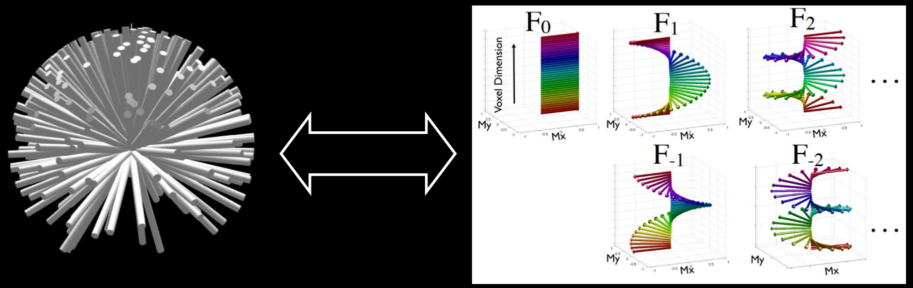
Hands on - Digital Twin of a low field scanner

Speaker: Prof. Dr. Moritz Zaiss

Simulation: Jonathan Endres

Friedrich-Alexander-Universität
Erlangen-Nürnberg

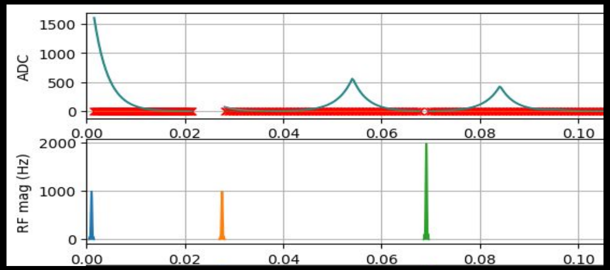
Overview

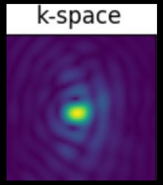


You know by now:

- That we can simulate MRI & most efficient with EPG/PDG simulations
- You can simulate Pulseq/PyPulseq definitions directly in Python/Colab

```
signal = mr0.execute_graph(graph, seq0, obj_p)
```





What we will do now

- Adapt the simulation
  - **For a low field system** including typical phantom parameters
  - **For a 3D sequence.** Required for SNR-reasons, but long and slow in simulation.
- Explore the low field system and 3D sequence

Exploring is playing

"Play is the highest form of research." – Albert Einstein

while playing at the MR scanner is the best way to learn,
it can be slow and can be restricted by hardware/software

A playful simulation allows you
..to test many new ideas in short time,
..to ask the questions „what if" with less limits
..to identify the most interesting things to „play" out at the scanner

General steps

An MR simulation requires four ingredients:

1. a defined MR sequence
2. a defined MR phantom with all tissue and system properties
3. a call of the simulator with sequence and phantom
4. a reconstruction of k-space signals to images

General steps

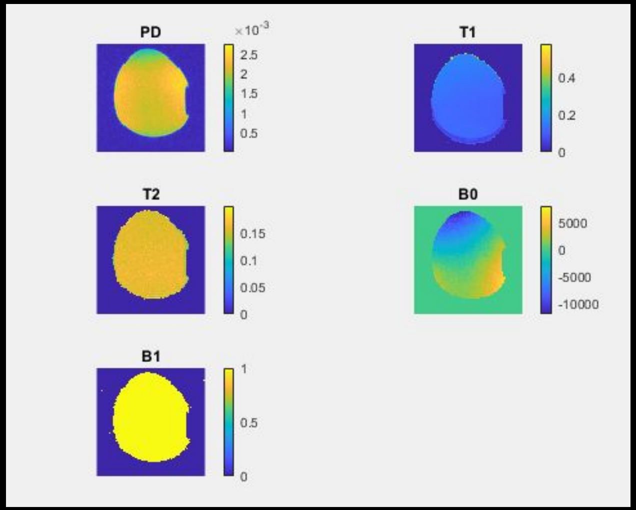
The MR simulation requires four ingredients:

Load Pulseq MTSE , ….
Simpler seqs in MR0 playground

1. a defined MR sequence
2. a defined MR phantom with all tissue and system properties
3. a call of the simulator with sequence and phantom
4. a reconstruction of k-space signals to images

ingredient #2

2.Phantom and system definition



- quantitative maps: PD,T1,T2, (T2',D)
- system-dependent maps: B0, B1+, (B1-)
- store in matlab or python, load:
  - `import MRzeroCore as mr0`
  - `obj_p = mr0.VoxelGridPhantom.load_mat('low_field_phantom.mat')`

In the colab script, go to section S3.

Inhalt

- MaRCoS system optimized 3D MTSE + MR-zero simulation
- Simulation:
- Exercises or Games - Explore Simulation:
- Code Sections for 3D MTSE forMaRCoS in a low field phantom
  - S0. environment setup (just need to run it one time)
  - S1. Pulseq sequence, wrapped as a function create_mtse
- Run all below wrapper
  - S2. interactive changes on seq
  - **S3. Load phantom and manipulate**
  - S4. Simulate
  - S5. Explore Interactive seq.plot() - experimental
  - S6. Reconstruction and image display
- Matlab Code for generating phantoms in MR-zero

+ Bereich

+ Code + Text T4 RAM Laufwerk Gemini

S3. Load phantom and manipulate

```
#@title S3. Load phantom and manipulate
dB0 = 0
phantom_fn = "low_field_phantom.mat" # @param ["l
slice_select =  [8]# @param {type:"raw"}
sz = [nRD, nPH, 1]
# %% S4: SETUP SPIN SYSTEM/object on which we can
obj_p = mr0.VoxelGridPhantom.load_mat(phantom_fn)
obj_p.T2dash[:] = 30e-3
obj_p.D *= 0

#obj_p = obj_p.interpolate(40, 40, 16) # all slic
#obj_p = obj_p.interpolate(40, 40, 16).slices([8]
obj_p = obj_p.interpolate(40, 40, 16).slices(slic

# change the size in SI untis (m), here15 cm
obj_p.size = torch.tensor([0.15,0.15,0.15])

# Manipulate loaded data
obj_p.B0*= 1/40  # alter the B0 inhomogeneity
rB1_factor= 1 # @param {type: "slider", min: 0, m
dB0_factor= 1 # @param {type: "slider", min: 0, m
T1_factor= 1 # @param {type: "slider", min: 0, ma
T2_factor= 1 # @param {type: "slider", min: 0, ma
T2dash_factor= 1 # @param {type: "slider", min: 0
D = 0  # @param {type: "slider", min: 0, max: 5}
obj_p.T1*= T1_factor
obj_p.T2*= T2_factor
obj_p.T2dash*= T2dash_factor
obj_p.B1*= rB1_factor
obj_p.B0*= dB0_factor
obj_p.D[:]= D
obj_p.plot()
```

phantom_fn: low_field_phantom.mat

slice_select: [8]

rB1_factor: 1

dB0_factor: 1

T1_factor: 1

T2_factor: 1

T2dash_factor: 1

D: 0

An single slice phantom, does not change the simulation outcome (except SNR), as the 3D sequence dynamic is unchanged.

-> valid speedup

To be able to play fast, we need speed-ups

Speed up “phantom” - less voxels to simulate

Speedup sequence

80x80x8 x4-> 40x40x2 x4

Speed up simulation

-> accuracy

-> GPU

To be able to play fast, we need speed-ups

Speed up phantom

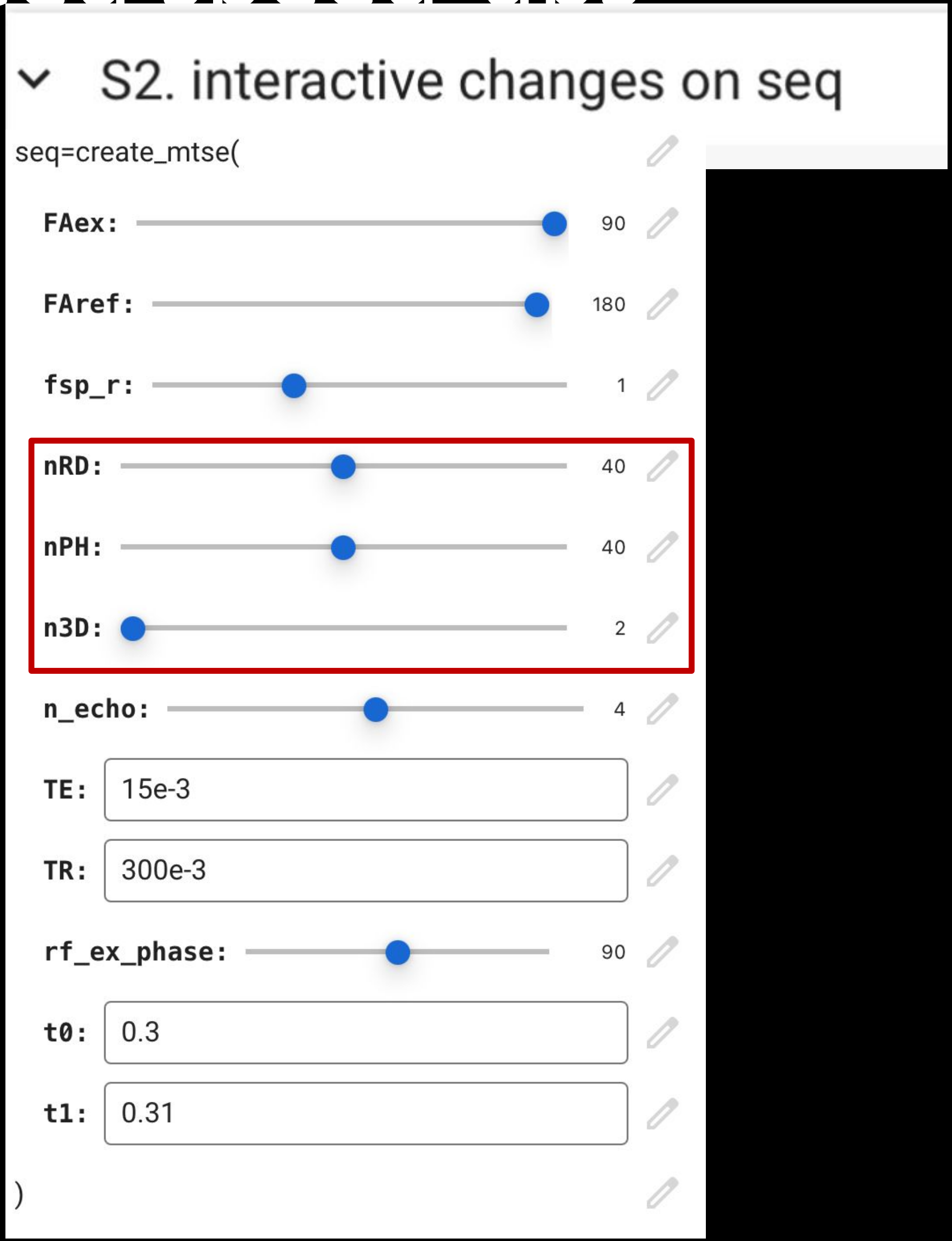
Speedup sequence

80x80x8 x4-> 40x40x2 x4

Speed up simulation

-> accuracy

-> GPU

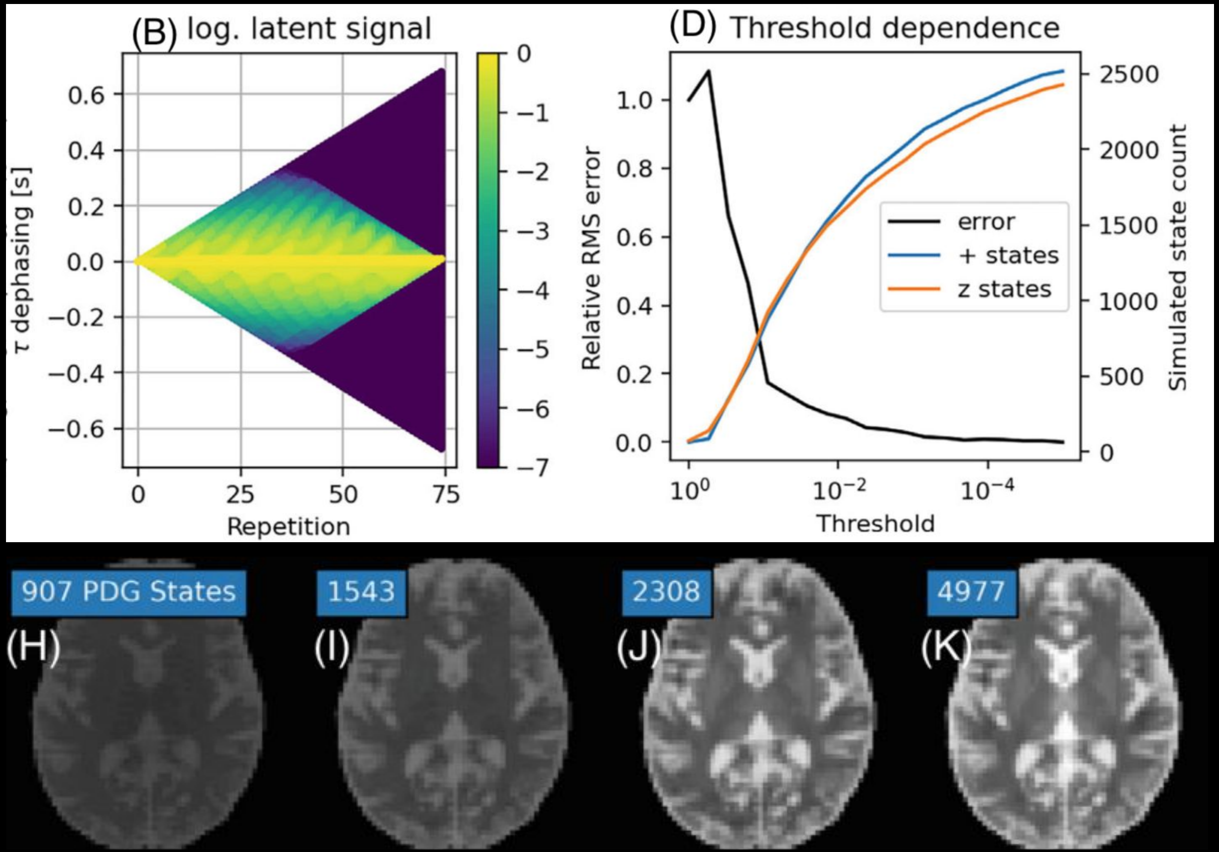


To be able to play fast, we need speed-ups

Speed up phantom

Speedup sequence

80x80x8 x4-> 40x40x2 x4



Speed up simulation

-> accuracy (Malik et al. Equivalence of EPG and Isochromats, ISMRM 2016)

```
signal = mr0.execute_graph(graph, seq0, obj_p)
#Faster simulation with lower accuracy:
signal = mr0.execute_graph(graph, seq0, obj_p,min_emitted_signal=0.05,min_latent_signal=0.05)
```

To be able to play fast, we need speed-ups

Speed up phantom

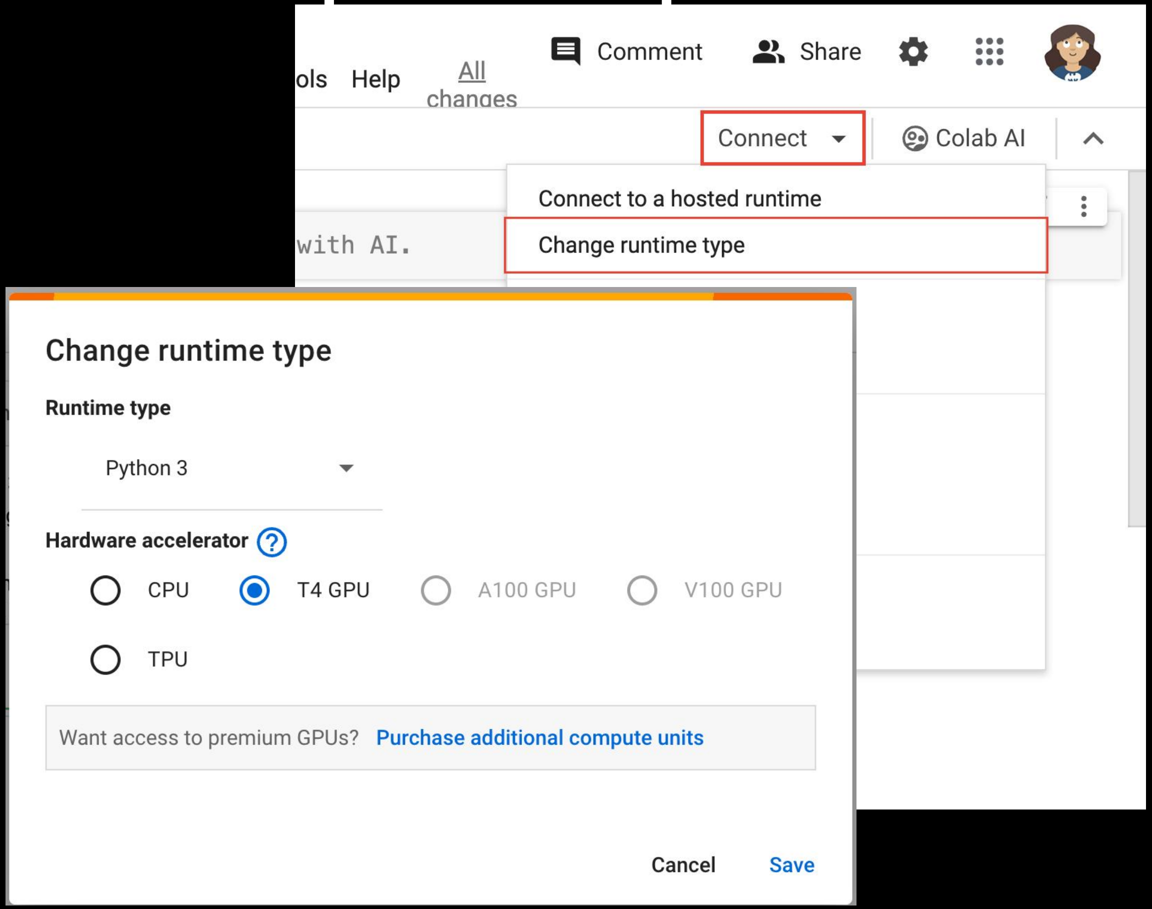
Speedup sequence

80x80x8 x4-> 40x40x2 x4

Speed up simulation

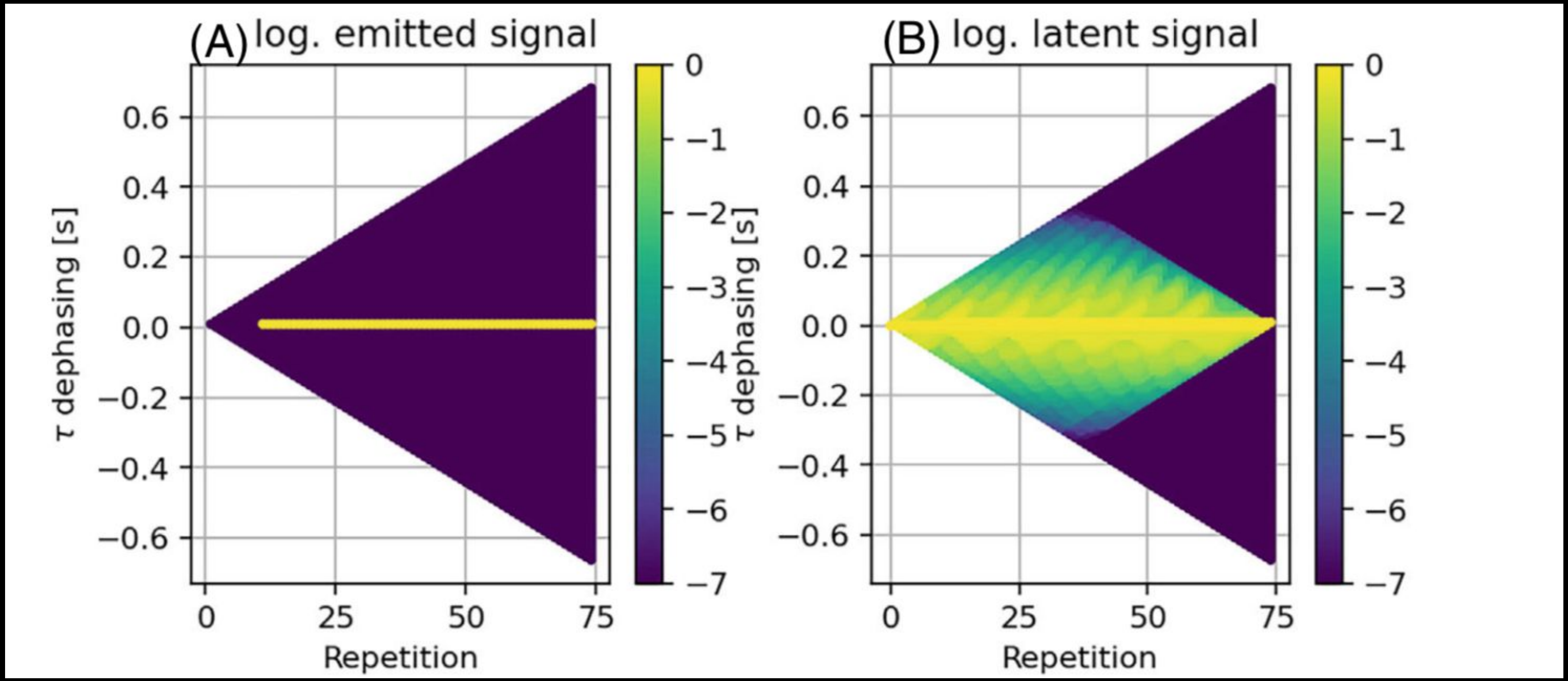
-> accuracy

-> GPU




```
signal=mr0.execute_graph(graph,
    seq0.cuda(), obj_p.cuda())
```

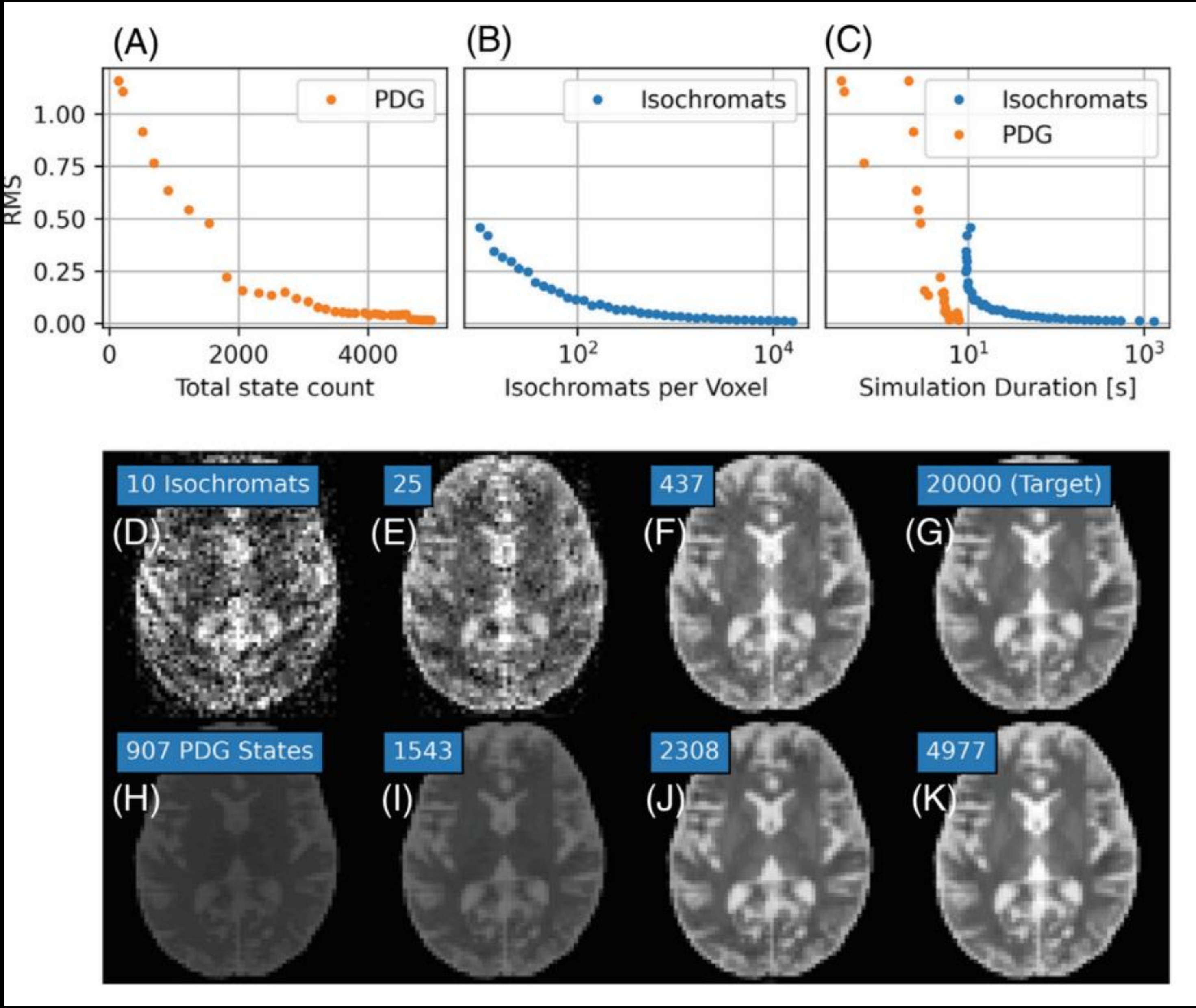
State selection and latent signal



Emitted signal:
Contribution of the state to the current signal

Latent signal:
Contribution of the state to any later acquired signal

Time efficiency

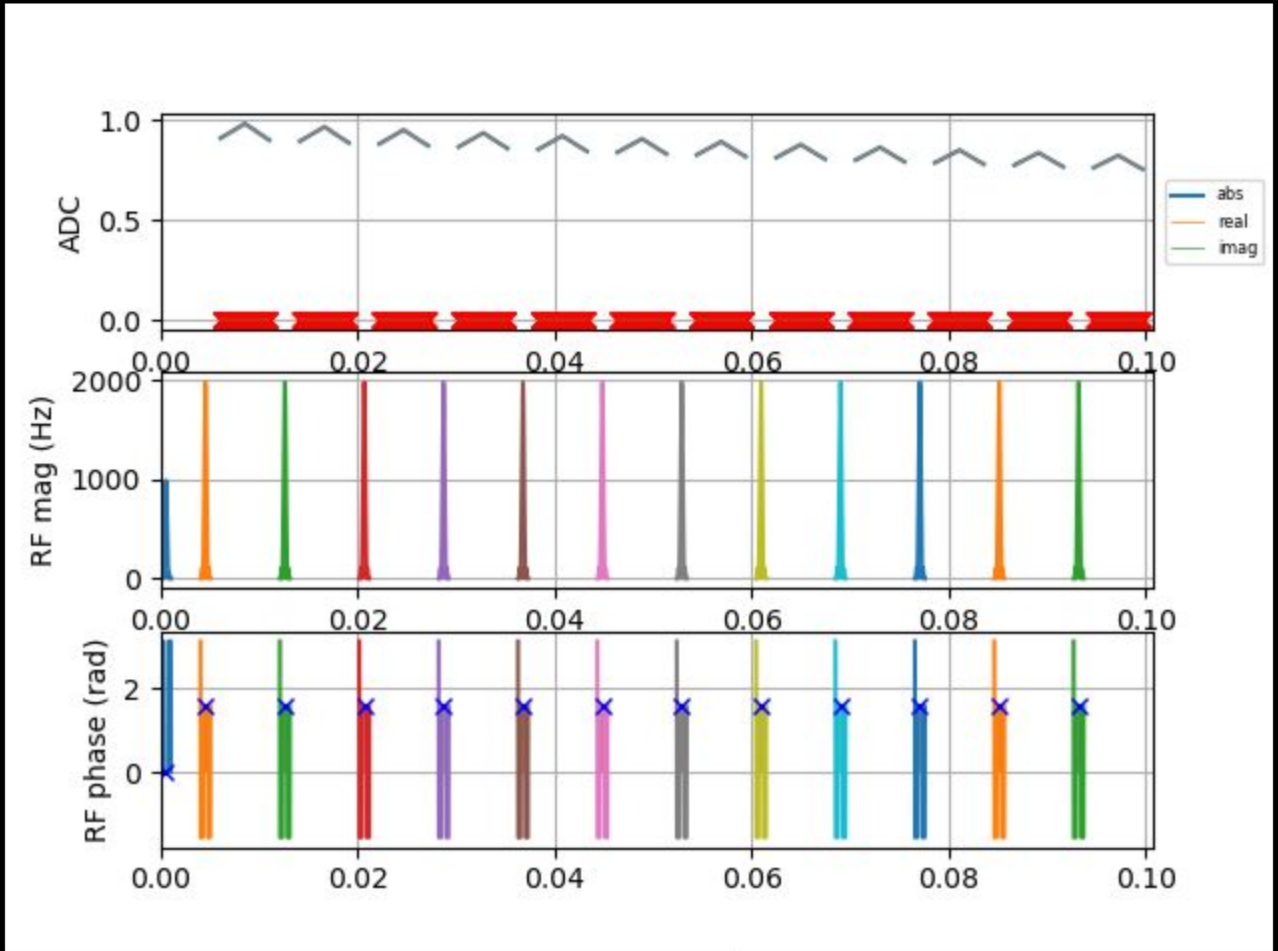


Interesting behavior:

- No Monte-Carlo noise, but missing signal
- Flexible threshold
- Faster with more diffusion, lower T2, shorter sequences
- “Regular” sequences are “detected” (EPG ⊆ PDG)
- Sequences with large, but dephased states benefit strongly from PDG

EPG ⊆ PDG and PDG > EPG for a simple example

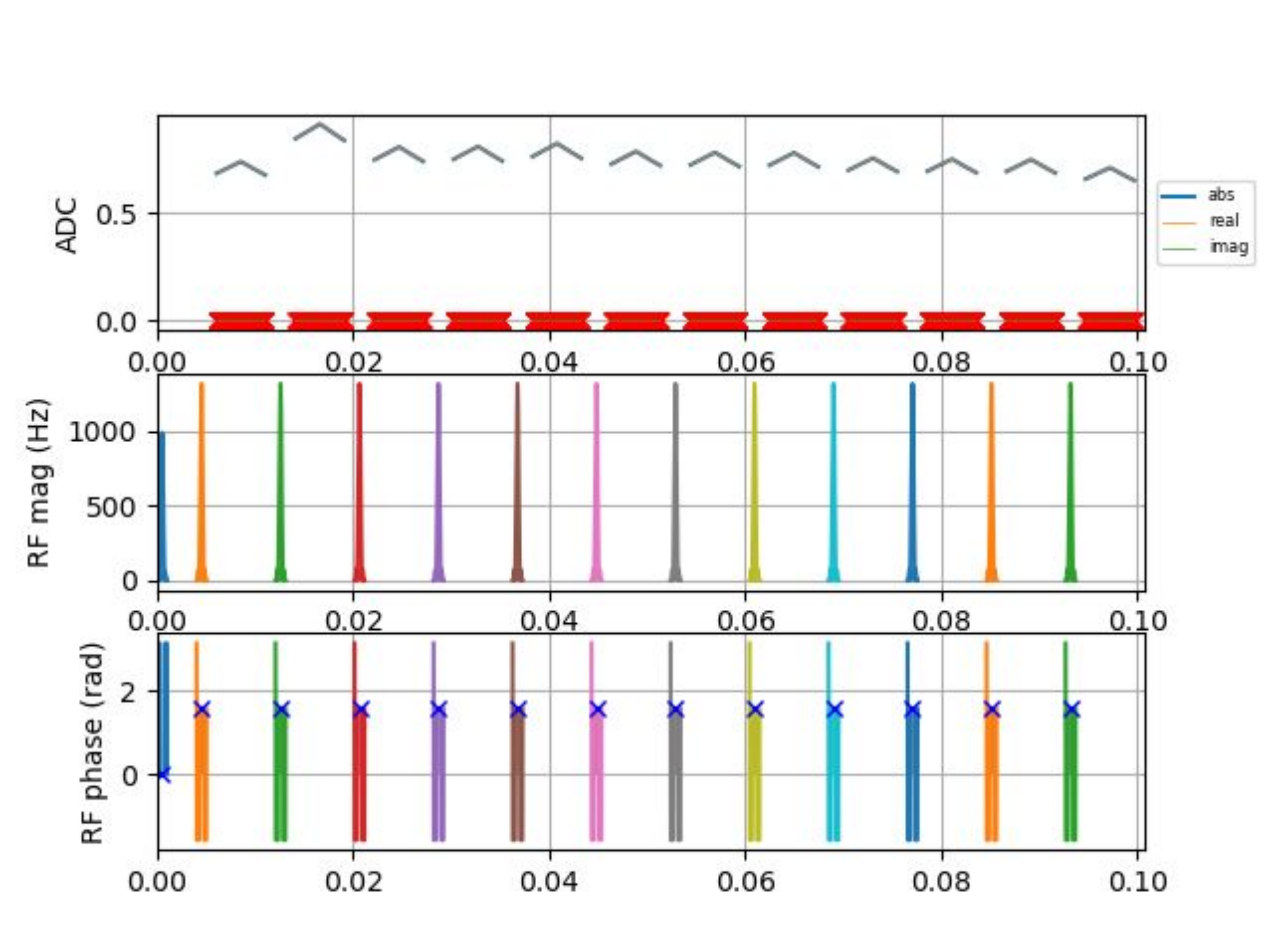
- EPG is most famous for describing TSE trains like this one:



For this 90° -180° case this is an exponential decay, but EPG can also describe e.g. 90°-120° TSEs, where stimulated echoes contribute.

EPG ⊆ PDG and PDG > EPG for a simple example

- EPG is most famous for describing TSE trains like this one:

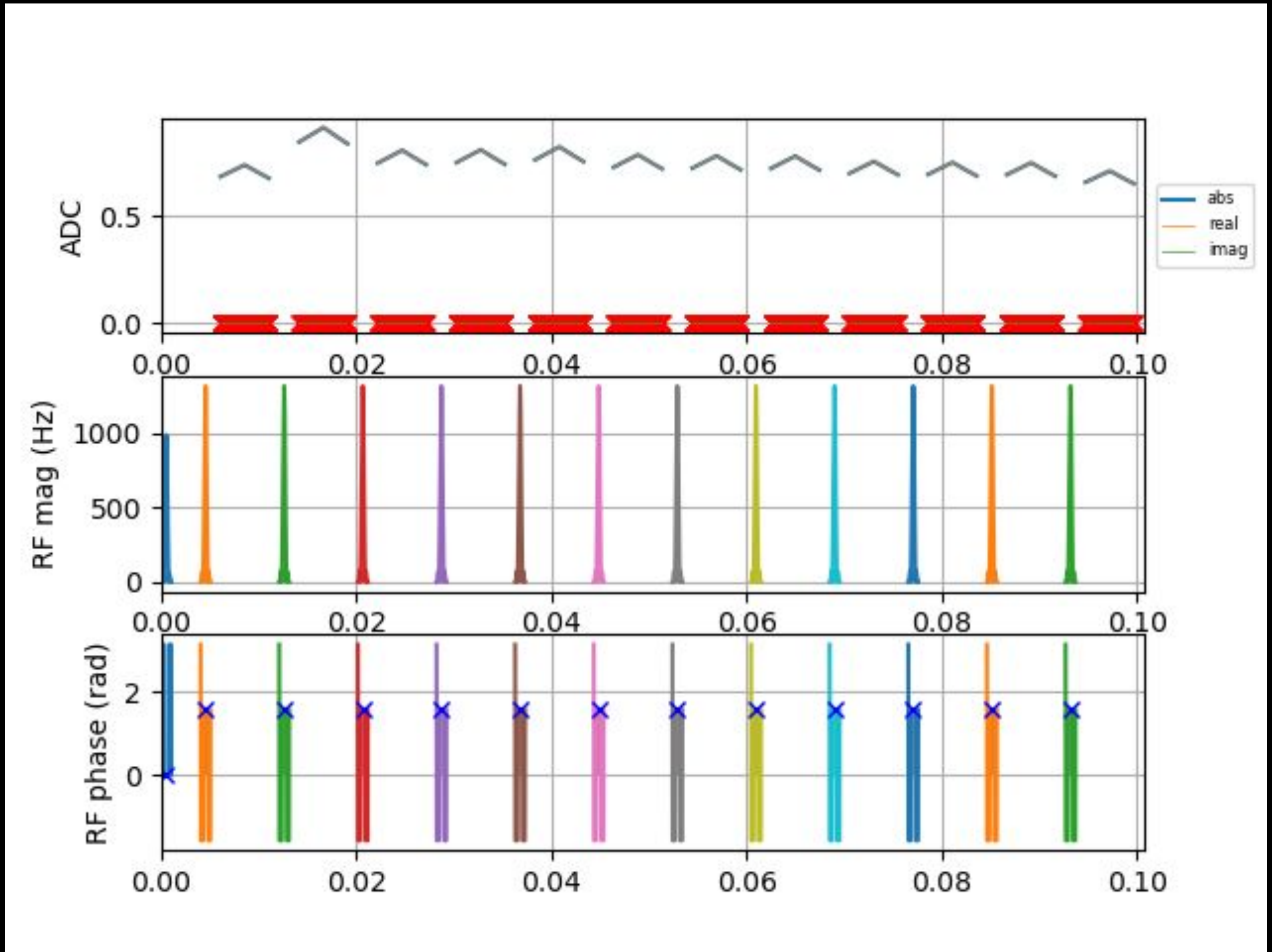


90°-120° TSE:
Stimulated echo contribution changes exp decay

-> described by EPG

EPG ⊆ PDG and PDG > EPG for a simple example

- EPG is most famous for describing TSE trains like this one:



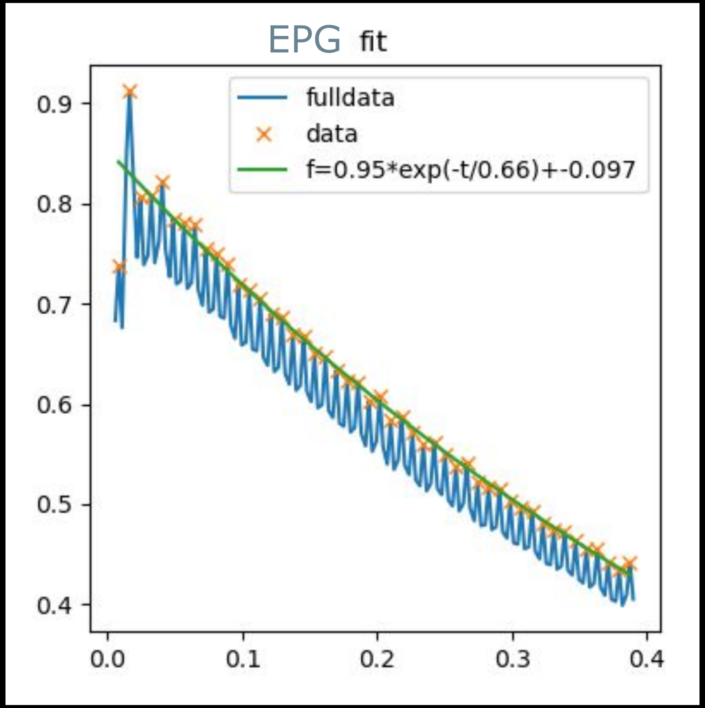
90°-120° TSE:
Stimulated echo contribution changes exp decay

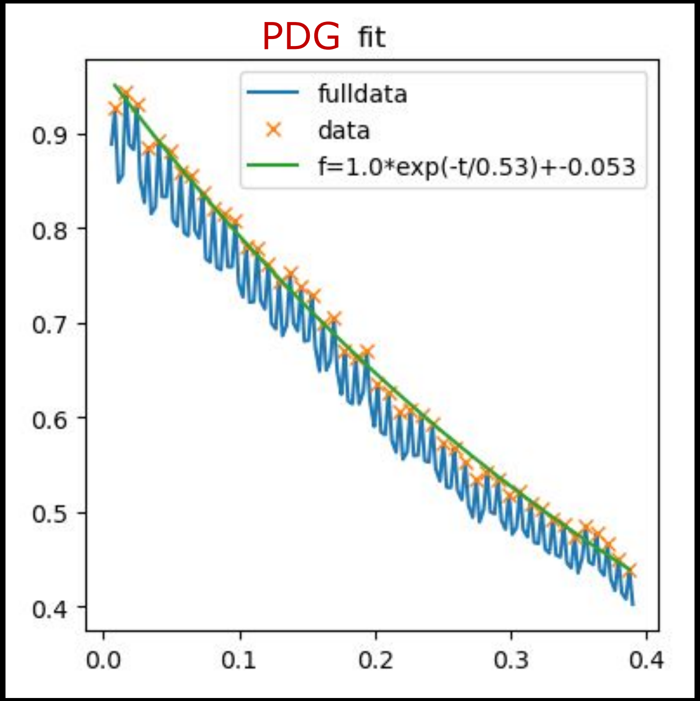
-> described by EPG

Let’s look at the exact same sequence with PDG!

EPG ⊆ PDG and PDG > EPG for a simple example

- This even leads to a different decay rate!





90°-120° TSE:

Stimulated echo contribution changes exp decay

-> described by EPG

FID + SE contribution
Also changes exp decay

-> not described by EPG
-> described by PDG