E-MaRSSim: A Python-based open-source MRI pulse sequence simulation tool

T. Tensen,[1] M. Božić-Iven,[1,2] C. Coletti,[1] S. Weingärtner[1]

[1]Department of Imaging Physics, Delft University of Technology, Delft, The Netherlands

[2]Computer Assisted Clinical Medicine, Heidelberg University, Mannheim, Germany

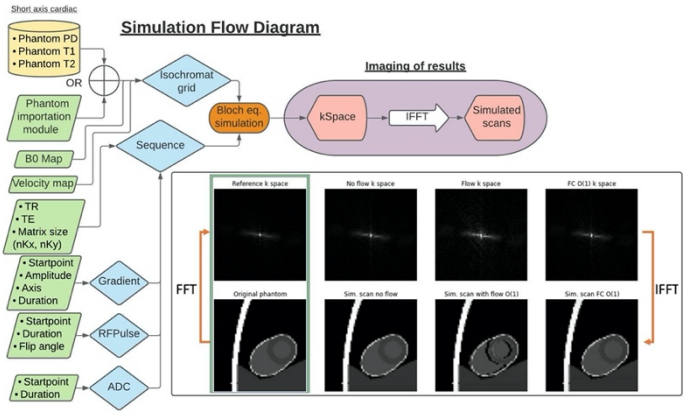

Figure 1: Workflow of the proposed simulation tool with example simulated k-space (top row) and images (bottom row). Ideal phantom is compared with simulated scans without and with blood flow effects..

Introduction:

Numerical simulations are an immediate, flexible and accessible tool for understanding the fundamentals of MRI physics.

Several open-source tools have been proposed recently[1,2,3], which provide comprehensive simulation environments. However, the core of the simulation software is hidden from the final users, who often interact only with the graphical user-interface. With this work, we propose a Python-based, hands-on, fully customizable open-source MRI pulse sequence simulation framework for educational purposes. The aim is to provide an easy-to-use simulation tool, with minimal computational hardware requirements, to facilitate the practical study of MRI basic principles.

Methods:

The simulation framework is based on Python and can be operated across platforms using Google Colab notebooks. The simulation workflow is illustrated in Fig. 1. Voxel wise tissue properties (PD, T_1, T_2) can be imported from custom images, and serve to simulate a grid of isochromats. Other supported inputs are phantom flow velocity and acceleration maps (Fig. 1), as well as a B_0 map. The pulse sequence is defined by calculating the traces for the RF pulses, the gradients G_x, G_y, and G_z, and the ADC. Users can freely adjust the strength, timing and shape of gradient element to implement their custom sequence diagram. Bloch simulations are then performed on the isochromat grids, including relaxation and off-resonance effects. Finally, the fully sampled, simulated image is reconstructed using inverse FFT. For a demonstration of the simulator functionalities, two built-in pulse sequences can be chosen: a gradient-echo and a spin-echo sequence. The phantom has a size of 61x61 pixels and contains one slice with different proton density, T_1 and T_2 values (Fig. 1).The following sequence parameters are used: TE/TR = 11 ms/19 ms, flip angle = 90, Gradient amplitudes: G_x = 0.00385, G_y = 0.00012 and G_z = 0.00164 T/m.

Results:

The framework described above was used to simulate an image acquisition with a gradient-echo sequence on a numerical short-axis cardiac phantom. The simulation time for a phantom of this size (61x61x1) is 9min 33s ± 2min 27s (mean ± std. for 7 runs). Fig. 1 shows the simulated k-space and corresponding reconstructed image, which appears slightly noisy compared to the original phantom. In a second simulation, flow effects in the ventricles are emulated. The k-space and reconstructed image show distinct artifacts, as can be seen in Fig. 1. Lastly, when flow compensation gradients are added to the sequence, the flow related artifacts are reduced in the simulated k-space and image as depicted in Fig. 1.

Discussion:

In this project, we demonstrated an open-source, Python-based framework for MRI sequence simulations. Without flow effects, the simulation yields a k-space and reconstructed image which are consistent with the used numerical phantom. As expected, when flow is simulated, significant image artifacts occur, which visually resemble typical flow artefacts[4]. Incorporating gradient moment nulling in the sequence, however, mitigates flow-related dephasing and significantly improves simulated image quality. Similarly, other interesting MRI phenomena could be simulated using this tool. The tool assumes that R_2'= 0 ($T_2 = T_2^*$). An additional reversible dephasing component will be added in future versions. The tool is presently limited by long simulation times when large matrix sizes and/or high resolutions are used. This warrants further acceleration through the use of graphic processing units (GPU) in future releases.

Conclusion:

We have designed an open-source MRI simulation framework that is user-friendly and fully customizable. The user can apply and adapt already implemented MRI sequences, or easily design new customized ones. Together with the possibility for custom phantoms, this gives rise to a plethora of examples and use-cases. In educational contexts, this framework can help to easily demonstrate and understand the effects of MRI pulse sequences. Code can be retrieved at: https://github.com/MarsLab-TUDelft/E-MaRSSim.

References:

[1]Layton, K. J., et al., MRM (2017).

[2]Ravi, K. S., Geethanath, S., and Vaughan, J.T., JOSS (2019).

[3]imr-framework/py2jemris. Accessed October 8, 2020. https://github.com/imr-framework/py2jemris.

[4]Ferreira, P. F., et al., JCMR (2013).

[5]Bydder, G. M., and Young, I.R., JCAT (1985).

[6]Felmlee, J. P., and Ehman, R.L., Radiology (1987).