Pulseq Principles

Maxim Zaitsev

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

Feb 8, 2024

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

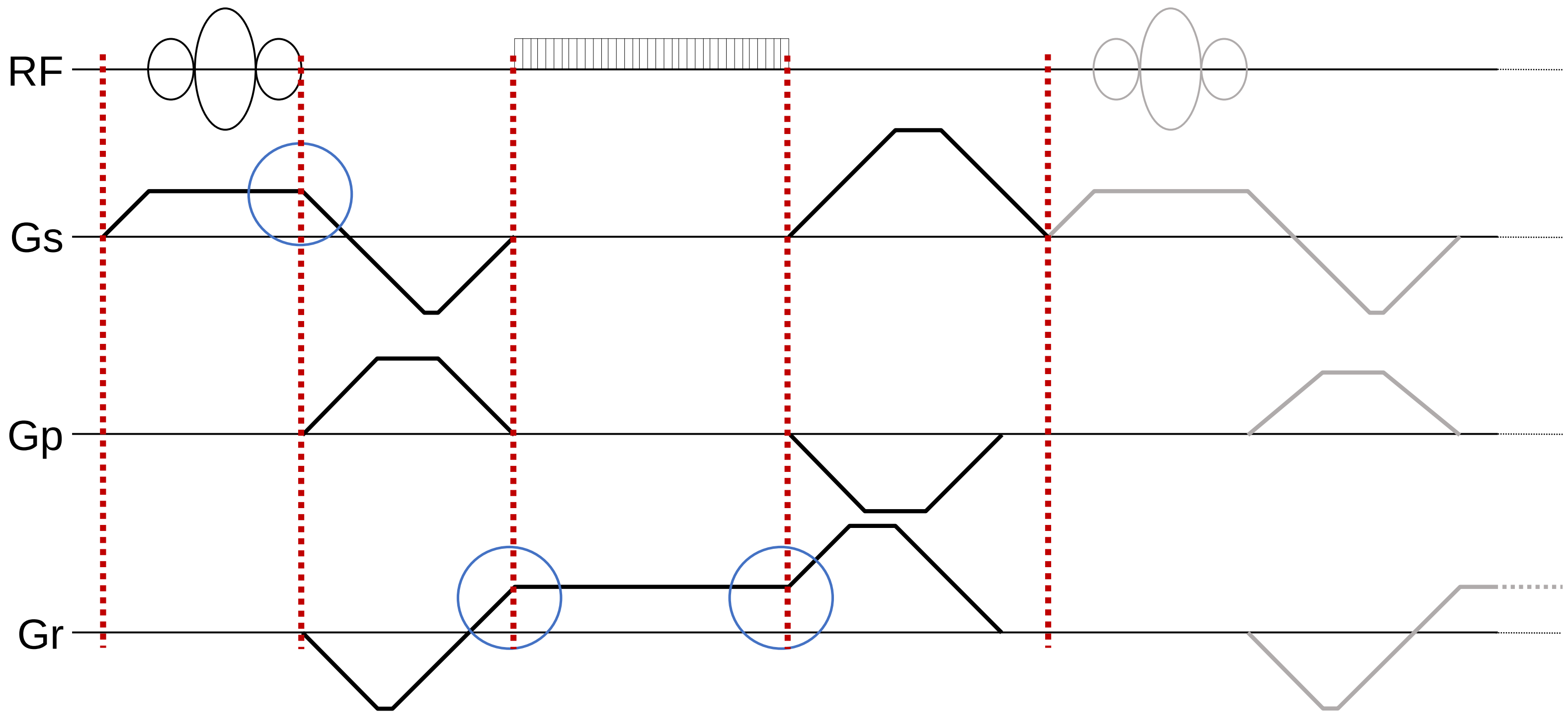
Pulseq ecosystem includes sequences, software to generate them and software and hardware to consume them

Pulseq 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!

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

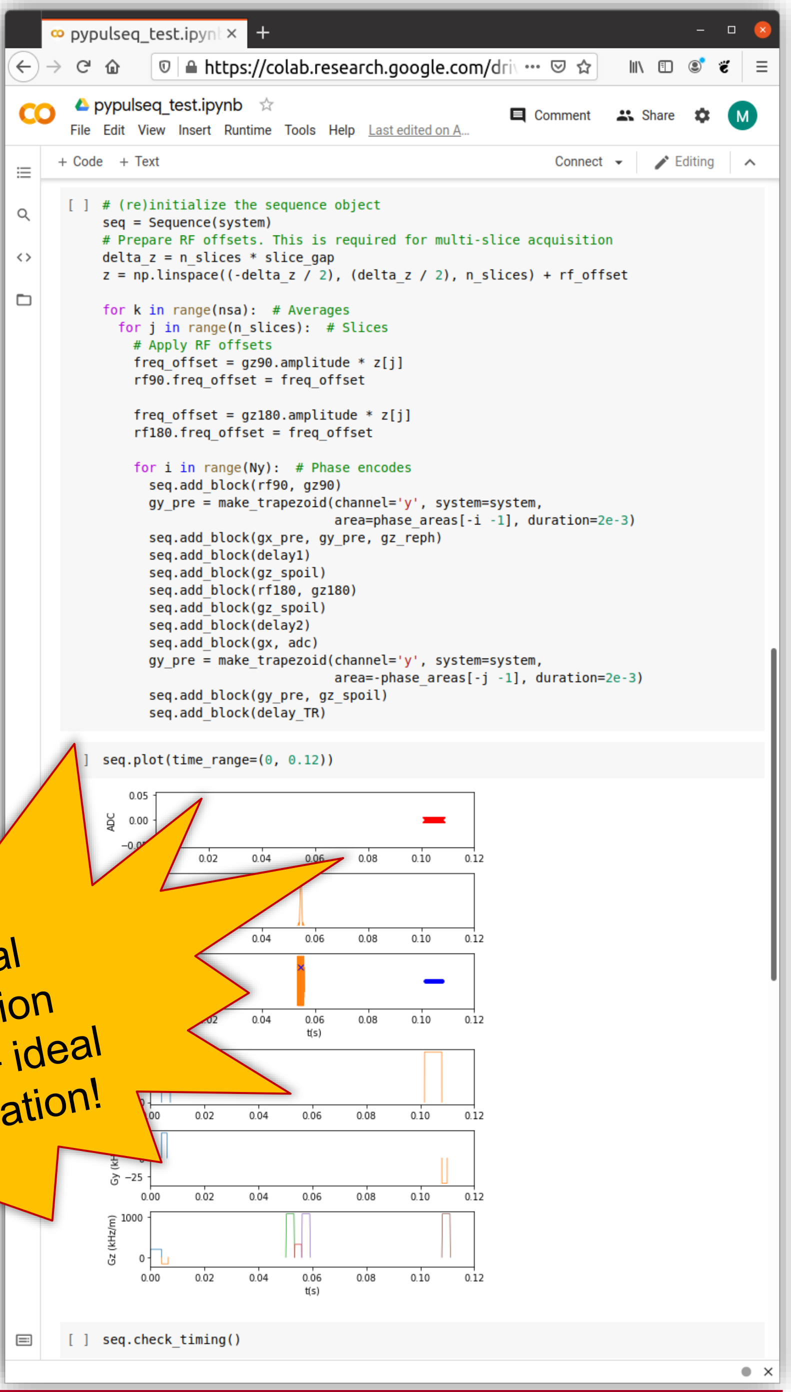
```
system = mr.opts('MaxGrad',30,'GradUnit','mT/m',...
    'MaxSlew',170,'SlewUnit','T/m/s');
seq=mr.Sequence(system);

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
- 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!*



No local installation needed – ideal for education!

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 them to the sequence

validation steps

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

Pulseq objects & blocks

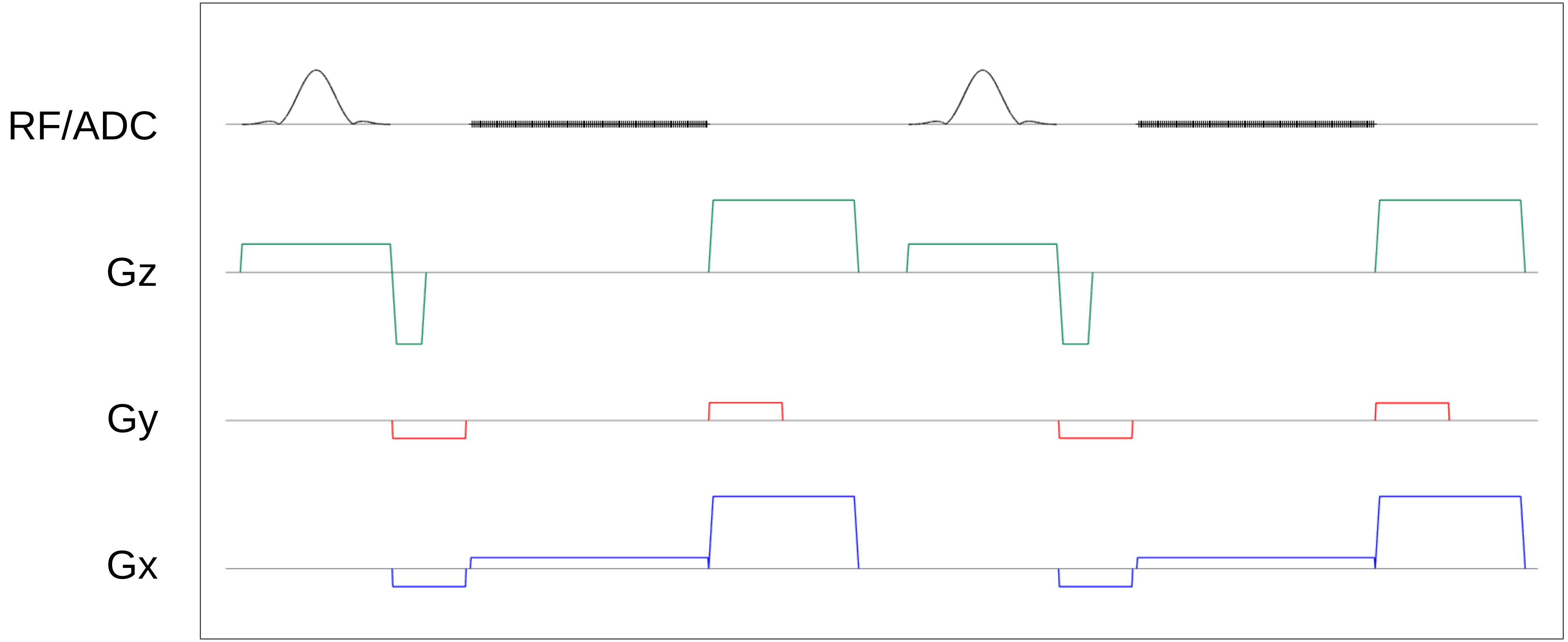
- *Pulseq* objects are created and then added to *Pulseq* blocks
  - Gradient & RF pulses

    mr.makeTrapezoid(…) or pp.make_trapezoid(…)

    mr.makeSincPulse(…) or pp.make_sinc_pulse(…)
  - ADC objects

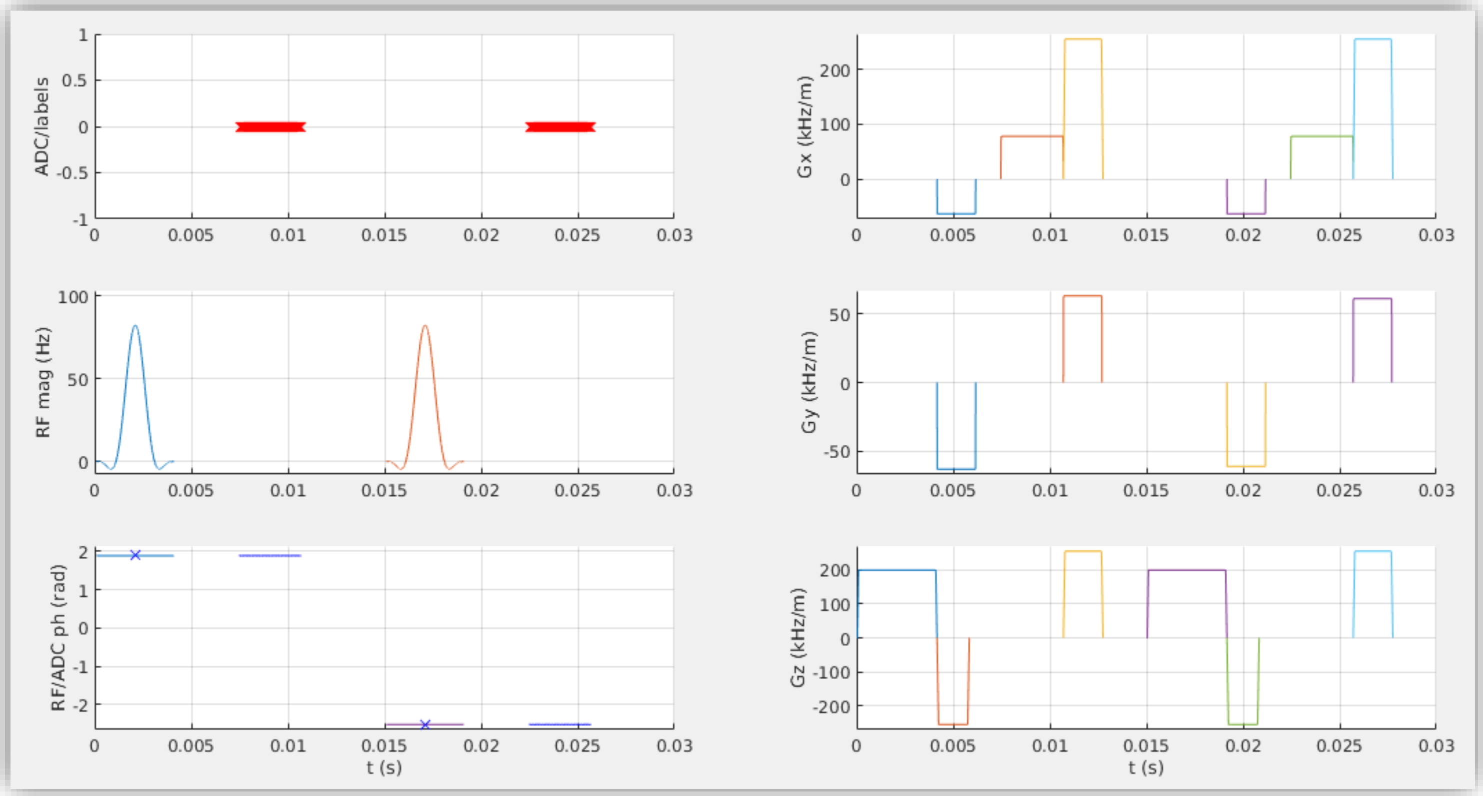
    mr.makeADC(…) or pp.make_adc(…)
  - Delays, extension objects, etc.
    (data labels, cardiac trigger directives, trigger pulses)
  - Use seq.addBlock(…) or seq.add_block(…) to add objects & blocks
- Block concept in *Pulseq* is probably the most demanding part
  - Blocks and objects need to be aligned to raster times

Example 1: simple gradient echo



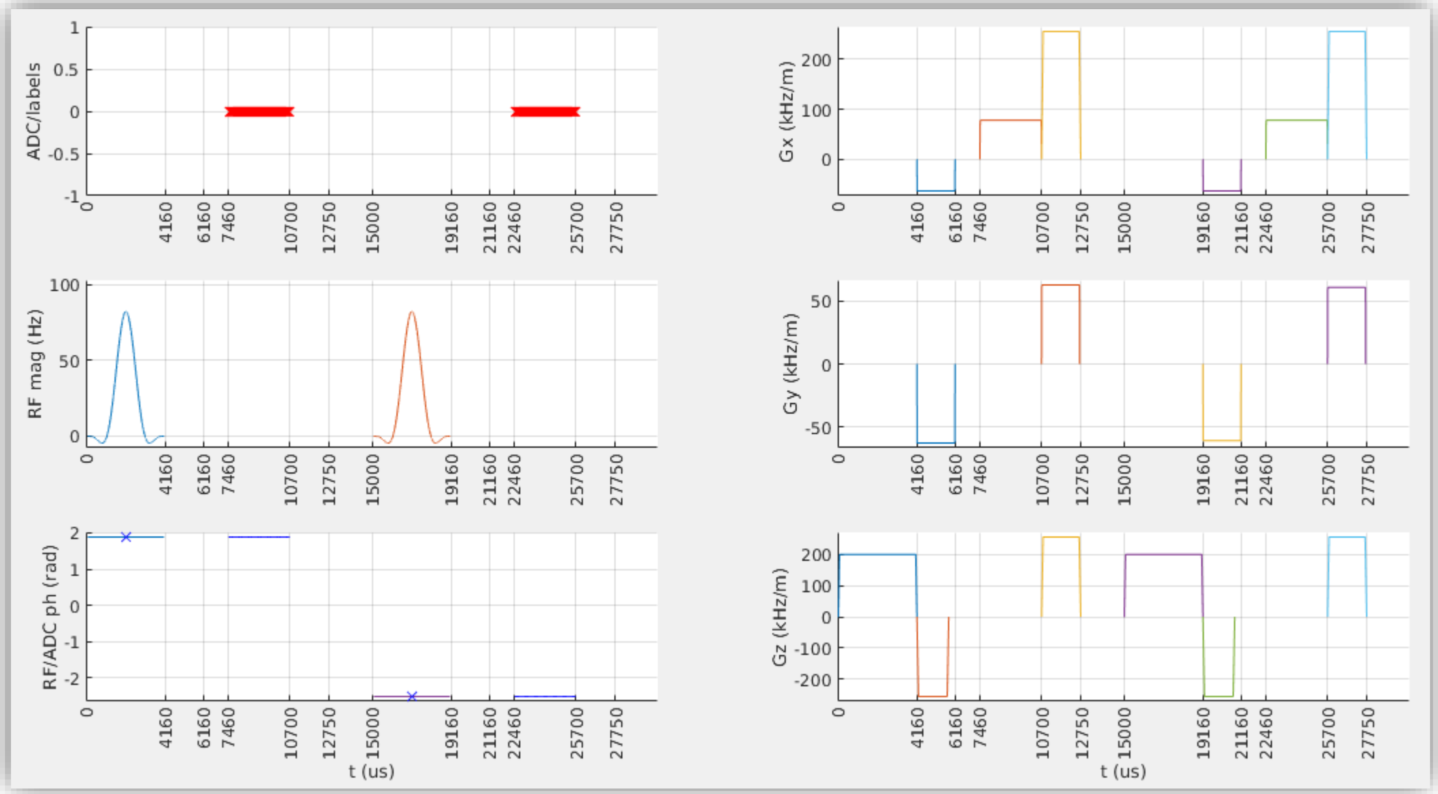
- No overlapping gradient ramps on different axes
- Events are clearly separated

Example 1: basic sequence display options



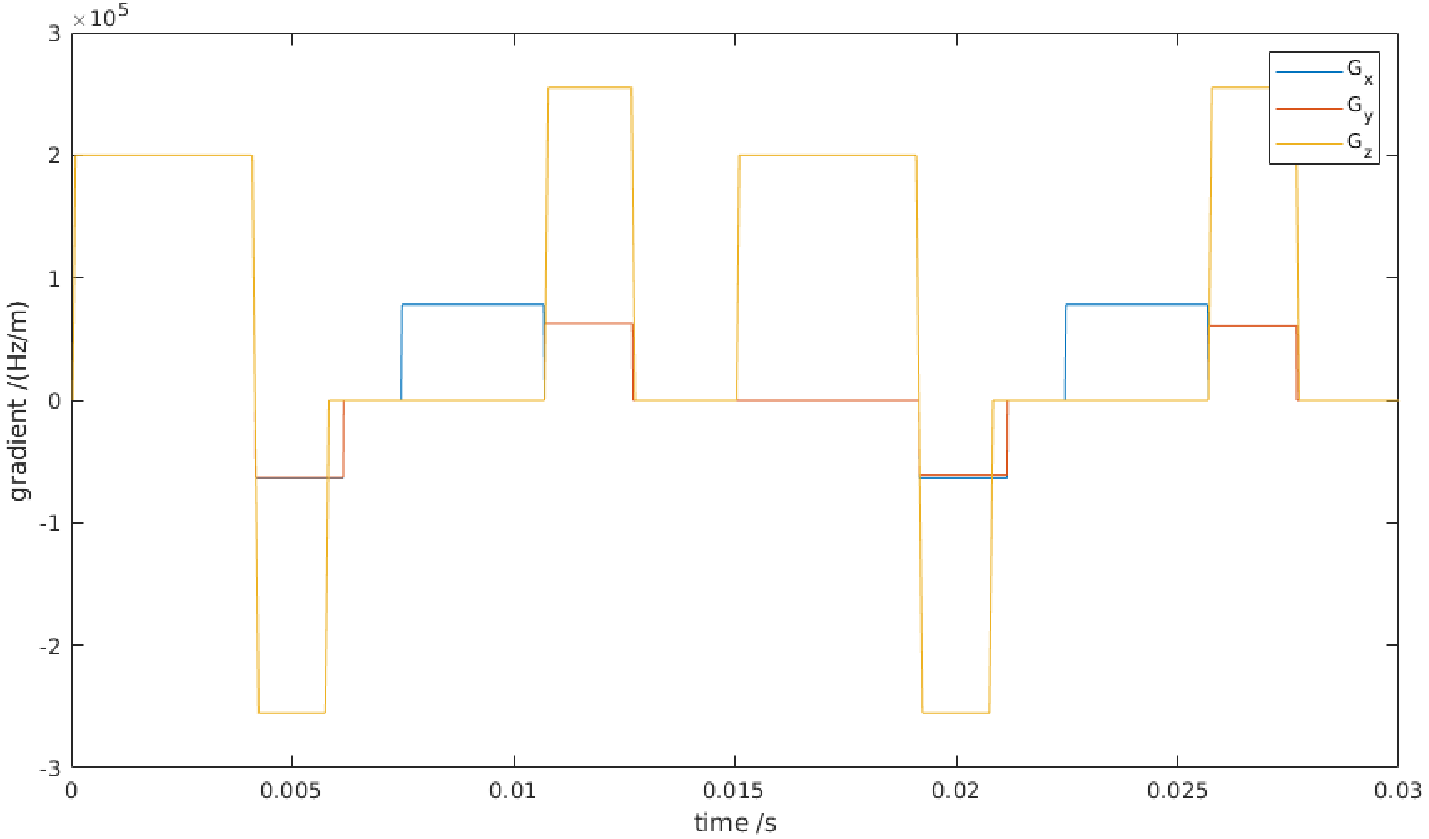
- Pulseq 6-panel plot: seq.plot()
  - ADC, RF magnitude, RF phase, Gx, Gy, Gz
  - Each event plotted in its own color

Example 1: display block structure



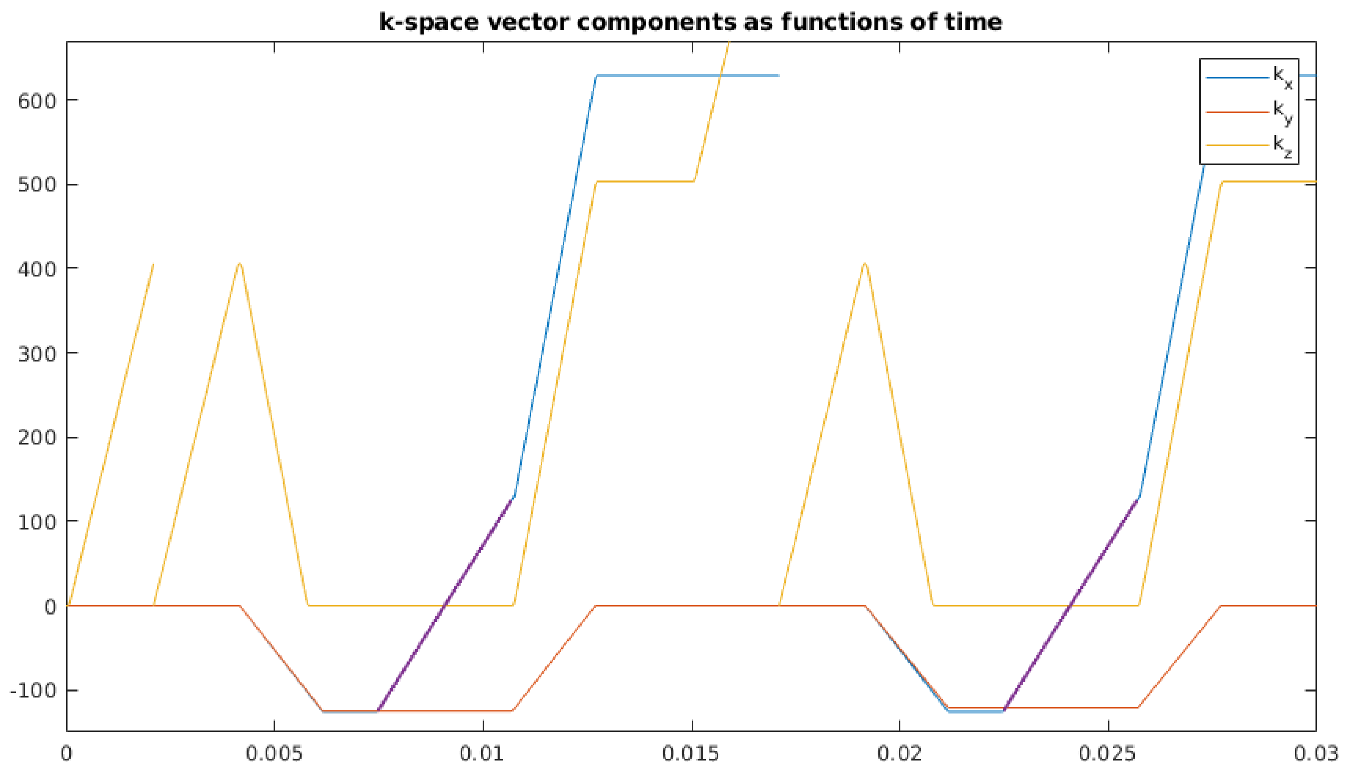
- Advantageous to separate PE & PR gradients into different blocks
- To visualize block structure:
  - seq.plot('showBlocks',true,'timeDisp','us');

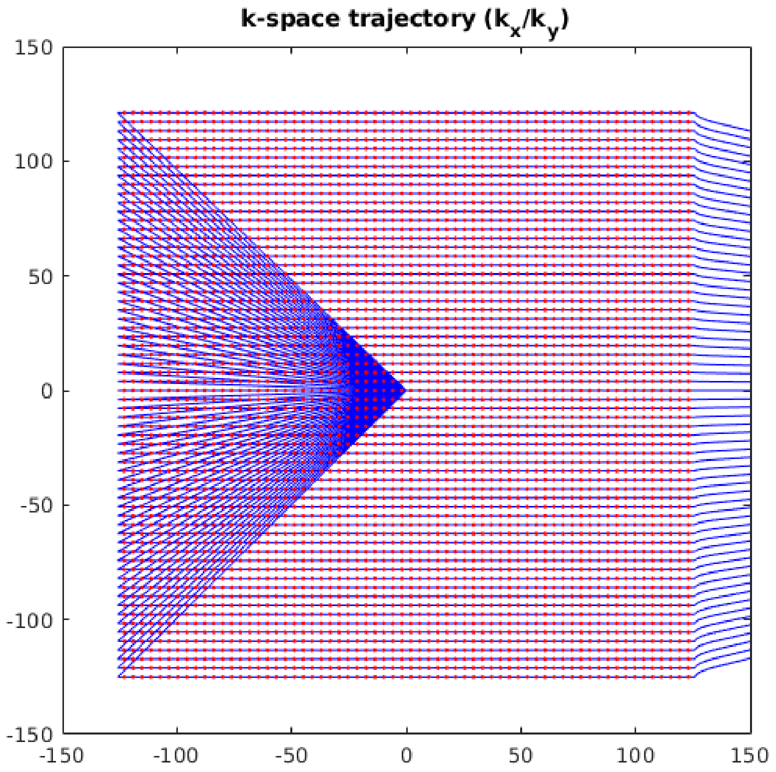
Plot gradient waveforms



- Plot entire waveforms for all axes
  - Native gradient unit in *Pulseq*: Hz/m

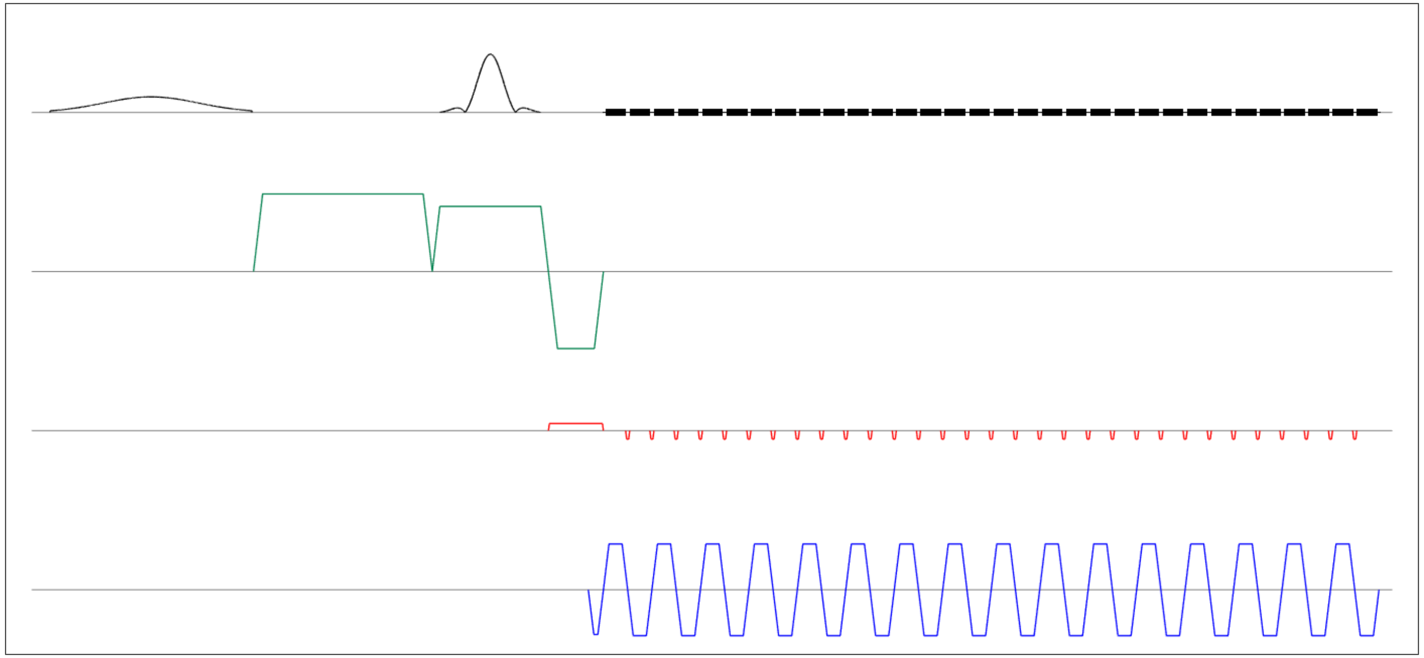
Basic sequence display options: k-space

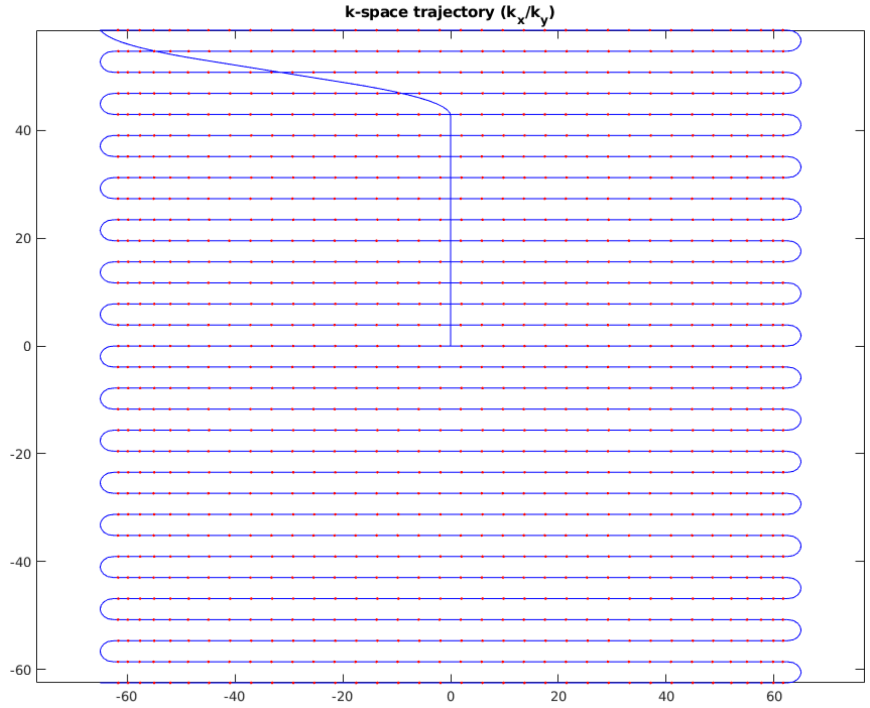




- Plot k-space time evolution or 2D (or even 3D) trajectories
  - Native k-space unit in *Pulseq*: m^{-1}

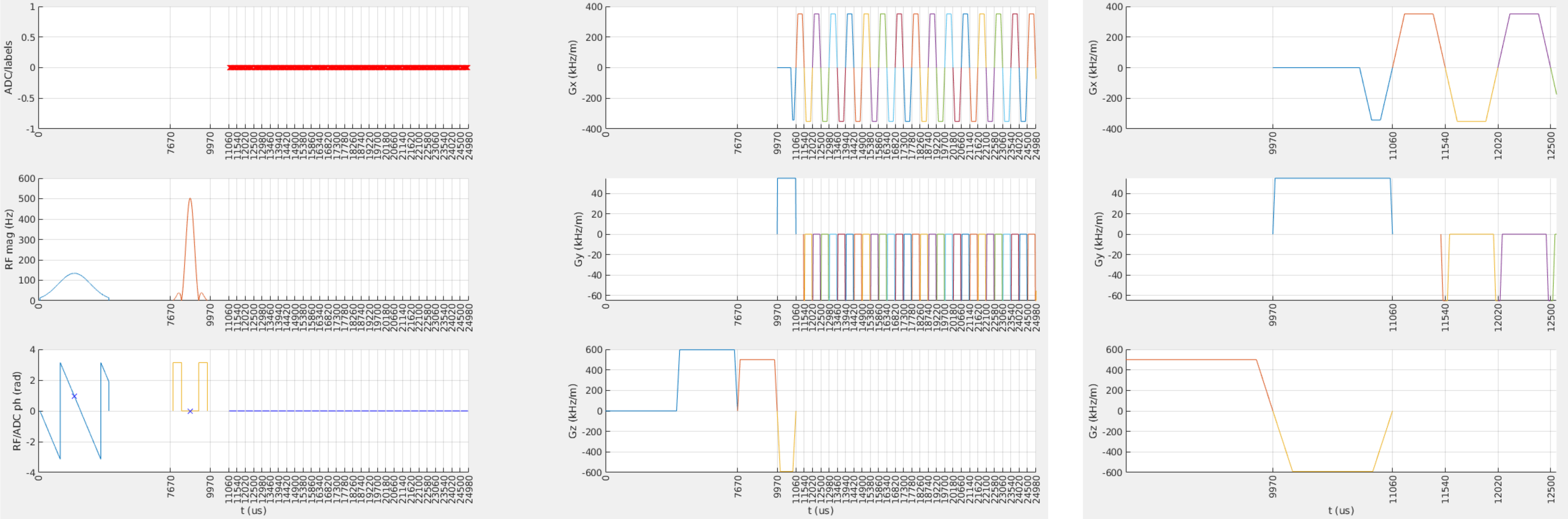
Example 2: echo planar imaging (EPI)





- 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



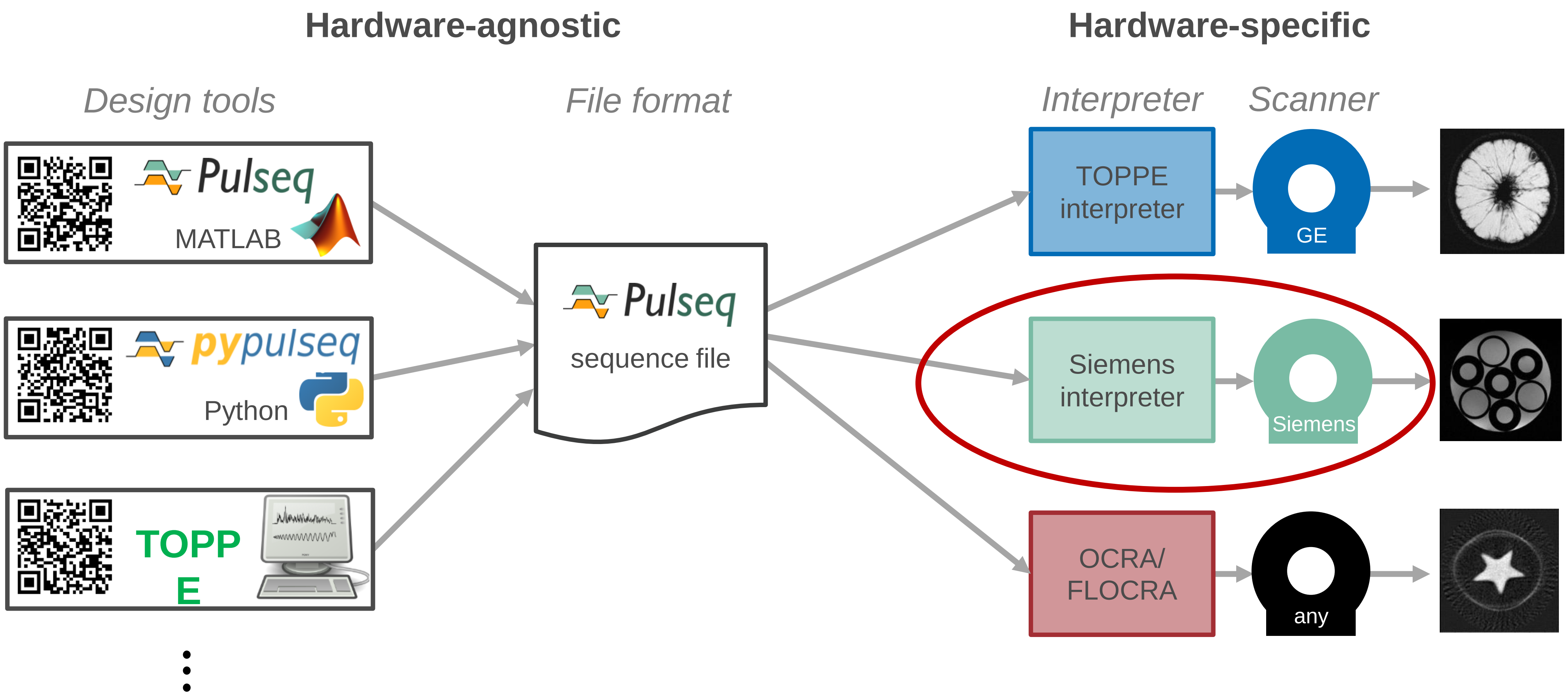
- One possible solution:
  - Keep readout gradient as trapezoid
  - Convert blips to shapes and split them at the center

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…

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

Pulseq framework overview



Before you start on the scanner

- Peripheral nerve analysis

- Avoiding acoustic resonances (for fast sequences)

Pulseq PNS prediction

- Peripheral neural stimulation (PNS) prediction on Siemens is based on the so-called SAFE model

  F.X. Herbank and M. Gebhardt. SAFE-Model - A New Method for Predicting Peripheral Nerve Stimulations in MRI. ISMRM 2000, #2007. https://cds.ismrm.org/ismrm-2000/PDF7/2007.PDF

- Open-source implementation by Filip Szczepankiewicz and Thomas Witzel: https://github.com/filip-szczepankiewicz/safe_pns_prediction

- Direct interface in Matlab-Pulseq

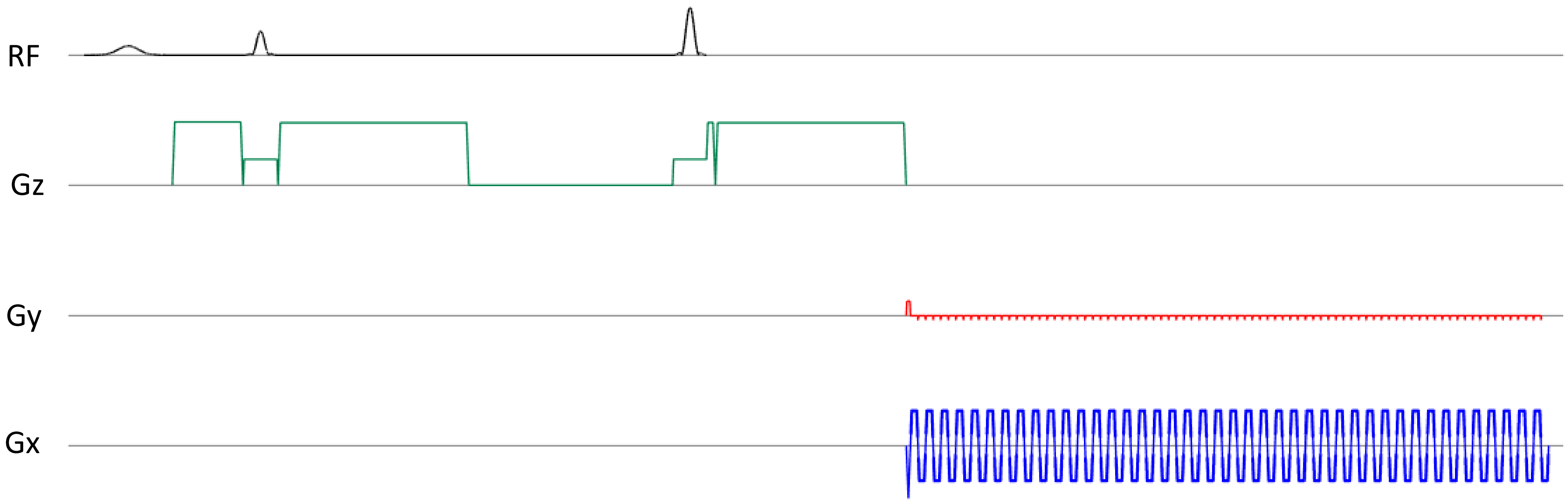
Caveat & Solution

- Need scanner-specific model parameters

- Extract your gradient system description file from IDEA
  - Type “sys” in the IDEA shell selecting your system
    (you may need to select something else first to see the verbose output)
  - Note the strings after “GPA Type” and “GC Type”
  - Go to C:\MIDEA\N4_VE####\n4\pkg\MrServers\MrMeasSrv\Config\InitMeas and pick the file named MP_GPA_<your_GPA>_<your_coil>.asc and copy it somewhere where your Matlab or Python can read it
  - *Path under NumarsX is left as an exercise for the interested reader*
- You can now predict PNS in the same way IDEA and scanner do it

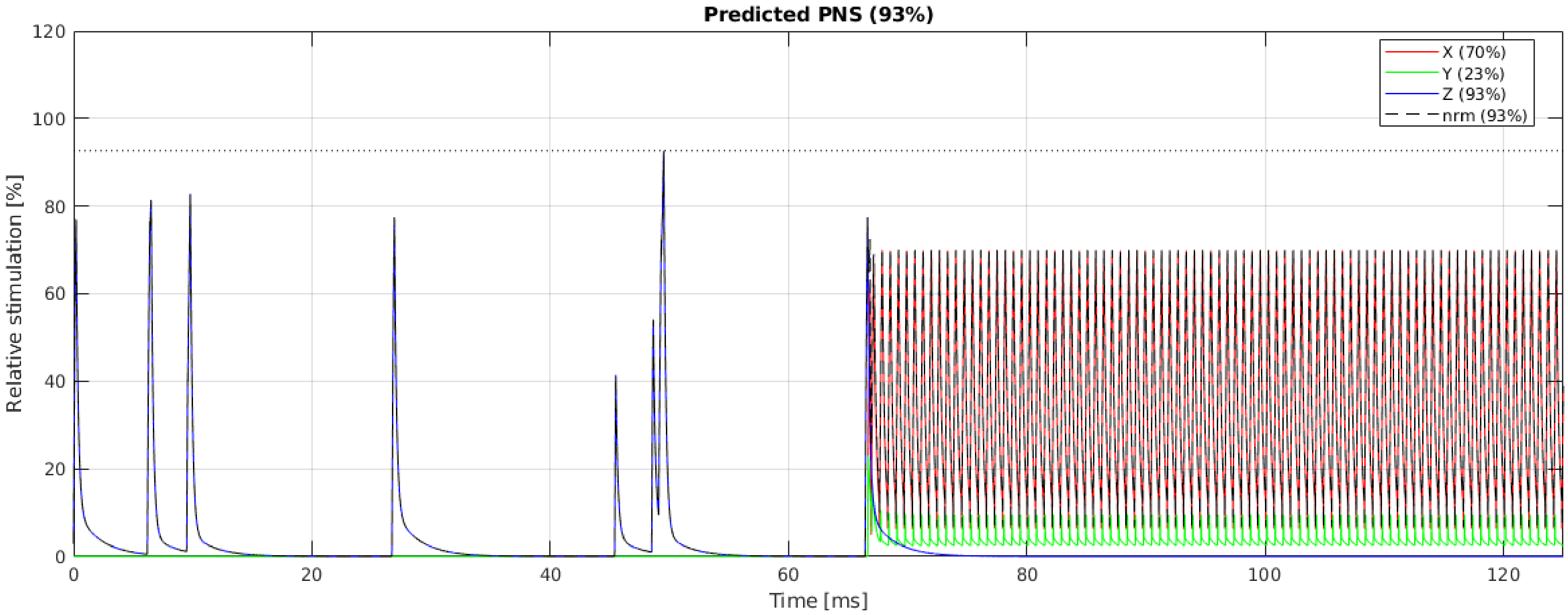
Example: DW-EPI

- bFactor=1000, TE=78ms, Gmax=38mT/m SR=180T/m/s



PNS Stimulation: DW-EPI

seq.calcPNS('idea/asc/MP_GPA_K2309_2250V_951A_AS82.asc');



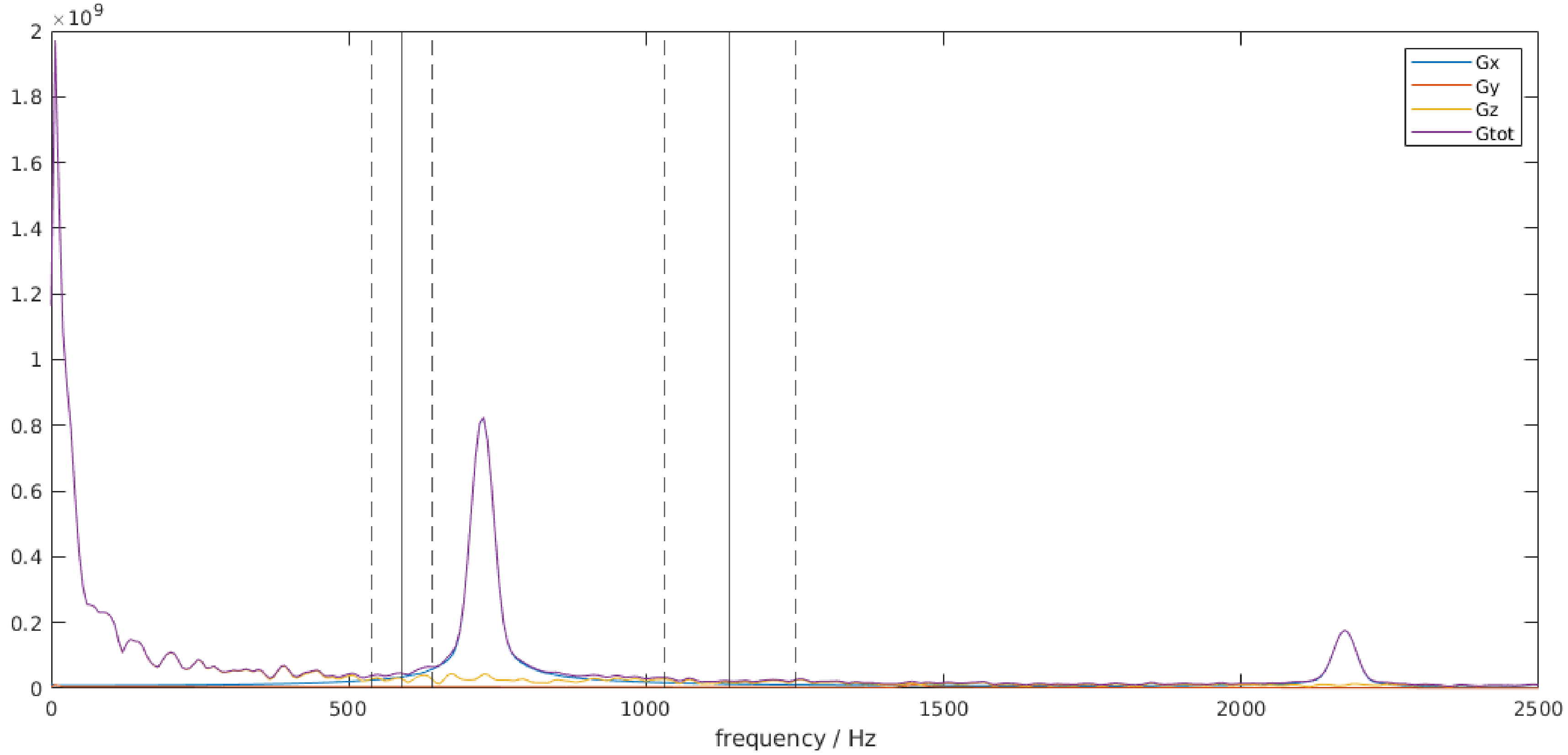
Acoustic resonance analysis

- Extract your gradient system description file from IDEA as above

- Create your sequence in memory
(populate the “seq” object with events)

- Run “gradSpectrum.m” script in “matlab/demoUnsorted”
  - Set ascName=‘….’ to your gradient system .asc file to see resonances marked

Acoustic analysis: DW-EPI

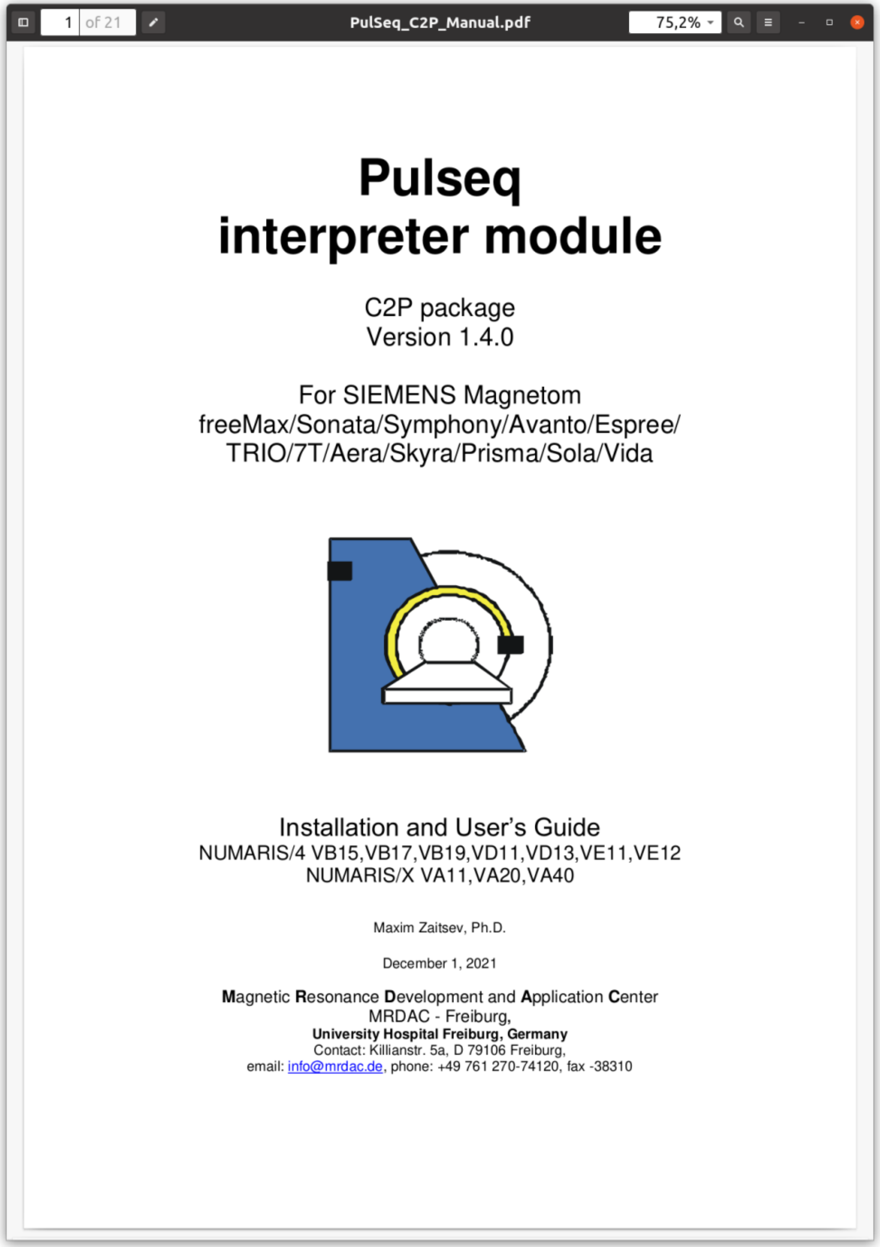


seq.sound(); % another option for acoustic assessment in Matlab ;-)

Pulseq Siemens interpreter

- Just a “normal” sequence
  - Loads its “content” from a Pulseq file
  - Almost all aspects of the sequence are pre-defined in the Pulseq file
  - FOV positioning and rescaling possible
- Based on miniFlash
  - No product code
  - No hacks, no backdoors
- Distributed as a C2P package in **source form**

- Standard SAR calculation
- “Gradient Health” libBalance applicable to all sequences
- PNS and acoustic resonance analysis possible in Matlab
- **Safety equal or higher than a typical IDEA sequence**



Pulseq
interpreter module

C2P package
Version 1.4.0

For SIEMENS Magnetom
freeMax/Sonata/Symphony/Avanto/Espree/
TRIO/7T/Aera/Skyra/Prisma/Sola/Vida

Installation and User's Guide
NUMARIS/4 VB15,VB17,VB19,VD11,VD13,VE11,VE12
NUMARIS/X VA11,VA20,VA40

Maxim Zaitsev, Ph.D.

December 1, 2021

Magnetic **R**esonance **D**evelopment and **A**pplication **C**enter
MRDAC - Freiburg,
University Hospital Freiburg, Germany
Contact: Killianstr. 5a, D 79106 Freiburg,
email: info@mrdac.de, phone: +49 761 270-74120, fax -38310

Pulseq on Siemens platforms

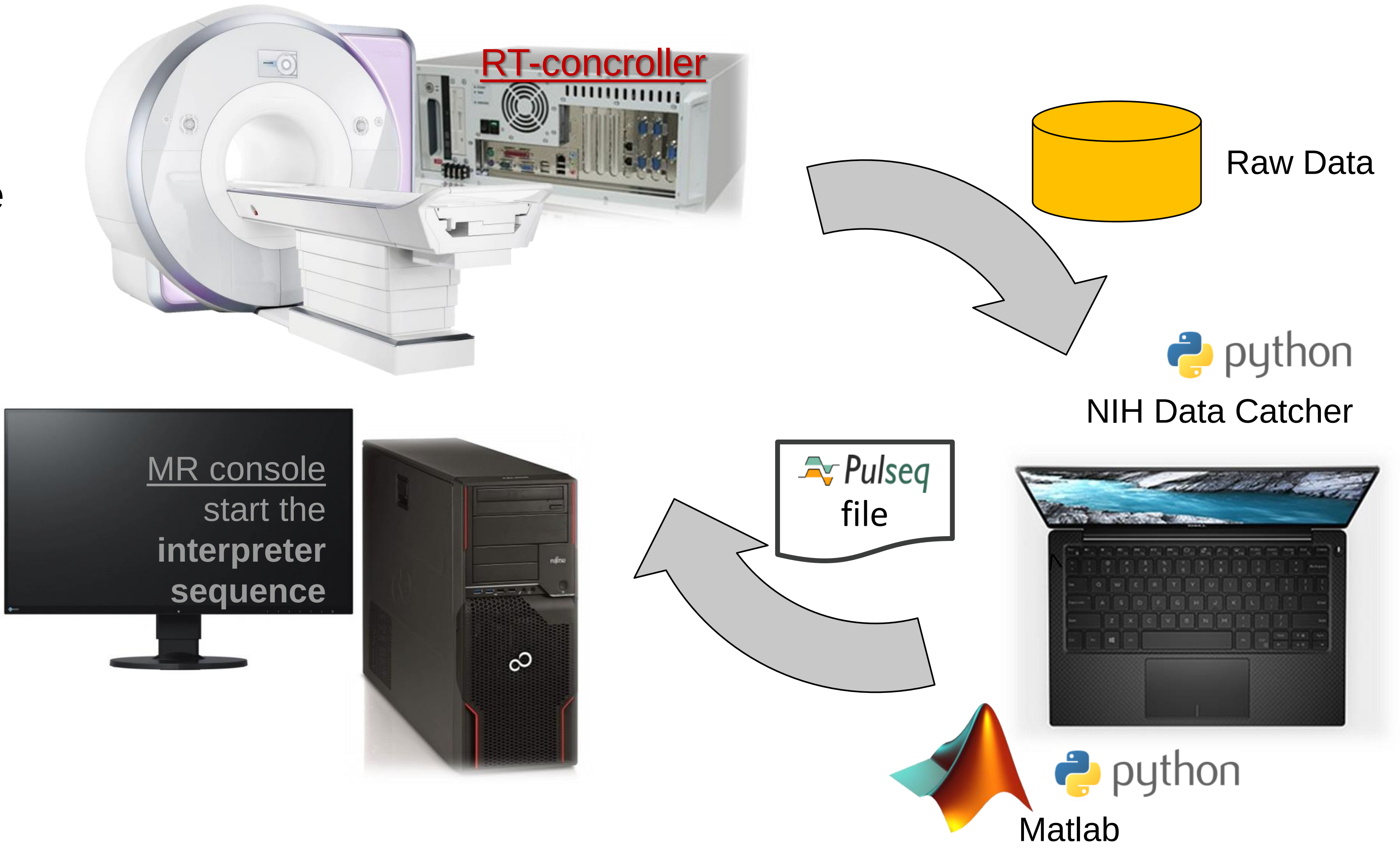
- Over 100 C2P sites
- *Pulseq* works well on:
  - All Numaris4 platforms (tested on vb15…ve12u) and numerous hardware platforms (Symphony, Trio, 7T, 3T Connectom, Skyra, Prisma,…)
  - All released NumarisX versions (xa11, xa20, xa30, xa40, xa5x, xa6x …)





Pulseq on Siemens scanners

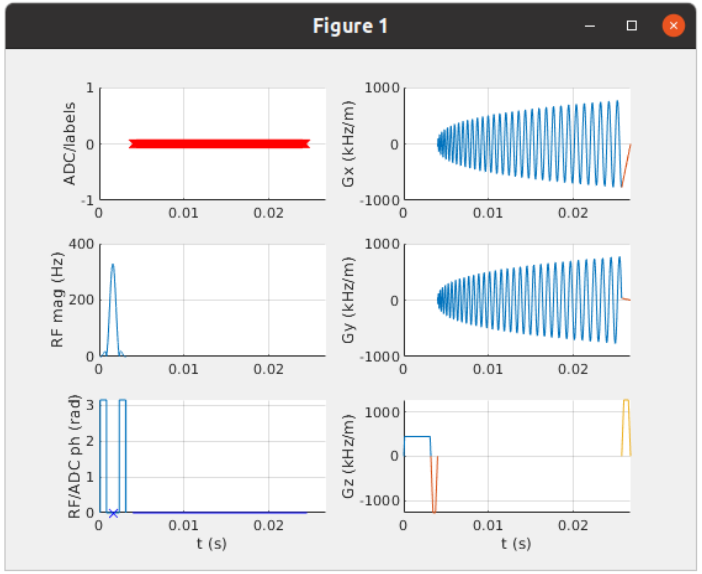
- *Optional initial step: connect your PC to the scanner*
- Save the .seq file on the scanner as external.seq
- Run the interpreter_sequence on the scanner
- *Optional step: stream raw data to your PC with NIH_DataCatcher*
- or export raw data manually

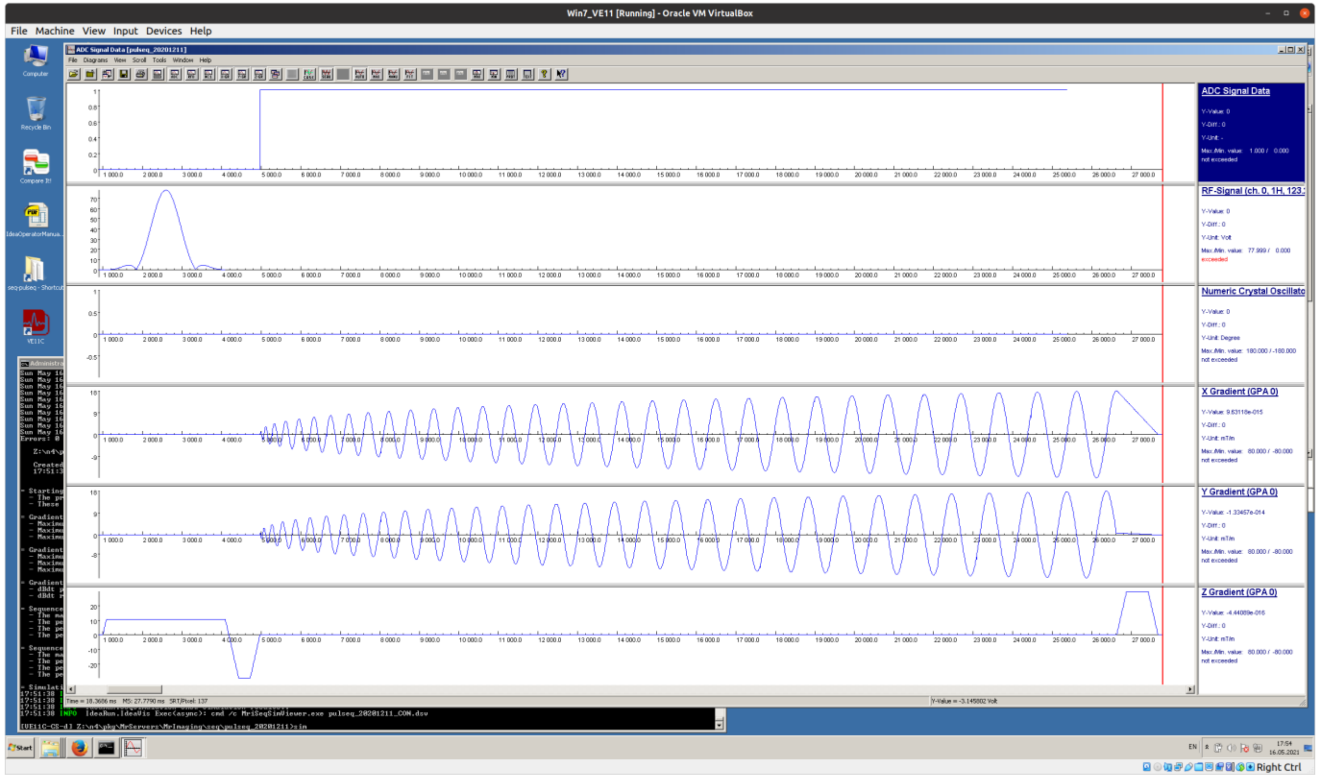


IDEA simulation with *Pulseq*

Pulseq interpreter sequence can also be used with the Siemens' IDEA

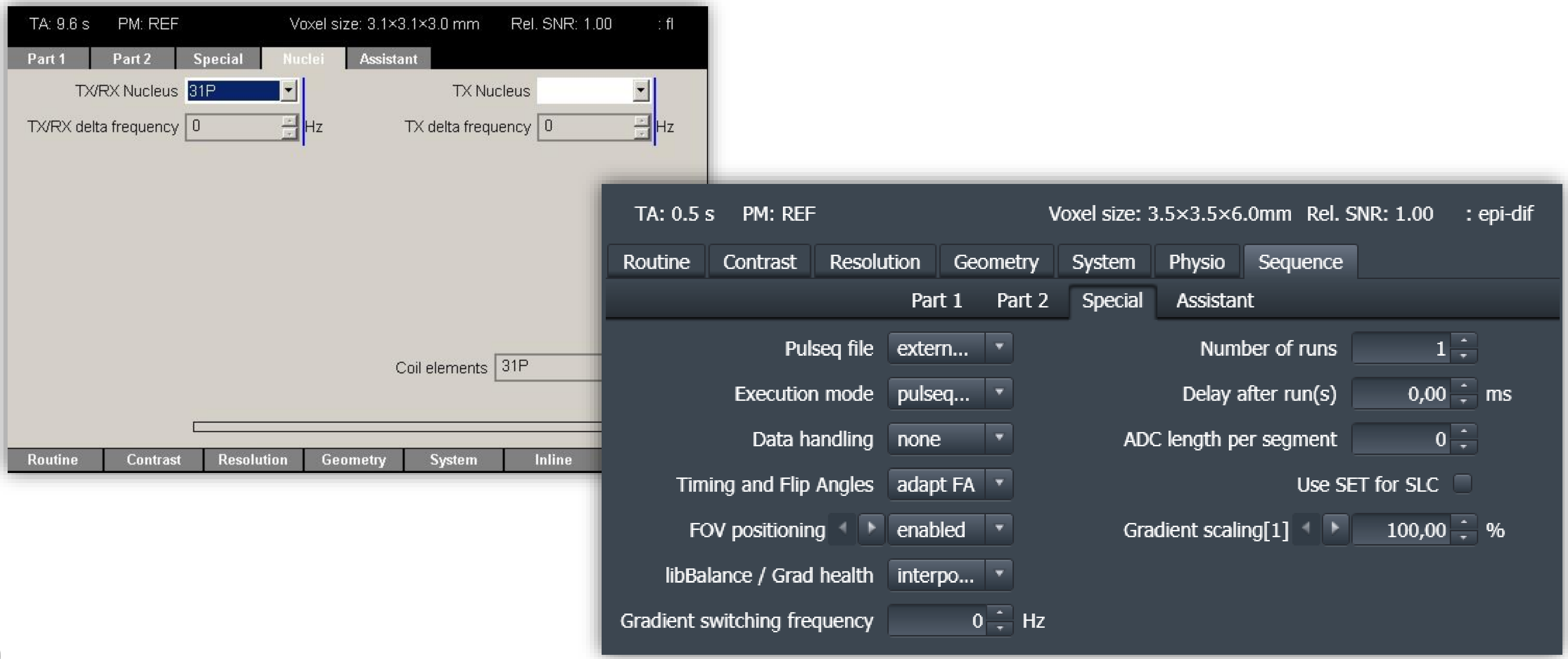
1. Save your .seq file as %CustomerSeq%/Pulseq/external.seq
2. In the IDEA command run
   sim





Pulseq interpreter parameters

- Yes, you can still change few things on the console



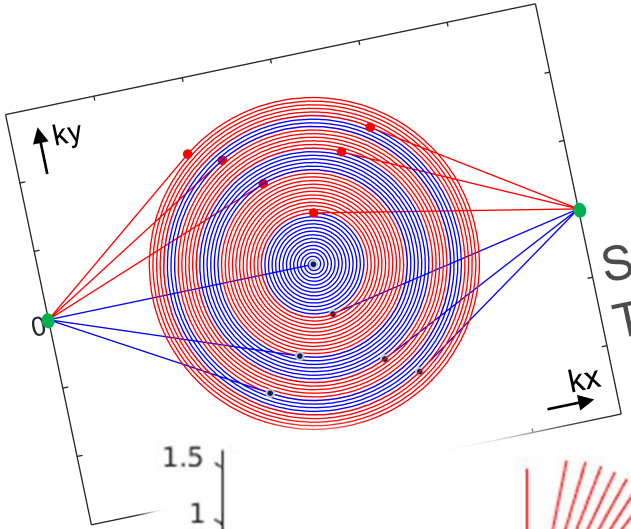
Pulseq for Siemens: image recon

- Image reconstruction is up to the researcher
- Integration with ICE is possible for some sequences
  - 2D GRE
  - 2D EPI with ramp sampling
  - 3D MPRAGE with GRAPPA
- Online & offline reconstruction with Gadgetron for some sequences
- Offline reconstruction in Matlab & Python
  - Examples for 2D / 3D Cartesian reconstruction
  - Simple gridding reconstruction
  - Example of automated BART reconstuction
- All-in-one FIRE solution : Marten Veldmann et all, MRM 2022, doi:10.1002/mrm.29384

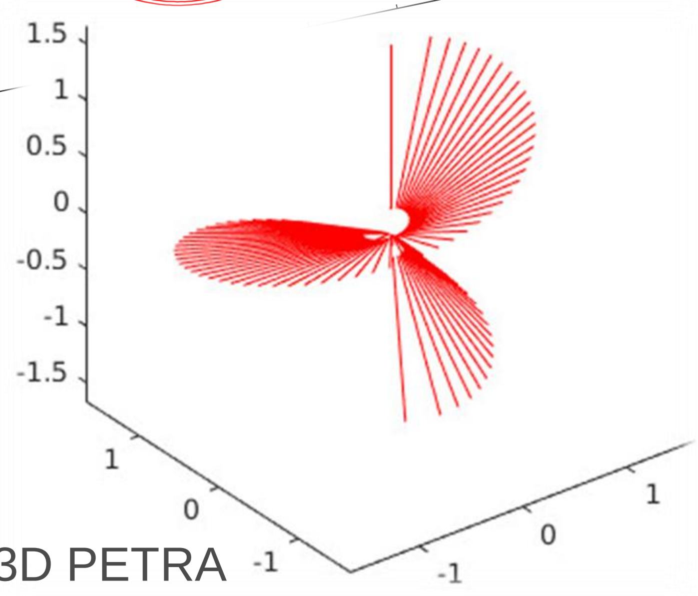
Qingping Chen @ ISMRM

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
- Pulseq files play out on many scanners
  - Siemens & GE : works
  - Philips : two working interpreters @ISMRM24
  - *Bruker ???*
- New release v1.4.2 (Dec 2023)



Spiral TSE



3D PETRA

UNIVERSITÄTS KLINIKUM FREIBURG

Acknowledgements:

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