Sycomore: an MRI simulation toolkit

Julien LAMY. ICube, Université de Strasbourg-CNRS
Paulo LOUREIRO DE SOUSA. ICube, Université de Strasbourg-CNRS

Synopsis

Sycomore is an open-source MRI simulation toolkit which provides a user-friendly and consistent interface in Python for five different simulation models (Bloch simulation, three variants of EPG and the Configuration Model). Its C++ computing core, additionally helped by OpenMP, offers efficient computation of those five simulation models on desktop computers. The interactive run-time achievable for classical MRI experiments make it a valuable tool for rapid development of sequence prototypes and for teaching.

Summary of Main Findings

Sycomore is an open-source MRI simulation toolkit which is both fast and user-friendly. It includes five different models (Bloch simulation, 3 variants of EPG and the Configuration Model), and is available on Linux, macOS and Windows.

Body of the abstract

Introduction

Simulation of MRI sequences has proven to be an invaluable tool to explore and teach the underlying physical phenomena, and to allow prototyping new sequences without requiring either access to the scanner or a wide range of phantom objects. Multiple models have been proposed, from the direct solutions of Bloch equations for a single isochromat to the Fourier-based states of Extended Phase Graphs (EPG). Although these models have been supported by multiple toolkits, those are mostly dedicated to a single model and performance is often at the price of a steep learning curve.

This abstract describes a new simulation toolkit, called Sycomore, which provides a user-friendly and consistent interface in Python for five different simulation models (Bloch simulation, three variants of EPG and the Configuration Model), and is released under a permissive free software license. With a core developed in C++, additionally helped by OpenMP, Sycomore offers efficient computation of those five simulation models, even on computers not dedicated to high-performance computing. After describing the design and features of the software, sample results -- along with representative runtimes -- will be presented.

Design and features

The features of Sycomore can be grouped in three main parts: a unit system to define type-safe physical quantities, a set of objects common to every model (e.g. species or RF pulses), and the simulation models themselves.

Including a unit system in a simulation toolkit was motivated by the fact that in most MRI textbooks and papers, the physical quantities (magnetic field strength, durations, angles, etc.) describing the basic concepts (e.g. species or pulses) are usually expressed with different SI prefixes: it is common to find ms for relaxation times, µs for pulse durations, and s/mm^2 for b-values. Converting between those is burdensome and error-prone. To alleviate this, Sycomore provides an explicit unit system in which quantities are defined in a way closer to their usual representation: a gradient moment can be defined as `gradient=100*mT/m*ms` and later

converted to cgs units with the expression `gradient.convert_to(G/cm*s)`. The unit system defines all base units of the SI and their prefixes, and also includes type checking, so that adding or converting incompatible quantities will raise errors.

The three main objects common to every model follow the high-level concepts involved in MRI sequences: *RF pulses* interleaved with *time intervals* (with or without magnetic field gradients), acting on *species*. The basic RF pulses are hard pulses, defined only by their flip angle and their phase; using the hard-pulse approximation, shaped pulses can however be defined by providing a function describing the pulse envelope. Time intervals are simply defined by their duration and amount of dephasing caused by gradients. Lastly, a species is defined by its relaxation rates (R_1, R_2 and $R_2'=R_2^*-R_2$), its diffusion coefficient (for isotropic species) or diffusion tensor (for anisotropic species) and its off-resonance frequency.

Based on these simple concepts, all models provide a consistent interface with two main actions: `apply_pulse` and `apply_time_interval`. The Bloch simulation model is based on a matrix-vector formalism[1], extended to homogeneous coordinates to yield a fully matrix-based simulation. It simulates isolated isochromats and provides insights about the steady-state of a sequence through spectral analysis, but does not include diffusion. Three EPG models are provided: in order of increasing complexity (both of the sequence being modeled and of runtime), they are a 1D regular model[2] in which the gradient-induced dephasing in each time interval is the same, a 1D discrete model[3] in which the dephasings in each time interval may vary (but are discretized, i.e. binned, to avoid numerical instability) and a 3D discrete model in which the gradients and diffusion coefficient are defined in three dimensions to simulate slice profiles and anisotropic diffusion. All three EPG models include diffusion. The last model is the Configuration Model[4], which includes diffusion and easy simulation of off-resonance effects.

Results

The figures showcase results of simulations on classical MRI experiments using Sycomore, as well as a code sample. Indicative runtimes on a 2015 MacbookPro are as follow (only the parameters affecting the runtimes are specified). The Bloch simulation of a RARE sequence with a time step of 5 ms and 50 isochromats (total of 12000 iterations) shown on Figure 1 takes 38 ms. The simulation of 160 repetitions of a GRE sequence with regular EPG for 4 different conditions, shown on Figure 2 takes 14 ms; the corresponding code is shown in Figure 3. The diffusion attenuation of an anisotropic species probed with 2402 gradient directions (Figure 4) takes 446 ms. All of these show that interactive runtime can be reached with simulations representative of real-life experiments.

Conclusion

Sycomore has been released under a free software license (MIT), and its source code is available on GitHub[5]. Documentation, including code samples is available on ReadTheDocs[6], and pre-compiled packages are distributed through PyPi[7] and Anaconda[8].

The provided models are mostly complete, but a few operators remain non-implemented: diffusion for Bloch simulation[9], flow in EPG[10] and MT for all models[11]. In addition to implementing those missing features, future work includes adding a GPU-based computation back-end, further improving the performance in more demanding situations, e.g. sequence optimization.

References

1. Hargreaves et al., *Characterization and reduction of the transient response in steady-state MR imaging.* Magnetic Resonance in Medicine 46(1), 2001.
2. Hennig, *Multiecho imaging sequences with low refocusing flip angles.* Journal of Magnetic Resonance 78(3), 1988.

3. Weigel et al., *Extended phase graphs with anisotropic diffusion*. Journal of Magnetic Resonance 205(2), 2010.
4. Ganter, *Configuration Model*. ISMRM 2018
5. https://github.com/lamyj/sycomore/
6. https://sycomore.readthedocs.io/
7. https://pypi.org/project/sycomore/
8. https://anaconda.org/conda-forge/sycomore/
9. Jochimsen et al., *Efficient simulation of magnetic resonance imaging with Bloch–Torrey equations using intra-voxel magnetization gradients*. Journal of Magnetic Resonance 180(1), 2006.
10. Weigel, *Extended phase graphs: Dephasing, RF pulses, and echoes - pure and simple*. Journal of Magnetic Resonance Imaging 41(2), 2015.
11. Malik et al., *Extended phase graph formalism for systems with magnetization transfer and exchange: EPG-X: Extended Phase Graphs With Exchange*. Magnetic Resonance in Medicine 80(2), 2018

Figures

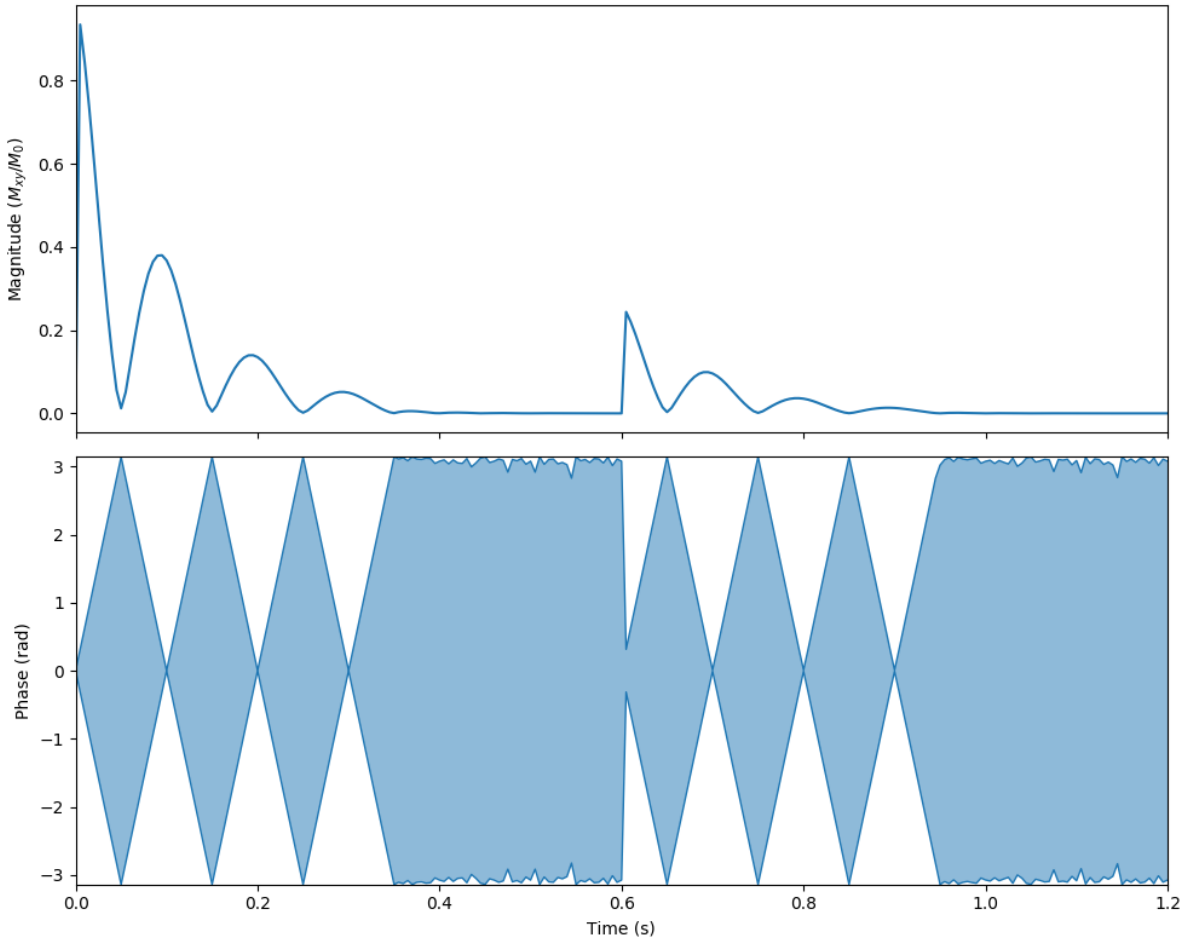

Figure 1 - Bloch simulation of a RARE sequence. Magnitude of mean signal and phase range of all isochromats. T1=1000 ms, T2=100 ms, TR=600 ms, TE=100 ms, excitation flip angle=90°, refocalization flip angle=180°, echo train length=3, time step=5 ms, 50 isochromats.

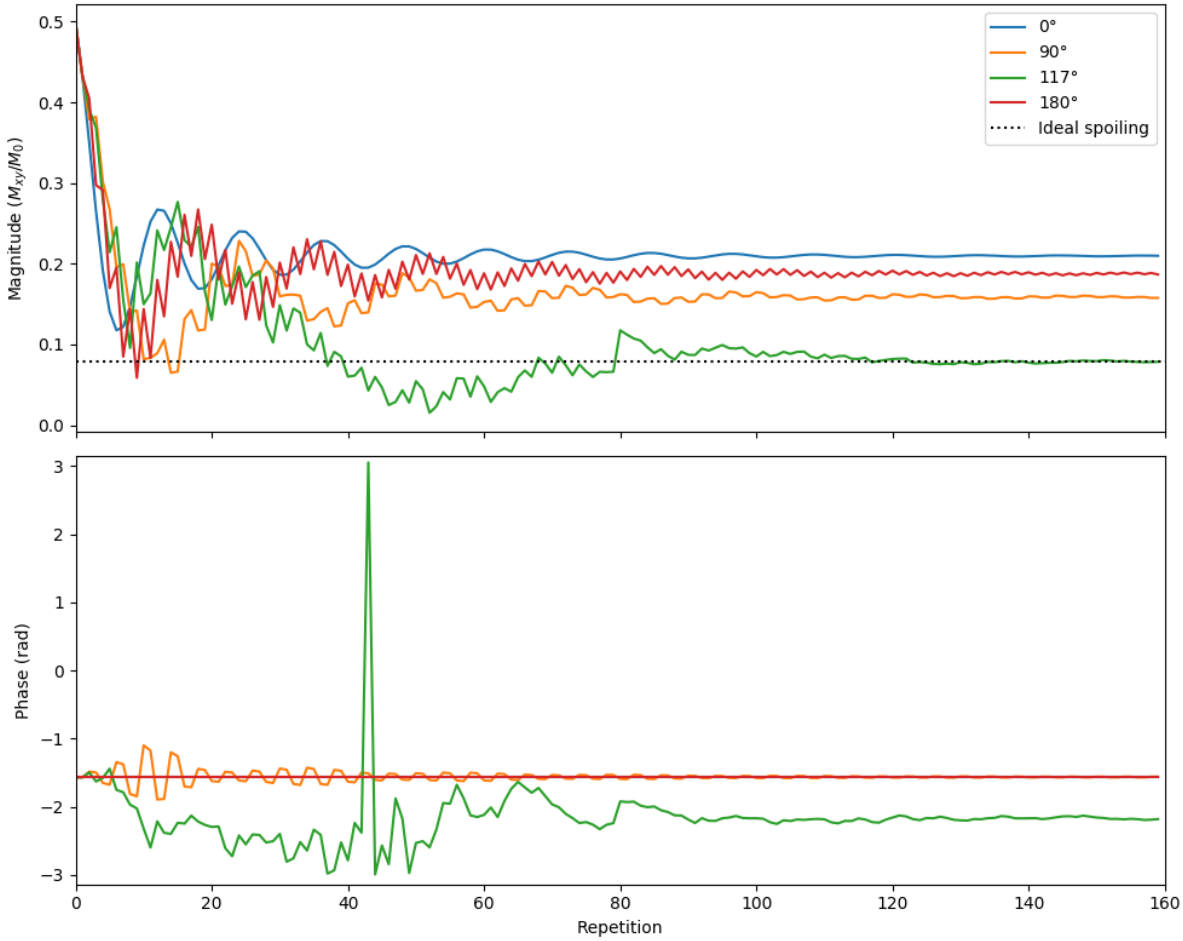

Figure 2 - Regular EPG simulation of an RF-spoiling experiment with four phase shift increments (0°, 90°, 117° and 180°). Magnitude and phase at echo time. T1=1000 ms, T2=1000 ms, flip angle=30°, TE=5 ms, TR=25 ms, 160 repetitions.

```
species = sycomore.Species(1000*ms, 100*ms)
flip_angle=30*deg
TE = 5*ms
TR = 25*ms
phase_step = 117*deg
slice_thickness = 1*mm
G_readout = 2*pi*rad / (sycomore.gamma * slice_thickness)

model = sycomore.epg.Regular(species)
repetitions = int((4*species.T1/TR).magnitude)

echo = numpy.zeros(repetitions, dtype=complex)
for r in range(0, repetitions):
    phase = (phase_step * 1/2*(r+1)*r)

    model.apply_pulse(flip_angle, phase)
    model.apply_time_interval(TE)

    rewind = exp(-1j*phase.convert_to(rad))
    echo[r] = model.echo*rewind

    model.apply_time_interval(TR-TE, G_readout/(TR-TE))
```

Figure 3 - Code sample for the RF-spoiling experiment

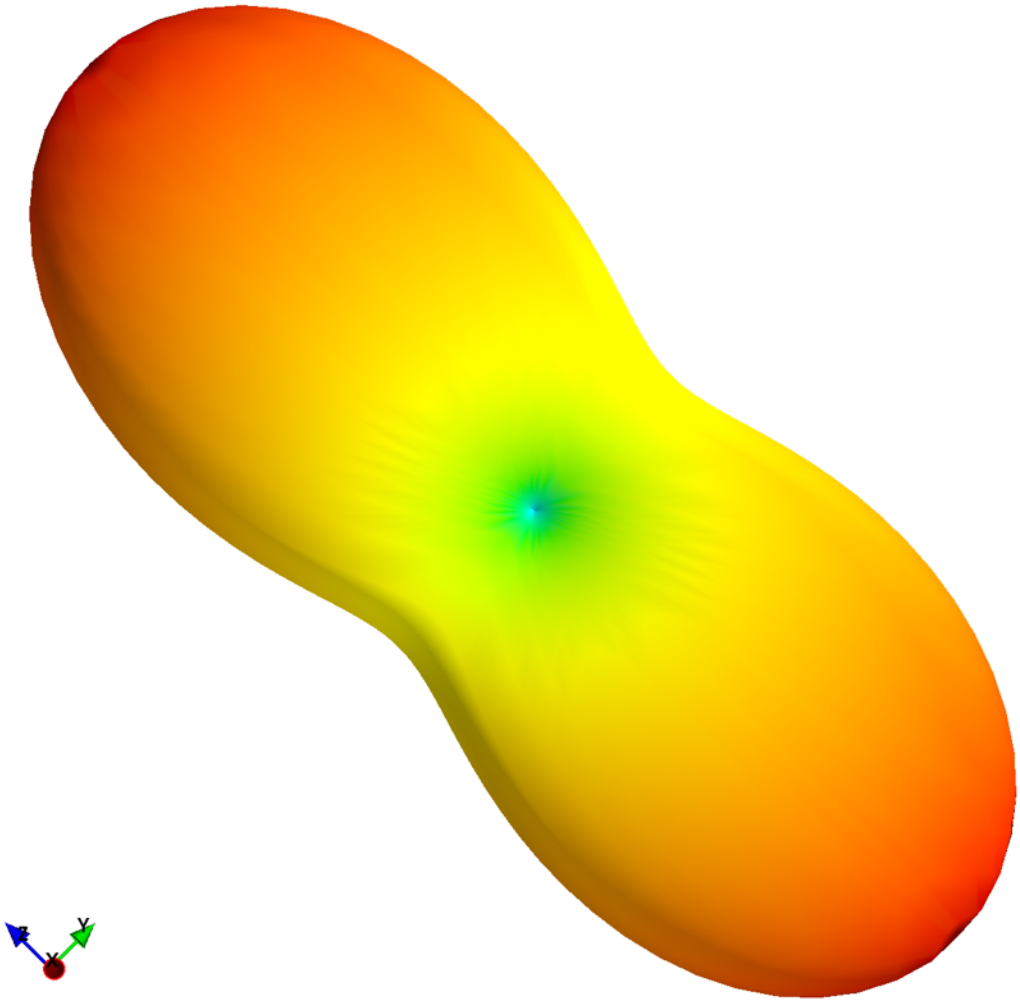

Figure 4 - 3D Discrete EPG simulation of the diffusion-related signal attenuation in a PGSE sequence. T_1=1000 ms, T_2=100 ms, D=(1708, 303, 114) μm²/s, TE=50 ms, b=1000 s/mm² (δ=15.3 ms, Δ=34.7 ms), 2402 directions.