MRI Simulations with JEMRIS:
A Quick Guide for Hands-On Training

draft version, September 2007, T. Stöcker

1. Introduction

The Jülich environment for MRI simulations (JEMRIS) provides functionality for the design and the experimental numerical testing of MRI sequences. The framework utilises high performance computing on parallel processor architectures, since MR simulations of realistically sized spin ensembles are time consuming.

Mode of action: The ***Sequence*** returns values of all 5 pulse-axes (RF magnitude and phase, and 3 gradient axes) at any desired time point to the ***Model***, which numerically solves the Bloch equation individually for each spin. The spins and their physical properties (position and M_0,T_1,T_2) are defined by the ***Sample***. The ***Model*** sums up the net magnetization (M_x,M_y, M_z) and stores the resulting ***Signal*** at every desired time point of the sequence.

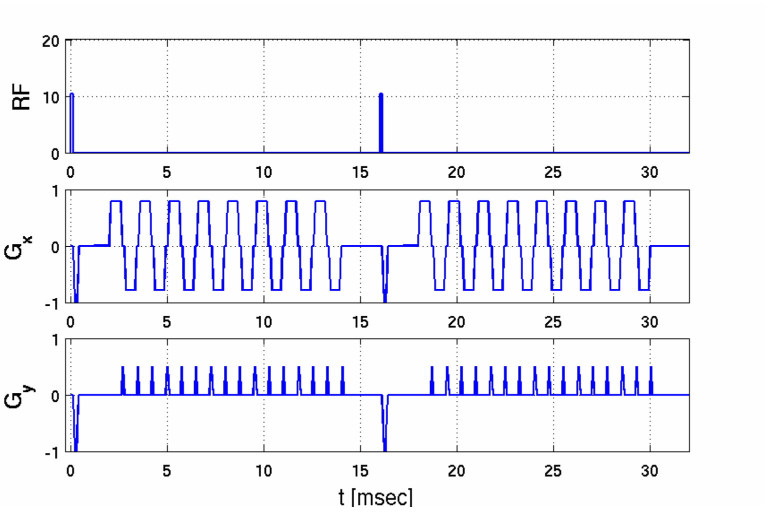
3D simulations are very time consuming, however, for many purposes 2D simulation are sufficient. Here, sample properties only depend on two spatial coordinates and, thus, the sequence only needs to encode the spatial information along these two directions with gradients. The rest of this document only treats the case of 2D simulations. Therefore, no slice selection gradients will be considered. The excitation is always performed with non-selective RF pulses, i.e. the complete 2D sample ("the slice") will be excited. Then, x- and y-gradients encode the MR image exactly in the same way as it is the case in real MRI experiments after slice selective excitation.

The MRI ***Sequence***: In the framework, the sequence is given by a set of instructions written in XML. The XML file is automatically parsed into a set of interacting C++ objects. Thus, sequence development is reduced to writing the XML file, which might be seen as a scripting language for MRI sequence development.

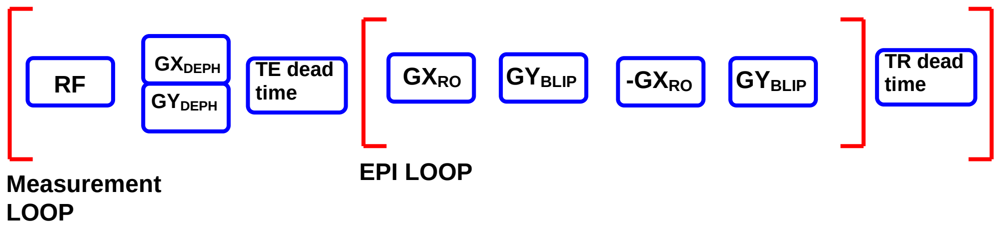
The MRI sequence is sub-divided in two basic parts: loops and modules. They are called *ConcatSequence* and *AtomicSequence*. The first one is a container which holds other *Sequences* (either *Atomic* or *Concat*) in a defined order. This composition of sub-sequences can be repeated, defined by the loop properties of the *ConcatSequence*. The modules, i.e. the *AtomicSequence*, plays out the pulses. It has therefore functionality to set pulses of various types.

The remainder of this document explains how to design pulse sequences (Chapter 2) and then finally how to invoke the MRI simulation (Chapter 3).

On the next page, a simple example is given to understand why and how a tree structure is well suited to represent an MRI sequence. The first figure depicts the pulse diagram of a basic EPI sequence, where the complete measurement is repeated two times:

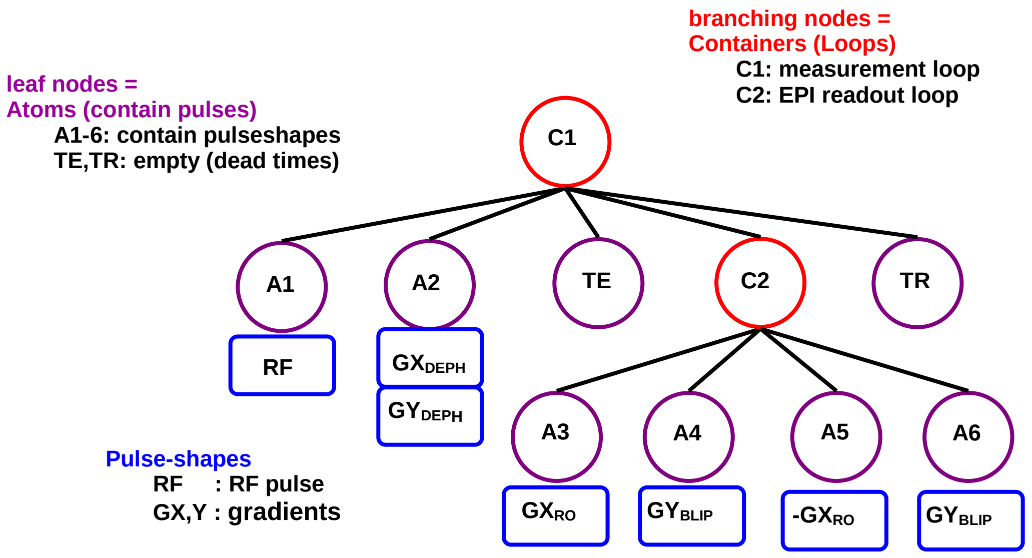


Schematically, this diagram might be sketched in the following way:



Here, the modules for different RF and gradient pulses, as well as the inserted fill times to reach correct echo time and repetition time, are given by blue squares, whereas the repetition of these modules take place in a nested loop structure (red brackets).

Equivalently, this diagram might be drawn as a left-right ordered tree, which is very suitable and efficient for computer-aided design and access of the pulse sequence:

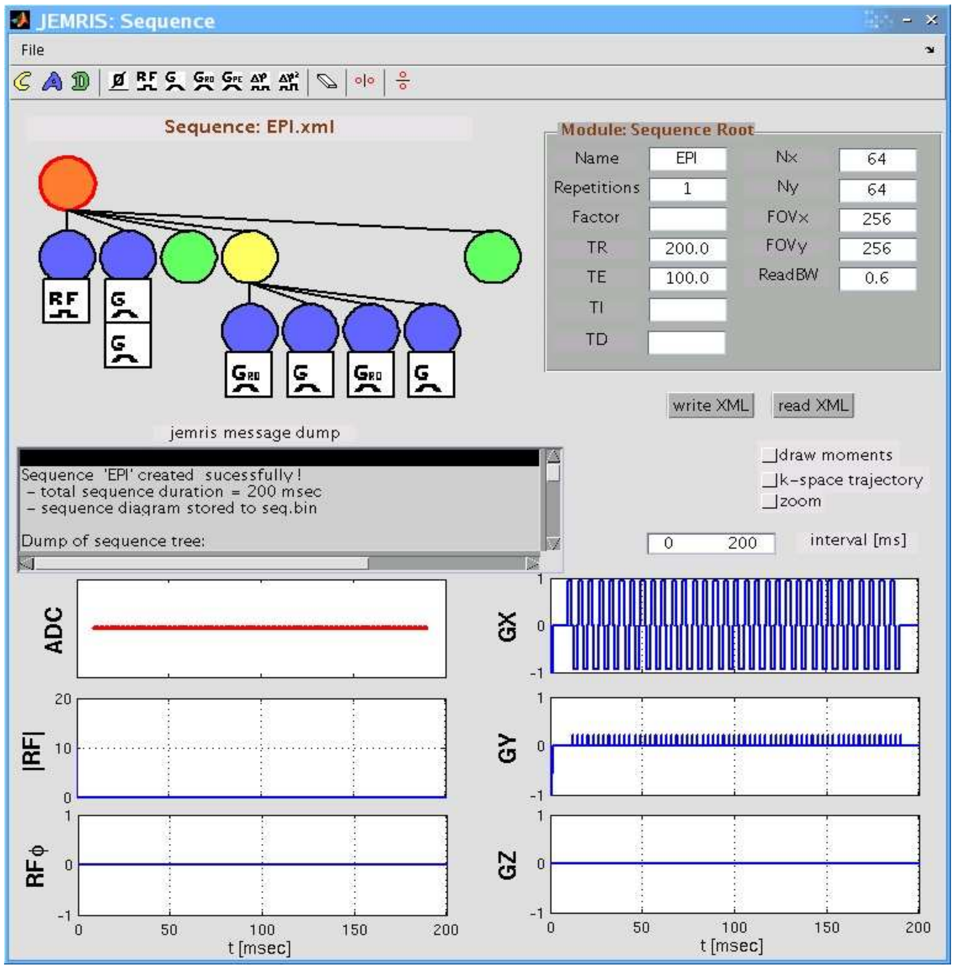


MATLAB GUIs were developed, for design and visualisation of MRI sequences and the corresponding MRI simulations on different MR samples. There are two tools in MATLAB available: one for designing and testing the sequence and plotting the sequence pulse diagram and another one for performing the simulation and viewing the results. These tools automatically call the C++ program on an external HPC cluster. Chapter 2 explains (the more complex) sequence design GUI. Finally, chapter 3 introduces the simulation GUI.

2. Sequence Development

2.1 Introduction to the sequence GUI

The MATLAB command **`JEMRIS_seq3`** launches the GUI for sequence testing. Before running the GUI, one has to move to the same directory in which the sequence XML file is stored. The figure below shows the GUI for the previously shown EPI sequence:



Module: Sequence Root

Name	EPI	Nx	64
Repetitions	1	Ny	64
Factor		FOVx	256
TR	200.0	FOVy	256
TE	100.0	ReadBW	0.6
TI			
TD			

```
Sequence  'EPI' created  sucessfully !
 - total sequence duration  =  200 msec
 - sequence diagram stored to seq.bin

Dump of sequence tree:
```

☐ draw moments
☐ k-space trajectory
☐ zoom

interval [ms]: 0 200

The GUI allows to interactively build the sequence tree. The tools for this are given in the top panel. The **`“File”`** dialog allows to load and save sequences (XML format). The **`“write XML”`** button overwrites the current sequence file according to the changes.

Further, this button executes a run through the sequence and draws the sequence diagram. Possible error output during sequence execution is thrown to the "**message dump**". Instead, the **"read XML"** button discards all changes made to the sequence tree and re-reads the current sequence from its XML file. For viewing the sequence diagram, i.e. the ADC, RF, and gradient events, the GUI has additional features:

- The **"interval [ms]"** dialogue changes the time axis for all pulses, whereas the **"zoom"** button allows to interactively zoom into a specific axis.
- The **"draw moment"** flag draws the gradient moments, i.e. the time integral along the gradient axes, instead of the gradient waveforms. The moment is set to zero after each RF pulse, however, it is inverted for 180° pulses. Thus, it shows the correct k-space position encoding for gradient echo and spin echo sequences.
- A different view of the gradient moments is given by switching the **"k-space trajectory"** flag: it draws the k-space trajectory. Once it is drawn, it is possible to further add the ADC events to this plot by another check box.

The rest of this chapter describes all sequence building modules, which can be inserted to the sequence tree from the top panel.

2.2 The root sequence node and specifying general sequence parameters

The outermost loop – defining the root node of the sequence – is always present in the GUI. It allows to set general purpose sequence parameters, e.g.:

```
TE=30 TR=100 TI=100 TD=20
Nx=64 Ny=64 FOVx=100 FOVy=100 ReadBW=0.1
```

These attributes are used for defining several properties of the imaging sequence such as the echo time, TE, repetition time, TR, inversion time, TI, some arbitrary delay, TD, matrix size, (N_x,N_y), field of view, FOV_x,FOV_y), and the imaging bandwidth, *ReadBW*. If the *Parameter* tag is provided, then its values might be used by the subsequently defined pulses. For the simulations, the units of these values have no meaning, however, one might think of durations as given in milliseconds, field-of-view in millimetre, and the bandwidth in kHz.

The parameter tag makes life easy, since changing of sequence parameters does not require changing values across the complete sequence tree but only at the root node.

2.3 Tree nodes for MRI sequence loops and sequence atoms

The MR sequence tree structure is defined by the nodes *ConcatSequence* and *AtomicSequence*. The latter can not include the former. The *ConcatSequence* represents a loop – thus it has another attribute called *Repetitions*. By default a loop is executed only once, if the setting of the repetition attribute is omitted.

Inside the *AtomicSequence* at least one pulse-shape tag needs to be defined (replacing the dots in the example). Note, that pulse-shapes inside an *AtomicSequence* are played out simultaneously.

2.4 Definition of RF pulses

For this brief introduction, only the Hard RF pulse is discussed. It has the attributes:

```
FlipAngle=90.0 Phase=180.0 Duration=0.1
```

It is a box-car shaped RF pulse (a “hard pulse” or non-selective pulse) which has three attributes for the definition of flip angle, phase, and duration of the pulse.

2.5 Definition of data acquisition

For the MRI simulation, one has to define at which time-points of the sequence the MR signal shall be acquired. For this, every pulse-shape can have the attribute *ADCs* (analogue-digital-converter, in analogy to a “real” MR scanner) which defines how many values of the magnetisation are going to be sampled during the application of this pulse shape. For instance, a pulse with `ADCs=100` acquires 100 data-points of the net magnetisation, (M_x,M_y,M_z), during the pulse. The sample points are always equidistantly distributed across the duration of the pulse.

However, in many imaging sequences the explicit definition of ADCs does not have to be performed. Below a special gradient pulse is introduced – the *RO_TGPS* – which takes care of correctly setting the ADCs from the parameters.

2.6 Definition of delays

The contrast in MRI strongly depends on the dead-times or delays in which the spins are subject to relaxation. Therefore, delays have to be set very accurate. There are two possibilities of inserting delays: i) empty pulse-shapes and ii) the so called *DelayAtom*, which is a specialised *AtomicSequence* for the easy definition of delays.

i) Empty pulse-shapes: The *EmtpyPulseShape*, is waiting for a certain duration:

ii) Inserting delays with a specialised sequence atom: However, mostly not the actual dead-times are of interest but the time between certain events. For instance, the echo time is defined as the duration from the centre of the RF pulse to the centre of the data acquisition. Usually, apart from the dead-times, within this interval several gradients are played out as well. Thus, with the above concept of inserting empty pulse-shapes it is necessary to calculate the durations in beforehand. These calculations are automatically performed by the *DelayAtom* tag.

The *DelayAtom* is a specialised atomic sequence, which means that it can be inserted in the same way into a *ConcatSequence*. Generally, four attributes need to be provided:

```
Delay="TE" StartSeq="S1" StopSeq="S2" DelayType="C2C"
```

Here, the value of `Delay` could be a keyword representing a parameter value (`"TE", "TR", "TI", "TD"`) or a number. The delay duration is counted from a start-sequence to a stop-sequence, both of which have to be defined within the same *ConcatSequence* as the *DelayAtom* itself. The actual delay which has to be inserted is then calculated from all other sequence modules in between the start and the stop sequence. The type of delay between the start and the stop sequence can be either begin-to-end or centre-to-centre, i.e. `DelayType="C2C"` or `"B2E"`. If the definition of the start or the stop sequence is omitted, the *DelayAtom* is itself the start or the stop sequence, respectively.

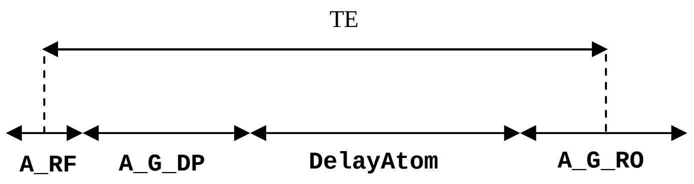
Further, the *DelayAtom* allows setting the `Factor` attribute to reach a multiple of a certain parameter duration. For instance, in spin echo sequences it is needed to match half the echo time between certain events, which is possible with `Delay="TE"` and `Factor="0.5"`.

The following three examples show the most common use of a *DelayAtom*:

a) Simple insertion of 30 ms delay (equivalent to an empty pulse in an *AtomicSequence*):

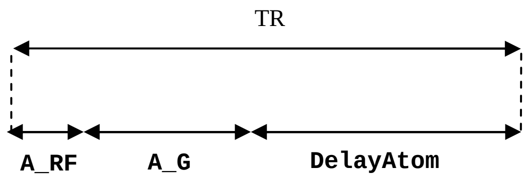
```
Delay="30" DelayType="B2E"
```

b) Echo time delay: Here, the echo time is defined from the centre of **A_RF** to the centre of **A_G_RO**. The needed dead-time, which is automatically inserted by the **DelayAtom**, is therefore TE minus the duration of **A_G_DP** and half the durations of **A_RF** and **A_G_RO**:



```
Delay="TE" StartSeq="A_RF" StopSeq="A_G_RO" DelayType="C2C" />
```

b) Repetition time delay: Here, the echo time is defined from the beginning of **A_RF** to the end of the DelayAtom itself. The actual dead-time which is automatically inserted is therefore TR minus the duration of **A_RF** and **A_G**:

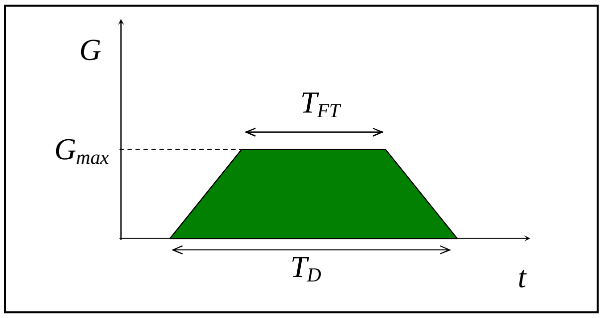


```
Delay="TR" StartSeq="A_RF" DelayType="B2E"
```

Finally, it is also possible to set the *ADCs* attribute inside the *DelayAtom* in order to acquire data in the dead-time.

2.7 Definition of simple gradient pulses

The most basic form of a gradient is a trapezoidal-shaped gradient pulse. It consists of a ramp up, flat top, and a ramp down period:



The maximum possible gradient strength and the slew rate are constants obtained from the hardware settings of the simulation. Other parameters such as the gradient area, the duration and the spatial axis can be set with the attributes.

```
Area=10 Duration=5 Axis=GX
```

where the axis has to be one of the keywords **"GX","GY","GZ"**.

If the duration is omitted, the gradient is applied in the shortest possible time according to the hardware limits – this is the most common way to access it.

There are several predefined keywords for the area, from which the actual area is computed from the parameter settings (FOV and matrix size):

```
Area="KMAXx", "KMAXy", "DKx", "DKy"
```

They correspond to the maximum k-value and the increment in k-space for the x- and y-direction, respectively. Here, k_{max} corresponds to the total size of k-space, i.e. from the minimal to the maximal k-value. Note, that the unit of the k-vector is “radiant per length-unit” (e.g. rad/mm) defining the dephasing across the object, since the gyromagnetic ratio is set to 1 in the simulator.

Further, it is possible to define the **Factor** attribute in order to scale the area, e.g. to achieve a dephasing gradient for half the k-space size:

```
Area=KMAXx Factor=0.5 Axis=GX
```

Another possibility for setting the area is to equalise the gradient area to that of another pulse by setting the name of the other pulse into the area value:

Gradient 1: **Name=TGPS_A Area=5 Axis=GX**
Gradient 2: **Name=TGPS_B Area=TGPS_A Factor=-1.0 Axis=GX**

In this example, the area of the gradient pulse *TGPS_B* is set to the negative value of the area of the gradient pulse *TGPS_A*. For this functionality, the order of appearance of the pulses in the XML file does not matter, i.e. it is possible to refer to the name of pulses which are going to be defined later in the XML file.

Next, there exist predefined methods for important specialised TGPS gradients, i.e. the readout and the phase-encode gradient.

2.6 Definition of the read gradient (frequency encoding gradient)

The read gradient, or read-out (RO) gradient event, is a gradient event which automatically i) computes its area and duration and ii) set the *ADCs* for data acquisition.

The only mandatory setting of the RO gradient is the corresponding spatial axis, e.g:

```
        Axis=GX
```

The gradient area is automatically calculated from the parameter values `FOVx` , `Nx` and the duration from `ReadBW`. Note that the read bandwidth defines the duration of the flat-top, not the duration of the complete gradient shape. Also, the size of k-space equals the area under the flat-top, not the area of the complete gradient shape. Finally, the sampling of the N_x data points is equidistantly performed on the flat-top. These are important requirements for a correct k-space encoding, and it would be very difficult to achieve the same result with a simple gradient event (and the ADC settings in another *EmptyPulse* event), also possible.

2.7 Definition of phase-encode gradient tables

A phase-encode (PE) gradient needs to change the value of its area within a loop. Therefore, it is often called a phase-encode table, holding all values which have to be switched in the loop.

Again, there is a predefined gradient event which automatically computes this table. However, if there are multiple loops defined in the sequence, it has to be specified which of those loops switches through the table. This is done with the `ConnectToLoop` attribute:

```
        Axis=GY ConnectToLoop=1
```

PE gradient takes the counter of the first loop (*ConcatSequence*) for switching the table, where “first” means the position of the *ConcatSequence* in the sequence tree as it is seen from the PE gradient. Thus, the `ConnectToLoop` attribute specifies the number of steps “upwards” in the tree to find the loop for switching the phase-encode table.

Further, it is important to note that the PE gradient sets the number of repetitions of the connected looping *ConcatSequence.*

Again, as with the read gradient, the area values of the phase-encode table are automatically calculated from the field-of-view and the number of samples of the corresponding axis in the parameter settings.

The PE gradient is a dynamic event, since it changes its shape during execution depending on some loop counter. Instead, the simple gradient and the RO gradient are static events. Thus, it is not useful to set the area of a *TGPS* with the `Area="PE_TGPS"` method, since it will not follow the dynamics (it will simply take the first value of the table).

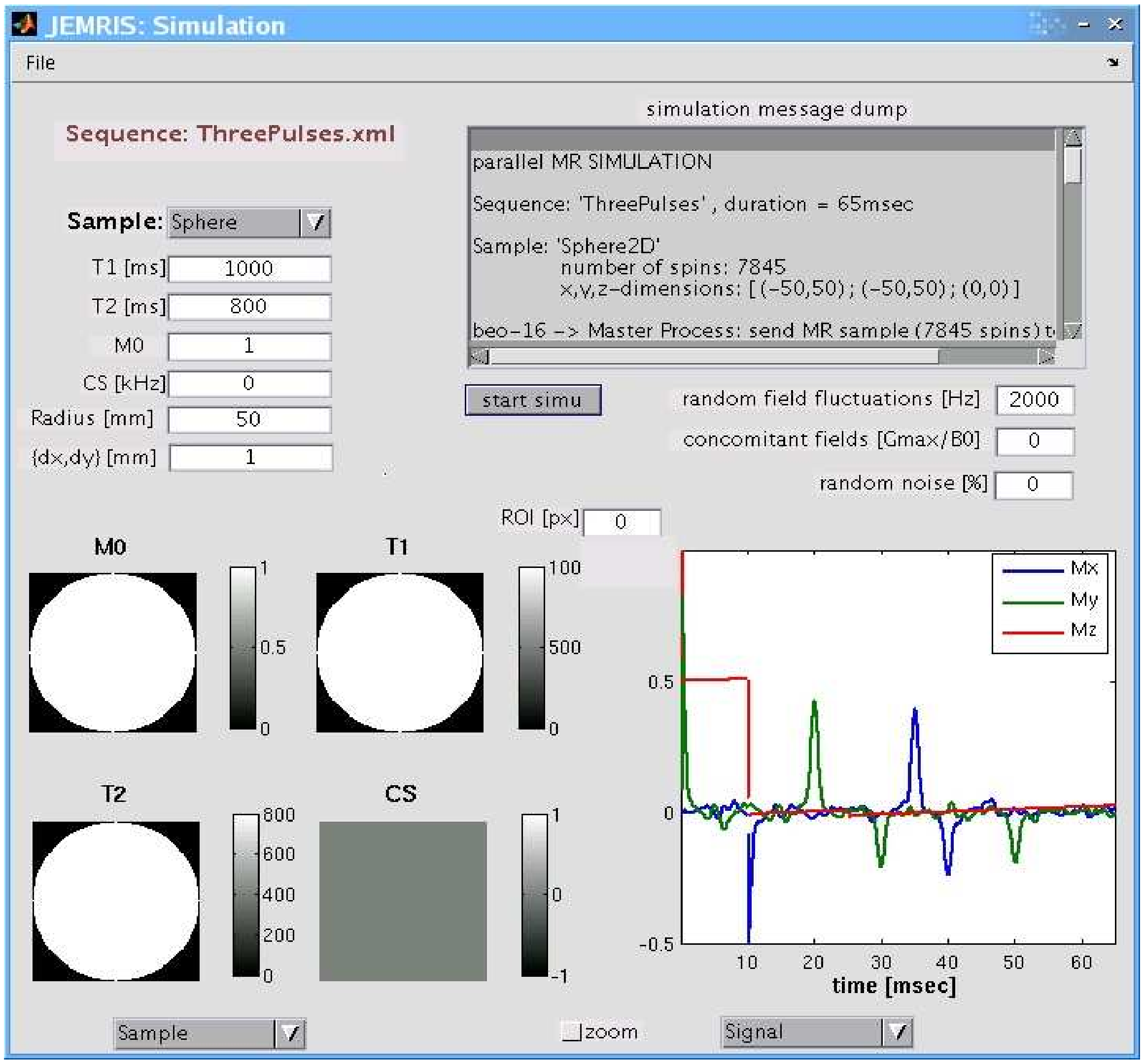
However, often it is needed to “rewind” a phase-encode table, which is possible by adding another PE gradient which has inverted amplitudes:

PE gradient 1: `Axis=GY ConnectToLoop=2`
`...`
PE gradient 2: `Axis=GY Factor=-1.0 ConnectToLoop=2`

Here, after the second phase-encode table the net gradient area along the y-axis is zero in every step of the loop.

3. Simulation: sample and model specifications, execution, and visualisation

The MATLAB command `JEMRIS_sim` launches the simulation GUI. The following figure shows the GUI for a simple sequence of three RF pulses, each followed by data acquisition:



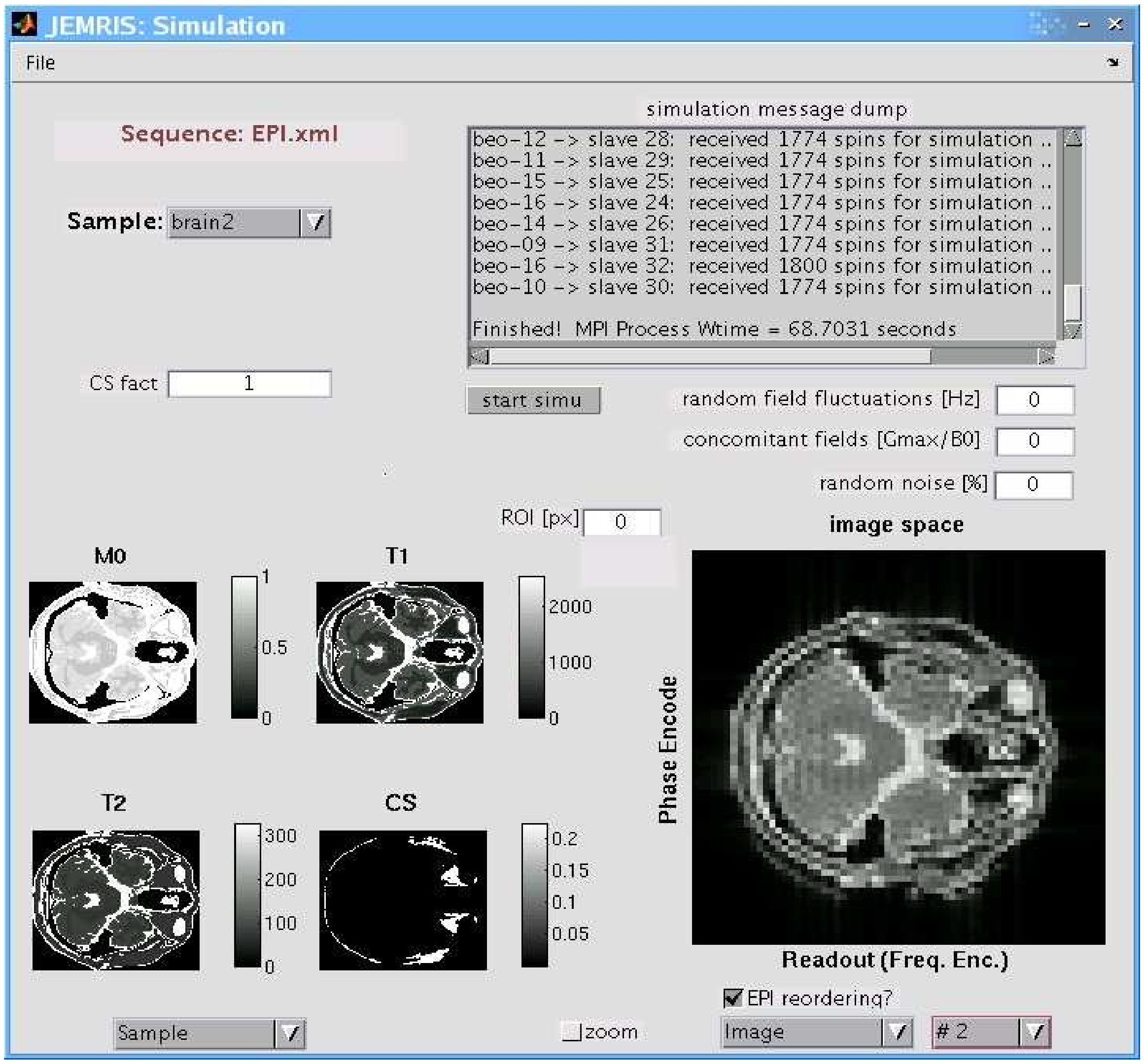
```
parallel MR SIMULATION

Sequence: 'ThreePulses', duration = 65msec

Sample: 'Sphere2D'
        number of spins: 7845
        x,y,z-dimensions: [(-50,50);(-50,50);(0,0)]

beo-16 -> Master Process: send MR sample (7845 spins) t
```

- Again, the **"File"** dialogue allows to specify the sequence for the simulations.
- The **"Sample"** selection allows the choice between four different types of samples: i) a homogenous sphere, ii) two spheres, and iii) and iv) are two different slices across the human brain (both need a FOV of approx. 256 mm). For i) and ii) the physical properties of the sphere(s) can be changed in the following dialogues. The last dialogue, **"{dx, dy} mm"**, specifies the resolution of the sample. Choosing a small number results in a huge number of spins and therefore long calculation times.
- The **"start simu"** button starts the simulation. Once it is finished, it also dumps some output and depicts the sample in the lower left display and the simulated

signal in the lower right display, respectively. The views of this can be changed with the selection buttons below. Here, the choices **“k-space”** or **“image”** are only functional for imaging sequences.

- The **“random field fluctuation”** dialogue allows adding small random field fluctuations to each spin, changing the apparent transverse relaxation time (T_2^*). Of course, the simulation has to be restarted to take effect. (The same holds for adding **“concomitant field”** terms to the simulation.)
- Instead, adding **“random noise”** is performed after the simulation and the result will be displayed immediately on the signal/image.
- Displaying k-spaces/images allows further a reordering of the k-space signal in case of an EPI sequence (flipping every other line).

The following figure shows an imaging simulation examples for the EPI sequence:



Sequence: EPI.xml

Sample: brain2

CS fact 1

simulation message dump

```
beo-12 -> slave 28:  received 1774 spins for simulation ..
beo-11 -> slave 29:  received 1774 spins for simulation ..
beo-15 -> slave 25:  received 1774 spins for simulation ..
beo-16 -> slave 24:  received 1774 spins for simulation ..
beo-14 -> slave 26:  received 1774 spins for simulation ..
beo-09 -> slave 31:  received 1774 spins for simulation ..
beo-16 -> slave 32:  received 1800 spins for simulation ..
beo-10 -> slave 30:  received 1774 spins for simulation ..

Finished!  MPI Process Wtime = 68.7031 seconds
```

random field fluctuations [Hz] 0

concomitant fields [Gmax/B0] 0

random noise [%] 0

ROI [px] 0

[✓] EPI reordering?

Sample | Image | # 2