UNIVERSITÄTS
KLINIKUM FREIBURG

Pulse Sequence Programming using *Pulseq*

Sebastian Littin, Qingping Chen,

Maxim Zaitsev

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

March 25th, 2024

What is a MRI pulse sequence?

- Method to control the MRI scanner
- Way of controlling RF, gradients and ADC
- Strategy to acquire MR signals
- What defines contrast
- …

What is *Pulseq*?

- *Pulseq* is a language to describe MR pulse sequences

- *Pulseq* is the software to generate such pulse sequence descriptions

- *Pulseq* sequences are fixed successions of RF and gradient pulses and ADC events

- *Pulseq* scripts can re-generate *Pulseq* sequences to accommodate user input

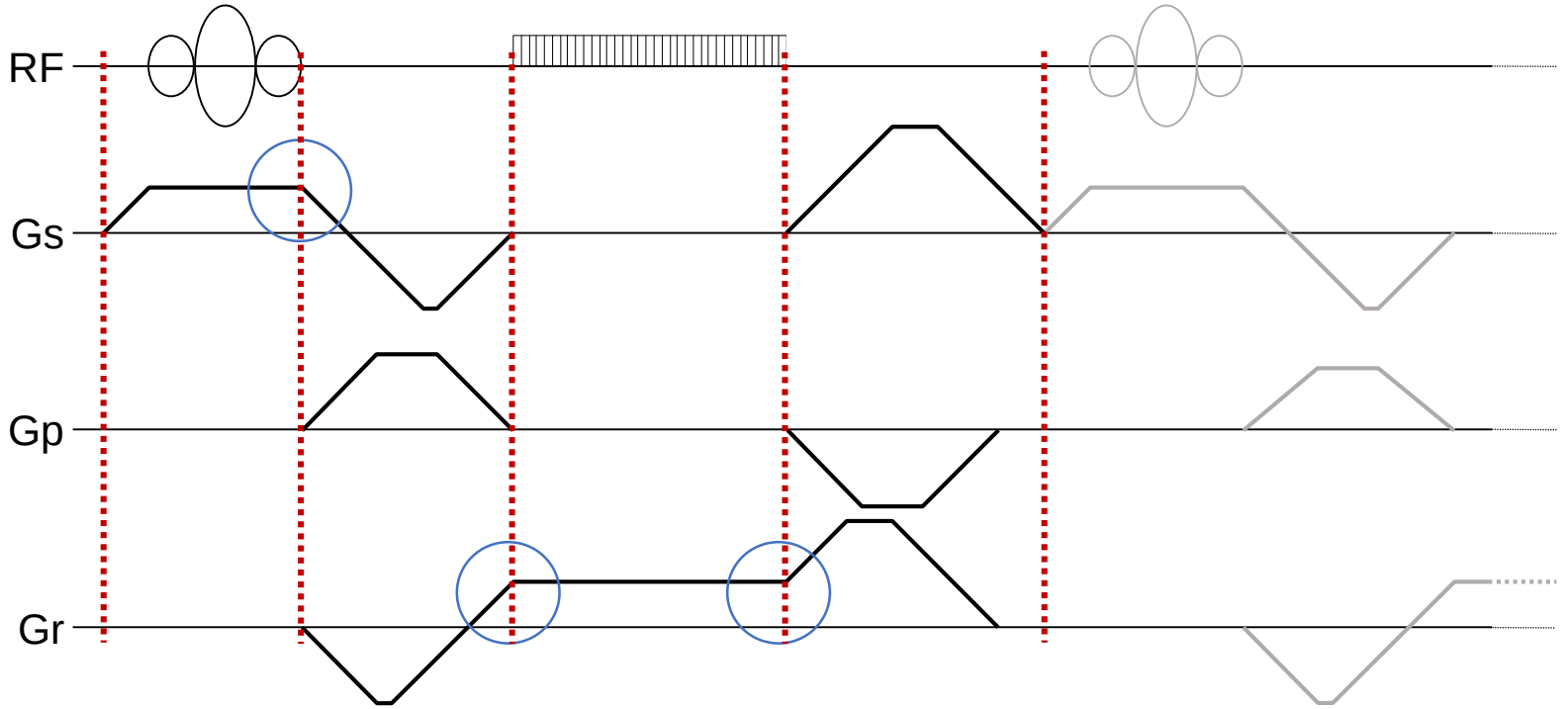
Pulseq ecosystem includes sequences, software to generate them and software and hardware to consume them

Pulseq Philosophy

- 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

Pulse sequence definition in *Pulseq*



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

- Sequence is a concatenation of non-overlapping blocks
- Gradients do not have to start or end at 0 at the block boundaries

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

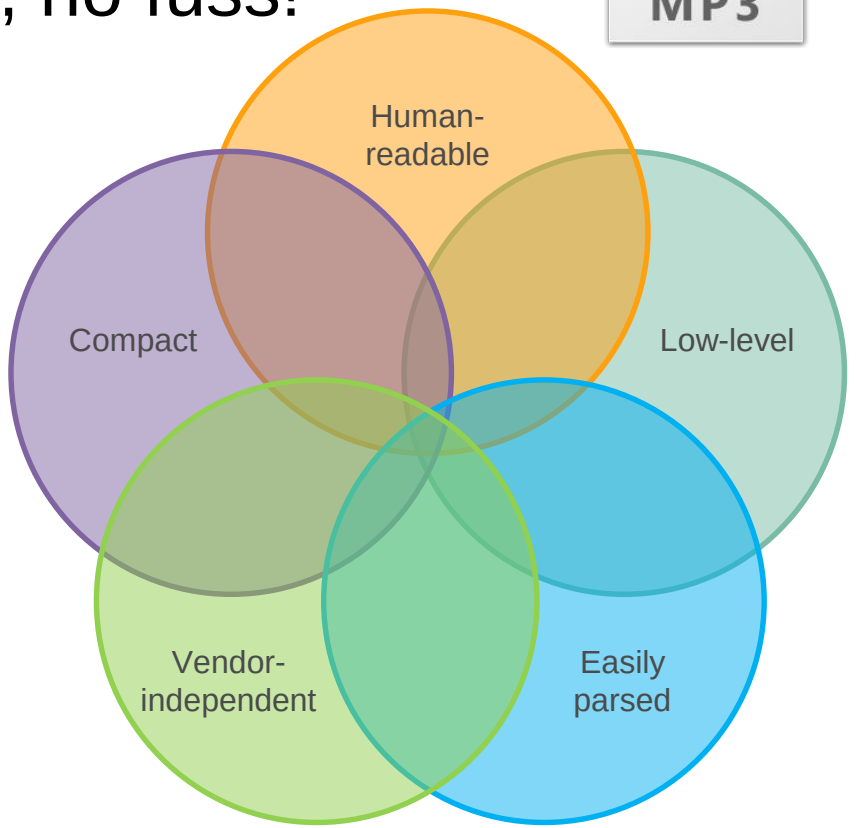
Pulseq block concept in detail

- 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 – like a recorded piece of music!

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



High-level programming environments

- Matlab *Pulseq* toolbox
- Python *PyPulseq* toolbox

- Further options
  - TOPPE is primarily targeted at GE but can import and export *Pulseq* files
  - GammaStar can export *pulseq* files
  - JEMRIS Bloch simulator can export *pulseq* files
  - CoreMRI Bloch simulator can export *pulseq* files
  - …

Matlab *Pulseq* workflow

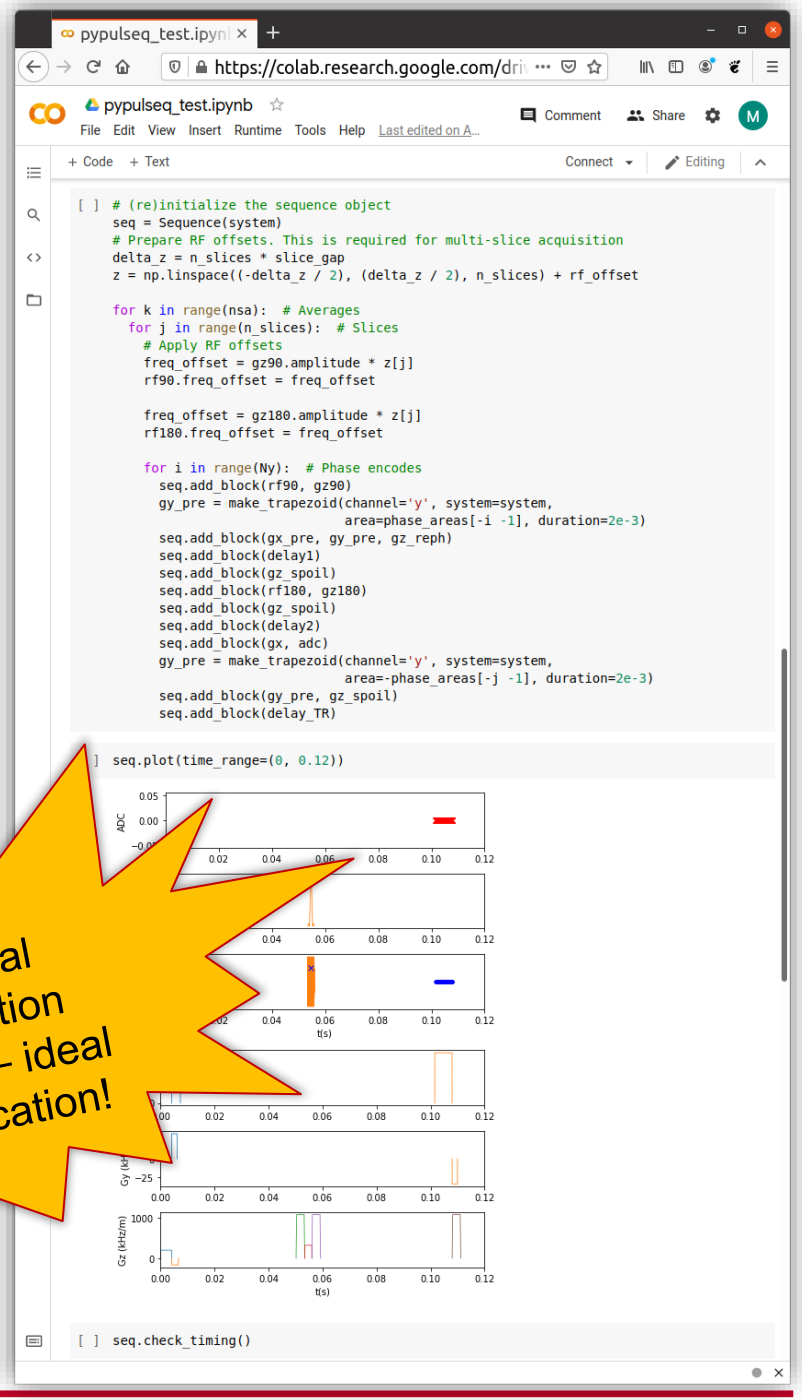
```
system = mr.opts('MaxGrad',30,'GradUnit','mT/m',...
    'MaxSlew',170,'SlewUnit','T/m/s');
seq=mr.Sequence(system);

fov = 220e-3; Nx=64; Ny=64; TE = 10e-3; TR = 20e-3;

[rf, gz] = mr.makeSincPulse(15*pi/180,system,'Duration',4e-3,...
    'SliceThickness',5e-3,'apodization',0.5,'timeBwProduct',4);

gx = mr.makeTrapezoid('x',system,'FlatArea',Nx/fov,'FlatTime',6.4e-3);
adc = mr.makeAdc(Nx,'Duration',gx.flatTime,'Delay',gx.riseTime);
gxPre = mr.makeTrapezoid('x',system,'Area',-gx.area/2,'Duration',2e-3);
gzReph = mr.makeTrapezoid('z',system,'Area',-gz.area/2,'Duration',2e-3);
phaseAreas = ((0:Ny-1)-Ny/2)*1/fov;

delayTE = TE - mr.calcDuration(gxPre) - mr.calcDuration(rf)/2 ...
    - mr.calcDuration(gx)/2;
delayTR = TR - mr.calcDuration(gxPre) - mr.calcDuration(rf) ...
    - mr.calcDuration(gx) - delayTE;
delay1 = mr.makeDelay(delayTE);
delay2 = mr.makeDelay(delayTR);

for i=1:Ny
    seq.addBlock(rf,gz);
    gyPre = mr.makeTrapezoid('y',system,'Area',phaseAreas(i),...
                             'Duration',2e-3);
    seq.addBlock(gxPre,gyPre,gzReph);
    seq.addBlock(delay1);
    seq.addBlock(gx,adc);
    seq.addBlock(delay2)
end

seq.write('gre.seq')
```

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

- 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
- Duration of each block is defined by the duration of the longest event
- *Copy '*.seq' to the scanner and run it!*

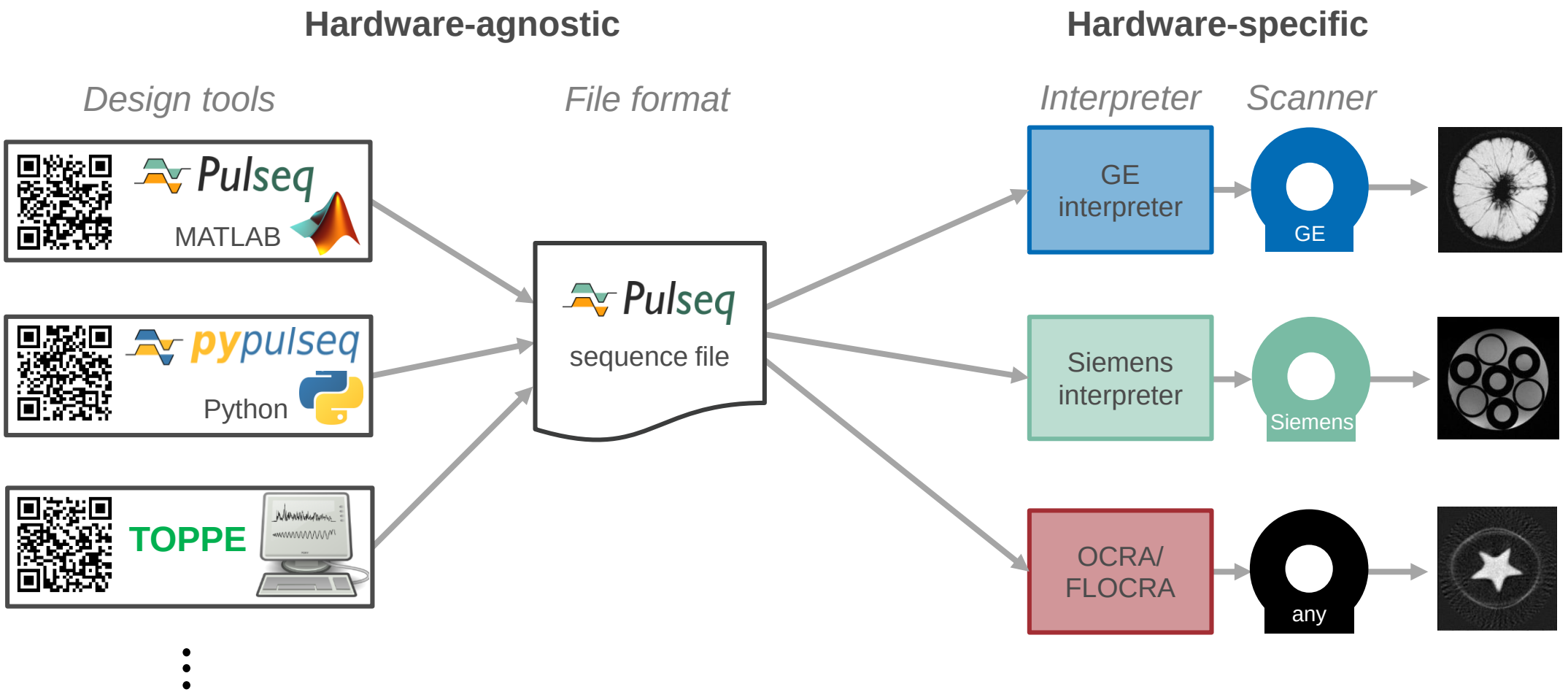
PyPulseq workflow

- ***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
- 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!*



No local installation needed – ideal for education!

Pulseq framework overview



Exercises

Go to: www.github.com/pulseq

⇒ Repositories

⇒ Ankara-UMRAM-Hands-on-Course--March-2024

⇒ Tutorials

⇒ Ankara_SplitGREsequence4demo.ipynb

⇒ Open in Colab

Acknowledgements:

Berkin Bilgic
Borjan Gagoski
Douglas Noll
Frank Zijlstra
Imam Shaik
Jeff Fessler
Jon-Fredrik Nielsen
Juergen Hennig
Mojtaba Shafiekhani
Moritz Zaiss
Naveen Murthy
Niklas Wehkamp
Qiang Liu
Qingping Chen
Scott Peltier
Sebastian Littin
Will Grissom
Yogesh Rathi

THANK YOU FOR YOUR ATTENTION!

Pulseq MATLAB

pypulseq Python

Supported by NIH U24-NS120056 (Nielsen, Zaitsev)
and R01-EB032378 (Rathi, Bilgic)

NIH National Institutes of Health