#47700

epgpy: *an easy-to-use python library for MRI simulations*

Pierre-Yves Baudin

Institute of Myology, Neuromuscular Investigation Center, NMR Laboratory, Paris, France

Introduction

- Numerical signal simulations play an important role in today's research in the field of MRI. Typical use cases: **parameter mapping**, MR fingerprinting **dictionary generation** and **sequence optimization**.
- The extended phase-graph[1] (EPG) formalism provides an **efficient alternative to classical isochromats sampling** in Bloch equations simulations.
- We implemented several EPG extensions – anisotropic diffusion, magnetization transfer, 3D gradients, B0/R2* time-dependent dephasing – into epgpy, a simple-to-use and lightweight pure python library.

Basics

- A pure python library, with few dependencies: numpy and (optionally) cupy for GPU
- Multiple examples reproducing published works are provided along with the source code

Thanks to vectorization and GPU compatibility, large dictionaries can be generated efficiently.

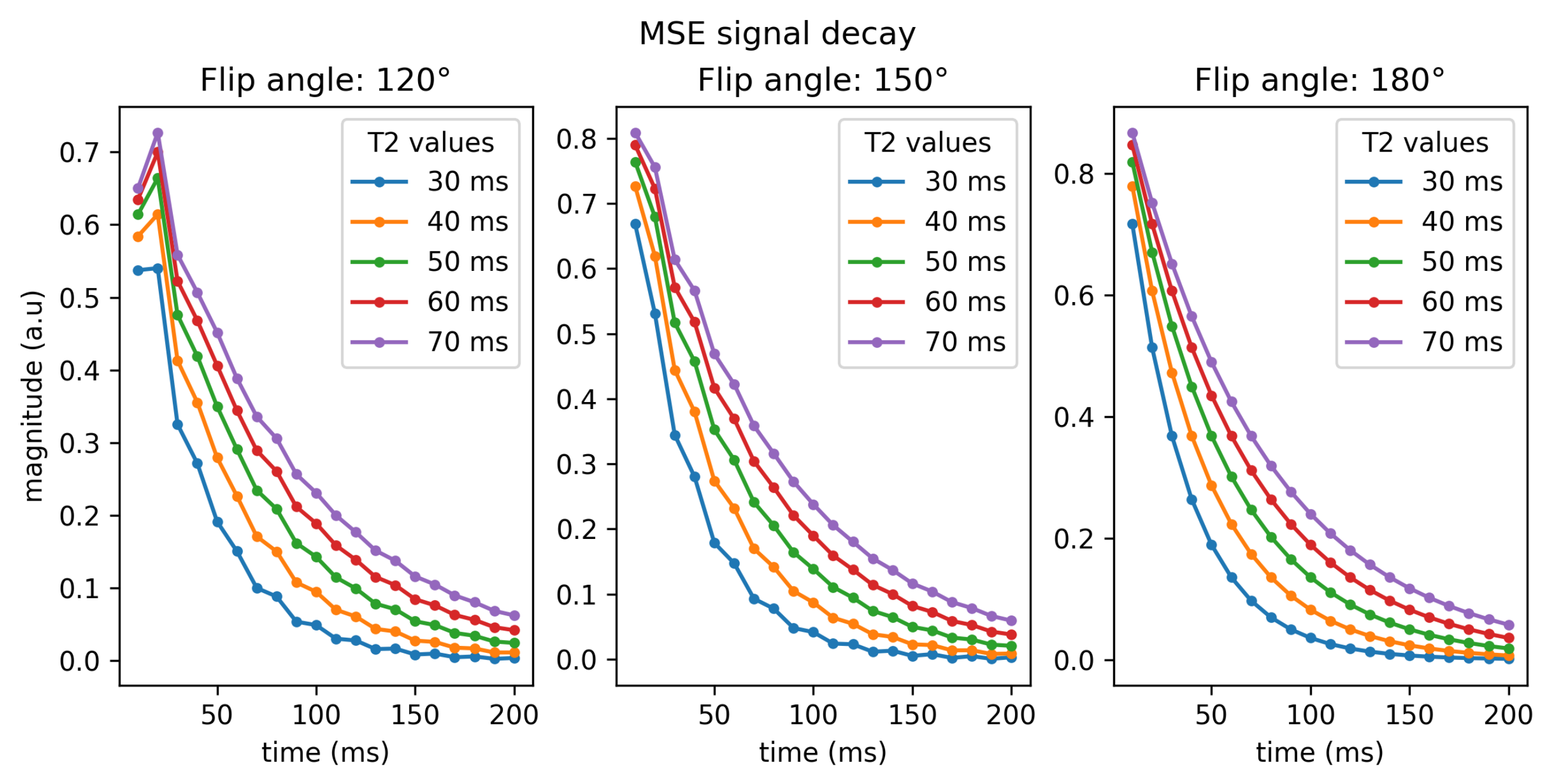
The three basic EPG operators[1] can be used to simulate simple multi-spin echo and steady-state sequences and generate phase graphs:

- **Evolution operator E**, for relaxation and precession
- **Transition operator T**, for instantaneous RF pulses
- **Shift operator S**, for phase states changes (spoiler gradients)

Sequences consists of python lists of EPG operators, with an additional ADC operator to record the current signal (e.g. the F0 state of the state matrix).

Example: simulate the effects of various T2 and flip angle values in a multi-spin echo sequence (MSE). The stimulated echo phenomenon is visible for FA < 180°.

All signals are generated simultaneously thanks to vectorization.



python code for the MSE example

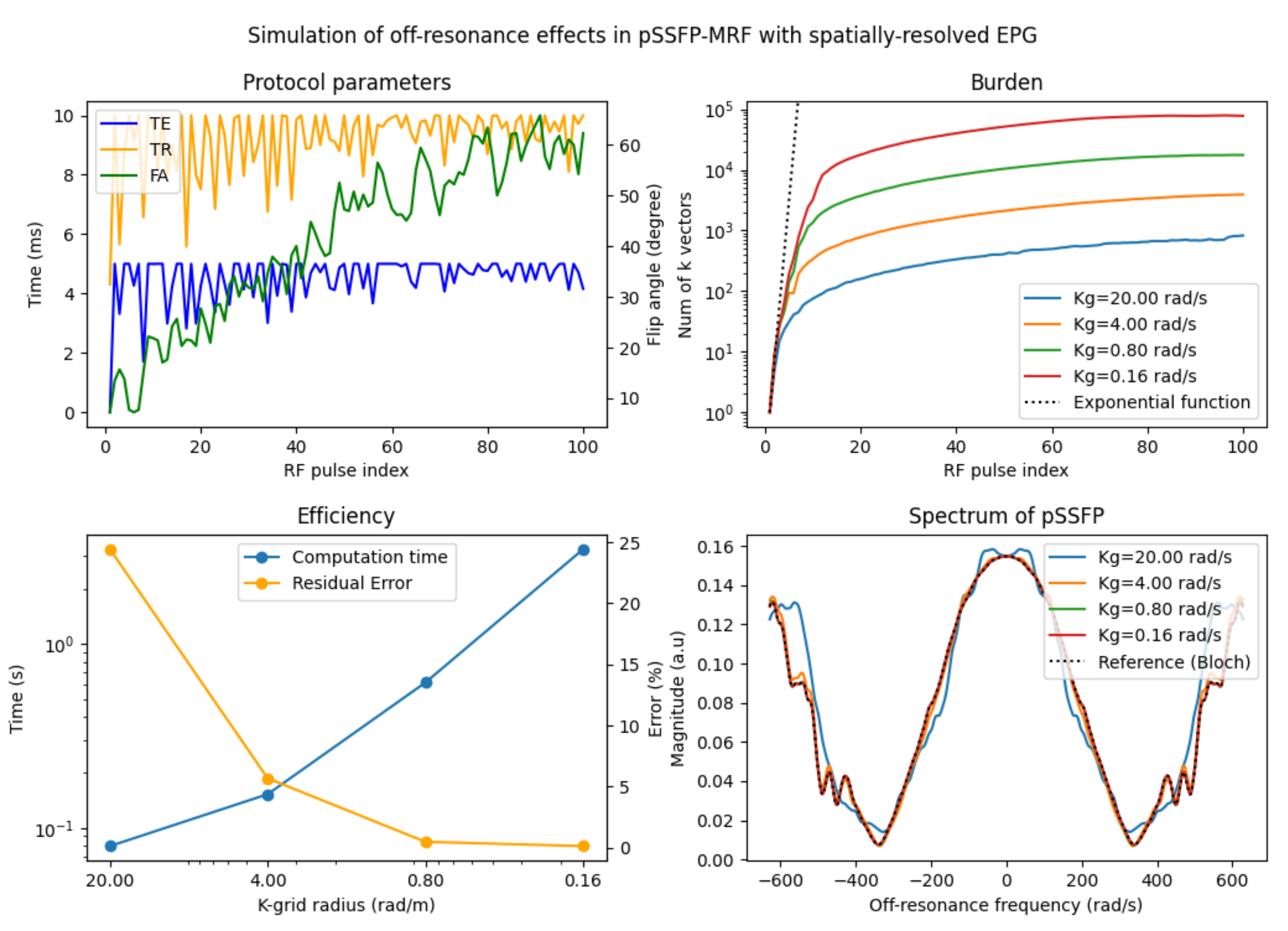
```
from epgpy import epg

# sequence
num_echo = 20 # number of echoes
FA = [120, 150, 180] # flip angles (degrees)
ESP = 10 # echo spacing (ms)

# relaxation times
T1 = 150 # ms
T2 = [[30, 40, 50, 60, 70]] # ms

# operators
exc = epg.T(90, 90) # excitation
rfc = epg.T(FA, 0) # refocusing
rlx = epg.E(ESP / 2, T1, T2) # relaxation
spl = epg.S(1, duration=ESP / 2) # spoiler
adc = epg.ADC # ADC flag

# sequence (repeated spin-echoes)
seq = [exc] + [[spl, rlx, rfc, spl, rlx, adc]] * num_echo

# signal
signal = epg.simulate(seq)
```

3D gradients

Generalized shift operator S
S uses a quantize-and-merge algorithm[4] for non-integer phase shifts, applied onto a 4D coordinates array: 3D spatial wavenumbers + time accumulation.
S can simulate:

- Arbitrary 3D gradients
- Local field effects (T2*/B0)

Example: simulation of a pSSFP-MRF sequence on a laptop PC with random TE/TR and its sensitivity to off-resonance patterns, as a function of the k-grid resolution and number of phase states.
Results closely match those of the original publication[4].

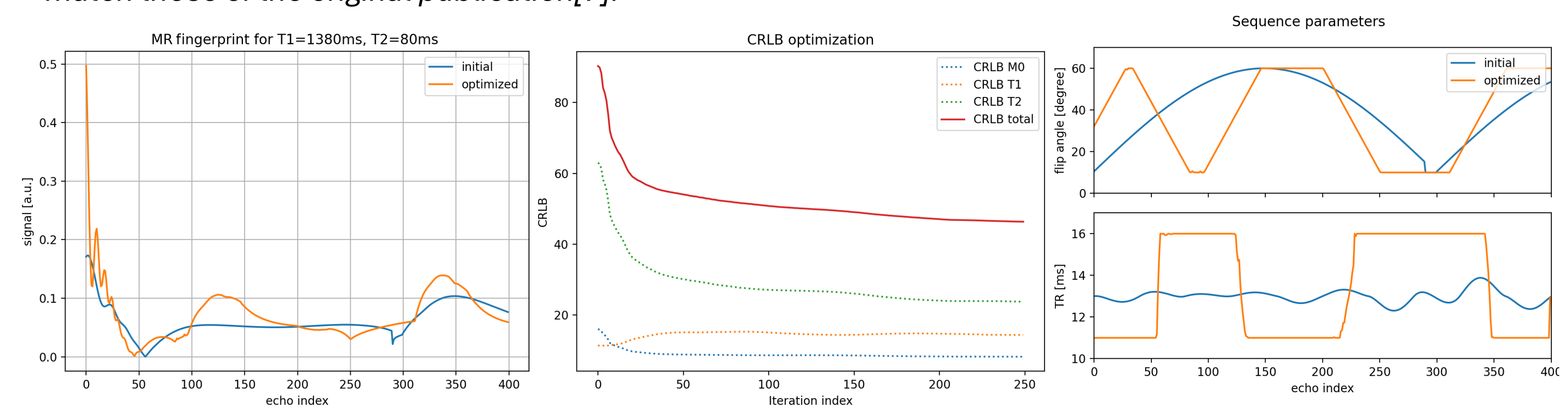


Differentiation

Differentiable EPG operators for **sequence optimization** and **confidence interval** calculation

- 1st and 2nd order explicit differentiation for the three classical EPG operators
- Convenient framework for sequence optimization with a Cramer-Rao bound objective

Example: optimization of a pseudo random MRF sequence[7] with 400 TRs and flip angles, using a CRB objective for magnitude, T1 and T2.
The 1st and cross derivatives of the signal w.r.t T1, T2, and all FAs and TRs are computed. Results closely match those of the original publication[7].



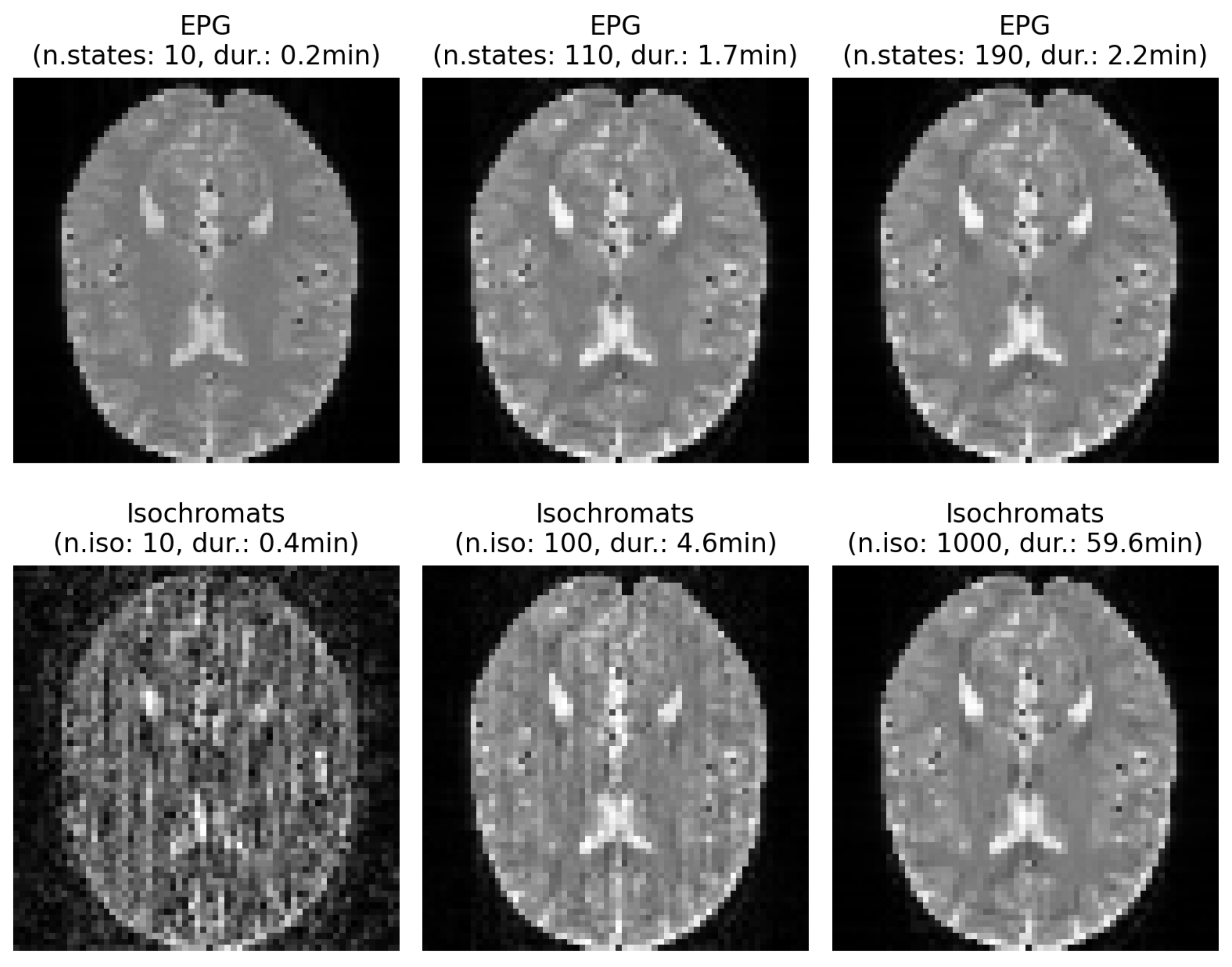
Advanced operators

Advanced EPG operators, including:

- **Diffusion operator D**, for anisotropic diffusion[2]
- **Exchange operator X**, for compartment exchange and magnetization transfer[3]
- **Generalized Shift operator S**

In **S**, the 4D spatial-temporal coordinates array achieves a formalism similar to phase distribution graphs[5] where reversible T2*/B0 effects can be simulated. All operators can be combined to simulate a realistic imaging system more efficiently than with isochromat sampling.

Example: 2D imaging sequence on a 64 x 64 brain phantom[6] on a laptop PC with parameters: proton-density, T1, T2 and T2 and a FOV of 20x20 cm^2.*
Comparison of the EPG or the isochromats sampling approaches as a function of the number of phase states or isochromat number (reproduction of an example from [5]).


EPG (n.states: 10, dur.: 0.2min)
EPG (n.states: 110, dur.: 1.7min)
EPG (n.states: 190, dur.: 2.2min)
Isochromats (n.iso: 10, dur.: 0.4min)
Isochromats (n.iso: 100, dur.: 4.6min)
Isochromats (n.iso: 1000, dur.: 59.6min)

References

[1] Weigel, JMRI 2015, doi: 10.1002/jmri.24619. [2] Weigel et al., JMR, 2010, doi: 10.1016/j.jmr.2010.05.011. [3] Malik et al., MRM, 2018, doi: 10.1002/mrm.27040. [4] Gao et al., MRM 2021, doi: 10.1002/mrm.28732. [5] Endres et al., MRM 2024, doi: 10.1002/mrm.30055. [6] Aubert-Broche et al., Neuroimage, 2006, doi: 10.1016/j.neuroimage.2006.03.052. [7] Lee et al., MRM, 2019, doi: 10.1002/mrm.27832.

Conclusion

With few dependencies, a pythonic syntax, simple sequence definitions, efficient numerical computations, epgpy is a flexible EPG simulation library, suitable for fast prototyping, teaching, dictionary generation or sequence optimization.

https://github.com/py-baudin/epgpy

ESMRMb 2025 Marseille