UNIVERSITÄTS KLINIKUM FREIBURG

Basic Pulseq Concepts

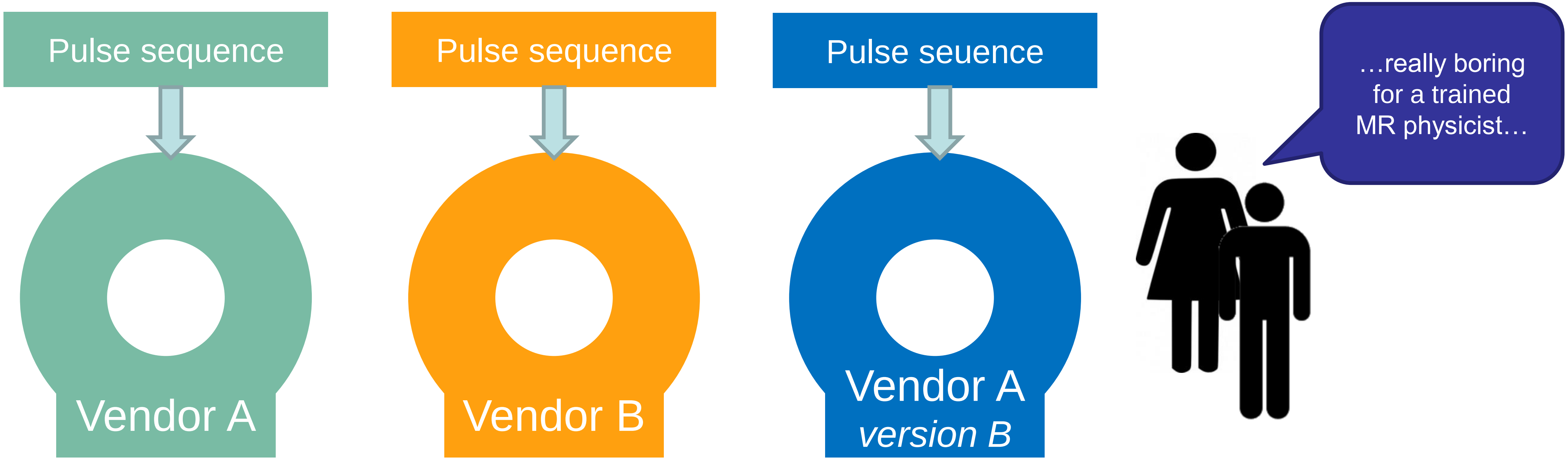
Maxim Zaitsev

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

November 23 2022

Did you participate in multi-center Studies?

- Huge effort to set up and maintain
  - Are sequences identical? Do they remain identical over years after updates?
  - Quality assurance is a major task in a heterogeneous hardware environment



Pulse sequence

Vendor A

Pulse sequence

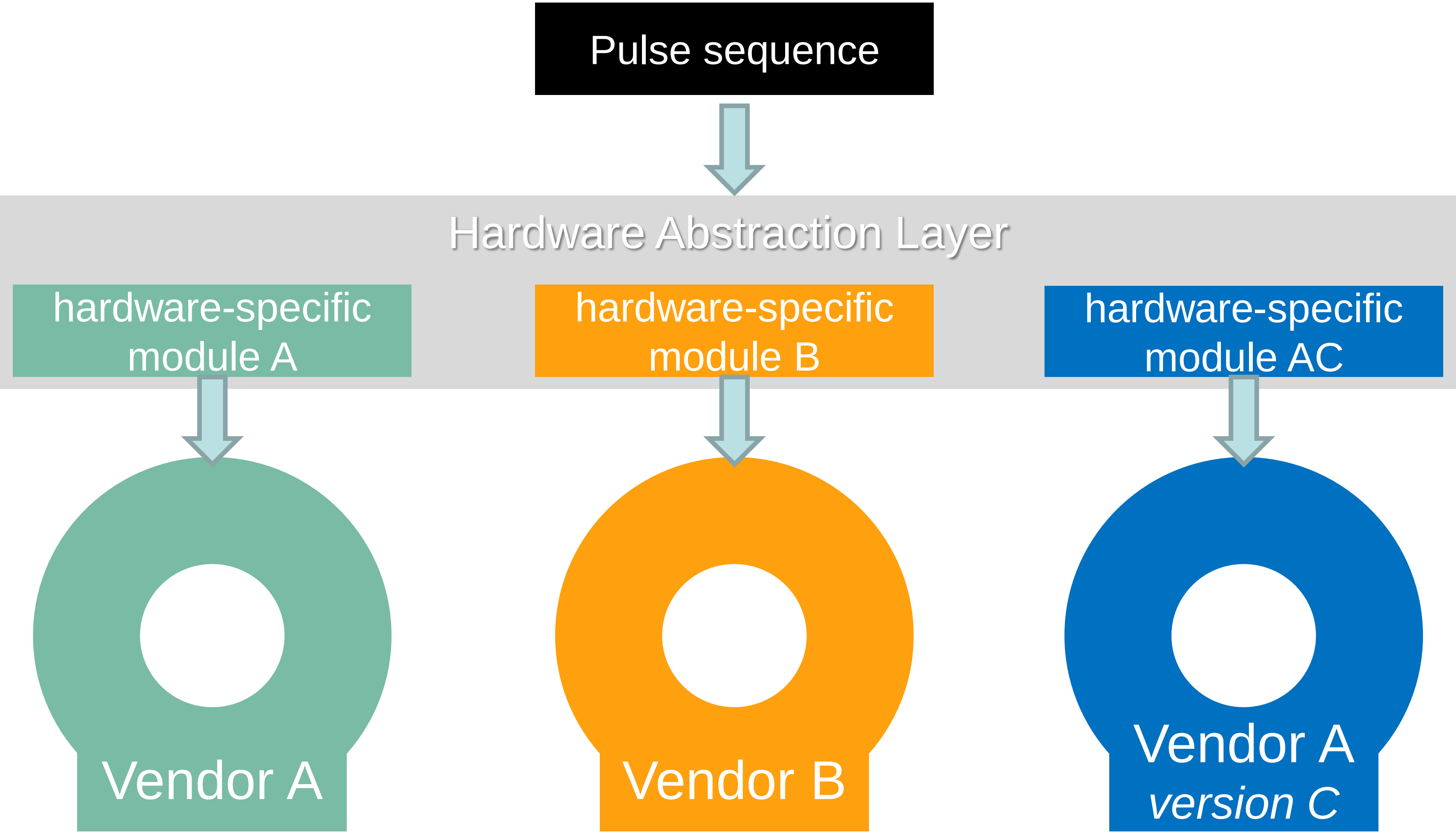
Vendor B

Pulse seuence

Vendor A *version B*

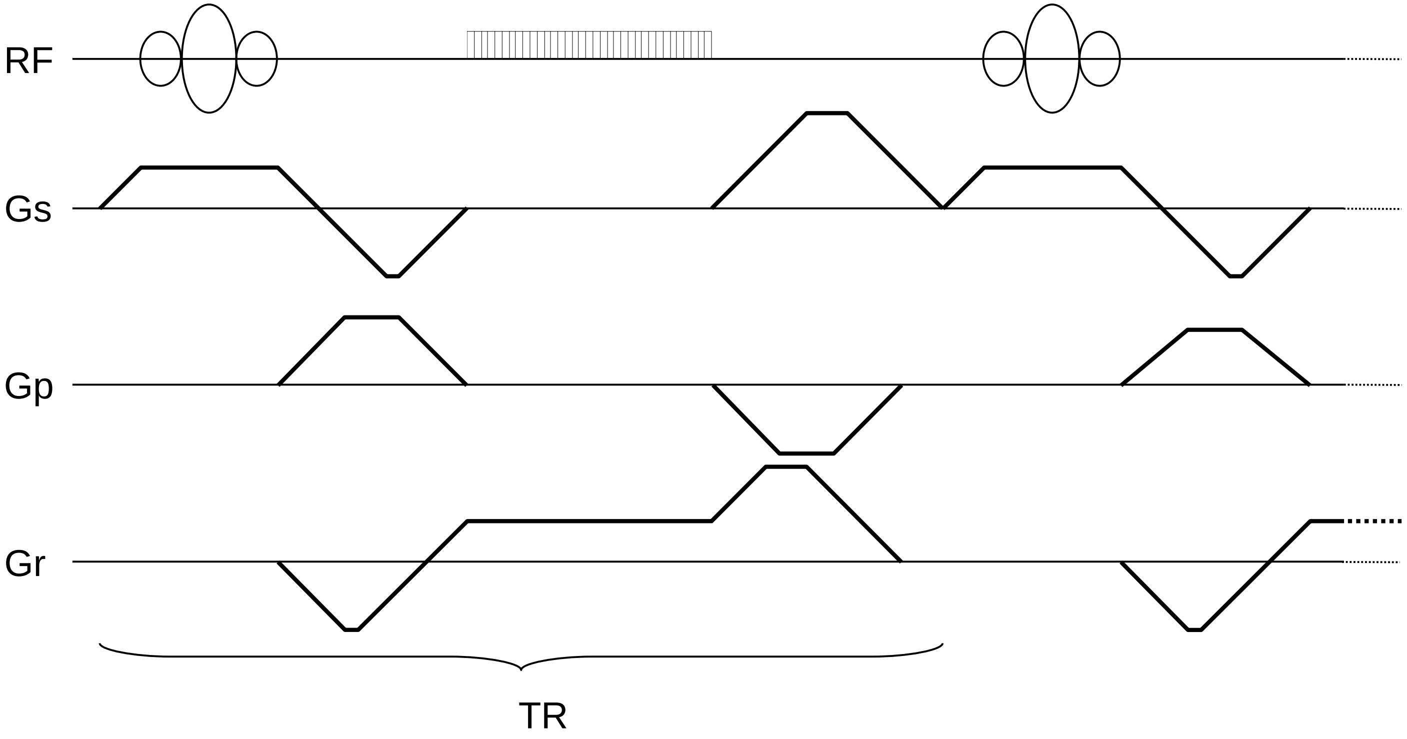
…really boring for a trained MR physicist…

Solution from computer science


Pulse sequence
Hardware Abstraction Layer
hardware-specific module A
hardware-specific module B
hardware-specific module AC
Vendor A
Vendor B
Vendor A *version C*

Lean hardware interface is possible…

- Because MR pulse sequences are simple and repetitive!
(in comparison to e.g. natural sounds)

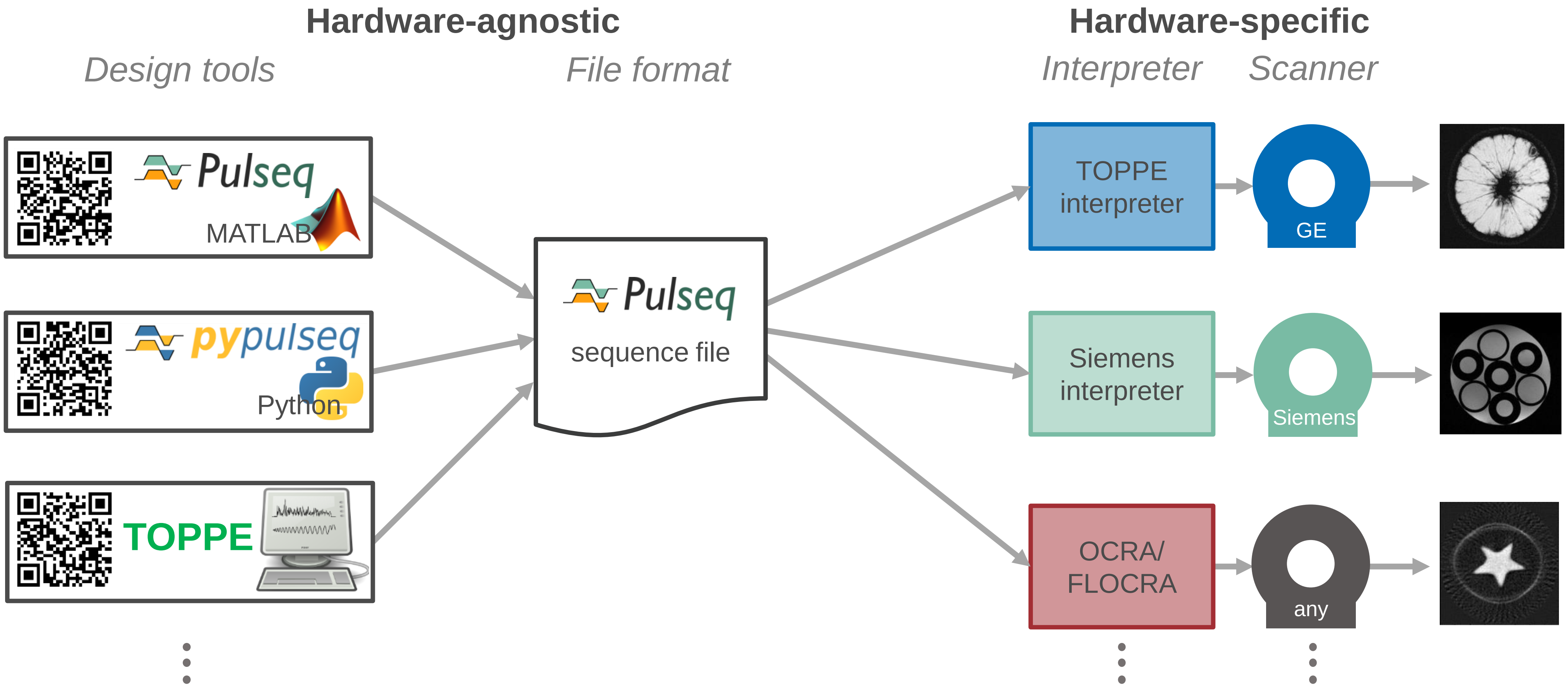


  - Low-level interface does the trick

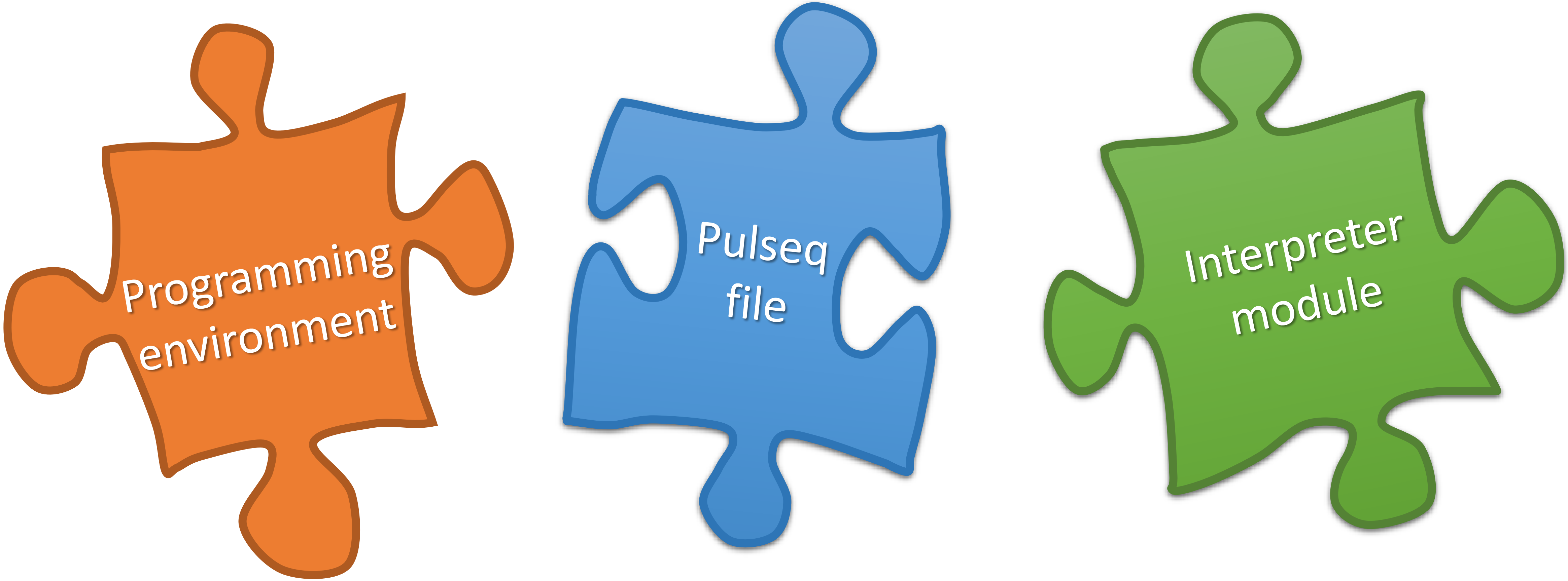
Pulseq Goals

- Remove the initial threshold in sequence programming
  - Turn simple things really simple
- Make researcher-oriented features easily accessible
  - Arbitrary gradients, arbitrary RF, flexible reordering, X-nuclei, …
- Prevent typical sources of (human) errors
  - Avoid timing errors with overlapping gradients
  - Make data flag and counter setting optional/unnecessary
- Minimize effort for implementation and support on hardware
  - Lean sequence-to-hardware interface

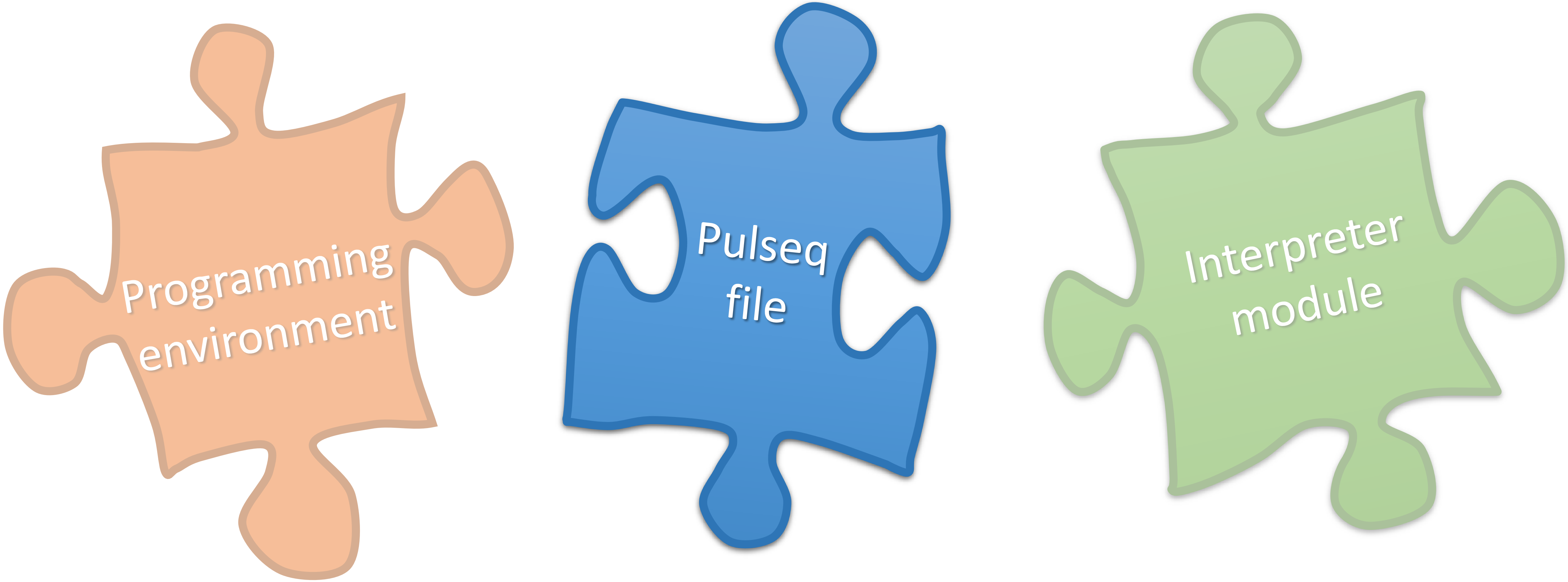
Pulseq framework overview



Pulseq : pieces of the puzzle

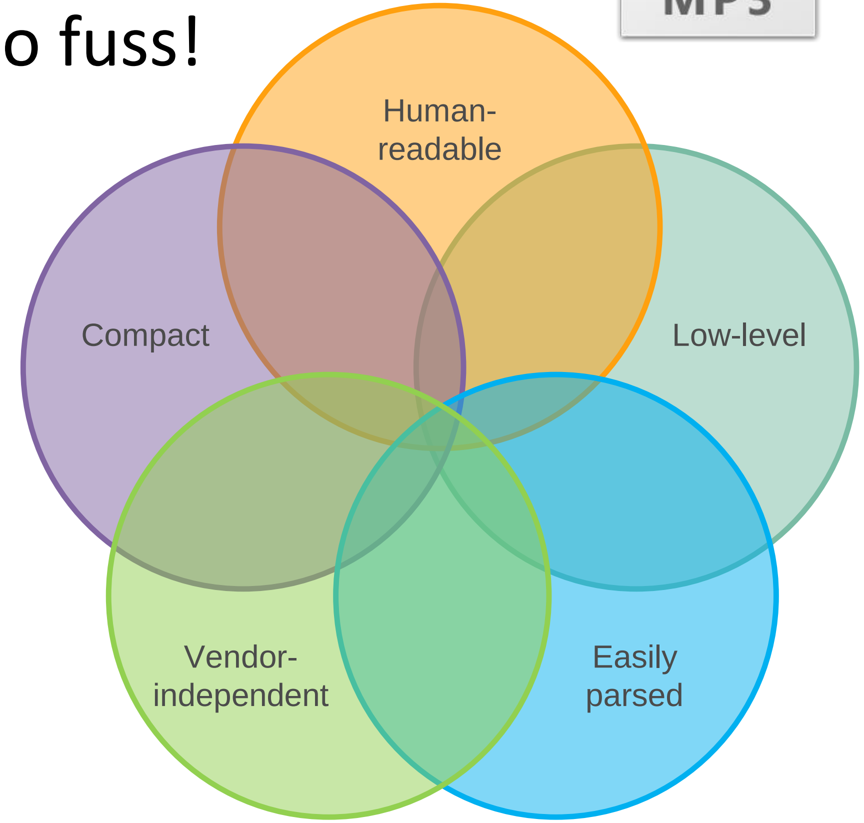


Pulseq : pieces of the puzzle



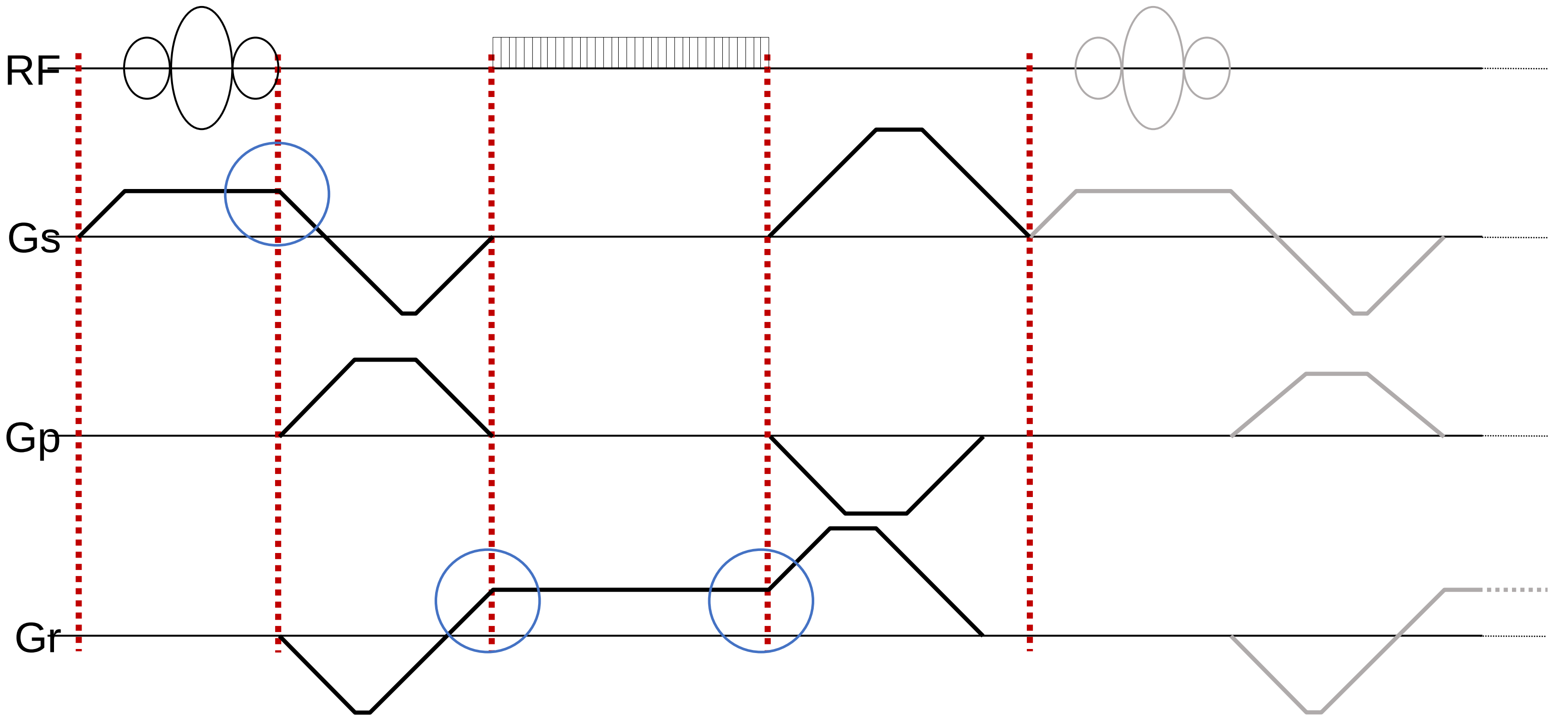
Pulseq file

- 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



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 …

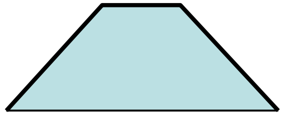
Gradients do not have to start or end at 0 at the block boundaries

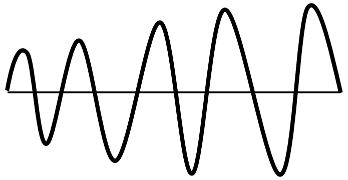
Pulseq block concept in detail

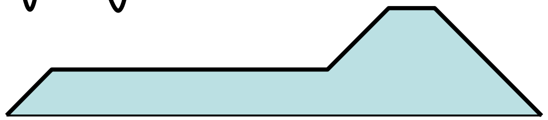
- Each block may contain following events:
  - One optional gradient pulse per axis
  - One optional RF pulse
  - One optional ADC event
- Individual events may define own start delays
- All events in the block overlap in time
- Duration of the block need to exceed that of the longest event
  - Matlab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

Advanced topics

- Pulseq native unit for gradient and RF amplitude is Hz
- Three types of gradient events in Pulseq
  - Trapezoid pulses
    ramp-up, flat top, ramp-down
  - Free shape defined on a regular
    raster (typically 10 us)
  - Extended Trapezoid: time-amplitude pairs
    with linear ramps in between

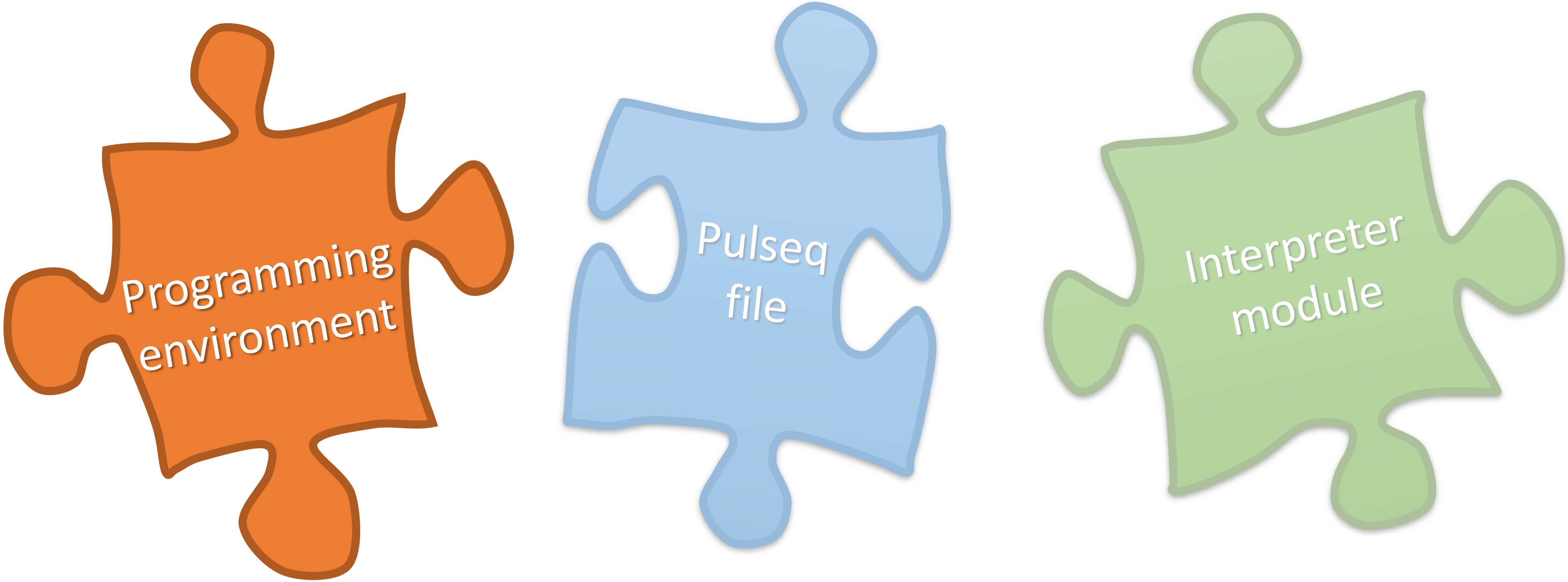






- RF and ADC events cannot touch block boundaries (system-specific limits)
- RF pulses may define custom dwell time (default 1 us) to overcome duration limit due to the 8192 points shape limit on Siemens
- Semi-automatic ADC splitting to segments
  (overcome 8192 points receive limit)

Pulseq : pieces of the puzzle



Programming environment

Pulseq file

Interpreter module

High-level programming environments

- Matlab *Pulseq* toolbox
- Python *pypulseq* toolbox

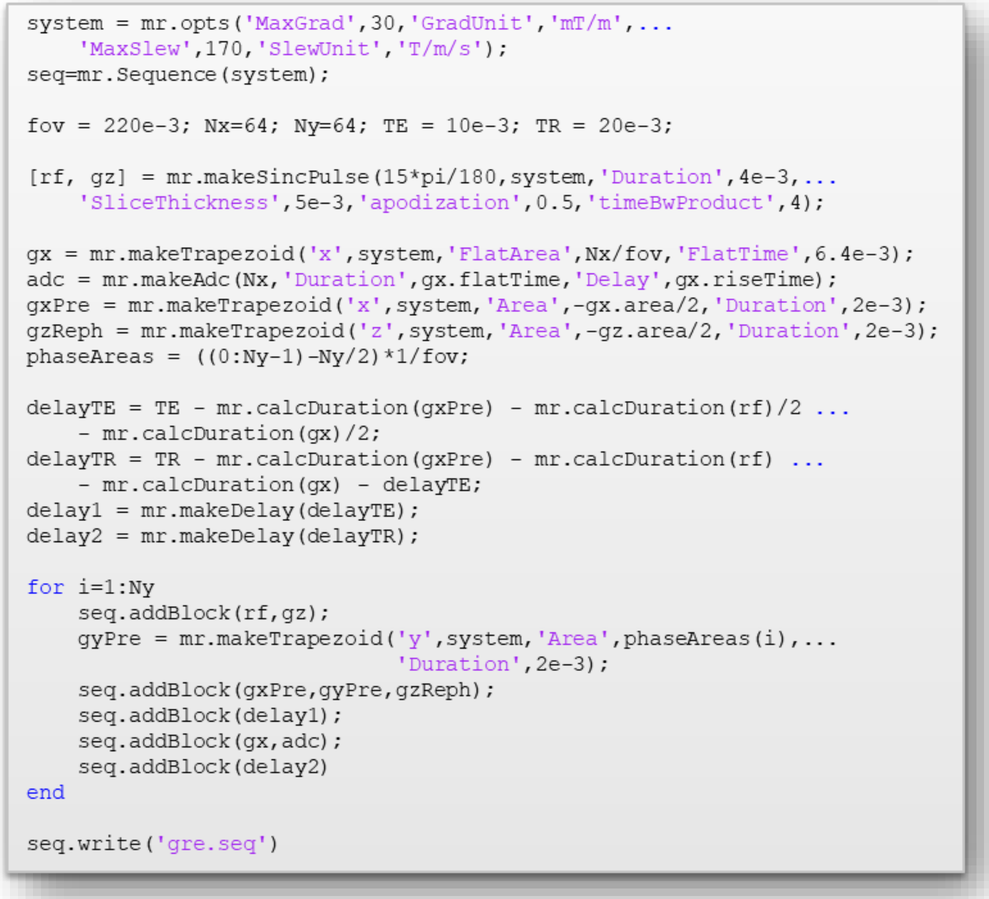
Pulseq MATLAB

pypulseq Python

- Further options
  - TOPPE is primarily targeted at GE but can import and export *pulseq* files (Jon-Fredrik Nielsen will talk about it today)
  - GammaStar can export *pulseq* files
  - JEMRIS Bloch simulator can export *pulseq* files
  - CoreMRI Bloch simulator can export *pulseq* files
  - …

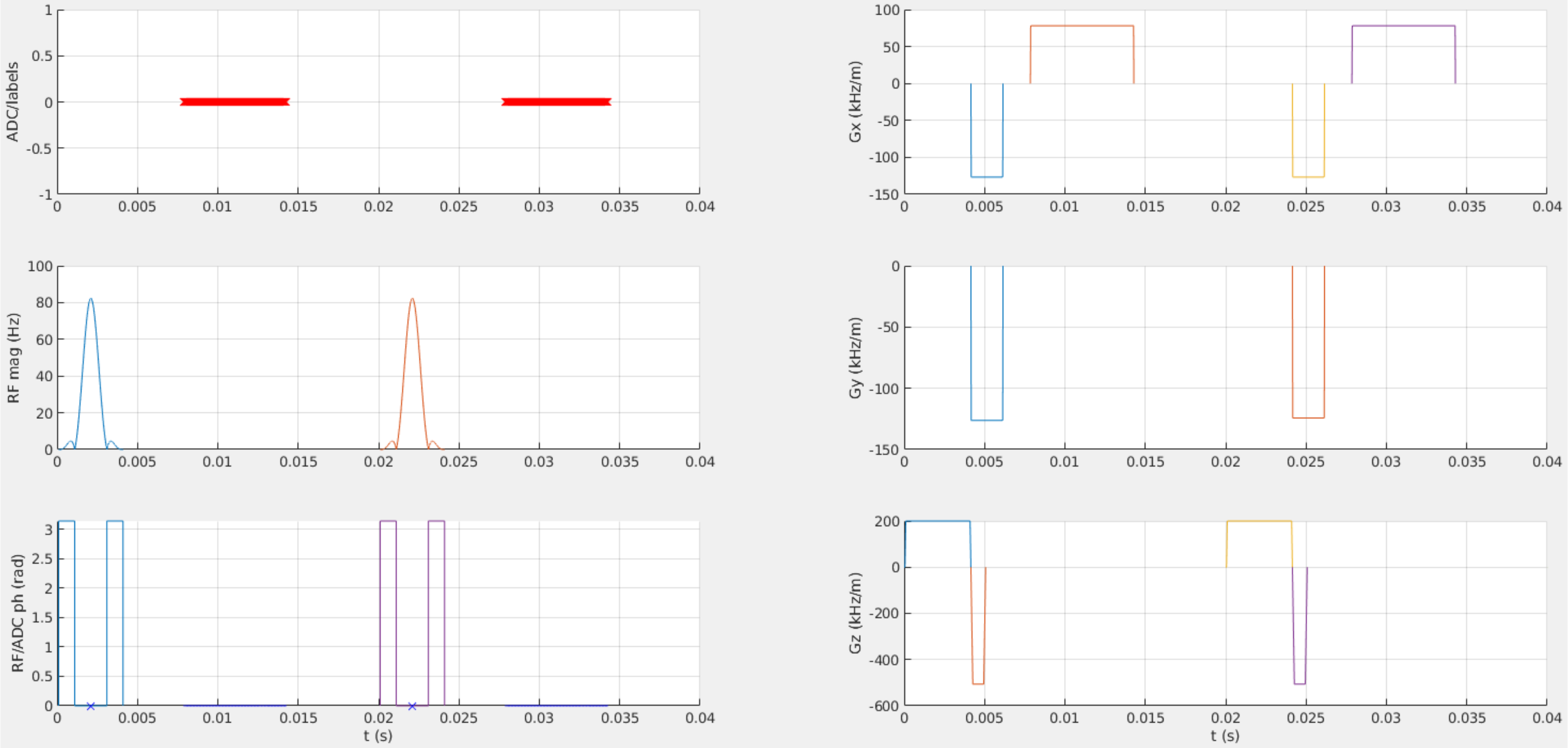
Matlab *Pulseq* workflow

- Define the system properties
- Define high-level parameters (convenience)
- Define pulses 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 ‘gre.seq’ to the scanner and run it!
- *Screenshot shows an entire runnable gradient echo sequence code (similar to Siemens’ example miniFlash)*



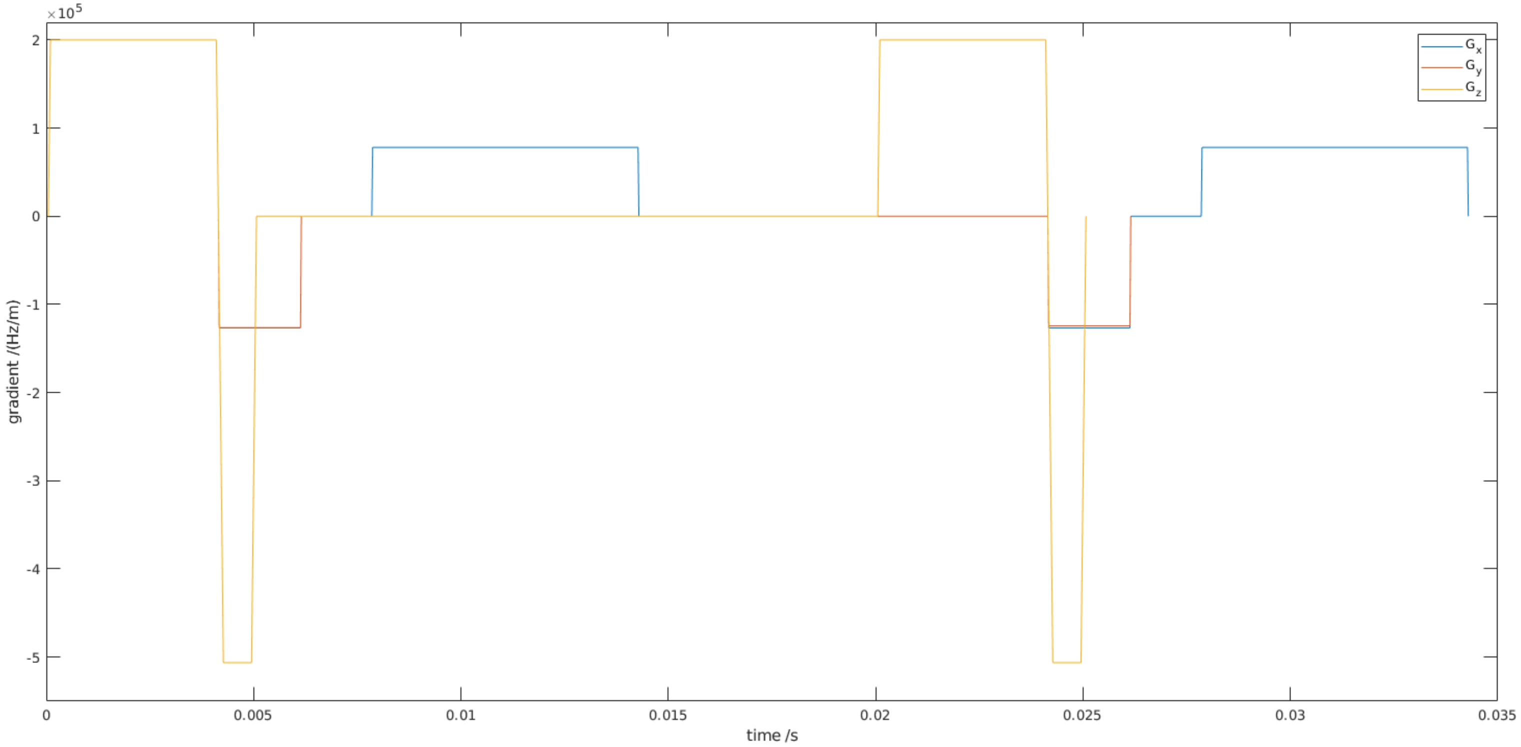
```
system = mr.opts('MaxGrad',30,'GradUnit','mT/m',...
    'MaxSlew',170,'SlewUnit','T/m/s');
seq=mr.Sequence(system);

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')
```

Basic sequence display options



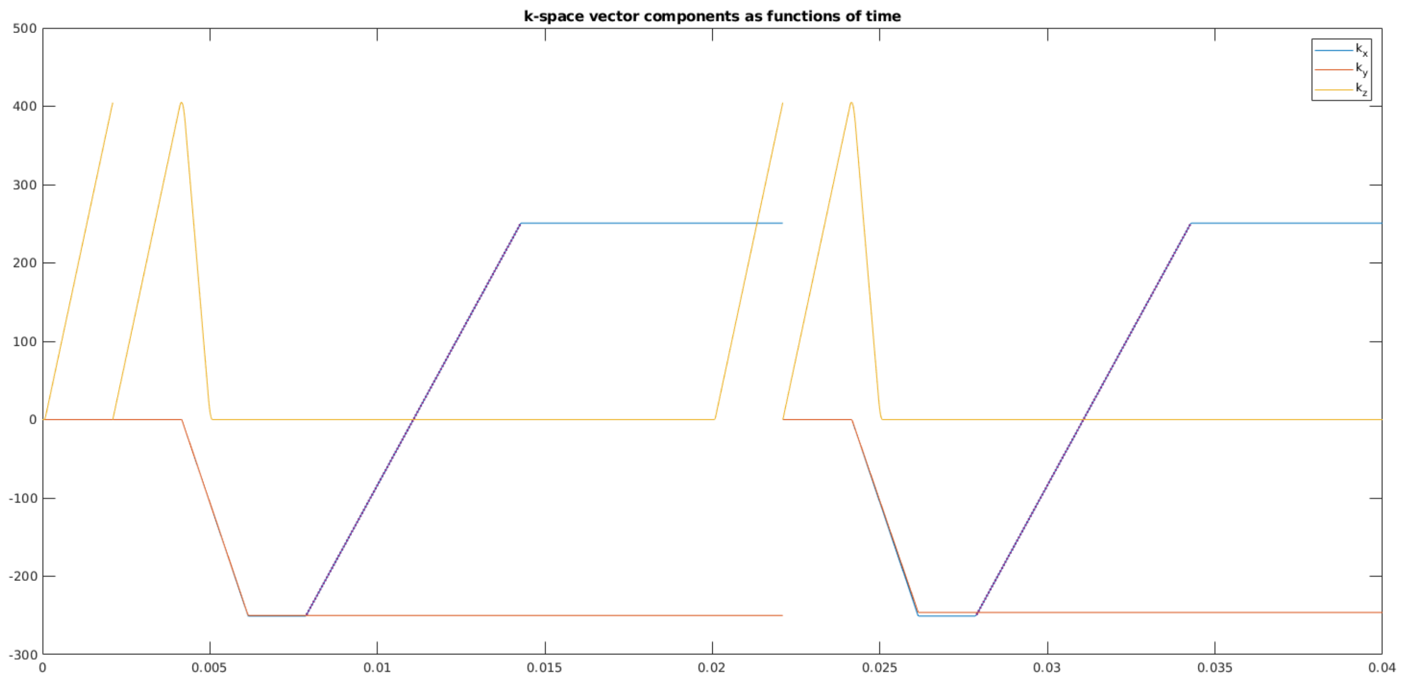
- Pulseq 6-panel plot
  - ADC, RF magnitude, RF phase, Gx, Gy, Gz
  - Each event in own color

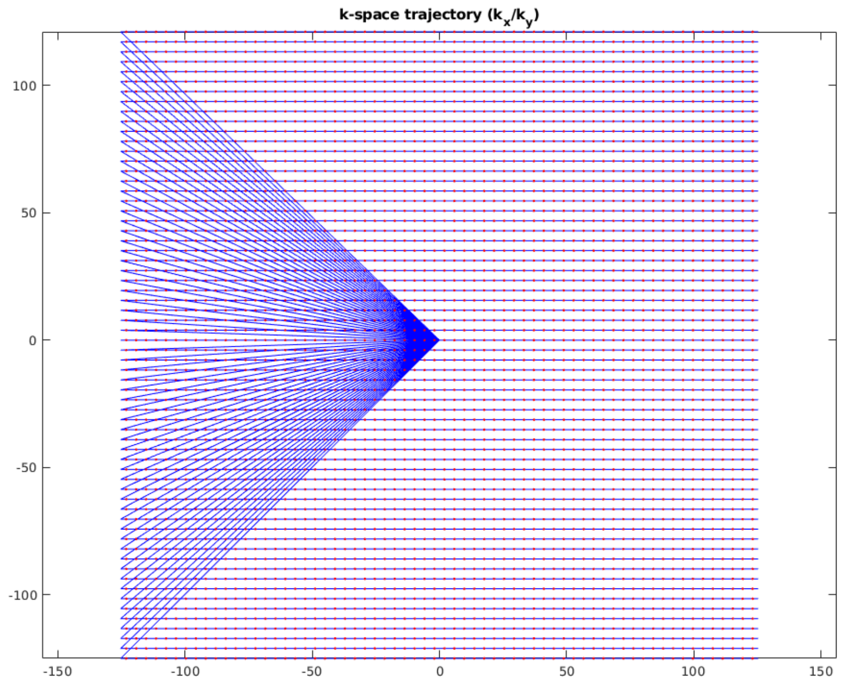
Basic sequence display options cntd.



- Plot entire waveform for all axes
  - Native gradient unit: Hz/m

Basic sequence display options: k-space





- Plot k-space time evolution or 2D (or even 3D) trajectories
- Native k-space unit in Pulseq: m^{-1}

How to design a sequence in Pulseq

conceptual design steps

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

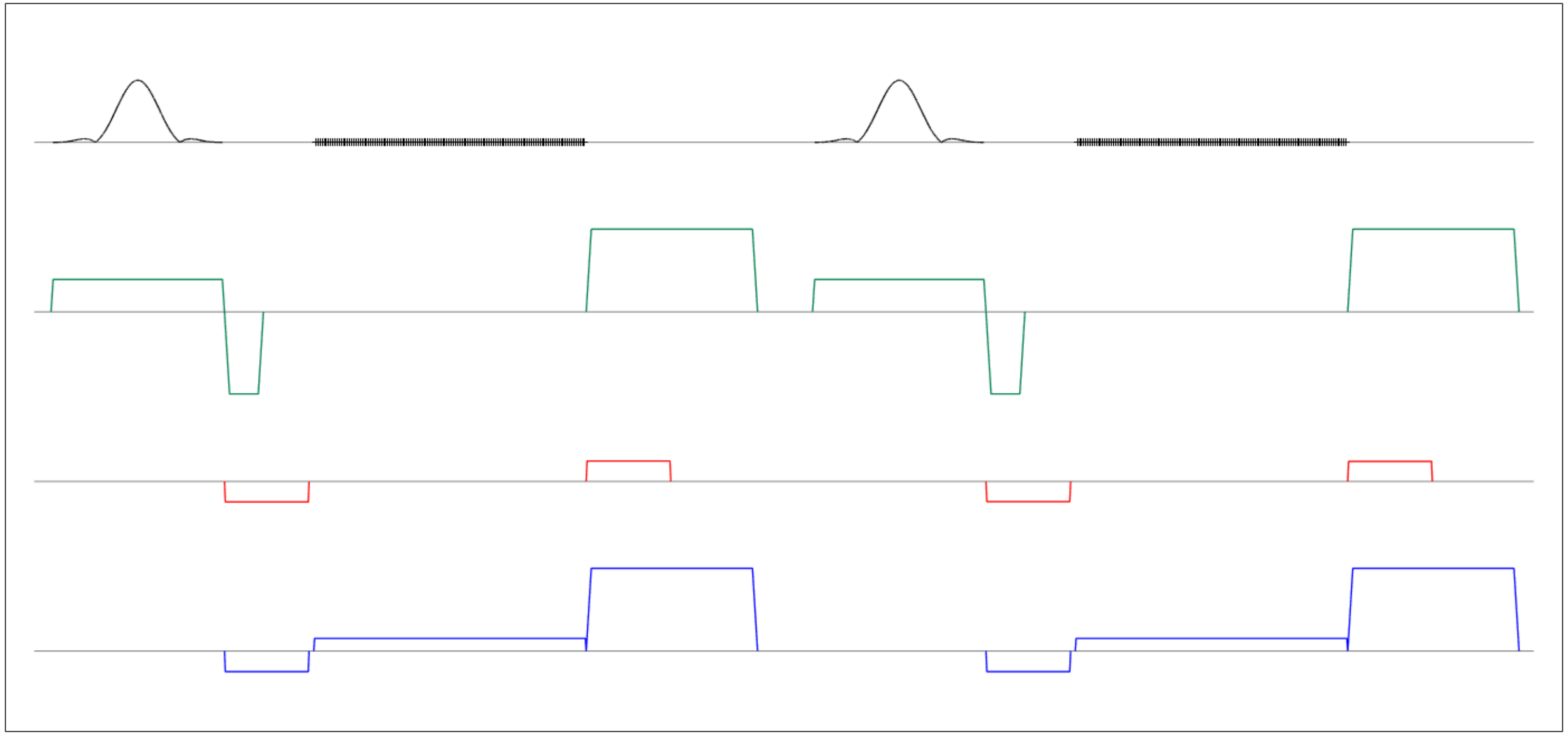
practical implementation steps

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

validation steps

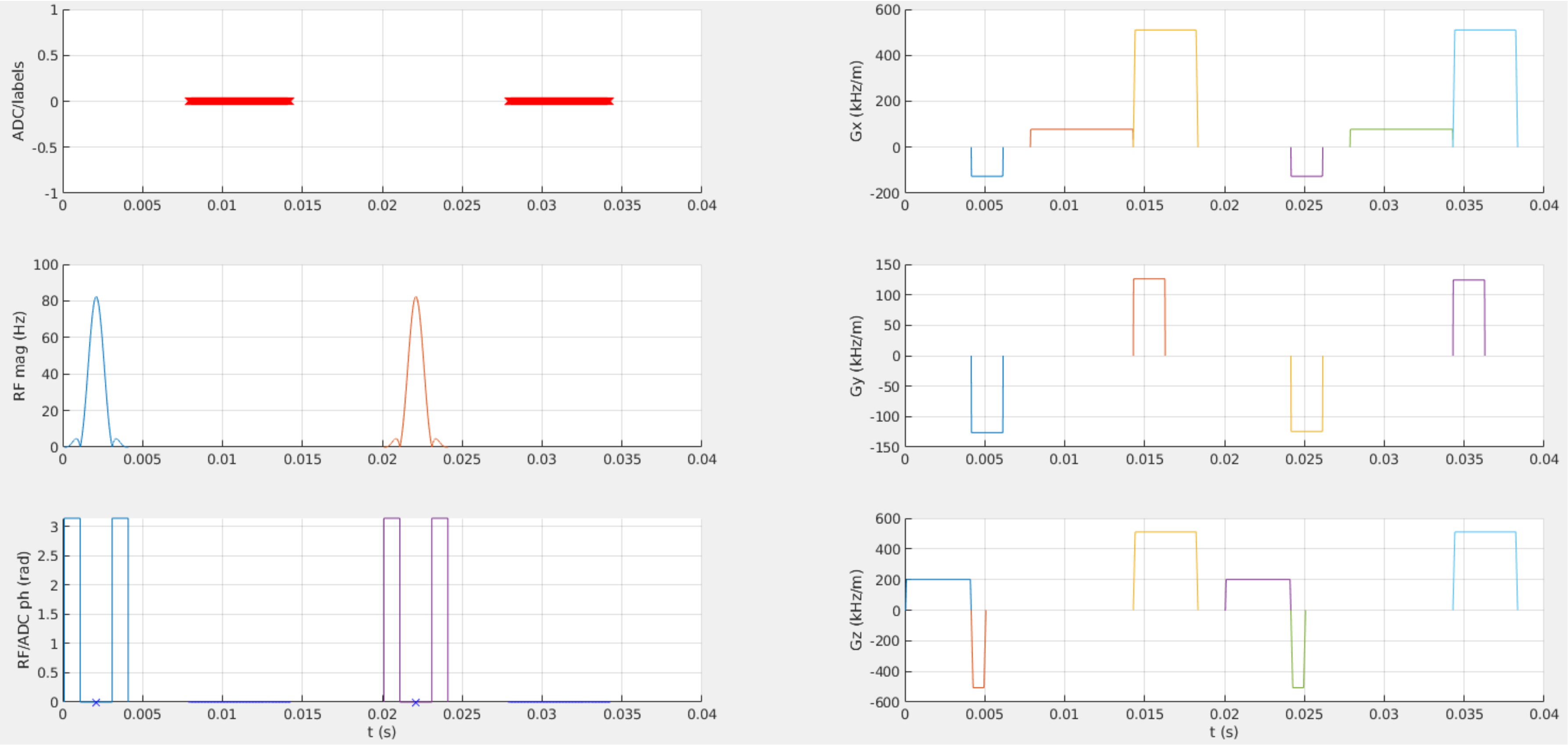
- Step 5: check timing, verify k-space trajectory, hardware and PNS limits, etc

Example 1: simple gradient echo



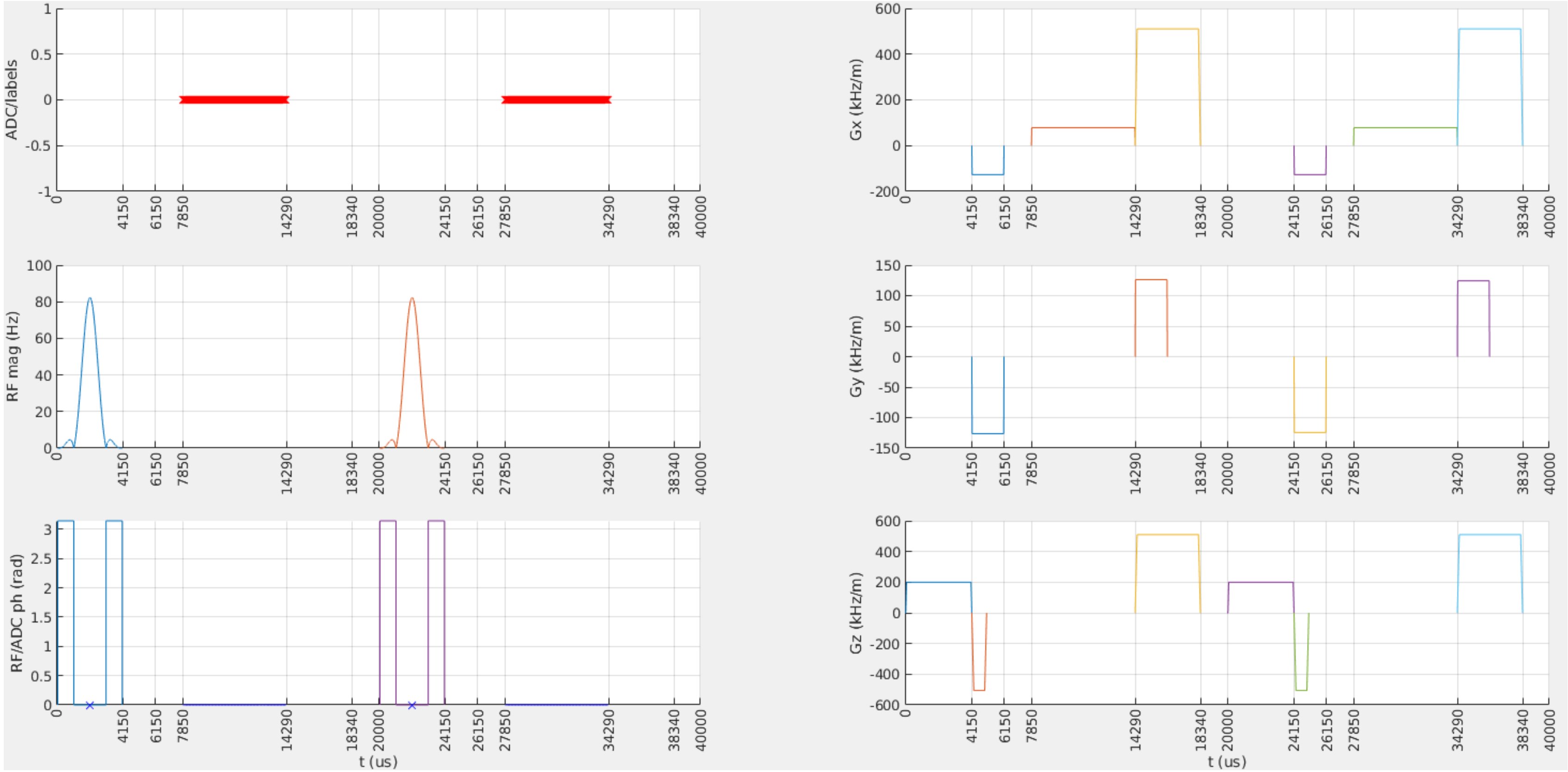
- No overlapping gradient ramps on different axes
- Events are clearly separated

Example 1 ctnd.: simple gradient echo



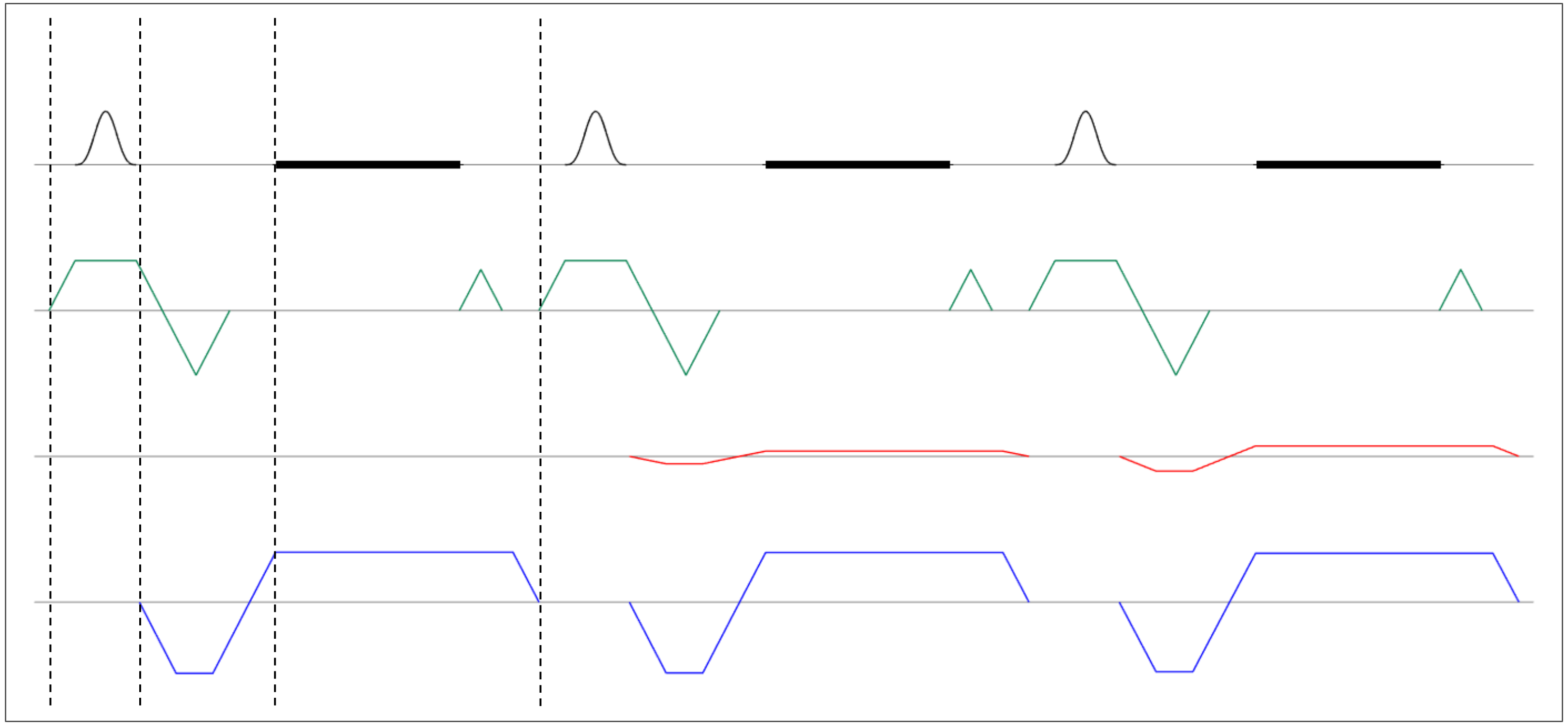
- Advantageous to separate PE & PR gradients into different blocks

Simple gradient echo – block structure



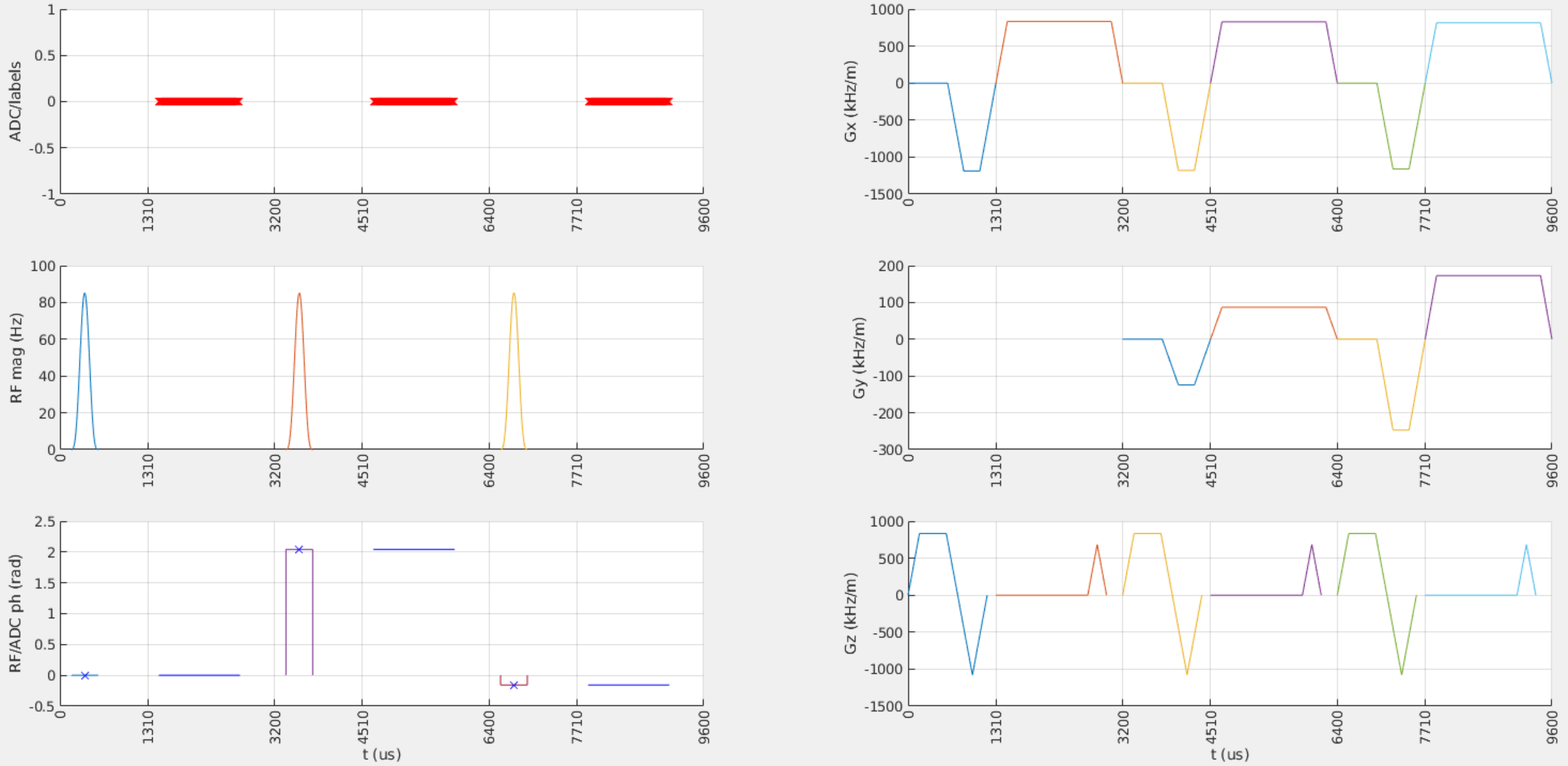
- It is possible to visualize the block structure
    - seq.plot('showBlocks',true,'timeDisp','us');

Example 2: fast radial gradient echo



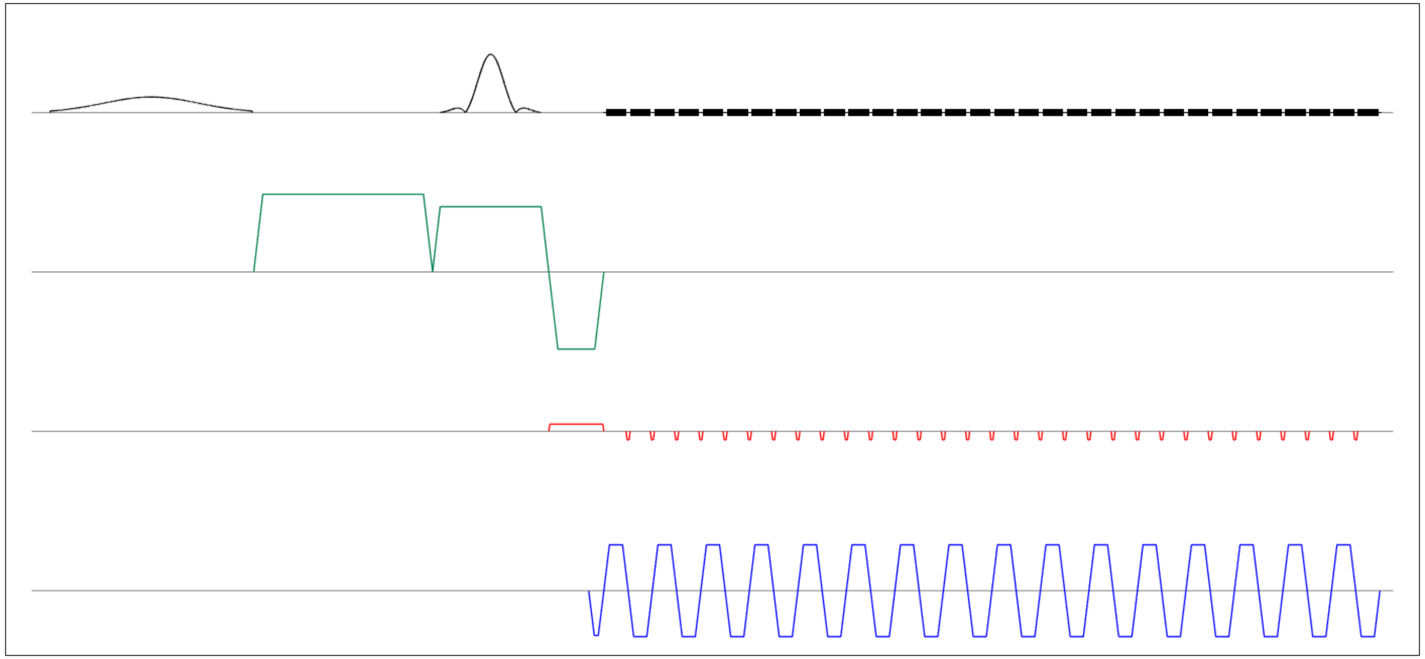
- Block separation is less obvious
  - Merging all into one block is possible, but this would make Z spoiler a part of a shaped gradient….

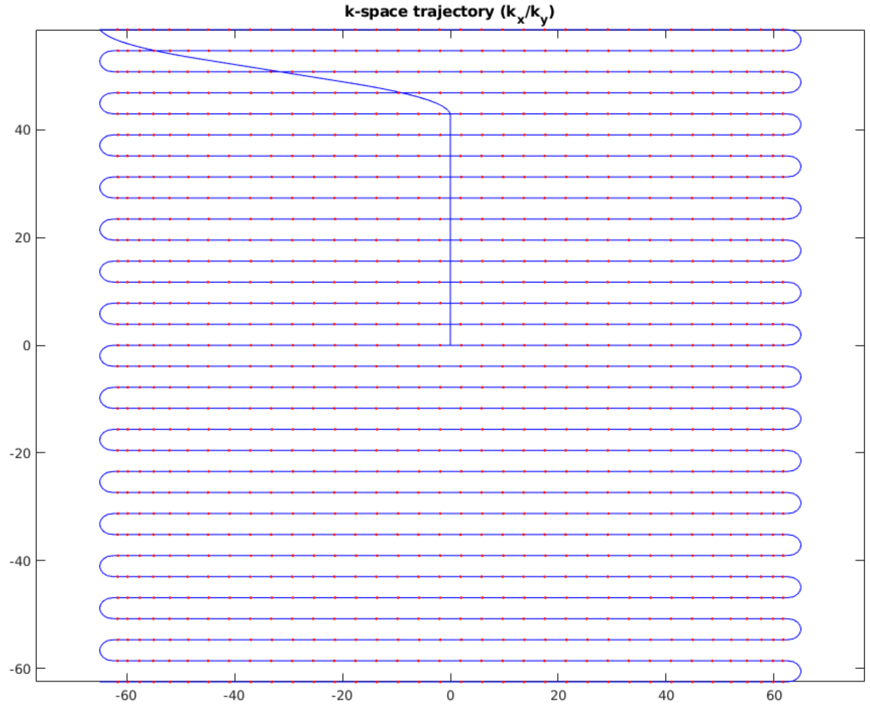
Fast radial gradient echo block structure



- This is one of many possible solutions
  - Many options work, but your Pulseq file size may vary

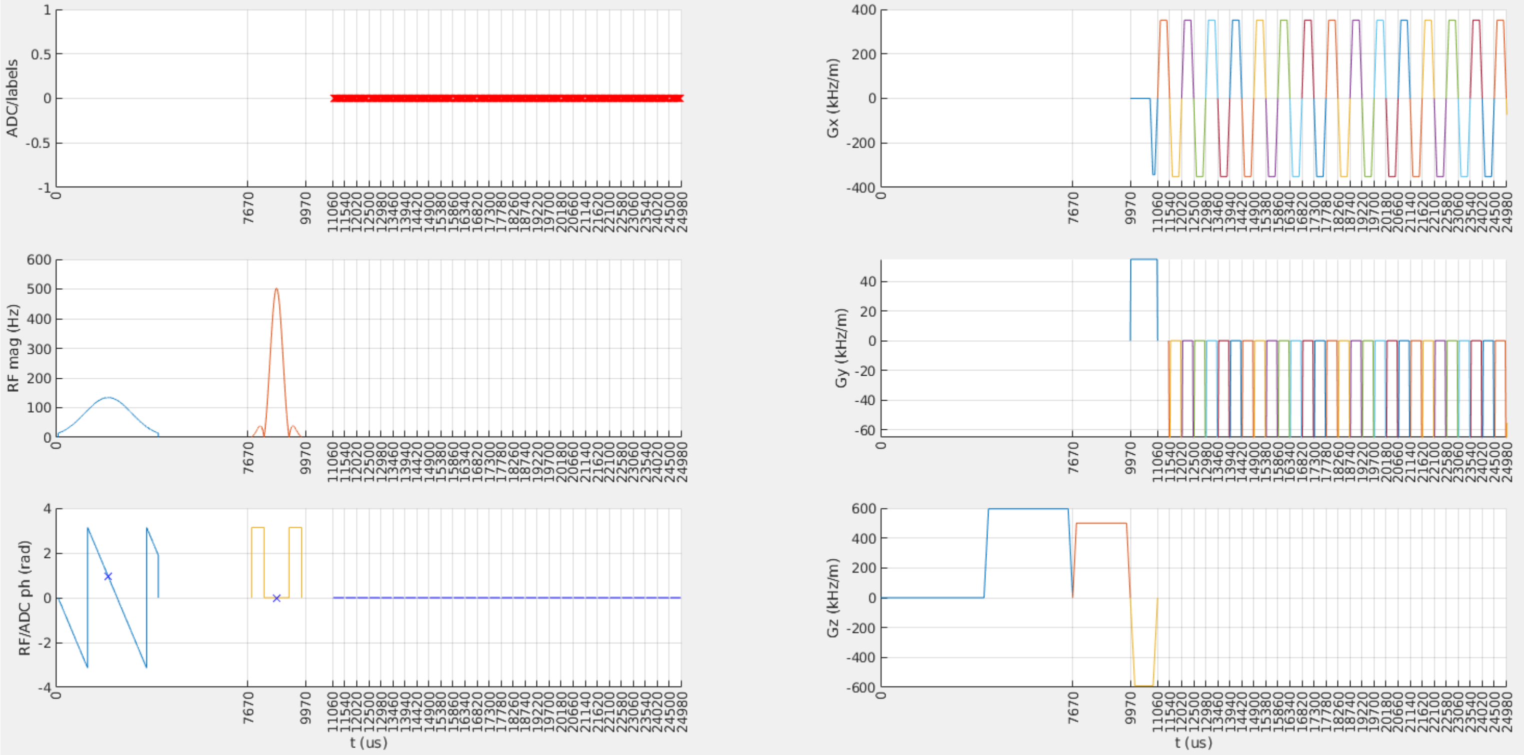
Example 3: echo planar imaging (EPI)

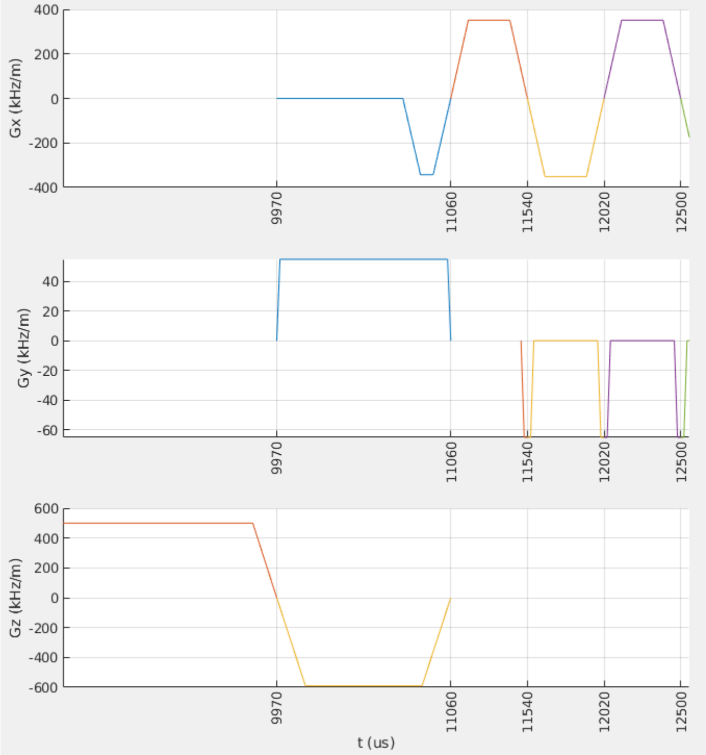




- Two RF pulses need to be in separate blocks
- Each ADC event needs to be in a separate block
  - Challenge with blips, readout ramp and optimal sampling window

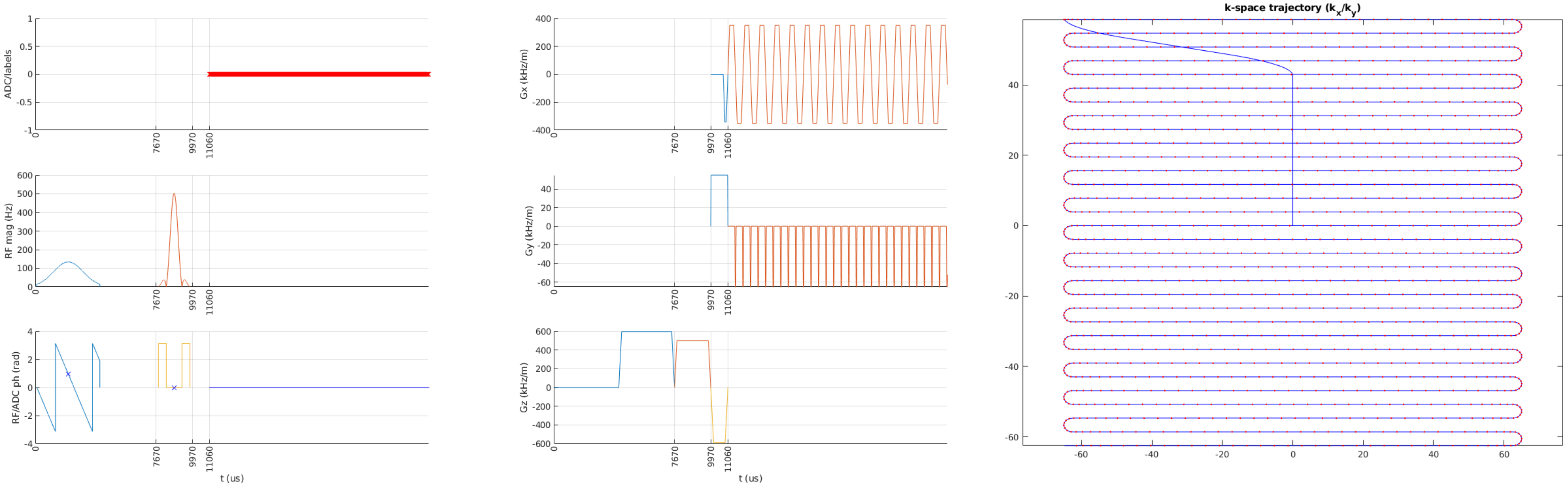
EPI block structure





- One possible solution:
  - Keep readout gradient as trapezoid
  - Convert blips to shapes and split them at the center

Alternative EPI block structure



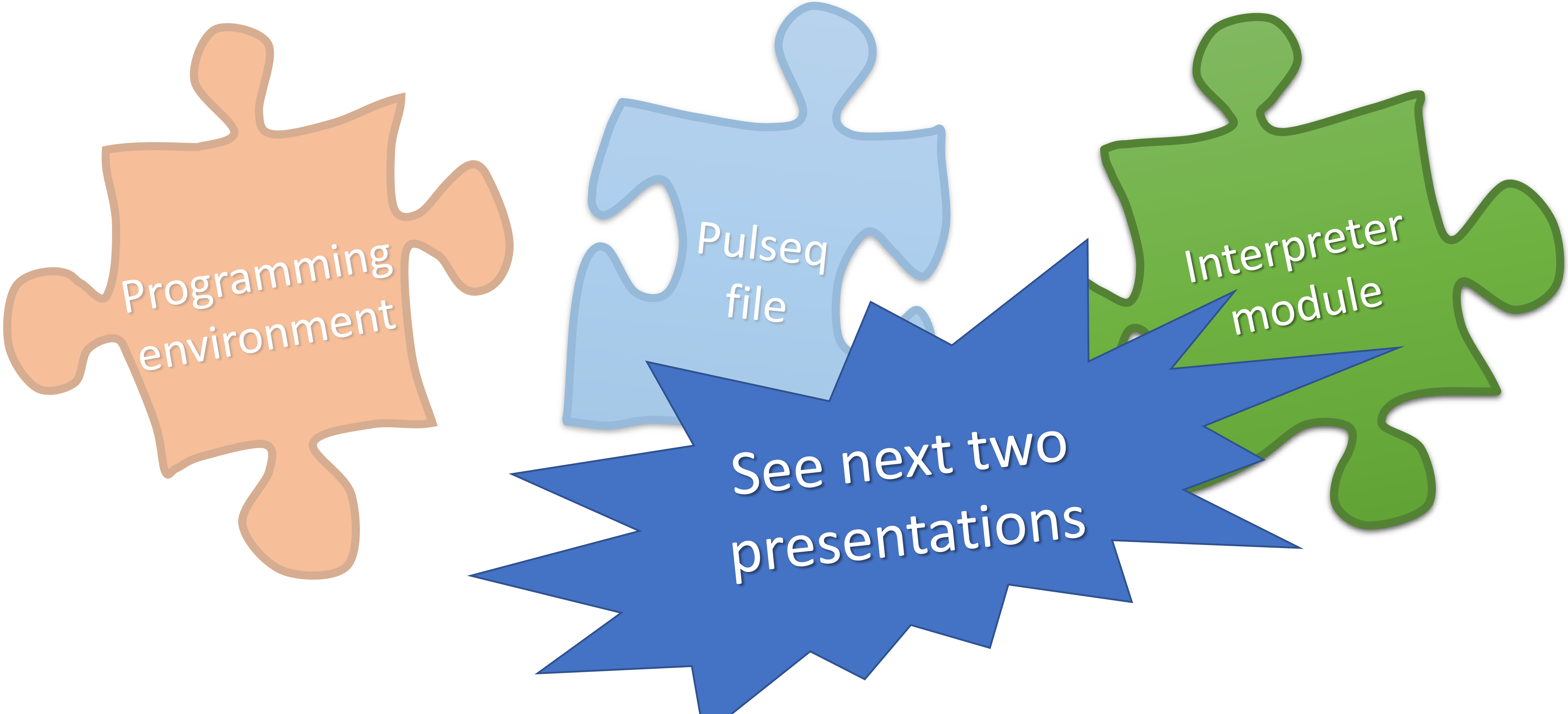
- Put the entire readout into one block
  - Gx and Gy are now both shaped gradients
  - ADC is sampling continuously (need to crop some points in the recon)

Pulseq blocks summary

- Block concept takes some time to get used to
- Blocks help to organize events and eliminate timing errors
- There is a lot of flexibility
  - Different strategies possible
- Some interpreters expose additional limitations
  - Explicit and implicit delays, number of ADCs per TR, etc…
  - You will hear more about this in the next talks

- **Blocks make it easier for the interpreter to play things out**

Pulseq : pieces of the puzzle



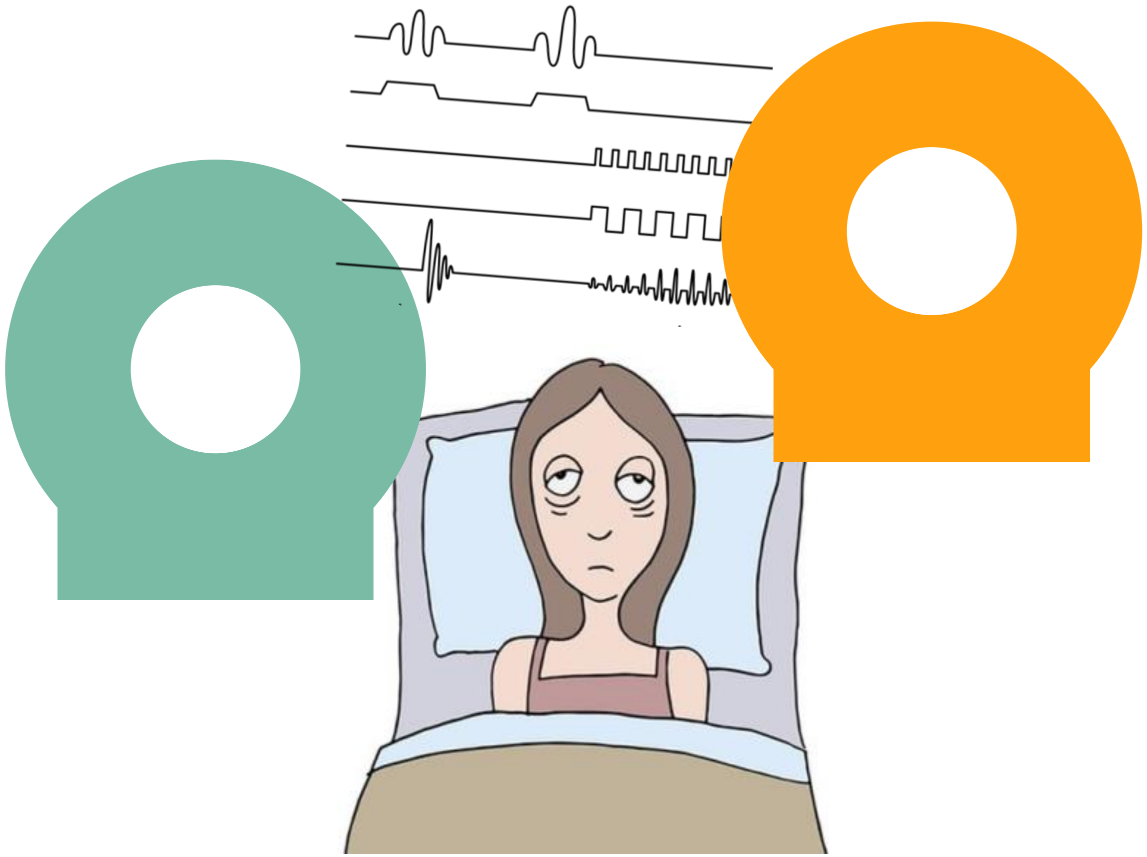
Programming environment

Pulseq file

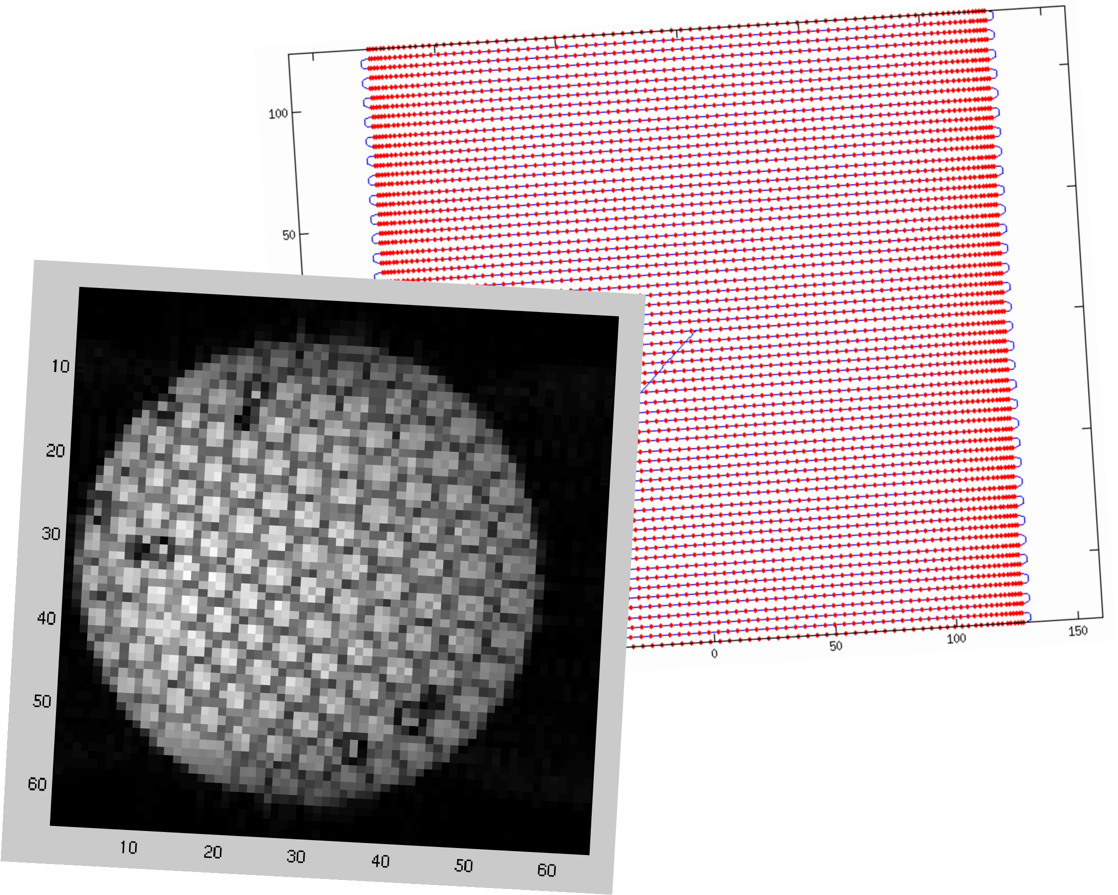
Interpreter module

See next two presentations

Pulseq – that’s the way we do it!



…dream of a sequence in the morning…



…check the images in the afternoon!

Cross-platform sequence development with *Pulseq*

… ease your access to MR physics

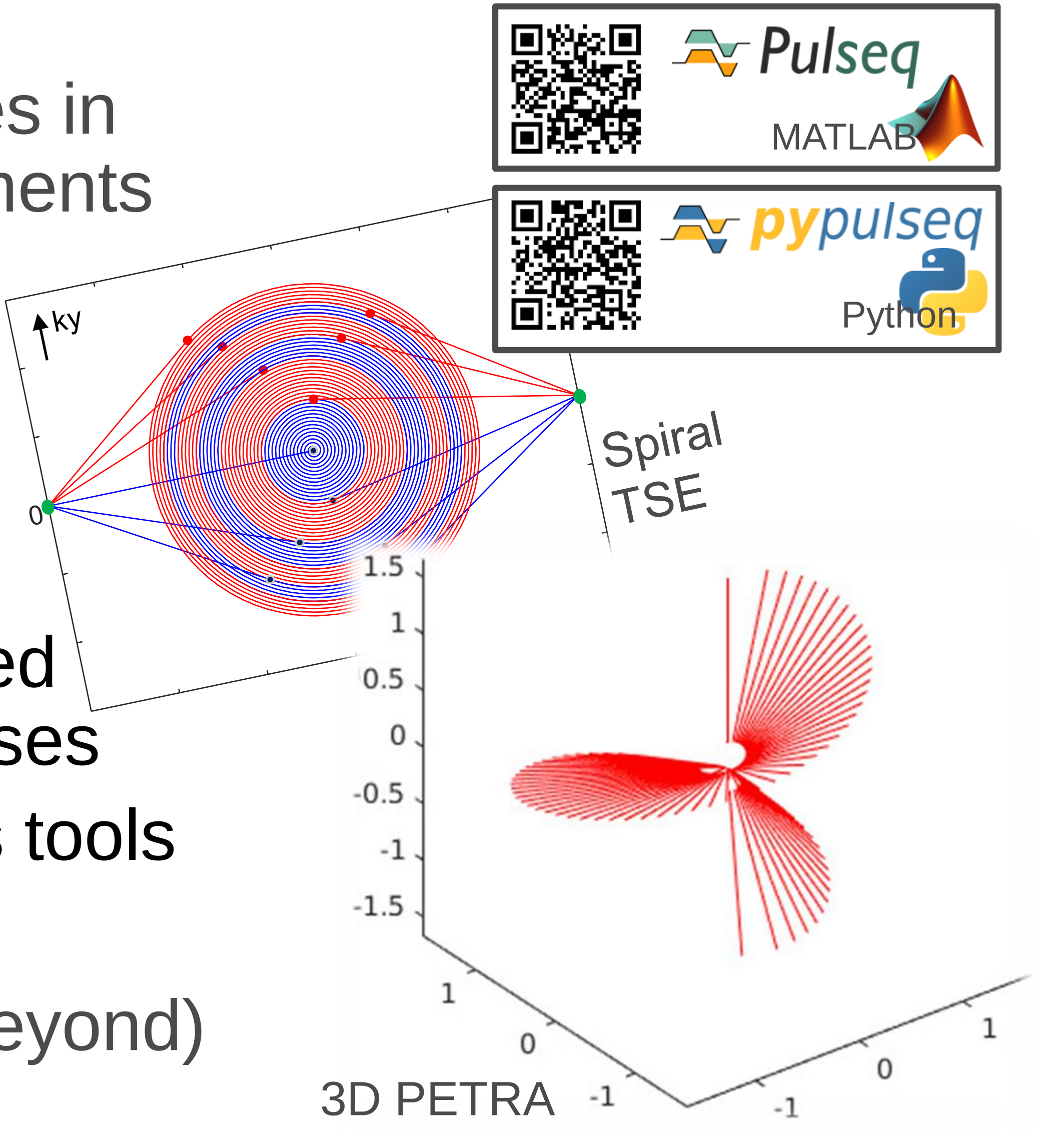
Develop, visualize and analyze sequences in modern interactive programming environments

- Matlab *Pulseq* toolbox
- Python *pypulseq* toolbox

MR physics-oriented workflow

- Write your sequeces 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 & beyond)



Pulseq use cases

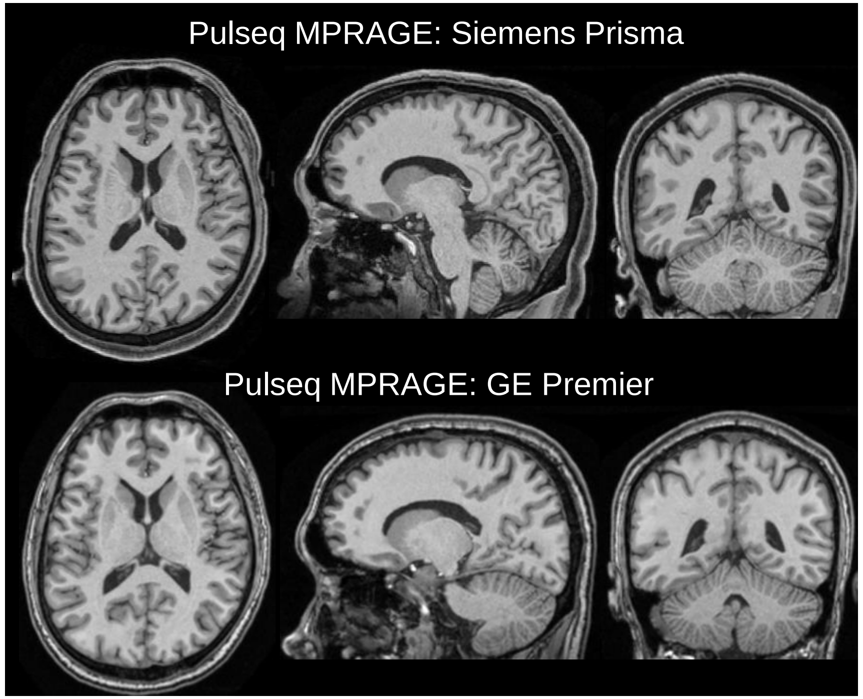
From prototyping to harmonization

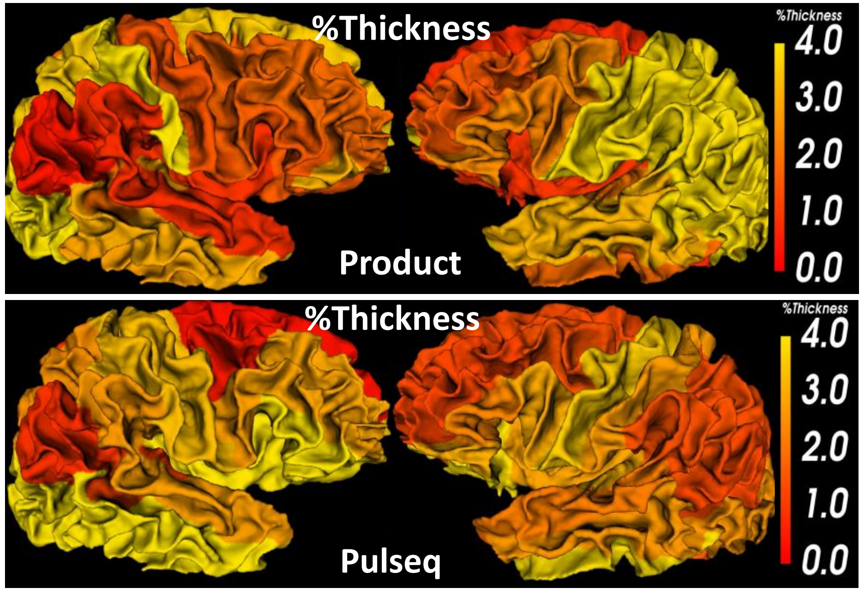
Growing Pulseq user base

Great prototyping tool

Increasing use for multicenter MR pulse sequence harmonization

- MPRAGE on Siemens & GE: Pulseq reduces cortical thickness variance
- ISMRM 2022, supported by NIH*





*NIH U24 S120056, PIs: Nielsen & Zaitsev; 1R01EB032378, Pis: Rathi, Bilgic, Gadoski, Grissom, Nielsen, Zaitsev

THANK YOU FOR YOUR ATTENTION!

…ready to take questions…

Raw data and code for today’s Demo1 on Dropbox:
https://www.dropbox.com/scl/fo/owlucnr91dnogdwcvhnje/h?dl=0&rlkey=vbqd5jjthp5kgh1gi65kx1muo