Sequence analysis tools

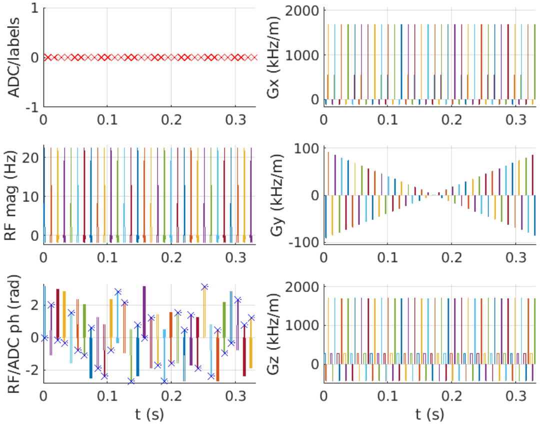
Jon-Fredrik Nielsen

November 15, 2023

Outline

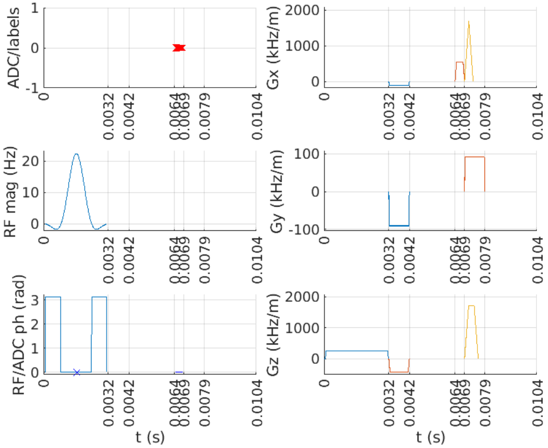
Example sequence: 2D RF-spoiled gradient echo

```
>> write2DGRE__;        % creates 'seq' object
>> seq.plot('blockRange', [1 6], 'showBlocks', true);
```



Example sequence: 2D RF-spoiled gradient echo

```
>> write2DGRE__;         % creates 'seq' object
>> seq.plot('blockRange', [1 6], 'showBlocks', true);
```



Code excerpt

```
% write2DGRE__short.m

rf_phase = 0;
rf_inc = 0;
rfSpoilingInc = 117;        % RF spoiling increment (degrees)

for iY = 1:Ny
    % RF spoiling
    rf.phaseOffset = rf_phase/180*pi;
    adc.phaseOffset = rf_phase/180*pi;
    rf_inc = mod(rf_inc+rfSpoilingInc, 360.0);
    rf_phase = mod(rf_phase+rf_inc, 360.0);

    % Excitation block
    seq.addBlock(rf, gz);

    % Slice-select refocus and readout prephasing
    pesc = peScales(iY,1);  % phase-encode gradient scaling (pre-computed outside loop)
    seq.addBlock(gxPre, mr.scaleGrad(gyPre, pesc), gzReph);

    % delay to achieve desired TE
    seq.addBlock(mr.makeDelay(delayTE));

    % ADC
    seq.addBlock(gx, adc);

    % Spoil and PE rephasing, and TR delay
    seq.addBlock(gxSpoil, mr.scaleGrad(gyPre, -pesc), gzSpoil);
    seq.addBlock(mr.makeDelay(delayTR));
end
```

seq.checkTiming()

- Checks timing of all blocks and objects in the sequence
- Optionally returns the detailed error log as cell array of strings
- Usage:

```
[ok, error_report] = seq.checkTiming;
if (ok)
    fprintf('Timing check passed successfully\n');
else
    fprintf('Timing check failed! Error listing follows:\n');
    fprintf([error_report{:}]);
    fprintf('\n');
end
```

seq.testReport()

- Optional slow step, useful for testing during development
- Detects TE, TR, flip angle, gradient peak amp/slew, ...
- Usage:

```
>> rep = seq.testReport;
>> fprintf([rep{:}]);
```

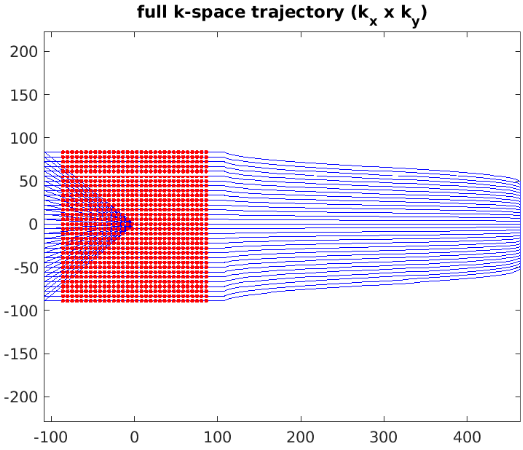
seq.testReport()

```
Number of blocks: 192
Number of events:
   RF:       32
   Gx:       96
   Gy:       64
   Gz:       96
   ADC:      32
Sequence duration: 0.333120s
TE: 0.005020s
TR: 0.010410s
Flip angle: 6.00 degrees
Unique k—space positions (a.k.a. columns, rows, etc): 32
Unique k—space positions (a.k.a. columns, rows, etc): 32
Dimensions: 2
   Spatial resolution: 5.81 mm
   Spatial resolution: 5.62 mm
Repetitions/slices/contrasts: 1  range: [1 1]
   1024 k—space position(s) repeated 1 times
Cartesian encoding trajectory detected
Block timing check passed successfully
Max. Gradient: 1693122 Hz/m == 39.77 mT/m
Max. Gradient: 90703 Hz/m == 2.13 mT/m
Max. Gradient: 1702128 Hz/m == 39.98 mT/m
Max. Slew Rate: 8.06248e+09 Hz/m/s == 189.37 T/m/s
Max. Slew Rate: 4.53515e+09 Hz/m/s == 106.52 T/m/s
Max. Slew Rate: 8.51064e+09 Hz/m/s == 199.89 T/m/s
Max. Absolute Gradient: 2402525 Hz/m == 56.43 mT/m
Max. Absolute Slew Rate: 1.25699e+10 Hz/m/s == 295.23 T/m/s
```

k-space analysis: seq.calculateKspacePP()

Usage:

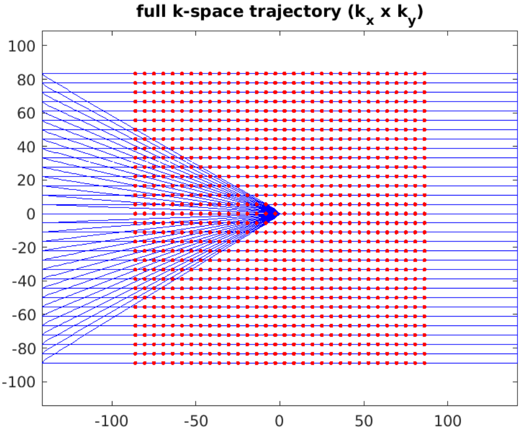
```
[ktraj_adc,t_adc,ktraj,t_ktraj,t_excitation,t_refocusing] = seq.calculateKspacePP();
figure; plot(ktraj(1,:),ktraj(2,:),'b'); % a 2D k-space plot
axis('equal'); % enforce aspect ratio for the correct trajectory display
hold;plot(ktraj_adc(1,:),ktraj_adc(2,:),'r.'); % plot the sampling points
title('full k-space trajectory (k_x x k_y)');
```



k-space analysis: seq.calculateKspacePP()

Usage:

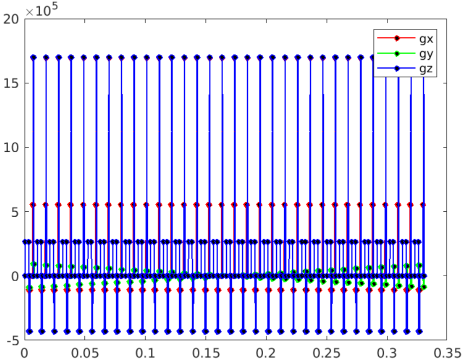
```
[ktraj_adc,t_adc,ktraj,t_ktraj,t_excitation,t_refocusing] = seq.calculateKspacePP();
figure; plot(ktraj(1,:),ktraj(2,:),'b'); % a 2D k-space plot
axis('equal'); % enforce aspect ratio for the correct trajectory display
hold;plot(ktraj_adc(1,:),ktraj_adc(2,:),'r.'); % plot the sampling points
title('full k-space trajectory (k_x x k_y)');
```



Waveform shape analysis

- Use seq.waveforms_and_times() to decompress waveforms
- Example usage:

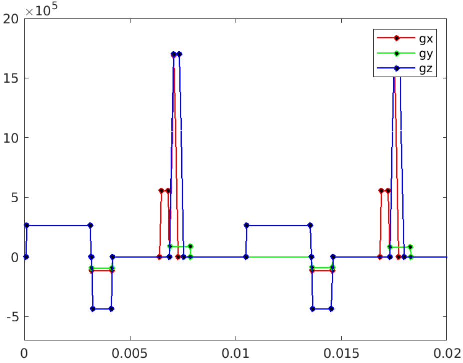
```
w = seq.waveforms_and_times();
gx.tt= w{1}(1,:); gx.waveform = w{1}(2,:);
gy.tt= w{2}(1,:); gy.waveform = w{2}(2,:);
gz.tt= w{3}(1,:); gz.waveform = w{3}(2,:);
```



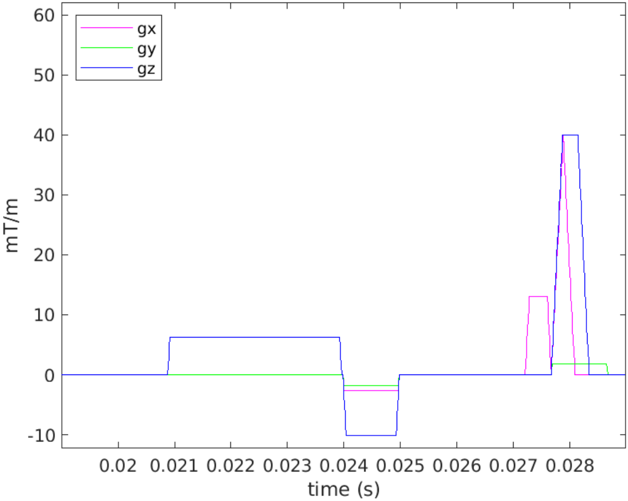
Waveform shape analysis

- Use seq.waveforms_and_times() to decompress waveforms
- Example usage:

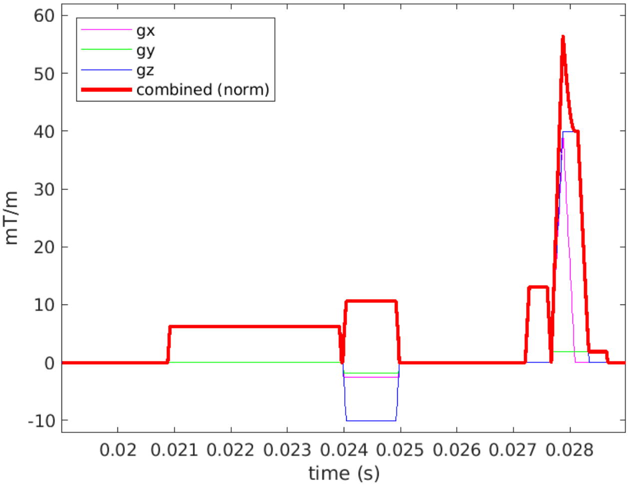
```
w = seq.waveforms_and_times();
gx.tt= w{1}(1,:);  gx.waveform  = w{1}(2,:);
gy.tt= w{2}(1,:);  gy.waveform  = w{2}(2,:);
gz.tt= w{3}(1,:);  gz.waveform  = w{3}(2,:);
```



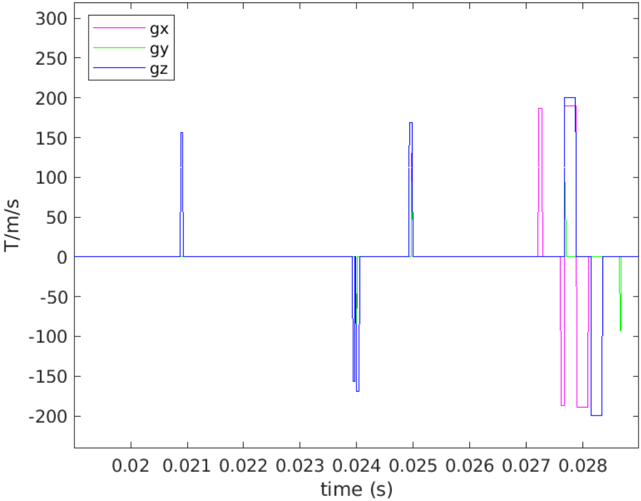
Waveform shape analysis: combined gradient (norm)



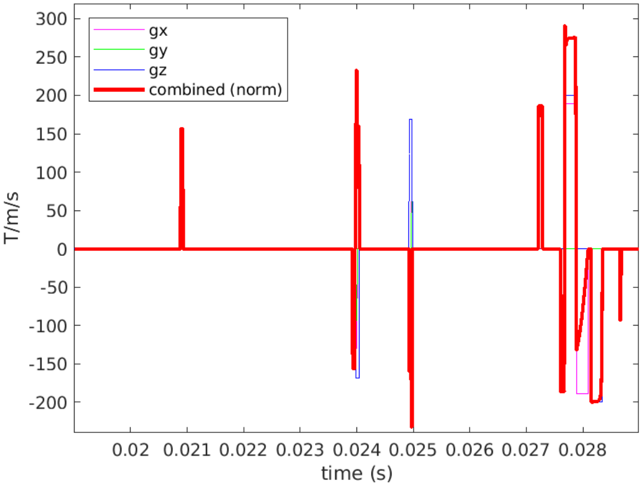
Waveform shape analysis: combined gradient (norm)



Waveform shape analysis: combined slew rate (norm)



Waveform shape analysis: combined slew rate (norm)



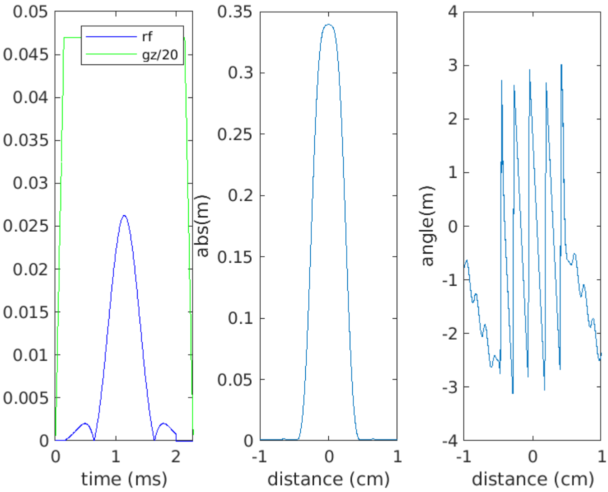
Simulate slice profile

- Slice profile impacts SNR, partial volume effects, slice cross-talk
- Fortunately, easy to simulate. Example:

```
% rfsim__.m

% Design RF and z gradient events
[rfe, gze] = mr.makeSincPulse(20/180*pi, 'Duration', 2e-3, 'SliceThickness', 5e-3, ...
    'apodization', 0.42, 'timeBwProduct', 4, 'system', sys);

% Interpolate waveforms to a common raster period
dt = 4e-6;    % sec
rf = interp1([0 rfe.delay+rfe.t], [0 rfe.signal], dt/2:dt:rfe.t(end));
gz = interp1([0 gze.riseTime gze.riseTime+gze.flatTime ...
              gze.riseTime+gze.flatTime+gze.fallTime], ...
              gze.amplitude*[0 1 1 0], dt/2:dt:mr.calcDuration(gze));

% pad to same length and simulate
l = length(gz); rf = [rf zeros(1,l-length(rf))];
rf = rf/sys.gamma*1e4;                  % Gauss
gz = gz/sys.gamma*1e2;                  % Gauss/cm
m0 = [0 0 1];                           % initial magnetization along mz
Z = linspace(-1,1,100);                 % cm
T1 = 1000; T2 = 100;                    % ms
toppe.utils.rf.slicesim(m0, rf, gz, dt*1e3, Z, T1, T2);
```

Simulate slice profile

- Sli lk
- Fo




```
% rfsim_
% Design
[rfe , gze                                           3, ...
    'apodi
% Interpo
dt = 4e-6
rf = inte
gz = inte

% pad to
l = lengt
rf = rf/s
gz = gz/s
m0 = [0 0
Z = linsp
T1 = 1000
toppe.uti
```

PNS modeling (GE)

- Nerve impulse response model [1]
- Example:

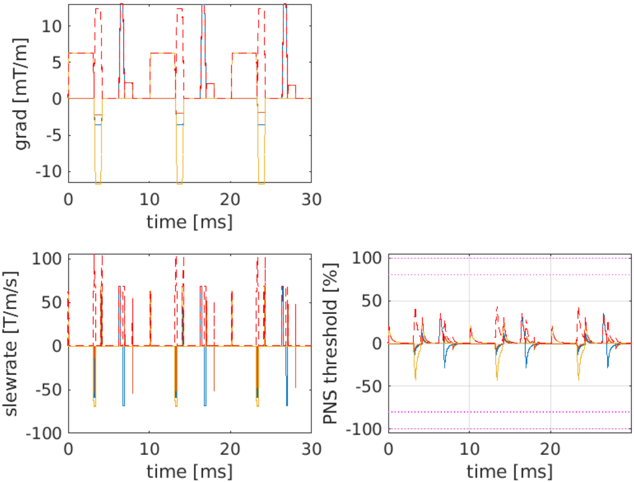
```
sysGE = toppe.systemspecs('maxGrad', 5, 'maxSlew', 20);

% Convert .seq file to GE scan files
seq2ge('gre2d.seq', sysGE, 'gre2d.tar');

% Untar scan files to current working directory
system('tar xf gre2d.tar');

% Simulate and plot PNS waveform
p = toppe.calcsequencepns(sysGE, 'timeRange', [0 0.03]);
```

[1] Schulte RF, Noeske R. Peripheral nerve stimulation-optimal gradient waveform design. Magnetic resonance in medicine. 2015 Aug;74(2):518-22

PNS modeling (GE)

- Ne
- Ex

```
sys(

% C
seq!

% U
syst

% S
p =
```



[1]Schulte RF, Noeske R. Peripheral nerve stimulation-optimal gradient waveform design. Magnetic resonance in medicine. 2015 Aug;74(2):518-22