

MATLAB Tools for RF Pulse Design

- Simulator
 - * Calculation of α and β – abr
 - * Slice Profiles – ab2ex, ab2se, ab2inv, ab2sat
 - * Spatial scale – gt2cm
- Basic 1D RF pulse design – dzrf
 - * Designs small-tip, 90's, 180's, sat pulses
 - * Min, max, and linear phase
 - * Many options, most with defaults
- 2D pulse design
 - * Spiral 2D pulses – dz2d
 - * Echo-planar spin-echo pulses – dzepse
- Utilities
 - * Scale RF to kHz – rfscale
 - * Verse – verse
 - * Conversion to signal waveforms – signal
 - * Reading Signal 5X waveforms – loadwave
- Location
 - * Files are in

lad : /user/local/mrsrl/matlab/rf_tools

This should be in your matlab path

RF Simulator

- Basic rf simulator – abr
 - * Assumes no T_1 or T_2 decay
 - * Computes all slice profiles at once
- Simplest invocation:

```
>> [a b] = abr(rf,x);
```

- * rf scaled so that

$$\text{sum}(\text{rf}) = \text{flip angle in radians}$$

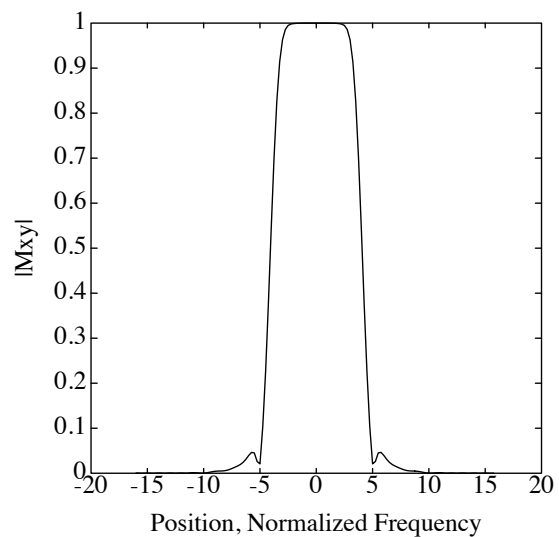
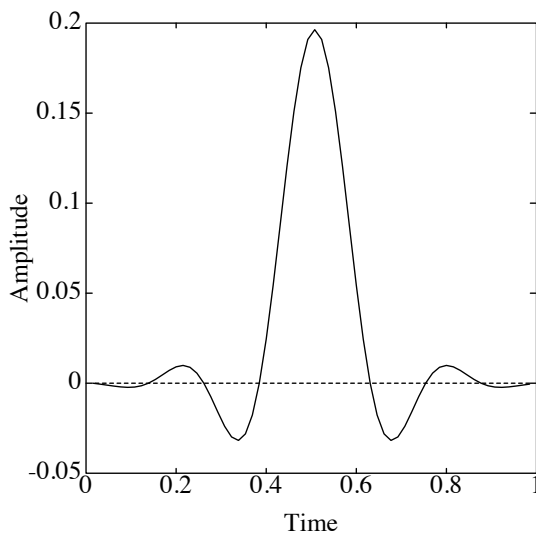
- * x scaled in normalized frequency
- * Constant gradient of area $2 * \pi$

- Example:

```
>> rf = (pi/2) * msinc(64,2);
```

```
>> [a b] = abr(rf,x);
```

```
>> cplot(2 * conj(a). * b);
```



Simulator – Outputs

- Output can be either α and β vectors, or a matrix containing each as a column
- Vectors are more convenient if
 - * You are really interested in α and β . For most people this will be almost never.
 - * Example:

```
>> [a b] = abr(rf, x);  
>> cplot(b)
```
- Matrix is more convenient if
 - * You are really interested in a slice profile. This is the usual case.
 - * Example:

```
>> ab = abr(rf, x);  
>> cplot(ab2se(ab))
```

Slice Profiles

- Several routines convert α and β to slice profiles.
 - * Inputs can be either vectors or a matrix
 - * Output is a complex vector or matrix
- Available routines:
 - * ab2ex – excitation profile $2\alpha^*\beta$ from initial M_z
 - * ab2se – spin-echo profile $i\beta^2$ from initial M_y
 - * ab2inv – inversion profile $1 - 2|\beta|^2$ from initial M_z
 - * ab2sat – saturation pulse profile, same as ab2inv
- Examples:
 - >> cplot(ab2ex(abr(rf,x)))
 - >> cplot(ab2inv(a,b))

Simulator Options

- A time varying gradient g can be specified. This should have a k -space span of 2π for consistency with other routines.

- * Example: gv is a time-varying gradient, and rfv is a VERSE'd rf pulse

- ```
>> mxy = ab2ex(abr(rfv,gv,x))
```

- A two dimensional gradient may be specified, simulated either over one dimension, or over a two dimensional surface.

- \* The gradient is specified as the complex vector

- \* Example:

- ```
>> g = gx + i * gy;
```

- ```
>> x = [-16 : 15]/4;
```

- ```
>> y = [-16 : 15]/2;
```

- ```
>> ab = abr(rf,g,x,y);
```

- ```
>> mxy = ab2ex(ab);
```

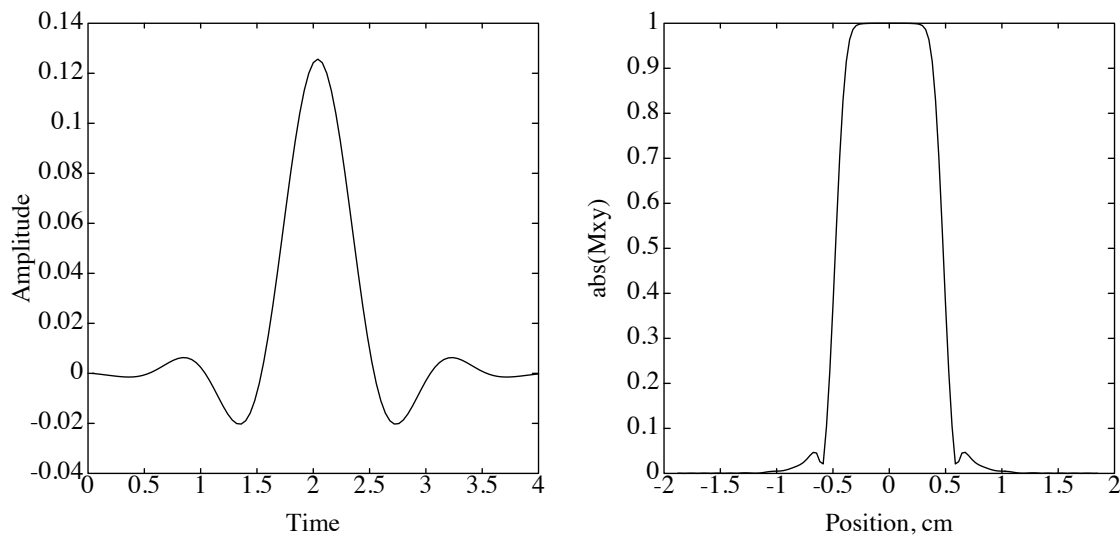
- ```
>> mesh(abs(mxy))
```

# Spatial and Frequency Scale

---

- Conversion of normalized frequency to space – gt2cm
- Inputs:
  - \* Vector of normalized frequencies
  - \* Gradient strength in G/cm
  - \* Pulse duration in ms
- Example: an rf pulse played with a 0.5 G/cm gradient for 4 ms.

```
>> plot(gt2cm(x,0.5,4),abs(ab2ex(abr(rf,x))))
```



- Also, conversion of normalized frequency to kHz – t2hz
  - \* To plot the previous example in kHz instead of cm,  

```
>> plot(t2hz(x,4),abs(ab2ex(abr(rf,x))))
```

# RF Pulse Design

---

- RF pulse design front end is `dzrf`
- Inputs are the number of points and the time-bandwidth of the pulse. The pulse type is optional.

```
>> rf = dzrf(160,8,'se');
```

- Types of pulses are:

'st' – Small tip excitation pulse (default, returns filter)

'ex' –  $\pi/2$  excitation pulse

'inv' – Inversion pulse

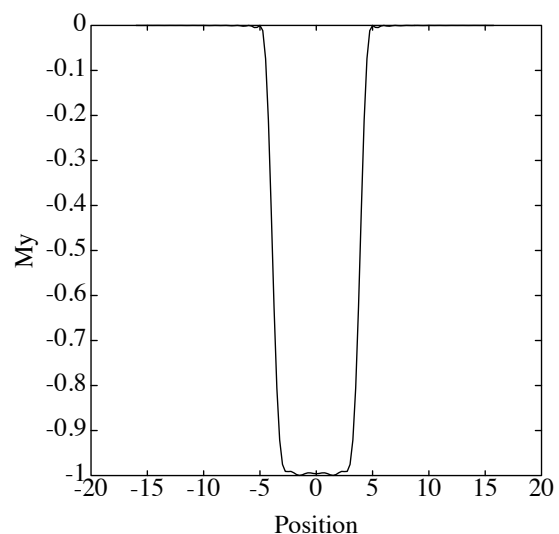
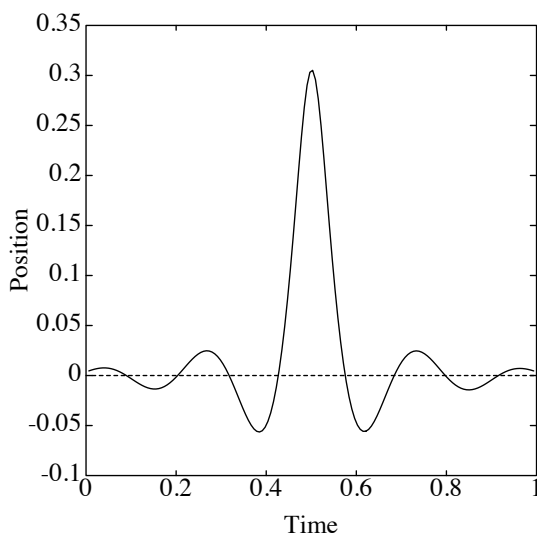
'se' – Spin-echo pulse

'sat' –  $\pi/2$  saturation pulse

- Default assumptions:

\* In-slice and out-of-slice ripples are 1%

\* The underlying filter is designed using least-squares



# RF Pulse Design – Filter Types

---

- The method used to design the underlying digital filter can be specified.

\* Example: to use an msinc to design a saturation pulse,

```
>> rf = dzrf(160,8,'sat','ms');
```

- Filter options are

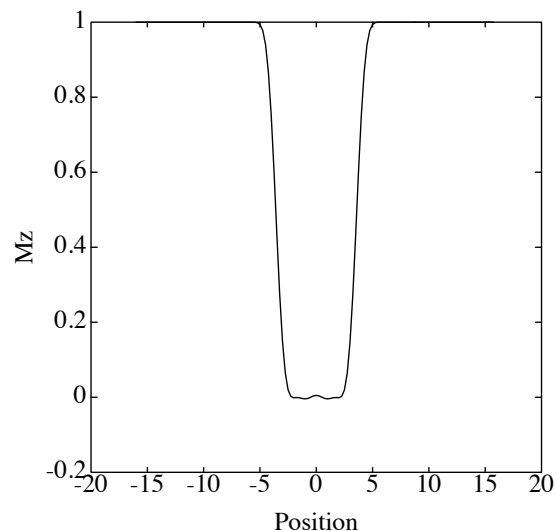
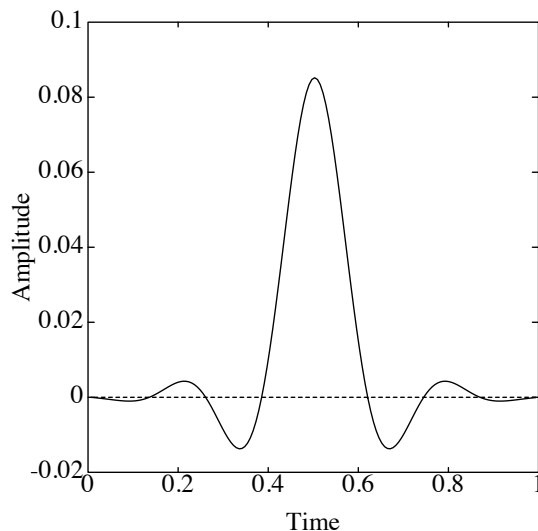
'ms' – Msinc, Hamming windowed sinc

'pm' – Parks-McClellan equal-ripple (can be slow)

'ls' – Least-squares (default, recommended)

'min' – Minimum-phase using factored pm filter

'max' – Maximum-phase using factored pm filter



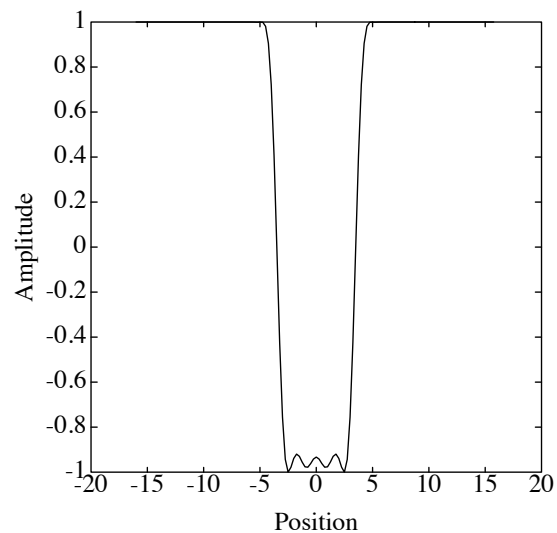
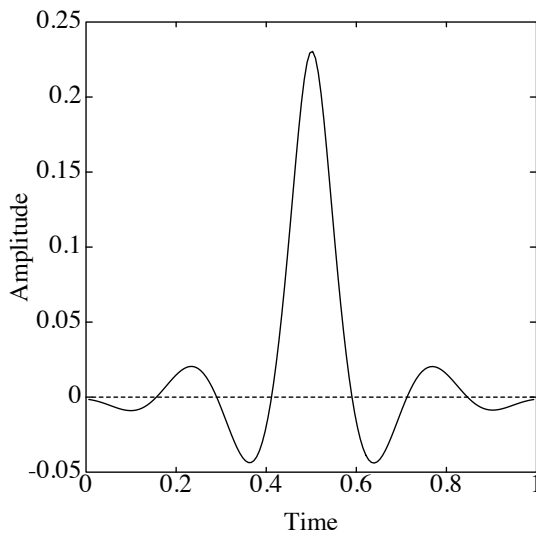


# RF Pulse Design – Ripple Specification

---

- In-slice and out-of-slice ripple can be specified. Transition widths are automatically calculated.
- Default in-slice and out-of-slice ripples are 1%.
- Example: to design a least squares inversion with 0.1% out-of-slice, and 5% in-slice,

```
>> rf = dzrf(160,8,'inv','ls',0.05,0.001);
>> plot(x,ab2inv(abr(rf,x)));
```

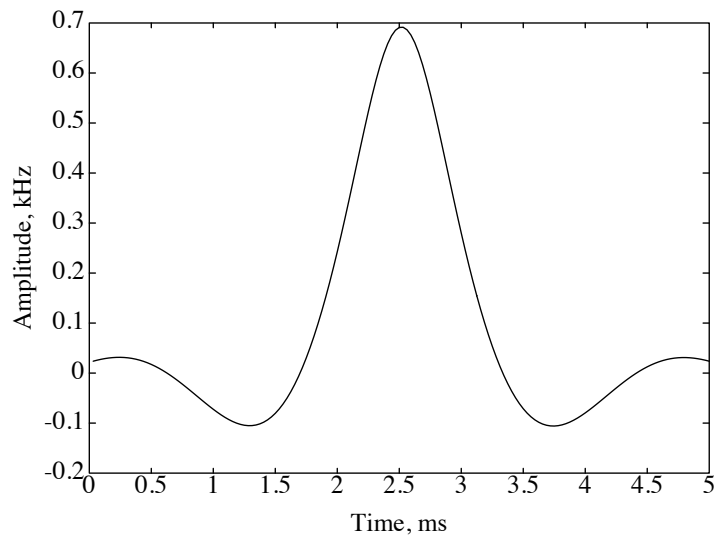


# RF Scale

---

- RF amplitude on signa is limited
  - \* Phantoms – about 1 kHz
  - \* Humans – about 700 Hz
- RF amplitude for a given duration – rfscale
- Inputs:
  - \* RF pulse, area normalized to flip angle in radians
  - \* Pulse duration in ms
- Example:

```
>> rf = dzrf(160,4,'se');
>> t = [1 : 160]/160;
>> plot(t * 5,rfscale(rf,5));
```

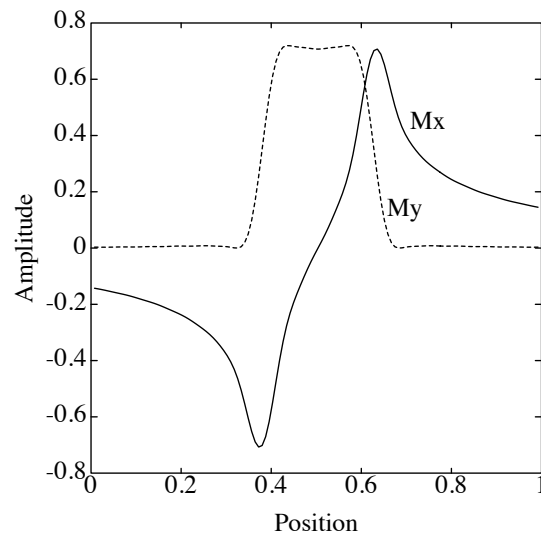
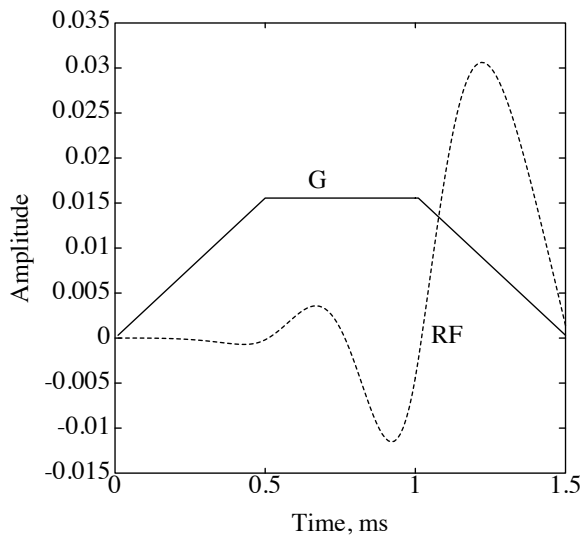


# VERSE

---

- Verse an RF pulse – verse
- Inputs:
  - \* RF pulse
  - \* Time-varying gradient (always positive)
- Example:

```
>> gv = [[1 : 50]/50 ones(1,50) [50 : -1 : 1]/50]
>> rf = dzrf(200,8,'st','ms');
>> rfv = verse(gv,rf(1 : 100));
>> plot(t,gv,t,rfv);
>> cplot(ab2ex(abr((rf,x))));
```



# Signa Waveform File Conversion

---

- Save a signa waveform file – signa
  - \* Rescales to full scale, unless specified
  - \* Masks off low bit
  - \* Creates a 4.X waveform file
  - \* Use xlatebin to translate to 5.X
- Inputs:
  - \* RF pulse
  - \* File base name, .wav is automatically appended
  - \* An additional .r and .i are appended for complex waveforms
- Example: this creates the signa file new\_se.wav

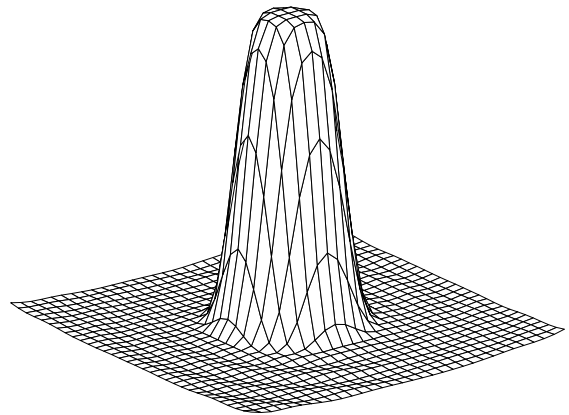
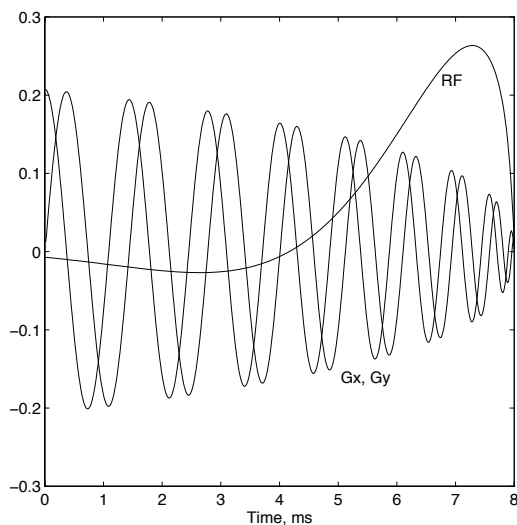
```
>> rf = dzrf(160,4,'se');
>> signa(rf,'new_se');
```

## 2D Spiral Pulses

---

- Design spiral gradient and rf waveform – dz2d
  - \* Inward slew and amplitude limited spiral
  - \* Windowed jinc k-space weighting
- Example: design an 8 ms 8 turn spiral and simulate it

```
>> [rf g] = dz2d(8, 1, 4, 512, 1, 2);
>> xs = [-16 : 15]/2;
>> mesh(abs(ab2ex(abr(rf * (pi/2), g, xs, xs))));
```



# Echo-Planar Spin-Echo Pulses

---

- Spectral-spatial and EP pulses – dzepse
  - \* Full 2D inverse SLR design
- Example: a 20 ms spectral-spatial spin-echo pulse using a trapezoid waveform g1

```
>> g1 = [[0.5 : 15.5]/16ones(1,16)[15.5 : -1 :
0.5]/16];
```

```
>> rf = dzepse(pi,g1,4,1.5,13,0.15);
```

```
>> g = [g1 - g1 . . . g1] * 2 * pi/sum(g1);
```

```
>> om = ones(1,length(rf)) * 2 * pi/length(rf);
```

```
>> mesh(-imag(ab2se(abr(rf,g + i * om,xs,xs))));
```

