Working with Pulseq objects

Jon-Fredrik Nielsen

November 15, 2023

Preliminaries

- Code for this PDF (Latex, MATLAB) available on course Github site
- Custom functions written for this talk end with '__' suffix. Example:

```
>> sincrf__;          % creates example Pulseq events ('rf' and 'gz')
```

- Other functions belong to the Pulseq MATLAB toolbox, either as standalone functions or attached to a 'seq' object. Examples:

```
>> [rf, gz] = mr.makeSincPulse(pi/2, 'Duration', 2e-3, 'SliceThickness', 5e-3, ...
                'apodization', 0.42, 'timeBwProduct', 4, 'system', sys);
>> seq.plot('timeRange', [0 0.1]);
```

Outline

Units

- RF
  - Unit: Hz
  - divide by sys.gamma to convert to Tesla
- Gradients
  - Unit: Hz/m
  - divide by sys.gamma to convert to Tesla/m
- Time in seconds
- Flip angles in radians
- RF/ADC phase in radians

Gradient events

3 types:

- Trapezoid
- Extended trapezoid
- Arbitrary gradient

Pulseq toolbox defaults

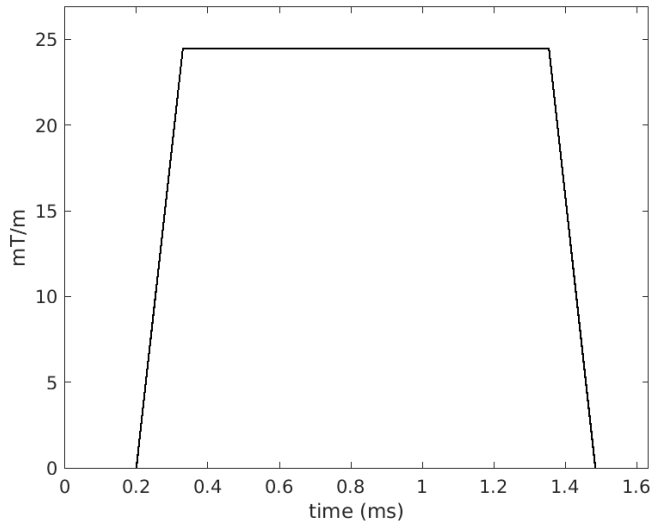
- gradRasterTime = 10e-6 s

Trapezoid

Shape specified by rise/flat/fall times

```
Nx = 256;
fov = 240e-3;       % m
dwell = 4e-6;       % sec
deltak = 1/fov;
gx = mr.makeTrapezoid('x', 'FlatArea', Nx*deltak, 'FlatTime', Nx*dwell, ...
                      'delay', 200e-6, 'system', sys);
```

```
gx =
          type: 'trap'
       channel: 'x'
     amplitude: 1.0417e+06
      riseTime: 1.3000e-04
      flatTime: 0.0010
      fallTime: 1.3000e-04
          area: 1.2021e+03
      flatArea: 1.0667e+03
         delay: 2.0000e-04
         first: 0
          last: 0
```

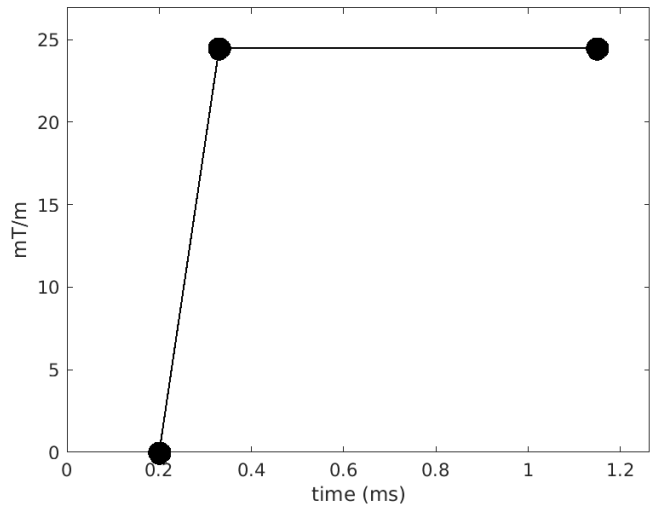


Extended trapezoid

Shape sampled on vertices

```
gxtmp = mr.makeTrapezoid('x', 'FlatArea', Nx*deltak, 'FlatTime', Nx*dwell, ...
                         'delay', 200e-6, 'system', sys);
[gx,~] = mr.splitGradientAt(gxtmp, gxtmp.riseTime + gxtmp.flatTime);

% Or:
gx = mr.makeExtendedTrapezoid('x', 'times', gx.tt, 'amplitudes', gx.waveform);
gx.delay = 200e-6;
```

```
gx =
          type: 'grad'
       channel: 'x'
      waveform: [0  1.0417e+06  1.0417e+06]
         delay: 2.0000e-04
            tt: [0  1.3000e-04  9.5000e-04]
     shape_dur: 9.5000e-04
          area: 921.8750
         first: 0
          last: 1.0417e+06
```



Arbitrary gradient

Shape sampled on regular raster, on center of raster periods

```
spiral;   % creates 'wav' vector, mT/m
gx = mr.makeArbitraryGrad('x', wav*sys.gamma/1e3, 'delay', 200e-6);
```

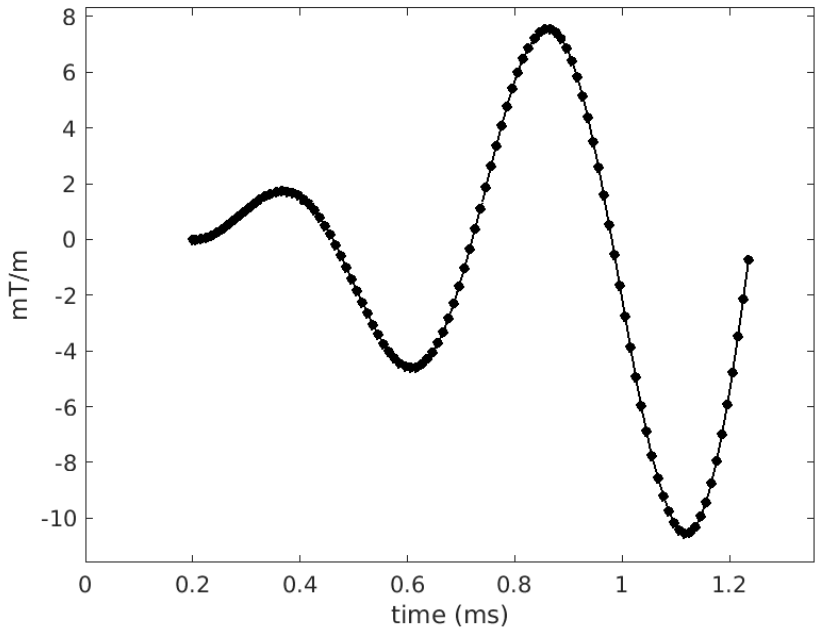
```
gx =
          type: 'grad'
       channel: 'x'
      waveform: [148.3082 1.3283e+03 3.6538e+03 7.0567e+03 1.1437e+04 1.6663e+04 2.2579e+04 2
         delay: 2.0000e-04
          area: -42.3091
            tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.5000e-05
     shape_dur: 0.0010
         first: -441.6805
          last: -666.5979
```

Arbitrary gradient

Shape sampled on regular raster, on center of raster periods

```
spiral;   % creates 'wav' vector, mT/m
gx = mr.makeArbitraryGrad('x', wav*sys.gamma/1e3, 'delay', 200e-6);
```

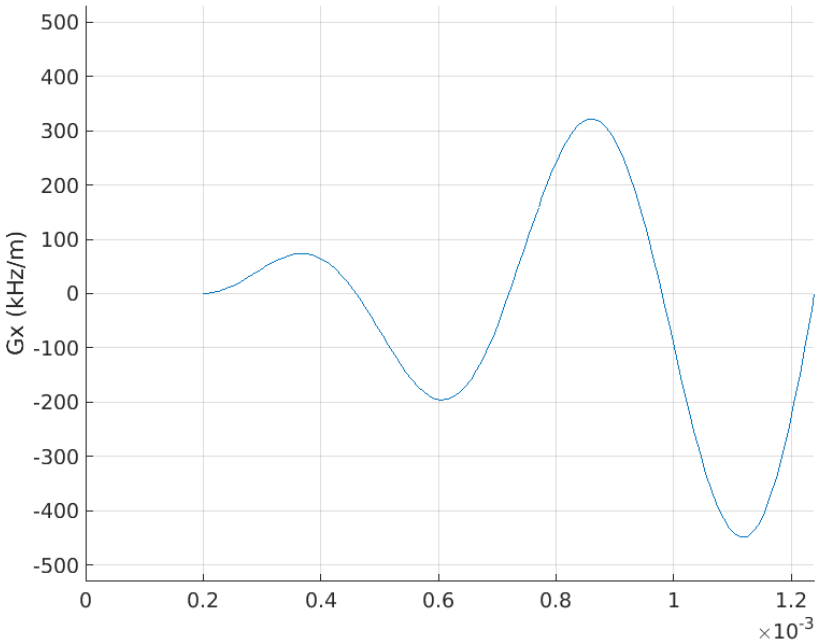
```
gx =
          type: 'grad'
       channel: 'x'
      waveform: [148.3082 1.3283e+03 3.6538e+(
         delay: 2.0000e-04
          area: -42.3091
            tt: [5.0000e-06 1.5000e-05 2.5000e
     shape_dur: 0.0010
         first: -441.6805
          last: -666.5979
```

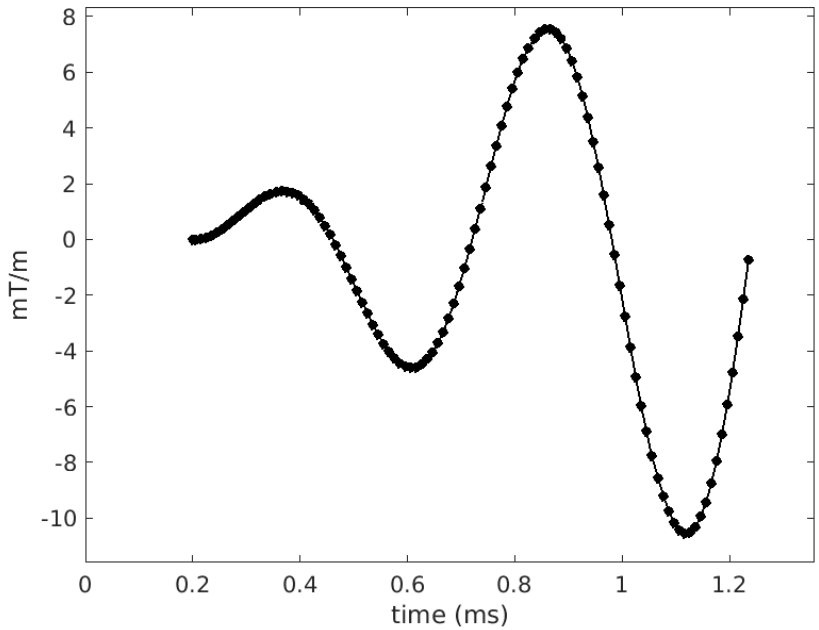


Arbitrary gradient

Shape sampled on regular raster, on center of raster periods

```
spiral;   % creates 'wav' vector, mT/m
gx = mr.makeArbitraryGrad('x', wav*sys.gamma/1e3, 'delay', 200e-6);
```

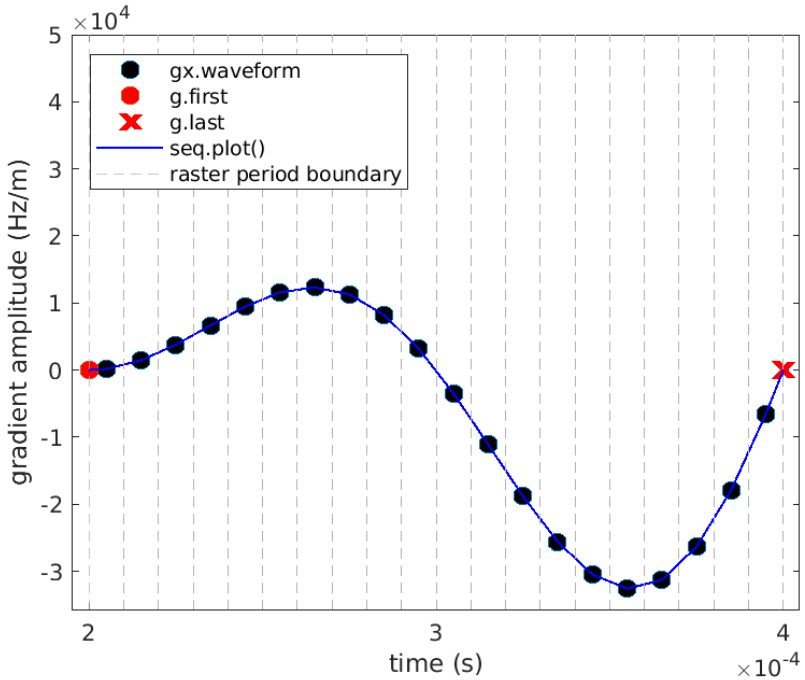




'first' and 'last'

- Waveform values at beginning of first raster period, and end of last raster period
- Calculated by makeArbitraryGradient(), or set manually
- Uses: seq.plot(); k-space calculation; check for waveform continuity across blocks
- not yet saved in .seq file (likely to change in the future)

```
gx =
        type: 'grad'
     channel: 'x'
    waveform: [166.5088  1.4497e+03  3.7632e+03  6.
       delay: 2.0000e-04
        area: -1.3608
          tt: [5.0000e-06  1.5000e-05  2.5000e-05
  shape_dur: 2.0000e-04
       first: 0
        last: 0
```



Modify gradients with scaleGrad()

- Function returns a scaled copy of the input event
- Typically used inside scan loop, e.g., to scale the y phase encode gradient:

```
for  j = -Ny/2:Ny/2-1
     seq.addBlock(mr.scaleGrad(gy, j/(Ny/2)));
     ...
end
```

- Opinion: it is good programming practice use scaleGradient() instead of creating a new event from scratch
  - Makes the purpose of the gradient event, and relationship to the input gradient, clear
  - Scaled events share the same shape ID $\rightarrow$ simplifies interpreter code

Modify gradients with rotate()

- Function returns single-axis gradient events resulting from rotation

```
gx =
          type: 'grad'
       channel: 'x'
      waveform: [166.5088 1.4497e+03 3.7632e+03 6.6387e+03 9.4617e+03 1.1564e+04 1.2329e
         delay: 2.0000e-04
          area: -1.3608
            tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.500
     shape_dur: 2.0000e-04

>> [gxr gyr] = mr.rotate('z',pi/4,gx)
gxr =
          type: 'grad'
       channel: 'x'
      waveform: [117.7395 1.0251e+03 2661 4.6943e+03 6.6904e+03 8.1772e+03 8.7179e+03 7.
         delay: 2.0000e-04
          area: -1.3608
            tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.500
     shape_dur: 2.0000e-04

gyr =
          type: 'grad'
       channel: 'y'
      waveform: [117.7395 1.0251e+03 2.6610e+03 4.6943e+03 6.6904e+03 8.1772e+03 8.7179e
         delay: 2.0000e-04
          area: -1.3608
            tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.500
     shape_dur: 2.0000e-04
```

Gradient 'surgery' with splitGradientAt()

```
gyBlip =
          type: 'trap'
       channel: 'y'
     amplitude: 5.5556e+05
      riseTime: 9.0000e-05
      flatTime: 0
      fallTime: 9.0000e-05
          area: 50

>> gy_parts = mr.splitGradientAt(gyBlip, gyBlip.riseTime);
gy_parts(1) =
          type: 'grad'
       channel: 'y'
      waveform: [0  5.5556e+05]
         delay: 0
            tt: [0  9.0000e-05]
     shape_dur: 9.0000e-05
          area: 25

gy_parts(2) =
          type: 'grad'
       channel: 'y'
      waveform: [5.5556e+05  0]
         delay: 9.0000e-05
            tt: [0  9.0000e-05]
     shape_dur: 9.0000e-05
          area: 25
```

addGradients()

Returns sum of gradients

```
gy =
          type: 'trap'
       channel: 'y'
     amplitude: 8.3333e+05
      riseTime: 1.2000e-04
      flatTime: 0
      fallTime: 1.2000e-04
          area: 100
      flatArea: 0
         delay: 0

>> gy_sum = mr.addGradients({gy, gy}, 'system', sys)

gy_sum =
          type: 'trap'
       channel: 'y'
     amplitude: 1.6667e+06
      riseTime: 1.2000e-04
      flatTime: 0
      fallTime: 1.2000e-04
          area: 200
      flatArea: 0
         delay: 0
```

RF events

- Shape sampled on regular raster on center of raster periods, or on vertices
- Pulseq toolbox defaults
  - rfRasterTime = 1e-6 s

RF event: Example 1

sinc pulse

```
[rf, gz] = mr.makeSincPulse(20/180*pi, 'Duration', 2e-3, 'SliceThickness', 5e-3, ...
    'apodization', 0.42, 'timeBwProduct', 4, 'system', sys);
```

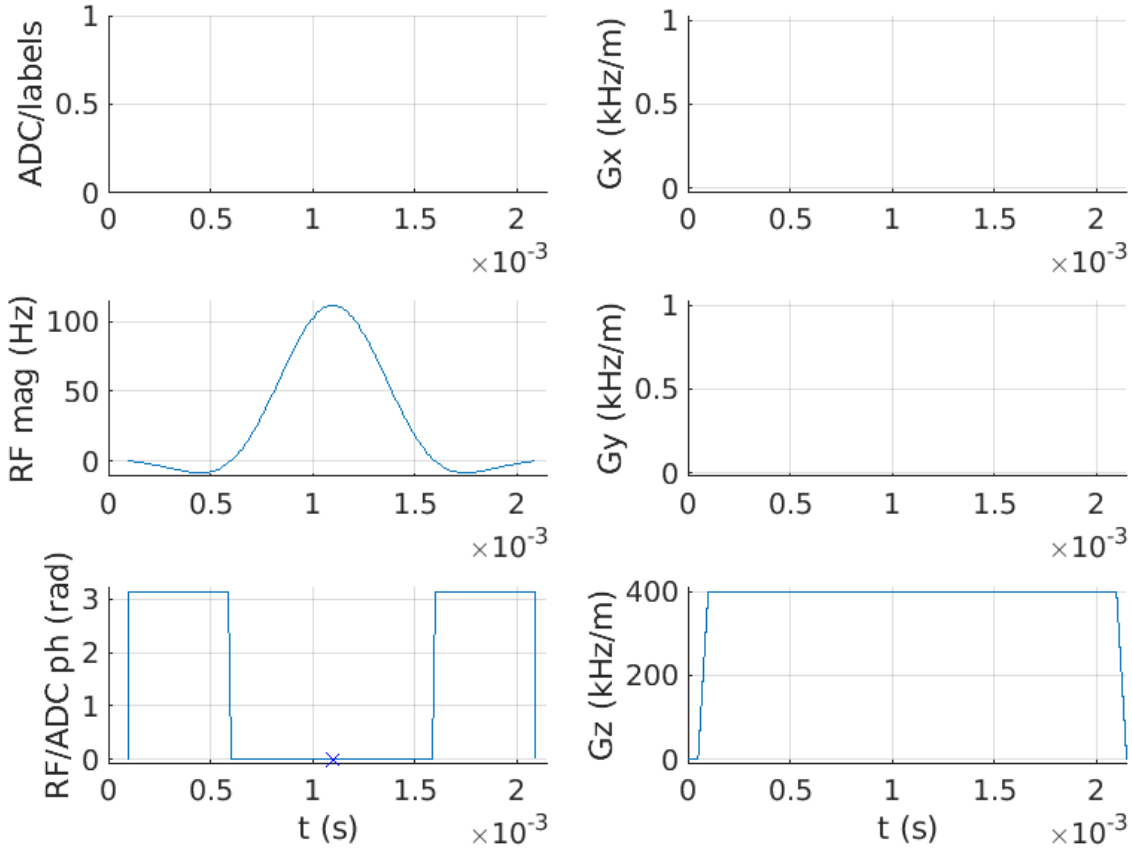
```
rf =
              type: 'rf'
            signal: [-0.0089 -0.0268 -0.0448 -0.0628 -0.0808 -0.0988 -0.1169 -0.1350 -0.1532 -
                 t: [5.0000e-07 1.5000e-06 2.5000e-06 3.5000e-06 4.5000e-06 5.5000e-06 6.5000e-
         shape_dur: 0.0020
        freqOffset: 0
       phaseOffset: 0
          deadTime: 1.0000e-04
      ringdownTime: 6.0000e-05
             delay: 1.0000e-04
gx =
           type: 'grad'
        channel: 'x'
       waveform: [166.5088 1.4497e+03 3.7632e+03 6.6387e+03 9.4617e+03 1.1564e+04 1.2329e+04 1.
          delay: 2.0000e-04
           area: -1.3608
             tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.5000e-05
      shape_dur: 2.0000e-04
          first: 0
           last: 0
```

RF event: Example 1

sinc pulse

```
[rf, gz] = mr.makeSincPulse(20/180*pi, 'Duration', 2e-3, 'SliceThickness', 5e-3, ...
    'apodization', 0.42, 'timeBwProduct', 4, 'system', sys);
```

```
rf =
              type: 'rf'
            signal: [-0.0089 -0.02
                 t: [5.0000e-07 1.
         shape_dur: 0.0020
        freqOffset: 0
       phaseOffset: 0
          deadTime: 1.0000e-04
      ringdownTime: 6.0000e-05
             delay: 1.0000e-04
gx =
              type: 'grad'
           channel: 'x'
          waveform: [166.5088 1.4497e
             delay: 2.0000e-04
              area: -1.3608
                tt: [5.0000e-06 1.500
         shape_dur: 2.0000e-04
             first: 0
              last: 0
```



RF dead time, ringdown time

- Some time is required to turn on RF amplifier
  - Varies by vendor (of order $\sim$100 μs)
- Some time is also required to turn off the RF amplifier
  - Varies by vendor (of order tens of μs)
- Pulseq toolbox:
  - Requires that rf.delay $\geq$ sys.rfDeadTime. makeSincPulse() will insert this delay if needed.
  - Requires a gap $\geq$ sys.rfRingdownTime between end of RF waveform and end of block. makeSincPulse() will extend the block duration if needed.
- Depending on the vendor, it may not always be necessary to specify non-zero dead/ringdown times (more on that later)

RF event: Example 2

SMS pulse (6 simultaneous slices)

```
load arbrf.mat;    % rfwav (T) and gzwav (mT/m)

flip = pi;   % dummy value, will rescale anyhow
rf = mr.makeArbitraryRf(rfwav, flip, 'delay', sys.rfDeadTime, 'system', sys);
rf.signal = rf.signal/max(abs(rf.signal))*max(abs(rfwav)*sys.gamma); % Hz

gzwav_Hzm = gzwav*sys.gamma/1e3;
gz = mr.makeArbitraryGrad('z', gzwav_Hzm, sys, 'delay', 10e-6);
```

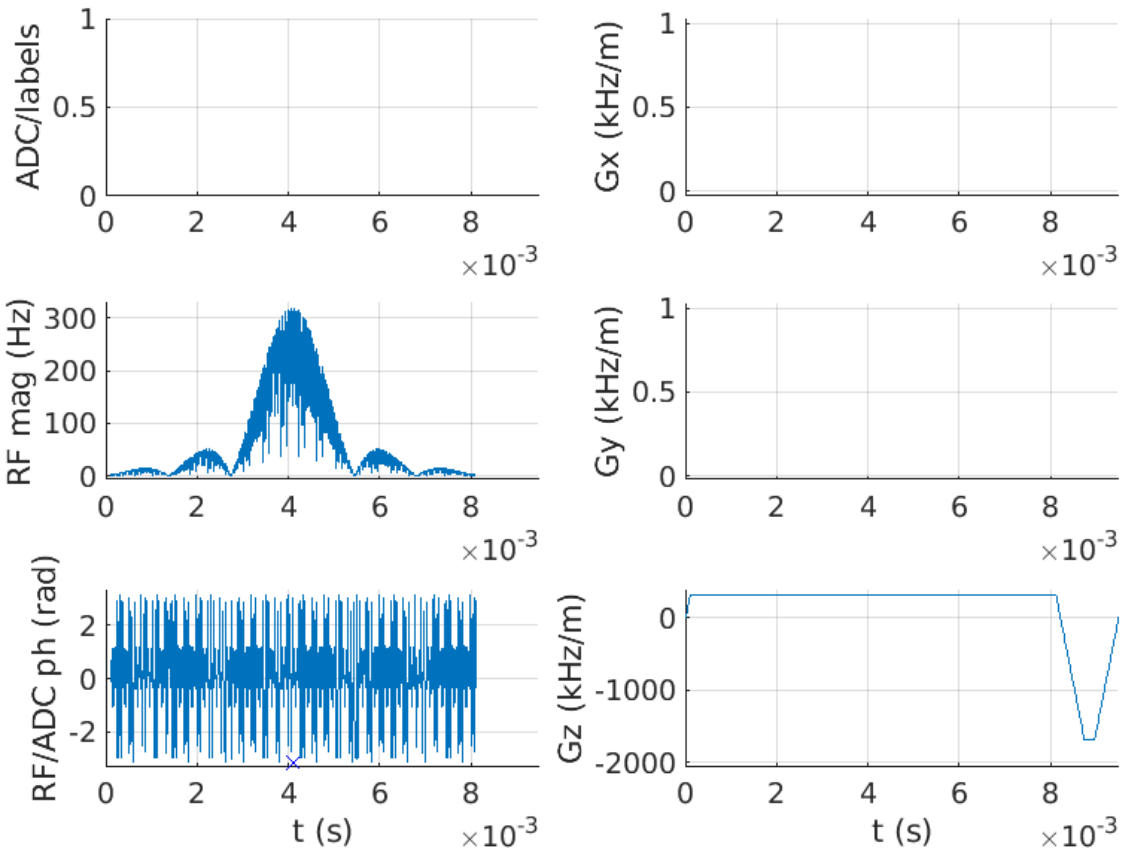
```
rf =
            type: 'rf'
          signal: [0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000 + 0.0000i 0.0000
               t: [5.0000e-07 1.5000e-06 2.5000e-06 3.5000e-06 4.5000e-06 5.5000e-06 6.5000e-
       shape_dur: 0.0080
      freqOffset: 0
     phaseOffset: 0
        deadTime: 1.0000e-04
    ringdownTime: 6.0000e-05
           delay: 1.0000e-04
gx =
          type: 'grad'
       channel: 'x'
      waveform: [166.5088 1.4497e+03 3.7632e+03 6.6387e+03 9.4617e+03 1.1564e+04 1.2329e+04 1.
         delay: 2.0000e-04
          area: -1.3608
            tt: [5.0000e-06 1.5000e-05 2.5000e-05 3.5000e-05 4.5000e-05 5.5000e-05 6.5000e-05
    shape_dur: 2.0000e-04
         first: 0
          last: 0
```

RF event: Example 2

SMS pulse (6 simultaneous slices)

```
load arbrf.mat;     % rfwav (T) and gzwav (mT/m)

flip = pi;    % dummy value, will rescale anyhow
rf = mr.makeArbitraryRf(rfwav, flip, 'delay', sys.rfDeadTime, 'system', sys);
rf.signal = rf.signal/max(abs(rf

gzwav_Hzm = gzwav*sys.gamma/1e3;
gz = mr.makeArbitraryGrad('z', g
```

```
rf =
              type: 'rf'
            signal: [0.0000 + 0.00
                 t: [5.0000e-07 1.
         shape_dur: 0.0080
        freqOffset: 0
       phaseOffset: 0
          deadTime: 1.0000e-04
      ringdownTime: 6.0000e-05
             delay: 1.0000e-04
gx =
              type: 'grad'
           channel: 'x'
          waveform: [166.5088 1.4497e
             delay: 2.0000e-04
              area: -1.3608
                tt: [5.0000e-06 1.500
         shape_dur: 2.0000e-04
             first: 0
              last: 0
```



ADC events

- Samples assumed to occur at centers of dwell time periods
- The delay defines the timing of the starting edge of the first period

```
Nx = 256;
dwell = 4e-6;   % sec
adc = mr.makeAdc(Nx, 'Duration', Nx*dwell, 'Delay', 0.2e-3, 'system', sys);
```

```
adc =
           type: 'adc'
     numSamples: 256
          dwell: 4.0000e-06
          delay: 2.0000e-04
     freqOffset: 0
    phaseOffset: 0
       deadTime: 4.0000e-05
```

ADC dead time

- Some time is required to turn on data acquisition board
  - Varies by vendor (typically a few tens of μs)
- Pulseq toolbox:
  - Requires that adc.delay $\geq$ sys.adcDeadTime. makeAdc() will insert this delay if needed.
  - Requires a gap $\geq$ sys.adcDeadTime between end of ADC window and end of block. makeAdc() will extend the block duration if needed.
- Depending on the vendor, it may not always be necessary to specify a non-zero dead time before/after the ADC window

Delay events

- Not real objects: their only function is to (possibly) extend the block duration
- Not stored in the Pulseq file

```
del = mr.makeDelay(2e-3);
```

```
delay =
      type: 'delay'
     delay: 0.0020
```

Get event duration: mr.calcDuration()

- Example:

```
rf =
            type: 'rf'
          signal: [-0.0089 -0.0268 -0.0448 -0.0628 -0.0808 -0.0988 -0.1169 -0.1350 -0.1
               t: [5.0000e-07 1.5000e-06 2.5000e-06 3.5000e-06 4.5000e-06 5.5000e-06 6.
       shape_dur: 0.0020
      freqOffset: 0
    phaseOffset: 0
       deadTime: 1.0000e-04
   ringdownTime: 6.0000e-05
          delay: 1.0000e-04

>> mr.calcDuration(rf)*1e3

ans =

    2.1600
```

- Common use: calculate delays for desired TE, TR, etc

Creating blocks

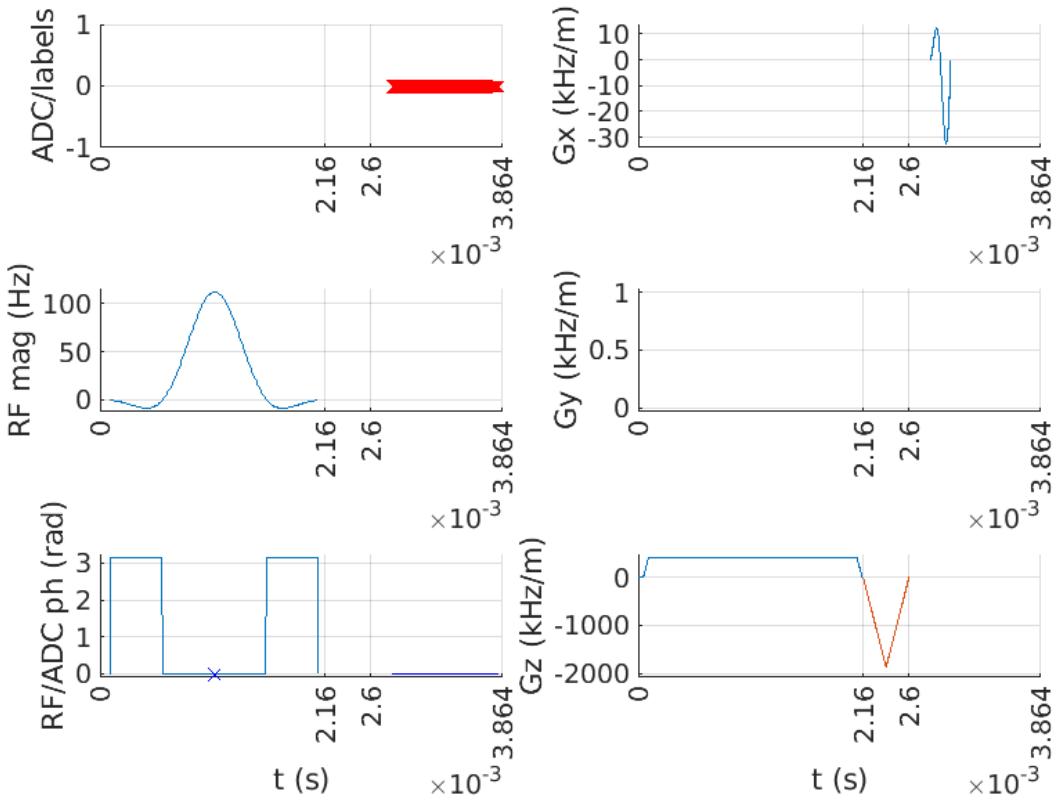
```
seq = mr.Sequence();

% slice-select pulse
sincrf;                                          % create rf and gz
seq.addBlock(rf, gz, mr.makeDelay(2e-3));        % what is effect of the delay event here??

% slice rephaser gradient
gzrep = mr.makeTrapezoid('z', 'Area', -gz.area/2, 'system', sys);
seq.addBlock(gzrep);

% spiral readout
spiral;                         % create gx
seq.addBlock(gx, adc);

close all; seq.plot('showBlocks', true);
```

```
seq.getBlock(1) =
    blockDuration: 0.0022
               rf: [1x1 struct]
               gx: []
               gy: []
               gz: [1x1 struct]
              adc: []
```

Creating blocks

```
seq = mr.Sequence();

% slice-select pulse
sincrf;                                                    % create rf and gz
seq.addBlock(rf, gz, mr.makeDelay(2e-3));                  % what is effect of the delay event here??

% slice rephaser gradient
gzrep = mr.makeTrapezoid('z', 'Are
seq.addBlock(gzrep);

% spiral readout
spiral;                                % crea
seq.addBlock(gx, adc);

close all; seq.plot('showBlocks',
```

```
seq.getBlock(1) =
    blockDuration: 0.0022
               rf: [1x1 struct]
               gx: []
               gy: []
               gz: [1x1 struct]
              adc: []
```



Creating blocks

Rules:

- Time is referenced to start of block
- Block duration is determined by the longest event
- Only one event on each channel (per block)

To pass seq.checkTiming(), additional rules apply:

- Minimum RF delay = sys.rfDeadTime
- Minimum RF ringdown time before end of block = sys.rfRingdownTime
- Minimum ADC delay = sys.adcDeadTime
- Minimum time after ADC event before end of block = sys.adcDeadTime
- Many more constraints are verified/checked by checkTiming(), which will be covered in the 'Sequence Analysis' talk.