

# MRiLab v1.0 User Guide

Fang Liu  
(leoliuf@gmail.com)

October 3, 2013



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	What is MRiLab . . . . .	1
1.2	Obtaining MRiLab . . . . .	1
1.3	Installing and Running MRiLab . . . . .	2
<b>2</b>	<b>Platform Overview</b>	<b>5</b>
2.1	MRiLab Simulation Platform . . . . .	5
2.2	Simulation Workflow . . . . .	7
2.3	Gradient Echo: Starting A Simple Scan . . . . .	9
<b>3</b>	<b>Simulation Settings</b>	<b>11</b>
3.1	Loading Virtual Object . . . . .	11
3.2	Loading Sequence . . . . .	13
3.2.1	Loading Predefined Sequence . . . . .	13
3.2.2	Predefined Sequences . . . . .	15
3.3	Loading Coil . . . . .	16
3.3.1	Loading Predefined Coil . . . . .	16
3.3.2	Predefined Coil . . . . .	18
3.4	Loading Magnet . . . . .	18
3.4.1	Loading Predefined Magnet . . . . .	19
3.4.2	Predefined Magnet . . . . .	20
3.5	Loading Gradient . . . . .	20
3.5.1	Loading Predefined Gradient . . . . .	20
3.5.2	Predefined Gradient . . . . .	21
3.6	Loading Motion . . . . .	22
3.6.1	Loading Predefined Motion . . . . .	22
3.6.2	Predefined Motion . . . . .	23
3.7	Setting Scan Parameters . . . . .	23
3.8	Parameter List . . . . .	23
3.8.1	Imaging . . . . .	24
3.8.2	Advanced . . . . .	25
3.8.3	Hardware . . . . .	25
3.8.4	Recon . . . . .	26
3.8.5	CVs . . . . .	26

3.8.6	SpecialTech . . . . .	27
3.9	Parallel Computation . . . . .	29
<b>4</b>	<b>Simulation</b>	<b>31</b>
4.1	Running Simulation . . . . .	31
4.2	Image, SNR and SAR . . . . .	32
<b>5</b>	<b>MRiLab Toolboxes</b>	<b>35</b>
5.1	RF Pulse Design . . . . .	35
5.1.1	RF Design GUI . . . . .	35
5.1.2	RF Macro Library . . . . .	41
5.1.3	Make Your Own RF Macro . . . . .	52
5.2	MR Sequence Design . . . . .	55
5.2.1	Sequence Design GUI . . . . .	55
5.2.2	Virtual Structure . . . . .	62
5.2.3	GzSS Macro Library . . . . .	63
5.2.4	GyPE Macro Library . . . . .	68
5.2.5	GxR Macro Library . . . . .	73
5.2.6	ADC Macro Library . . . . .	79
5.2.7	Ext Macro Library . . . . .	81
5.2.8	Make Your Own Ext Plugin . . . . .	83
5.2.9	Make Your Own Gradient Macro . . . . .	86
5.2.10	Make Your Own MR Sequence . . . . .	88
5.2.11	Make Your Own Virtual Object . . . . .	88
5.3	Coil B1 Design . . . . .	90
5.3.1	Coil Design GUI . . . . .	90
5.3.2	Coil Element Macro Library . . . . .	93
5.3.3	Make Your Own Coil . . . . .	94
5.4	Magnet dB0 Design . . . . .	94
5.4.1	Magnet Design GUI . . . . .	95
5.4.2	Magnet Element Macro Library . . . . .	97
5.4.3	Make Your Own Magnet . . . . .	98
5.5	Gradient Design . . . . .	98
5.5.1	Gradient Design GUI . . . . .	99
5.5.2	Gradient Element Macro Library . . . . .	101
5.5.3	Make Your Own Gradient . . . . .	102
5.6	Motion Design . . . . .	102
5.6.1	Motion Design GUI . . . . .	103
5.6.2	Motion Element Macro Library . . . . .	104
5.6.3	Make Your Own Motion . . . . .	105
5.7	Image Reconstruction . . . . .	106
5.7.1	Default Engine . . . . .	106
5.7.2	External Engine . . . . .	106
5.8	Simulation Output . . . . .	108

<b>6 Image Display and Analysis</b>	<b>109</b>
6.1 MatrixUser . . . . .	109
6.1.1 Data Import . . . . .	110
6.1.2 Window Layout . . . . .	111
6.1.3 Function Library . . . . .	116
6.2 arrayShow . . . . .	130
6.3 SpinWatcher . . . . .	130
6.4 SARWatcher . . . . .	132
<b>7 MRiLab Applications</b>	<b>133</b>
7.1 bSSFP with Non-uniform B0 . . . . .	133
7.2 Fat Chemical Shift . . . . .	133
7.3 Multi RF Transmitting . . . . .	135
7.4 Multi Receiving Coil . . . . .	136
7.5 Image Gradient . . . . .	136
7.6 Motion Artifact . . . . .	137
<b>8 FAQs</b>	<b>139</b>
<b>Acknowledgment</b>	<b>143</b>
<b>Afterword</b>	<b>145</b>



# Preface

Numerical MRI simulation can dramatically speed the understanding and development of new MR imaging methods. In this work, I have developed a new simulation package named 'MRiLab' for performing fast 3D parallel MRI numerical simulation on a simple desktop computer. This simulation package is aimed to provide a fast, comprehensive and effective numerical MRI simulation solution with minimum computation hardware requirement.

This manual is aimed to provide a comprehensive introduction for various functionality included in MRiLab, through the workflow of simulation, to detailed toolboxes that assist customizing different aspects of MR simulation experiment. You should be able to perform all the functions available in MRiLab after reading this User Guide.

It's my greatest pleasure to know that MRiLab can help you in your MR research or education. Please don't hesitate to leave me feedback about any aspect of MRiLab and/or about this User Guide. All the effects for improving MRiLab will hopefully help MR researchers including myself for better understanding and improving future MR techniques.



# Chapter 1

## Introduction

### 1.1 What is MRiLab

The MRiLab is a numerical MRI simulation package. It has been developed to simulate factors that affect MR signal formation, acquisition and image reconstruction. The simulation package features highly interactive graphic user interface (GUI) for various simulation purposes. MRiLab provides several toolboxes for MR researchers to analyze RF pulse, design MR sequence, configure multiple transmitting and receiving coils, investigate B0 in-homogeneity and object motion sensitivity etc. The main simulation platform combined with these toolboxes can be applied for customizing various virtual MR experiments which can serve as a prior stage for prototyping and testing new MR technique and application.

The MRiLab provides high computation accuracy by simulating multiple spin evolution within small time interval using discrete Bloch-equation. In order to manipulate large multidimensional spin matrix, MRiLab employs parallel computation by incorporating latest GPU technique and multi-threading CPU technique. With the accelerated computation, MRiLab can accomplish multidimensional multiple spin species MR simulation with high accuracy within short time and under low computation hardware requirements.

### 1.2 Obtaining MRiLab

The current version of MRiLab is made available and can be downloaded from SourceForge website. MRiLab is released as free software. This means that you are free to use and modify this software as your needs, as long as you acknowledge the original author in any future work. If you find MRiLab useful for the publication of any scientific results, including a line in your acknowledgments section referencing to MRiLab and this belowing address is requested.

MRiLab downloading address:

<https://sourceforge.net/projects/mrilab/>

### 1.3 Installing and Running MRiLab

In order to run MRiLab, the Matlab core is required. Simulink 3D Animation relevant packages are required for analyzing object motion in MRiLab. The current version of MRiLab has been tested under the following Matlab versions:

- Matlab R2011a 64-bit Windows
- Matlab R2012b 64-bit Unix

Installing and running MRiLab is easy, you only need to download MRiLab source code in a compressed file, then uncompress the MRiLab root folder, put the folder to any location in your computer. To run MRiLab, you need open Matlab, then simply run the MRiLab.m script that is under the MRiLab root folder.

The graphic interface of MRiLab is developed using Matlab GUIDE environment. Majority of the configuration code is written using pure Matlab language and XML, however, the computation intensive functions are optimized and written using Matlab C MEX. This includes the computation kernel working with GPU using NVIDIA CUDA and multi-threaded CPU using OpenMP. Some 3D rendering functions are also coded using Visualization Toolkit (VTK). I have compiled these MEX library files for 64-bit Windows and Linux OS system and shipped them with MRiLab source code, so that the user should be able to use these files under 64-bit Windows and Linux without compiling. However, if these MEX files are incompatible with your OS system for any reason or if you wish to modify these MEX files, you have to recompile them using the source code.

Before compiling these MEX files, the following dependent packages are required.

1. CMake (required)

The MRiLab uses CMake for cross platform building for these MEX files.

CMake : <http://www.cmake.org/cmake/resources/software.html>

2. IPP or Framework (required)

The current version of MRiLab uses matrix manipulation libraries from either Intel IPP or AMD Framework. Please notice that Intel IPP is the only MRiLab dependency that is not free open source, however if you are planning to use MRiLab IPP version (i), you can download Intel C Studio XE by following Intel non-commercial license. As an alternative, MRiLab

provides Framewave version (f), and Framewave is a free open source library.

Intel IPP : <http://software.intel.com/en-us/intel-ipp>  
AMD Framewave : <http://framewave.sourceforge.net>

### 3. CUDA (optional)

The NVIDIA driver is required for driving GPU and for MRiLab to work with GPU parallel computation. The current version of MRiLab only supports GPU which supports NVIDIA CUDA technique. And the libraries from CUDA 5.0 are used in the current version of MRiLab.

NVIDIA : <http://www.nvidia.com/page/home.html>  
CUDA : [http://www.nvidia.com/object/cuda\\_home\\_new.html](http://www.nvidia.com/object/cuda_home_new.html)

Although GPU acceleration can dramatically improve computation efficiency for MRiLab, the GPU is also optional. Parallel computation can be accomplished by using multi-threaded CPU which requires no additional package to be installed on today's regular computer.

### 4. VTK (optional)

The VTK is required for rendering 3D K-Space trajectory. The current version of MRiLab uses libraries from VTK 5.10. However, VTK is optional since Matlab based rendering is also provided.

VTK : <http://www.vtk.org>

### 5. ISMRMRD (optional)

MRiLab supports data conversion from MAT to ISMRMRD which is used as a default data storage format by Gadgetron MRI image reconstruction framework. The interested user needs to install ISMRMRD required packages in order to compile data conversion MEX.

ISMRMRD : <http://ismrmrd.sourceforge.net/#obtaining-and-installing>

If you finish installing the necessary dependent packages for MRiLab, you also need to set a few necessary environment variables in your system :

- MATLAB\_ROOT : Matlab root folder path
- IPP\_ROOT : IPP root folder path if IPP is used
- FRAMEWAVE\_ROOT : Framewave root folder path if Framewave is used

Notice that the C source code for these MEX files is in /MRiLab/Lib/src folder. To compile and install MEX files

### 1. Linux Installation

The command line that compiles the MEX files is

```
mkdir build
cd build
cmake MRiLab/Lib/src
make
sudo make install
```

You can also use CMake GUI for building and other graphic compiling tools (e.g. Eclipse) for compiling.

### 2. Windows Installation

It is recommended to use CMake GUI for building a Visual Studio project. Then you can compile by using Visual Studio.

If installation problems do occur to you, feel free to let me know and I may help you out. For your information, I provide here my development environment (Table 1.1) for the current version of MRiLab.

Environment	Desktop	Laptop
Machine	Dell Precision T3500	Lenovo Thinkpad R61
CPU	Intel Xeon W3530	Intel Core2 Duo T8100
GPU	NVIDIA Quadro 4000	Null
OS	Windows 7 64-bit	Linux Fedora 16 64-bit
Matlab	Matlab R2011b 64-bit	Matlab R2012b 64-bit
C Compiler	Visual Studio 10 Win64	GCC 4.6.3
VTK	VTK 5.10	VTK 5.10
CUDA	CUDA 5.0	Null
IPP	Intel IPP 7.0	Intel IPP 7.0
Framewave	AMD Framewave 1.3	AMD Framewave 1.3

Table 1.1: Fang's Computer Environment

# Chapter 2

## Platform Overview

### 2.1 MRiLab Simulation Platform

The MRiLab consists of

1. A Main Simulation Control Console

The main simulation control console (Figure 2.1) functions like a MR scanner console for graphically adjusting imaging setup and conducting simulation control. Simulation feedback are dynamically reflected on corresponding information windows based on simulation process.

2. Function Toolboxes

The function toolboxes (Figure 2.2) provide independent panels for designing RF pulse (e.g. SLR, non-adiabatic and adiabatic pulse etc.), for constructing arbitrary pulse sequence (e.g. SPGR, SSFP and FSE etc.), for configuring RF coil profile and main magnet field (i.e. B1 and B0 field) and for designing motion track. Additional image display and analysis tools (SpinWatcher, MatrixUser and arrayShow) are also developed and tailored to work with MRiLab reconstructed high dimensional images.

3. A Discrete Bloch-equation Solving Kernel

The solving kernel simulates spin evolution in tissue within small discrete time interval for accurately capturing spin behavior given the running pulse sequence. This kernel is accelerated using Matlab MEX functions that are optimized for running GPU and multi-threaded CPU parallel computation. Also, this kernel is taking charge of virtual signal acquisition followed by image reconstruction with saved K-Space data in the reconstruction module.

4. Macro Library

MRiLab uses the concept of macros for simplifying experiment design. The macro in MRiLab is defined as a programming-free module that can

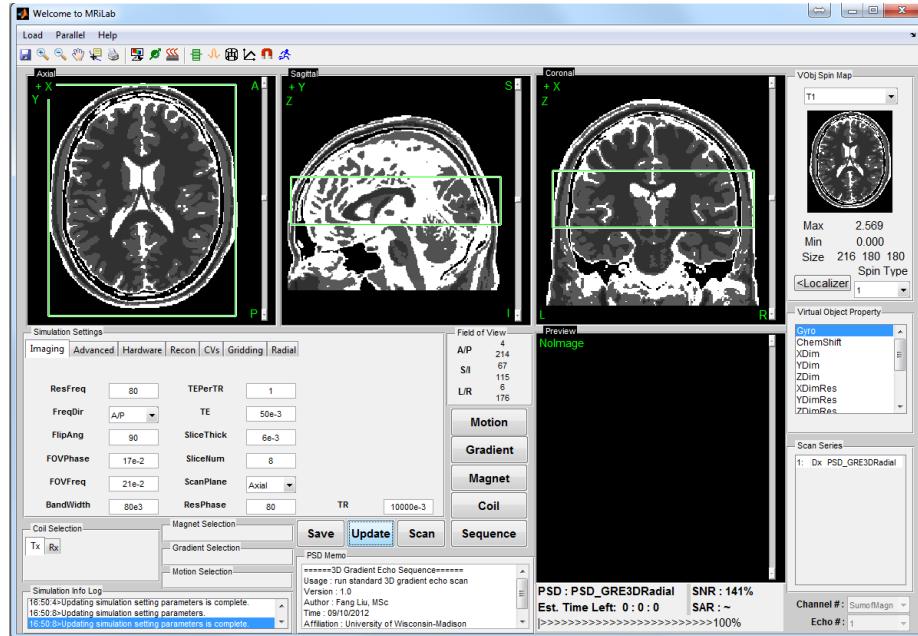


Figure 2.1: The MRiLab Main Simulation Control Console. This control console functions like a MR scanner console for graphically adjusting imaging setup and conducting simulation control.



Figure 2.2: The Shortcut for Function Toolbox on Simulation Control Console. Each icon is associated with individual toolbox for specific functions.

be added, deducted and modified in constructing MR sequences, coil profile, field inhomogeneity and motion track. For instance, a Sinc RF pulse (`rfSinc`) is considered as a simple macro that can be used for constructing a gradient echo sequence, and the attributes of this macro include pulse starting time (`tStart`), pulse ending time (`tEnd`) and the time bandwidth product (`TBP`) etc. MRiLab provides a macro library (Figure 2.3) covering a wide range of macros. Using these predefined macros, you should be able to accomplish most of your design work. However, if special macros are needed, MRiLab also provides interfaces for communicating with user-defined macros. More detailed description for macros will be provided in separate sections in Chapter 5.

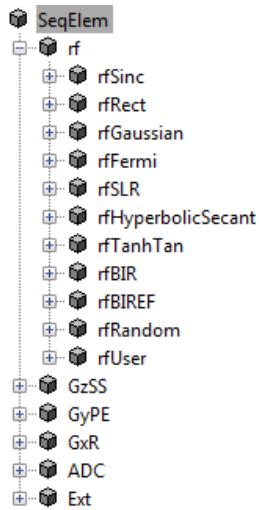


Figure 2.3: The Macro Library Tree Structure in MRiLab. Individual tree nodes under `SeqElem` root are functional macros that can be used for designing MR sequences. Notice that only RF nodes are unfolded here for display purpose.

MRiLab uses XML files for storing simulation information, which simplifies simulation transition across different studies. MRiLab also supports external plugins written with either Matlab or C language for creating extendable simulation system.

## 2.2 Simulation Workflow

The workflow diagram of MRiLab simulation is shown in Figure 2.4. One typical simulation requires input :

- Virtual object with specific tissue properties including Rho, T1 and T2 etc.

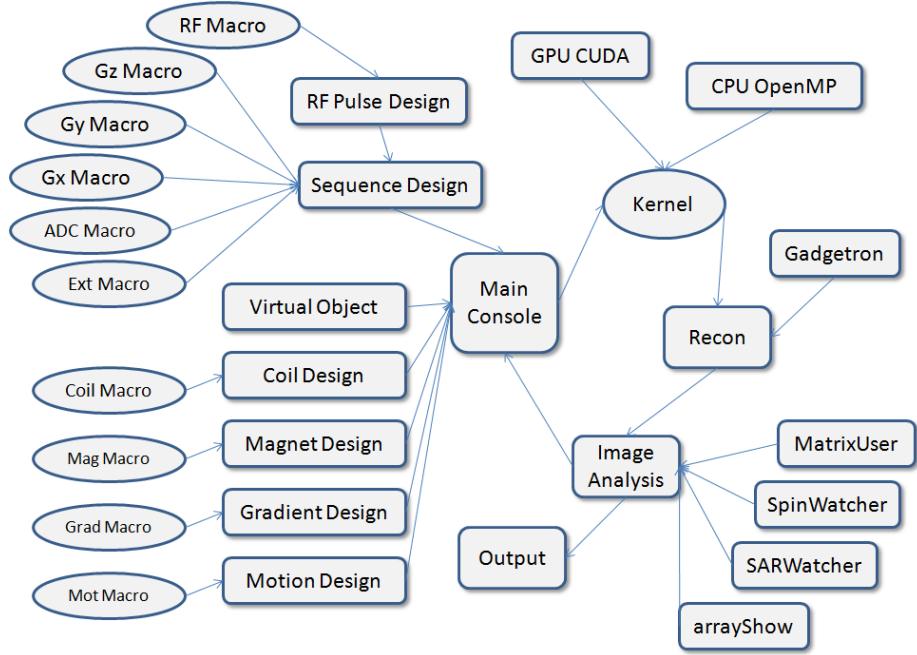


Figure 2.4: MRiLab Workflow Diagram

- MR sequence that provides desired tissue contrast in reconstructed images
- B1 field for RF transmitting and receiving
- dB0 field for describing main static field inhomogeneity
- Gradient field for signal spatial encoding
- Motion pattern for describing object movement during real time simulation

All the input information gets configured at the MRiLab main control console where the user can graphically customize any aspect of the simulation experiment. The main console preprocesses the input information then converts them into kernel signal. The discrete solving kernel executes each voxel based on the kernel signal with either GPU or multi-threading CPU acceleration. The acquired MR signal and K-Space information from the kernel then passes to image reconstruction module where either default recon code or external recon tool (e.g. Gadgetron) is applied. The reconstructed image can be analyzed using MRiLab image display tools including :

- MatrixUser : An image display and analysis tool for manipulating multi-dimensional real value matrix

- SpinWatcher : An analysis tool for analyzing spin evolution behavior of single voxel
- SARWatcher : An analysis tool for analyzing local spatial SAR distribution
- arrayShow : A Matlab image viewer for the evaluation of multidimensional complex images

## 2.3 Gradient Echo: Starting A Simple Scan

Up to this point, you may wonder how I can start to use MRiLab for my simulation. Here is a simple example for you to get some basic feelings of using MRiLab. A 3D Gradient Echo (GRE) sequence will be used for simulating images. Let's get it started!

- Open MRiLab by running MRiLab.m under the root folder. MRiLab will try to detect your CPU and GPU, and initialize the simulation environment. After initialization (typically in couple seconds), MRiLab Main Simulation Control Console will pop-up. Empty Axial, Sagittal, Coronal and Preview views show up. Notice that on the console, several push buttons are disabled at this point, it means more inputs are needed for enabling them.
- To simulate images, the virtual object or digital phantom is needed. Go to ‘Load’, ‘Load Phantom Example’, choose one of the predefined digital phantoms for the experiment. Several digital phantoms that are suitable for different experiment purposes are already provided. Here, let's choose ‘Brain (Standard Resolution 108x90x90)’. After loading this phantom, you will notice the preview image showing up at the top right corner under ‘VObj Spin Map’ and the basic information about this phantom are also shown in ‘Virtual Object Property’. Let's choose T2 map from the pop-up box, and click ‘Localizer’ button for showing image details. At this point, Axial, Sagittal and Coronal views are filled with this 3D digital phantom. **Then click ‘Update’ button to accept your phantom loading.** Notice that after updating, all push buttons are enabled.
- Since we want to simulate 3D GRE sequence, we need to load 3D GRE sequence first. Click the ‘Sequence’ button located at the center portion of the main control console. The SeqList will pop-up where you can choose and load your sequences. Let's click ‘Dimension’ pop-up box and choose ‘3D’. The Category list will show a full list of sequence type. Click ‘GradientEcho’, then click ‘PSD\_GRE3D’ from the Sequence list. Click ‘Accept’ button to accept and load this 3D GRE sequence. Notice that the ‘Simulation Settings’ tabs update and change to the setting for the current GRE sequence. **Then click ‘Update’ button to accept your sequence loading.**

- Under ‘Imaging’ tab, there are several imaging setting parameters you can play with, for instance, Field-of-View in frequency encoding direction (FOVFreq). We can simply accept the default setting at this moment, however, if you do make any changes, **you need to click ‘Update’ button to accept your changes before proceeding to Scan.**
- The final step is to click ‘Scan’ button and wait for the simulation to be performed. After simulation finishes in a short period of time, the reconstructed image will be shown in the ‘Preview’ view.

The final looks for this 3D GRE experiment should be somewhat like this (Figure 2.5).



Figure 2.5: The Final Simulation Result for 3D GRE Experiment

If you managed to get this simulated gradient echo image, congratulations! You have successfully performed your first MRiLab experiment. So you should be prepared for deeper understanding of MRiLab by following the rest of this manual.

## Chapter 3

# Simulation Settings

### 3.1 Loading Virtual Object

Before performing any simulation in MRiLab, the virtual object has to be loaded first. To load the virtual object, the user can go to menu ‘Load’ and ‘Load Phantom’ to load the user customized phantom or ‘Load Phantom Example’ to load MRiLab default phantoms. MRiLab provides a wide range of phantoms that cover several tissue types in human. After the virtual object being loaded, the phantom will show as a preview thumbnail at the top right corner of the main control console (Figure 3.1). The user can switch among T1, T2 and Rho etc. using the pop-up menu above the thumbnail. If multiple spin species exist in the phantom, the user can also switch the spin type by using ‘Spin Type’ pop-up menu below the thumbnail image. In addition, the virtual object properties will be provided at the ‘Virtual Object Property’ list, the user can check them by clicking the name of certain property. The properties of one virtual object typically include:

- Gyro (rad/s/T) : The gyromagnetic ratio of the spin
- ChemShift (Hz/T) : The chemical shift of the spin
- XDim : The number of voxels in X direction
- YDim : The number of voxels in Y direction
- ZDim : The number of voxels in Z direction
- XDimRes (m) : The voxel size in X direction
- YDimRes (m) : The voxel size in Y direction
- ZDimRes (m) : The voxel size in Z direction
- Type : The type of the spin

- TypeNum : The number of spin species
- Rho : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing spin density
- T1 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T1 relaxation time
- T2 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T2 relaxation time
- T2Star (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T2\* relaxation time

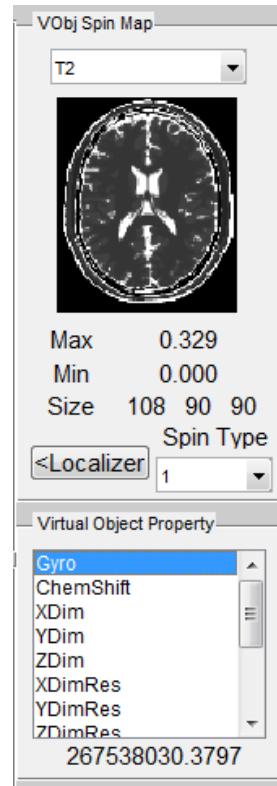


Figure 3.1: The Virtual Object Preview

The user can press the ‘Localizer’ button to push the thumbnail image to the main control console image display panel. MRiLab provides three image axes to hold axial, sagittal and coronal view section for the three dimensional virtual object. The scroll bar beside each axes allows changing the image slice in corresponding direction. This serves as an anatomical reference for further

image parameter setup. For example, the green box in each axes that indicates the current imaging field of view can be adjusted using free hand dragging, and the field of view information is updated at ‘Field of View’ panel when dragging.

## 3.2 Loading Sequence

One of the central roles of MRiLab is to enable simulation of a wide range of MR sequences and be able to manipulate sequence parameters in order to investigate and optimize desired MR contrast. In order to simulate those sequences, MRiLab provides an sequence loading interface for users to choose predefined sequences from default library or choose user customized sequences. MRiLab reads the chosen sequence and translates the sequence into specific signal that triggers the simulation kernel. Of course, the generated signal depends on imaging and other parameters that are adjustable on the simulation control console. The sequence loading interface provides functions to load MR sequence. The MR sequence design module is separate from loading interface and will be explained in detail in Chapter 5.

### 3.2.1 Loading Predefined Sequence

To open the sequence loading interface (Figure 3.2), you need to click the ‘Sequence’ button located at the center portion of the main control console. Once the interface is open, the ‘Dimension’ specifies the sequence spatial encoding scheme (2D or 3D). By default, 3D sequences are first recognized and provided. ‘Category’ provides a wide list of sequence classes including Gradient Echo, Spin Echo, Inversion Recovery, Fast Spin Echo, Others and User. After clicking one sequence category, the sequences belonging to chosen category will show up in the sequence list on the right. By clicking the specific sequence, the sequence will be highlighted and chosen. Then pressing ‘Accept’ will load the chosen sequence and this sequence loading interface will close. However, pressing ‘Cancel’ button will close the interface without loading any sequence.

Notice that there is a ‘Special Setting’ panel down below the sequence list. When a specific sequence is chosen, and if this sequence uses special techniques, the corresponding checkboxes beside the special techniques will be chosen. For example, by default PSD\_FIESTA3D uses the special techniques called DummyPulse, therefore, by clicking PSD\_FIESTA3D, the DummyPulse will be chosen accordingly. However, you can uncheck the checkbox for enforcing not loading DummyPulse module, but this may cause incomplete simulation for PSD\_FIESTA3D. It is recommended to keep default loading therefore much more sequence control is preserved. On the other hand, for the sequences that have no special techniques, you can also add special technique module by checking corresponding checkbox. This will load parameter tabs of special techniques on the simulation control console for further configuration. The special technique strategy enables the ability for reusing ‘capsulized’ module for different

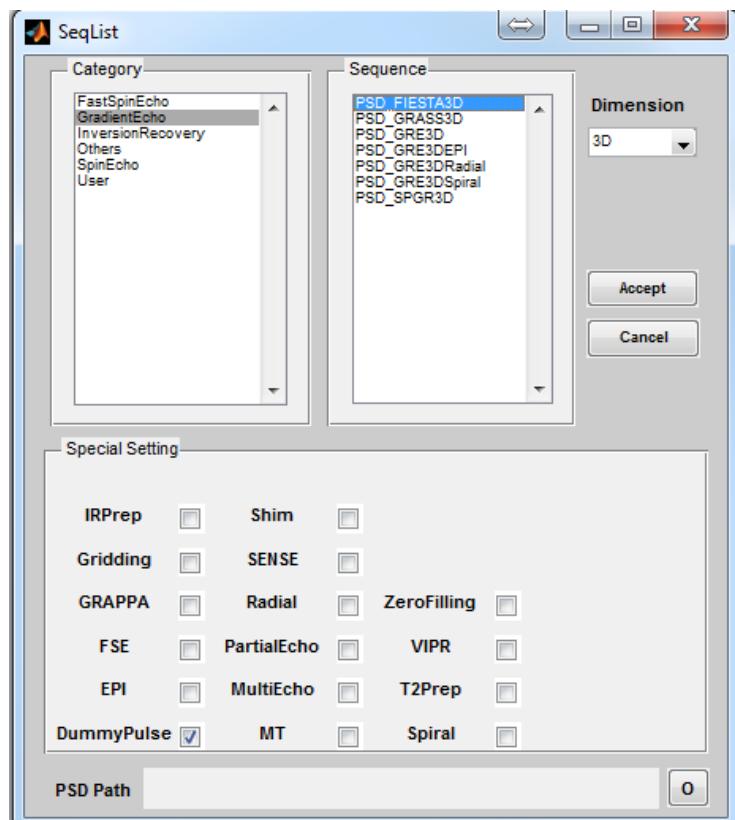


Figure 3.2: The Sequence Loading Interface. A sequence named PSD\_FIESTA3D from 3D GradientEcho Category is chosen, and DummyPulse special technique is checked by default.

sequences.

MRiLab provides several default sequences under the folder /PSD, notice the /PSD folder shares the same hierachic structure scheme as the loading interface. Typically a new sequence can be put anywhere in the computer, however it is recommended to put the sequence under these predefined categories under /PSD therefore they are visible to the loading interface. However, if a customized sequence is not directly visible to the loading interface, it can also be loaded using the PSD loading button marked as ‘o’ at the bottom. If the PSD loading button is used, loading interface will ignore other ordinary loading.

### 3.2.2 Predefined Sequences

MRiLab provides several predefined MR sequences including

#### 1. Fast Spin Echo

- PSD\_FSE3D

Three dimensional multishot Fast Spin Echo (FSE) sequence with interleaved K-Space sampling in Kx-Ky and conventional phase-encoding along Kz.

#### 2. Gradient Echo

- PSD\_FIESTA3D

Three dimensional balanced Steady State Free Precession (bSSFP) sequence.

- PSD\_SPGR3D

Three dimensional Spoiled Gradient Echo (SPGR) sequence.

- PSD\_GRE3D

Three dimensional gradient echo sequence with Cartesian readout.

- PSD\_GRE3DEPI

Three dimensional gradient echo sequence with multishot Echo Planar Imaging (EPI) readout using interleaved K-Space sampling in Kx-Ky and conventional phase-encoding along Kz.

- PSD\_GRE3DRadial

Three dimensional gradient echo sequence with radial readout in Kx-Ky and conventional phase-encoding along Kz, usually referred to as stack-of-stars sequence.

- PSD\_GRE3DSpiral

Three dimensional gradient echo sequence with multishot spiral readout in Kx-Ky and conventional phase-encoding along Kz.

#### 3. Inversion Recovery

- PSD\_IR3D  
Three dimensional Inversion Recovery (IR) sequence with Cartesian readout.
- 4. SpinEcho
  - PSD\_SE3D  
Three dimensional Spin Echo (SE) sequence with Cartesian readout.
- 5. User
  - PSD\_SPGR3DMT  
Three dimensional SPGR sequence with MT saturation.
- 6. Others
  - PSD\_AFI  
Three dimensional SPGR sequence for performing flip angle mapping using Actual Flip Angle Imaging (AFI) [1].

### 3.3 Loading Coil

In order to simulate multi-transmitting and receiving coil behavior, MRiLab provides a coil loading interface that allows users to choose different coil configurations for Tx (i.e. Transmitting) and/or Rx (i.e. Receiving). MRiLab translates the coil configuration and calculates the B1+/B1- field accordingly. For multi-transmitting coil, each coil element could be treated separately and receives individual RF signal source. This allows users to investigate B1 shimming and multiple RF excitation etc. For multi-receiving coil, each coil element also connects to an individual signal channel and produces signal according to its specific coil sensitivity. This allows users to investigate coil encoding methods including parallel imaging. The coil loading interface provides function to load coil configuration. The coil design module is separate from loading interface and will be explained in detail in Chapter 5.

#### 3.3.1 Loading Predefined Coil

To open the coil loading interface (Figure 3.3), you need to click the ‘Coil’ button located at the center portion of the main control console. The ‘Category’ list specifies different coil configuration category corresponding to anatomical structure. The ‘Coils’ list beside the ‘Category’ list provides coil configuration belonging to chosen category. By clicking the coil configuration name, the interface will calculate the coil sensitivity map and display in the preview axes. The user can specify displayed spatial resolution using the ‘Precision’ with the highest spatial resolution as the same as that of digital phantom, the lowest one as one of the sixteenth. Also the user can specify displayed color map from ‘Jet’, ‘Gray’ or ‘Hot’. Then pressing ‘Accept’ will load the chosen coil configuration and this coil loading interface will close. However, pressing ‘Cancel’ button

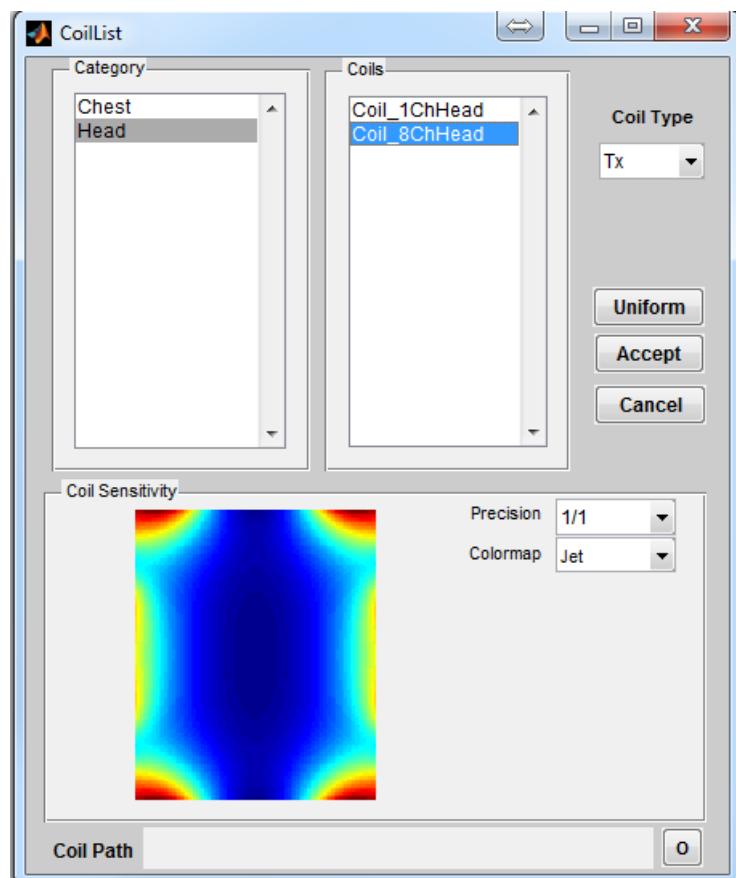


Figure 3.3: The Coil Loading Interface. An eight channel coil configuration named Coil\_8ChHead from Head Category is chosen, preview image shows the total coil sensitivity with the same spatial resolution as that of the chosen digital phantom.

will close the interface without loading any coil configuration. If the uniform unit coil sensitivity is desired, the user can press ‘Uniform’ button to load that. By default, MRiLab uses the uniform unit coil sensitivity for both RF transmitting and receiving. To indicate the usage of the coil configuration, the user has to specify ‘Coil Type’ to be either Tx for transmitting or Rx for receiving.

MRiLab provides several default coil configuration under the folder /Coil, notice the /Coil folder shares the same hierachic structure scheme as the loading interface. Typically a new coil configuration can be put anywhere in the computer, however it is recommended to put the coil configuration folder under these predefined categories under /Coil therefore they are visible to the loading interface. However, if a customized coil configuration is not directly visible to the loading interface, it can also be loaded using the Coil loading button marked as ‘o’ at the bottom. If the Coil loading button is used, loading interface will ignore other ordinary loading.

### 3.3.2 Predefined Coil

MRiLab provides several predefined coil configuration including

1. Head

- **Coil\_1ChHead**

A coil configuration consists of one single Biot-Savart circle, primarily used for testing purpose.

- **Coil\_8ChHead**

A coil configuration consists of 8 Biot-Savart circles, which produces relative flat B1 field in X-Y plane along X direction.

2. Chest

- **Coil\_9ChSurfChest**

A coil configuration consists of 9 Biot-Savart circles, which produces relative flat B1 field in X-Y plane along Y direction at the surface region.

## 3.4 Loading Magnet

Although in practical situation, the main field inhomogeneity is aimed to be alleviated as much as possible for better field performance, there are also cases that field inhomogeneity are required for particular purposes, such as investigating susceptibility artifact near metallic implant or developing sequences that are less sensitive to field inhomogeneity. To satisfy those developing intention, MRiLab provides the magnet loading interface allowing users to load customized B0 field inhomogeneity map (i.e. dB0). The magnet loading interface provides functions to load dB0 map. The magnet design module is separate from loading interface and will be explained in detail in Chapter 5.

### 3.4.1 Loading Predefined Magnet

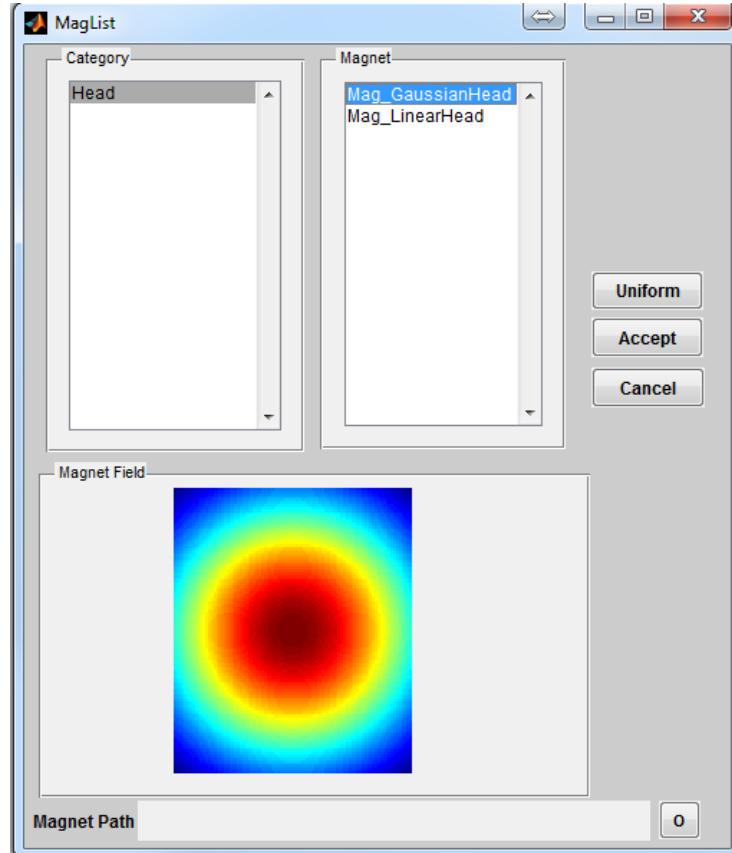


Figure 3.4: The Magnet Loading Interface. A magnet named Mag\_GaussianHead from Head Category is chosen, preview image shows the dB0 map in the X-Y plane with the same spatial resolution as that of the chosen digital phantom.

To open the magnet loading interface (Figure 3.4), you need to click the 'Magnet' button located at the center portion of the main control console. The 'Category' list specifies different magnet category corresponding to anatomical structure. The 'Magnet' list beside the 'Category' list provides magnet belonging to chosen category. By clicking the magnet name, the interface will calculate the dB0 map (i.e. a map for indicating main field variation) and display in the preview axes. Then pressing 'Accept' will load the chosen and this loading interface will close. However, pressing 'Cancel' button will close the interface without loading any. If the uniform B0 field (i.e. zero dB0) is desired, the user can press 'Uniform' button to load that. By default, MRiLab uses the uniform B0 field for simulation. That is to say there is no field inhomogeneity across the

entire virtual object.

MRiLab provides several default magnet under the folder /Mag, notice the /Mag folder shares the same hierachic structure scheme as the loading interface. Typically a new can be put anywhere in the computer, however it is recommended to put the folder under these predefined categories under /Mag therefore they are visible to the loading interface. However, if a customized is not directly visible to the loading interface, it can also be loaded using the Magnet loading button marked as ‘o’ at the bottom. If the loading button is used, loading interface will ignore other ordinary loading.

### 3.4.2 Predefined Magnet

MRiLab provides two predefined magnet configuration including

1. Head

- Mag\_GaussianHead  
A produces Gaussian dB0 field in three dimensional space.
- Mag\_LinearHead  
A magnet produces linear dB0 field in three dimensional space.

## 3.5 Loading Gradient

Regular MR spatial encoding is performed in a linear (flat) sense. Recently encoding in a nonlinear (curved) sense becomes interesting and challenging. MRiLab provides gradient loading interface to load customized 3D nonlinear gradient field. This function can help users investigate any arbitrary curved gradient field and use it directly for image simulation. Along with the powerful functions for MR sequence design, nonlinear gradient encoding techniques such as PatLoc can be simulated with minimal efforts in MRiLab. The gradient loading interface provides functions to load nonlinear gradient. The gradient design module is separate from loading interface and will be explained in detail in Chapter 5.

### 3.5.1 Loading Predefined Gradient

To open the gradient loading interface (Figure 3.5), you need to click the ‘Gradient’ button located at the center portion of the main control console. The ‘Category’ list specifies different gradient category corresponding to anatomical structure. The ‘Gradient’ list beside the ‘Category’ list provides gradient profile belonging to chosen category. After clicking the gradient name, pressing ‘Accept’ will load the chosen gradient profile and this gradient loading interface will close. However, pressing ‘Cancel’ button will close the interface without loading any gradient profile. If the constant unit gradient is desired, the user can press ‘Constant Unit’ button to load that. By default, MRiLab uses the

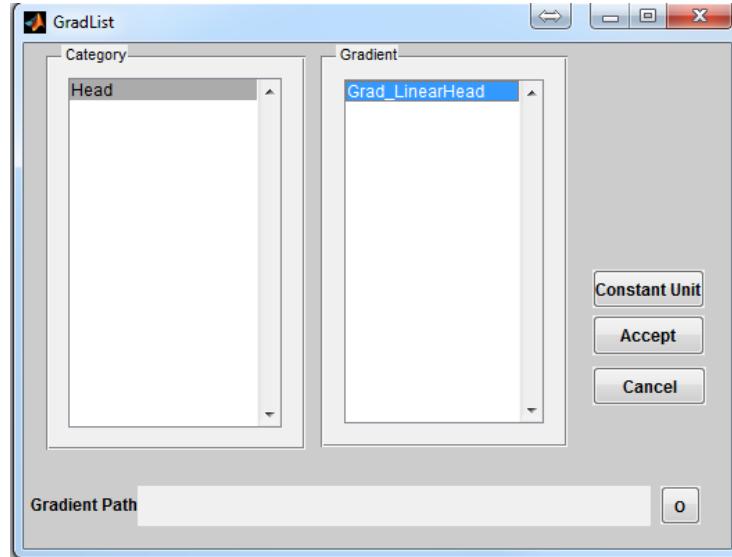


Figure 3.5: The Gradient Loading Interface. A gradient profile named `Grad.LinearHead` from Head Category is chosen.

constant unit gradient profile in X, Y and Z direction for simulation. This will maintain the conventional linear spatial encoding in all three dimensions.

MRiLab provides gradient profiles under the folder `/Grad`, notice the `/Grad` folder shares the same hierarchic structure scheme as the loading interface. Typically a new gradient profile can be put anywhere in the computer, however it is recommended to put the gradient profile under these predefined categories under `/Grad` therefore they are visible to the loading interface. However, if a customized gradient is not directly visible to the loading interface, it can also be loaded using the Gradient loading button marked as 'o' at the bottom. If the Gradient loading button is used, loading interface will ignore other ordinary loading.

### 3.5.2 Predefined Gradient

MRiLab provides one predefined gradient profile

#### 1. Head

- `Grad.LinearHead`

A gradient profile produces constant gradient field with varying gradient value in three dimensions, which can cause image contraction, expansion or shearing etc.

## 3.6 Loading Motion

MRiLab provides a motion loading interface that implements the functions for simulating object motion in three dimensional space. Motion simulation implementation in MRiLab is helpful in the sense that it enables an approach for simulating four dimensional imaging (+ time) techniques such as k-t blast and enables the development for real time image reconstruction algorithms. It also helps investigate motion insensitive sequence and observe motion artifact behavior in various types of sequences and conditions. The motion loading interface provides functions to load motion trajectory. The motion design module however is separate from loading interface and will be used for designing motion in three dimensional space.

### 3.6.1 Loading Predefined Motion

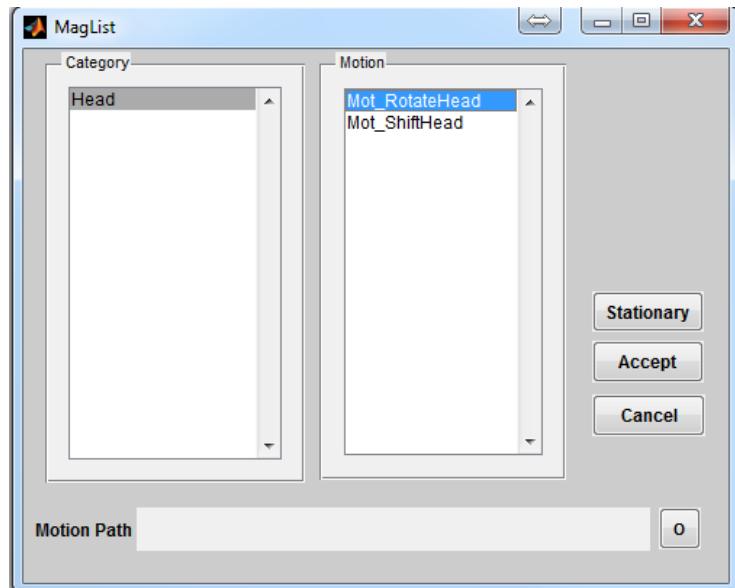


Figure 3.6: The Motion Loading Interface. A motion profile named Mot\_RotateHead from Head Category is chosen. This motion profile creates object rotation in three dimensions along any user defined axis.

To open the motion loading interface (Figure 3.6), you need to click the ‘Motion’ button located at the center portion of the main control console. The ‘Category’ list specifies different motion category corresponding to anatomical structure. The ‘Motion’ list beside the ‘Category’ list provides motion profile belonging to chosen category. After clicking the motion name, pressing ‘Accept’ will load the chosen motion profile and this motion loading interface will close. However, pressing ‘Cancel’ button will close the interface without loading any

motion profile. If motion isn't desired, the user can press 'Stationary' button. By default, no motion is assumed in MRiLab simulation.

MRiLab provides motion profiles under the folder /Mot, notice the /Mot folder shares the same hierarchic structure scheme as the loading interface. Typically a new motion profile can be put anywhere in the computer, however it is recommended to put the motion profile under these predefined categories under /Mot therefore they are visible to the loading interface. However, if a customized motion is not directly visible to the loading interface, it can also be loaded using the Motion loading button marked as 'o' at the bottom. If the Motion loading button is used, loading interface will ignore other ordinary loading.

### 3.6.2 Predefined Motion

MRiLab provides two predefined motion profile

1. Head

- Mot\_RotateHead  
A motion profile produces object rotation in three dimensional space along any user defined axis.
- Mot\_ShiftHead  
A motion profile produces object translation in any user defined direction.

## 3.7 Setting Scan Parameters

These loading interfaces provide a mechanism to interpret and convert configuration file to MRiLab readable parameters. After loading, the loaded information is present at corresponding fields at the main simulation control console. The 'Coil Selection', 'Magnet Selection', 'Gradient Selection' and 'Motion Selection' show the current chosen configuration. The user can change these configuration by reloading new configuration file from above. If any of these fields are empty, the default setting will be used. Like real scanner system, MRiLab categorizes the scanning parameters into different groups and present them within different tabs in the 'Simulation Settings' panel (Figure 3.7). There are five tabs that are common to all sequences, including Imaging, Advanced, Hardware, Recon and CVs. Additional tabs for one or more special techniques can also be present if loaded. Below are detailed explanation for each of these parameters.

## 3.8 Parameter List

Below are the full list of parameters that are supported by current version of MRiLab. Notice that MRiLab uses SI units for all the parameters unless otherwise specified.

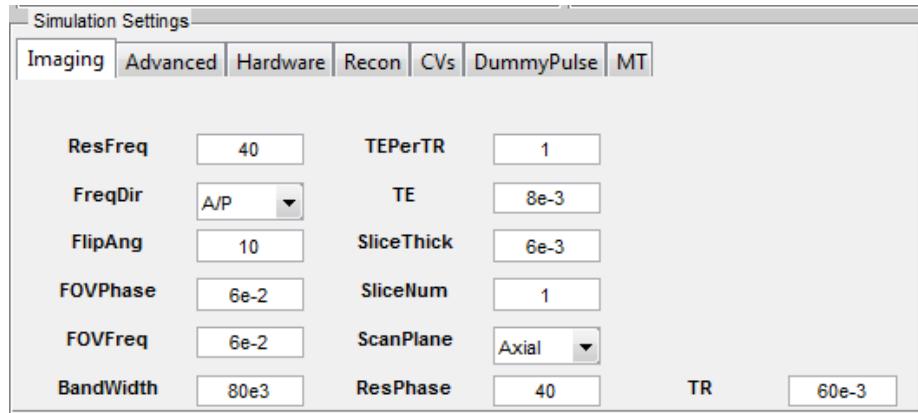


Figure 3.7: The Simulation Settings Panel. The panel includes Imaging, Advanced, Hardware, Recon and CVs tabs, two additional special technique DummyPulse and MT are also added for this sequence.

### 3.8.1 Imaging

The ‘Imaging’ tab contains parameters relevant to image resolution, field of view and timing setting etc.

- BandWidth (Hz) : Full receiver bandwidth
- FOVFreq (m) : Field of view in the frequency encoding direction
- FOVPhase (m) : Field of view in the first phase encoding direction
- FlipAng (Degree) : Flip angle of excitation pulse
- FreqDir : Frequency encoding direction
- ResFreq : Number of voxels in frequency encoding direction
- ResPhase : Number of voxels in the first phase encoding direction
- ScanPlane : The scanning plane
- SliceNum : The number of encoding slice
- SliceThick (m) : The thickness of one slice
- TE (s) : The time of echo
- TEPerTR : The number of echoes in multi echo mode, default one uses single echo, use a value above one requires MultiEcho tab to be loaded
- TR (s) : The time of repetition

### 3.8.2 Advanced

The ‘Advanced’ tab contains other further imaging parameters.

- MasterTxCoil : The master transmitting coil in multi RF transmitting mode
- MultiTransmit : The flag for turning on and off multi RF transmitting mode, default mode is off for single RF transmitting
- NEX : The number of excitation
- NoFreqAlias : The flag for preventing aliasing in frequency encoding direction, default on truncates object outside field of view in frequency encoding direction
- NoPhaseAlias : The flag for preventing aliasing in the first phase encoding direction, default on truncates object outside field of view in the first phase encoding direction
- NoSliceAlias : The flag for preventing aliasing in the second phase encoding (i.e. slice encoding) direction, default on truncates object outside field of view in slice encoding direction
- Shim : The flag for choosing shimming mode
- TEAnchor : The flag for choosing TE time offset with respect to the RF pulse width

### 3.8.3 Hardware

The ‘Hardware’ tab contains parameters relevant to system hardware setup.

- B0 (T) : Main static magnetic field strength
- B1Level (T) : The reference B1 field which produces prescribed flip angle
- MaxGrad (T/m) : Maximum allowable gradient strength
- MaxSlewRate (T/m/s) : Maximum allowable slew rate
- MinUpdRate (s) : Minimum update rate for producing sequence signal
- Model : System model with the closest setting to the real scanner system
- NoiseLevel : The level of adjustable noise, the higher the number, the more noise
- SpinPerVoxel : The number of spins in each voxel, default one for treating T2\* equal to T2, use a value above one for simulating T2\* effect

### 3.8.4 Recon

The ‘Recon’ tab contains parameters relevant to image reconstruction.

- AutoRecon : The flag for turning on and off automatic image reconstruction after signal acquisition
- ExternalEng : The script for doing image reconstruction using user customized external method
- OutputType : The type of output data including both image and signal, choosing options include ‘MAT’ and ‘ISMRRMRD’, the latter requires ISMRRMRD dependency packages to be installed
- ReconEng : The image reconstruction engine, choosing ‘Default’ uses MRiLab default reconstruction code, choosing ‘External’ uses external engine which requires ExternalEng to be provided
- ReconType : The type of image reconstruction

### 3.8.5 CVs

The ‘CVs’ tab contains Controllable Variable (CV) that can be freely referenced in the sequence and adjusted at the simulation control console. They are designed for conveniently transferring values among multiple sequence modules. The user can use them for customized purpose.

- CV1 : Controllable variable 1
- CV2 : Controllable variable 2
- CV3 : Controllable variable 3
- CV4 : Controllable variable 4
- CV5 : Controllable variable 5
- CV6 : Controllable variable 6
- CV7 : Controllable variable 7
- CV8 : Controllable variable 8
- CV9 : Controllable variable 9
- CV10 : Controllable variable 10
- CV11 : Controllable variable 11
- CV12 : Controllable variable 12
- CV13 : Controllable variable 13
- CV14 : Controllable variable 14

### 3.8.6 SpecialTech

The SpecialTech contains multiple tabs from which one or more are loaded during sequence loading according to sequence configuration and user choice.

#### 1. DummyPulse

The ‘DummyPulse’ tab are designed for adding dummy pulse section before image acquisition section. It can be used for skipping transient steady state signal.

- DP\_Flag : The flag for turning on and off dummy pulse
- DP\_FlipAng (Degree) : The flip angle of excitation pulse for dummy pulse
- DP\_Num : The number of TRs for dummy pulse
- DP\_TR (s) : The time of repetition for dummy pulse

#### 2. EPI

The ‘EPI’ tab contains parameters for performing multi shot interleaved EPI readout.

- EPI\_ESP (s) : The echo spacing for EPI
- EPI\_ETL : The echo train length for EPI
- EPI\_EchoShifting : The flag for turning on and off echo shifting
- EPI\_ShotNum : The number of EPI shots, multi shot EPI uses interleave mode

#### 3. FSE

The ‘FSE’ tab contains parameters for performing multi shot interleaved FSE readout.

- FSE\_ESP (s) : The echo spacing for FSE
- FSE\_ETL : The echo train length for FSE
- FSE\_ShotNum : The number of FSE shots, multi shot FSE uses interleave mode

#### 4. GRAPPA (:TODO)

The ‘GRAPPA’ tab contains parameters for performing parallel imaging using GRAPPA.

#### 5. Gridding

The ‘Gridding’ tab contains parameters for controlling gridding process in default Non-Cartesian reconstruction. MRiLab uses Voronoi diagram for K-Space density compensation, and uses Kaiser-Bessel kernel for gridding. Detailed explanation is beyond the scope of this manual, users who are interested are referred to [2, 3, 4].

- G\_Deapodization : The flag for turning on and off kernel deapodization (i.e. dividing reconstructed image with the iFFT of the gridding kernel)
- G\_KernelSample : The number of kernel sample point, the more sample points, the better kernel approximation
- G\_KernelWidth : The full width of kernel in the unit of gridding grid
- G\_OverGrid : The over gridding factor
- G\_Truncation : The flag for turning on and off image truncation for reconstructed image

#### 6. IRRPrep

The ‘IRRPrep’ tab contains parameters for inversion recovery sequence.

- TI (s) : The time of inversion recovery

#### 7. MT

The ‘MT’ tab contains parameters for Magnetization Transfer (MT) sequence. In order to perform MT experiment, particular MT phantom are required.

- MT\_Flag : The flag for turning on and off MT simulation

#### 8. MultiEcho

The ‘MultiEcho’ tab contains parameters for performing multi echo experiment, the number of echoes much match TEPerTR.

- ME\_TEs (s) : The array of multiple echo values

#### 9. PartialEcho (:TODO)

The ‘PartialEcho’ tab contains parameters for performing partial echo in readout.

#### 10. Radial

The ‘Radial’ tab contains parameters for performing 2D radial readout sampling.

- R\_AngPattern : The pattern for sampling the angle in K-Space
- R\_AngRange : The range of sampling angle
- R\_SampPerSpoke : The number of sampling points in each spoke
- R\_SpokeNum : The number of sampling spokes

#### 11. SENSE (:TODO)

The ‘SENSE’ tab contains parameters for performing parallel imaging using SENSE.

#### 12. Shim

The ‘Shim’ tab contains parameters for performing manual B0 shimming.

- Sh\_X : The constant for X term
- Sh\_Y : The constant for Y term
- Sh\_Z : The constant for Z term
- Sh\_ZX : The constant for ZX term
- Sh\_ZY : The constant for ZY term
- Sh\_Z2 : The constant for  $Z^2$  term
- Sh\_XYZ : The constant for XYZ term
- Sh\_X2\_Y2 : The constant for  $X^2Y^2$  term

13. Spiral

The ‘Spiral’ tab contains parameters for performing multi shot spiral read-out. The 2D spiral design uses the method described in [5].

- S\_Gradient (T/m) : The desired gradient amplitude
- S\_Lamda (1/m/rad) : A constant affecting radial sampling interval in the spiral trajectory
- S\_ShotNum : The number of spiral interleaves
- S\_SlewRate (T/m/s) : The desired slew rate. Notice that in this approximation, slew rate overshoots the desired value for part of the slew-rate-limited region
- S\_SlewRate0 (T/m/s) : The slew rate at the beginning

14. T2Prep (:TODO)

The ‘T2Prep’ tab contains parameters for T2 preparation sequence.

15. VIPR (:TODO)

The ‘VIPR’ tab contains parameters for performing Vastly Undersampled Isotropic Projection Reconstruction (VIPR) sequence.

16. ZeroFilling

The ‘ZeroFilling’ tab contains parameters for performing image interpolation in the K-Space using zero filling.

- ZF\_Kz : The zero filling factor in Kz
- ZF\_Ky : The number of point in Ky after zero filling
- ZF\_Kx : The number of point in Kx after zero filling

## 3.9 Parallel Computation

The current version of MRILab supports two types of parallel computation mechanism: GPU based parallel computation by using CUDA and multi-threaded CPU based parallel computation by using OpenMP. As mentioned before, the GPU support requires NVIDIA GPU with CUDA capability and the properly

installed GPU driver. The OpenMP is, on the other hand, less of hardware constraint and natively supported by most of modern multi core CPU. If both GPU and CPU are available in user's system, the user can choose to use any of these two methods. To switch parallel computation methods, go to 'Parallel' menu and 'Select Processing Unit' and choose available GPU or CPU.

# Chapter 4

## Simulation

### 4.1 Running Simulation

Once the simulation parameters are loaded and adjusted according to the experiment design, in order to make the changes effective, the user has to press ‘Update’ button located below ‘Simulation Settings’ panel. The user can also check the initially loaded value of each parameter by putting the mouse cursor on top of it. MRiLab converts the parameters from configuration files into temporary configuration structures during loading process, and uses these structures to categorize and organize the simulation settings. The ‘Update’ button not only updates these structures, but also performs a series of pre-scan processes including checking incompatibility error and initializing other necessary simulation variables.

On the left of ‘Update’ button, there is a ‘Save’ button which can save back the updated configuration structure into corresponding configuration files for later use. One particular case is that ‘CVs’ has to be updated first and then saved in order to make changes effective. This is because ‘CVs’ is one part of the sequence file which needs to be interpreted at the sequence signal generation module. The sequence memo is also provided at the ‘PSD Memo’ panel. It’s editable and will be saved once ‘Save’ button is pressed.

On the right of ‘Update’ button, there is a ‘Scan’ button which activates sequence signal generation, actual scan process and post-scan process including image reconstruction and data saving. The MRiLab will automatically detect any parameter changes and set ‘Scan’ button disabled. To enable ‘Scan’ button, simply press ‘Update’ button. Notice that the update process may take some time when large numbers of initialization is needed, so be patient and wait it to be finished before proceeding. The ‘Simulation Info Log’ is helpful for checking log information about each simulation step. Once simulation setting is configured properly, the user can press ‘Scan’ button to start scanning.

## 4.2 Image, SNR and SAR



Figure 4.1: The Preview of A Simulated Image. The preview is one slice of simulated image using gradient echo sequence with radial readout and reconstructed without deapodization.

Figure 4.1 demonstrates an example of a series of simulation operated with different sequences. Each sequence is labeled with an unique series number, and shows in the list at ‘Scan Series’ panel. This series number is also a reference for the saved image and data in the output database which will be introduced in the Chapter 5. There is also status label on the left of each sequence name. The status labels include:

- Dx : parameter setting and sequence loading
- ... : scanning
- V : scan complete successfully
- X : scan incomplete or fail

The example shows a series of successful simulation by using PSD\_GRE3D, PSD\_GRE3DRadial and PSD\_FIESTA3D sequences, but incomplete simulation

using PSD\_FIESTA3D at the second time. The preview axes is displaying the image preview for the series 2 simulated with PSD\_GRE3DRadial sequence. The user can switch the preview for successful simulation by clicking the scan series. In addition, the series name is also editable although in this example the series name is kept the same as the name of the sequence, which is not absolutely necessary. If the multiple channel coil for multiple receiving is performed (Figure 4.2), the user can specify displayed image to any channel with ‘Channel #’ popup menu, or choose ‘SumofMagn’ for summation of image magnitude of all channels or ‘SumofCplx’ for summation of complex image of all channels. If multiple echo is enabled in the sequence, the user can also specify displayed image to any echo with ‘Echo #’ popup menu. Default value of 1 is used for single echo. The preview image provides a quick overview of the simulated images, the further image analysis can be performed with image display and analysis tools in Chapter 6.

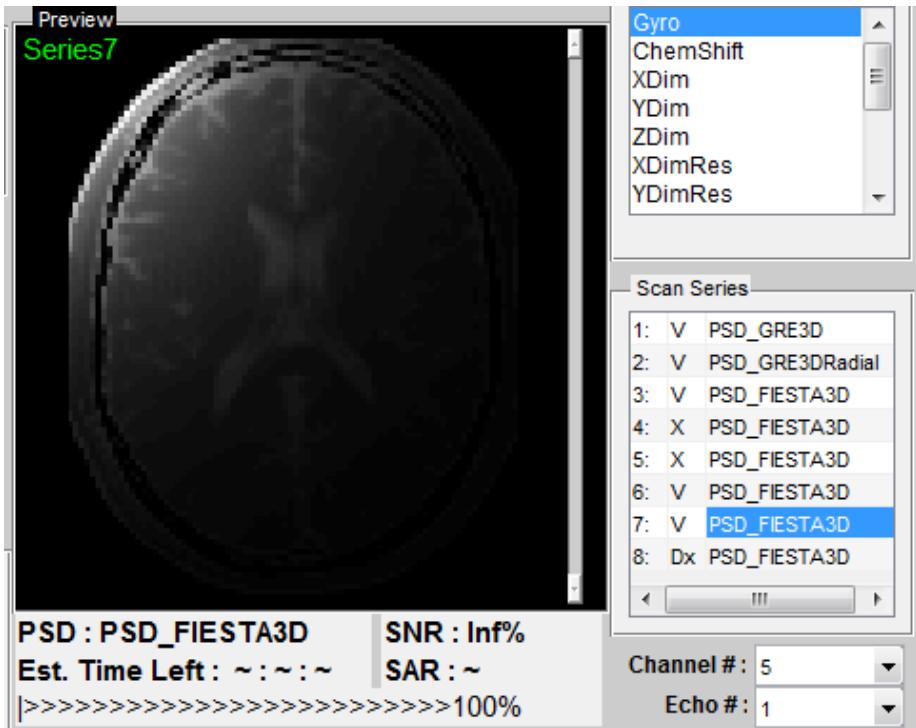


Figure 4.2: The Preview of A Simulated Image with Multiple Receiving. The preview is one slice of simulated image using FIESTA sequence with 8 channel head coil. The image is from the fifth coil channel.

Below the preview axes, there are information for the current running sequence, estimated remaining scan time in real time and the scan progress bar. The global relative SNR and SAR (:TODO) are also provided and automatic

updated in real time. The relative SNR is defined as the ratio of current SNR to the initial SNR when loading the sequence. The SNR value is calculated using Equation 4.1.

$$SNR = B0 \times RFreq \times RPhase \times RSlice \times \frac{\sqrt{ResFreq \times ResPhase \times SliceNum \times NEX}}{NoiseLevel \times \sqrt{BandWidth}} \quad (4.1)$$

where

$$\begin{aligned} RFreq &= \frac{FOVFreq}{ResFreq}; \\ RPhase &= \frac{FOVPhase}{ResPhase}; \\ RSlice &= SliceThick. \end{aligned} \quad (4.2)$$

In order to simulate image noise, MRiLab performs the noise adding process after the signal acquisition in K-Space. The Gaussian noise with zero mean and certain standard deviation is added to the complex signal. The standard deviation is determined using Equation 4.3. If no noise is desired, the user can set ‘NoiseLevel’ to zero to get infinite SNR.

$$Noise = \frac{\frac{NoiseLevel}{NoiseRef} \times \sqrt{\frac{BandWidth}{BWRef}}}{\frac{B0}{B0Ref} \times \frac{RFreq \times RPhase \times RSlice}{VolRef} \times \sqrt{\frac{NEX}{NEXRef} \times \frac{ResFreq \times ResPhase \times SliceNum}{ADCRef}}} \quad (4.3)$$

where these reference values are given as:

$$\begin{aligned} BWRef &= 1e3; \\ NoiseRef &= 1; \\ B0Ref &= 1.5; \\ VolRef &= 1e - 9; \\ NEXRef &= 1; \\ ADCRef &= 1e4. \end{aligned} \quad (4.4)$$

# Chapter 5

## MRiLab Toolboxes

MRiLab toolboxes consists of several separate GUIs for conducting RF pulse design, MR sequence design and Coil design etc. These toolboxes provided by MRiLab enable the user to customize and build their own specific MR experiment. This chapter covers the introduction for each toolbox and its associated macro libraries.

### 5.1 RF Pulse Design

The RF pulse Design toolbox can be activated by pressing ‘RF Design Panel’ toolbar icon located at the top of the main simulation console.



Figure 5.1: RF Design Panel Toolbar Icon

#### 5.1.1 RF Design GUI

Figure 5.2 demonstrates the overview of the RF Pulse Design interface. This interface consists of

1. RF and Gradient Macro Library

The user can use this interface to analyze the slice profile for the chosen RF pulse. To select a RF pulse macro, the user needs to click the macro library tree to unfold the tree structure and then click the desired RF macro. The properties of the chosen RF macro will show up on the left panel under ‘rf:rf name’ tab. The RF memo information is also shown at the ‘rf Memo’ panel below the tree structure. The user can press ‘Execute’ button to start calculation. MRiLab assumes a gradient is applied

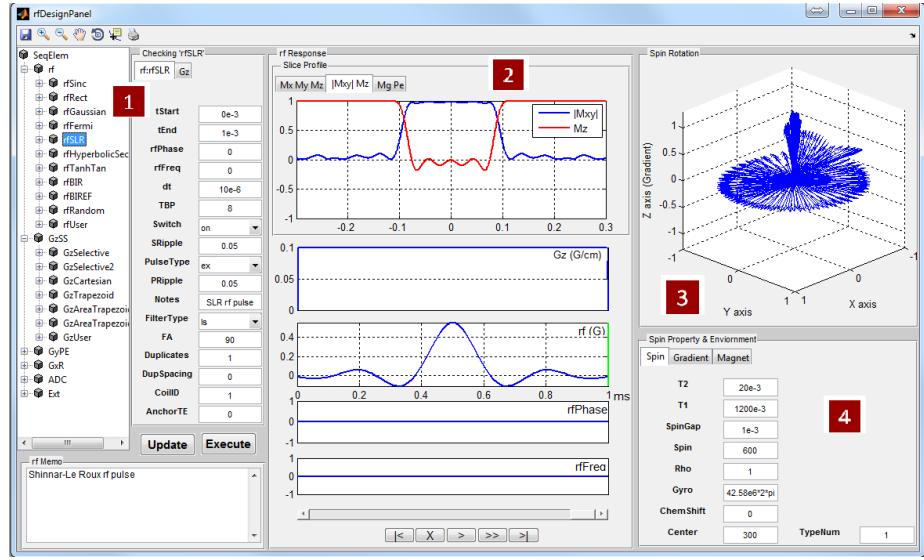


Figure 5.2: RF Design GUI. An example of one Shinnar-Le Roux RF pulse and its slice profile is shown.

in the Z direction, therefore a constant gradient will be applied if ‘Gz’ tab is empty (Figure 5.3). To select a ‘Gz’ gradient, the user needs to choose the gradient macro under ‘GzSS’. For example the ‘GzSelective’ is a recommended gradient macro that is normally used in MRiLab for performing slice selection. Once the user selected a gradient macro, the ‘Gz:gradient name’ tab will be activated and the properties of the gradient macro is accessible and editable. The user can modify macro attributes to meet their own needs. To make the modification effective, the user must press ‘Update’ button before executing the slice selection profile analysis. Although the library tree contains macros for another gradient line (e.g. GyPE), they are typically inaccessible here.

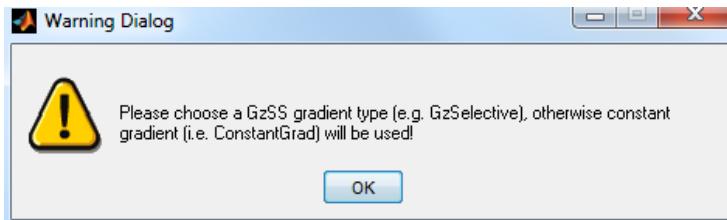


Figure 5.3: A Warning Window for Using Constant Gradient

## 2. Slice Profile

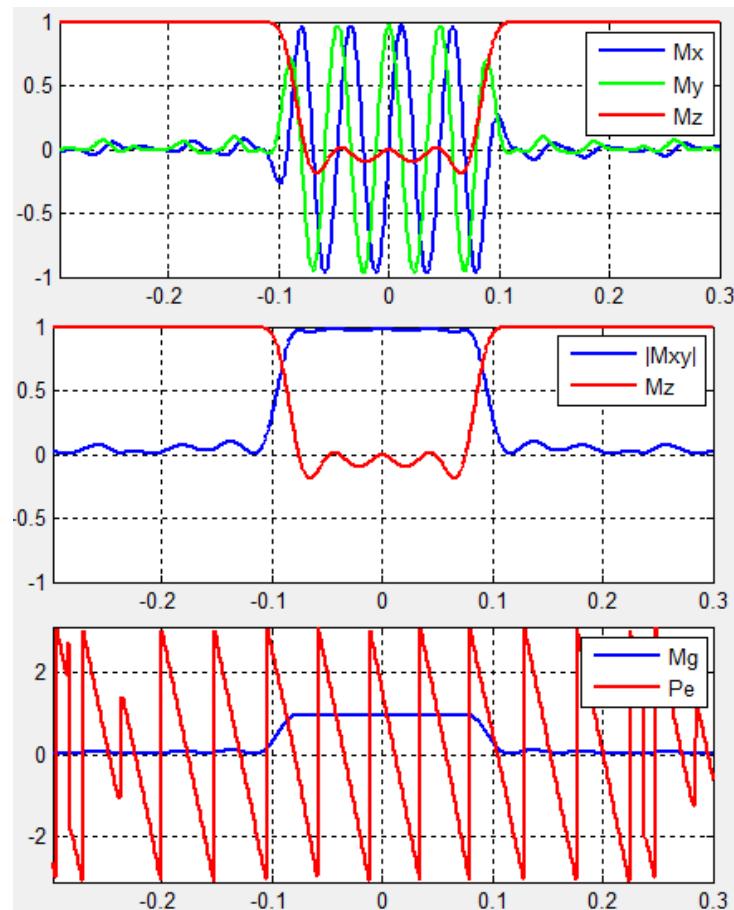


Figure 5.4: The Slice Profile for A SLR Pulse

MRiLab provides three slice profile figures (Figure 5.4) on the ‘Slice Profile’ panel under different tabs. These figures include slice profile in terms of

- $M_x M_y M_z$  : three independent spin component
- $|M_{xy}| M_z$  : transverse and longitudinal component
- $M_g P_e$  : transverse component magnitude and phase

The horizontal axis is the spin position in meters, and the vertical axis is the value of the components in normalized units. MRiLab also plots the RF and gradient waveform below the slice profile. In MRiLab, the RF pulse is defined using RF amplitude (T), RF phase (rad) and RF frequency (Hz). The RF frequency is defined as the spin Larmor frequency minus the laboratory frequency of the RF pulse. Notice that in the figures, the gradient amplitude is in G/cm and RF amplitude is in G. Both time axes are in units of milliseconds. At the bottom of this interface, there is a group of buttons (Figure 5.5) that enable the user to investigate the intermediate slice profile during the applied RF and gradient (Figure 5.6).



Figure 5.5: The Playback Control Group for Slice Profile

- Scroll Bar : Drag the scroll bar to any intermediate time point between beginning and end
- | < : Move to the beginning
- X : Pause playing, notice that the interface can only be closed when playing is paused
- O : Resume playing
- > : Play at normal speed
- >> : Play at double normal speed
- > | : Move to the end

### 3. Spin Response

MRiLab also presents the spin response in ‘Spin Rotation’ panel as a three dimensional representation of the slice profile. The user can inspect the behavior of spins at specific location with the chosen RF pulse and gradient. Notice that the Z axis is applied with the gradient and is in units of meters. The X-axis and Y-axis are in normalized units. The user can use Matlab default graphic tools for interactively changing display view and size.

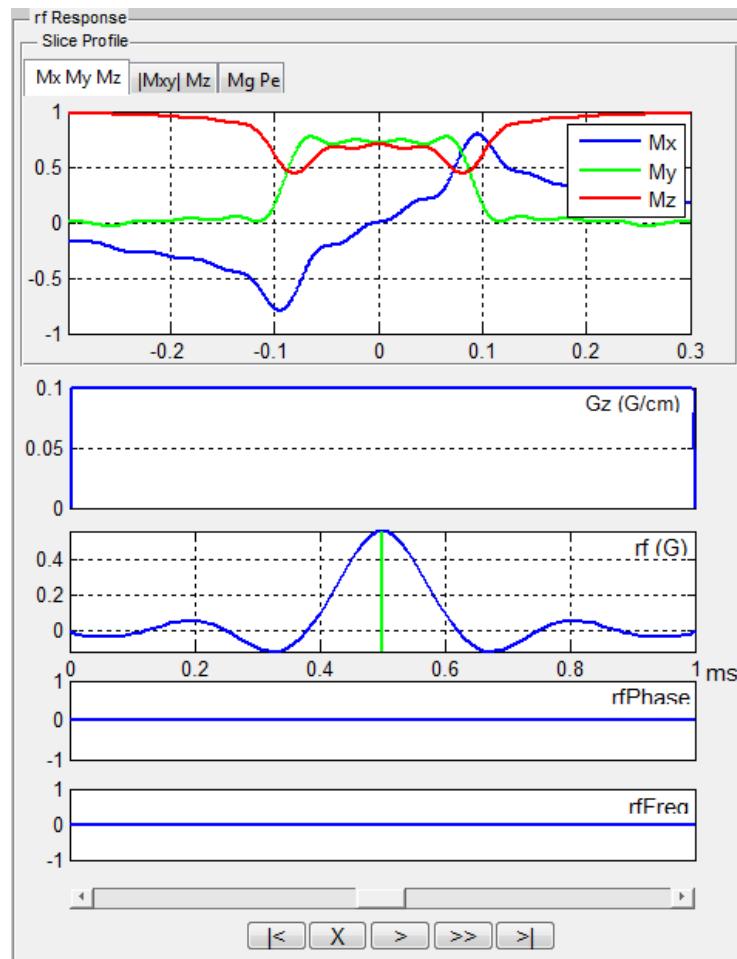


Figure 5.6: An Intermediate Slice Profile in The Middle of RF pulse



Figure 5.7: Matlab Default Graphical Tools

#### 4. Spin Property and Environment

The user can modify the spin properties and environment to meet their own needs. The editable properties provided in this interface include:

##### (a) Spin

- Center : The index of the center spin
- ChemShift (Hz/T): The chemical shift of the spin
- Gyro (rad/s/T): The gyromagnetic ratio of the spin
- Rho : The density of the spin
- Spin : The number of the spins
- SpinGap (m): The distance between adjacent spins
- T1 (s): The longitudinal relaxation time
- T2 (s): The transverse relaxation time
- TypeNum : The number of spin species

##### (b) Gradient

- ConstantGrad (T/m): The constant gradient applied when ‘Gz’ tab is empty

##### (c) Magnet

- dB0 (T): The main magnetic field offset

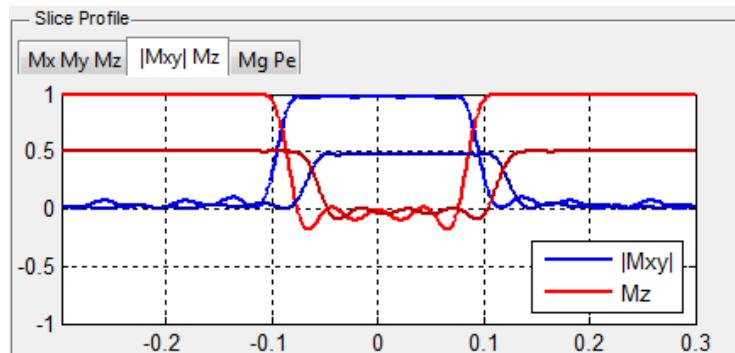


Figure 5.8: A Slice Profiles for Two Spin Species

Figure 5.8 demonstrates a slice profile for two different spin species at the same RF pulse and gradient. To enable slice profile analysis for multiple spin species, the user needs to provide multiple values for T1, T2, Rho and ChemShift, and give the correct number of spin species. The values must be separated with space. For example

- ChemShift = 0 -210
- Rho = 1.0 0.5
- T1 = 1.2 1.0
- T2 = 0.02 0.03
- TypeNum = 2

### 5.1.2 RF Macro Library

MRiLab uses the concept of macros for simplifying experiment design. A RF macro is a predefined module for a RF pulse. The macro is programming-free and the attributes of the macro can be easily adjusted using the RF Design interface. MRiLab RF macro library is a collection of RF macros covering from simple RF pulses such as hard pulse, to complex RF pulses such as those adiabatic pulses. There is also specific RF macro provided to interact with external RF pulse file to create more extensible pulse design environment. This section will give an introduction for each of the RF macros provided in MRiLab.

#### **rfSinc**

A RF macro that creates a Sinc type RF pulse. This macro contains attributes including:

- Apod : Apodization methods including ‘Non’, ‘Hamming’ and ‘Hanning’
- FA (Degree) : Prescribed flip angle
- TBP : The time bandwidth product
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting

- Notes : The notes of the RF pulse

The time dependence of the Sinc RF pulse is given by [6]:

$$B_1(t) = \begin{cases} At_0 \frac{\sin(\frac{\pi t}{t_0})}{\frac{\pi t}{t_0}} & -N_L t_0 \leq t \leq N_R t_0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.1)$$

where  $A$  is the peak RF amplitude automatically calculated and scaled according to flip angle,  $t_0$  is one-half the width of the central lobe, and the  $N_L$  and  $N_R$  are the number of zero-crossings to the left and right of the central peak, respectively. In MRiLab, the  $N_L \equiv N_R$ , thus the Sinc RF pulse is always symmetric. Notice that the The time bandwidth product of a Sinc pulse equals the number of zero-crossings including the start and end. In order to address the discontinuity at the start and end, apodization can be applied using ‘Hamming’ or ‘Hanning’ window as described by:

$$\text{Apodization}(t) = \begin{cases} (1 - \alpha) + \alpha \cos(\frac{\pi t}{N t_0}) & -N_L t_0 \leq t \leq N_R t_0 \\ 0 & \text{elsewhere} \end{cases} \quad (5.2)$$

where  $N$  equals  $N_L$  and  $N_R$ . Hamming window uses  $\alpha = 0.46$ , and Hanning window uses  $\alpha = 0.5$ . If ‘Non’ is used, apodization is disabled.

### **rfRect**

A RF macro that creates a hard RF pulse. This macro contains attributes including:

- FA (Degree) : Prescribed flip angle
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates

- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the hard RF pulse is given by [6]:

$$B_1(t) = \begin{cases} A & |t| \leq \frac{T}{2} \\ 0 & |t| > \frac{T}{2} \end{cases} \quad (5.3)$$

where  $A$  is the peak RF amplitude automatically calculated and scaled according to flip angle,  $T$  is the width of RF pulse that equals tEnd-tStart.

### rfGaussian

A RF macro that creates a Gaussian type RF pulse. This macro contains attributes including:

- FA (Degree) : Prescribed flip angle
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the Gaussian RF pulse is given by [6]:

$$B_1(t) = Ae^{-\frac{t^2}{2\sigma^2}} \quad \text{pulse centered at } t = 0 \quad (5.4)$$

where  $A$  is the peak RF amplitude automatically calculated and scaled according to flip angle,  $\sigma$  is linearly proportional to the pulse width. Also the Gaussian RF pulse is terminated with a 60-dB attenuation.

**rfFermi**

A RF macro that creates a Fermi RF pulse. This macro contains attributes including:

- PW : The measure of the pulse width
- FA (Degree) : Prescribed flip angle
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the Fermi RF pulse is given by [6]:

$$B_1(t) = \frac{A}{1 + e^{\frac{|t| - t_0}{\alpha}}} \quad \text{pulse centered at } t = 0 \quad (5.5)$$

where  $A$  is the peak RF amplitude automatically calculated and scaled according to flip angle,  $t_0$  is a measure of the pulse width that corresponds to PW,  $\alpha$  is a measure of the transition width. The Fermi pulse approximates more to a rectangle pulse with larger  $t_0$  value. Also the Fermi RF pulse is terminated with a 60-dB attenuation.

**rfSLR**

A RF macro that creates a RF pulse using Shinnar-Le Roux algorithm. This macro contains attributes including:

- PulseType : The type of this SLR pulse, including ‘st’ (small tip angle pulse), ‘ex’ (excitation pulse), ‘se’ (spin-echo pulse), ‘sat’ (saturation pulse) and ‘inv’ (inversion pulse)

- FilterType : The type of the applied filter design method, including ‘ls’ (least squares), ‘min’ (minimum phase), ‘max’ (maximum phase), ‘pm’ (Parks-McClellan equal ripple), and ‘ms’ (Hamming windowed sinc)
- PRipple : The ripple factor at passband
- SRipple : The ripple factor at stopband
- FA (Degree) : Prescribed flip angle
- TBP : The time bandwidth product
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

MRiLab implements a library of Matlab routines for designing SLR algorithm that is developed originally by John Pauly and published online at <http://rsl.stanford.edu/research/software.html>. Thorough explanation of the algorithm is beyond the scope of this manual, users who are interested in the SLR algorithm are referred to [6, 7].

### **rfHyperbolicSecant**

A RF macro that creates an adiabatic inversion RF pulse based on hyperbolic secant modulation. This macro contains attributes including:

- Adiab : The adiabatic factor
- MaxB1 (T) : The maximum B1 field

- TBP : The time bandwidth product
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the hyperbolic secant RF pulse is given by [6]:

$$\begin{aligned} A(t) &= A_0 \operatorname{sech}(\beta t) \quad \text{amplitude modulation} \\ F(t) &= \frac{-\mu\beta}{2\pi} \tanh(\beta t) \quad \text{frequency modulation} \end{aligned} \tag{5.6}$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $\mu$  is a dimensionless adiabatic factor corresponding to Adiab,  $\beta$  is an modulation angular frequency. It can be shown that TBP has the relationship with  $\beta$  and  $\mu$  as

$$TBP = \frac{T\mu\beta}{\pi} \tag{5.7}$$

where  $T$  is the pulse width.

To satisfy the Adiabatic Condition, the parameter setting has to meet

$$A_0 \gg \frac{\sqrt{\mu\beta}}{\gamma} \tag{5.8}$$

### rfTanhTan

A RF macro that creates an adiabatic inversion RF pulse based on tanh/tan modulation. This macro contains attributes including:

- MaxB1 (T) : The maximum B1 field
- TBP : The time bandwidth product

- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The tanh/tan RF pulse is constructed from an adiabatic half passage and its time-reversed adiabatic half passage. The time dependence of the first adiabatic half passage is given by [8]:

$$\begin{aligned} A(t) &= \gamma A_0 \tanh\left(\frac{2\xi t}{T}\right) \quad 0 \leq t \leq \frac{T}{2} \quad \text{amplitude modulation} \\ F(t) &= A \frac{\tan(\kappa(1 - \frac{2t}{T}))}{2\pi \tan(\kappa)} \quad 0 \leq t \leq \frac{T}{2} \quad \text{frequency modulation} \end{aligned} \quad (5.9)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $\xi = 10$ ,  $\tan(\kappa) = 20$ ,  $T$  is the pulse width. The TBP can be estimated using

$$TBP = \frac{A \cdot T}{\pi} \quad (5.10)$$

### rfBIR

A RF macro that creates an adiabatic B1 Independent Rotation (BIR) RF pulse. This macro contains attributes including:

- MaxB1 (T) : The maximum B1 field
- MaxFreq (Hz) : The maximum RF frequency
- Lambda : The  $\lambda$  adiabatic factor
- Beta : The  $\beta$  adiabatic factor

- BIRFlag : The type of BIR pulse, including ‘BIR-1’, ‘BIR-2’ and ‘BIR-4’
- dt (s) : The time interval of RF pulse sample points
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the BIR-1 RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \begin{cases} \hat{x}A_0\cos(\xi t) & 0 \leq t < \frac{T}{2} \\ \hat{y}A_0\cos(\xi t) & \frac{T}{2} \leq t \leq T \end{cases} \quad (5.11)$$

Frequency modulation:

$$F(t) = \begin{cases} F_0\sin(\xi t) & 0 \leq t < \frac{T}{2} \\ -F_0\sin(\xi t) & \frac{T}{2} \leq t \leq T \end{cases} \quad (5.12)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $F_0$  is the maximum RF frequency corresponding to MaxFreq. The RF pulse width is  $T = \frac{\pi}{\xi}$ , and  $\hat{x}$  and  $\hat{y}$  are unit vectors for indicating RF phase.

The time dependence of the BIR-2 RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \begin{cases} \hat{x}|A_0\cos(\xi t)| & 0 \leq t < \frac{T}{2} \\ \hat{y}|A_0\cos(\xi t)| & \frac{T}{2} \leq t < T \\ -\hat{y}|A_0\cos(\xi t)| & T \leq t \leq 2T \end{cases} \quad (5.13)$$

Frequency modulation:

$$F(t) = |F_0\sin(\xi t)| \quad (5.14)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $F_0$  is the maximum RF frequency corresponding to MaxFreq. The RF pulse width is  $T = \frac{\pi}{\xi}$ , and  $\hat{x}$  and  $\hat{y}$  are unit vectors for indicating RF phase.

The time dependence of the BIR-4 RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \begin{cases} A_0 \tanh[\lambda(1 - \frac{4t}{T})] & 0 \leq t < \frac{T}{4} \\ A_0 \tanh[\lambda(\frac{4t}{T} - 1)] & \frac{T}{4} \leq t < \frac{T}{2} \\ A_0 \tanh[\lambda(3 - \frac{4t}{T})] & \frac{T}{2} \leq t < \frac{3T}{4} \\ A_0 \tanh[\lambda(\frac{4t}{T} - 3)] & \frac{3T}{4} \leq t \leq T \end{cases} \quad (5.15)$$

Frequency modulation:

$$F(t) = \begin{cases} \frac{\tan(\frac{4\beta t}{T})}{\tan(\beta)} & 0 \leq t < \frac{T}{4} \\ \frac{\tan[\beta(\frac{4t}{T} - 2)]}{\tan(\beta)} & \frac{T}{4} \leq t < \frac{T}{2} \\ \frac{\tan[\beta(\frac{4t}{T} - 2)]}{\tan(\beta)} & \frac{T}{2} \leq t < \frac{3T}{4} \\ \frac{\tan[\beta(\frac{4t}{T} - 4)]}{\tan(\beta)} & \frac{3T}{4} \leq t \leq T \end{cases} \quad (5.16)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1. The RF pulse width is  $T$ . The  $\lambda$  and  $\beta$  are dimensionless constants that describe the degree to which extent the RF pulse satisfies the adiabatic condition.

### rfBIREF

A RF macro that creates an adiabatic B1 Independent Refocusing (BIREF) RF pulse. This macro contains attributes including:

- MaxB1 (T) : The maximum B1 field
- MaxFreq (Hz) : The maximum RF frequency
- BIREFFlag : The type of BIREF pulse, including ‘BIREF-1’, ‘BIREF-2a’ and ‘BIREF-2b’
- dt (s) : The time interval of RF pulse sample points
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape

- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

The time dependence of the BIREF-1 RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \begin{cases} \hat{x}A_0\sin(\xi t) & 0 \leq t < \frac{T}{2} \\ -\hat{x}A_0\sin(\xi t) & \frac{T}{2} \leq t \leq T \end{cases} \quad (5.17)$$

Frequency modulation:

$$F(t) = F_0|\cos(\xi t)| \quad (5.18)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $F_0$  is the maximum RF frequency corresponding to MaxFreq. The RF pulse width is  $T = \frac{\pi}{\xi}$ , and  $\hat{x}$  is an unit vector for indicating RF phase along the  $x$  axis.

The time dependence of the BIREF-2a RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \hat{x}A_0|\cos(\xi t)| \quad (5.19)$$

Frequency modulation:

$$F(t) = \begin{cases} F_0\sin(\xi t) & 0 \leq t < \frac{T}{2} \\ -F_0\sin(\xi t) & \frac{T}{2} \leq t \leq T \end{cases} \quad (5.20)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $F_0$  is the maximum RF frequency corresponding to MaxFreq. The RF pulse width is  $T = \frac{\pi}{\xi}$ , and  $\hat{x}$  is an unit vector for indicating RF phase along the  $x$  axis.

The time dependence of the BIREF-2b RF pulse is given by [6]:  
Amplitude modulation:

$$A(t) = \begin{cases} \hat{x}A_0|\cos(\xi t)| & 0 \leq t < \frac{T}{2} \\ -\hat{x}A_0\cos(\xi t) & \frac{T}{2} \leq t \leq T \end{cases} \quad (5.21)$$

Frequency modulation:

$$F(t) = \begin{cases} F_0\sin(\xi t) & 0 \leq t < \frac{T}{4} \\ -F_0\sin(\xi t) & \frac{T}{4} \leq t < \frac{3T}{4} \\ F_0\sin(\xi t) & \frac{3T}{4} \leq t \leq T \end{cases} \quad (5.22)$$

where  $A_0$  is the maximum B1 field corresponding to MaxB1,  $F_0$  is the maximum RF frequency corresponding to MaxFreq. The RF pulse width is  $T = \frac{2\pi}{\xi}$ , and  $\hat{x}$  is an unit vector for indicating RF phase along the  $x$  axis.

### **rfRandom**

A RF macro that creates a RF pulse with normally distributed pseudo-random amplitude. This macro is used for program testing purpose, however it shows that almost any arbitrary RF pulse is supported by MRiLab. This macro contains attributes including:

- rfGain : The standard deviation of the normal distribution
- dt (s) : The time interval of RF pulse sample points
- rfPhase (rad) : RF pulse phase
- rfFreq (Hz) : RF pulse frequency
- tStart (s) : RF pulse starting time
- tEnd (s) : RF pulse ending time
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

### **rfUser**

If the user has the RF pulse data saved in a MAT file, the user can easily import the RF file into MRiLab pulse design interface by using ‘rfUser’ macro. The RF pulse MAT file needs to contain four matrices including ‘rfTime’ (i.e. RF time points) , ‘rfAmp’ (i.e. RF amplitude), ‘rfPhase’ (i.e. RF phase) and ‘rfFreq’ (i.e. RF frequency). All four matrices must have the same size as m-by-n, where m is the number of TR sections and n is the number of RF waveform points. In typical MR sequence, the entire sequence is composed of multiple TR sections. The *i*th TR section uses the *i*th RF pulse stored in the *i*th row of these four matrices. If the number of row is less than the number of TR sections, the last RF pulse will be used for all the remaining TR sections. Notice that if ‘rfPhase’ and/or ‘rfFreq’ are not provided, MRiLab initializes them as the value of 0. However, ‘rfTime’ and ‘rfAmp’ must be provided. MRiLab only uses the first RF pulse in the MAT file for analysis in the RF pulse design interface. The ‘rfUser’ macro contains attributes including:

- rfFile : The path to the file that stores the RF pulse data, quoted using single quotes
- Switch : The flag for turning on and off RF pulse in the RF sequence line
- AnchorTE : The flag for turning on and off TE reference, TE is calculated from this RF pulse if this flag is turned on
- Duplicates : The number of the RF pulse duplicates, used for creating multiple RF pulses with the same shape
- DupSpacing : The time spacing between RF pulse duplicates
- CoilID : The ID of the coil element used with this RF pulse, applied for multiple RF transmitting
- Notes : The notes of the RF pulse

### 5.1.3 Make Your Own RF Macro

The RF pulse macro library covers several common types of RF pulse waveform in current MRI field. However, due to the extremely enriched novel RF pulse types that are under development, comprehensive coverage of those RF pulses is nearly impossible for almost any pulse sequence design tools. To somehow address this problem in MRiLab, the user can use the ‘rfUser’ to import RF pulses from files that are generated by other programs. Another way to import RF pulse is to simply write a RF macro. To create a RF macro, the user should follow the following steps :

1. Write RF macro code

It is strongly recommended to write your own RF macro code based on the closest RF macros in the MRiLab macro library, for example, the ‘rfRect’ macro is coded as:

```
function [rfAmp,rfPhase,rfFreq,rfCoil,rfTime]=rfRect(p)
%Create a hard RF pulse starting from tStart and ending at tEnd
%tStart RF start time
%tEnd RF end time
%FA RF actual flip angle
%dt RF sample time
%rfPhase RF phase
%rfFreq RF off-res freq

tStart=p.tStart;
tEnd=p.tEnd;
FA=p.FA;
dt=p.dt;
```

```

rfPhase=p.rfPhase;
rfFreq=p.rfFreq;
rfCoil=p.CoilID;
Duplicates=max(1,p.Duplicates);
DupSpacing=max(0,p.DupSpacing);

rfTime=linspace(tStart,tEnd,ceil((tEnd-tStart)/dt)+1);
rfAmp=ones(size(rfTime)); % Rectangle
rfAmp(1)=0;
rfAmp(end)=0;
rfAmp=DoB1Scaling(rfAmp,dt,FA)*rfAmp; %B1 Scaling

rfPhase=(rfPhase)*ones(size(rfTime));
rfFreq=(rfFreq)*ones(size(rfTime));
rfCoil=(rfCoil)*ones(size(rfTime));
rfPhase(1)=0;
rfPhase(end)=0;
rfFreq(1)=0;
rfFreq(end)=0;

% Create Duplicates
if Duplicates~=1 & DupSpacing ~=0
    rfAmp=repmat(rfAmp,[1 Duplicates]);
    rfFreq=repmat(rfFreq,[1 Duplicates]);
    rfPhase=repmat(rfPhase,[1 Duplicates]);
    rfCoil=repmat(rfCoil,[1 Duplicates]);
    TimeOffset = repmat(0:DupSpacing:(Duplicates-1)*DupSpacing, ...
        [length(rfTime) 1]);
    rfTime=repmat(rfTime,[1 Duplicates]) + (TimeOffset(:));
end

end

```

Your macro must start from the function declaration at the beginning, then followed by attribute input section. The ‘tStart’ and ‘tEnd’ need to be added for indicating the time scale. It’s also strongly recommended to add attribute input ‘rfCoil’,‘Duplicates’ and ‘DupSpacing’ for multi-transmitting and multi-echo support.

```

function [rfAmp,rfPhase,rfFreq,rfCoil,rfTime]=rfMacroName(p)
%Create a RF pulse based on your code

tStart=p.tStart;
tEnd=p.tEnd;
rfCoil=p.CoilID;
Duplicates=max(1,p.Duplicates);

```

```
DupSpacing=max(0,p.DupSpacing);
...
attribute1=p.attribute1;
attribute2=p.attribute2;
attribute3=p.attribute3;
...
```

The main code should deal with calculation for ‘rfAmp’, ‘rfPhase’, ‘rfFreq’ and ‘rfTime’. Notice that they all should have the same size as 1-by-m where m is the number of RF waveform points.

```
% The main code for your macro
...
rfTime = ...;
rfAmp = ...;
rfPhase = ...;
rfFreq = ...;
...
```

Then you should add several lines of code to end your macro,

```
% Avoid baseline offset
rfAmp(1)=0;
rfAmp(end)=0;
rfPhase(1)=0;
rfPhase(end)=0;
rfFreq(1)=0;
rfFreq(end)=0;

% Assign coil element index number
rfCoil=(rfCoil)*ones(size(rfTime));

% Create Duplicates
if Duplicates~=1 & DupSpacing ~=0
    rfAmp=repmat(rfAmp,[1 Duplicates]);
    rfFreq=repmat(rfFreq,[1 Duplicates]);
    rfPhase=repmat(rfPhase,[1 Duplicates]);
    rfCoil=repmat(rfCoil,[1 Duplicates]);
    TimeOffset=repmat(0:DupSpacing:(Duplicates-1)*DupSpacing, ...
                      [length(rfTime) 1]);
    rfTime=repmat(rfTime,[1 Duplicates]) + (TimeOffset(:))';
end
```

## 2. Register RF macro

The RF macro file can be put anywhere in the computer as long as the file is included in Matlab searching path, however it is recommended to put the file under /SeqElem/rf for consistent file organization. Besides the RF macro file that performs the main function, MRiLab also requires a memo .txt file that accompanies the RF macro with the name ‘rfMacroName\_Memo’. This file contains information about RF pulse description if necessary.

The customized RF macro needs to be registered in the macro library before using. To register the macro, the user need to open file ‘SeqElem.xml’ under /SeqElem, then add one entry under <rf> category with the proper attribute list. One example could be

```
<rfMacroName AnchorTE="$2'on','off'"  
CoilID="1"  
DupSpacing="0"  
Duplicates="1"  
Switch="$1'on','off'"  
tEnd="1e-3"  
tStart="0"  
Notes="A new RF macro"  
attribute1="0"  
attribute2="0"  
attribute3="0" />
```

Notice that in the above example, the first 7 attributes are required for MRiLab, The remaining attributes are optional based on the macro itself.

Once the RF macro is coded and registered to the library, the user can use this customized RF macro just like those default RF macros in the library.

## 5.2 MR Sequence Design

The MR Sequence Design toolbox can be activated by pressing ‘Sequence Design Panel’ toolbar icon located at the top of the main simulation console. The loaded sequence will show in the MR Sequence Design interface.



Figure 5.9: Sequence Design Panel Toolbar Icon

### 5.2.1 Sequence Design GUI

Figure 5.10 demonstrates the overview of the MR Sequence Design interface. This interface consists of

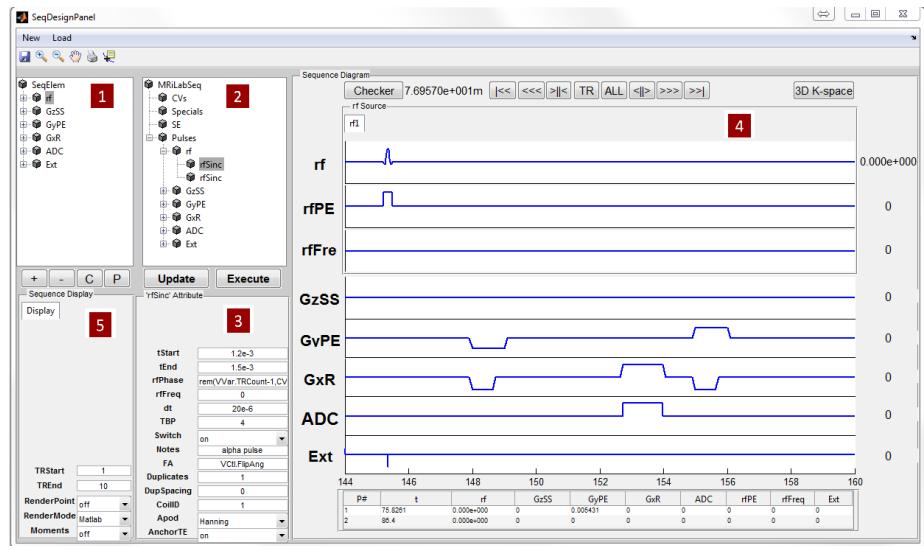


Figure 5.10: Sequence Design GUI. A bSSFP sequence is shown.

## 1. Macro Library

The MRiLab Macro Library contains a full set of pulse macros for constructing MR sequence in MRiLab. It covers not only RF macro library as described in above section, but also GzSS, GyPE and GxR gradient macro library, ADC macro library and Ext macro library. The latter libraries will be introduced in the following sections in this chapter. The user needs to click the ‘SeqElem’ root and then the subsequent nodes to unfold these macros.

## 2. Sequence Structure

In MRiLab, a MR sequence consists of the following parts :

- CVs : The controllable variables, linked to the ‘CVs’ tab on the main control console
- Specials : The applied special techniques, determining the default loading special techniques
- SE : The starting (tS) and ending (tE) time, determining time scale for each TR section, support varying TR value
- Pulse
  - RF : RF sequence line
  - GzSS : GzSS sequence line
  - GyPE : GyPE sequence line

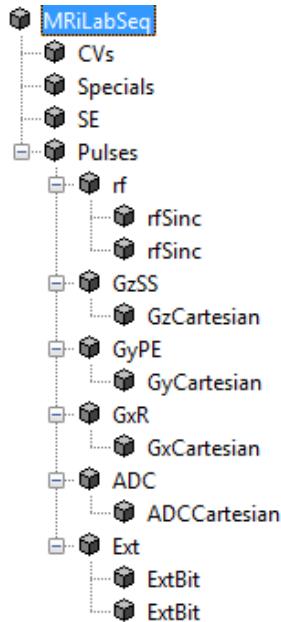


Figure 5.11: An Example of A Typical MR Sequence Structure in MRiLab

- GxR : GxR sequence line
- ADC : Signal acquisition sequence line
- Ext : Extended process sequence line

The user can construct wanted MR sequence by changing the content within the MR sequence structure. To add a macro into the sequence structure, the user needs to click one macro in the macro library, then click on the sequence line root (e.g. rf) to which this macro is inserted, then click ‘+’ macro operation button. To delete a macro from the sequence structure, the user needs to click the unwanted macro at the sequence line, then click ‘-’ macro operation button. To duplicate an existing macro, the user needs to first click the source macro, then click ‘C’ macro operation button for copying, click on the sequence line root, then click ‘P’ macro operation button for pasting. MRiLab requires that the pulse macro is operated within its belonging category for proper configuration. And MRiLab doesn’t allow empty sequence lines.

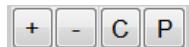


Figure 5.12: Macro Operation Buttons

### 3. Pulse Attribute

When clicking on the pulse macro within the MR sequence structure, the corresponding macro attributes will be updated at the pulse attribute panel down below the sequence structure. The user can edit those attributes to modify the MR sequence waveform. To make the modification effective, the user must press ‘Update’ button to update the sequence file. Pressing ‘Execute’ button will update and redraw the MR sequence waveform on this interface.

#### 4. Sequence Waveform

The sequence waveform associated with the sequence structure is displayed on the ‘Sequence Diagram’ panel on the right side of this interface. The user can use the waveform diagram to inspect sequence details and layout. The sequence diagram consists of individual sequence lines corresponding to RF, GzSS, GyPE, GxR, ADC and Ext. To accommodate multiple RF transmitting, MRiLab provides separate RF sequence lines for each RF source. When ‘MultiTransmit’ flag is turned on in the main control console and the chosen sequence structure contains multiple RF pulses for different RF sources (i.e. assign RF pulses to different coil channels by using ‘CoilID’ attribute), the multi-tab will be activated on the ‘rf Source’ panel (Figure 5.13). The user can switch between these tabs for checking individual RF source.

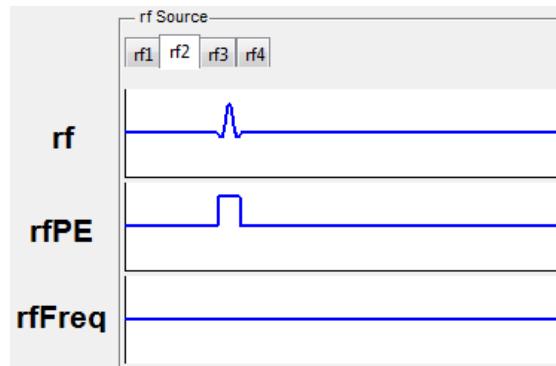


Figure 5.13: An Example of Multiple RF Source. The ‘MultiTransmit’ is turned on and this sequence contains total 4 RF sources. The RF source 2 is chosen and the corresponding RF pulse waveform for coil channel 2 is shown.

Notice that the vertical axes for all sequence lines are normalized and the horizontal axes are in units of milliseconds. MRiLab provides a group of sequence display button (Figure 5.14) to help inspecting the sequence waveform details.

The sequence display button group consists of :



Figure 5.14: The Sequence Display Button.

- Checker : The time checker toggle button
- Time Ruler : Display current time point according to the time checker
- | << : Move sequence waveform to the beginning
- <<< : Move sequence waveform backwards
- > || < : Zoom out
- TR : Display a sequence waveform section with a time interval of TR
- ALL : Display all sequence waveform
- < || > : Zoom in
- >>> : Move sequence waveform forwards
- >> | : Move sequence waveform to the end

To display a sequence waveform with any arbitrary time interval (Figure 5.15), the user can use the ‘Checker’ toggle button. First press the ‘Checker’ button, move the mouse cursor into the axes. Notice that the mouse cursor changes to a cross-hair. Move the cross-hair in the axes, the amplitude value for each sequence line will be displayed accordingly on right side of each line with their default units. The user can click on the axes for choosing one side of the time slot, then click on the another side. MRILab will change the sequence view between the chosen time points, and also save time point information in the list at the bottom of this interface. To disable ‘Checker’ function, simple press this button again.

##### 5. Display Control

The ‘Display’ tab on the ‘Sequence Display’ contains parameters for controlling sequence display and K-Space rendering.

- TRStart : The first TR to be displayed
- TREnd : The last TR to be displayed
- Moments (:TODO) : The flag for turning on and off the zeroth moment display for the gradient
- RenderMode : The K-Space rendering mode, including ‘Matlab’ (Figure 5.16) and ‘VTK’ (Figure 5.17)
- RenderPoint : The flag for turning on and off K-Space point rendering

Notice that the K-Space line color starts from green and ends to red in VTK rendering. If the user uses VTK for K-Space rendering, please press

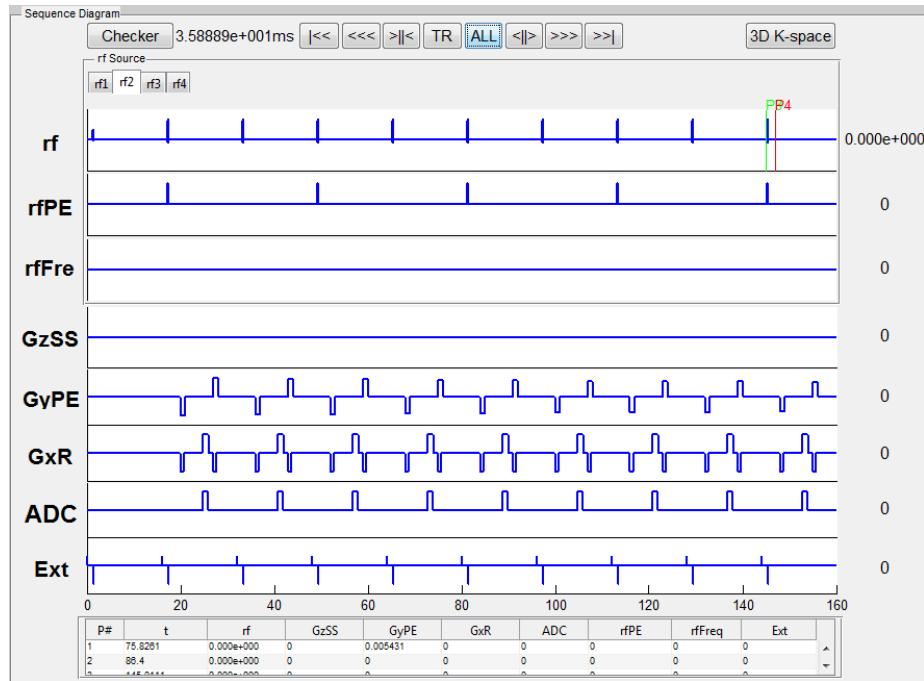


Figure 5.15: A Sequence View with Multiple TRs.

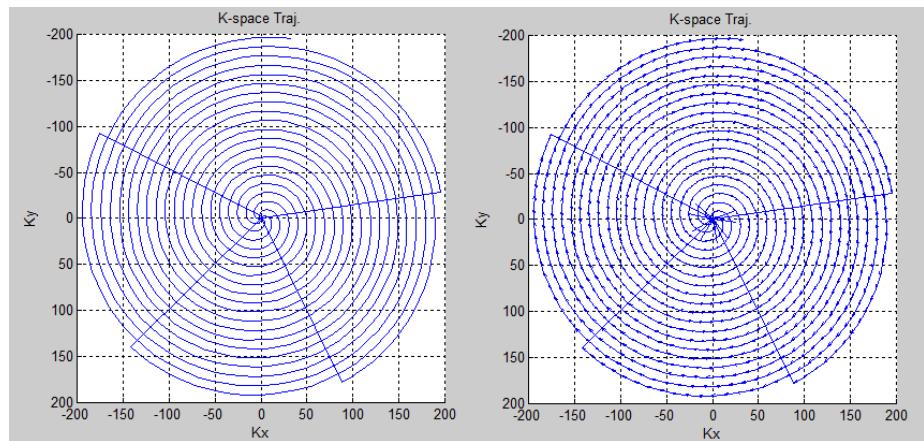


Figure 5.16: Matlab Rendered K-Space Trajectory for A Spiral Readout with 5 Interleaves. The left figure is without K-Space point rendering. The right figure is with K-Space point rendering with the arrows indicating K-Space traversing direction.

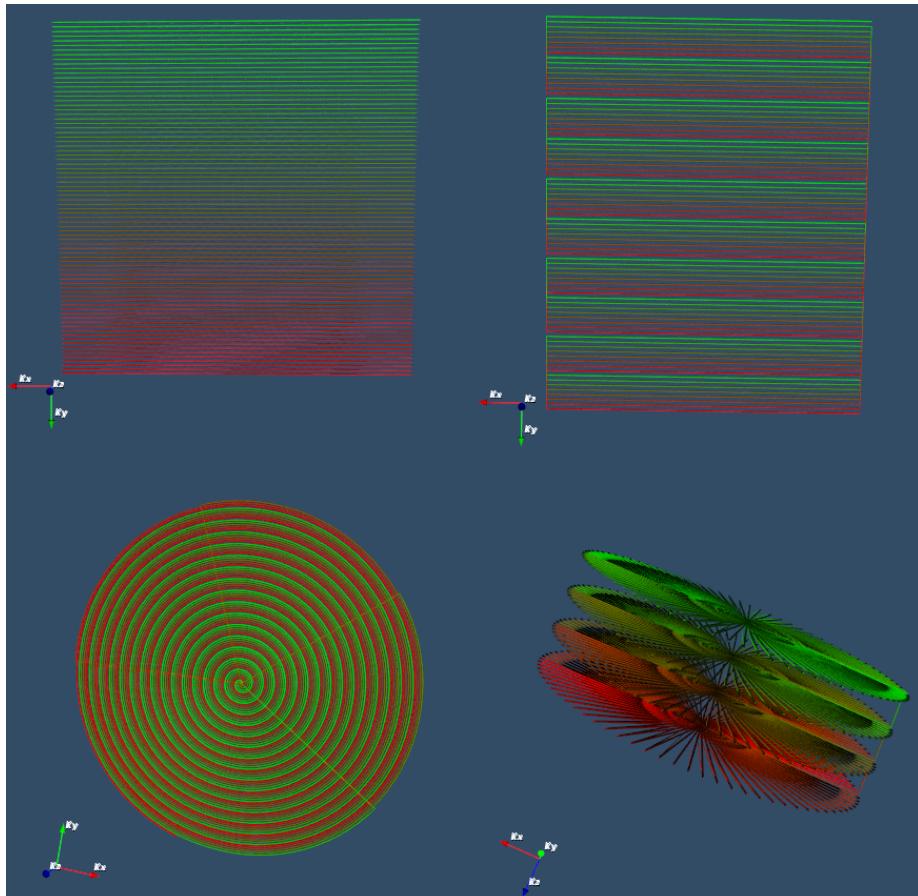


Figure 5.17: VTK Rendered K-Space Trajectory Examples. The top left figure shows a typical single slice Cartesian readout. The top right figure shows a single slice multishot EPI readout. The bottom left figure shows a single slice multishot spiral readout. The bottom right figure shows a 3D Stack-of-Stars radial readout.

keyboard ‘q’ to quit the VTK window before any subsequent simulation. Pressing the quit button on the VTK window under Linux system will force the entire Matlab to close, this is believed to be a compatibility bug between Matlab and OpenGL which is used by VTK.

### 5.2.2 Virtual Structure

For the convenience of transferring data and configuration information across different modules, MRiLab defined several Matlab structure variables in the global scope. These structures start with ‘V’ standing for ‘Virtual Structure’. Understanding what these structures are and how they are working is important to work with MRiLab and to customize specific experiment design. There are at least two virtual structures that are useful for designing MR sequences.

- VCtl : Virtual Control

VCtl encapsulates all the simulation setting parameters in the main control console. For example, the user can use ‘VCtl.TE’ to reference ‘TE’ value in the main control console; use ‘VCtl.FlipAng’ to reference ‘FlipAng’ value in the main control console. VCtl also allows the user to reference parameters in special techniques if loaded. Another example is that MRiLab uses ‘VCtl.TR’ in the ‘SE’ for determining time interval for each TR section. The user can use any legal Matlab syntax combined with VCtl to create desired effect, such as use ‘2 \* VCtl.TR’ to indicate twice of ‘TR’ value.

- VVar : Virtual Variable

VVar encapsulates the loop index variables that MRiLab uses for generating MR sequence signal. A section of code for generating MR sequence is shown:

```
% MR Sequence Generating Loop
VVar.SliceCount=0;
VVar.PhaseCount=0;
VVar.TRCount=0;
s=1;
j=1;

while s<=VCtl.SecondPhNum
    VVar.SliceCount=s;
    while j<=VCtl.FirstPhNum
        VVar.PhaseCount=j;
        VVar.TRCount=VVar.TRCount+1;

        ...
        Sequence Generating Code
        ...
    end
end
```

```

j=j+1;
end
j=1;
s=s+1;
end

```

The user can use the loop index variables in VVar

- VVar.TRCount : The TR section index
- VVar.PhaseCount : The first phase encoding index
- VVar.SliceCount : The second (i.e. slice) phase encoding index
- VCtl.FirstPhNum : The total number of first phase encoding steps
- VCtl.SecondPhNum : The total number of second phase encoding steps

For example, to create 180° RF phase cycling in bSSFP sequence, the user can set the attributes for the excitation RF pulse as

- CV3 : 2\*pi/2
- CV4 : 2
- rfPhase : rem(VVar.TRCount-1,CV4)\*CV3

### 5.2.3 GzSS Macro Library

A GzSS macro is a predefined module for a gradient pulse on the GzSS sequence line. MRILab GzSS macro library is a collection of GzSS macros covering different gradient pulse types including slice selection and slice phase encoding pulses. Notice that by default the area under the gradient ramp is ignored. This section will give an introduction for each of the GzSS macros provided in MRILab.

#### GzSelective

A GzSS macro that creates a typical slice selective gradient pulse (Figure 5.18). This macro contains attributes including:

- t2Start (s) : Slice selection gradient pulse starting time
- t2End (s) : Slice selection gradient pulse ending time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- GzAmp (T) : The amplitude of the gradient
- Gz1Sign : The polarity of the prephasing gradient, set 0 for nulling

- Gz2Sign : The polarity of the slice selection gradient, set 0 for nulling
- Gz3Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

Notice that both the prephasing gradient and rephasing gradient have half of the area of the slice selection gradient.



Figure 5.18: GzSelective Waveform

### GzSelective2

A GzSS macro that creates a slice selective gradient pulse straddled with crusher gradient (Figure 5.19). This macro contains attributes including:

- t2Start (s) : Slice selection gradient pulse starting time
- t2End (s) : Slice selection gradient pulse ending time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- tGz1 (s) : The duration of the left crusher
- tGz3 (s) : The duration of the right crusher
- Gz1Amp (T) : The amplitude of the left crusher
- Gz2Amp (T) : The amplitude of the slice selective gradient
- Gz3Amp (T) : The amplitude of the right crusher
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse



Figure 5.19: GzSelective2 Waveform

### GzTrapezoid

A GzSS macro that creates a trapezoid gradient pulse (Figure 5.20) on GzSS sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- tRamp (s) : The trapezoid pulse ramp time, assume symmetric ramp on both side
- sRamp : The sample points on the ramp, use the value of 2 for ignoring the area under the ramp, use above 2 for counting the ramp area
- GzAmp (T) : The amplitude of the gradient
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse



Figure 5.20: GzTrapezoid Waveform

### GzAreaTrapezoid

A GzSS macro that creates a trapezoid gradient pulse of specified area (Figure 5.21) on GzSS sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- Area (1/m) : The area under this gradient pulse

- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse



Figure 5.21: GzAreaTrapezoid Waveform

### GzAreaTrapezoid2

A GzSS macro that creates a trapezoid gradient pulse of specified area with highest system performance (Figure 5.22) on GzSS sequence line. This macro creates a gradient pulse with nearly shortest pulse width for the given system hardware constraint. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- Area (1/m) : The area under this gradient pulse
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

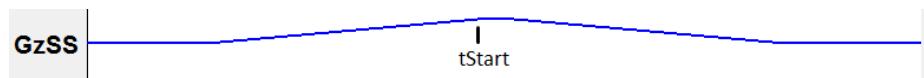


Figure 5.22: GzAreaTrapezoid2 Waveform

### GzCartesian

A GzSS macro that creates a Cartesian phase encoding gradient pulse (Figure 5.23) along the slice direction. This macro contains attributes including:

- t1Start (s) : The phase encoding gradient pulse starting time
- t1End (s) : The phase encoding gradient pulse ending time
- t2Start (s) : The rephasing gradient pulse starting time
- t2End (s) : The rephasing gradient pulse ending time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- Gz1Sign : The polarity of the phase encoding gradient, set 0 for nulling
- Gz2Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

Notice that the phase encoding gradient and the rephasing gradient have the same area that is automatically calculated based on the imaging parameters in the main control console.

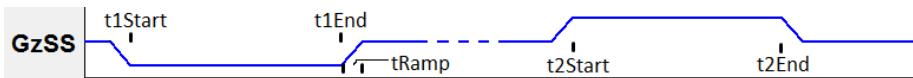


Figure 5.23: GzCartesian Waveform

### GzUser

If the user has the gradient pulse data saved in a MAT file, the user can easily import the gradient file into MRILab by using ‘GzUser’ macro. The gradient pulse MAT file needs to contain two matrices including ‘GTime’ (i.e. gradient time points) and ‘GAmp’ (i.e. gradient amplitude). Both matrices must have the same size as m-by-n, where m is the number of TR sections and n is the number of gradient waveform points. In typical MR sequence, the entire sequence is composed of multiple TR sections. The  $i$ th TR section uses the  $i$ th gradient pulse stored in the  $i$ th row of these two matrices. If the number of row is less than the number of TR sections, the last gradient pulse will be used for all the remaining TR sections. The ‘GzUser’ macro contains attributes including:

- GzFile : The path to the file that stores the gradient pulse data, quoted using single quotes
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

#### 5.2.4 GyPE Macro Library

A GyPE macro is a predefined module for a gradient pulse on the GyPE sequence line. MRiLab GyPE macro library is a collection of GyPE macros covering different gradient pulse types for performing phase encoding. Notice that by default the area under the gradient ramp is ignored. This section will give an introduction for each of the GyPE macros provided in MRiLab.

##### GyTrapezoid

Similar to GzTrapezoid (Figure 5.20), GyTrapezoid creates a trapezoid gradient pulse on GyPE sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- tRamp (s) : The trapezoid pulse ramp time, assume symmetric ramp on both side
- sRamp : The sample points on the ramp, use the value of 2 for ignoring the area under the ramp, use above 2 for counting the ramp area
- GyAmp (T) : The amplitude of the gradient
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GyAreaTrapezoid**

Similar to GzAreaTrapezoid (Figure 5.21), GyAreaTrapezoid creates a trapezoid gradient pulse of specified area on GyPE sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- Area (1/m) : The area under this gradient pulse
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GyAreaTrapezoid2**

Similar to GzAreaTrapezoid2 (Figure 5.22), GyAreaTrapezoid2 creates a trapezoid gradient pulse of specified area with highest system performance on GyPE sequence line. This macro creates a gradient pulse with nearly shortest pulse width for the given system hardware constraint. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- Area (1/m) : The area under this gradient pulse
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GyCartesian**

Similar to GzCartesian (Figure 5.23), GyCartesian creates a Cartesian phase encoding gradient pulse on GyPE sequence line. This macro contains attributes including:

- t1Start (s) : The phase encoding gradient pulse starting time

- t1End (s) : The phase encoding gradient pulse ending time
- t2Start (s) : The rephasing gradient pulse starting time
- t2End (s) : The rephasing gradient pulse ending time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- Gy1Sign : The polarity of the phase encoding gradient, set 0 for nulling
- Gy2Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

Notice that the phase encoding gradient and the rephasing gradient have the same area that is automatically calculated based on the imaging parameters in the main control console.

### GyRadial

A GyPE macro that creates a phase encoding gradient pulse for radial K-Space trajectory (Figure 5.24) on GyPE sequence line. This macro contains attributes including:

- t1Start (s) : The prephasing gradient pulse starting time
- t2Middle (s) : The phase encoding gradient pulse middle time
- t3Start (s) : The rephasing gradient pulse starting time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- Gy1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gy2Sign : The polarity of the phase encoding gradient, set 0 for nulling
- Gy3Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that both the prephasing gradient and rephasing gradient have half of the area of the phase encoding gradient that is automatically calculated based on the imaging parameters in the main control console. MRILab requires the ‘Radial’ special technique tab to be loaded for properly configuring the ‘GyRadial’, ‘GxRadial’ and ‘ADCRadial’ macro. The user can set t2Middle value as ‘VCtl.TE’ to acquire the echo signal.

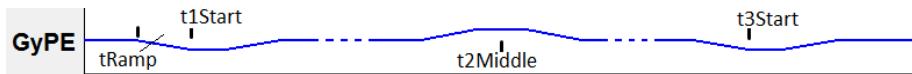


Figure 5.24: GyRadial Waveform

### GySpiral

A GyPE macro that creates a phase encoding gradient pulse for spiral K-Space trajectory (Figure 5.25) on GyPE sequence line. This macro contains attributes including:

- tStart (s) : The phase encoding gradient pulse starting time
- dt (s) : The time interval of gradient pulse sample points
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that the area of the phase encoding gradient is automatically calculated based on the imaging parameters in the main control console. MRILab requires the ‘Spiral’ special technique tab to be loaded for properly configuring the ‘GySpiral’, ‘GxSpiral’ and ‘ADCSpiral’ macro. The user can set tStart value as ‘VCtl.TE’ to acquire the echo signal.

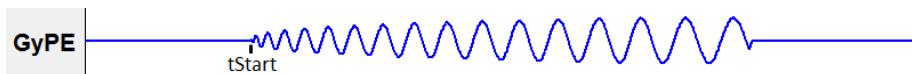


Figure 5.25: GySpiral Waveform

### GyFSE

A GyPE macro that creates a FSE phase encoding gradient pulse train (Figure 5.26) on GyPE sequence line. This macro contains attributes including:

- tMiddle (s) : The middle time of the gradient pulse train
- tOffset (s) : The time offset of the gradient pulse

- tGy1 (s) : The duration of the phase encoding gradient
- tGy2 (s) : The duration of the rephasing gradient
- Gy1Sign : The polarity of the phase encoding gradient, set 0 for nulling
- Gy2Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that the phase encoding gradient and the rephasing gradient have the same area that is automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘FSE’ special technique tab to be loaded for properly configuring the ‘GyFSE’, ‘GxFSE’ and ‘ADCFSE’ macro. To satisfy Carr Purcell Meiboom Gill (CPMG) condition and acquire echo signal at the center between two consecutive refocusing RF pulse, the effective TE value must equal  $(\text{floor}(\text{FSE\_ETL}/2)+1)*\text{FSE\_ESP}$ . The user can set tMiddle value as ‘VCtl.TE’ to acquire the echo signal, where the ‘VCtl.TE’ becomes the effective TE value.

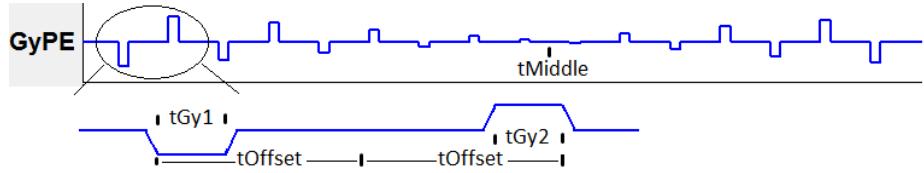


Figure 5.26: GyFSE Waveform

### GyEPI

A GyPE macro that creates an EPI phase encoding gradient pulse train (Figure 5.27) on GyPE sequence line. This macro contains attributes including:

- t2Middle (s) : The middle time of the blip gradient pulse train
- t1Start (s) : The prephasing gradient starting time
- Gy1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gy2Sign : The polarity of the blip gradient train, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that the area of the prephasing gradient and the blip gradient are automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘EPI’ special technique tab to be loaded for properly configuring the ‘GyEPI’, ‘GxEPI’ and ‘ADCEPI’ macro. The user can set t2Middle value as ‘VCtl.TE’ to acquire the echo signal, where the ‘VCtl.TE’ becomes the effective TE value.



Figure 5.27: GyEPI Waveform

### GyUser

If the user has the gradient pulse data saved in a MAT file, the user can easily import the gradient file into MRiLab by using ‘GyUser’ macro. The gradient pulse MAT file needs to contain two matrices including ‘GTime’ (i.e. gradient time points) and ‘GAmp’ (i.e. gradient amplitude). Both matrices must have the same size as m-by-n, where m is the number of TR sections and n is the number of gradient waveform points. In typical MR sequence, the entire sequence is composed of multiple TR sections. The  $i$ th TR section uses the  $i$ th gradient pulse stored in the  $i$ th row of these two matrices. If the number of row is less than the number of TR sections, the last gradient pulse will be used for all the remaining TR sections. The ‘GyUser’ macro contains attributes including:

- GyFile : The path to the file that stores the gradient pulse data, quoted using single quotes
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### 5.2.5 GxR Macro Library

A GxR macro is a predefined module for a gradient pulse on the GxR sequence line. MRiLab GxR macro library is a collection of GxR macros covering different gradient pulse types for performing frequency encoding. Notice that by default the area under the gradient ramp is ignored. This section will give an introduction for each of the GxR macros provided in MRiLab.

### **GxTrapezoid**

Similar to GzTrapezoid (Figure 5.20), GxTrapezoid creates a trapezoid gradient pulse on GxR sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- tRamp (s) : The trapezoid pulse ramp time, assume symmetric ramp on both side
- sRamp : The sample points on the ramp, use the value of 2 for ignoring the area under the ramp, use above 2 for counting the ramp area
- GxAmp (T) : The amplitude of the gradient
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GxAreaTrapezoid**

Similar to GzAreaTrapezoid (Figure 5.21), GxAreaTrapezoid creates a trapezoid gradient pulse of specified area on GxR sequence line. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- tEnd (s) : The trapezoid gradient pulse ending time
- Area (1/m) : The area under this gradient pulse
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GxAreaTrapezoid2**

Similar to GzAreaTrapezoid2 (Figure 5.22), GxAreaTrapezoid2 creates a trapezoid gradient pulse of specified area with highest system performance on GxR sequence line. This macro creates a gradient pulse with nearly shortest pulse width for the given system hardware constraint. This macro contains attributes including:

- tStart (s) : The trapezoid gradient pulse starting time
- Area (1/m) : The area under this gradient pulse
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### **GxCartesian**

A GxR macro that creates a Cartesian frequency encoding gradient pulse (Figure 5.28) on GxR sequence line. This macro contains attributes including:

- t1Start (s) : The prephasing gradient pulse starting time
- t2Middle (s) : The frequency encoding gradient pulse middle time
- t3Start (s) : The rephasing gradient pulse starting time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- Gx1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gx2Sign : The polarity of the frequency encoding gradient, set 0 for nulling
- Gx3Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape
- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

Notice that both the prephasing gradient and rephasing gradient have half of the area of the frequency encoding gradient.

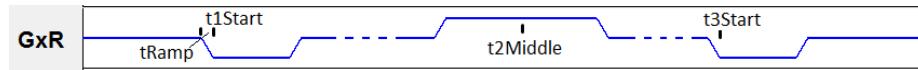


Figure 5.28: GxCartesian Waveform

### GxRadial

Similar to GyRadial (Figure 5.24), GxRadial creates a phase encoding gradient pulse for radial K-Space trajectory on GxR sequence line. This macro contains attributes including:

- t1Start (s) : The prephasing gradient pulse starting time
- t2Middle (s) : The phase encoding gradient pulse middle time
- t3Start (s) : The rephasing gradient pulse starting time
- tRamp (s) : Gradient pulse ramp time, assume symmetric ramp on both side
- Gx1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gx2Sign : The polarity of the phase encoding gradient, set 0 for nulling
- Gx3Sign : The polarity of the rephasing gradient, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that both the prephasing gradient and rephasing gradient have half of the area of the phase encoding gradient that is automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘Radial’ special technique tab to be loaded for properly configuring the ‘GyRadial’, ‘GxRadial’ and ‘ADCRadial’ macro. The user can set t2Middle value as ‘VCtl.TE’ to acquire the echo signal.

### GxSpiral

Similar to GySpiral (Figure 5.25), GxSpiral creates a phase encoding gradient pulse for spiral K-Space trajectory on GxR sequence line. This macro contains attributes including:

- tStart (s) : The phase encoding gradient pulse starting time
- dt (s) : The time interval of gradient pulse sample points
- Switch : The flag for turning on and off gradient pulse in the sequence line

- Notes : The notes of the gradient pulse

Notice that the area of the phase encoding gradient is automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘Spiral’ special technique tab to be loaded for properly configuring the ‘GySpiral’, ‘GxSpiral’ and ‘ADCSpiral’ macro. The user can set tStart value as ‘VCtl.TE’ to acquire the echo signal.

### GxFSE

A GxR macro that creates a FSE frequency encoding gradient pulse train (Figure 5.29) on GxR sequence line. This macro contains attributes including:

- t2Middle (s) : The middle time of the gradient pulse train
- t1Start (s) : The prephasing gradient starting time
- Gx1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gx2Sign : The polarity of the frequency encoding gradient train, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that the prephasing gradient has half of the area of the frequency encoding gradient that is automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘FSE’ special technique tab to be loaded for properly configuring the ‘GyFSE’, ‘GxFSE’ and ‘ADCFSE’ macro. To satisfy CPMG condition and acquire echo signal at the center between two consecutive refocusing RF pulse, the effective TE value must equal  $(\text{floor}(\text{FSE\_ETL}/2)+1)*\text{FSE\_ESP}$ . The user can set t2Middle value as ‘VCtl.TE’ to acquire the echo signal, where the ‘VCtl.TE’ becomes the effective TE value.

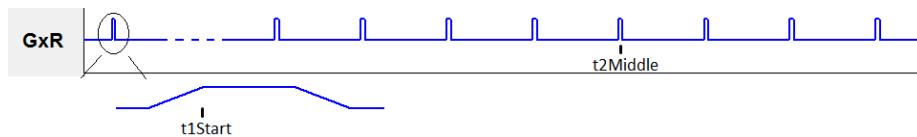


Figure 5.29: GxFSE Waveform

### GxEPI

A GxR macro that creates an EPI frequency encoding gradient pulse train (Figure 5.30) on GxR sequence line. This macro contains attributes including:

- t2Middle (s) : The middle time of the frequency encoding gradient pulse train
- t1Start (s) : The prephasing gradient starting time
- Gx1Sign : The polarity of the prephasing gradient, set 0 for nulling
- Gx2Sign : The polarity of the frequency encoding gradient train, set 0 for nulling
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Notes : The notes of the gradient pulse

Notice that the area of the prephasing gradient and the frequency encoding gradient are automatically calculated based on the imaging parameters in the main control console. MRiLab requires the ‘EPI’ special technique tab to be loaded for properly configuring the ‘GyEPI’, ‘GxEPI’ and ‘ADCEPI’ macro. The user can set t2Middle value as ‘VCtl.TE’ to acquire the echo signal, where the ‘VCtl.TE’ becomes the effective TE value.



Figure 5.30: GxEPI Waveform

### GxUser

If the user has the gradient pulse data saved in a MAT file, the user can easily import the gradient file into MRiLab by using ‘GxUser’ macro. The gradient pulse MAT file needs to contain two matrices including ‘GTime’ (i.e. gradient time points) and ‘GAmp’ (i.e. gradient amplitude). Both matrices must have the same size as m-by-n, where m is the number of TR sections and n is the number of gradient waveform points. In typical MR sequence, the entire sequence is composed of multiple TR sections. The *i*th TR section uses the *i*th gradient pulse stored in the *i*th row of these two matrices. If the number of row is less than the number of TR sections, the last gradient pulse will be used for all the remaining TR sections. The ‘GxUser’ macro contains attributes including:

- GxFile : The path to the file that stores the gradient pulse data, quoted using single quotes
- Switch : The flag for turning on and off gradient pulse in the sequence line
- Duplicates : The number of the gradient pulse duplicates, used for creating multiple gradient pulses with the same shape

- DupSpacing : The time spacing between gradient pulse duplicates
- Notes : The notes of the gradient pulse

### 5.2.6 ADC Macro Library

An ADC macro is a predefined module for an ADC flag pulse on the ADC sequence line. Signal acquisition starts when ADC flag is 1 and stops when ADC flag is 0. The ADC flag pulse is sampled at certain sampling rate that is determined by the imaging parameters (default by using ‘BandWidth’) in the main control console. MRiLab ADC macro library is a collection of ADC macros covering different pulse types for performing signal acquisition. This section will give an introduction for each of the ADC macros provided in MRiLab.

#### ADCBlock

An ADC macro that creates an ADC flag pulse with user defined sampling rate on ADC sequence line. This macro contains attributes including:

- tStart (s) : The ADC starting time
- tEnd (s) : The ADC ending time
- sSample : The number of linear sample points when ADC flag is 1
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Duplicates : The number of the ADC pulse duplicates, used for creating multiple ADC pulses with the same shape
- DupSpacing : The time spacing between ADC pulse duplicates
- Notes : The notes of the ADC pulse



Figure 5.31: ADCBlock Waveform

#### ADCCartesian

An ADC macro that creates an ADC flag pulse for Cartesian readout on ADC sequence line. This macro contains attributes including:

- tMiddle (s) : The ADC middle time, typically set ‘VCtl.TE’ for acquiring echo signal
- Switch : The flag for turning on and off ADC pulse in the sequence line

- Duplicates : The number of the ADC pulse duplicates, used for creating multiple ADC pulses with the same shape
- DupSpacing : The time spacing between ADC pulse duplicates
- Notes : The notes of the ADC pulse

### **ADCRadial**

An ADC macro that creates an ADC flag pulse for radial readout on ADC sequence line. This macro needs ‘Radial’ tab to be loaded. This macro contains attributes including:

- tMiddle (s) : The ADC middle time, typically set ‘VCtl.TE’ for acquiring echo signal
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Notes : The notes of the ADC pulse

### **ADCSpiral**

An ADC macro that creates an ADC flag pulse for spiral readout on ADC sequence line. This macro needs ‘Spiral’ tab to be loaded. This macro contains attributes including:

- tStart (s) : The ADC starting time, typically set ‘VCtl.TE’ for acquiring echo signal
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Notes : The notes of the ADC pulse

### **ADCFSE**

An ADC macro that creates an ADC flag pulse train for FSE readout on ADC sequence line. This macro needs ‘FSE’ tab to be loaded. This macro contains attributes including:

- tMiddle (s) : The ADC pulse train middle time, typically set ‘VCtl.TE’ for acquiring echo signal
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Notes : The notes of the ADC pulse

### **ADCEPI**

An ADC macro that creates an ADC flag pulse train for EPI readout on ADC sequence line. This macro needs ‘EPI’ tab to be loaded. This macro contains attributes including:

- tMiddle (s) : The ADC pulse train middle time, typically set ‘VCtl.TE’ for acquiring echo signal
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Notes : The notes of the ADC pulse

### **ADCUser**

If the user has the ADC pulse data saved in a MAT file, the user can easily import the ADC file into MRiLab by using ‘ADCUser’ macro. The ADC pulse MAT file needs to contain two matrices including ‘GTime’ (i.e. ADC time points) and ‘GAmp’ (i.e. ADC amplitude, use 1 for signal acquisition, 0 for no signal acquisition). Both matrices must have the same size as m-by-n, where m is the number of TR sections and n is the number of ADC sample points. In typical MR sequence, the entire sequence is composed of multiple TR sections. The *i*th TR section uses the *i*th ADC pulse stored in the *i*th row of these two matrices. If the number of row is less than the number of TR sections, the last ADC pulse will be used for all the remaining TR sections. The ‘ADCUser’ macro contains attributes including:

- ADCFile : The path to the file that stores the ADC pulse data, quoted using single quotes
- Switch : The flag for turning on and off ADC pulse in the sequence line
- Duplicates : The number of the ADC pulse duplicates, used for creating multiple ADC pulses with the same shape
- DupSpacing : The time spacing between ADC pulse duplicates
- Notes : The notes of the ADC pulse

Notice that ‘ADCUser’ macro sets the first and last ADC sample points to 0 regardless of their original value, therefore the signal is not acquired at the first and last time points.

#### **5.2.7 Ext Macro Library**

An Ext macro is a predefined module for an Ext flag pulse on the Ext sequence line. MRiLab specifies Ext signal for performing extended real time processes including calculating remaining scan time, manipulating K-Space location and triggering object motion etc. Ext macro library only contains ‘ExtBit’ macro, however the ‘Ext’ attribute in this macro triggers different processes.

### ExtBit

The ‘ExtBit’ macro (Figure 5.32) creates a triangle blip pulse on Ext sequence line. This macro contains attributes including:

- tStart (s) : The Ext starting time
- Ext : The Ext flag
- Switch : The flag for turning on and off Ext pulse in the sequence line
- Duplicates : The number of the Ext pulse duplicates, used for creating multiple Ext pulses with the same shape
- DupSpacing : The time spacing between Ext pulse duplicates
- Notes : The notes of the Ext pulse



Figure 5.32: ExtBit Waveform

Different Ext flags execute different extended real time processes during runtime scan. These extended processes are implemented using Plugin code in the /Plugin folder. MRiLab reserved several Ext flags for particular purposes.

- 1 : Plugin\_ResetK, reset Kx, Ky and Kz to zero
- 2 : Plugin\_ReverseK, reverse Kx, Ky and Kz
- 3 : Plugin\_LockK, remember current K space location
- 4 : Plugin\_ReleaseK, set K space location to the latest remembered one, used after Plugin\_LockK
- 5 : Plugin\_Timer, calculate remaining scan time, display it on the main control console
- 6 : Plugin\_IdealCrusher, dephase transverse magnetization of all the spins, set them to zero
- 7 : Plugin\_rfRef, remember current rfPhase value and demodulate signal phase according to this value
- 8 : Plugin\_ExecuteMotion, trigger object motion

### 5.2.8 Make Your Own Ext Plugin

The user can define Ext flags and use Ext Plugin to create extended real time process. To make your own Ext flag and Plugin code, you should follow the following steps :

1. Write Ext Plugin code

It is strongly recommended to write your Plugin code based on a template like :

```
function Plugin_YourPluginName
%Create a Ext Plugin based on your code
global VCtl      % use VCtl structure, read only
global VVar       % use VVar structure, read only
global VObj       % use VObj structure, read and write
global VMag      % use VMag structure, read and write
global VCoil     % use VCoil structure, read and write

...
% The main code for your Ext Plugin
...

end
```

As mentioned before, VCtl encapsulates all the simulation setting parameters in the main control console. VVar not only encapsulates the loop index variables that MRiLab uses for generating MR sequence signal, but also contains variables for temporarily buffering instant sequence line values during runtime. **Do keep in mind, VCtl and VVar are read only, changing values inside these two structures may cause MRiLab crash.**

- VVar.rfAmp (T) : A array with the size of  $TxCoilNum \times 1$  for storing current RF amplitude
- VVar.rfPhase (rad) : A array with the size of  $TxCoilNum \times 1$  for storing current RF phase
- VVar.rfFreq (Hz) : A array with the size of  $TxCoilNum \times 1$  for storing current RF frequency
- VVar.rfCoil : The CoilID of current working coil channel, used in multiple RF transmitting

- VVar.rfRef (rad) : The remembered RF phase, used for demodulating signal phase at ADC
- VVar.GzAmp (T/m) : The current GzSS gradient amplitude
- VVar.GyAmp (T/m) : The current GyPE gradient amplitude
- VVar.GxAmp (T/m) : The current GXR gradient amplitude
- VVar.ADC : The current ADC flag
- VVar.Ext : The current Ext flag
- VVar.t (s) : The current time
- VVar.Kz (1/m) : The current Kz value
- VVar.Ky (1/m) : The current Ky value
- VVar.Kx (1/m) : The current Kx value
- VVar.TRCOUNT : The current TR section index

Notice that there are three new Virtual Structures (VObj, VMag and VCoil) in this template, they store the variables about the virtual object, B0 field and coil B1 field which are accessible and editable in Plugin code. **Do keep in mind, do not change the size of the matrices inside these three structures.**

- VObj.Rho : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing spin density
- VObj.T1 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T1 relaxation time
- VObj.T2 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T2 relaxation time
- VObj.Mx : A matrix with the size of  $YDim \times XDim \times ZDim \times SpinPerVoxel \times TypeNum$  for describing magnetization in  $x$  direction
- VObj.My : A matrix with the size of  $YDim \times XDim \times ZDim \times SpinPerVoxel \times TypeNum$  for describing magnetization in  $y$  direction
- VObj.Mz : A matrix with the size of  $YDim \times XDim \times ZDim \times SpinPerVoxel \times TypeNum$  for describing magnetization in  $z$  direction
- VMag.dWRnd (rad/s) : A matrix with the size of  $YDim \times XDim \times ZDim \times SpinPerVoxel \times TypeNum$  for describing microscopic resonance frequency variation caused by T2\* effect
- VMag.dB0 (T) : A matrix with the size of  $YDim \times XDim \times ZDim$  for describing local B0 field variation
- VMag.Gxgrid (m) : A matrix with the size of  $YDim \times XDim \times ZDim$  for describing spatial grid in X direction

- VMag.Gygrid (m) : A matrix with the size of  $YDim \times XDim \times ZDim$  for describing spatial grid in Y direction
- VMag.Gzgrid (m) : A matrix with the size of  $YDim \times XDim \times ZDim$  for describing spatial grid in Z direction
- VCoi.TxCoilmg (T) : A matrix with the size of  $YDim \times XDim \times ZDim \times TxCoilNum$  for describing the magnitude of transmitting B1+ field
- VCoi.TxCoilpe (rad) : A matrix with the size of  $YDim \times XDim \times ZDim \times TxCoilNum$  for describing the phase of transmitting B1+ field
- VCoi.RxCoilx (T) : A matrix with the size of  $YDim \times XDim \times ZDim \times RxCoilNum$  for describing the  $x$  component of receiving B1- field
- VCoi.RxCoily (T) : A matrix with the size of  $YDim \times XDim \times ZDim \times RxCoilNum$  for describing the  $y$  component of receiving B1- field

where ‘TxCoilNum’ is the number of transmitting coil channels, ‘RxCoilNum’ is the number of receiving coil channels.

## 2. Register Ext Plugin

The user needs to register customized Plugins before MRiLab can use it. To register Plugins and assign Ext flags to them, simply open ‘DoExtPlugin.m’ under /Src folder.

```
function DoExtPlugin
% entry function for extended plugin based on Ext flag
global VVar

switch VVar.Ext
%% System Reserved Ext Flags (Positive)
    case 0 % normal status
        % do nothing
    case 1 % reset K space location
        Plugin_ResetK;
    case 2 % reverse K space location
        Plugin_ReverseK;
    case 3 % lock K space location
        Plugin_LockK;
    case 4 % release K space location
        Plugin_ReleaseK;
    case 5 % calculate remaining scan time
        Plugin_Timer;
```

```

        case 6 % ideal crusher, dephase Mxy
        Plugin_IdealCrusher; % wrapper for mex function
        case 7 % rfRef, demodulate signal phase referring to RF phase at ADC
        Plugin_rfRef;
        case 8 % trigger object motion
        Plugin_ExecuteMotion;
%% User Defined Ext Flags (Negative)

end
end

```

Add one more Switch case line, assign a distinct Ext flag, then add your Plugin function name under the case line. It is recommended to use negative Ext flag for user defined Plugin to make differences from system reserved ones.

### 5.2.9 Make Your Own Gradient Macro

Besides using macros to import gradient pulse from external files, the user can make customized gradient macros and use them to create desired K-Space trajectory. To make your own gradient macro, you should follow the following steps :

1. Write gradient macro code

It is strongly recommended to write your own gradient macro code based on the closest gradient macros in the MRiLab macro library. One template is like :

```

function [GAmp,GTIme]=GYourGradientMacroName(p)
%Create a gradient macro based on your code

global VCtl      % use VCtl structure, read only
global VObj       % use VObj structure, read only

% Create attribute list
Duplicates=max(1,p.Duplicates);
DupSpacing=max(0,p.DupSpacing);
...
attribute1=p.attribute1;
attribute2=p.attribute2;
attribute3=p.attribute3;
...
%
% The main code for your macro

```

```

...
GTime = ...;
GAmp = ...;
...

% Avoid baseline offset
GAmp(1)=0;
GAmp(end)=0;

% Create Duplicates
if Duplicates~=1 & DupSpacing ~=0
    GAmp=repmat(GAmp,[1 Duplicates]);
    TimeOffset = repmat(0:DupSpacing:(Duplicates-1)*DupSpacing, ...
        [length(GTime) 1]);
    GTime=repmat(GTime,[1 Duplicates]) + (TimeOffset(:))';
end
end

```

## 2. Register gradient macro

The gradient macro file can be put anywhere in the computer as long as the file is included in Matlab searching path, however it is recommended to put the the file under corresponding gradient folder under /SeqElem folder for consistent file organization. The customized gradient macro needs to be registered in the macro library before using. To register the macro, the user need to open file ‘SeqElem.xml’ under /SeqElem, then add one entry under gradient category with the proper attribute list. One example could be

```

<GzGradientMacroName  Switch="$1'on','off'"
DupSpacing="0"
Duplicates="1"
Notes="A new Gz gradient macro"
attribute1="$1'on','off'"
attribute2="0"
attribute3="0" />

```

Notice that in the above example, the first 3 attributes are required for MRiLab, The remaining attributes are optional based on the macro itself.

Once the gradient macro is coded and registered to the library, the user can use this customized gradient macro just like those default gradient macros in the library.

### 5.2.10 Make Your Own MR Sequence

To load a sequence into the MR Sequence Design interface, it is recommended to load the sequence into the main control console first and then click toolbar icon to activate the MR Sequence Design interface. This is mainly because the associated imaging parameters that are necessary for configuring sequence will also be loaded during the sequence loading process at the main control console. However, if the interface has already been activated, the user can also use the sequence loading function to load another sequence with current imaging parameters. To load a sequence, click ‘Load’ menu then click ‘Load Sequence File’, choose a sequence XML file.

The user can create a new MR sequence in the MR Sequence Design interface. To create a new sequence, click ‘New’ then click ‘Create Sequence File’. A sequence creation window (Figure 5.33) will show up and ask for new sequence name and notes. To follow MRILab naming convention, the user is recommended to use ‘PSD\_’ followed by a legal name string that is distinct to the remaining sequences in MRILab. Then click ‘OK’ to select a path for storing the sequence XML file. It’s strongly recommended to put the sequence under the MRILab sequence root folder /PSD according to the sequence type so that the sequence is visible to MRILab. Finally, MRILab will create a new sequence XML file based on the content of ‘PSD\_GRE3D’.

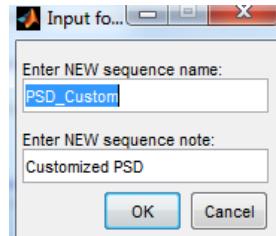


Figure 5.33: The Sequence Creation Window

### 5.2.11 Make Your Own Virtual Object

To design and optimize MR sequence, the user may need virtual objects with specific geometry and properties according to their experiment purpose. Although MRILab provides a wide range of phantoms that cover several tissue types in human, the user may still need to define customized virtual object. To make your own virtual object, you should follow the following steps :

1. Make Regular Virtual Object

Create a Matlab structure ‘VObj’. ‘VObj’ must contains variables including:

- Gyro (rad/s/T) : The gyromagnetic ratio of the spin
- ChemShift (Hz/T) : A array with the size of  $1 \times TypeNum$  for describing the chemical shift of the spins
- XDim : The number of voxels in X direction
- YDim : The number of voxels in Y direction
- ZDim : The number of voxels in Z direction
- XDimRes (m) : The voxel size in X direction
- YDimRes (m) : The voxel size in Y direction
- ZDimRes (m) : The voxel size in Z direction
- Type : A string for describing the type of the spin
- TypeNum : The number of spin species
- Rho : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing spin density
- T1 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T1 relaxation time
- T2 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T2 relaxation time
- T2Star (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times TypeNum$  for describing T2\* relaxation time

Then save the ‘VObj’ structure as a MAT file.

## 2. Make Virtual Object with Magnetization Transfer Properties

The current version of MRiLab only supports two-pool MT model [9] with GPU acceleration. So the ‘VObj’ must have the structure like:

- Gyro (rad/s/T) : The gyromagnetic ratio of the spin
- ChemShift (Hz/T) : A array with the size of  $1 \times 2$  for describing the chemical shift of the spins, free pool first, bound pool second
- XDim : The number of voxels in X direction
- YDim : The number of voxels in Y direction
- ZDim : The number of voxels in Z direction
- XDimRes (m) : The voxel size in X direction
- YDimRes (m) : The voxel size in Y direction
- ZDimRes (m) : The voxel size in Z direction
- Type : A string for describing the type of the spin
- TypeNum : 2

- Rho : A matrix with the size of  $YDim \times XDim \times ZDim \times 2$  for describing spin density,  $Rho_f$  (free pool) first,  $Rho_b$  (bound pool) second
- T1 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times 2$  for describing T1 relaxation time, free pool first, bound pool second
- T2 (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times 2$  for describing T2 relaxation time, free pool first, bound pool second
- T2Star (s) : A matrix with the size of  $YDim \times XDim \times ZDim \times 2$  for describing T2\* relaxation time, free pool first, bound pool second
- K (1/s) : A matrix with the size of  $YDim \times XDim \times ZDim \times 2$  for describing MT cross-relaxation rate,  $K_{fb}$  (free pool to bound pool) first,  $K_{bf}$  (bound pool to free pool) second

Notice that the two-pool MT model needs to satisfy chemical equilibrium described as  $Rho_f \times K_{fb} = Rho_b \times K_{bf}$ . After making VObj, then save the ‘VObj’ structure as a MAT file. In MRiLab, the Bloch-equation solving kernel is only executed when the value of any sequence line needs to be updated. The time point at which this updating is happening is referred to as execution point. To accurately model MT exchange during scan, the user needs to create execution points on the entire RF sequence line. A typical method is to insert a long RF pulse with zero amplitude at the empty portion of the RF sequence line. The interested users are referred to PSD\_SPGR3DMT for more information.

### 5.3 Coil B1 Design

The Coil B1 Design toolbox can be activated by pressing ‘Coil Design Panel’ toolbar icon located at the top of the main simulation console. Depending on ‘Tx’ or ‘Rx’ is selected on the ‘Coil Selection’ panel, the loaded coil will show in the Coil B1 Design interface.



Figure 5.34: Coil Design Panel Toolbar Icon

#### 5.3.1 Coil Design GUI

Figure 5.35 demonstrates the overview of the Coil B1 Design interface. This interface consists of

1. Coil Element Macro Library

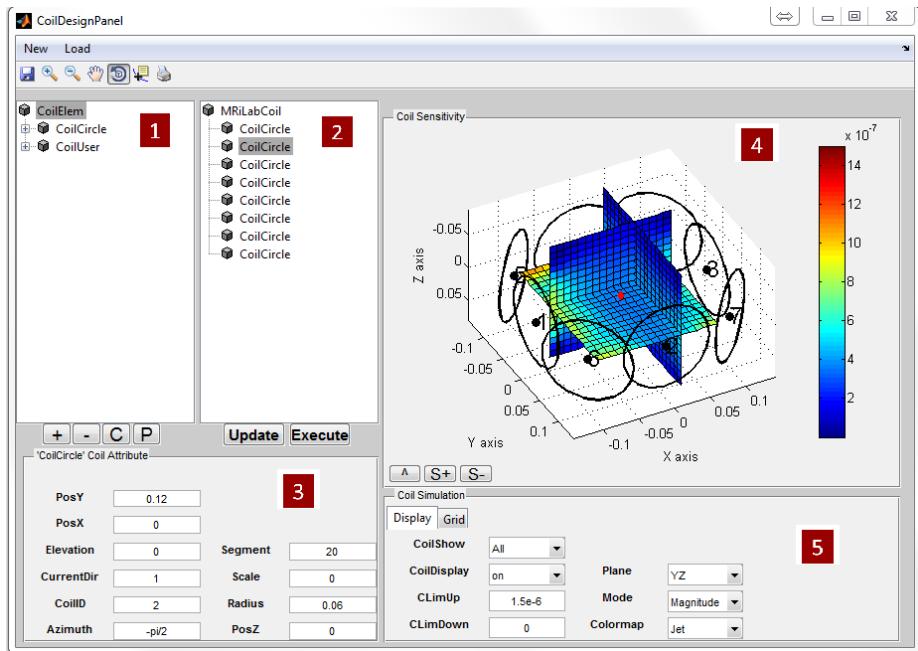


Figure 5.35: Coil B1 Design GUI. An eight channel coil is shown.

The Coil Element Macro Library contains coil element macros for constructing coil structure in MRiLab. The user needs to click the ‘CoilElem’ root to unfold subsequent macros.

## 2. Coil Structure

In MRiLab, a coil structure consists of arbitrary number of coil elements that are combined to create desired B1 field. The user can construct wanted B1 field by changing the content within the coil structure. To add a macro into the coil structure, the user needs to click one macro in the macro library, then click on the coil structure root (i.e. MRiLabCoil) to which this macro is inserted, then click ‘+’ macro operation button. To delete a macro from the coil structure, the user needs to click the unwanted macro, then click ‘-’ macro operation button. To duplicate an existing macro, the user needs to first click the source macro, then click ‘C’ macro operation button for copying, click on the coil structure root, then click ‘P’ macro operation button for pasting. MRiLab doesn’t allow empty coil structure.

## 3. Coil Element Attribute

When clicking on the coil element macro within the coil structure, the cor-

responding macro attributes will be updated at the coil element attribute panel down below the coil structure. The user can edit those attributes to modify the coil element so as to change generated B1 field. To make the modification effective, the user must press ‘Update’ button to update the coil file. Pressing ‘Execute’ button will update and redraw the B1 field map on this interface.

#### 4. Coil B1 Field

The coil configuration and B1 field (T) associated with the coil structure is displayed on the ‘Coil Sensitivity’ panel on the right side of this interface.

#### 5. Coil Simulation Control

The ‘Display’ and ‘Grid’ tab on the ‘Coil Simulation’ panel contains parameters for controlling B1 field display.

- Colormap : The colormap for the B1 field, includes ‘Jet’, ‘Gray’ and ‘Hot’
- CLimDown (T) : The lower bound of color limits
- CLimUp (T) : The upper bound of color limits
- CoilDisplay : The flag for turning on and off coil display
- CoilShow : The flag for choosing active coil for B1 field display
- Mode : The B1 field display mode, includes ‘Magnitude’, ‘Phase’, ‘Real’ and ‘Imaginary’
- Plane : The flag for activating B1 field slicing plane, includes ‘XY’, ‘XZ’ and ‘YZ’
- XDimRes (m) : The displayed spatial resolution in X direction
- YDimRes (m) : The displayed spatial resolution in Y direction
- ZDimRes (m) : The displayed spatial resolution in Z direction

MRiLab provides a group of B1 field display button (Figure 5.36) to help inspecting the B1 field details.

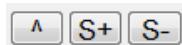


Figure 5.36: The B1 Field Display Button.

The B1 field display button group consists of:

- ^ : Undock B1 field on active B1 slicing plane (Figure 5.37)
- S+ : Move active B1 slicing plane forwards
- S- : Move active B1 slicing plane backwards

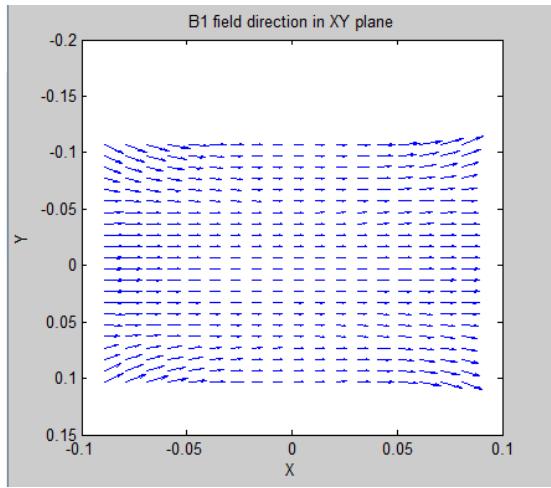


Figure 5.37: An Example of B1 Field on XY Plane.

### 5.3.2 Coil Element Macro Library

A coil element macro is a predefined module for creating a coil element that generates B1 field in three dimensional space. MRiLab coil element macro library is a collection of coil element macros covering simple coil geometries. This section will give an introduction for each of the coil element macro provided in MRiLab.

#### **CoilCircle**

A coil element macro that creates a Biot-Savart coil circle. This macro contains attributes including:

- Azimuth (rad) : The azimuth angle of the circle plane
- Elevation (rad) : The elevation angle of the circle plane
- Radius (m) : The coil circle radius
- PosZ (m) : The Z position of coil circle center
- PosY (m) : The Y position of coil circle center
- PosX (m) : The X position of coil circle center
- CurrentDir : The current direction in the coil circle, 1 for clockwise, -1 for counterclockwise
- Scale : The scale factor for the B1 field amplitude

- Segment : The number of line segments for approximating circle, MRiLab requires the same ‘Segment’ for each coil circle
- CoilID : The assigned coil ID, each coil element must have distinct ID

### **CoilUser**

If the user has the B1 field data saved in a MAT file, the user can easily import the B1 field into MRiLab coil design interface by using ‘CoilUser’ macro. The B1 field MAT file needs to contain two matrices including ‘B1x’ (i.e.  $x$  component of B1 field) and ‘B1y’ (i.e.  $y$  component of B1 field). Both of the two matrices must have the same size. The ‘CoilUser’ macro contains attributes including:

- B1File : The path to the file that stores the B1 field data, quoted using single quotes
- Interp : The interpolation method, includes ‘linear’, ‘nearest’, ‘spline’ and ‘cubic’
- PosZ (m) : The Z position of coil center
- PosY (m) : The Y position of coil center
- PosX (m) : The X position of coil center
- CoilID : The assigned coil ID, each coil element must have distinct ID

#### **5.3.3 Make Your Own Coil**

The user can also use the coil loading function to load another coil. To load a coil, click ‘Load’ menu then click ‘Load Coil File’, choose a coil XML file. When the coil is loaded, press ‘Execute’ to display field.

The user can create a new coil configuration in the Coil Design interface. To create a new coil, click ‘New’ then click ‘Create Coil File’. A coil creation window (Figure 5.38) will show up and ask for new coil name and notes. To follow MRiLab naming convention, the user is recommended to use ‘Coil\_’ followed by the number of coil elements and applied anatomy (e.g. Coil\_16ChChest), make sure that the new coil name is distinct to the remaining coil names in MRiLab. Then click ‘OK’ to select a path for storing the coil XML file. It’s strongly recommended to put the coil under the MRiLab coil root folder /Coil according to the coil type so that the coil is visible to MRiLab. Finally, MRiLab will create a new coil XML file based on the content of ‘Coil\_1ChHead’.

### **5.4 Magnet dB0 Design**

The Magnet dB0 Design toolbox can be activated by pressing ‘Magnet Design Panel’ toolbar icon located at the top of the main simulation console. The loaded magnet will show in the Magnet dB0 Design interface.

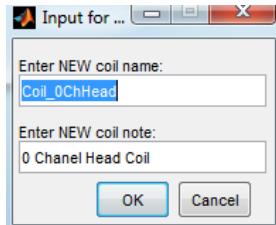


Figure 5.38: The Coil Creation Window



Figure 5.39: Magnet Design Panel Toolbar Icon

#### 5.4.1 Magnet Design GUI

Figure 5.40 demonstrates the overview of the Magnet dB0 Design interface. This interface consists of

1. Magnet Element Macro Library

The Magnet Element Macro Library contains magnet element macros for constructing magnet structure in MRiLab. The user needs to click the ‘MagElem’ root to unfold subsequent macros.

2. Magnet Structure

In MRiLab, a magnet structure consists of arbitrary number of magnet elements that are combined to create desired dB0 field. The user can construct desired dB0 field by changing the content within the magnet structure. To add a macro into the magnet structure, the user needs to click one macro in the macro library, then click on the magnet structure root (i.e. MRiLabMag) to which this macro is inserted, then click ‘+’ macro operation button. To delete a macro from the magnet structure, the user needs to click the unwanted macro, then click ‘-’ macro operation button. To duplicate an existing macro, the user needs to first click the source macro, then click ‘C’ macro operation button for copying, click on the magnet structure root, then click ‘P’ macro operation button for pasting. MRiLab doesn’t allow empty magnet structure.

3. Magnet Element Attribute

When clicking on the magnet element macro within the magnet structure, the corresponding macro attributes will be updated at the magnet element

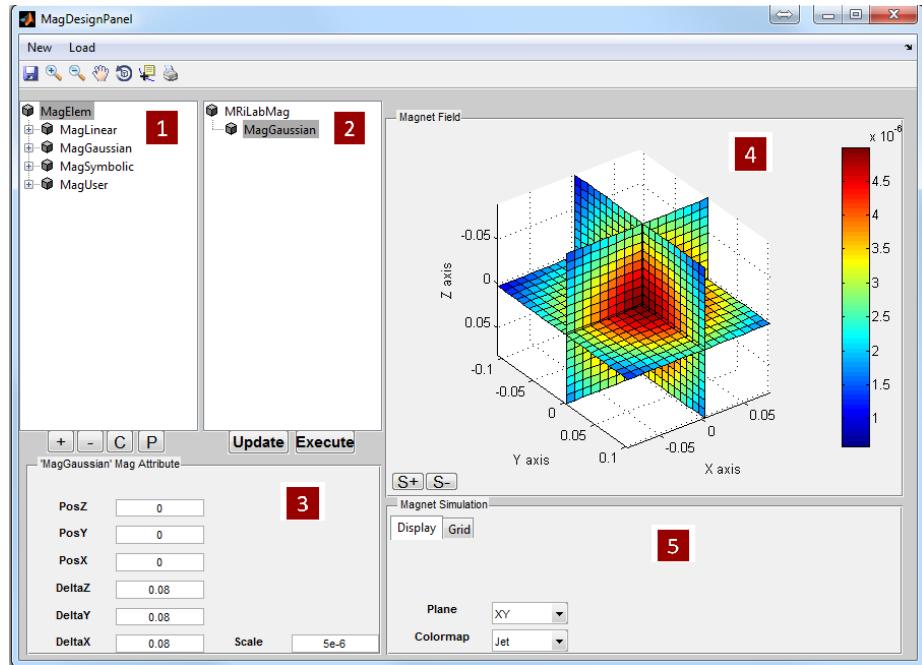


Figure 5.40: Magnet dB0 Design GUI

attribute panel down below the magnet structure. The user can edit those attributes to modify the magnet element so as to change generated dB0 field. To make the modification effective, the user must press ‘Update’ button to update the magnet file. Pressing ‘Execute’ button will update and redraw the dB0 field map on this interface.

#### 4. Magnet Field

The dB0 field (T) associated with the magnet structure is displayed on the ‘Magnet Field’ panel on the right side of this interface.

#### 5. Magnet Simulation Control

The ‘Display’ and ‘Grid’ tab on the ‘Magnet Simulation’ panel contains parameters for controlling dB0 field display.

- Colormap : The colormap for the dB0 field, includes ‘Jet’, ‘Gray’ and ‘Hot’
- Plane : The flag for activating dB0 field slicing plane, includes ‘XY’, ‘XZ’ and ‘YZ’
- XDimRes (m) : The displayed spatial resolution in X direction

- YDimRes (m) : The displayed spatial resolution in Y direction
- ZDimRes (m) : The displayed spatial resolution in Z direction

MRiLab provides a group of dB0 field display button to help inspecting the dB0 field details. The dB0 field display button group consists of:

- S+ : Move active dB0 field slicing plane forwards
- S- : Move active dB0 field slicing plane backwards

### 5.4.2 Magnet Element Macro Library

A magnet element macro is a predefined module for creating a magnet element that generates dB0 field in three dimensional space. MRiLab magnet element macro library is a collection of magnet element macros. This section will give an introduction for each of the magnet element macro provided in MRiLab.

#### **MagLinear**

A magnet element macro that creates a linear dB0 field. This macro contains attributes including:

- GradZ (T/m) : The linear gradient of dB0 field in Z direction
- GradY (T/m) : The linear gradient of dB0 field in Y direction
- GradX (T/m) : The linear gradient of dB0 field in X direction
- Scale : The scale factor for the dB0 field

#### **MagGaussian**

A magnet element macro that creates a 3D Gaussian dB0 field. This macro contains attributes including:

- PosZ (m) : The Z position of the center
- PosY (m) : The Y position of the center
- PosX (m) : The X position of the center
- DeltaZ (m) : The width of Gaussian function in Z direction
- DeltaY (m) : The width of Gaussian function in Y direction
- DeltaX (m) : The width of Gaussian function in X direction
- Scale : The scale factor for the dB0 field

### **MagSymbolic**

A magnet element macro that creates a dB0 field based on symbolic equation. This macro contains attributes including:

- Equation : An dB0 field equation

The symbolic equation could be any legal Matlab equation using variables ‘X’, ‘Y’ and ‘Z’. For example, ‘X+Y’, ‘2\*X.\*Y’ and ‘2\*sin(X)’ etc. Notice that use element operations for variables in the equation. And the user needs to fill the equation between a pair of single quotes.

### **MagUser**

If the user has the dB0 field data saved in a MAT file, the user can easily import the dB0 field into MRiLab magnet design interface by using ‘MagUser’ macro. The dB0 field MAT file needs to contain one matrix ‘dB0’. The ‘MagUser’ macro contains attributes including:

- MagFile : The path to the file that stores the dB0 field data, quoted using single quotes
- Interp : The interpolation method, includes ‘linear’, ‘nearest’, ‘spline’ and ‘cubic’

### **5.4.3 Make Your Own Magnet**

The user can also use the magnet loading function to load another magnet. To load a magnet, click ‘Load’ menu then click ‘Load Magnet File’, choose a magnet XML file.

The user can create a new magnet in the Magnet Design interface. To create a new magnet, click ‘New’ then click ‘Create Magnet File’. A magnet creation window will show up and ask for new magnet name and notes. To follow MRiLab naming convention, the user is recommended to use ‘Mag\_’ followed by a legal string and applied anatomy (e.g. Mag\_CustomHead), make sure that the new magnet name is distinct to the remaining magnet names in MRiLab. Then click ‘OK’ to select a path for storing the magnet XML file. It’s strongly recommended to put the magnet under the MRiLab magnet root folder /Mag according to the magnet type so that the magnet is visible to MRiLab. Finally, MRiLab will create a new magnet XML file based on the content of ‘Mag\_LinearHead’.

## **5.5 Gradient Design**

The Gradient Design toolbox can be activated by pressing ‘Gradient Design Panel’ toolbar icon located at the top of the main simulation console. The loaded gradient will show in the Gradient Design interface.



Figure 5.41: Gradient Design Panel Toolbar Icon

### 5.5.1 Gradient Design GUI

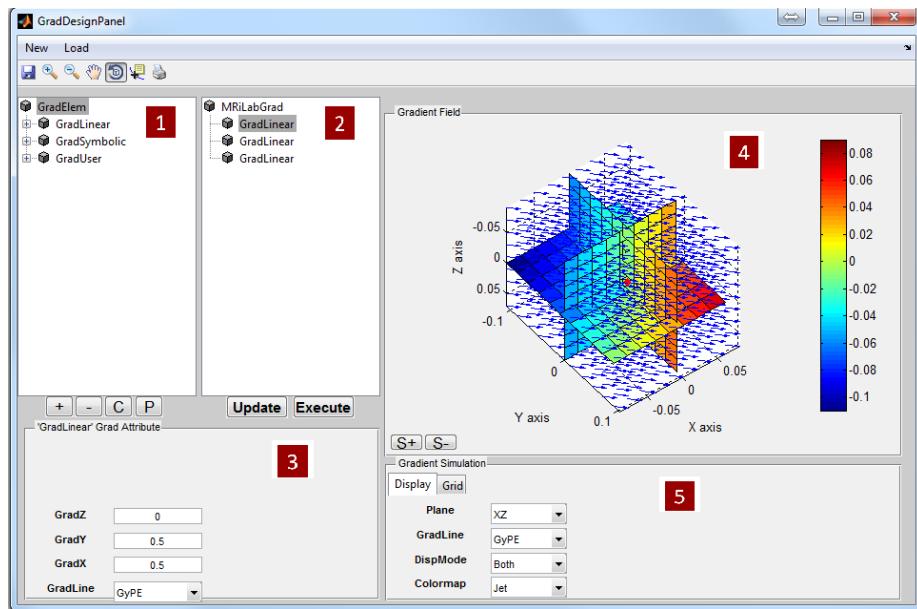


Figure 5.42: Gradient Design GUI

Figure 5.42 demonstrates the overview of the Gradient Design interface. This interface consists of

1. Gradient Element Macro Library

The Gradient Element Macro Library contains gradient element macros for constructing gradient structure in MRILab. The user needs to click the 'GradElem' root to unfold subsequent macros.

2. Gradient Structure

In MRILab, a gradient structure consists of three gradient elements that are combined to create gradient field for GzSS, GyPE and GxR. The user can construct desired gradient field by changing the content within the gradient structure. To add a macro into the gradient structure, the user needs to click one macro in the macro library, then click on the gradient

structure root (i.e. MRiLabGrad) to which this macro is inserted, then click ‘+’ macro operation button. To delete a macro from the gradient structure, the user needs to click the unwanted macro, then click ‘-’ macro operation button. To duplicate an existing macro, the user needs to first click the source macro, then click ‘C’ macro operation button for copying, click on the gradient structure root, then click ‘P’ macro operation button for pasting. MRiLab doesn’t allow empty gradient structure, and requires that one gradient field must be assigned to each of the gradient sequence line.

### 3. Gradient Element Attribute

When clicking on the gradient element macro within the gradient structure, the corresponding macro attributes will be updated at the gradient element attribute panel down below the gradient structure. The user can edit those attributes to modify the gradient element so as to change generated gradient field. To make the modification effective, the user must press ‘Update’ button to update the gradient file. Pressing ‘Execute’ button will update and redraw the gradient field map on this interface.

### 4. Gradient Field

The gradient field is represented as a three-dimensional quiver plot on the ‘Gradient Field’ panel. If the constant unit gradient is used, the regular linear spatial location is applied. However, for example if non-unit gradient is used, the spatial location may be either dilated or shrunk along the applied direction. This means the original spatial location will be mapped to a new location in the spatial grid. The transformed spatial grid is represented as three slicing planes on the ‘Gradient Field’ panel. The value (color) of the spatial grid equals to the spatial location in the direction that is indicated by ‘GradLine’ in ‘Gradient Simulation’ panel.

### 5. Gradient Simulation Control

The ‘Display’ and ‘Grid’ tab on the ‘Gradient Simulation’ panel contains parameters for controlling gradient field display.

- Colormap : The colormap for the spatial grid, includes ‘Jet’, ‘Gray’ and ‘Hot’
- Plane : The flag for activating grid slicing plane, includes ‘XY’, ‘XZ’ and ‘YZ’
- GradLine : The flag for activating gradient sequence line, includes ‘GzSS’, ‘GyPE’ and ‘GxR’
- DispMode : The display mode, includes ‘Gradient’, ‘Grid’ and ‘Both’
- XDimRes (m) : The displayed spatial resolution in X direction

- YDimRes (m) : The displayed spatial resolution in Y direction
- ZDimRes (m) : The displayed spatial resolution in Z direction

MRiLab provides a group of grid display button to help inspecting the grid details. The grid display button group consists of:

- S+ : Move active grid slicing plane forwards
- S- : Move active grid slicing plane backwards

### 5.5.2 Gradient Element Macro Library

A gradient element macro is a predefined module for creating a gradient element that generates gradient field in three dimensional space. MRiLab gradient element macro library is a collection of gradient element macros. This section will give an introduction for each of the gradient element macro provided in MRiLab.

#### **GradLinear**

A gradient element macro that creates a linear gradient field. This macro contains attributes including:

- GradZ : A constant of gradient field vector in Z direction, 1 for unit gradient
- GradY : A constant of gradient field vector in Y direction, 1 for unit gradient
- GradX : A constant of gradient field vector in X direction, 1 for unit gradient
- GradLine : The gradient sequence line that is assigned to this gradient field

#### **GradSymbolic**

A gradient element macro that creates a gradient field based on symbolic equation. This macro contains attributes including:

- GradZEQU : An equation for gradient field vector in Z direction
- GradYEQU : An equation for gradient field vector in Y direction
- GradXEQU : An equation for gradient field vector in X direction
- GradLine : The gradient sequence line that is assigned to this gradient field

The symbolic equation could be any legal Matlab equation using variables ‘X’, ‘Y’ and ‘Z’. For example, ‘X+Y’, ‘2\*X.\*Y’ and ‘2\*sin(X)’ etc. Notice that use element operations for variables in the equation. And the user needs to fill the equation between a pair of single quotes.

### GradUser

If the user has the gradient field data saved in a MAT file, the user can easily import the gradient field into MRiLab gradient design interface by using ‘GradUser’ macro. The gradient field MAT file needs to contain one four dimensional matrix ‘G’ with the size of the fourth dimension to be 3.  $G(:,:,1)$  is the  $x$  component of the gradient vector,  $G(:,:,2)$  is the  $y$  component of the gradient vector and  $G(:,:,3)$  is the  $z$  component of the gradient vector. The ‘GradUser’ macro contains attributes including:

- GradFile : The path to the file that stores the gradient field data, quoted using single quotes
- Interp : The interpolation method, includes ‘linear’, ‘nearest’, ‘spline’ and ‘cubic’
- GradLine : The gradient sequence line that is assigned to this gradient field

### 5.5.3 Make Your Own Gradient

The user can also use the gradient loading function to load another gradient. To load a gradient, click ‘Load’ menu then click ‘Load Gradient File’, choose a gradient XML file.

The user can create a new gradient in the Gradient Design interface. To create a new gradient, click ‘New’ then click ‘Create Gradient File’. A gradient creation window will show up and ask for new gradient name and notes. To follow MRiLab naming convention, the user is recommended to use ‘Grad\\_’ followed by a legal string and applied anatomy (e.g. Grad\_CustomHead), make sure that the new gradient name is distinct to the remaining gradient names in MRiLab. Then click ‘OK’ to select a path for storing the gradient XML file. It’s strongly recommended to put the gradient under the MRiLab gradient root folder /Grad according to the gradient type so that the gradient is visible to MRiLab. Finally, MRiLab will create a new gradient XML file based on the content of ‘Grad\_LinearHead’.

## 5.6 Motion Design

The Motion Design toolbox can be activated by pressing ‘Motion Design Panel’ toolbar icon located at the top of the main simulation console.



Figure 5.43: Motion Design Panel Toolbar Icon

### 5.6.1 Motion Design GUI

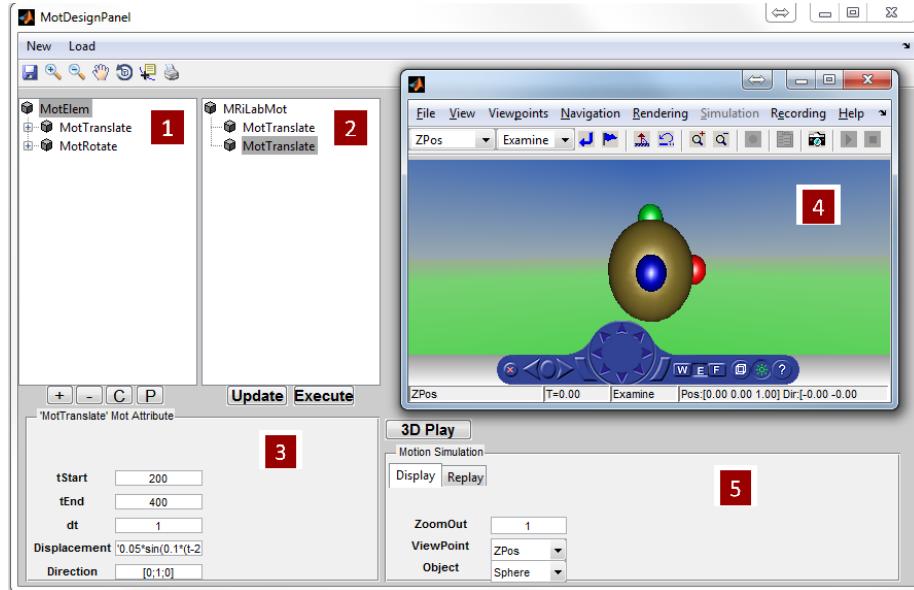


Figure 5.44: Motion Design GUI

Figure 5.44 demonstrates the overview of the Motion Design interface. This interface consists of

1. Motion Element Macro Library

The Motion Element Macro Library contains motion element macros for constructing motion structure in MRiLab. The user needs to click the 'MotElem' root to unfold subsequent macros.

2. Motion Structure

In MRiLab, a motion structure consists of arbitrary number of motion elements that are combined to create desired motion pattern. The user can construct wanted motion pattern by changing the content within the motion structure. To add a macro into the motion structure, the user needs to click one macro in the macro library, then click on the motion structure root (i.e. MRiLabMot) to which this macro is inserted, then click '+' macro operation button. To delete a macro from the motion structure, the user needs to click the unwanted macro, then click '-' macro operation button. To duplicate an existing macro, the user needs to first click the source macro, then click 'C' macro operation button for copying,

click on the motion structure root, then click ‘P’ macro operation button for pasting. MRiLab doesn’t allow empty motion structure.

### 3. Motion Element Attribute

When clicking on the motion element macro within the motion structure, the corresponding macro attributes will be updated at the motion element attribute panel down below the motion structure. The user can edit those attributes to modify the motion element so as to change generated motion pattern. To make the modification effective, the user must press ‘Update’ button to update the motion file. Pressing ‘Execute’ button will update and recalculate the motion track.

### 4. Motion Tracker

MRiLab uses Matlab Simulink 3D Animation toolbox to investigate the motion track of the object in the motion tracker window. To open this window, the user needs to press ‘3D Play’ button. In the 3D animation, the image object is represented using a sphere, the three attached small spheres indicating axis directions (red for  $x$  axis, green for  $y$  axis and blue for  $z$  axis).

### 5. Motion Simulation Control

The ‘Display’ and ‘Replay’ tab on the ‘Motion Simulation’ panel contains parameters for controlling 3D animation.

- Object : The object model, currently only supports ‘Sphere’
- ViewPoint : The default view point, includes ‘ZPos’, ‘YPos’ and ‘XPos’
- ZoomOut : The factor for view zoom out
- Sample : The sample steps between two adjacent positions during motion
- Repeat : The repeat time for playback

#### 5.6.2 Motion Element Macro Library

A motion element macro is a predefined module for creating a motion element that generates a motion track in three dimensional space. MRiLab motion element macro library is a collection of motion element macros. This section will give an introduction for each of the motion element macro provided in MRiLab.

### **MotTranslate**

A motion element macro that creates translation motion. This macro contains attributes including:

- tStart (s) : The motion starting time
- tEnd (s) : The motion ending time
- dt (s) : The time interval of the motion track sample steps
- Direction : A vector describing translation direction in 3D space
- Displacement (m) : An equation of translation displacement pattern with respect to time

The displacement equation could be any legal Matlab equation using variables ‘t’. For example, ‘ $2*t$ ’, ‘ $t+200e-3$ ’ and ‘ $0.05*\sin(0.1*t)$ ’ etc. Notice that the user needs to fill the equation between a pair of single quotes.

### **MotRotate**

A motion element macro that creates rotation motion. This macro contains attributes including:

- tStart (s) : The motion starting time
- tEnd (s) : The motion ending time
- dt (s) : The time interval of the motion track sample steps
- Axis : A vector describing rotation axis in 3D space
- Angle (rad) : An equation of rotation angle with respect to time

The rotation angle equation could be any legal Matlab equation using variables ‘t’. For example, ‘ $2*t$ ’, ‘ $t+200e-3$ ’ and ‘ $\sin(0.1*t)$ ’ etc. Notice that the user needs to fill the equation between a pair of single quotes.

### **5.6.3 Make Your Own Motion**

The user can also use the motion loading function to load another motion. To load a motion, click ‘Load’ menu then click ‘Load Motion File’, choose a motion XML file.

The user can create a new motion in the Motion Design interface. To create a new motion, click ‘New’ then click ‘Create Motion File’. A motion creation window will show up and ask for new motion name and notes. To follow MRi-Lab naming convention, the user is recommended to use ‘Mot\_’ followed by a legal string and applied anatomy (e.g. Mot\_CustomHead), make sure that the

new motion name is distinct to the remaining motion names in MRiLab. Then click ‘OK’ to select a path for storing the motion XML file. It’s strongly recommended to put the motion under the MRiLab motion root folder /Mot according to the motion type so that the motion is visible to MRiLab. Finally, MRiLab will create a new motion XML file based on the content of ‘Mot\_ShiftHead’.

The user needs to notice that adding a motion pattern is not guaranteed to stimulate motion, the user also needs to use the extended real time process to trigger motion at the Ext sequence line. To trigger motion, one or more Ext flag 8 needs to be inserted into Ext line. A better motion tracking can be achieved with both small time interval of the motion track sample steps and frequent motion triggering, but requires more simulation computation and time. The interested users are referred to PSD\_GRE3D for more information.

## 5.7 Image Reconstruction

MRiLab provides default image reconstruction code for a few types of Cartesian and Non-Cartesian reconstruction. External reconstruction code is also acceptable.

### 5.7.1 Default Engine

To choose default reconstruction for Cartesian readout, the user needs to choose ‘Cartesian’ for ‘ReconType’ under the ‘Recon’ tab. The Cartesian reconstruction can be applied to typical Cartesian readout, FSE readout and EPI readout that are performed using default gradient macros. To choose default reconstruction for Non-Cartesian readout, the user needs to choose ‘NonCart’ for ‘ReconType’. The corresponding Non-Cartesian reconstruction code can be applied to radial readout and spiral readout. Notice that the Non-Cartesian reconstruction performs 2D gridding process for the Non-Cartesian K-Space trajectory using a Kaiser-Bessel kernel, followed by a regular iFFT. The 2D gridding process should also in theory be able to be applied to other Non-Cartesian readout. In order to perform Non-Cartesian reconstruction, the ‘gridding’ tab is also required in MRiLab.

### 5.7.2 External Engine

If the user designed a special K-Space trajectory that requires particular reconstruction code, the user needs to indicate using external reconstruction by changing ‘ReconEng’ to ‘External’, and then provides a reconstruction function quoted with a pair of single quotes for ‘ExternalEng’. It is recommended to put the external code under /Recon/External, if not, make sure the reconstruction code is in Matlab searching path. This setting will simply ignore default reconstruction code and apply external code. It is strongly recommended to write your own reconstruction code based on the template :

```

function DoCustomRecon
%Create an external reconstruction based on your code

global VCtl      % use VCtl structure, read only
global VSig      % use VSig structure, read only
global VIImg     % use VIImg structure, read and write

...
% The main code for your reconstruction
VIImg.Real = ...;
VIImg.Imag = ...;
VIImg.Mag = ...;
VIImg.Phase = ...;
VIImg.Sx = ...;
VIImg.Sy = ...;
...
end

```

Notice that there are two new Virtual Structures (VSig and VIImg) in this template, VSig encapsulates acquired MR signal. VIImg is declared to store reconstructed images. For the structure ‘VSig’, MRiLab provides :

- VSig.Sx : A array storing  $x$  component of acquired MR signal
- VSig.Sy : A array storing  $y$  component of acquired MR signal
- VSig.Kz (1/m) : A array storing Kz location of K-Space trajectory
- VSig.Ky (1/m) : A array storing Ky location of K-Space trajectory
- VSig.Kx (1/m) : A array storing Kx location of K-Space trajectory

The sample points in these arrays are organized in the order of

Sample points in one readout < Multiple echos < First phase encoding <  
Second phase encoding < Coil channel < Spin species

For the structure ‘VIImg’, MRiLab provides :

- VIImg.Real : A matrix for real component of reconstructed complex image with a size of  $RYDim \times RXDim \times RZDim \times RxCoilNum \times TEPerTR$
- VIImg.Imag : A matrix for imaginary component of reconstructed complex image with a size of  $RYDim \times RXDim \times RZDim \times RxCoilNum \times TEPerTR$
- VIImg.Mag : A matrix for magnitude of reconstructed complex image with a size of  $RYDim \times RXDim \times RZDim \times RxCoilNum \times TEPerTR$

- VIImg.Phase : A matrix for phase of reconstructed complex image with a size of  $RYDim \times RXDim \times RZDim \times RxCoilNum \times TEPerTR$
- VIImg.Sx : A matrix for  $x$  (real) component of sorted complex MR signal in Cartesian grid
- VIImg.Sy : A matrix for  $y$  (imaginary) component of sorted complex MR signal in Cartesian grid

where  $RXDim$ ,  $RYDim$  and  $RZDim$  are reconstructed image resolution in three spatial space.

### Gadgetron

If the user are interested to use Gadgetron for image reconstruction, MRiLab also provides a simple MEX code to convert the acquired data into ISMRMRD file that can be used in Gadgetron process. To activate this function, the user needs to install ISMRMRD dependency packages to enable properly compiling MEX. I am working on improving MRiLab support for Gadgetron since Gadgetron is a very promising generic image reconstruction framework. I will be very pleased to talk with the interested users who are willing to contribute to improving MRiLab Gadgetron support. Please don't hesitate to contact me. The ISMRMRD relevant code is ‘DoToHDF5.m’ under /Src and ‘DoMatToHDF5’ under /Lib/src/interface.

## 5.8 Simulation Output

MRiLab can save simulation output into two file formats including ‘MAT’ and ‘ISMRMRD’. The user can choose ‘OutputType’ under ‘Recon’ tab. If the user choose ‘ISMRMRD’ format, the ‘DoMatToHDF5’ MEX has to be functional, if not, MAT file will be saved instead. MRiLab saves simulation output of each series into a folder named by the MRiLab startup time under /Output folder. For MAT file, MRiLab saves:

- SeqXMLFile : Applied sequence XML file
- SeriesName : Series name on the main control console
- VCtl : Virtual Control structure
- VIImg : Virtual Image structure
- VSig : Virtual Signal structure

# Chapter 6

## Image Display and Analysis

MRiLab not only provides a graphic user interface for image simulation, but also incorporates a set of image display and analysis tools that are easy to use and powerful in functionality. The user requires no needs to install any other software packages and to switch between developing environment. These image display and analysis tools are also designed under Matlab, and carefully tuned for manipulating image data that are generated from MRiLab.

### 6.1 MatrixUser

MRiLab incorporated a toolbox named ‘MatrixUser’ for performing image display and analysis. MatrixUser is originally designed to let users manipulating multi-dimensional matrix in Matlab in a graphic way. This toolbox can be activated by pressing ‘MatrixUser’ toolbar icon located at the top of the main simulation console.



Figure 6.1: MatrixUser Toolbar Icon

MatrixUser will search through the current output folder and load all image series from this folder into Matlab base workspace. The MatrixUser main window (Figure 6.2) opens up as a matrix manager for loaded images. The user can choose to display the image series by using the popup menu. The matrix size, type and value range are calculated and provided on the right side. The user can press ‘MatrixUser’ button to activate MatrixUser display window. Current version of MatrixUser supports displaying any valid Matlab multi-dimensional matrix and Matlab structure variable.

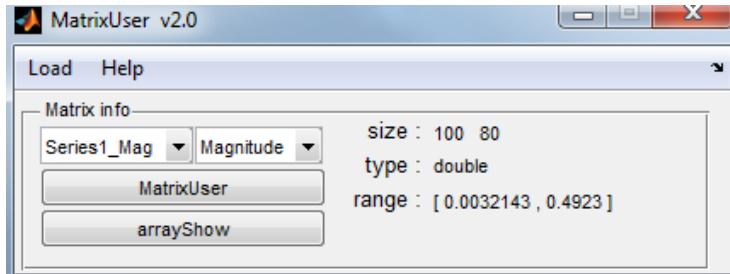


Figure 6.2: MatrixUser Main Window

### 6.1.1 Data Import

MatrixUser supports valid Matlab matrix. By default, MatrixUser reads the Matlab base workspace, and scans any matrices that are currently existing in the Matlab session, then creates a matrix list for tracking their content. Once these matrices are updated, MatrixUser will also update the matrix list. In addition, there are several different approaches to import data from outside Matlab into MatrixUser. The imported matrices will be saved into Matlab base workspace. The import functions are located under ‘Load’ menu, including:

- Load MAT file  
The default Matlab .mat file is natively supported by MatrixUser.
- Load System Clipboard  
If image content exists in the system clipboard, it can be converted into a RGB image that contains a three slice matrix with each slice corresponds to the individual Red, Green and Blue channel.
- Load ScreenShot  
MatrixUser takes a full screenshot for current monitor and saves it into a RGB image as described above.
- Load from Binary file  
Binary data file is supported by MatrixUser. The user needs to properly configure loading parameters (Figure 6.3) according to the matrix size and data type information.
- Load DICOM file(s)  
MatrixUser supports loading multiple DICOM files by using a file filter interface (Figure 6.4). The user needs to load DICOM files into the loading interface by selecting wanted DICOM files (multiple selection supported) from a data folder. The selected files are listed in the DICOM file list. The user can click a single file to read its DICOM header and image preview. To manually create a matrix using DICOM files, the user needs to choose

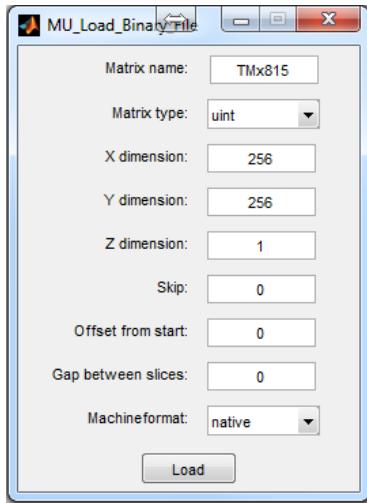


Figure 6.3: MatrixUser Binary File Loading Interface

files from the left list, then press ‘>>>>>’ to push the files into selected DICOM file list. A matrix name must also be provided. Then pressing ‘Convert’ button will create a matrix based on chosen DICOM files. The user can load these created matrices into workspace by pressing ‘Load matrix’ button.

- Load DICOM file(s) in Batch

MatrixUser also supports loading DICOM files in a batch mode from a data folder. This function requires the folder path is provided. MatrixUser will try to create separate matrices for DICOM files that may come from different image series. A matrix selection interface will provides converted matrices that can be loaded into Matlab workspace.

- Load from NIfTI file

NIfTI file with .nii suffix is supported by MatrixUser.

### 6.1.2 Window Layout

The user can press ‘MatrixUser’ button to activate MatrixUser display window. If the selected matrix contains complex value, four options are available for displaying magnitude, phase, real and imaginary of the matrix. Figure 6.5 demonstrates an overview of the window layout of the MatrixUser display window. The window consists of

1. Matlab Default Toolbar

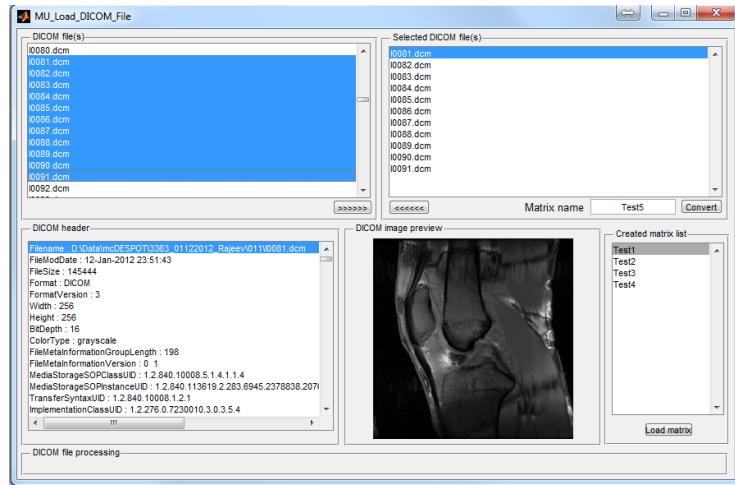


Figure 6.4: MatrixUser DICOM File Loading Interface

Matlab toolbar provides basic functions for analyzing displayed matrix. These functions include:

- : Save current image axes into an image file
- : Zoom in matrix area
- : Zoom out matrix area
- : Manually move matrix position
- : Check individual voxel value and index
- : Print current figure
- : Turn on/off color bar

## 2. MatrixUser Function Library

Most of the matrix analysis functions are represented under function bench panel. MatrixUser groups these functions into categories and dynamically loads them according to the size and compatibility of current displayed matrix. A multi-tab is used to contain individual function button that corresponds to each function. The tabs under the multi-tab are used to switch between function categories, which include

- Display : Multi-dimensional matrix display relevant functions

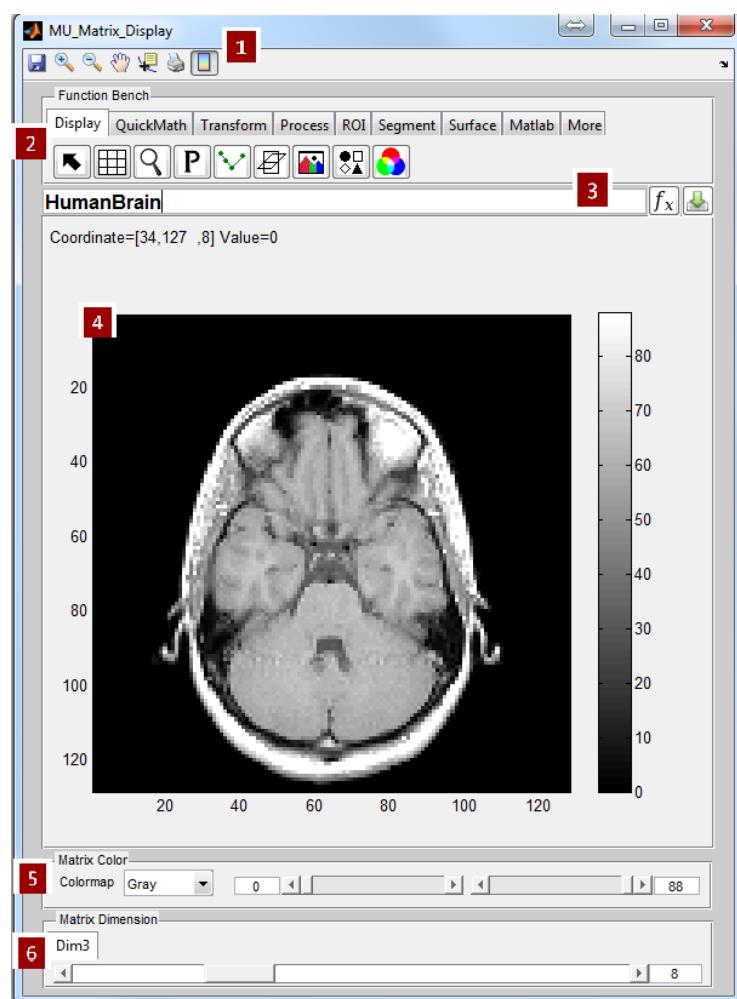


Figure 6.5: MatrixUser Display Window

- QuickMath : Perform math calculation for current matrix
- Transform : Transform current matrix
- Process : Basic matrix processing functions
- ROI : Region-of-Interest relevant functions
- Segment : Segmentation relevant functions
- Surface : Generate surface or mesh plot for current image
- Matlab : Matlab default image tools
- More : Uncategorized functions

### 3. Matrix Calculator

The matrix calculator consists of three control units, including a matrix expression editbox, an execution button (  ) and a matrix saving button (  ). Any valid matrix calculation expression can be edited and executed in the calculator and updated in the display window, which serves as a convenient way for analyzing matrix calculation result. Matrix concatenation and recombination can also be done in the calculator, for example, to side-by-side compare multiple 3D matrices (Figure 6.6). Some valid calculation examples are, but not limited to:

- A,B Combines A and B in one row
- A,B;C,D Combines A and B in the first row, C and D in the second row
- A,B;C,zeros(size(C)) Combines A and B in the first row, C in the second row, pad with zero value
- sin(A),cos(B) Calculates voxel-wise sin for A and cos for B, then combine them in one row
- A(:,1:10,:) Extracts submatrix from A and display it

Where A, B, C and D are multi-dimensional matrices with proper matrix size in order to maintain valid Matlab matrix calculation rule as in above examples. Also note that all the source matrices have to exist in the workspace. Pressing the execution button will perform the calculation and save the result as a temporary matrix. The user can also save the temporary matrix into workspace by pressing matrix saving button. The saved temporary matrix will have a ‘\_tmp’ suffix by default.

### 4. Matrix Display Axes

One slice of current matrix is displayed in the display axes. The user can use mouse cursor to inspect the coordinate and value of any voxel. Moving mouse wheel back and forth moves the slice location in the current dimension and updates the display axes.

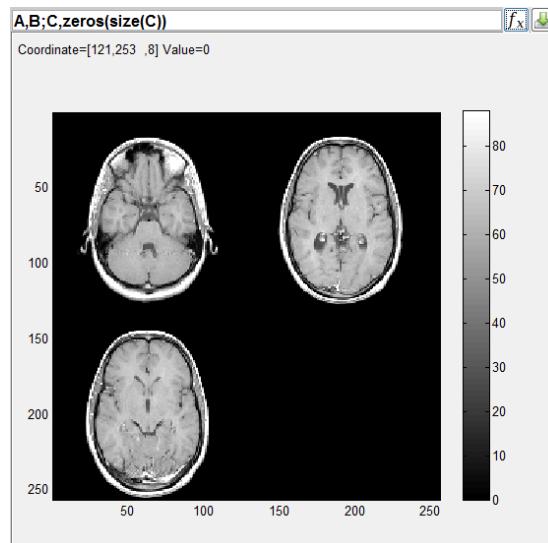


Figure 6.6: MatrixUser Concatenation Example

#### 5. Matrix Color Control Group

The matrix color control group provides a set of sliders, editboxes and popup menu that help control image color map and contrast. This group consists of

- Colormap Popup Menu: Choose different colormap scheme
- Upper Color Bound Slider (right slider): Slide to control the upper bound of color limits
- Upper Color Bound EditText (right editbox): Edit to control the upper bound of color limits
- Lower Color Bound Slider (left slider): Slide to control the lower bound of color limits
- Lower Color Bound EditText (left editbox): Edit to control the lower bound of color limits

#### 6. Matrix Dimension Control Group

MatrixUser measures the size of the displayed matrix and assigns one slider and editbox for each dimension that is above 2 (x and y). These control units are located in individual dimension tab and can be used to switch among slices in current active dimension.

### 6.1.3 Function Library

#### 1. Display

Matrix display relevant functions are listed under this tab.

- : Reset matrix display and erase additional display effect
- : Turn on and off black grid line on the image display axes
- : Open an instant magnifier which zooms in specific image area
- : Plot and update a profile curve along a resizable checking line (Figure 6.7)

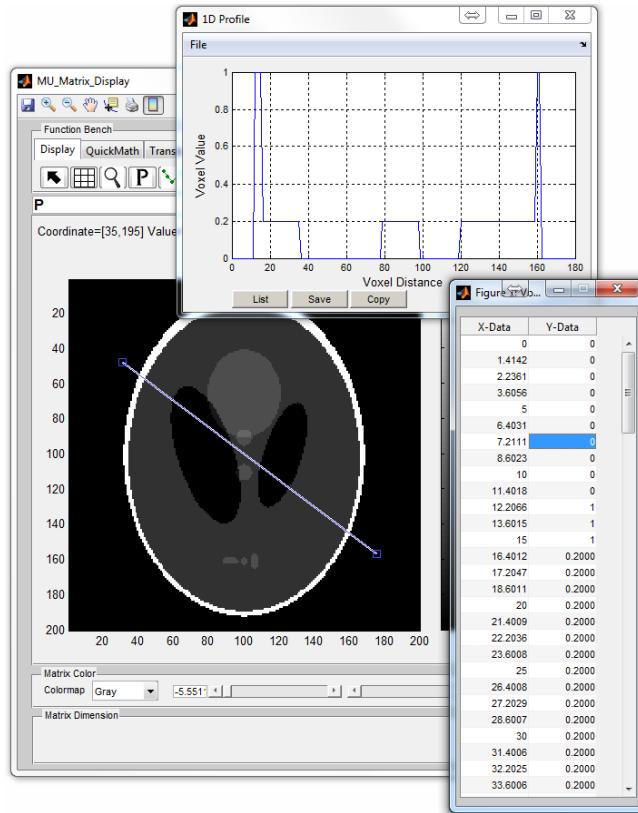


Figure 6.7: An Example of 1D Profile. The user can relocate and resize the profile checking line on the image for inspecting live 1D profile. The buttons on the profile window can list profile data, save the data as ‘data\_plt’ array into workspace or copy the data into clipboard.

- : Plot and update a profile curve along one given matrix dimension (Figure 6.8)

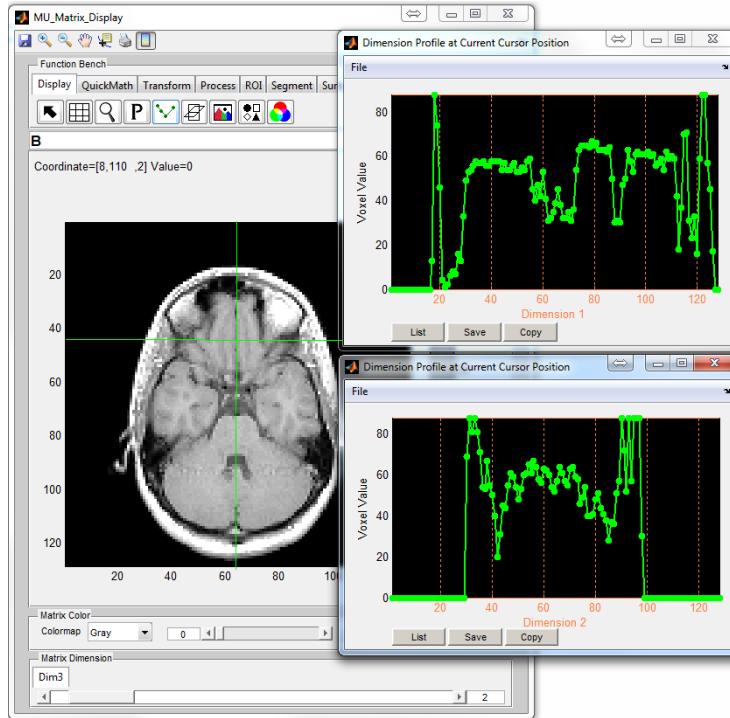


Figure 6.8: An Example of Plotting the First (Column) and Second (Row) Matrix Dimension Profile. The user needs to specify which matrix dimension to plot. The profile curve will update according to current mouse cursor position. The user can pause or resume live updating by right click on main display window.

- : Open a separate window for another two orthogonal matrix display axes (Figure 6.9). The operation buttons on the second window consists of
  - : Activate or deactivate localizer line on main display
  - : Activate or deactivate localizer line on main display
  - : Switch matrices between main display and second display. This operation simply permutes current matrix into its orthogonal version. The user can save the transformed matrix using ().
- : Create a RGB Image, assign current slice and following two slices to Red, Green and Blue channel, respectively

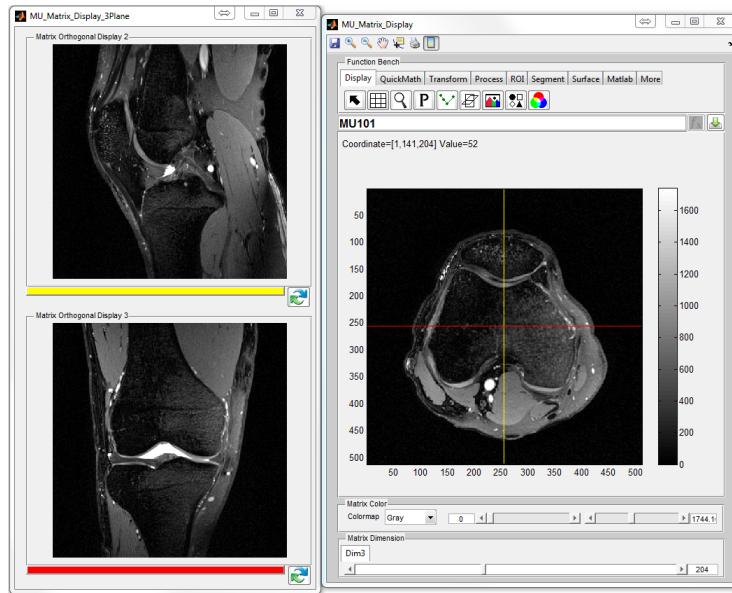


Figure 6.9: 3D Slicer. To change view slice, the user needs to activate localizer line and then click on wanted coordinate on the main display window.

- : Create a montage image using multiple slices (Figure 6.10)
- : Overlap two matrices that have the same matrix size (Figure 6.11), notice that the user can press to remove foreground matrix from overlapping with background matrix.

## 2. QuickMath

This function category performs quick math calculation for current matrix. A few commonly used math calculation are provided under this tab. Instead, complex calculation can be performed using matrix calculator as mentioned above.

- : Calculate absolute value
- : Calculate negative value
- : Calculate natural logarithm
- : Calculate common (base 10) logarithm
- : Calculate exponential
- : Calculate the power of 10

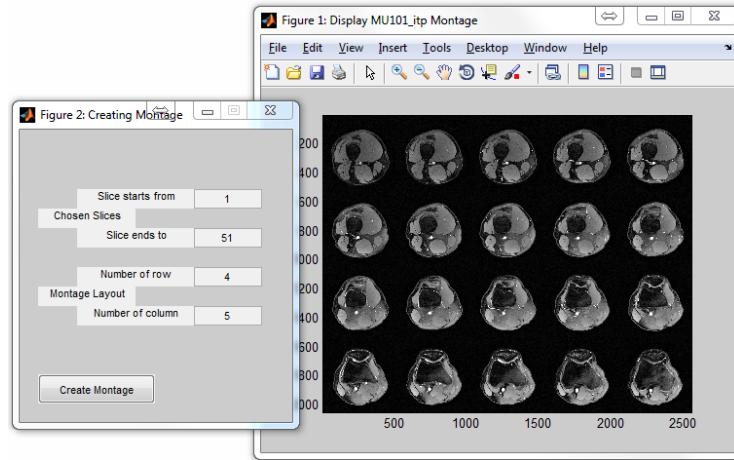


Figure 6.10: Montage Image. An example of creating a 4-by-5 montage image using multiple matrix slices starting from the first slice.

- : Calculate sine
- : Calculate cosine
- : Calculate tangent
- : Calculate inverse sine
- : Calculate inverse cosine
- : Calculate inverse tangent

### 3. Transform

This function category performs spatial transformation or fast Fourier transform (FFT) to current matrix.

- : Flip matrix horizontally (along the first dimension)
- : Flip matrix vertically (along the second dimension)
- : Flip matrix along slice direction (the third dimension)
- : Rotate matrix 90 degree in the counter clockwise direction
- : Rotate matrix 90 degree in the clockwise direction
- : Rotate matrix certain degree along an axis in the 3D space, which is specified using the rotation axis origin and direction.

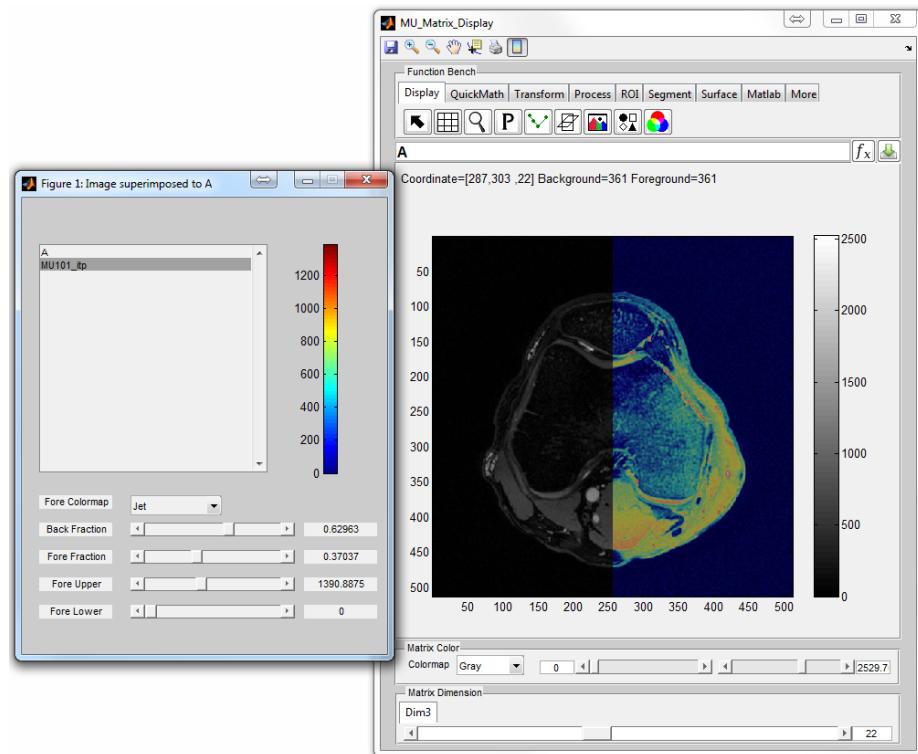


Figure 6.11: An Example of Overlapping Two Matrices. A second window is provided for adjusting overlapping effect. The user can switch to overlap any size compatible matrices from the matrix list. There are also control units to change foreground colormap scheme, to modify image intensity fraction, or to change foreground color limits. The matrix values at current cursor for both background and foreground matrices are updated simultaneously at main display.

-  : Translate matrix along certain direction
-  : Perform multi-dimensional FFT for current matrix, the user needs to specify up to which dimension to perform FFT.

#### 4. Process

This function category performs basic matrix processing functions.

-  : Create binary mask according to given threshold
-  : Perform smoothing operation to current matrix
-  : Perform sharpening operation to current matrix
-  : Perform edge detection to current matrix
-  : Provides various image filters (Figure 6.12)

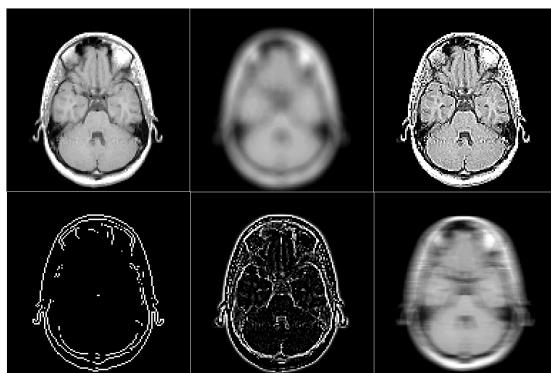


Figure 6.12: Image Filters. An example is shown for applying various image filters.

-  : Add noise to matrix (Figure 6.13)

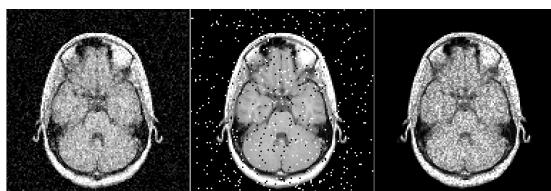


Figure 6.13: Image Noise. An example is shown for applying various image noise.

- : Replace voxel value for the voxels within certain value range and outside a polygon area. The user needs to draw a polygon first (double click to confirm the polygon).
- : Replace voxel value for the voxels within certain value range and inside a polygon area. The user needs to draw a polygon first (double click to confirm the polygon).
- : Replace voxel value for the voxels inside a polygon area. The user needs to draw a polygon first (double click to confirm the polygon).
- : Replace matrix area with a selected matrix source region. The user needs to draw a free-hand area (source region), drag the free-hand region to target matrix area, then double click to confirm the operation.
- : Crop current matrix using a rectangle box (double click to confirm the cropping)
- : Extract parts of current matrix using irregular shape (double click to confirm the extracting)
- : Resample current matrix by using chosen interpolation method (Figure 6.14). The user can specify sampling factors in x, y and z direction for 3D matrix.

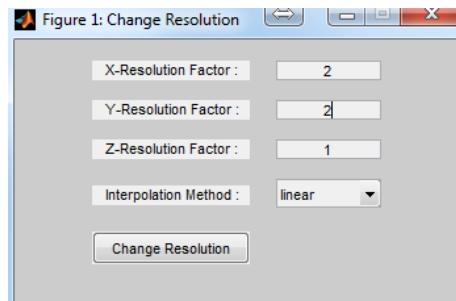


Figure 6.14: Interpolation Window. An example is shown for doubling resolution in x and y direction by using linear interpolation.

## 5. ROI

MatrixUser provides a set of function buttons for performing Region-of-Interest (ROI) analysis (Figure 6.15). To create a ROI, the user needs to click ROI button first, then draw a ROI on the image axes. The statistical measures (i.e. mean, standard deviation and relative standard deviation) for voxels in delineated ROI is calculated and updated with moving ROI position or changing ROI shape. The ROI function buttons consists of

- : Draw a free hand ROI
- : Draw a rectangle or square ROI
- : Draw a circle or ellipse ROI
- : Draw a polygon ROI
- : Draw a straight line for measuring distance in units of pixels
- : Draw a polygon for measuring the interior angles in degrees
- : Record existing ROI shape and location into a ROI list, allow to redraw selected ROI in multiple image axes. To redraw a existing ROI, the user needs to select a ROI in the list, then press 'Show' button. Note that the copied ROI is no longer resizable and movable.
- : Plot histogram for current slice (Figure 6.16); if ROIs exist, plot histogram for latest activated ROI

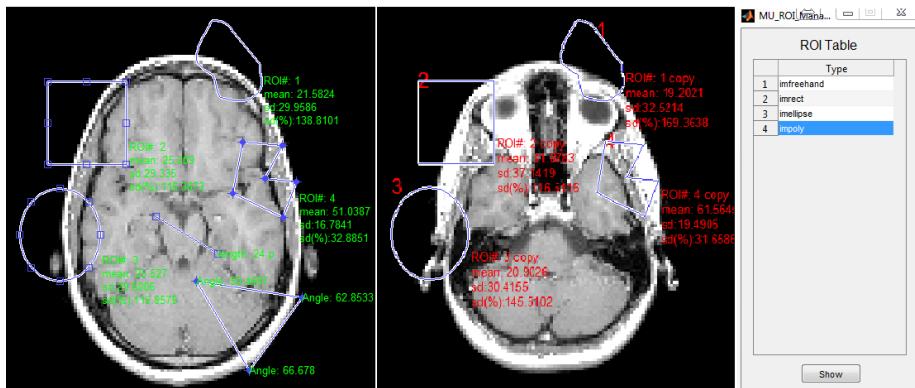


Figure 6.15: Draw Multiple ROIs and Redraw on A Second Image. The source ROIs are in green, copied ROIs are in red.

## 6. Segment

MatrixUser supports functions for performing manual segmentation, editing, saving and loading segmented regions for multiple slices. To create a segmentation, the user needs to click segmentation button first, then draw a region on the image axes. After finishing drawing, the user can modify the region location and shape before double clicking the region to confirm segmentation. The segmentation buttons consists of

- : Do a free-hand segmentation
- : Do a circle or ellipse segmentation

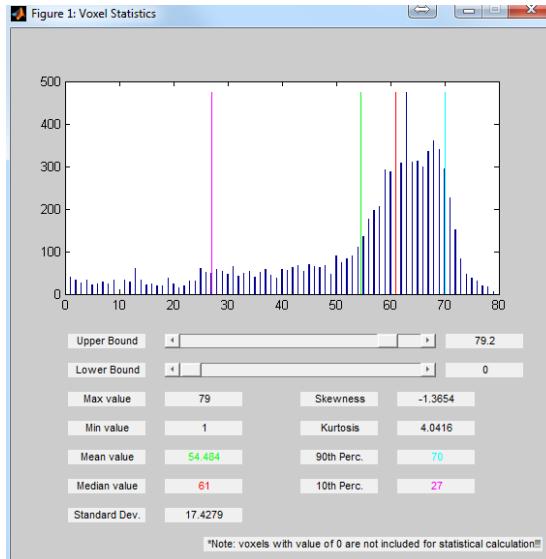


Figure 6.16: Histogram for One Image Slice

- : Do a polygon segmentation
- : Edit segmentation
- : Save segmentation into a MAT file
- : Load segmentation from a MAT file

To edit segmented region (Figure 6.17), the user needs to press to open segmentation manager. The manager will automatically detect the type and location of segmented regions. The user can click any region item to inspect the location of the region. To edit chosen region, the user must click ‘Edit’ button to activate region outline. Both the shape and mask flag are editable for segmented region. After editing, the user must click ‘Update’ to conform modification. The user can press to save current segmentation into a MAT file which contains a mask matrix with the same size of the original matrix and a cell array storing segmentation location information. If only the mask matrix is desired, the user can simply press to save it into workspace. The user can press to load previous segmented regions from saved MAT file. Notice that the user can press to remove segmentation from overlapping with background matrix.

## 7. Surface

This function category generates surface or mesh plot for current image.

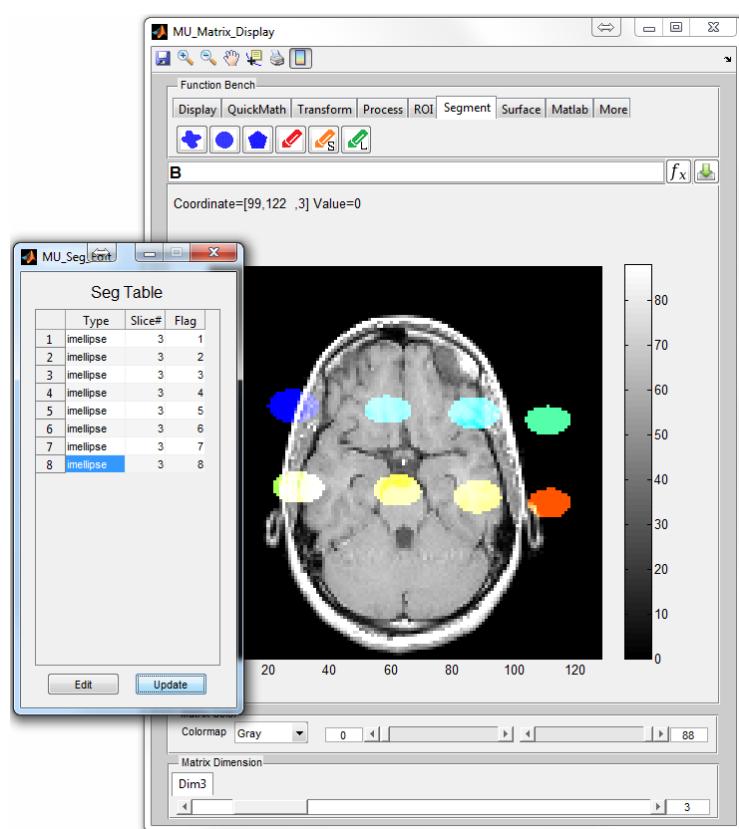


Figure 6.17: Editing Segmentation

-  : Create contour plot of current matrix
-  : Create 3D contour plot of current matrix
-  : Create filled 2D contour plot
-  : Create 3D shaded surface plot
-  : Create surface plot and contour
-  : Create surface plot with colormap based lighting
-  : Create mesh plot
-  : Create mesh plot and contour
-  : Create a curtain around a mesh plot
-  : Create a ribbon plot
-  : Create a waterfall plot
-  : Create pcolor (checkerboard) plot

#### 8. Matlab

Matlab default image tools (Figure 6.18) are tailored for MatrixUser and included in this category.

-  : Perform imtool for current image
-  : Perform immovie for playback current 3D matrix
-  : Perform imcontrast for adjusting image contrast

#### 9. More

Uncategorized functions are categorized under this tab.

-  : Create a 3D graph for rendering current matrix (Figure 6.19)
-  : Perform projection along any given matrix dimension. Support multi-dimensional matrix projection
-  : Perform 3D projection along x, y or z axis with certain angle increment (Figure 6.20)
-  : Reslice 3D matrix at given direction. The user needs to draw a line and double click to confirm for indicating the reslicing direction (Figure 6.21)
-  : Create a movie using current matrix display. Support making movie for overlapped matrices.

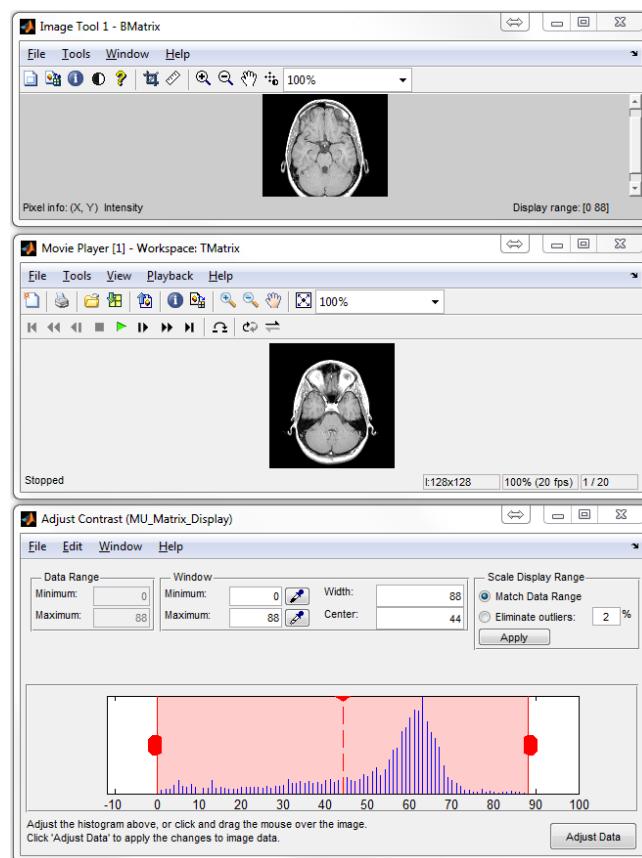


Figure 6.18: Matlab Tools

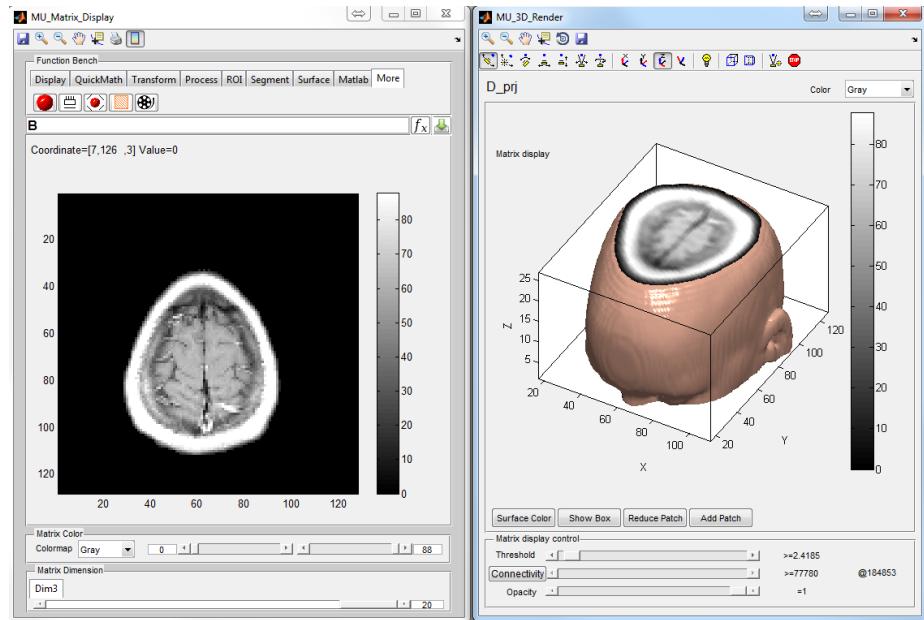


Figure 6.19: An Example of 3D Human Brain Rendering. Control units on the rendering window are provided for fine tuning the renderer. The user can select isosurface threshold, cutoff connectivity threshold (i.e. object with total voxels less than the threshold will be removed from the rendering. ‘@’ is followed by the voxel number of current largest object) and object opacity. A set of pushbuttons are also available for changing surface color, display box and patches. The default Matlab camera toolbar are provided for adjusting the lighting effect.

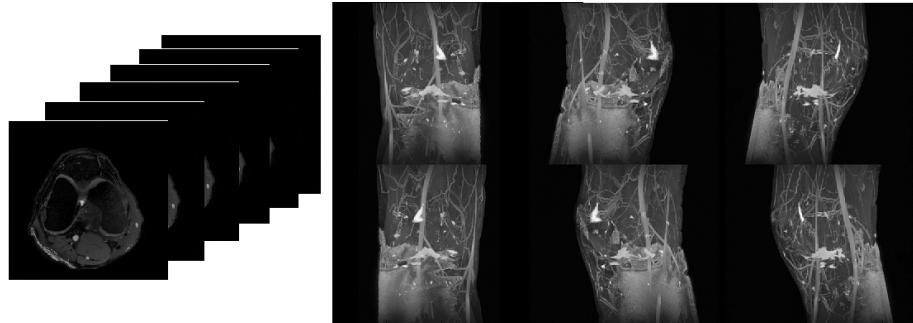


Figure 6.20: 3D Maximum Intensity Projection (MIP). An example of 3D MIP around axial axis of a human knee MRI image stack shows the vascular system of the knee joint.

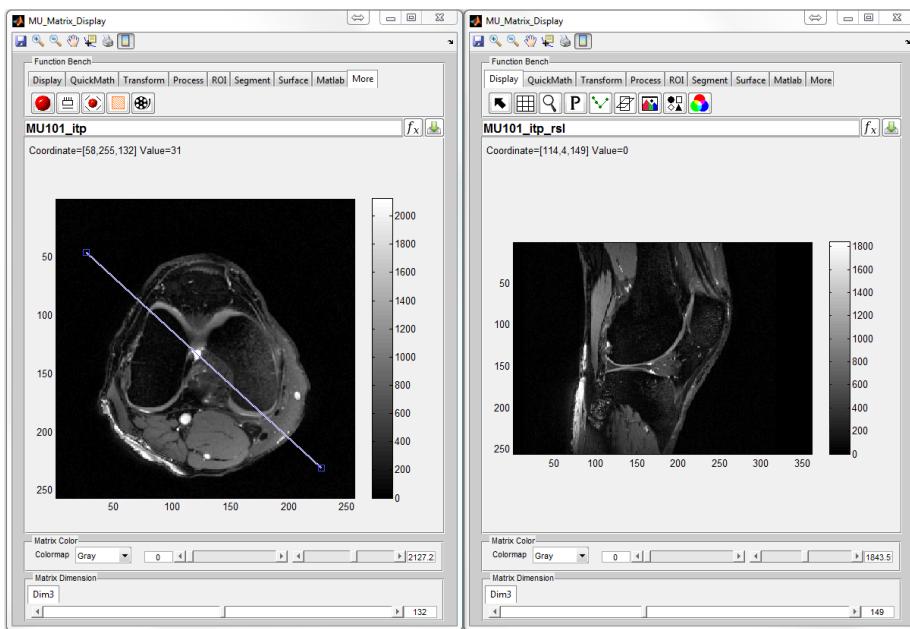


Figure 6.21: Reslice 3D Matrix. An example of 3D reslicing generates a new stack of images in the oblique plane from an axial human knee MRI image stack. Note that the resliced images are extracted from the plane perpendicular to the indicating line on the left window.

## 6.2 arrayShow

The arrayShow tool is a Matlab image viewer which has been designed for the evaluation of multidimensional complex images. It is originally designed by Tilman Johannes Sumpf at Biomedizinische NMR Forschungs GmbH. I have made some modification for making it work with image data directly from MRI-Lab. The user can press ‘arrayShow’ button to activate this viewer. Detailed information about arrayShow can be found at [http://www.biomednmr.mpg.de/index.php?option=com\\_content&task=view&id=137&Itemid=43;](http://www.biomednmr.mpg.de/index.php?option=com_content&task=view&id=137&Itemid=43;)

## 6.3 SpinWatcher

The SpinWatcher is designed for monitoring spin evolution behavior of a single voxel at given MR sequence and field environment. This function can be activated by pressing ‘SpinWatcher’ toolbar icon located at the top of the main simulation console.



Figure 6.22: SpinWatcher Toolbar Icon

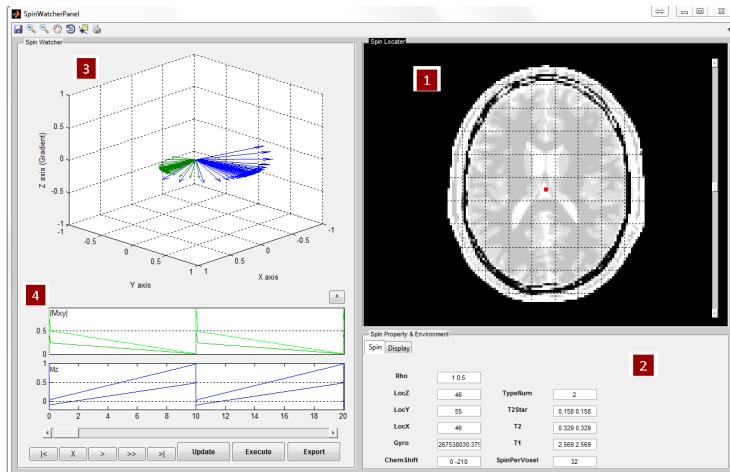


Figure 6.23: SpinWatcher Main Interface

Figure 6.23 demonstrates the overview of the SpinWatcher main interface. This interface consists of

1. Voxel Locator

The default voxel is chosen as the one at the isocenter which is indicated

with a red isocenter marker. However, the user can select any voxel within the virtual object by using  . The slider beside the image can help change image slices.

## 2. Spin Property and Environment

When one voxel is selected, the spin properties will be automatically updated. The user can also modify the spin properties and environment to meet their own needs. The editable properties provided in this interface include:

### (a) Spin

- ChemShift (Hz/T): The chemical shift of the spin
- Gyro (rad/s/T): The gyromagnetic ratio of the spin
- Rho : The density of the spin
- T1 (s): The longitudinal relaxation time
- T2 (s): The transverse relaxation time
- T2Star (s): The T2\* relaxation time
- SpinPerVoxel : The number of spins in each voxel, default one for treating T2\* equal to T2, use a value above one for simulating T2\* effect
- TypeNum : The number of spin species
- LocZ : The Z location of the selected voxel
- LocY : The Y location of the selected voxel
- LocX : The X location of the selected voxel

### (b) Display

- WindowSize : The window width of the spin evolution plot
- ISOHighlight : The flag for turning on and off isocenter mark
- Grid : The flag for turning on and off grid line
- Axes : The flag for turning on and off axes label

The SpinWatcher supports monitoring multiple spin species. The user needs to provide multiple values for T1, T2, T2\*, Rho and ChemShift, and give the correct number of spin species. The values must be separated with space. For example

- ChemShift = 0 -210
- Rho = 1.0 0.5
- T1 = 1.2 1.0
- T2 = 0.02 0.03

- T2Star = 0.002 0.003
- TypeNum = 2

### 3. Spin Watcher Window

The 3D spin evolution animation is displayed in the spin watcher window. The 3D animation can be controlled using a set of control buttons and sidebar

- Scroll Bar : Drag the scroll bar to any intermediate time point between beginning and end
- | < : Move to the beginning
- X : Pause playing, notice that the interface can only be closed when playing is paused
- O : Resume playing
- > : Play at normal speed
- >> : Play at double normal speed
- > | : Move to the end

### 4. Spin Evolution Plot

SpinWatcher provides two plots for capturing the spin evolution (i.e.  $|M_{xy}|$  and  $M_z$ ) with respect to time. The user can press ‘Execute’ button to re-calculate the spin evolution plot after making changes to spin property and environment. The settings can be saved into a file by pressing ‘Update’. To export temporary variables of spin evolution into Matlab base workspace, press ‘Export’, the exported variables include :

- Muts : A array for time points
- MxySum : A array for  $|M_{xy}|$
- MzSum : A array for  $M_z$
- Mx :  $x$  component of magnetization
- My :  $y$  component of magnetization
- Mz :  $z$  component of magnetization

The user can also undock the spin evolution plot by pressing ‘^’.

## 6.4 SARWatcher

(:TODO) SARWatcher is a graphic toolbox for monitoring real time spatial Specific Absorption Rate (SAR) of the virtual object with given experiment design. This toolbox is still under development.

## Chapter 7

# MRiLab Applications

This chapter shows more examples for demonstrating the applications of MRiLab simulation

### 7.1 bSSFP with Non-uniform B0

This example (Figure 7.1) simulates the dark banding artifact in bSSFP images arisen from non-uniform B0 field. To perform this simulation, the following steps are needed:

1. Load Brain (Standard Resolution  $108 \times 90 \times 90$ )
2. Load PSD\_FIESTA3D
3. Load Mag\_GaussianHead

The user can adjust the ‘FlipAng’, ‘TR’ and ‘TE’ to modify the pattern of the banding artifact.

### 7.2 Fat Chemical Shift

This example (Figure 7.2) simulates chemical shift artifact at the interface of water and fat in a GRE sequence. To perform this simulation, the following steps are needed:

1. Load Water Fat Phantom ( $256 \times 256 \times 32 \times 2$ )
2. Load PSD\_GRE3D

The user can adjust the ‘BandWidth’ and ‘FreqDir’ to modify the appearance of the chemical shift.

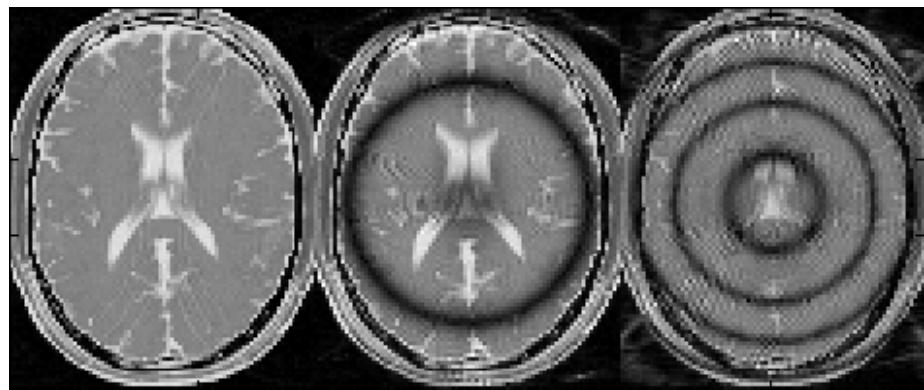


Figure 7.1: Simulated 3D bSSFP brain image of matrix size of  $100 \times 80 \times 20$ . The circular banding artifact is due to added Gaussian field simulating the main magnet field inhomogeneity. Left: TR/TE=16/8ms, FA=40°; Middle: TR/TE=16/8ms, FA=40°, Gaussian B0 applied; Right: TR/TE=32/16ms, FA=40°, Gaussian B0 applied

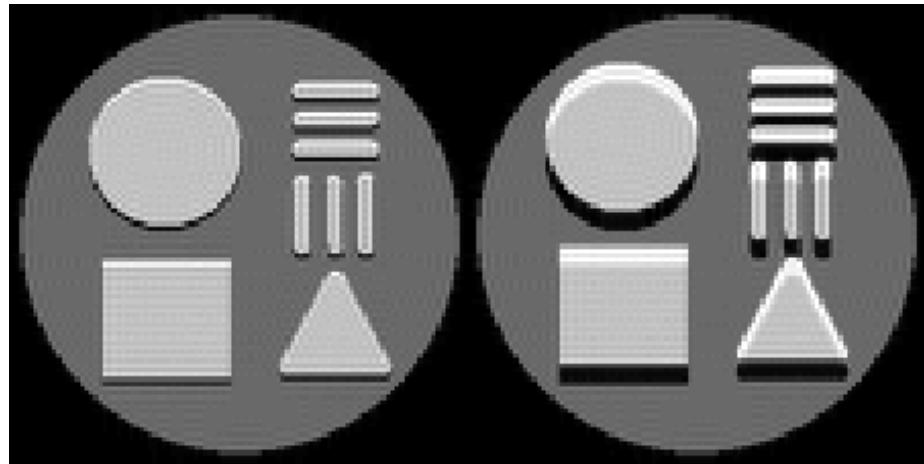


Figure 7.2: Simulated 3D gradient echo image at 3.0T using cartilage-fat phantom showing water fat chemical shift at different readout bandwidth. The simulated in-plane matrix size is  $100 \times 100$ , TR/TE = 10s/50ms, Axial plane, A/P readout. Left: BW=400Hz/pixel; Right: BW=100Hz/pixel.

### 7.3 Multi RF Transmitting

This example (Figure 7.3) simulates multiple RF transmitting using a bSSFP sequence. To perform this simulation, the following steps are needed:

1. Load Brain (Standard Resolution  $108 \times 90 \times 90$ )
2. Load PSD.FIESTA3D
3. Load Coil\_8ChHead to Tx
4. Set ‘MultiTransmit’ to ‘on’

The user can adjust the ‘B1Level’ to modify the actual flip angle, modify the RF pulse using MR sequence Design Toolbox for individual RF source, or modify the coil configuration for generating desired B1+ field.

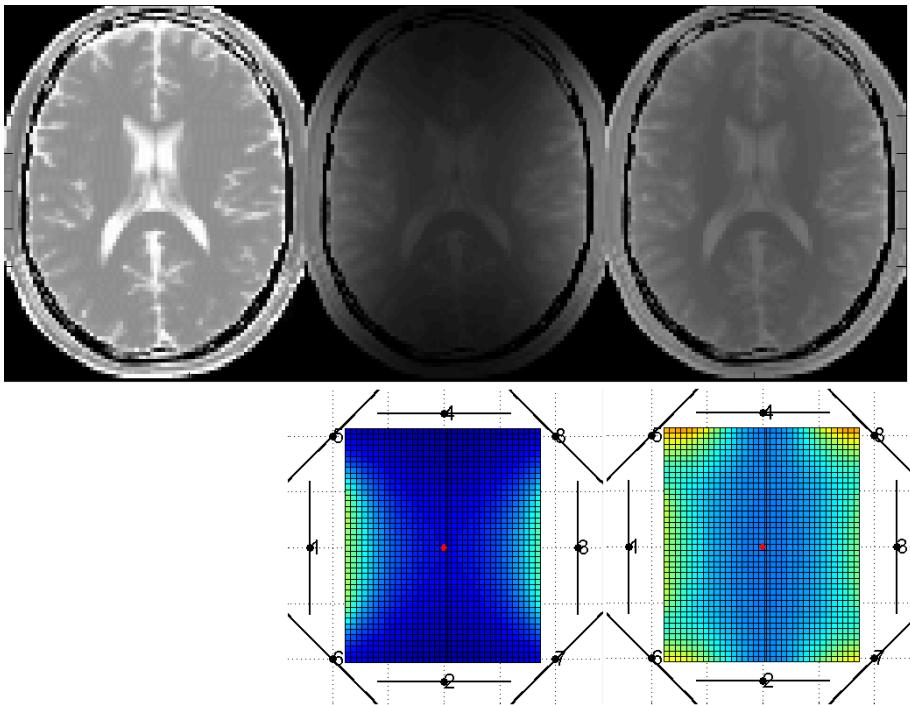


Figure 7.3: Simulated 3D bSSFP brain image of matrix size of  $100 \times 80 \times 4$  with different Tx coil configuration. The simulated image uses  $FA=40^\circ$ ,  $TR/TE = 16/8\text{ms}$ . Left: uniform unit B1+ field, RF transmitting from ‘MasterTxCoil’; Middle: multi RF transmitting from Coil1 and Coil3; Right: multi RF transmitting from all coil channels

## 7.4 Multi Receiving Coil

This example (Figure 7.4) simulates multiple receiving using a SE sequence. To perform this simulation, the following steps are needed:

1. Load Brain (Standard Resolution  $108 \times 90 \times 90$ )
2. Load PSD\_SE3D
3. Load Coil\_8ChHead to Rx

The user can adjust the coil configuration for generating desired B1- field. All eight channels will be receiving MR signal from the virtual object individually.

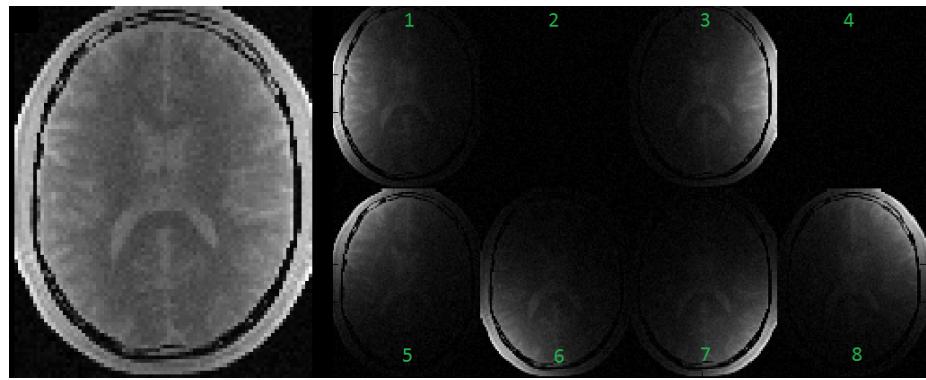


Figure 7.4: Simulated brain spin echo image of matrix size of  $100 \times 80$  with multi receiving Rx coil. The simulated image uses  $\text{FA}=90^\circ$ ,  $\text{TR}/\text{TE} = 10\text{s}/10\text{ms}$ . Left: the combined image using ‘SumofMagn’; Right: individual image for each coil channel, notice that Coil2 and Coil4 receive no signal due to zero B1- field.

## 7.5 Image Gradient

This example (Figure 7.5) simulates applying non-unit gradient with a 3D SPGR sequence. To perform this simulation, the following steps are needed:

1. Load Brain (Standard Resolution  $108 \times 90 \times 90$ )
2. Load PSD\_SPGR3D
3. Load Grad\_LinearHead

The user can adjust the gradient structure for generating desired gradient field. Notice that this applied gradient in GyPE GradLine has a factor of 0.5 in the Y direction. This will cause image contraction in the Y direction.

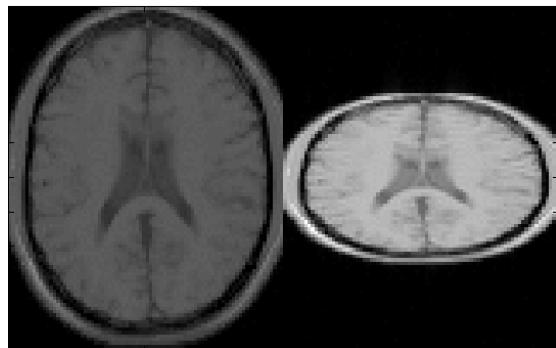


Figure 7.5: Simulated brain 3D SPGR image of matrix size of  $100 \times 80 \times 2$  with non-unit gradient. The simulated image uses FA=20°, TR/TE = 60/8ms. Left: unit gradient applied; Right: non-unit gradient applied in Y direction.

## 7.6 Motion Artifact

This example (Figure 7.6) simulates motion artifact with a 3D GRE sequence. To perform this simulation, the following steps are needed:

1. Load Brain (Standard Resolution  $108 \times 90 \times 90$ )
2. Load PSD\_GRE3D
3. Load Mot\_ShiftHead

The user can adjust the motion structure to generate different motion track patterns, and/or modify motion triggering in the Ext sequence line to sample object movement.

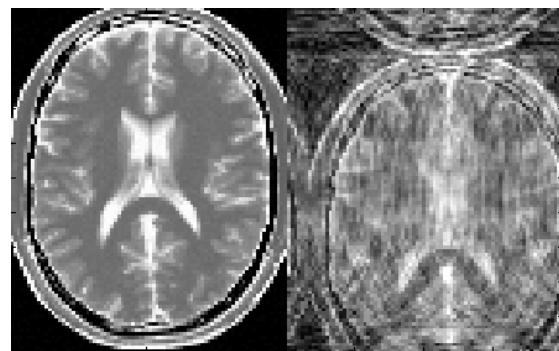


Figure 7.6: Simulated brain GRE image of matrix size of  $100 \times 80$  with translation motion. The simulated image uses  $\text{FA}=90^\circ$ ,  $\text{TR}/\text{TE} = 10\text{s}/50\text{ms}$ , Axial plane, A/P readout. Motion triggering happens one time per TR, and motion lasts for 400s. Left: no motion; Right: translation motion applied.

# **Chapter 8**

## **FFAQs**

Waiting for questions...



# Bibliography

- [1] V.L. YARNYKH: *Actual Flip-Angle Imaging in the Pulsed Steady State: A Method for Rapid Three-Dimensional Mapping of the Transmitted Radiofrequency Field* , Magn. Reson. Med., **57**, (2007), 192-200.
- [2] J.I. JACKSON, C.H. MEYER, D.G. NISHIMURA, AND A. MACOVSKI: *Selection of a Convolution Function for Fourier Inversion Using Gridding* , IEEE Trans. Med. Imag., vol. 10, no. 3, pp 473-478, 1991.
- [3] V. RASCHE, R. PROSKA, R. SINKUS, P. BOERNERT, AND H. EGGLERS: *Resampling of Data Between Arbitrary Grids Using Convolution Interpolation* , IEEE Trans. Med. Imag., vol. 18, no. 5, pp 385-392, 1999.
- [4] P.J. BEATTY, D.G. NISHIMURA, AND J.M. PAULY: *Rapid gridding reconstruction with a minimal oversampling ratio* , IEEE Trans. Med. Imag., vol. 24, no. 6, pp 799-808, 2005.
- [5] G.H. GLOVER: *Simple Analytic Spiral K-Space Algorithm* , Magn. Reson. Med., **42**, (2005), 412-415.
- [6] M.A. BERNSTEIN, K.F. KING, X.J. ZHOU: *Handbook of MRI Pulse Sequences* , Elsevier Academic Press, 2004.
- [7] J. PAULY, P. LE ROUX, D.G. NISHIMURA, AND A. MACOVSKI: *Parameter relations for the Shinnar-Le Roux selective excitation pulse design algorithm* , IEEE Trans. Med. Imag., vol. 10, no. 1, pp 53-65, 1991.
- [8] T. HWANG, P.C.M. VAN ZIJL AND M. GARWOOD: *Fast Broadband Inversion by Adiabatic Pulses* , J. Magn. Reson., 133, pp 200-203, 1998.
- [9] V.L. YARNYKH: *Pulsed Z-Spectroscopic Imaging of Cross-Relaxation Parameters in Tissues for Human MRI: Theory and Clinical Applications* , Magn. Reson. Med., **47**, (2002), 929-939.



# Acknowledgment

I am indebted to all my teachers and mentors. I would like to thank Dr. Neil Gelman for introducing me into the MRI field and inspiring me for developing the MatrixUser project. I would also like to thank Drs. Wally Block and Richard Kijowski who guide me to acquire knowledge of MR sequence design and medical image application. My gratitude is owed to Dr. Alexey Samsonov for inspiring me developing MT relevant functions. Many people also contribute significantly to the MRiLab project through publishing free code online which largely extends MRiLab functionality, in particular, Tilman Johannes Sumpf, Dr. John Pauly and Dr. Brian Hargreaves. Finally, my appreciation goes to my wife Zhaoye Zhou who never gets bored of my talking about MRiLab and supports me for finishing the whole project.



# Afterword

Although MRiLab has been very carefully designed and tuned for performing best versatile MRI simulation, there may also exist bugs and misfunctions that may cause problems. In addition, several important parts are still missing in MRiLab, including multi-pool water exchange simulation, water diffusion and Eddy-Current simulation etc. I am also working to further optimize solving kernel for better performance and to enhance the compatibility with Gadgetron. The MRiLab project is open to the whole MRI community, any user who are interested to improve MRiLab is very welcome to do so. I will be very pleased if you can leave me feedback and ideas for better enhancing MRiLab functionality. I will also be very happy to talk about any kinds of collaboration for future MRiLab development. Please don't hesitate to contact me ([leoliuf@gmail.com](mailto:leoliuf@gmail.com)).