducted separately for the tasks executed in the front-end and those performed in the back-end. For the front-end, the initial page (Fig. 2) loads in approximately 3 seconds, most of which are spent downloading the compiled WebAssembly (`.wasm`) file of the sequence editor. Regarding phantom visualization, the vtk.js slice-based viewer runs entirely in the front-end, while the volume-rendering viewer is computed in the back-end. Both processes thus run in parallel, with times depending on phantom size and complexity. For a 2D brain phantom, slice rendering



(a)



(b)

Fig. 7: Simulation of an axial slice of a cylindrical phantom with flow inside. (a) EPI sequence, showing no contrast between static wall and flowing interior. (b) bSSFP sequence, where the lumen appears brighter due to the time-of-flight effect.

takes about 2 seconds and volume rendering about 150 ms; for a 3D brain phantom, these times increase to 10 seconds and 1.5 seconds, respectively. For back-end simulations, while more detailed benchmarking has already been made [1, 21], the simple 2D EPI sequence of Fig. 2 takes approximately 1 second end-to-end —from the moment the simulation is launched in the front-end until the results are displayed back in the interface— when executed on a CPU-only server with an AMD Ryzen 7 5800X 8-core 3.8 GHz processor.

All these performance measurements are strongly influenced by multiple factors, including client and server specifications, and network speed and bandwidth.

4 Discussion

This application constitutes an *all-in-one* environment for MRI sequence development and simulation. Unlike existing approaches that require switching between different software tools for sequence programming, visualization, and MRI simulation, our platform integrates this complete pipeline in a single environment.

The application runs entirely in the web browser and requires no local installations. Trivial operations, such as configuring blocks or loading/saving sequence files, are executed directly in the client, ensuring immediate interactivity. In practice, the initial page loading and phantom viewer rendering only take a few seconds, and once everything is loaded, interactions with the interface are immediate and fully responsive. MRI simulations are performed in the KomaMRI-powered back-end, which is physics-based for accuracy and GPU-compatible for accelerated performance; simulations also run correctly on CPU-only servers, albeit more slowly. As described elsewhere [1, 21], KomaMRI overcomes the main physics-based MR simulators available in the public-domain.

Another distinctive feature of our platform is its support for motion-inclusive simulations. Thanks to a recent enhancement to KomaMRI [21], arbitrary phantom motion can be defined and simulated. This functionality has been integrated into the web application, allowing users to load dynamic phantoms and observe the effects of motion on MRI acquisitions, as illustrated in the time-of-flight experiment. Building on these capabilities, additional features are planned: first, the ability to create and configure phantom motion directly within the web interface, since currently only predefined dynamic phantoms[5] can be loaded. Then, we also plan to include a cine cardiac viewer, as well as specific reconstruction methods and phase image viewers for MRA acquisitions.

The tool provides a fully integrated MRI test environment that gives rise to the aforementioned concept of *sequence design and simulation as a service*. Users can store sequences and simulation results in the cloud, associated with their account, and access them seamlessly at any time. Related improvements include flexible execution on GPU or CPU depending on user privileges (e.g., premium vs. standard) and enhanced user management through database support. Future updates will also allow users to adjust the phantom complexity —mainly the number of spins— directly from the web interface, optimizing simulations for available computational resources.

The results presented in this paper illustrate three examples aligned with a radiographer's workflow, based on loading predefined sequences and adjusting global parameters. Additionally, the application is fully compatible with research-oriented use cases, where sequences can be built from scratch and customized. Although its full-scale educational or research deployment is pending, it has already been tested in pilot scenarios by members of the MRI community; a preliminary version was presented at the ISMRM Iberian Chapter 2025 [29], where it received the best poster award. This recognition underscores the strong interest from the MRI community and the potential of the platform to support both education and research.

[5] Dynamic phantoms can be created and configured in KomaMRI, as described in https://juliahealth.org/KomaMRI.jl/stable/explanation/2-motion/.

All the code for both the front-end and back-end is publicly available on GitHub[6]. Complete installation instructions, including prerequisites and compilation guidelines, are provided for users who wish to deploy the application on their own server.

5 Conclusion

We have presented a web-based application that unifies the entire pipeline of pulse sequence design and MRI simulation within a single environment. By combining an interactive graphical interface with the physics-based KomaMRI back-end, the platform provides a responsive, user-friendly experience while delivering accurate, cloud-powered simulations. The tool is intended for a broad spectrum of users, ranging from radiographers to MRI researchers. Moreover, the implemented *MRI sequence design and simulation as a service* framework includes support for motion-inclusive simulations. Overall, the platform constitutes a versatile, accessible, and extensible resource that lowers the entry barrier to MRI simulation while opening new opportunities for teaching, testing, and research in MR technology. The application is under continuous development, with ongoing improvements —such as enhanced motion handling, cloud resource management, and broader sequence compatibility— that will further expand its capabilities.

Supplementary information

Additional supporting material may be found online in the Supporting Information tab for this article.

- **Video S1**: EPI sequence simulation experiment. A GE-EPI sequence is loaded into the application GUI and some of its parameters are modified. Then, simulation is launched and the result is plotted within the "Results" panel.
- **Video S2**: Spin Echo simulation experiment. A SE sequence is loaded into the GUI and three consecutive simulations are conducted, with varying TE and TR values.
- **Video S3**: Time-of-flight experiment. Two different sequences —a GE-EPI and a bSSFP— are consecutively loaded and simulated over a cylindrical phantom with flow inside.

Declarations

Funding

This work was supported by the Agencia Estatal de Investigación with grants PID2020-115339RB-I00, TED2021-130090B-I00 and PID2022-142166NA-I00, and to Fundación La Caixa with grant HR22-00533. Additional funds correspond to ANID-Chile grants FONDECYT 1210747, FONDECYT Post-doctoral 3240576 and Millennium Science Initiative Program - ICN2021_004.

[6] https://github.com/pvillacorta/MRSeqStudio

Conflict of interest/Competing interests

The authors declare that they have no competing interests.

Ethics approval and consent to participate

Not applicable

Consent for publication

Not applicable

Data and Materials availability

The source code of the application has been deposited in a public GitHub repository: https://github.com/pvillacorta/MRSeqStudio.

Authors' contributions

P. Villacorta-Aylagas was responsible for the conceptualization, implementation, and software development of the tool, as well as for writing the first draft of the manuscript. M. Rodríguez-Cayetano and F. Simmross-Wattenberg contributed to the conceptualization and software implementation, and participated in reviewing and refining the manuscript. C. Castillo-Passi and P. Irarrázaval-Mena assisted in revising the manuscript and clarifying specific aspects related to KomaMRI, which is used in the backend of the tool. C. Alberola-López supervised the work throughout all its stages and contributed substantially to shaping and improving the final version of the manuscript.

All authors have read and approved the final manuscript.

Acknowledgments

The authors acknowledge the financial support of the Agencia Estatal de Investigación, Fundación La Caixa, and ANID-Chile through the projects mentioned in the Funding section. The authors also wish to thank the members of the ISMRM Iberian Chapter for testing the application and providing valuable feedback.

Clinical Trial Number

Not applicable.

References

[1] Castillo-Passi, C., Coronado, R., Varela-Mattatall, G., Alberola-López, C., Botnar, R., Irarrazaval, P.: KomaMRI.jl: An open-source framework for general MRI simulations with GPU acceleration. Magn. Reson. Med. **90**, 329–342 (2023) https://doi.org/10.1002/mrm.29635

[2] Treceño-Fernández, D., Calabia-Del-Campo, J., Bote-Lorenzo, M.L., Gómez-Sánchez, E., Luis-García, R., Alberola-López, C.: A Web-Based Educational Magnetic Resonance Simulator: Design, Implementation and Testing. Journal of Medical Systems **44**, 1–11 (2019) https://doi.org/10.1007/s10916-019-1470-7

[3] Weine, J., McGrath, C., Kozerke, S.: CMRSeq - A Python package for intuitive sequence design. In: 2023 ISMRM & ISMRT Annual Meeting & Exhibition, Toronto, Canada (2023). https://archive.ismrm.org/2023/2398.html

[4] Artiges, A., Saimbhi, A.S., Castillo-Passi, C., Lattanzi, R., Block, K.T.: mtrk—A flexible environment for developing open-source MRI pulse sequences. Magn. Reson. Med. (2025) https://doi.org/10.1002/mrm.70067

[5] Magland, J.F., Li, C., Langham, M.C., Wehrli, F.W.: Pulse sequence programming in a dynamic visual environment: SequenceTree. Magn. Reson. Med. **75**(1), 257–265 (2016) https://doi.org/10.1002/mrm.25640

[6] Nielsen, J.-F., Noll, D.C.: TOPPE: A framework for rapid prototyping of MR pulse sequences. Magn. Reson. Med. **79**(6), 3128–3134 (2018) https://doi.org/10.1002/mrm.26990

[7] Cordes, C., Konstandin, S., Porter, D., Günther, M.: Portable and platform-independent MR pulse sequence programs. Magn. Reson. Med. **83**(4), 1277–1290 (2020) https://doi.org/10.1002/mrm.28020

[8] Cencini, M., Wang, K., Huang, S., Schulte, R.F., Sprenger, T., Noll, D.C., Tosetti, M., Nielsen, J.-F.: Pulserver: an open-source Pulseq-based client-server framework for vendor agnostic, interactive MR sequence design. In: 2025 ISMRM & ISMRT Annual Meeting & Exhibition., Honolulu, Hawaii, USA (2025). https://archive.ismrm.org/2025/1275_4GECnHR7z.html

[9] Konstandin, S., Günther, M., Hoinkiss, D.C.: gammaSTAR: A framework for the development of dynamic, real-time capable MR sequences. Magn. Reson. Med. **94**(4), 1485–1499 (2025) https://doi.org/10.1002/mrm.30573

[10] Layton, K.J., Kroboth, S., Jia, F., Littin, S., Yu, H., Leupold, J., Nielsen, J.-F., Stöcker, T., Zaitsev, M.: Pulseq: A rapid and hardware-independent pulse sequence prototyping framework. Magn. Reson. Med. **77**(4), 1544–1552 (2017) https://doi.org/10.1002/mrm.26235

[11] Ravi, K.S., Geethanath, S., Vaughan, J.T.: PyPulseq: A Python Package for MRI Pulse Sequence Design. Journal of Open Source Software **4**(42), 1725 (2019) https://doi.org/10.21105/joss.01725

[12] Tönnes, C., Licht, C., Schad, L.R., Zöllner, F.G.: VirtMRI: a tool for teaching MRI. Journal of Medical Systems **47**(1), 110 (2023) https://doi.org/10.1007/s10916-023-02004-4

[13] Stöcker, T., Vahedipour, K., Pflugfelder, D., Shah, N.: High-performance computing MRI simulations. Magn. Reson. Med. **64**(1), 186–193 (2010) https://doi.org/10.1002/mrm.22406

[14] Liu, F., Kijowski, R., Block, W.: MRiLab: performing fast 3D parallel MRI numerical simulation on a simple PC. In: 2013 ISMRM & ISMRT Annual Meeting & Exhibition, vol. 2072. Salt Lake City, Utah, USA (2013). https://ismrm.gitlab.io/2013/2373.html

[15] Liu, F., Velikina, J., Block, W., Kijowski, R., Samsonov, A.: Fast realistic MRI Simulations based on Generalized Multi-Pool Exchange Tissue Model. IEEE Transactions on Medical Imaging **36**(2), 527–537 (2016) https://doi.org/10.1109/TMI.2016.2620961

[16] Weine, J., McGrath, C., Dirix, P., Buoso, S., Kozerke, S.: CMRsim–A python package for cardiovascular MR simulations incorporating complex motion and flow. Magn. Reson. Med. **91**(6), 2621–2637 (2024) https://doi.org/10.1002/mrm.30010

[17] Kose, R., Kose, K.: Blochsolver: A GPU-optimized fast 3D MRI simulator for experimentally compatible pulse sequences. Journal of Magnetic Resonance **281**, 51–65 (2017) https://doi.org/10.1016/j.jmr.2017.05.007

[18] Xanthis, C., Venetis, I., Chalkias, A., Aletras, A.: MRISIMUL: a GPU-based parallel approach to MRI simulations. IEEE Transactions on Medical Imaging **33**(3), 607–617 (2014) https://doi.org/10.1109/tmi.2013.2292119

[19] Xanthis, C., Venetis, I., Chalkias, A., Aletras, A.: High performance MRI simulations of motion on multi-GPU systems. Journal of Cardiovascular Magnetic Resonance **16**(1), 48 (2014) https://doi.org/10.1186/1532-429X-16-48

[20] Xanthis, C., Aletras, A.: coreMRI: A high-performance, publicly available MR simulation platform on the cloud. PLOS ONE **14**(5), 1–26 (2019) https://doi.org/10.1371/journal.pone.0216594

[21] Villacorta-Aylagas, P., Castillo-Passi, C., Kierulf, R., Menchón-Lara, R.M., Rodríguez-Galván, J.R., Sierra-Pallares, J., Irarrazaval, P., Alberola-López, C.: Versatile and Highly Efficient MRI Simulation for Arbitrary Motion in KomaMRI. Magn. Reson. Med. (2025) https://doi.org/10.1002/mrm.70145 . (Early view)

[22] Nurdinova, A., Ruschke, S., Gestrich, M., Stelter, J., Karampinos, D.C.: GPU-accelerated JEMRIS for extensive MRI simulations. Magnetic Resonance Materials in Physics, Biology and Medicine, 1–16 (2025) https://doi.org/10.1007/s10334-025-01281-z

[23] Montin, E., Carluccio, G., Collins, C.M., Lattanzi, R.: CAMRIE - Cloud-Accessible MRI Emulator. In: 2020 ISMRM & ISMRT Annual Meeting & Exhibition. Virtual Conference, p. 1037 (2020). https://archive.ismrm.org/2020/1037.html

[24] Montin, E., Serrallés, J.E.C., Giannakopoulos, I., Artiges, A., Castillo-Passi, C., Lattanzi, R.: A Modular End-to-End Open-Source Software Pipeline to Simulate the Entire MRI Experiment. In: 2025 ISMRM & ISMRT Annual Meeting & Exhibition., Honolulu, Hawaii, USA (2025). https://archive.ismrm.org/2025/0943.html

[25] Haas, A., Rossberg, A., Schuff, D.L., Titzer, B.L., Holman, M., Gohman, D., Wagner, L., Zakai, A., Bastien, J.: Bringing the web up to speed with webassembly. SIGPLAN Not. **52**(6), 185–200 (2017) https://doi.org/10.1145/3140587.3062363

[26] Liang, Z., Lauterbur, P.C.: Principles of Magnetic Resonance Imaging: A Signal Processing Perspective, 1st edn. IEEE Press Series on Biomedical Engineering. Wiley–IEEE Press, New York, NY (1999)

[27] Bernstein, M.A., King, K.F., Zhou, X.J.: Handbook of MRI Pulse Sequences. Elsevier, Burlington, M.A., USA (2004)

[28] Saloner, D.: The AAPM/RSNA physics tutorial for residents. An introduction to MR angiography. RadioGraphics **15**(2), 453–465 (1995) https://doi.org/10.1148/radiographics.15.2.7761648 . PMID: 7761648

[29] Villacorta-Aylagas, P., Castillo-Passi, C., Irarrázaval, P., Simmross-Wattenberg, F., Rodríguez-Cayetano, M., Alberola-López, C.: A Free and Open-Source Web Application for Pulse Sequence Development and Simulation . In: ISMRM Iberian Chapter Annual Meeting 2025. ISMRM, Barcelona, Spain (2025)