

# Overview of the Pulseq Approach

Maxim Zaitsev

*Division of Medical Physics, Dept. of Radiology,  
University Medical Center Freiburg, Germany*

# Legacy approach to sequence programming

---

- Get a template sequence from the vendor  
**Challenge:** Time and paperwork  
**Why?** Implementation details even for basic things are often unknown  
**Actual challenge:** Generic opaqueness and secrecy in the MR field
- Implement additional features  
**Challenge:** complexity of product sequences  
**Challenge:** focus on a subset of features without breaking the whole thing  
**Challenge:** understanding / debugging is difficult due to reliance on libraries with inaccessible source code
- **Repeat from 0 for additional vendors or new software versions**



# Pulseq Approach

---

- Minimize effort for implementation and support on hardware
  - Lean sequence-to-hardware interface
- Remove the thresholds in sequence programming
  - Make simple things truly simple
- Make researcher-oriented features accessible
  - Arbitrary gradients, arbitrary RF, free ordering, X-nuclei, ...
- Prevent typical sources of (human) errors
  - Avoid timing errors with “overlapping” gradients
  - Make data flag and counter setting optional/unnecessary
- **Promote open-source thinking, sharing and exchange!**



Image source: wikipedia.org

# What is *Pulseq*?

---

- *Pulseq* is a language to describe MR pulse sequences
- *Pulseq* sequences are fixed successions of RF and gradient pulses and ADC events



- *Pulseq* is the software to generate such pulse sequence descriptions
  - *Pulseq* scripts can re-generate *Pulseq* sequences to accommodate user input

*Pulseq* ecosystem includes sequences, software to generate them and software and hardware to consume them

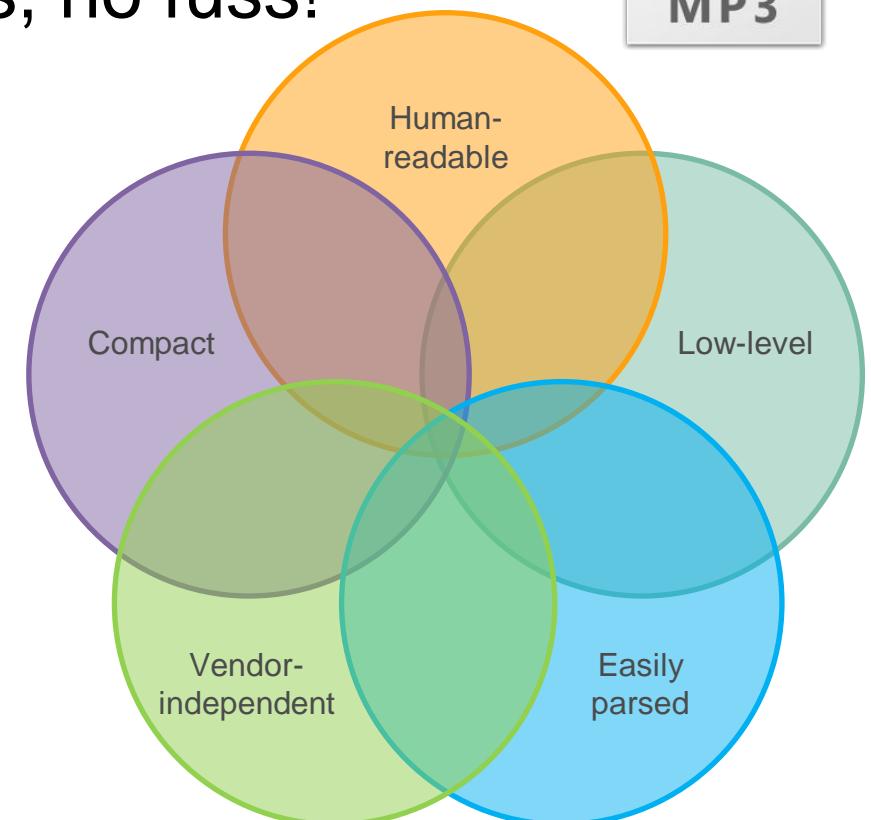
# *Pulseq* : pieces of the puzzle

---

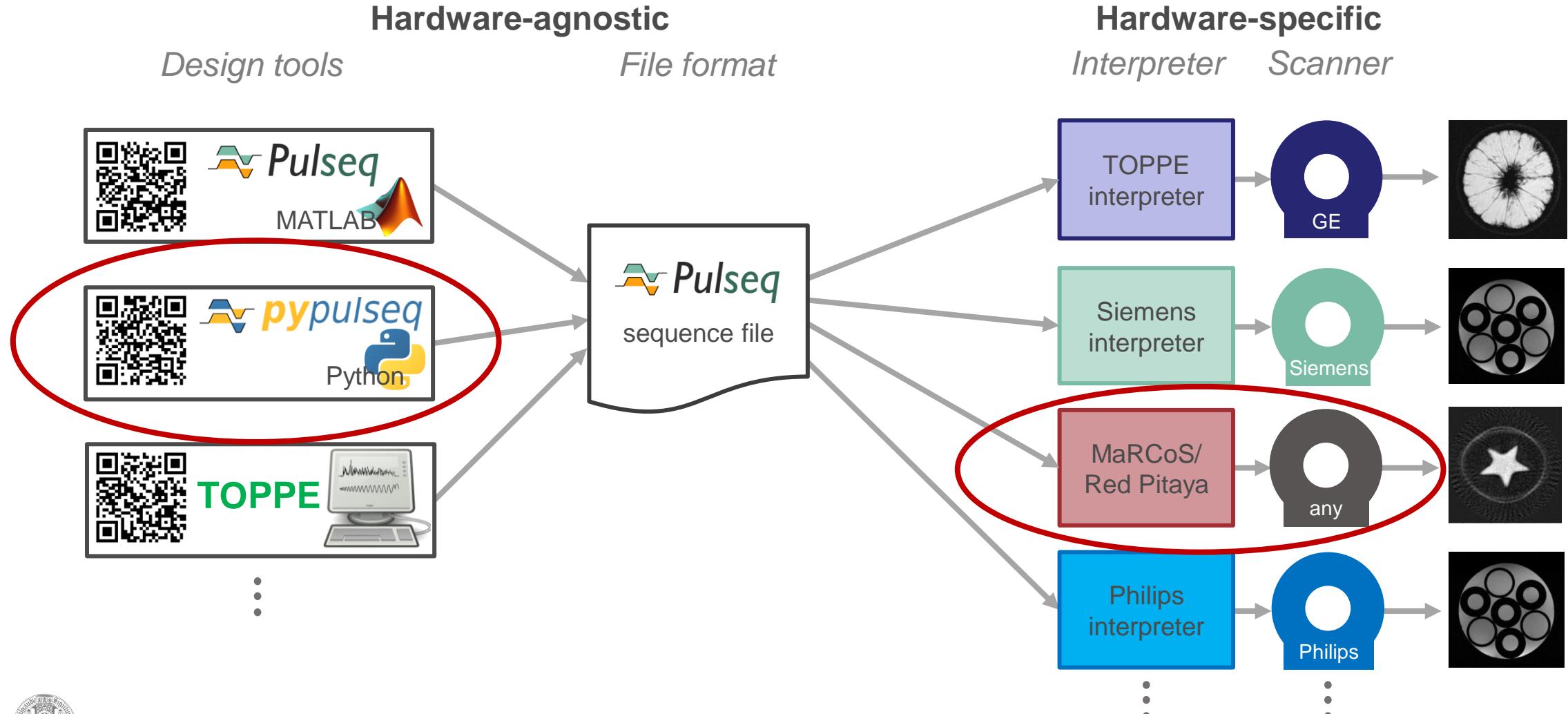


# *Pulseq* file internals

- Explicit (low level) specification of the pulse sequence
  - Think of an MP3 file (or more precisely lossless FLAC)
- No loops, no parameters, no dependencies, no fuss!
- Text file (human-readable)
  - Simple hierarchy  
(RF pulses, gradients, shapes)
  - Event table keeps it together
  - See <http://pulseq.github.io/specification.pdf> for more details

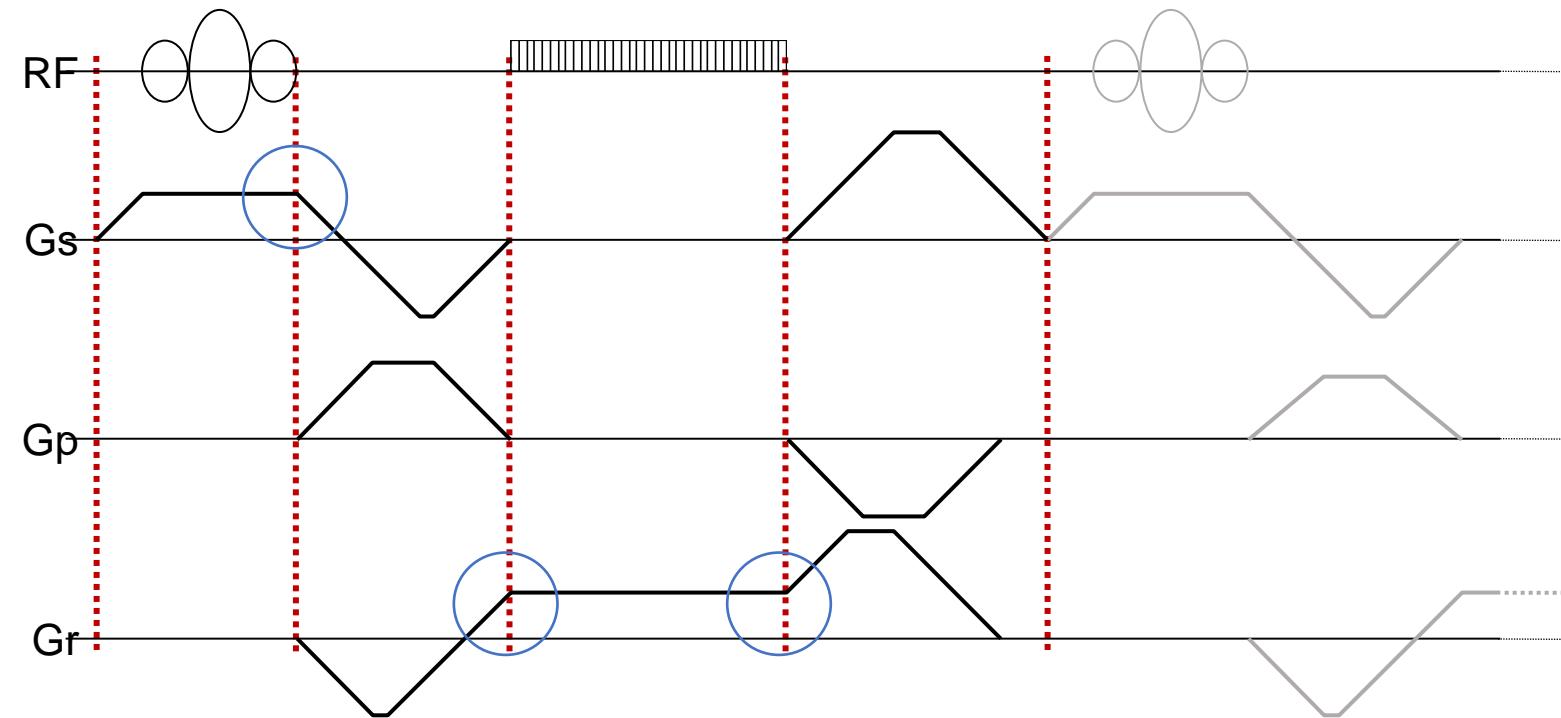


# Pulseq framework overview



# Pulse sequence definition in *Pulseq*

Concatenation of non-overlapping blocks



- Block 1: gradient and RF
- Block 2: only gradients
- Block 3: gradient and ADC
- Block 4: only gradients
- Block 5: gradient and RF ...

Gradients do not have to start or end at 0 at the block boundaries

see <http://pulseq.github.io/specification.pdf> for more details

# *Pulseq* block concept in detail

---

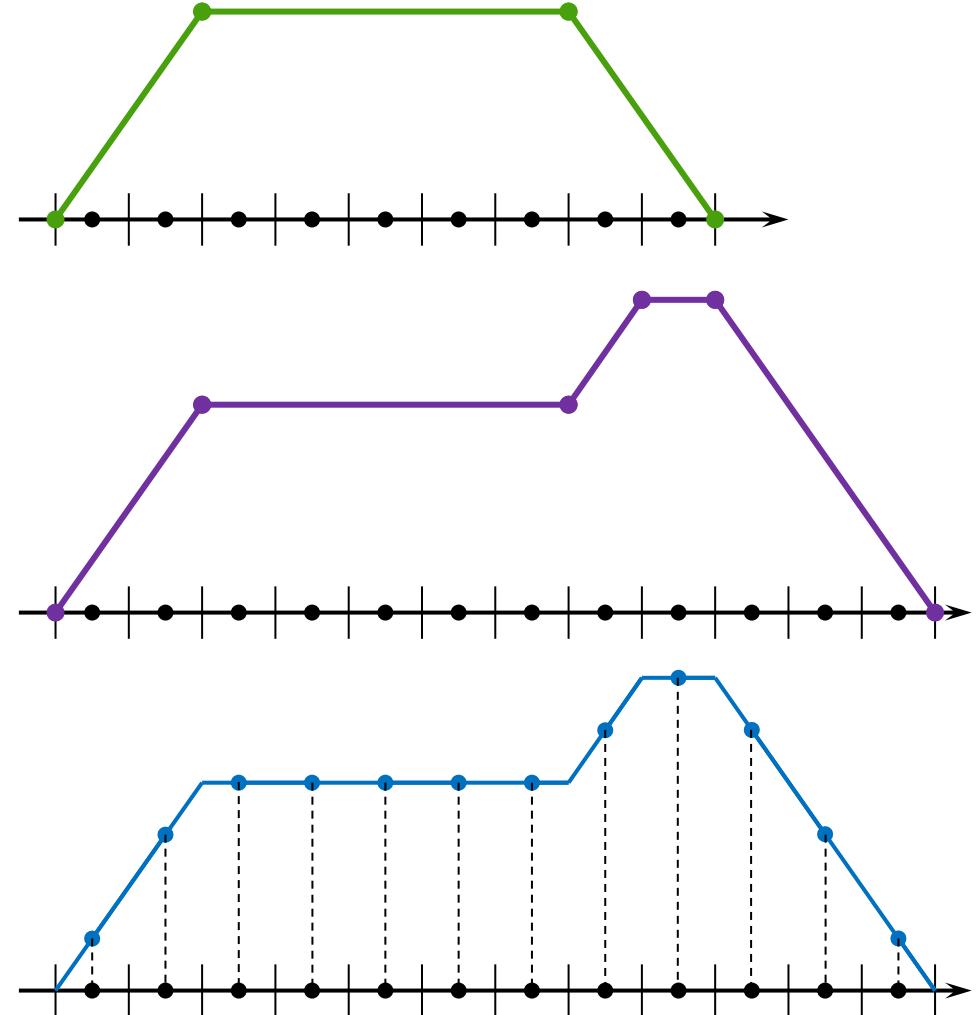
- Each block may contain following events:
  - One optional gradient pulse per axis
  - One optional RF pulse
  - One optional ADC event
- Individual events may define own start delays
- All events in the block overlap in time
- Duration of the block is defined by the longest event
  - Matlab/Python toolboxes use “dummy” delay objects to make blocks longer
- Explicit sequence description
  - No loops, no dependent parameters



# Shapes in *Pulseq*

---

- Conventional trapezoid gradients
- Two types of arbitrary shapes for both RF and gradient pulses:
  - Shapes with explicit time vectors
    - Hard (non-selective) RF pulses
    - **Extended trapezoids**
  - Shapes with a regular sampling on a fixed time grid
    - Most slice-selective / shaped RF pulses
    - Gradients for many non-Cartesian readouts (e.g. spirals)



# How to design a sequence in *Pulseq*

---

## **conceptual design steps**

- Step 1: split the time axis into blocks
- Step 2: assign events to the blocks

## **practical implementation steps**

- Step 3: create/calculate all events
- Step 4: populate the blocks and add them to the sequence

## **validation steps**

- Step 5: check timing, verify k-space trajectory, hardware and PNS limits, etc



# Matlab *Pulseq* workflow

```
fov = 256e-3; slThck = 5e-3;
Nx = 128; Ny = Nx;
TE = 10e-3; TR = 20e-3; alpha=15;

sys = mr.opts('MaxGrad',20,'GradUnit','mT/m',...
    'MaxSlew',100,'SlewUnit','T/m/s',...
    'rfRingdownTime', 20e-6, 'rfDeadtime', 100e-6, 'setAsDefault', true);
seq=mr.Sequence(sys);

[rf, gz, gzReph] = mr.makeSincPulse(alpha*pi/180, 'Duration', 4e-3, ...
    'SliceThickness', slThck, 'apodization', 0.5, 'timeBwProduct', 4);

gx = mr.makeTrapezoid('x', 'FlatArea', Nx/fov, 'FlatTime', 6.4e-3);
adc = mr.makeAdc(Nx, 'Duration', gx.flatTime, 'Delay', gx.riseTime);
gxPre = mr.makeTrapezoid('x', 'Area', -gx.area/2, 'Duration', 2e-3);
gyPre = mr.makeTrapezoid('y', 'Area', Ny/2/fov, 'Duration', 2e-3);

phaseAreas = (0:Ny-1)/(Ny/2)-1;
delayTE = round((TE - mr.calcDuration(gxPre) - mr.calcDuration(gz)/2 ...
    - mr.calcDuration(gx)/2)/seq.gradRasterTime)*seq.gradRasterTime;
delayTR = round((TR - mr.calcDuration(gxPre) - mr.calcDuration(gz) ...
    - mr.calcDuration(gx) - delayTE)/seq.gradRasterTime)*seq.gradRasterTime;

for i=1:Ny
    seq.addBlock(rf, gz);
    seq.addBlock(gxPre, mr.scaleGrad(gyPre,phaseAreas(i)), gzReph);
    seq.addBlock(delayTE);
    seq.addBlock(gx, adc);
    seq.addBlock(delayTR);
end

seq.setDefinition('FOV', [fov fov slThck]);
seq.setDefinition('Name', 'mini_gre');

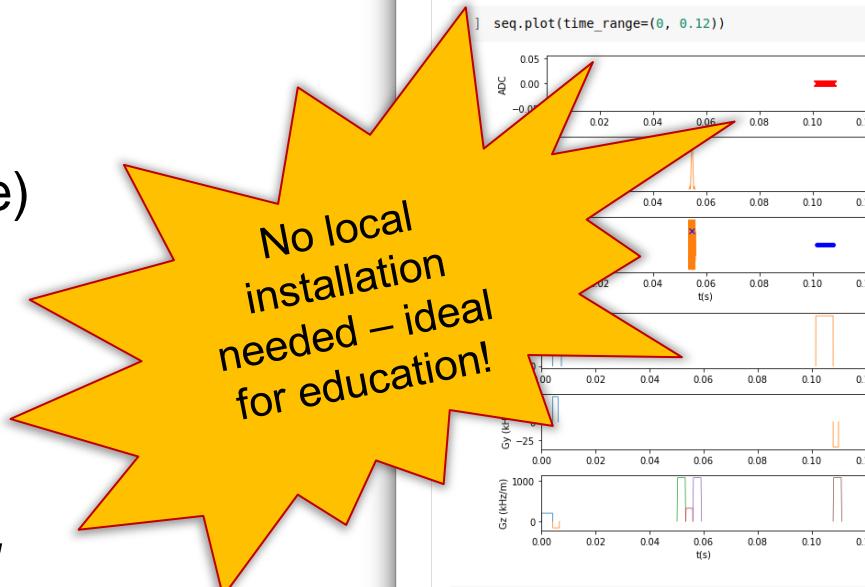
seq.write('mini_gre.seq')
```

a runnable gradient echo sequence code  
(similar to Siemens' example *miniFlash*)

- Define high-level parameters (convenience)
- Define the system properties
- Define pulses and ADC objects used in the sequence
- Calculate the delays and reordering tables
- Loop and define sequence blocks
  - Duration of each block is defined by the duration of the longest event
- Prepare sequence export and write it out
- Copy ‘\*.seq’ to the scanner and run it!

# PyPulseq workflow

- PyPulseq is a close replica of the original Pulseq toolbox that does not require a MATLAB license
- Runs in many Python environments, e.g. as notebook in Jupyter (<http://jupyter.org/>) or Google Colaboratory
- Identical workflow:
  - Define the system properties
  - Define high-level parameters (convenience)
  - Define pulses and ADC objects used in the sequence
  - Calculate the delays and reordering tables
  - Loop and define sequence blocks
  - *Download ‘\*.seq’ to the scanner and run it!*



```
# (re)initialize the sequence object
seq = Sequence(system)
# Prepare RF offsets. This is required for multi-slice acquisition
delta_z = n_slices * slice_gap
z = np.linspace(-delta_z / 2), (delta_z / 2), n_slices) + rf_offset

for k in range(ns): # Averages
    for j in range(n_slices): # Slices
        # Apply RF offsets
        freq_offset = gz90.amplitude * z[j]
        rf90.freq_offset = freq_offset

        freq_offset = gz180.amplitude * z[j]
        rf180.freq_offset = freq_offset

        for i in range(Ny): # Phase encodes
            seq.add_block(rf90, gz90)
            gy_pre = make_trapezoid(channel='y', system=system,
                                    area=phase_areas[-i - 1], duration=2e-3)
            seq.add_block(gx_pre, gy_pre, gz_reph)
            seq.add_block(delay1)
            seq.add_block(gz_spoil)
            seq.add_block(rf180, gz180)
            seq.add_block(gz_spoil)
            seq.add_block(delay2)
            seq.add_block(gx, adc)
            gy_pre = make_trapezoid(channel='y', system=system,
                                    area=phase_areas[-j - 1], duration=2e-3)
            seq.add_block(gy_pre, gz_spoil)
            seq.add_block(delay_TR)

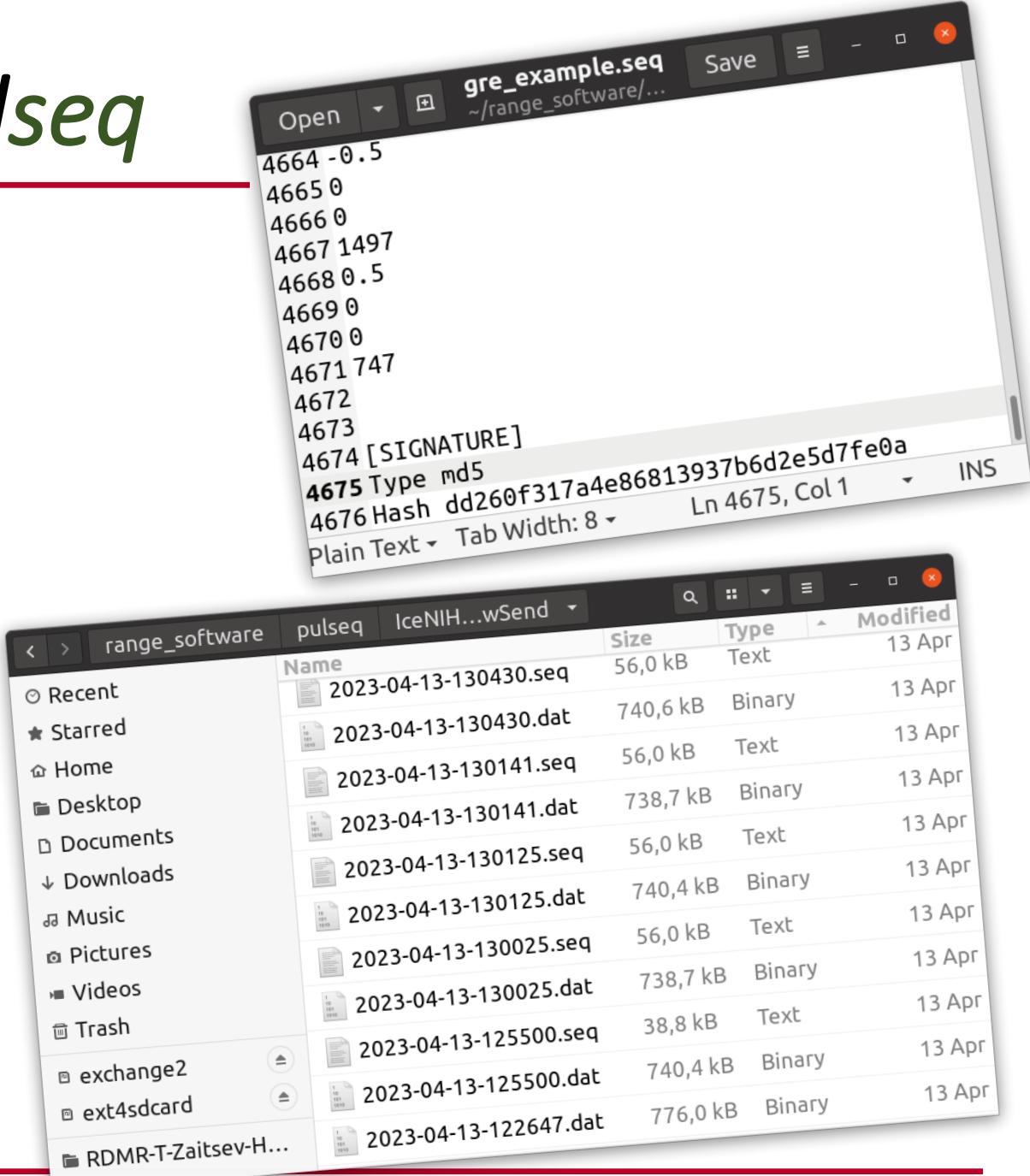
seq.plot(time_range=(0, 0.12))

seq.check_timing()
```

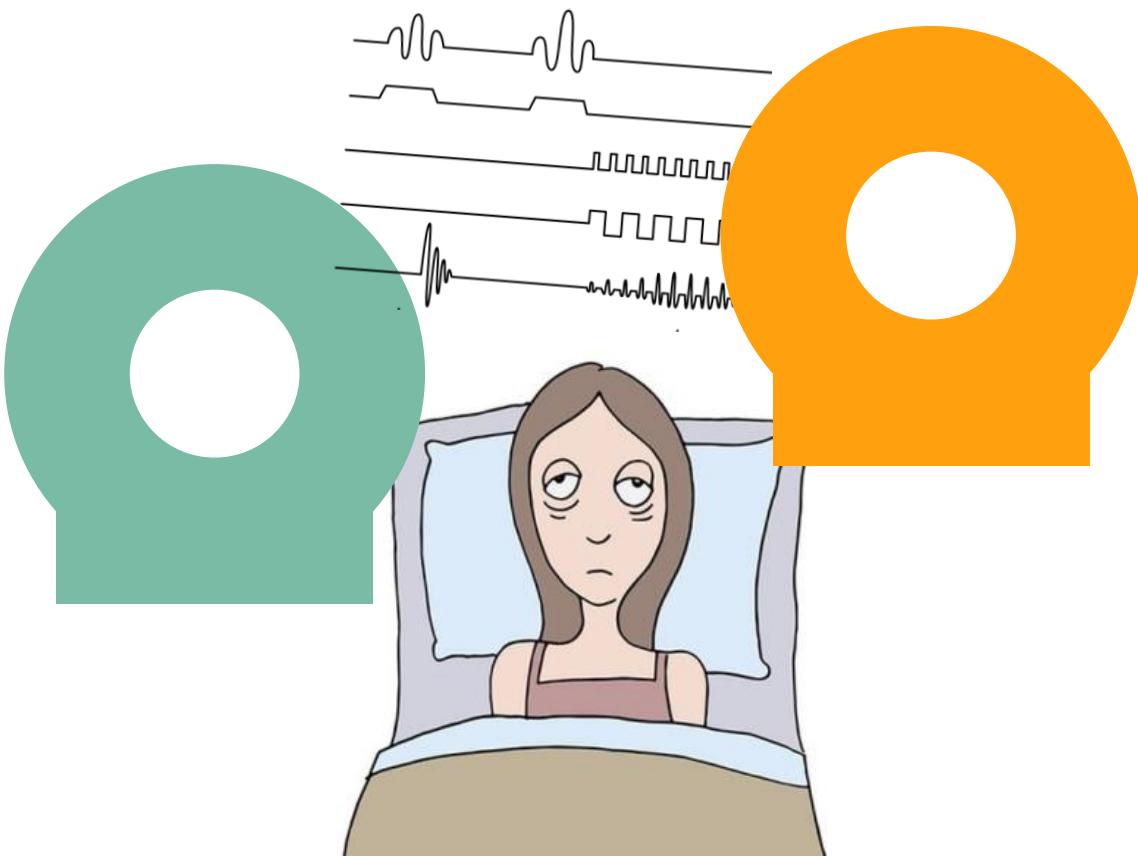


# Reproducibility with *Pulseq*

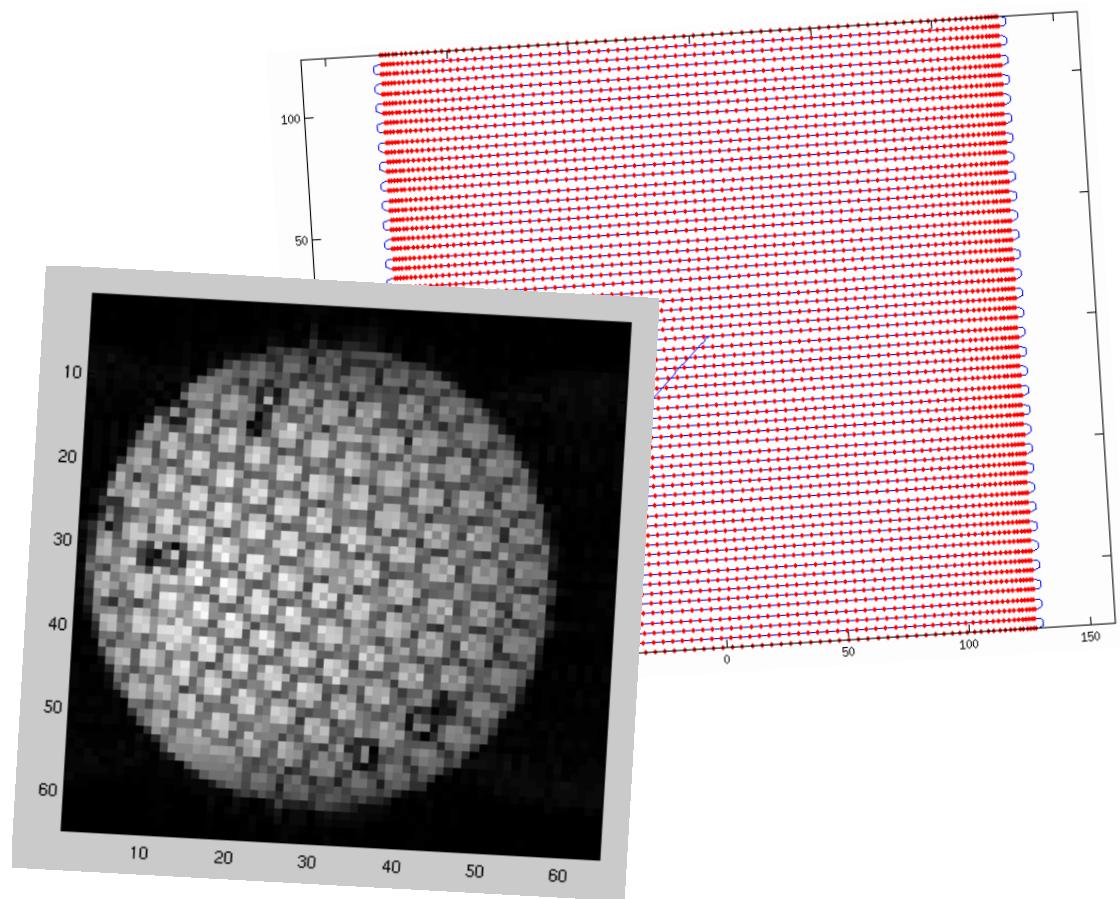
- All Pulseq sequences are signed with a 128-bit hash
- Siemens RAW data contain the Pulseq hash
  - Verify where the data were acquired with the given sequence
- In practice: always archive sequences along with the data
  - Tutorial recon scripts use .seq files to extract k-space trajectories and reordering information



# Prototyping from scratch in *Pulseq*



...dream of a sequence in the morning...

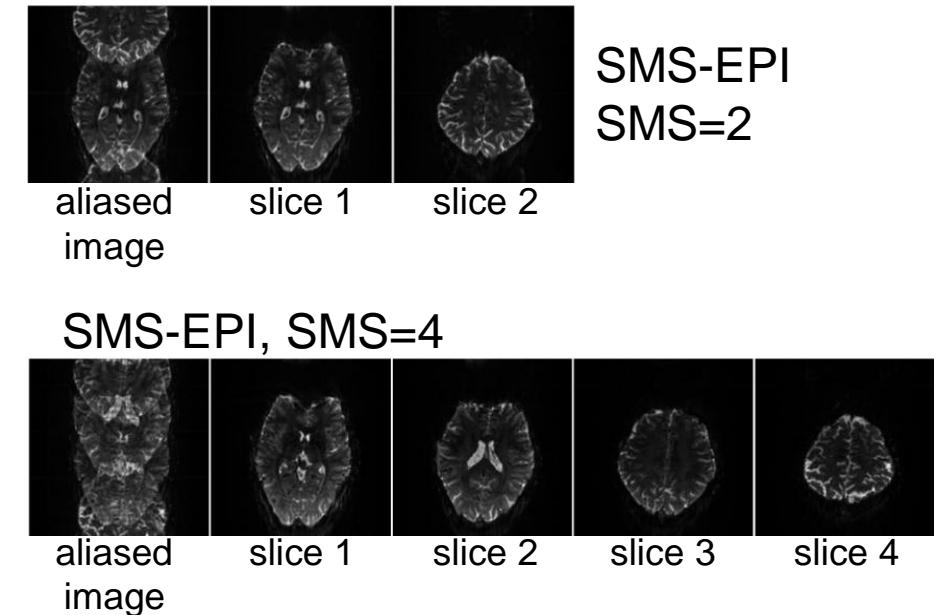
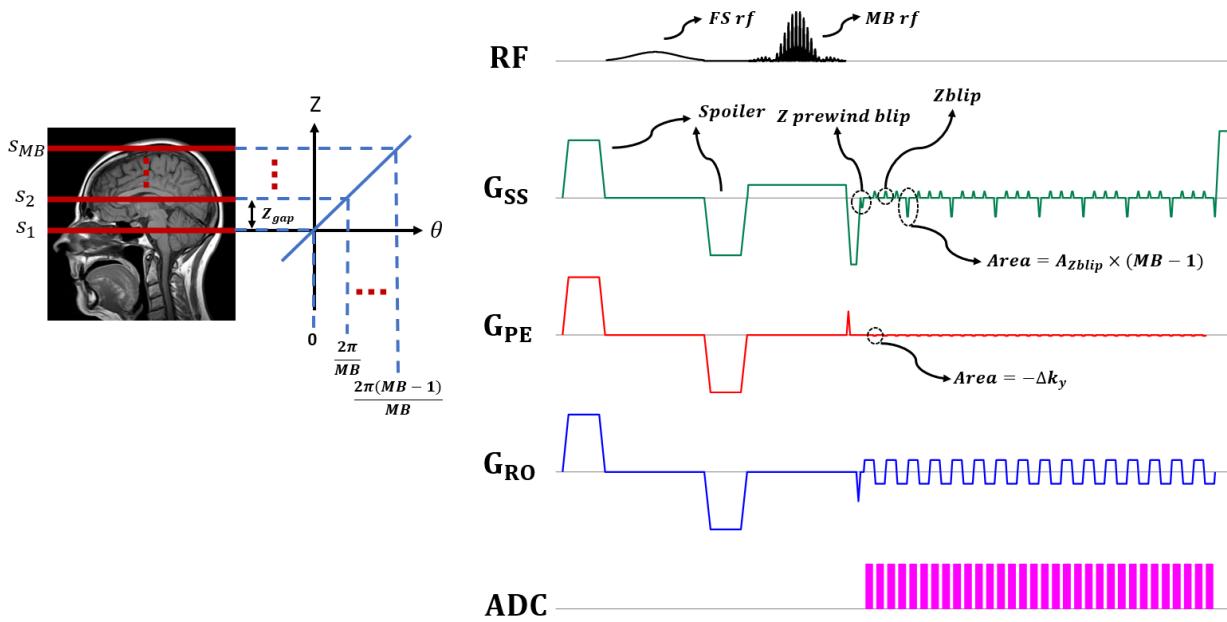


...check your data in the afternoon!



# Educational SMS EPI with CAIPI

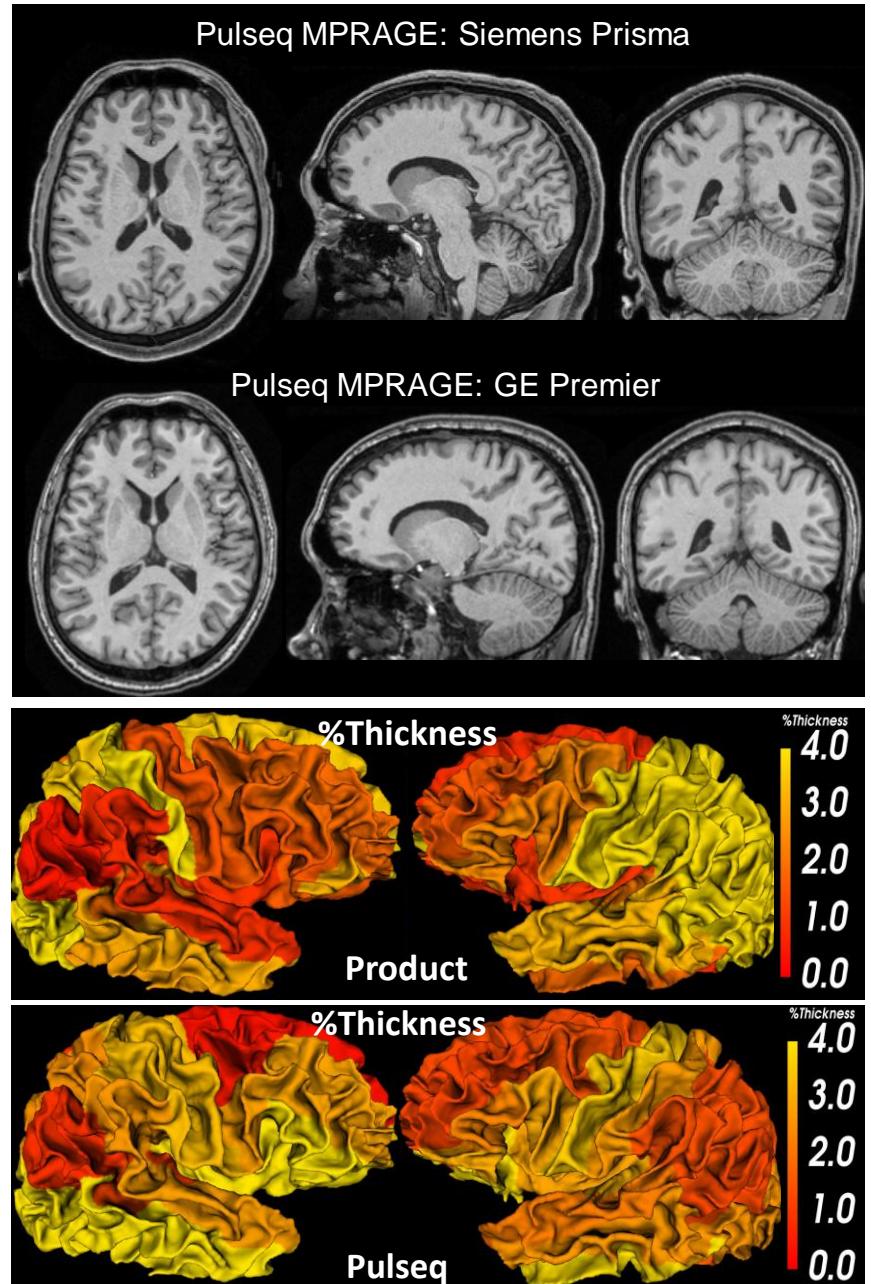
- Echo-planar imaging with simultaneous multi-slice excitation
- Complete calibration, acquisition and reconstruction scripts
- Presented at the ESMRMB 2024 in Barcelona\*



\*Shafiekhani M et al, ESMRMB 2024 proceedings #118

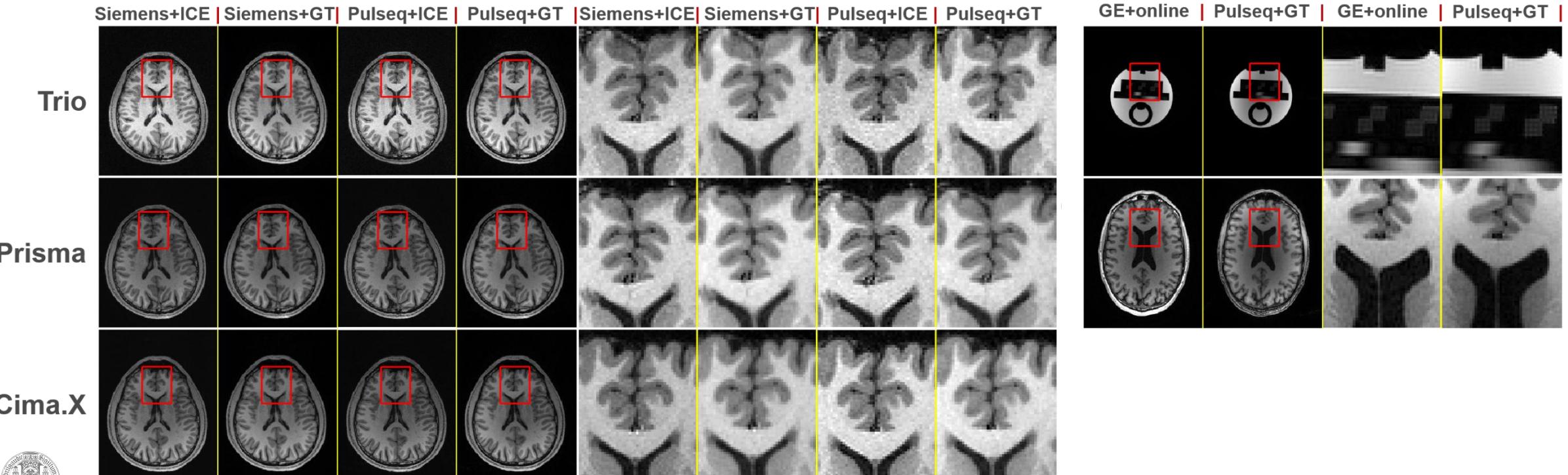
# Pulseq is not a toy

- Growing user community
  - Over 200 labs worldwide
- Great not only for prototyping or education
  - Multi-center MRI studies
- Identical 3D MPRAGE on Siemens & GE
  - Pulseq reduces cortical thickness variance  
*Rathi Y, Nielsen JF, Zaitsev M, Ning L, Chiou JR, Kim T, Grissom W, Gagoski B, Bilgic B, ISMRM 2022, #1618*



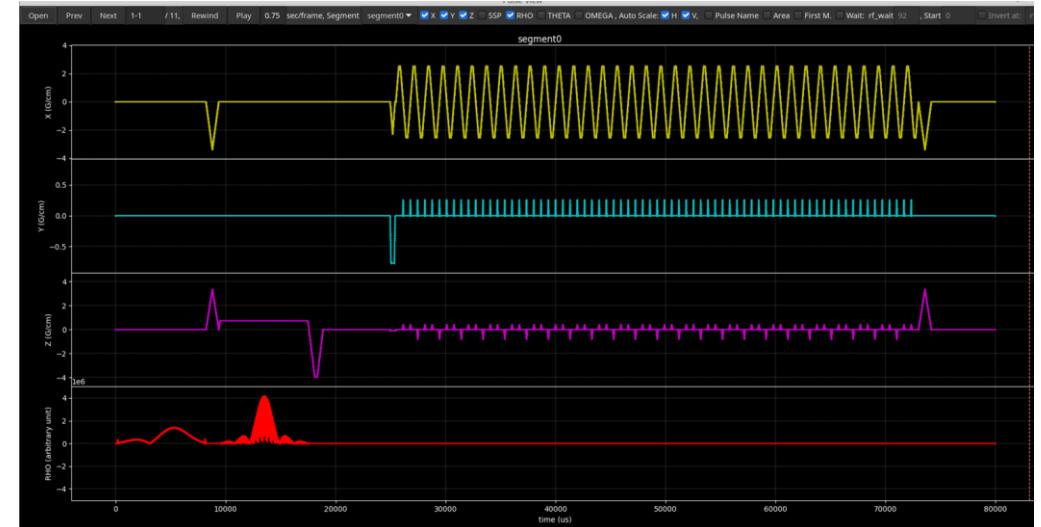
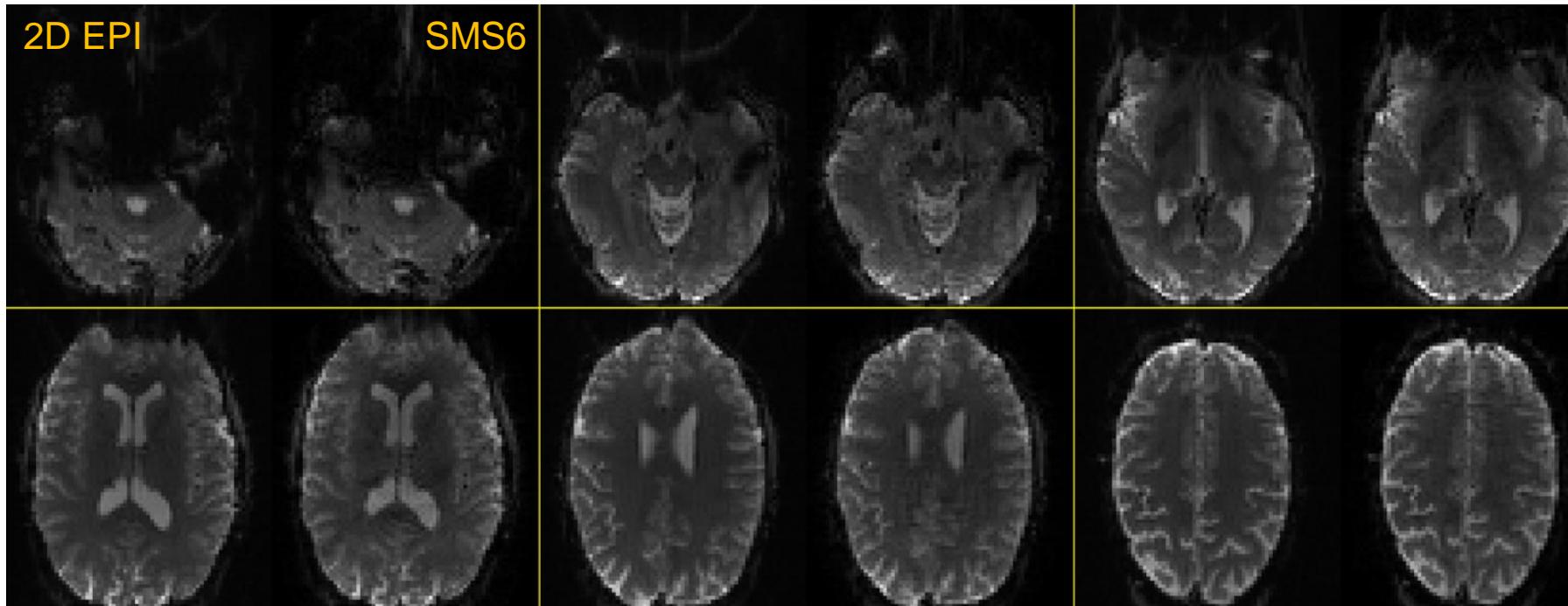
# MPRAGE with GRAPPA

- ISMRM reproducibility challenge 
- 3 sites, 2 vendors, 4 scanner models
- Image recon: either integrated Siemens (ICE) or Gadgetron



# fMRI-ready SMS EPI

- Target multicenter fMRI studies
  - Implemented ABCD-like fMRI protocol (SMS factor 6; 2.4 mm iso)
  - Tested on Siemens and GE



Pulseq SMS-EPI  
sequence simulated  
in GE's Pulse Studio

Pulseq  
2D-EPI vs. SMS-EPI  
Left: non-SMS (2D) EPI  
Right: EPI with SMS=6

# *Pulseq* is fun!

---

- mrMusic library for creating gradient music
- Located in demoUnsorted  
(lib + 5 demo scripts)
- Really easy to create nice 3-voice tunes
- Many more options available (explore examples)
  - Sound quality settings (wave vs. sawtooth)
  - Temperaments, Staccato & Legato
  - **Tuning for avoiding mechanical resonances**
- **Enjoy responsibly!**



# What's new in *Pulseq*

---

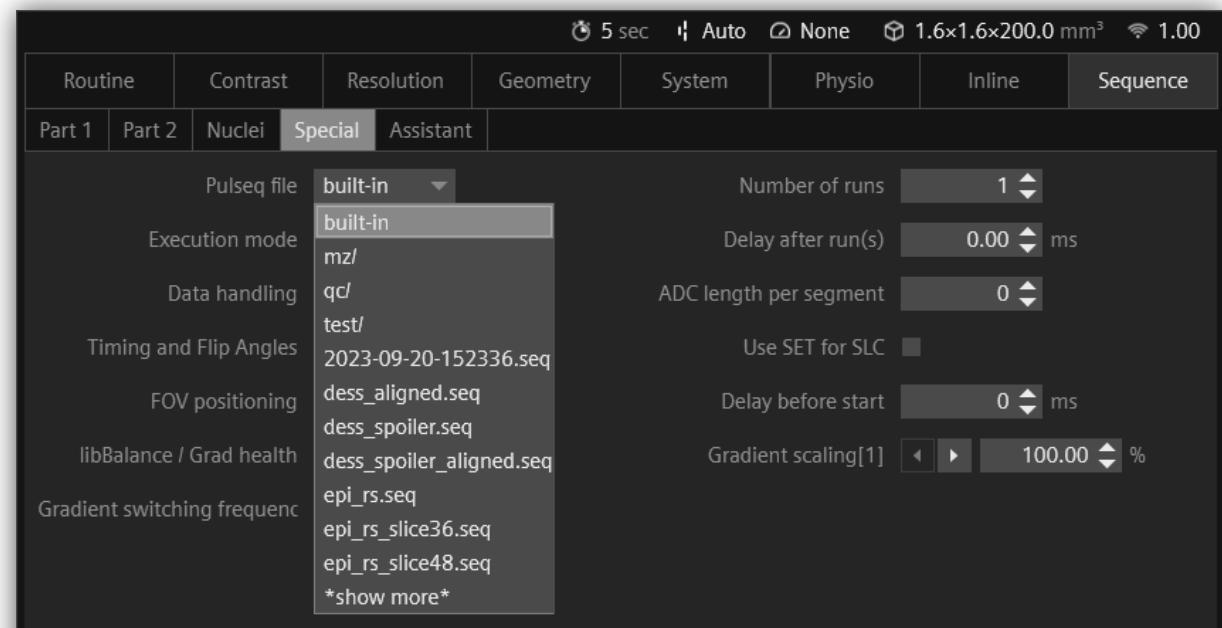
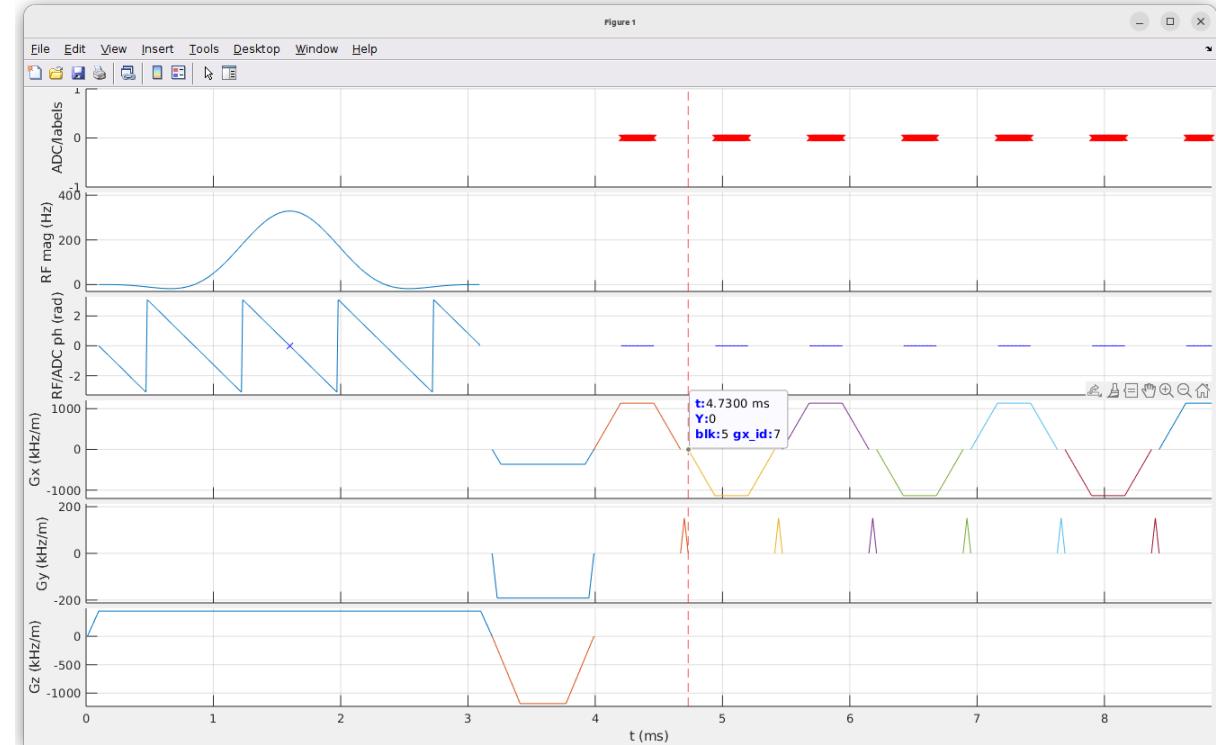
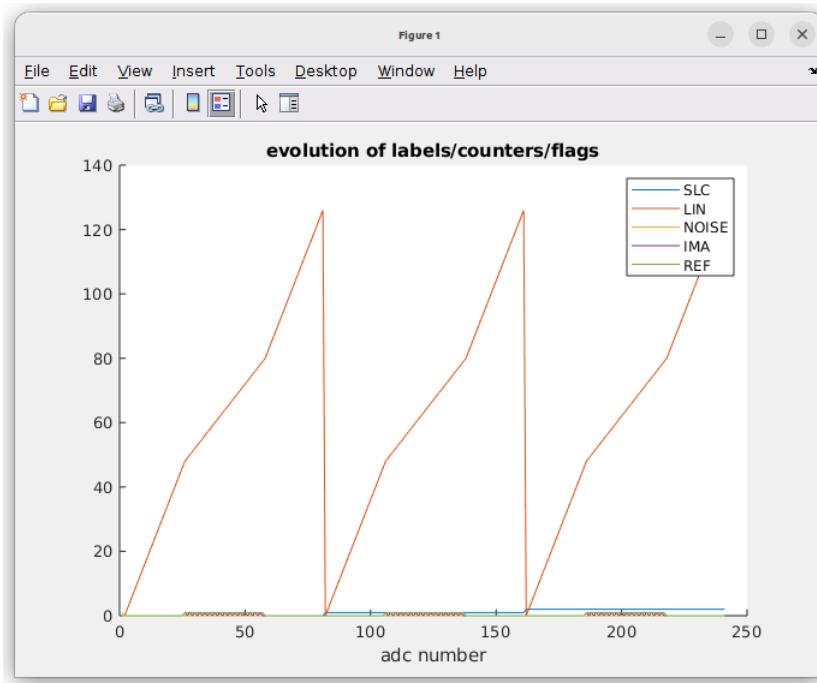
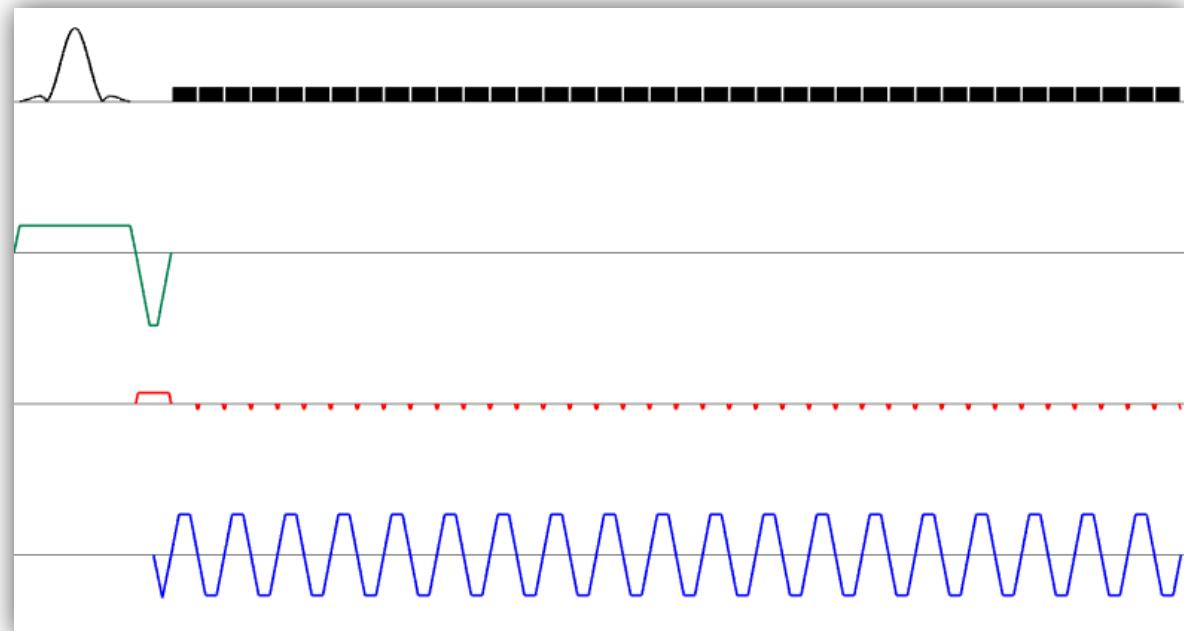
(Matlab Pulseq)

- Requested block duration in addBlock
- max RF amplitude as system property / RF amplitude warning
- (relative) SAR calculation
- Moments and b-factor calculation
- New display options: paperPlot() and plot('stacked')
- Plotting & debugging functions for labels & counters

(Siemens Interpreter)

- Install arbitrary number of sequences in different sub-directories





# *Pulseq* on Siemens platforms

---

- Over 150 C2P sites
- *Pulseq* works well on:
  - All Numaris4 platforms (tested on vb15...ve12u) and numerous hardware platforms (Symphony, Trio, 7T, 3T Connectom, Skyra, Prisma,Cima.X...)
  - All released NumarisX versions (xa11, xa20, xa30, xa40, xa5x, xa6x ...)



# *Pulseq* on Other Platforms

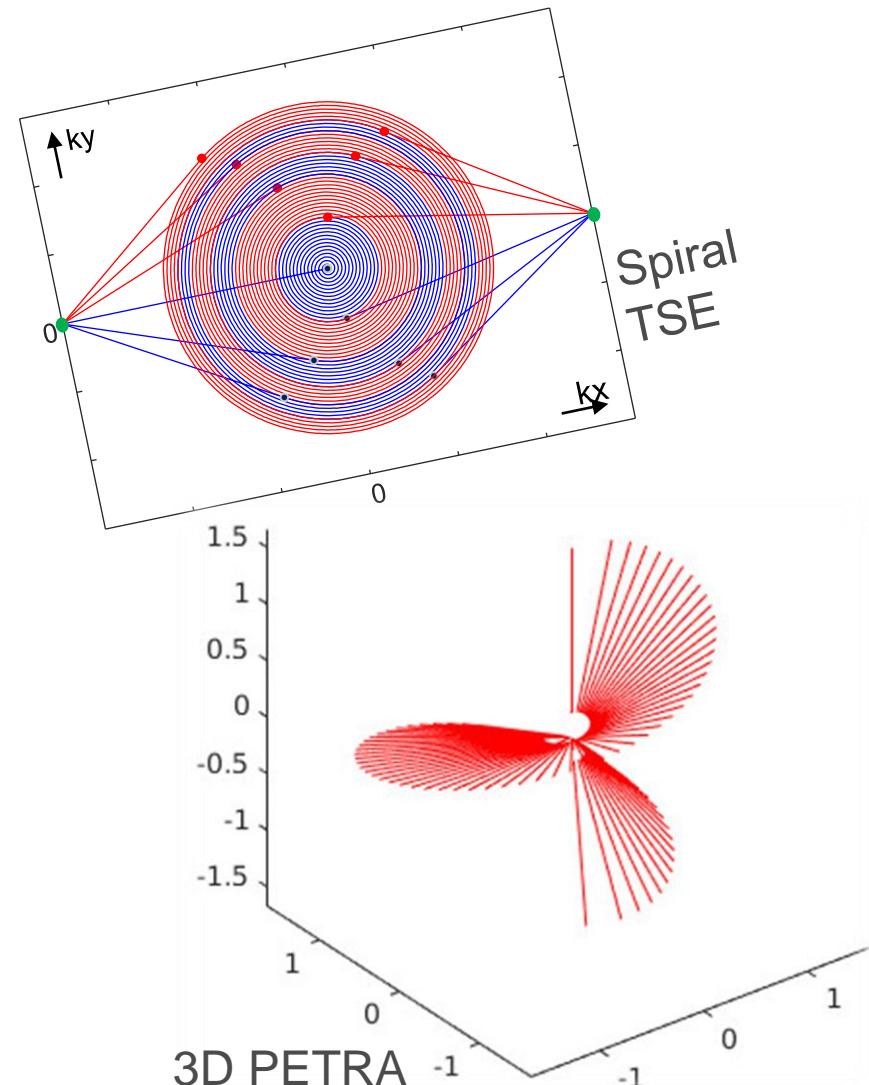
- Works very well on GE (Thanks to Jon-Fredrik Nielsen)
- MaRCoS / Red Pitaya / Portable System → → → →
- Reportedly works on United Imaging
- Two Philips interpreters @ ISMRM2024



# Cross-platform sequences with *Pulseq*

---

- MR physics-oriented workflow
  - Write your sequences from scratch
  - Non-Cartesian readouts, user-defined gradient shapes and custom RF pulses
  - Advanced visualization and analysis tools
  - Automatic k-space calculation
- Play out on many scanners
  - Siemens, GE, UI, MaRCoS...
  - Philips : ready to start
  - *Bruker* : we are talking to them...
- **Focus on sharing and open science**



Berkin Bilgic

Feng Jia

Jon-Fredrik Nielsen

Marten Veldmann

Niklas Wehkamp

Qingping Chen

Tony Stoecker

Borjan Gagoski

Imam Shaik

Juergen Hennig

Moritz Zaiss

Philipp Amrein

Scott Peltier

Will Grissom

Douglas Noll

Jeff Fessler

Lukas Winter

Naveen Murthy

Qiang Liu

Sebastian Littin

Yogesh Rathi

# THANK YOU FOR YOUR ATTENTION!



NIH U24 S120056  
NIH 1R01EB032378

EU MRITwins 101078393



Co-funded by  
the European Union

METROLOGY  
PARTNERSHIP



EURAMET 22HLT02 A4IM