Using KomaMRI.jl for Comprehensive Quantitative MRI

Carlos Castillo-Passi✉ , Ronal Coronado, Gabriel Varela-Mattatal,

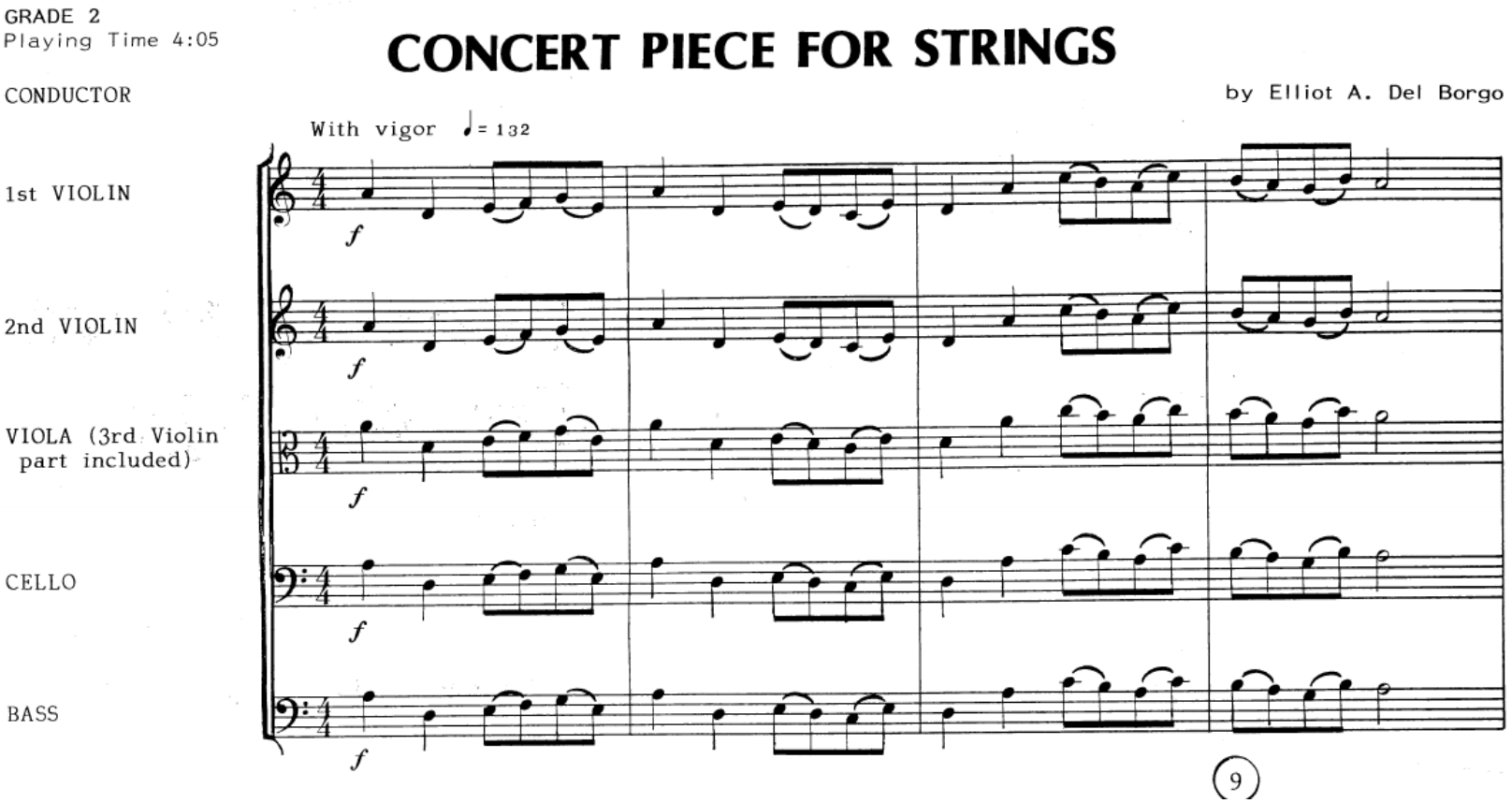
Carlos Alberola-López, René Botnar, and Pablo Irarrazaval

✉ *cncastillo@uc.cl / carlos.castillo_passi@kcl.ac.uk*

Introduction to MRI Simulation

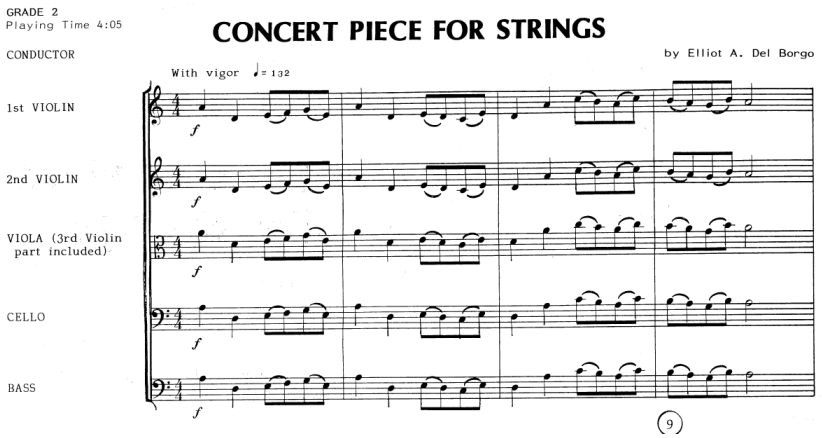
What is MRI?

How does MRI work?



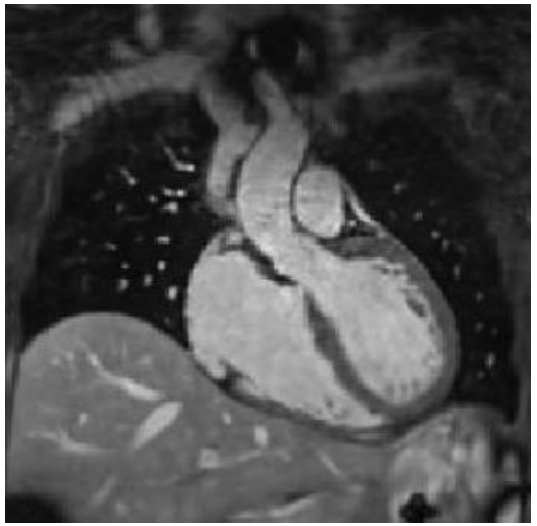
How does MRI work?

Sequence



Scanner

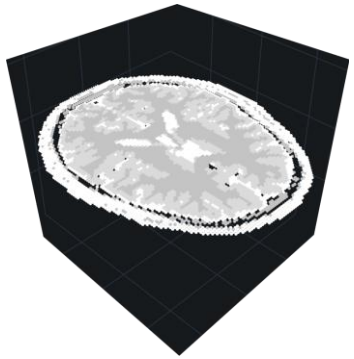
Image



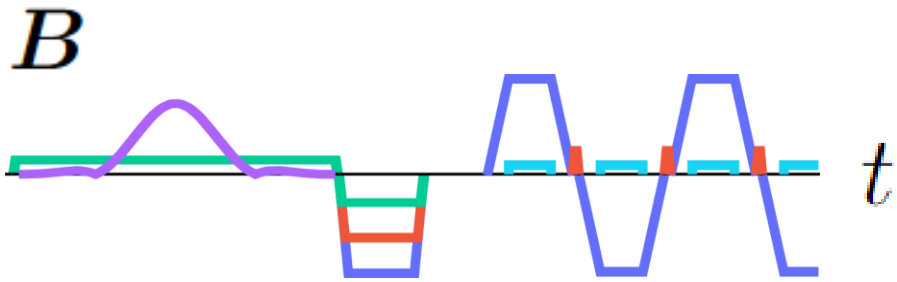
Virtual MRI experiment

Phantom

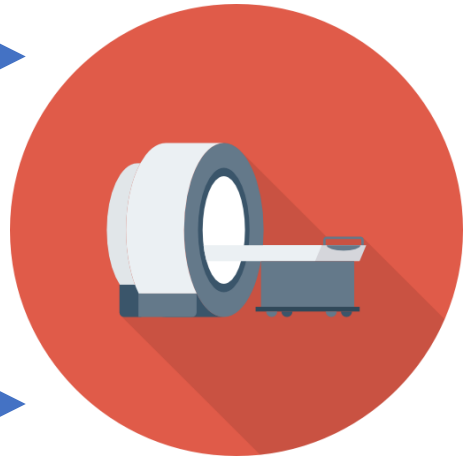
$M_0, T_1, T_2, \ldots$



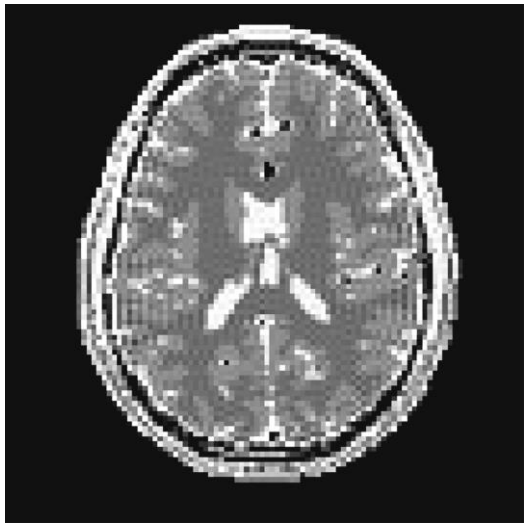
Sequence



Scanner



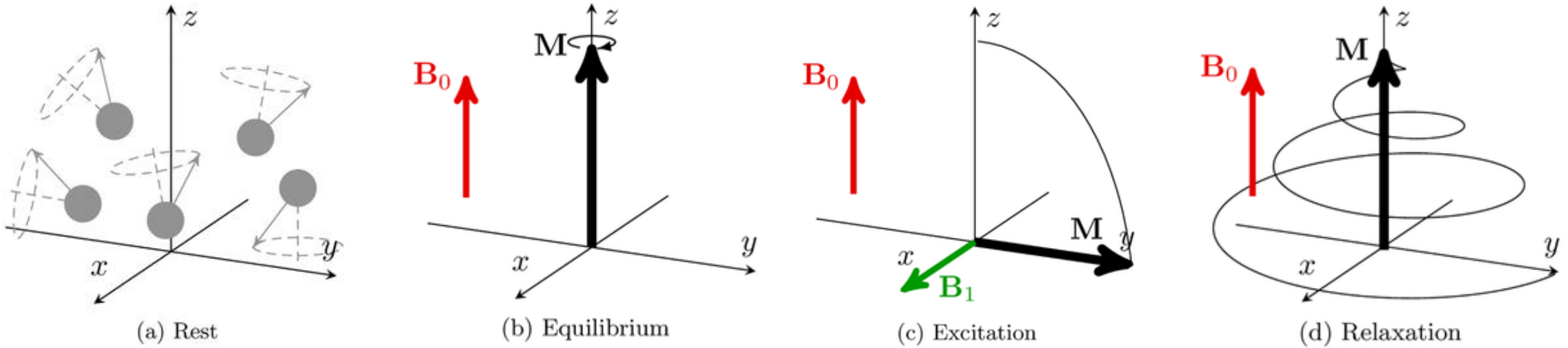
Image



We have indirect measurements of the object

Bloch equation:

$$\frac{\mathrm{d}\boldsymbol{M}}{\mathrm{d}t} = \gamma \boldsymbol{M} \times \boldsymbol{B} + \frac{(M_0 - M_z)\hat{\boldsymbol{z}}}{T_1} - \frac{M_x\hat{\boldsymbol{x}} + M_y\hat{\boldsymbol{y}}}{T_2}$$

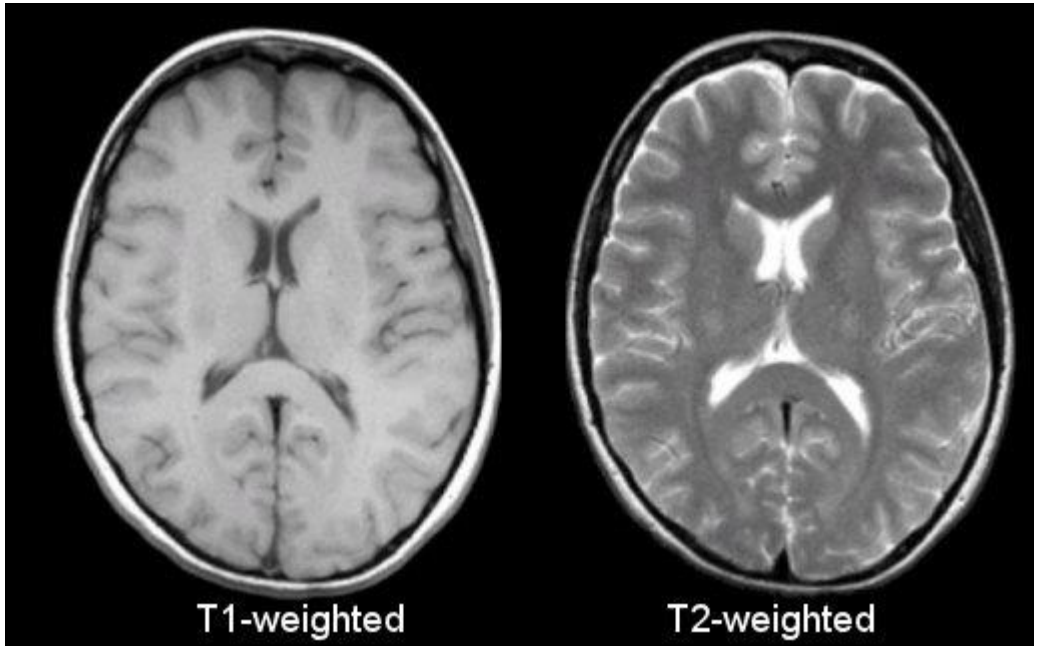


(a) Rest (b) Equilibrium (c) Excitation (d) Relaxation

Difference in magnetic prop. generate contrast

$$\frac{\mathrm{d}\boldsymbol{M}}{\mathrm{d}t} = \gamma \boldsymbol{M} \times \boldsymbol{B} + \frac{(M_0 - M_z)\hat{\boldsymbol{z}}}{T_1} - \frac{M_x\hat{\boldsymbol{x}} + M_y\hat{\boldsymbol{y}}}{T_2}$$

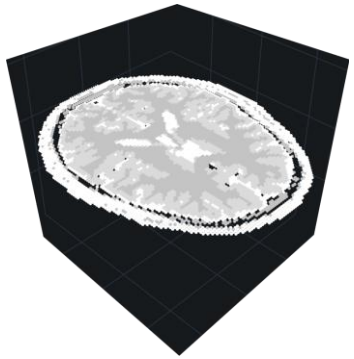
Sequence #1 **Sequence #2**



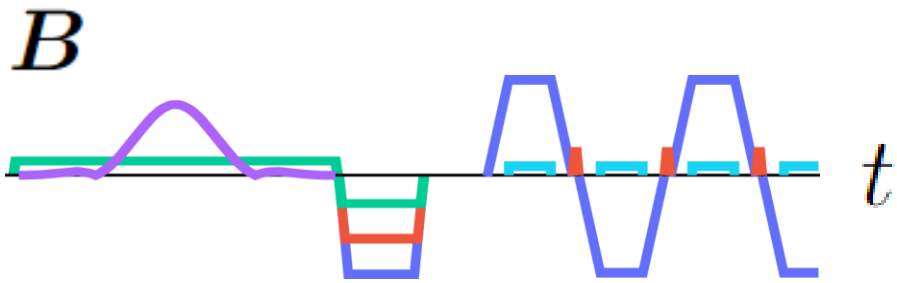
Simulation

Phantom

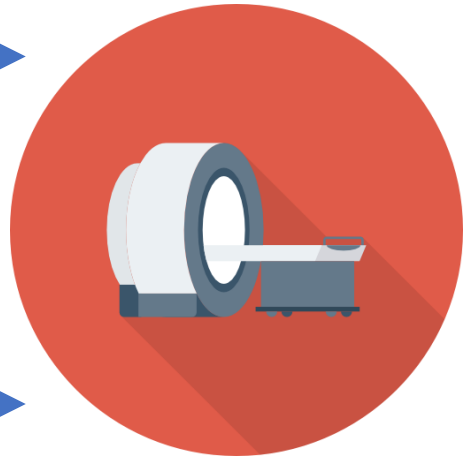
$M_0, T_1, T_2, \ldots$



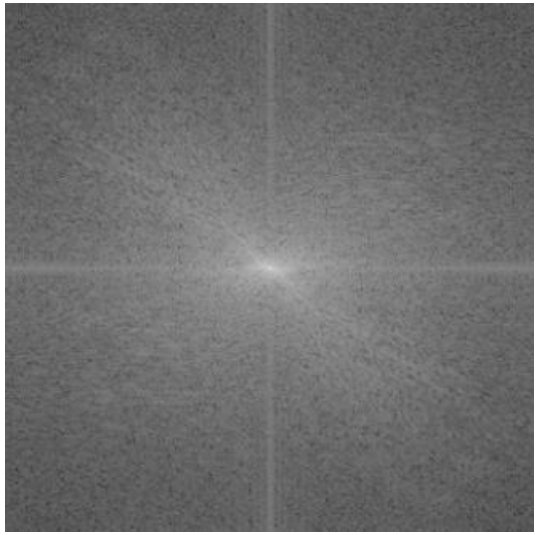
Sequence



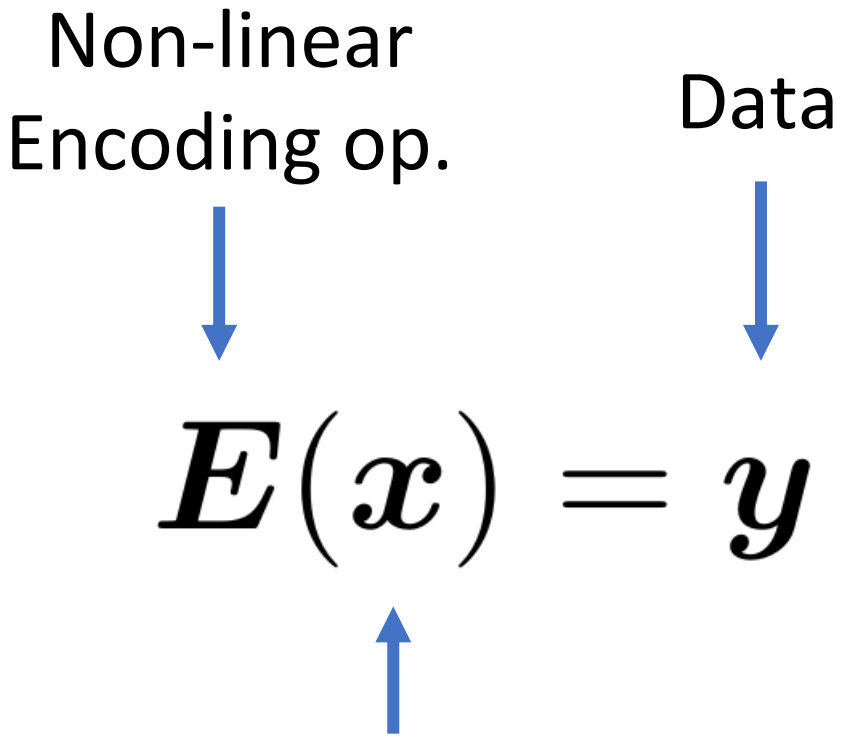
Scanner



Data

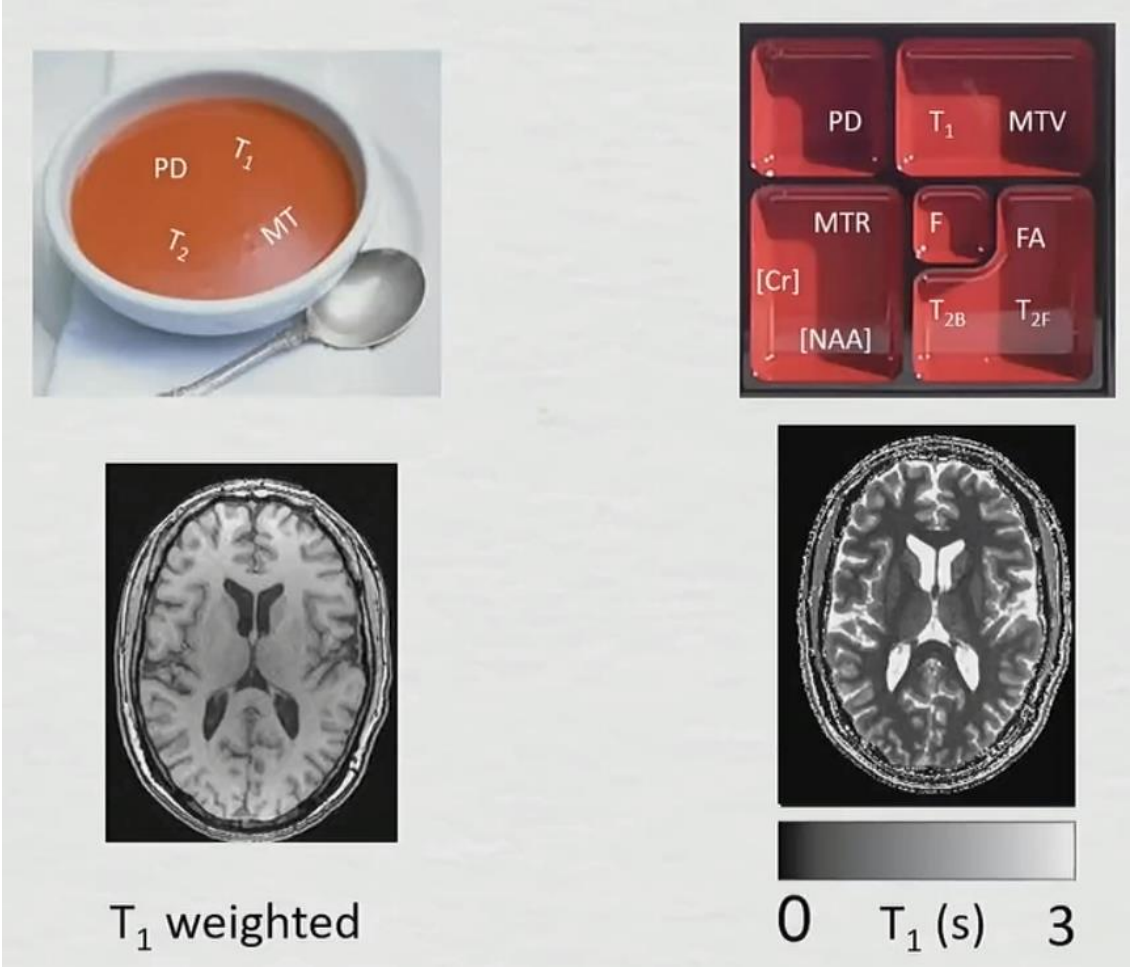


Simulation: Forward problem



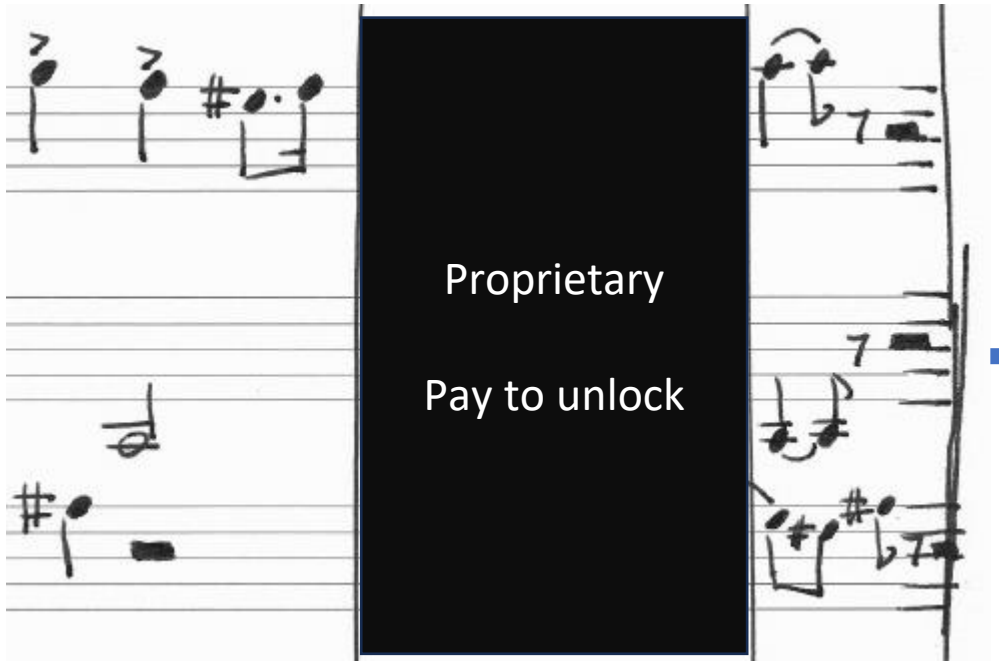
Two types of approach: weighted and quantitative

! The MAGNETOM is not a measurement device as defined by the medical product guidelines. Measured values obtained are for informational purposes and cannot be used only as the basis for diagnosis.



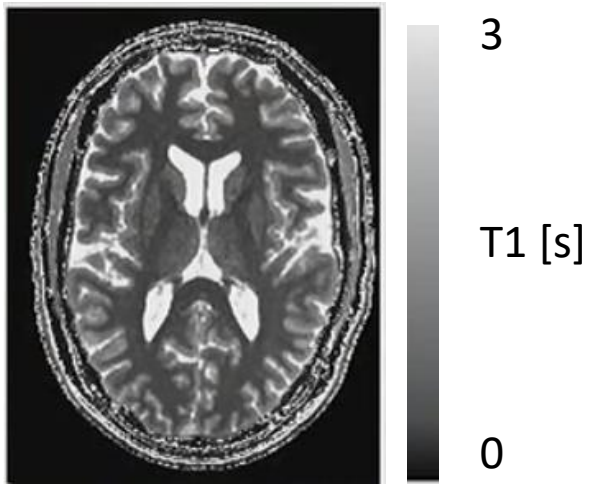
… How do we measure them?

Sequence

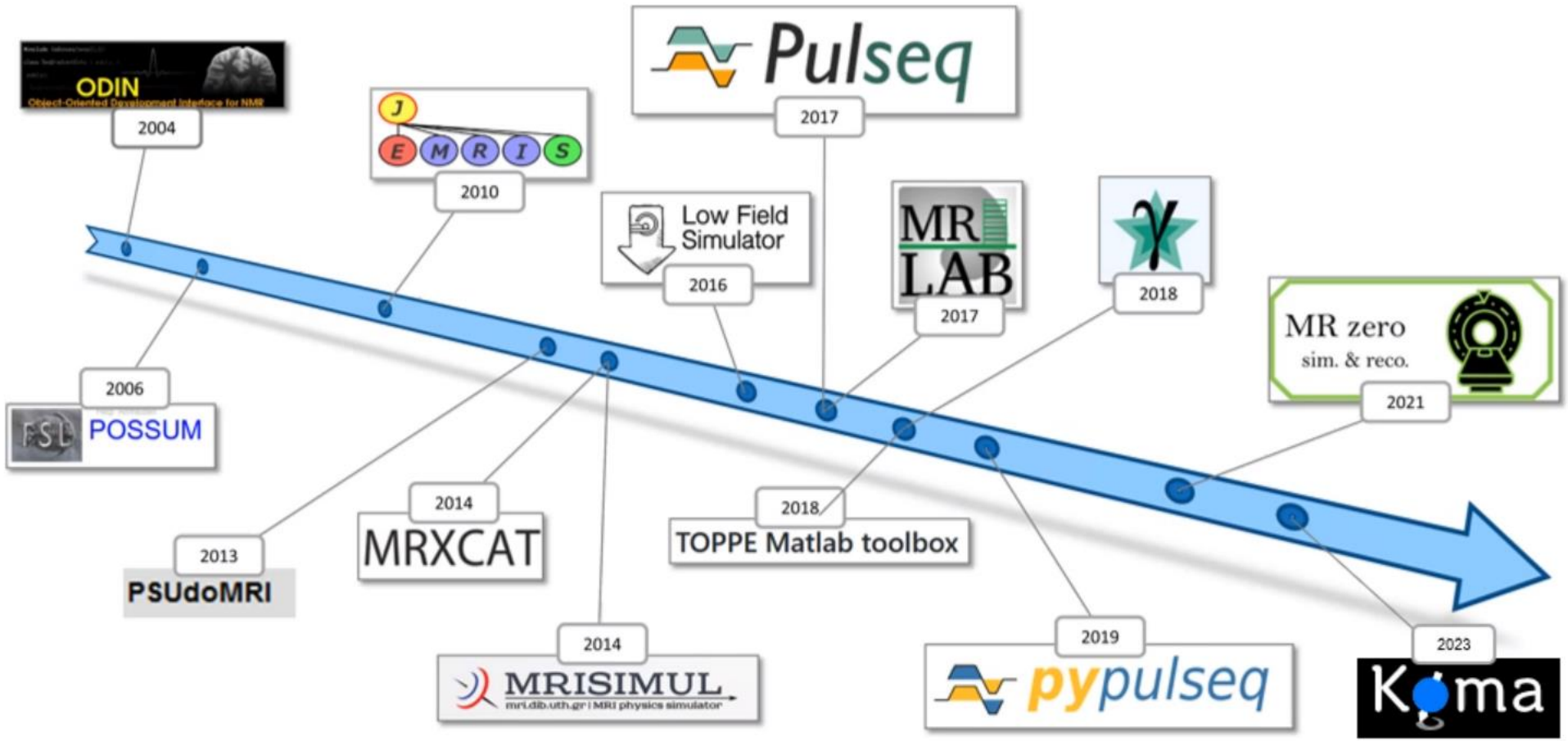


Scanner

Image or Quantitative map



Another MRI simulator … why?



KomaMRI.jl

KomaMRI.jl: Design goals

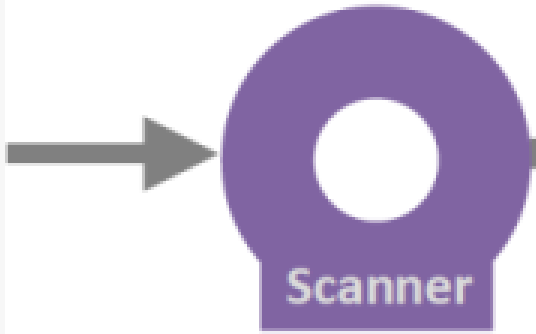
- Open-source
- High-performance
- Easy-to-use
- Extensible
- Cross-platform
- General sequences

Pulseq: A simple and open sequence definition

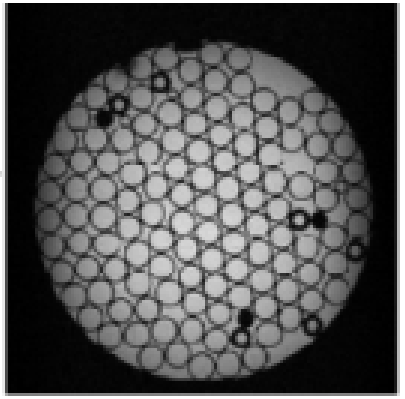
Pulseq

MATLAB

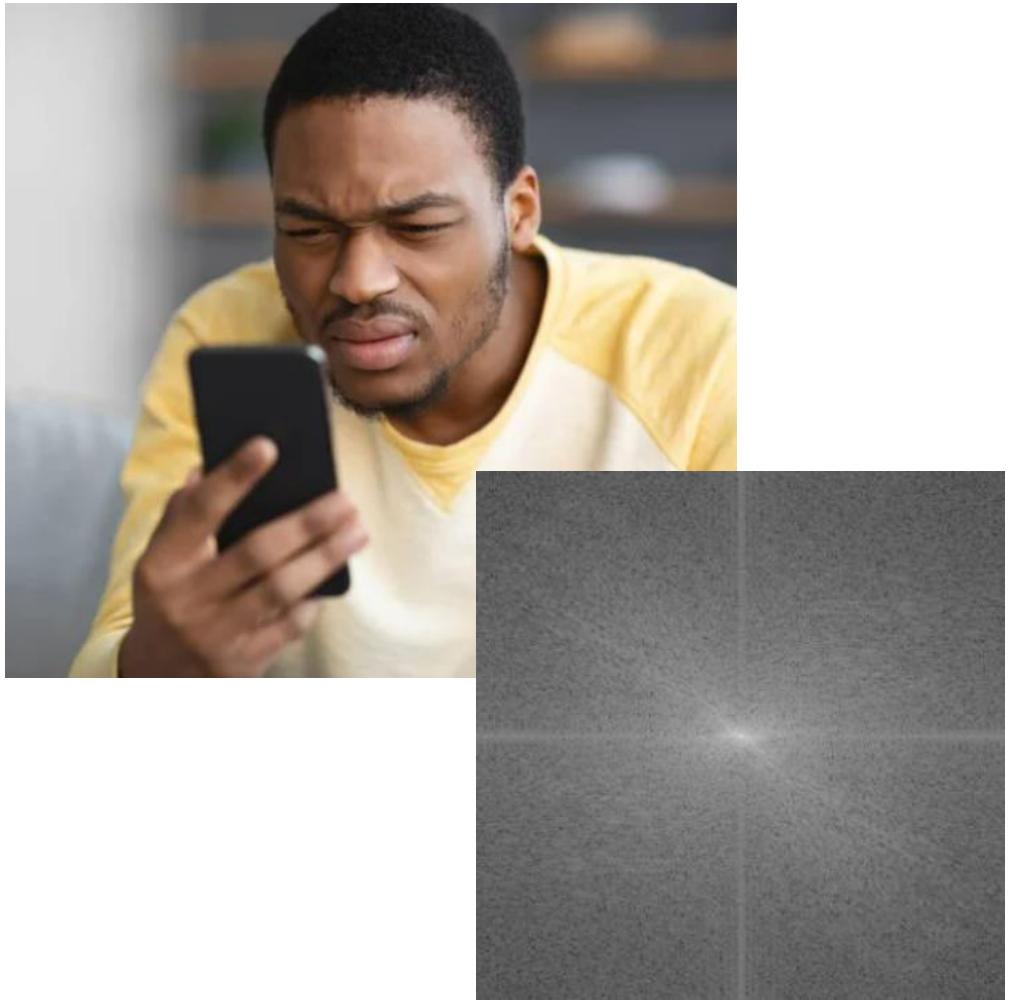
```
for i = 1:Ny
    seq.addBlock(rf, gz);
    gyPre = mr.makeTrapezoid('y', 'Area', phase);
    seq.addBlock(gxPre, gyPre, gzReph);
    seq.addBlock(gx, adc);
    seq.addBlock(delay2);
end
```



Experiment



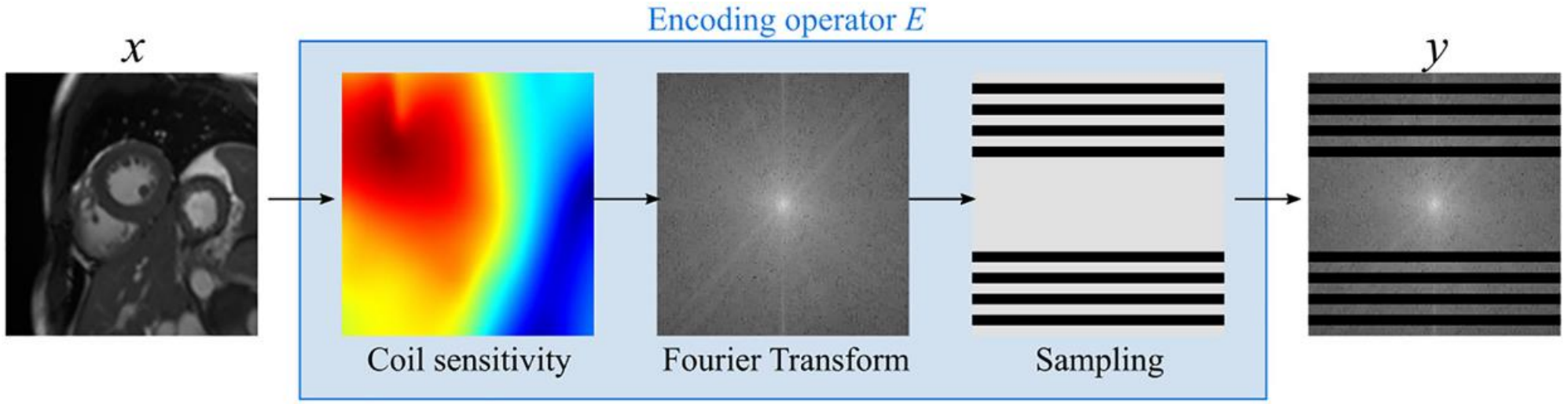
The importance of the “I” in MRI



Reconstruction: How is generally done

A **simplification** of the MRI system is used
to enable **rapid reconstruction**

$$\boldsymbol{E}(\boldsymbol{x}) \approx \mathbf{E}\,\boldsymbol{x} \Rightarrow \min_{\boldsymbol{x}} \frac{1}{2}\|\mathbf{E}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda R(\boldsymbol{x})$$



The answer?

Fast

Julia was designed for high performance. Julia programs automatically compile to efficient native code via LLVM, and support multiple platforms.

Dynamic

Julia is dynamically typed, feels like a scripting language, and has good support for interactive use, but can also optionally be separately compiled.

Reproducible

Reproducible environments make it possible to recreate the same Julia environment every time, across platforms, with pre-built binaries.

Composable

Julia uses multiple dispatch as a paradigm, making it easy to express many object-oriented and functional programming patterns. The talk on the Unreasonable Effectiveness of Multiple Dispatch explains why it works so well.

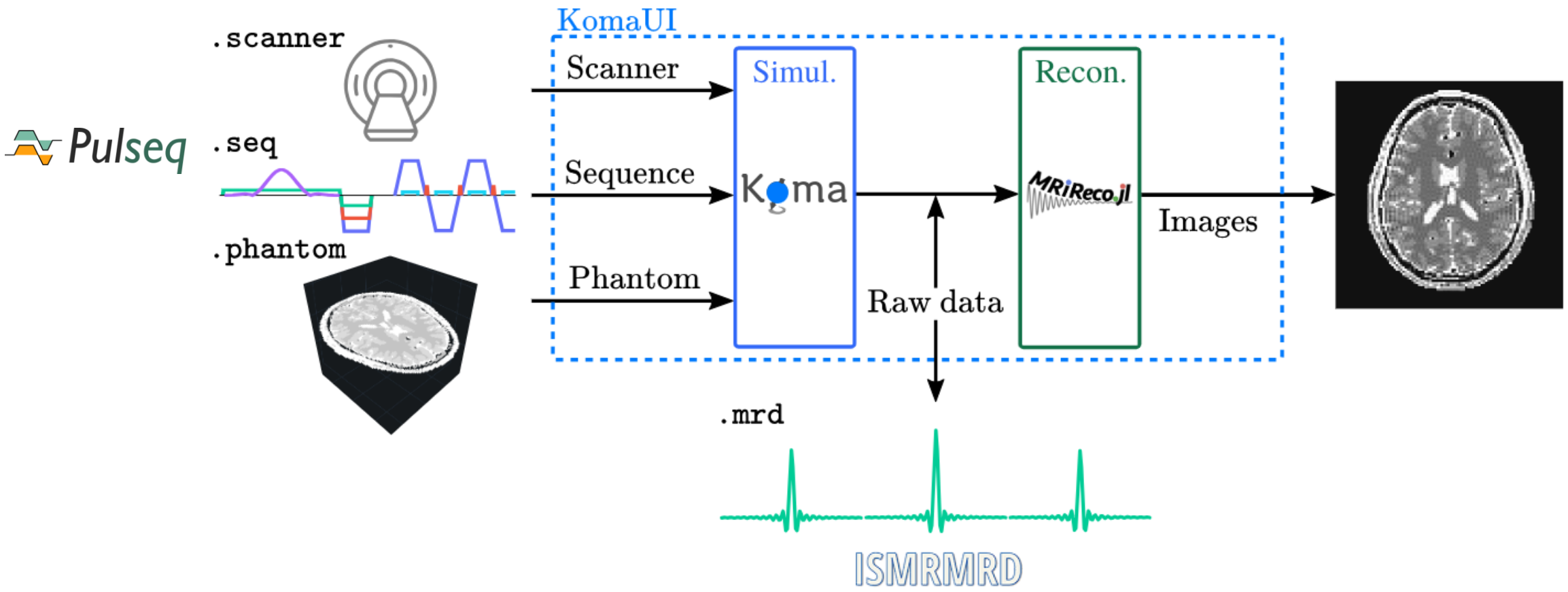
General

Julia provides asynchronous I/O, metaprogramming, debugging, logging, profiling, a package manager, and more. One can build entire Applications and Microservices in Julia.

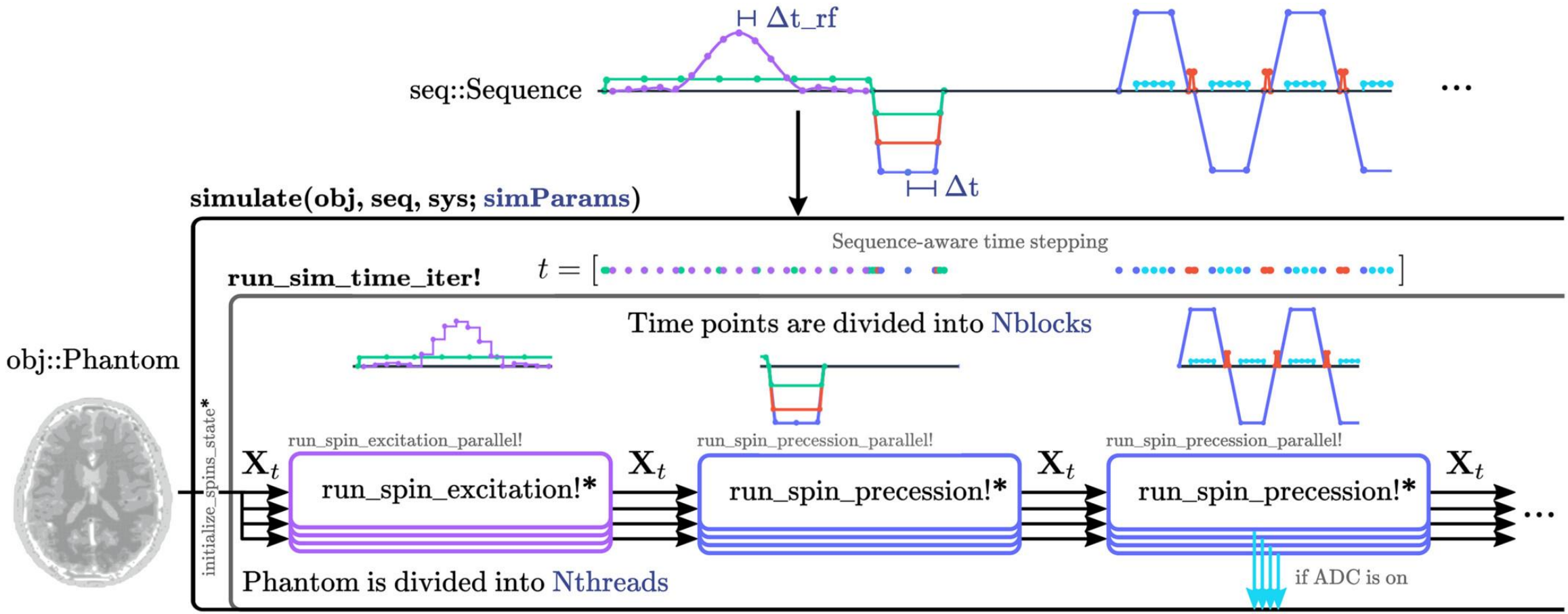
Open source

Julia is an open source project with over 1,000 contributors. It is made available under the MIT license. The source code is available on GitHub.

Koma uses open-source standards



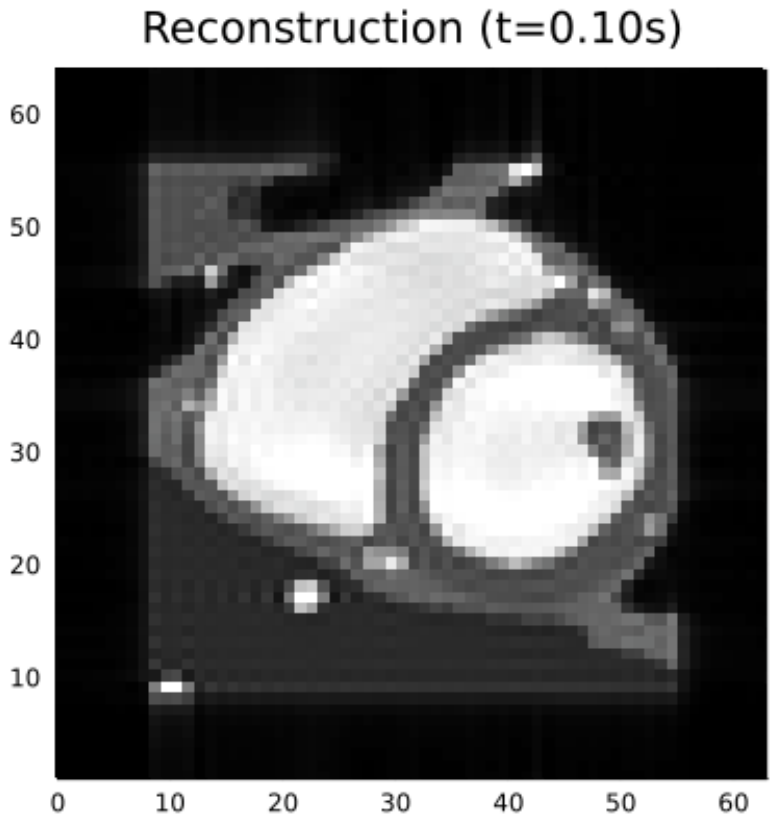
Parallelization and extensibility

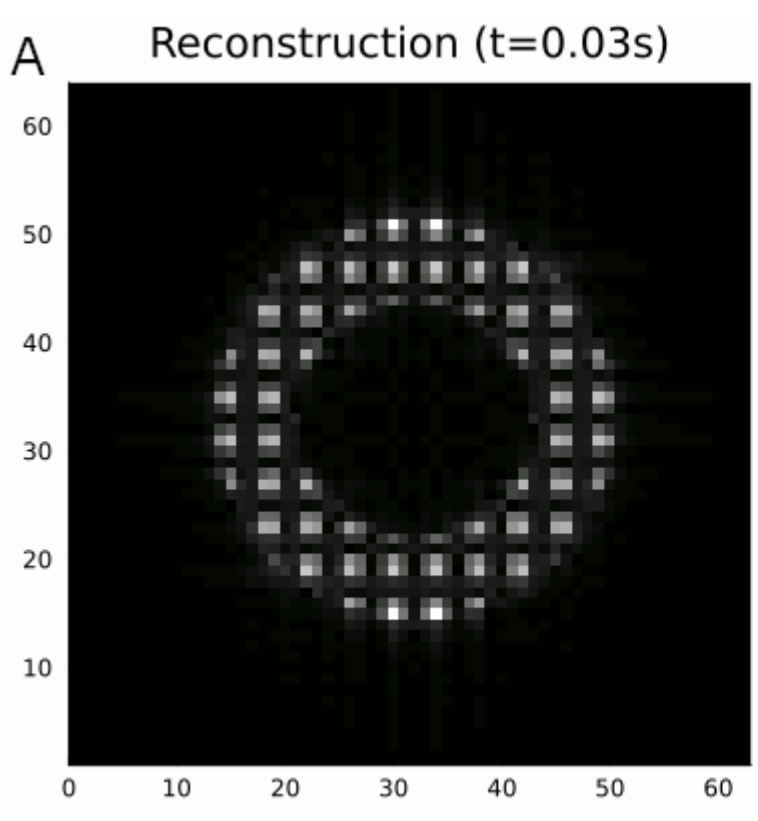


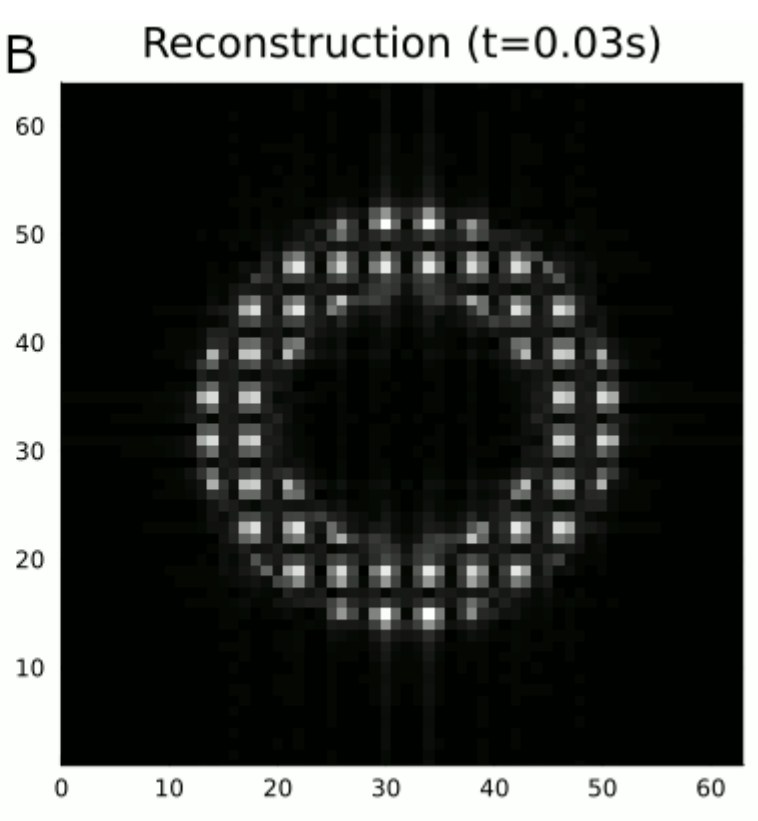
*These functions can be extended by changing the type of **sim_method** and/or **Xt**.
The default types are **Bloch**<:SimulationMethod and **Mag**<:SpinStateRepresentation respectively.

A KomaMRI Phantom Extension for Integrated Dynamic Imaging

Pablo Villacorta Aylagas, Carlos Castillo-Passi, Rosa María Menchón-Lara, Pablo Irarrazaval, Carlos Alberola-López,



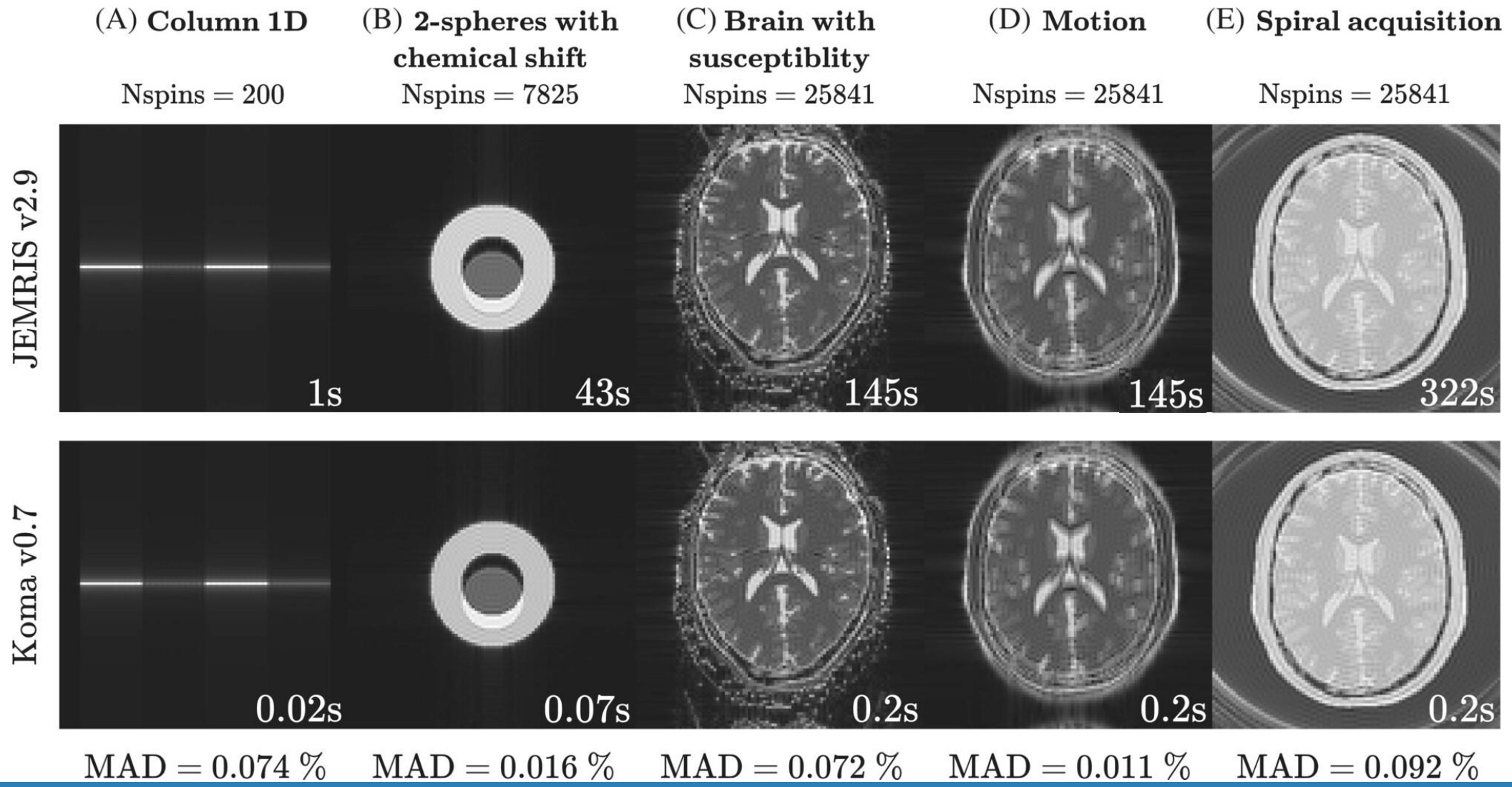




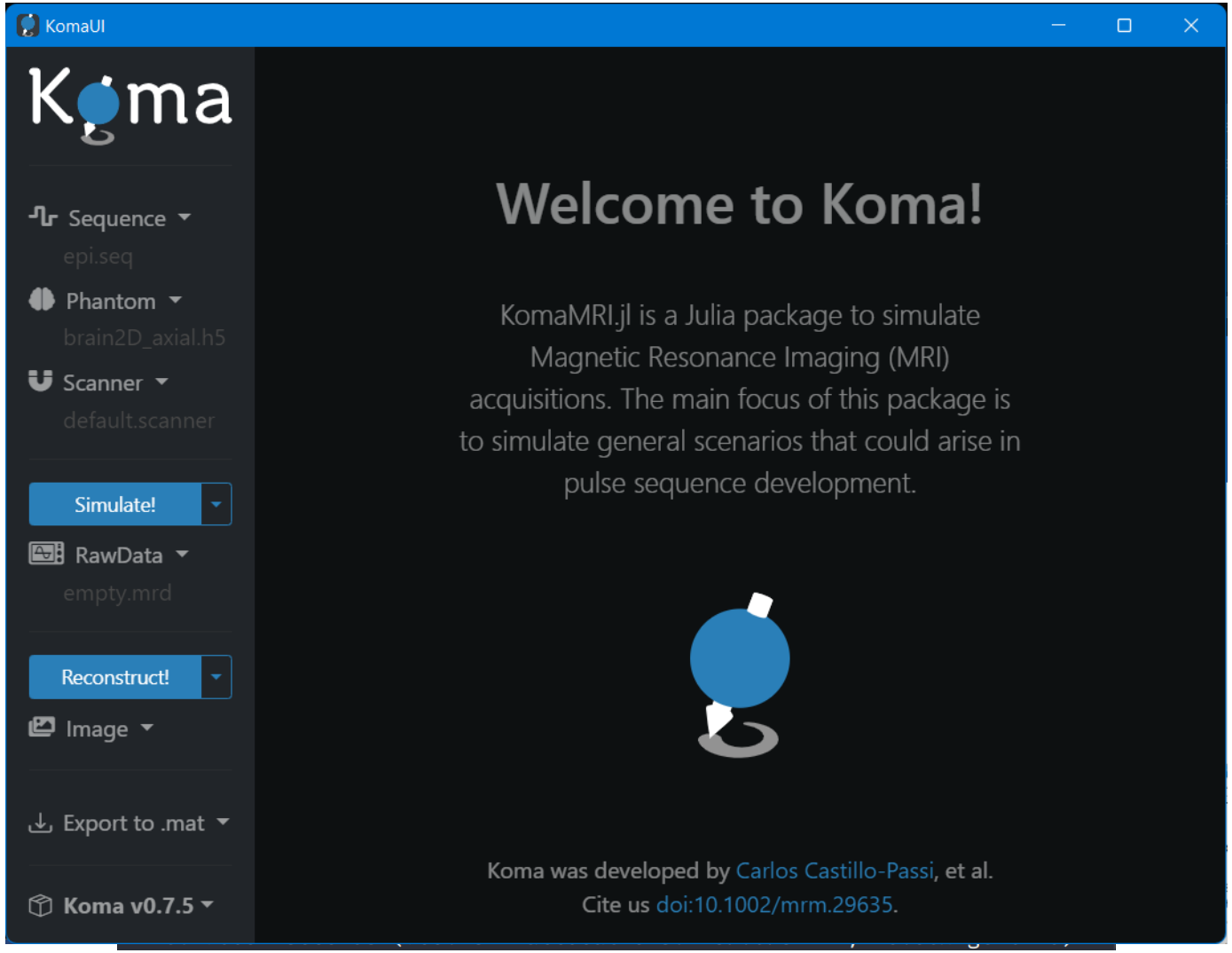
Speed: Comparing simulation speed

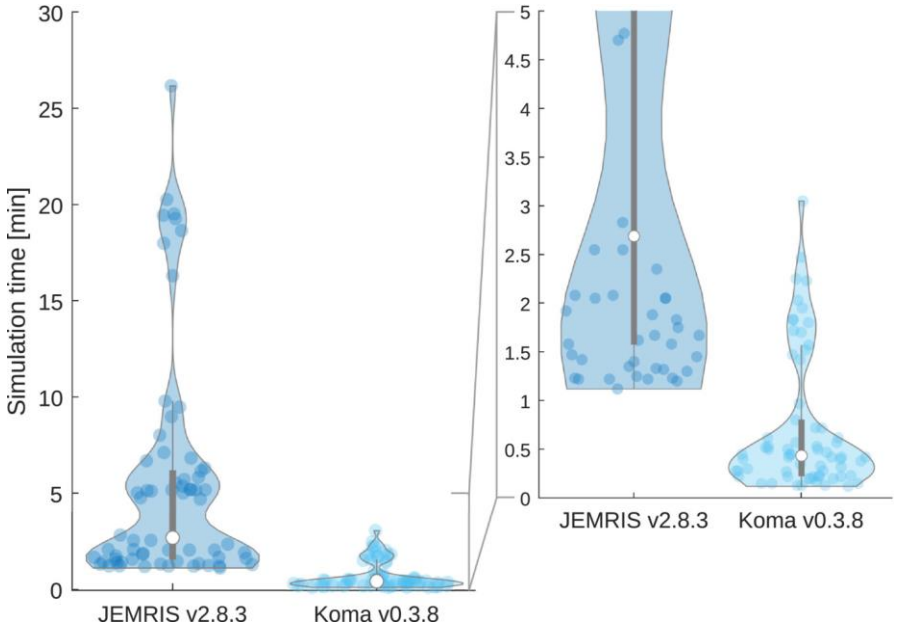
Name	CPU	GPU	
	Intel i7-1165G7	GTX 1650 Ti	RTX 2080 Ti
JEMRIS	≈ 7 min	-	-
MRiLab	**1.56 s ± 0.07 s**	0.84 s ± 0.02 s	0.91 s ± 0.02 s
Koma	1.82 s ± 0.17 s	**0.32 s ± 0.02 s**	**0.15 s ± 0.01 s**

Acc: Comparing simulation accuracy with JEMRIS



Easy-to-use: Students' experience (beta testers)





User experience

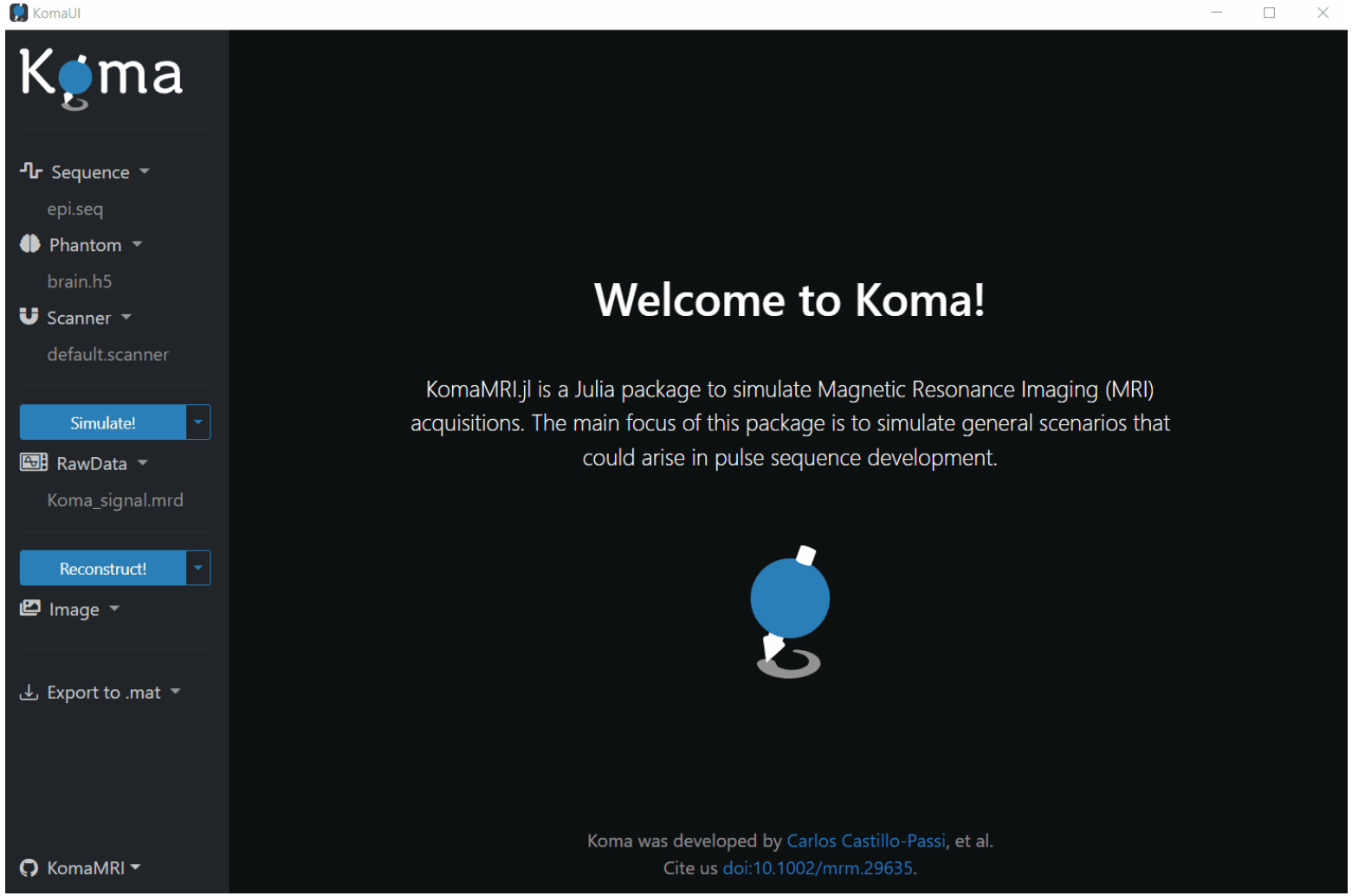
Students reported no problem installing Julia (mean 4.7/5), Koma (mean 4.2/5), JEMRIS (mean 3.8/5), and MRiLab (mean 4.3/5). Regarding the time taken to install each simulator, most of the students were able to install Koma (mean 13.2 min), JEMRIS (mean 33.8 min), and MRiLab (mean 16.9 min) in less than 40 min.

Their first simulation took them more time in JEMRIS (mean 19 min) and MRiLab (mean 13.9 min) than in Koma (mean 5.7 min). 31% of the students could not simulate on MRiLab (six students using Mac OS), so we decided to only use Koma and JEMRIS for the rest of the activities.

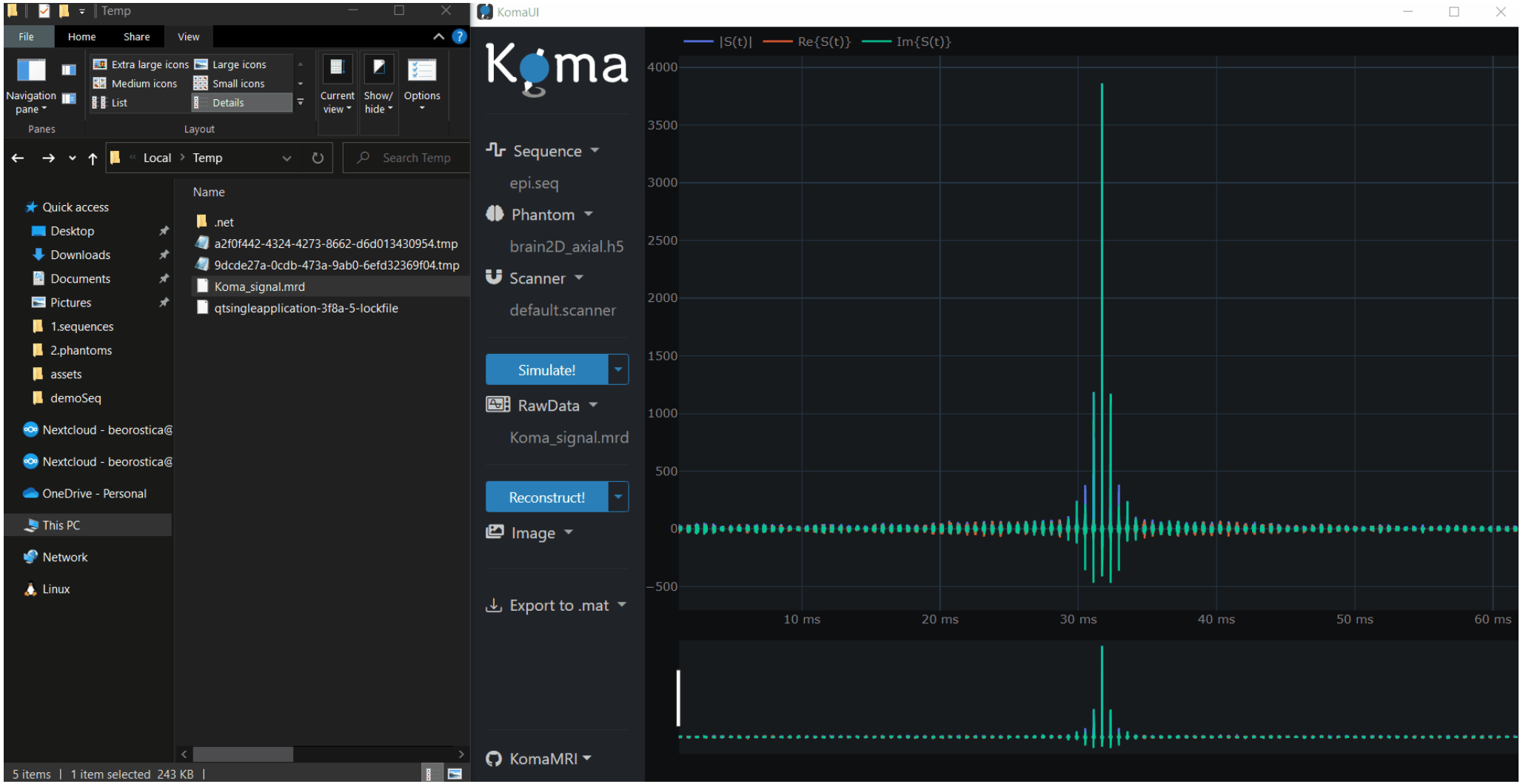
Blink.jl + PlotlyJS.jl powered

You can export your sequence to an interactive HTML! :)

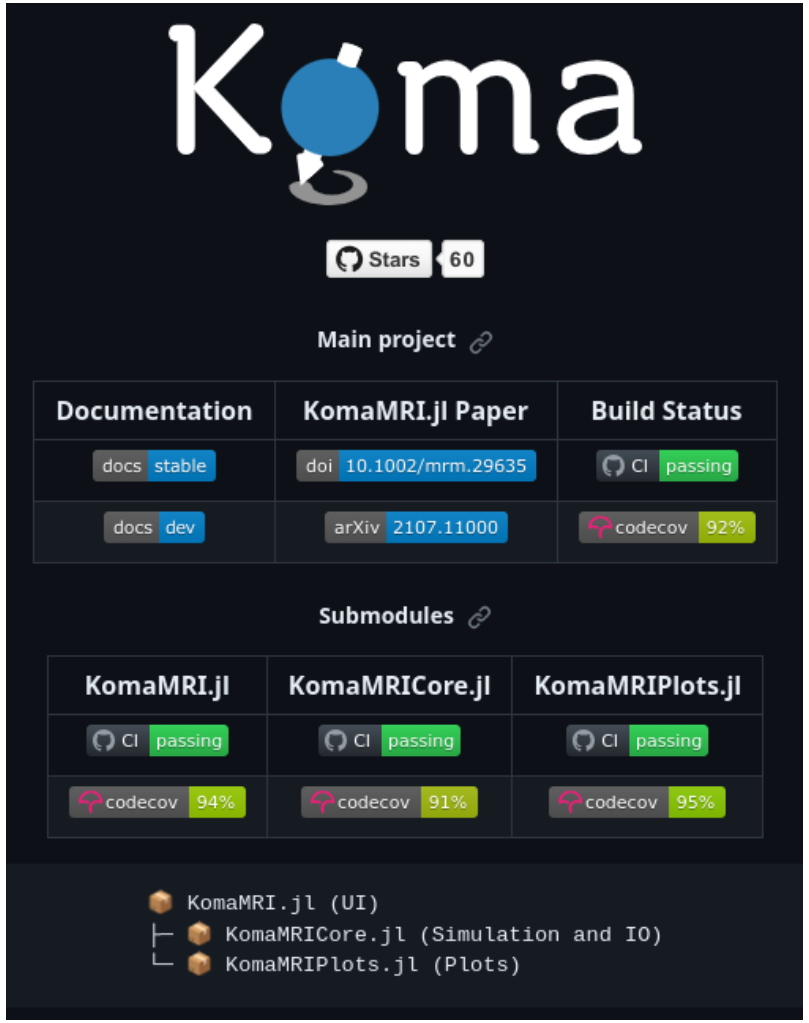
Using KomaMRI's GUI



Export results to .MAT



Koma, an open-source project



Documentation

Search docs

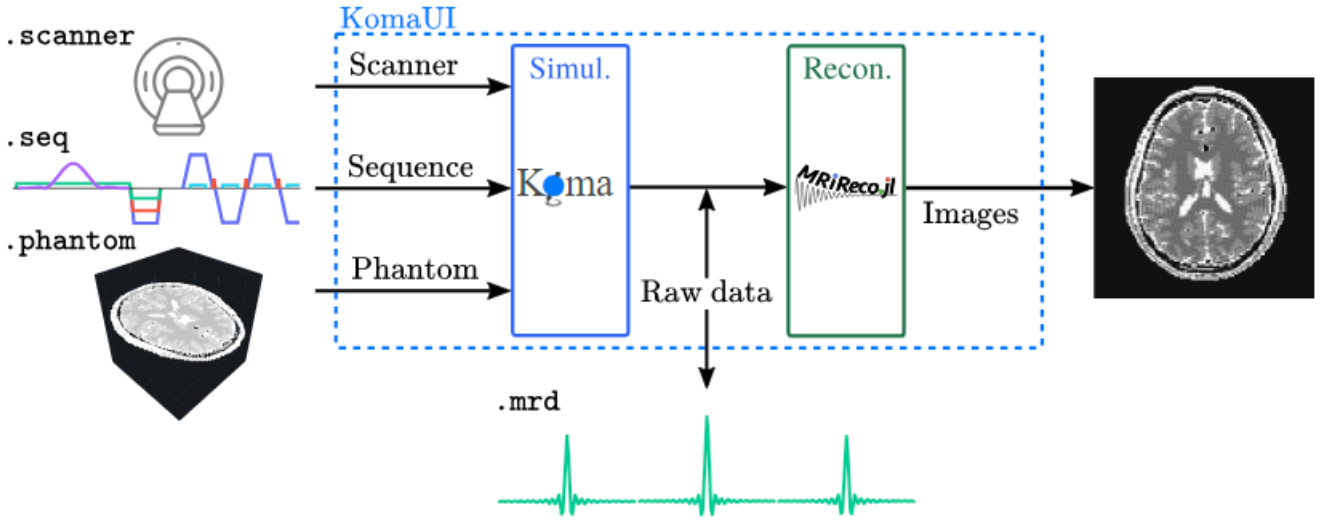
- Home
  - Introduction
  - Features
  - Potential Use Cases
- Getting Started
- Graphical User Interface
- Examples
  - Free Induction Decay
  - Small Tip Angle Approximation
  - Chemical Shift in an EPI sequence
  - Slice Selective Acquisition of 3D Phantom
- Simulation Method
- API Documentation

Home — Edit on GitHub

Introduction

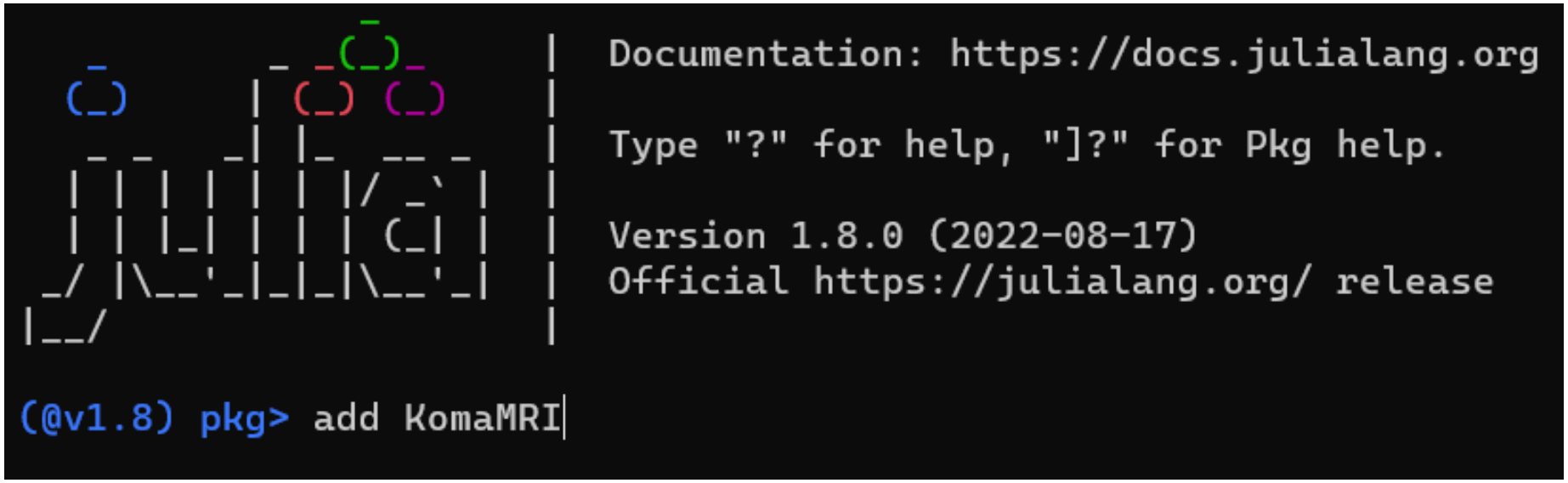
KomaMRI.jl is a Julia package meant to simulate general Magnetic Resonance Imaging (MRI) scenarios. Its name comes from the Japanese word for spinning-top こま (ko-ma) as they precess due to gravity like spins in a magnetic field.

KomaMRI generates **raw data** by solving the **Bloch equations** using the specified **scanner**, **phantom** and **sequence**. It also provides a Graphical User Interface (GUI) that encapsulates the whole imaging pipeline (simulation and reconstruction).



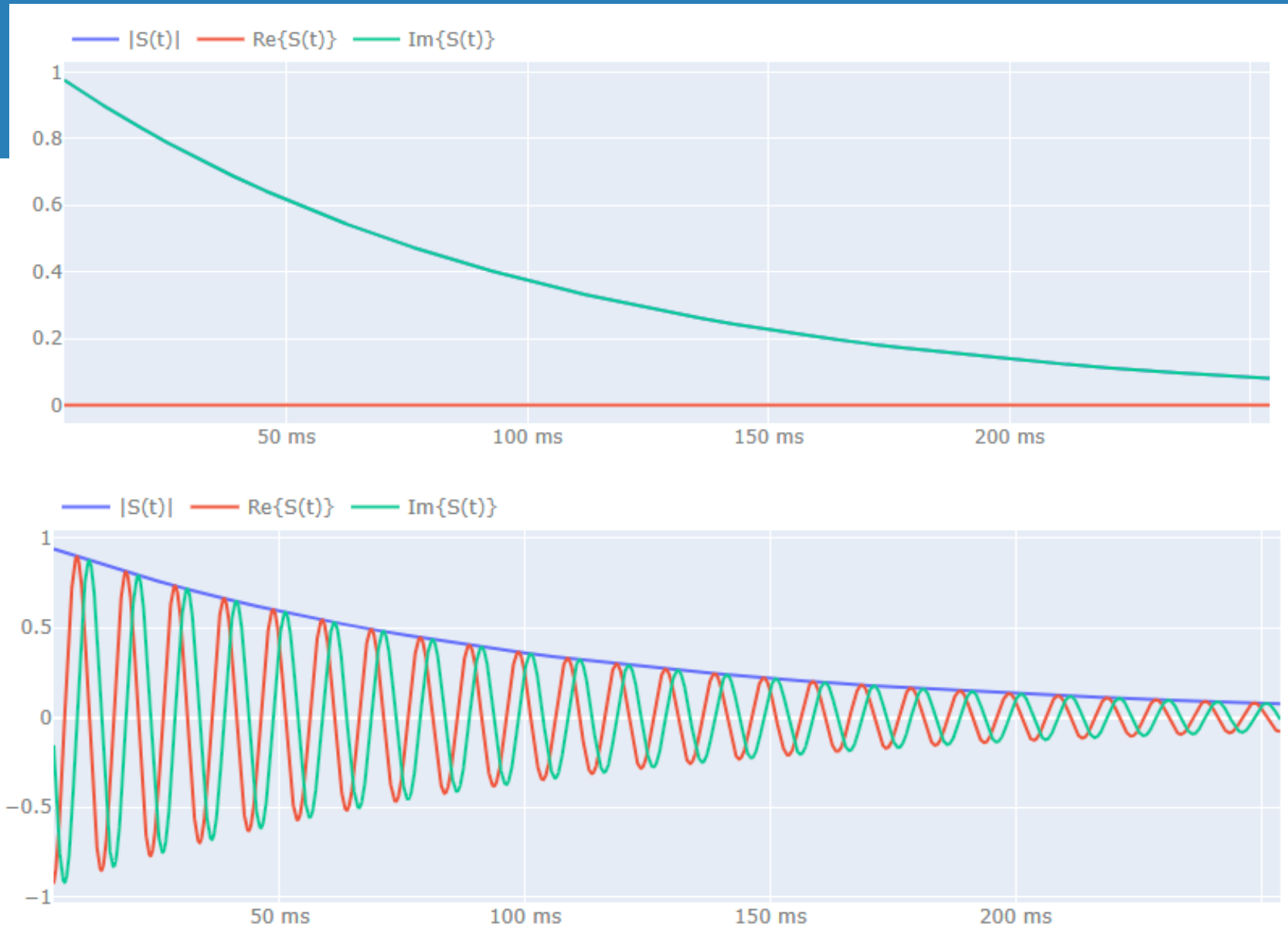
KomaMRI can be used by either:

Easy installation

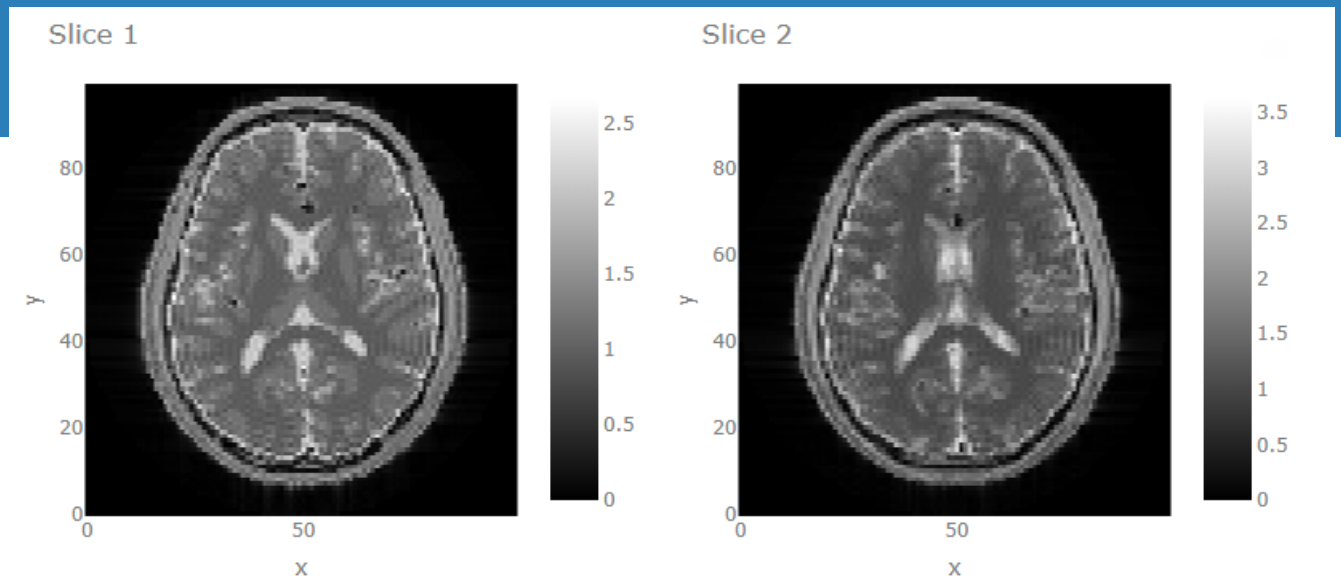


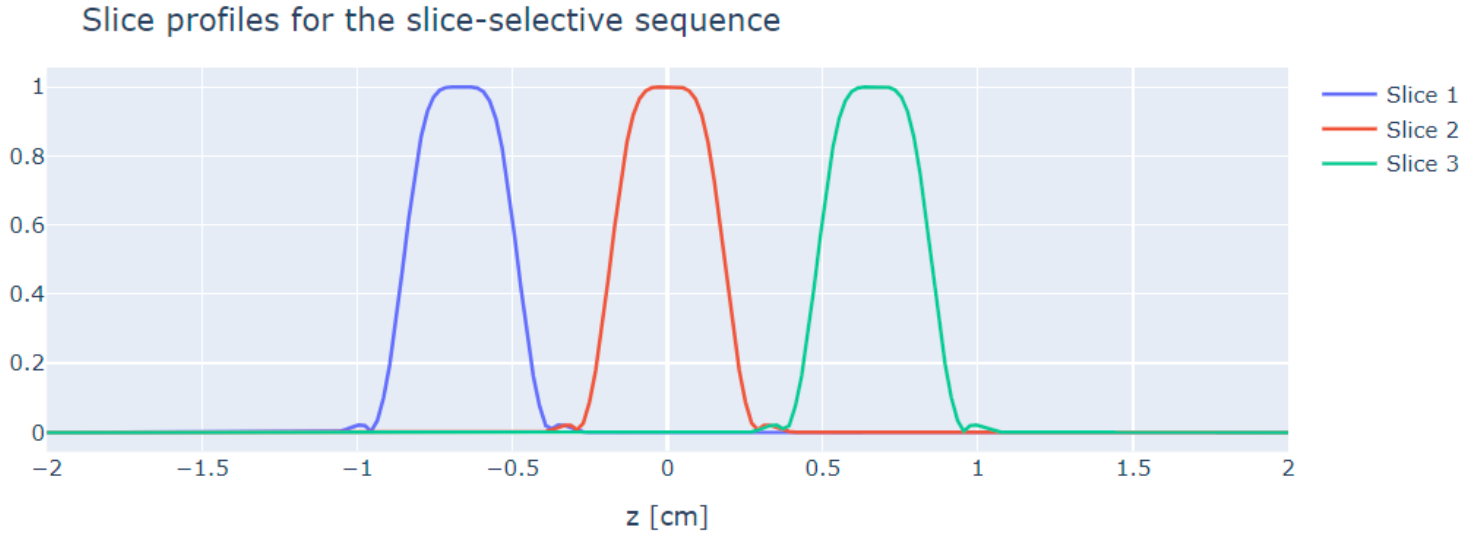
Applications

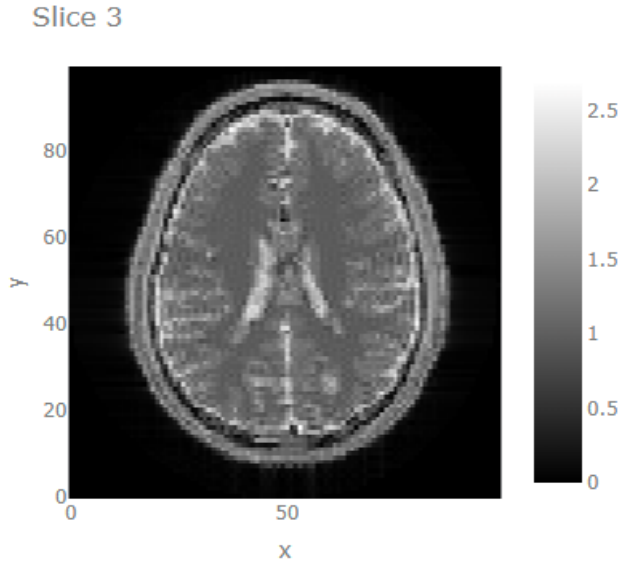
FIDs


|S(t)|
Re{S(t)}
Im{S(t)}
1
0.8
0.6
0.4
0.2
0
50 ms
100 ms
150 ms
200 ms
|S(t)|
Re{S(t)}
Im{S(t)}
1
0.5
0
−0.5
−1
50 ms
100 ms
150 ms
200 ms

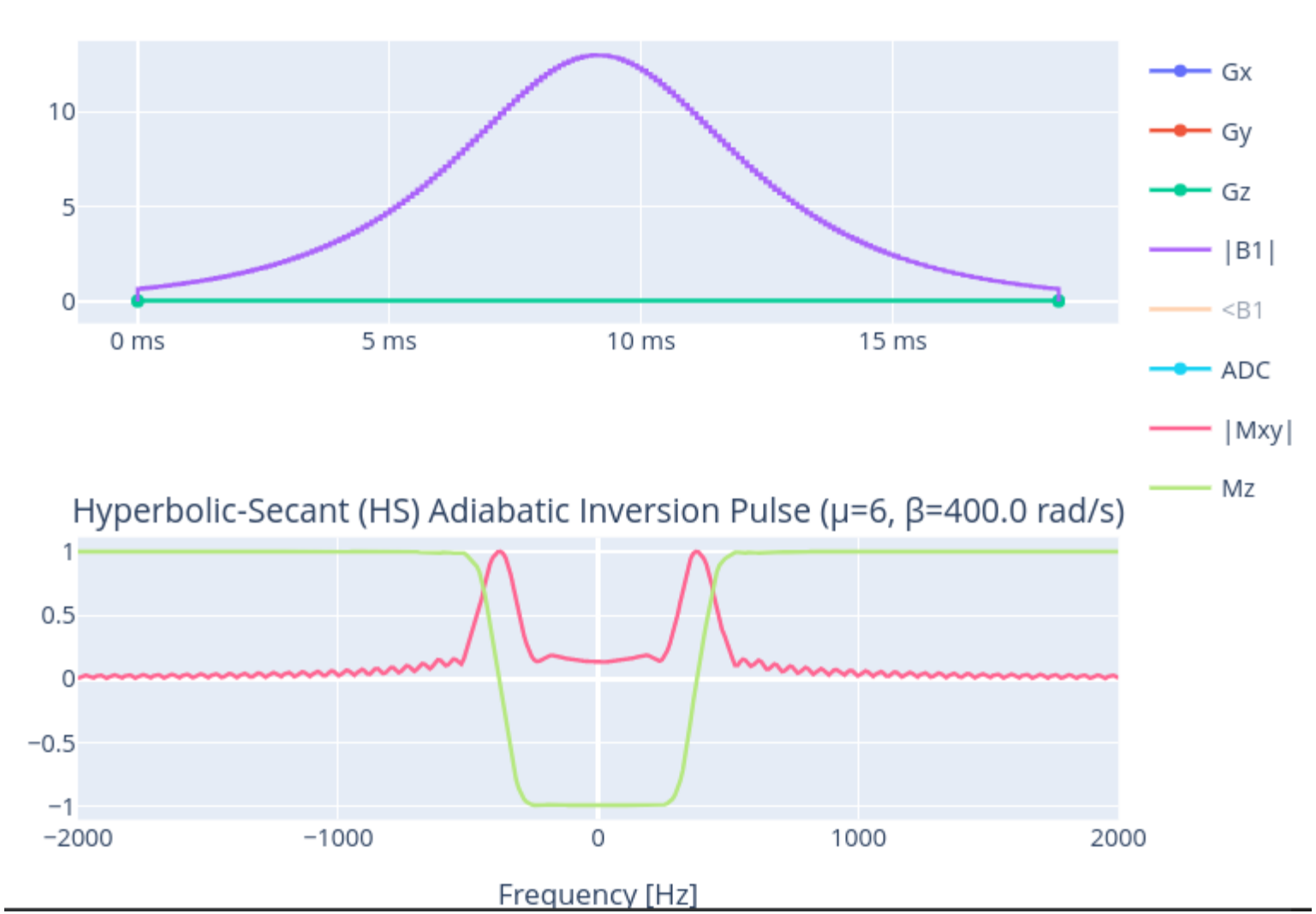
RF Slice Profiles

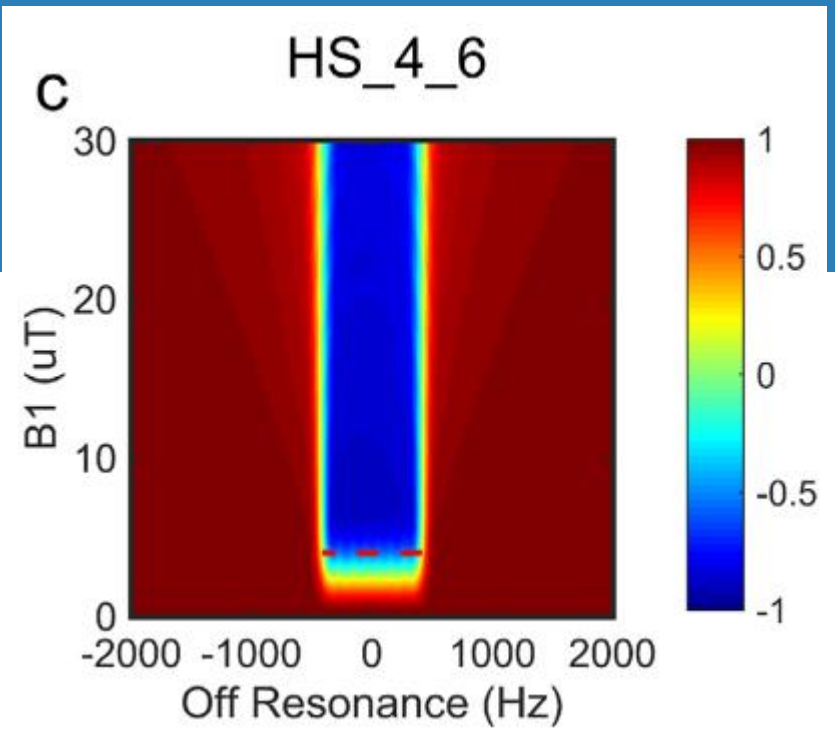


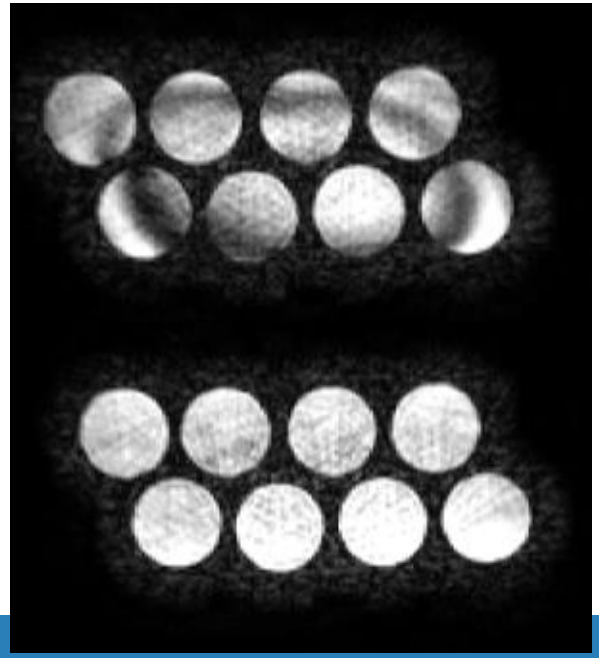




Adiabatic RFs



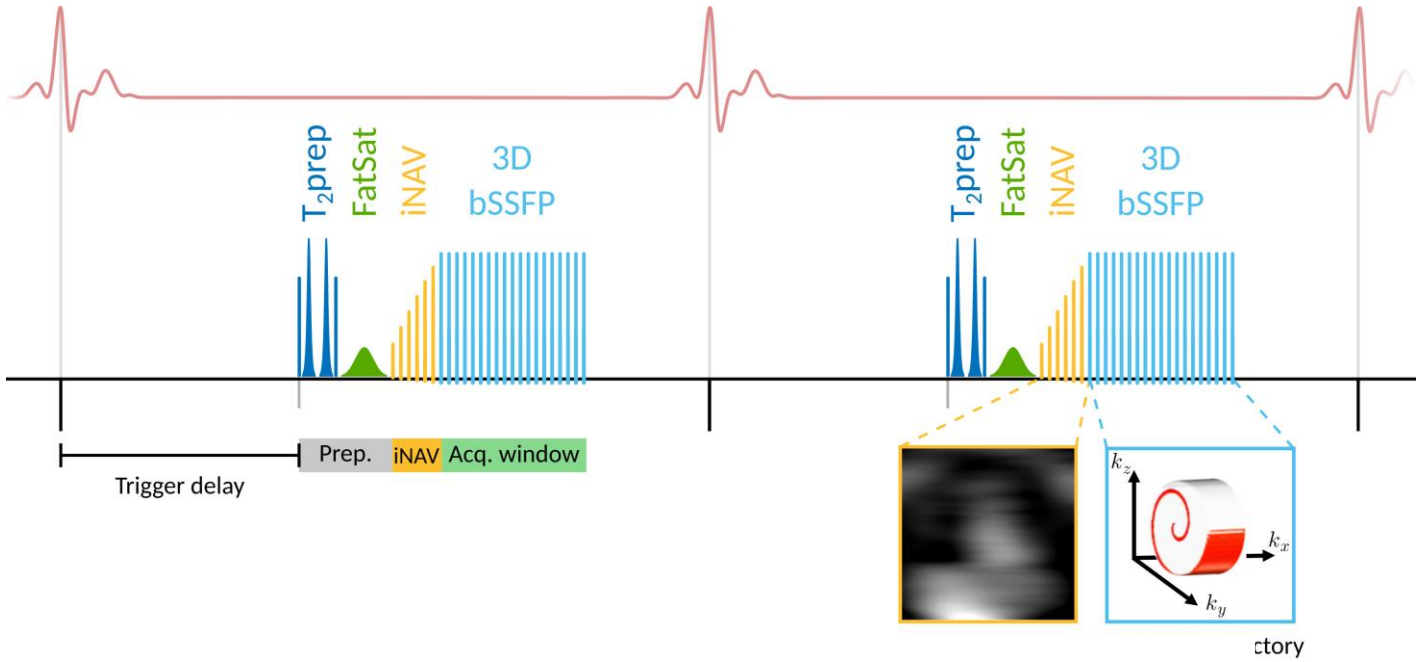


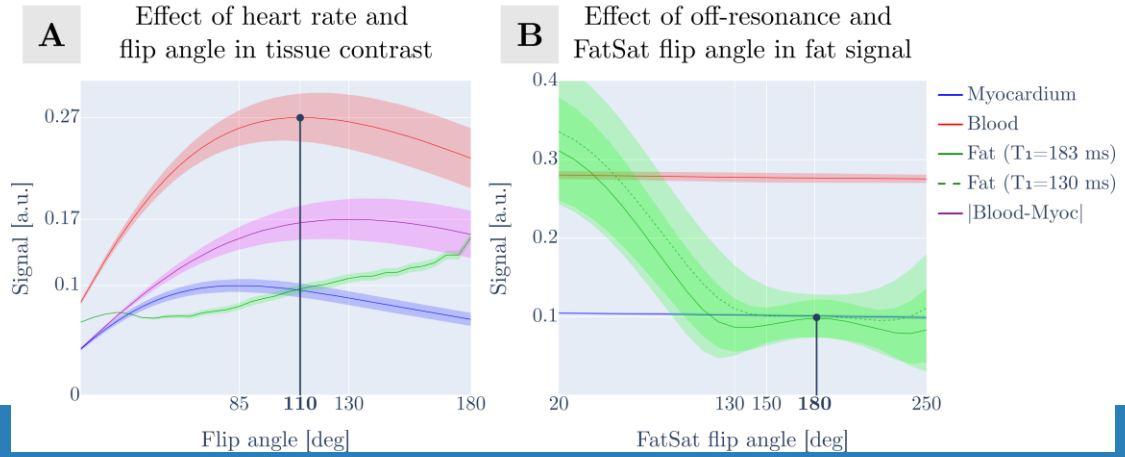


@3T

Whole-Heart Cardiovascular MRA with iNAV-based Non-Rigid Motion-Corrected Reconstruction at 0.55T

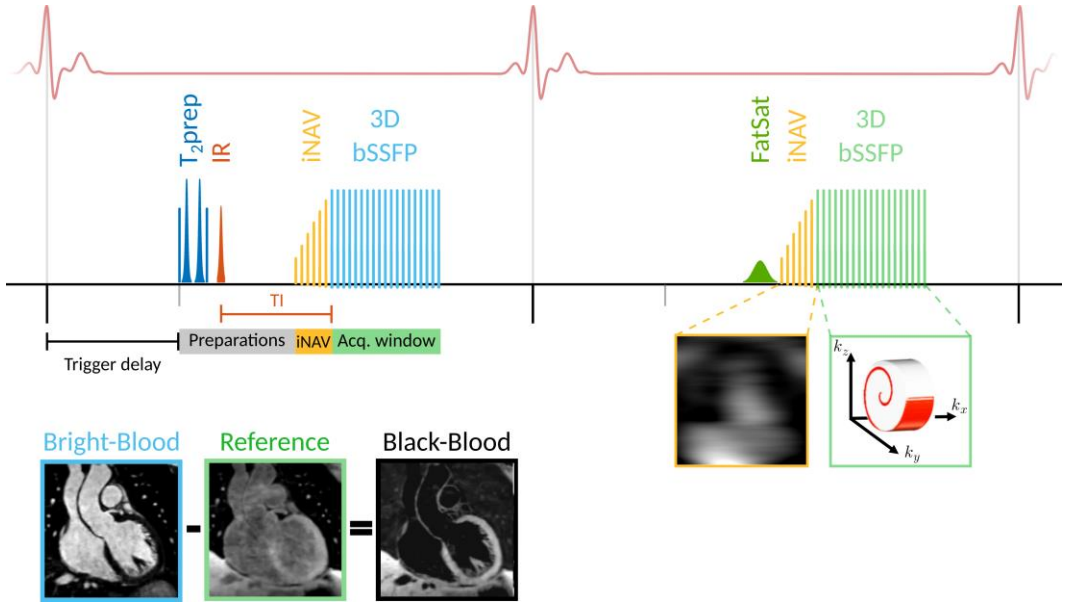
Carlos Castillo-Passi, Michael Crabb, Camila Munoz, Karl P. Kunze, Radhouene Neji, Pablo Irarrazaval, René M. Botnar, and Claudia Prieto

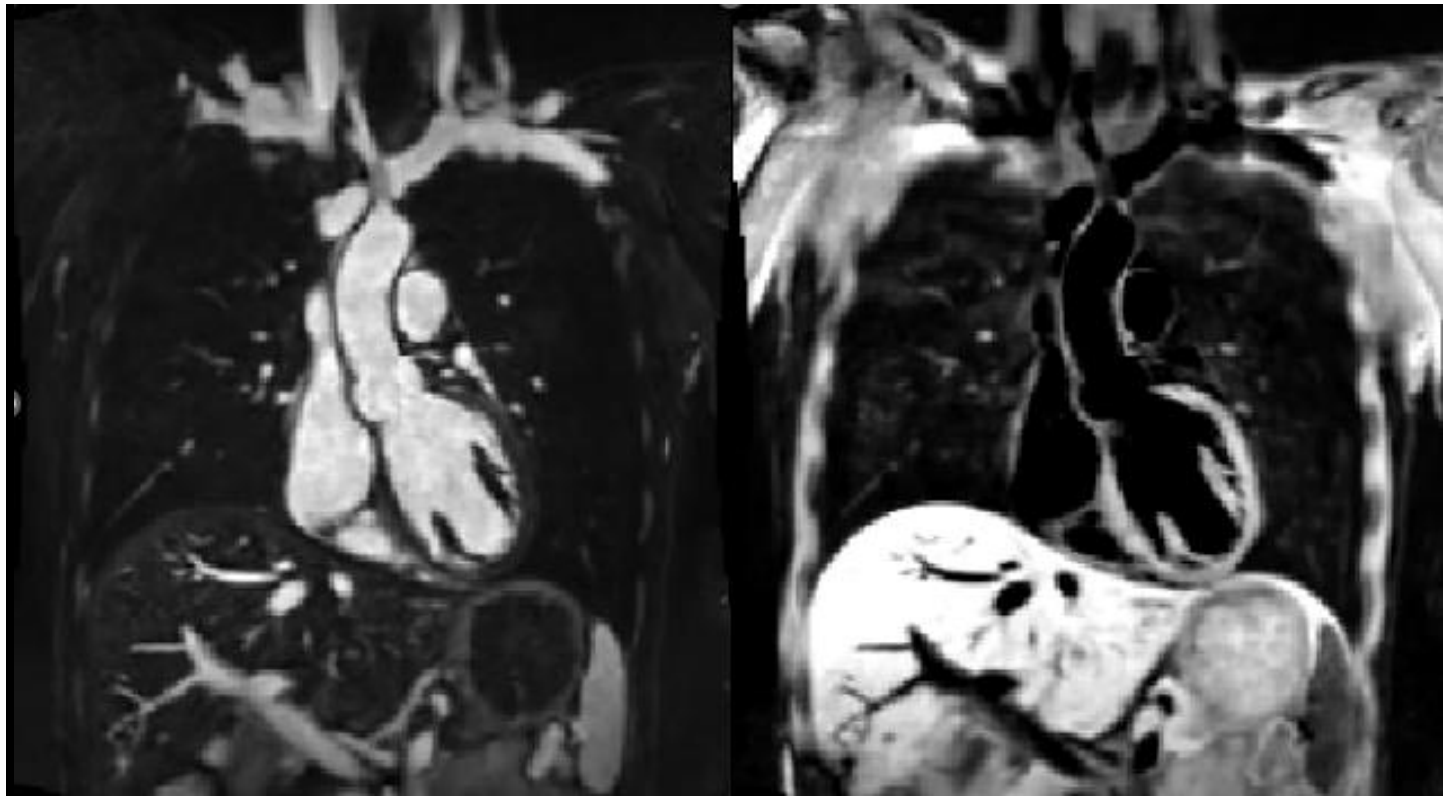




Simultaneous iNAV-based 3D Whole-Heart Bright-Blood and Black-Blood Imaging at 0.55T

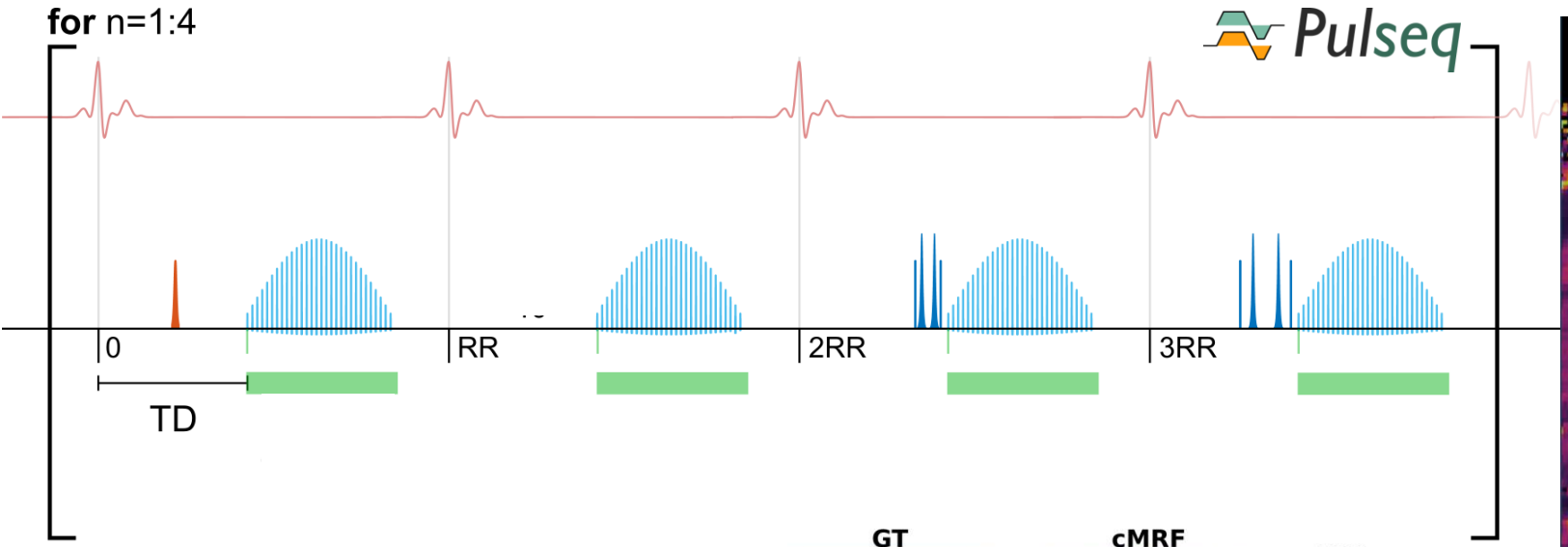
Carlos Castillo-Passi, Karl P. Kunze, Michael Crabb, Camila Munoz, Radhouene Neji,
Pablo Irarrazaval, René M. Botnar, and Claudia Prieto

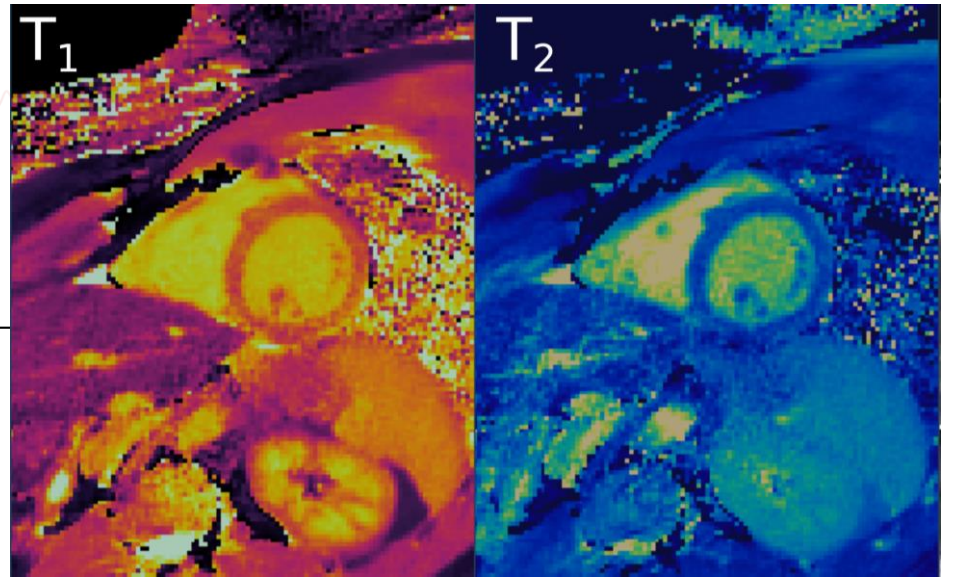


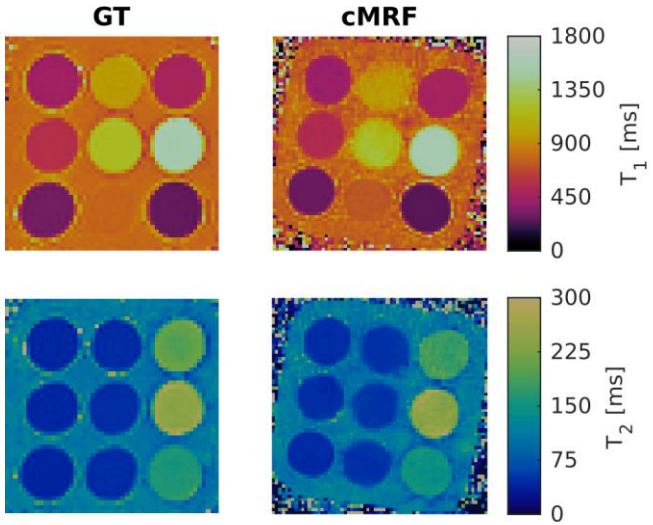


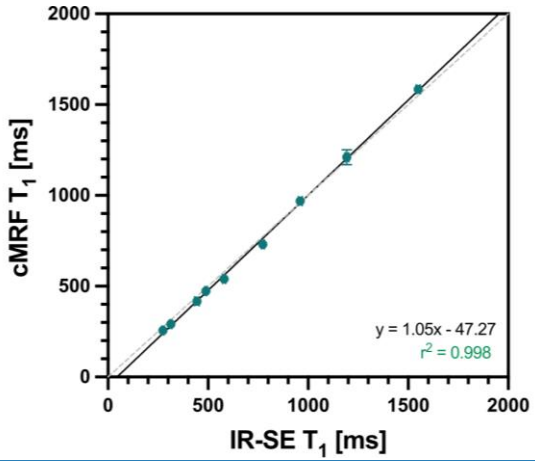
Cardiac Magnetic Resonance Fingerprinting for Simultaneous T1 and T2 Mapping at 0.55T

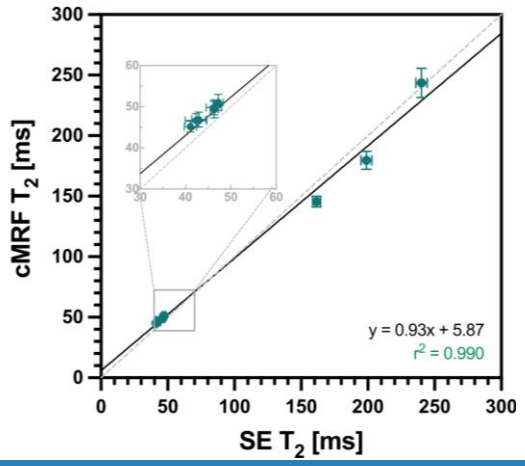
Carlos Castillo-Passi, Carlos Velasco, Donovan Tripp, Karl P. Kunze, Radhouene Neji, Pablo Irarrazaval , René M. Botnar, and Claudia Prieto





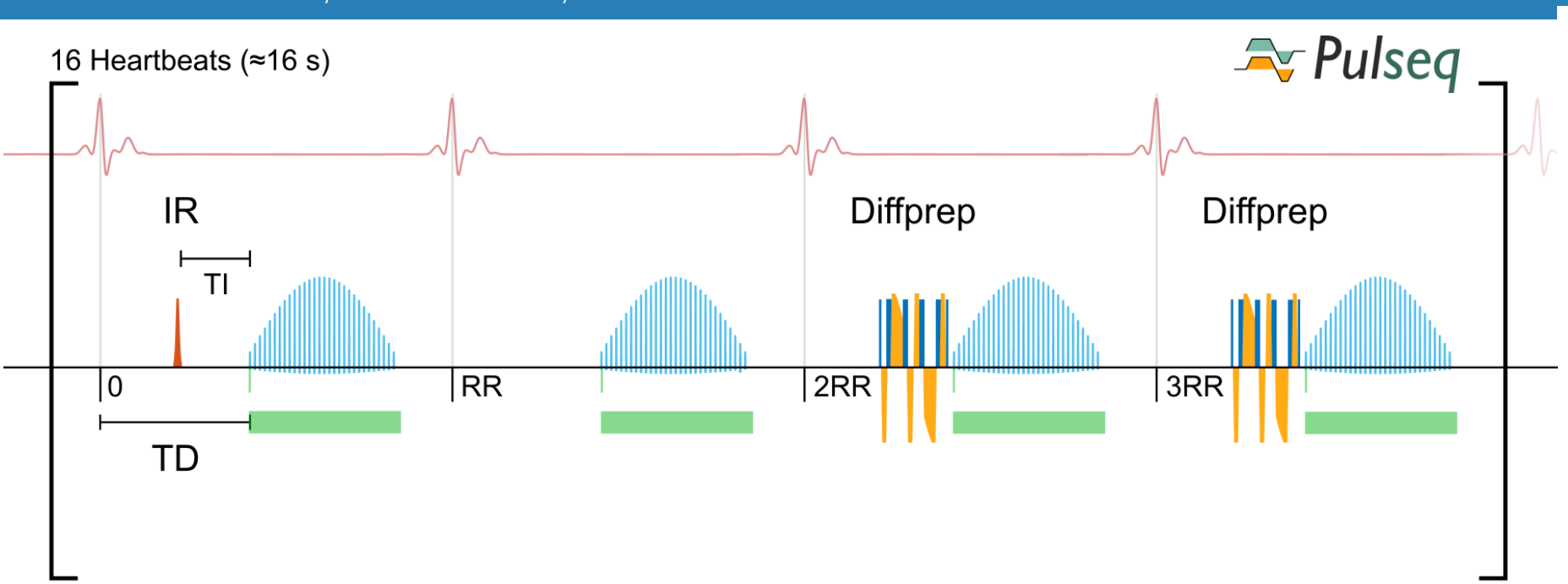


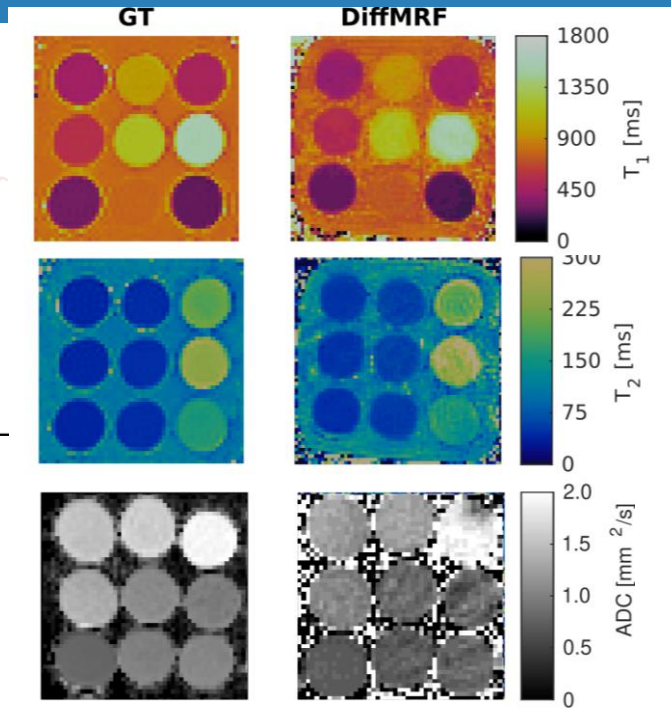


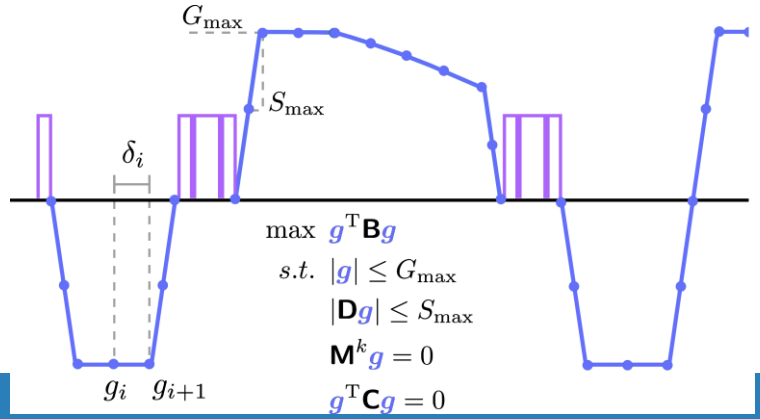


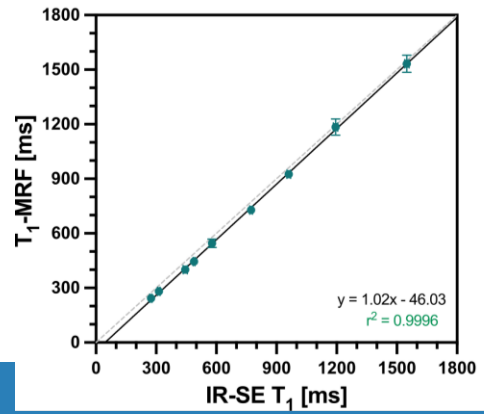
Cardiac Magnetic Resonance Fingerprinting for Simultaneous T1 and T2 Mapping at 0.55T

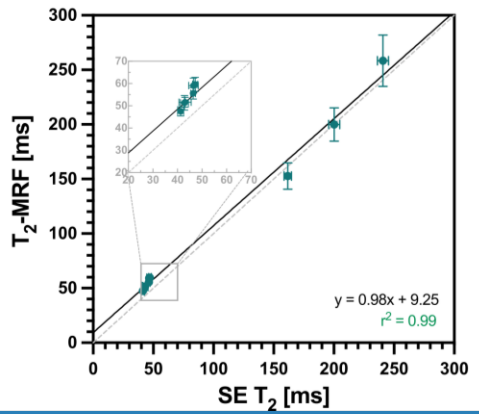
Carlos Castillo-Passi, Carlos Velasco, Donovan Tripp, Karl P. Kunze, Radhouene Neji, Pablo Irarrazaval, René M. Botnar, and Claudia Prieto

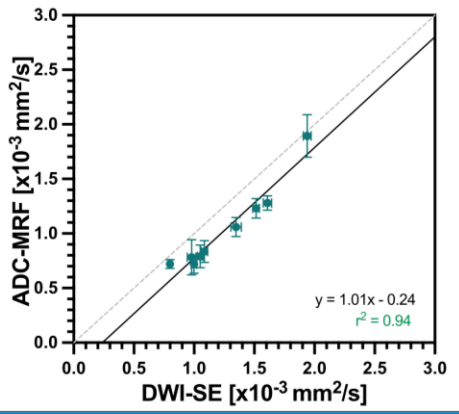






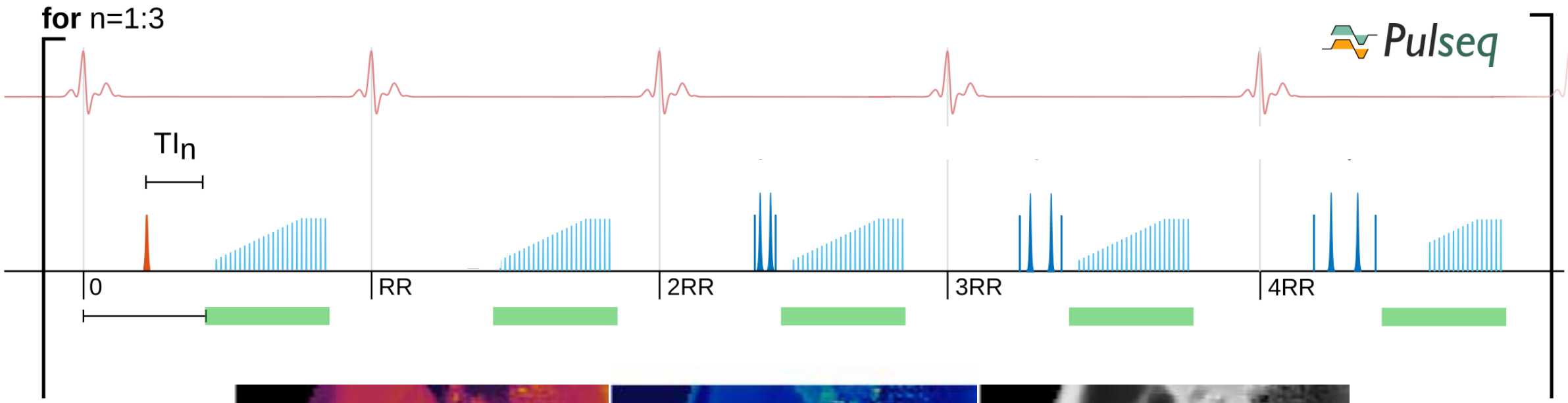


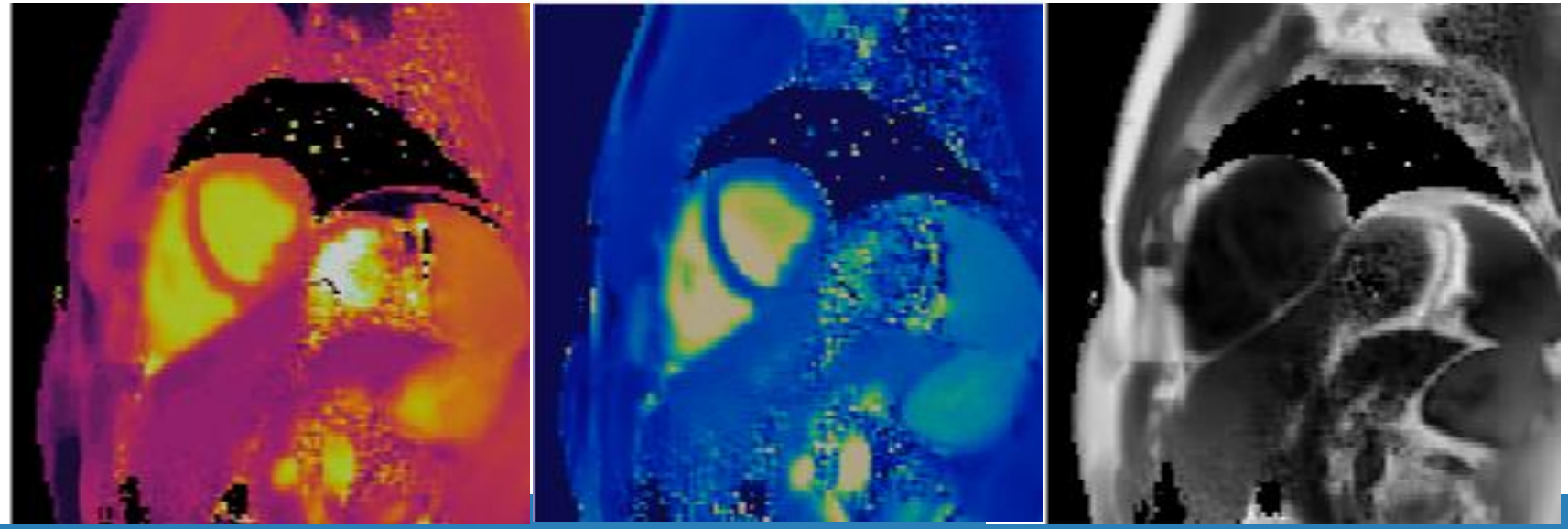




Cardiac MRF for Simultaneous T1, T2 and Fat-Fraction Quantification at 0.55T

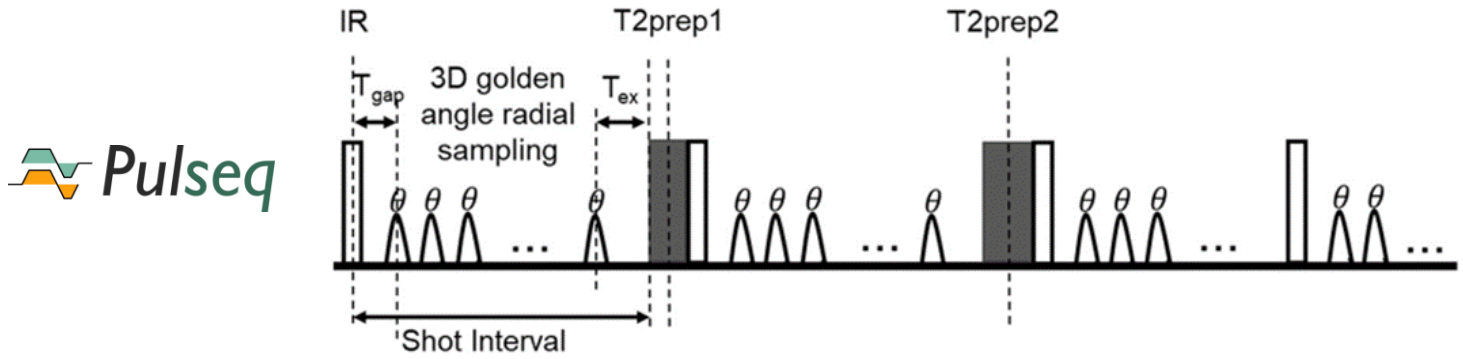
Diego Pedraza, Carlos Castillo-Passi, Nicole Araya, Carlos Velasco, René M Botnar, and Claudia Prieto

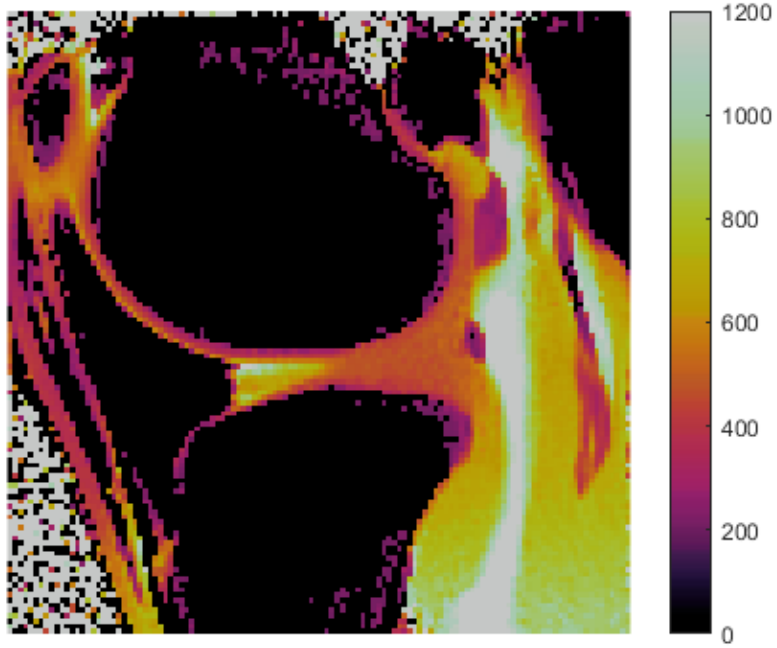


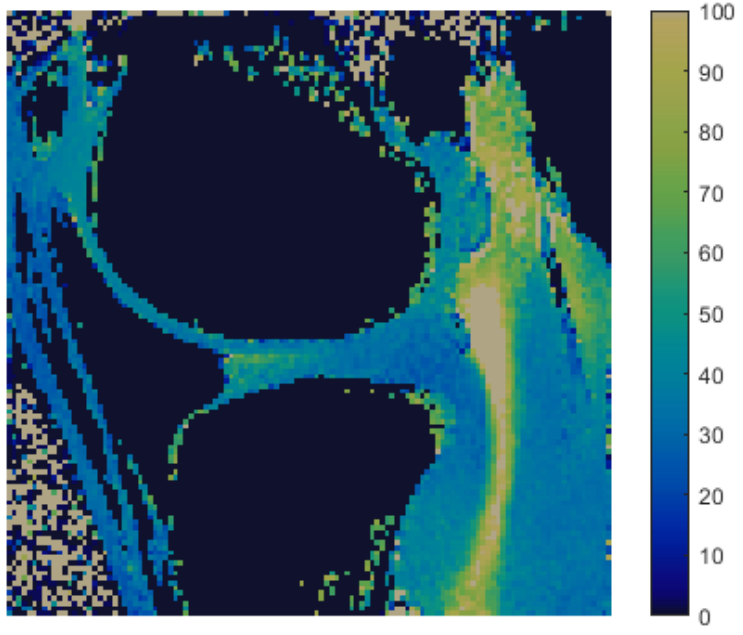


Highly efficient simultaneous joint T1-T2 mapping for isotropic resolution 3D knee imaging at 0.55 T

Nicolas Garrido, Carlos Castillo-Passi, Nicole Araya, Andrew Phair, Claudia Prieto, and René Botnar







Future work

Future work

- Multi-vendor GPU support (NVIDIA, AMD, Metal, Intel, etc)
- Performance improvements
- More examples in documentation
- and more!

Acknowledgments

ihealth | Millennium Institute for Intelligent Healthcare Engineering

iibm Instituto de Ingeniería Biológica y Médica

Special thanks to:

Boris Oróstica
Pablo Villacorta
Carlos Velasco
Nicolás Garrido
Diego Pedraza
Claudia Prieto
Maxim Zaitsev
Jon-Fredrik Nielsen

Star KomaMRI.jl on GitHub!!

https://github.com/cncastillo/KomaMRI.jl