An Extensible, Graphical Environment for Pulse Sequence Design and Simulation

William R. Overall and John M. Pauly
Stanford University Department of Electrical Engineering

Introduction

We have developed a flexible software platform for pulse-sequence design and simulation. It provides:

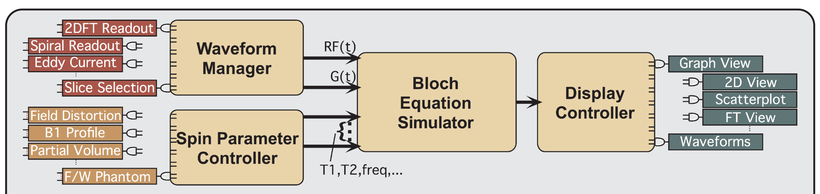
- A robust graphical environment for prototyping MR pulse sequences.
- An integrated, multithreaded Bloch-equation simulator for rapid signal validation.
- Multiple import and export formats for interoperability with existing tools.
- A modular program architecture that allows arbitrary extensibility through plugin APIs.
- Free access to the fully functioning software and online access to example code.

Typical sequence creation requires several iterations of design and simulation. Using this software, many sequences can be completely designed and simulated graphically. For unusual pulse shapes or specialized simulations, short programs can be written to add the desired functionality.

This software, named SpinBench, is available free of charge at http://www.SpinBench.com/

Methods

The program, for Mac OS X only, leverages the free XCode software environment to provide high-level document operations, including Load/Save, Cut/Paste, Undo/Redo, Print, etc.. Saved files are natively XML, which is readable by a number of existing tools.



SpinBench consists of 4 primary code sections:

Waveform Manager- Provides a graphical environment for pulse-sequence design.

Spin Parameter Controller- Specifies simulation values including T1, T2, frequency, velocity, etc.

Bloch Simulator- Computes signal for the given waveforms and parameters.

Display Controller- Organizes data for viewing in any of a number of formats.

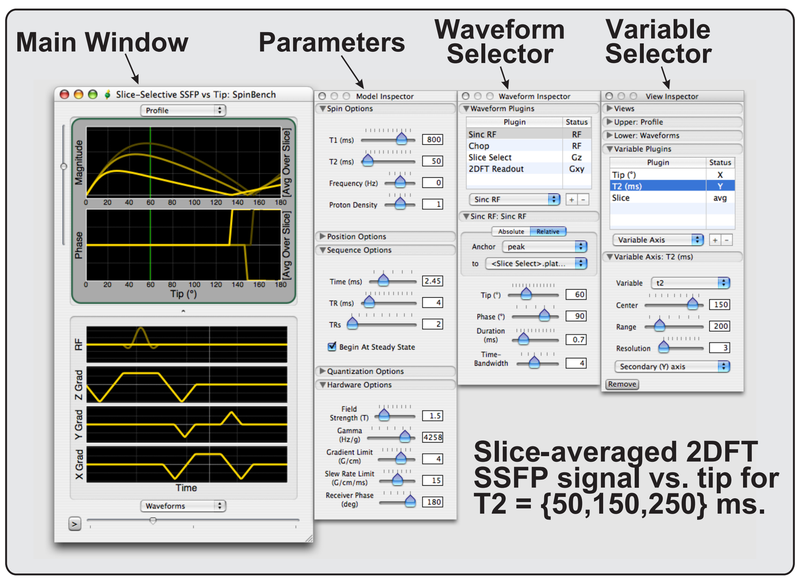
Point-by-point matrix computations provide maximal flexibility and accuracy [1]. Speed is maintained via several optimizations:

Complexity Management - The core Bloch Simulator uses fixed inputs and outputs; flexibility is managed by separate controllers.

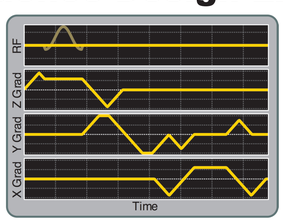
Multithreading - Independent computations are performed in parallel on multicore machines.

Variable Sampling - Waveforms are finely sampled only when necessary.

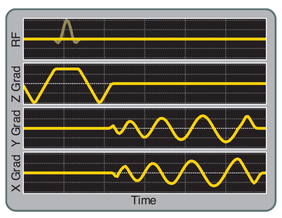
User Interface



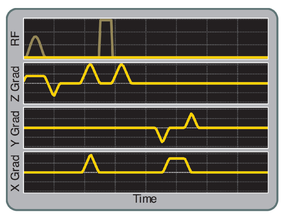
Sequence Design Examples



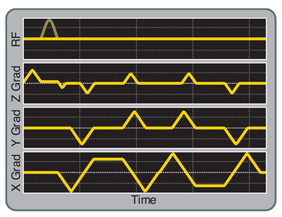
2D GRE Phase Contrast



2D Spiral SSFP

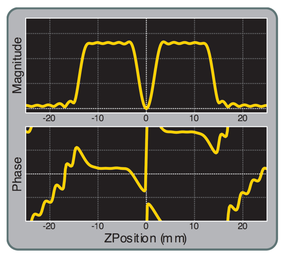


2D Spin Echo

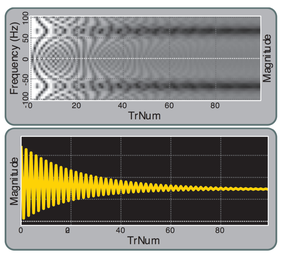


3D Dual-Echo GRE

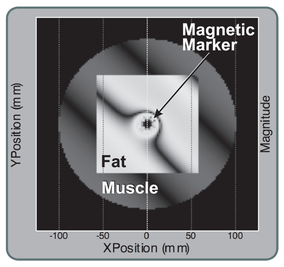
Spin Simulation Examples



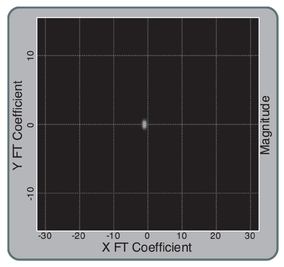
Excitation Profile for Dual-Band RF Pulse



Transition to Steady State vs. Resonant Frequency



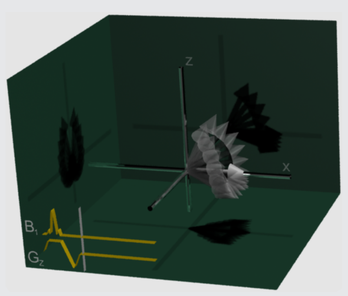
SSFP Phantom Excitation With Linear Field Variation

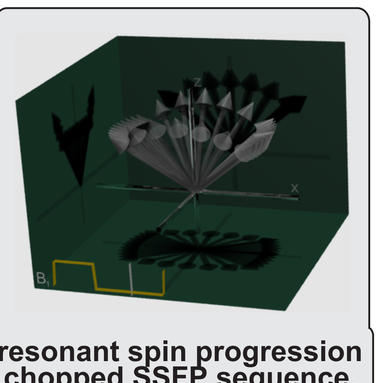


Point Spread Function for Off-Resonant Spins

Rendering Examples

Tip & phase errors along *z* for a 90° selective sinc excitation.





Off-resonant spin progression in a chopped SSFP sequence.

To implement unique waveforms, spin conditions, output formats, or alternative computational methods (including [2,3]), a *plugin interface* is provided.

- Plugins are short Objective-C programs.
- Provided APIs allow interfacing with SpinBench.
- Plugin binaries and/or code can be shared online.

Results

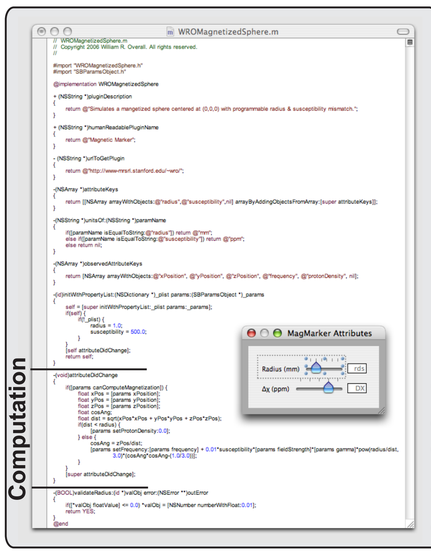
All simulations shown here required less than 6 sec. on a 2.0-GHz MacBook Pro. For comparison, a MATLAB simulation of the iron-marker result below required over 2 minutes on the same machine.

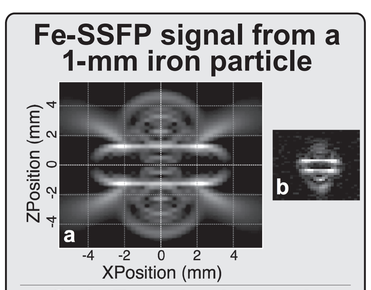
Plugins exist to simulate a number of standard waveform types including slice selection and 2DFT and spiral readouts. SpinBench was also used to support research presented here at ISMRM 2007:

#1117- Phase refocusing for improved visualization of interventional guidewires

#579- Field-encoded SSFP: A new method for positive-contrast visualization of paramagnetic agents

For the second of these, a plugin was written to simulate the dipole fields associated with an iron marker. The plugin code, its graphical interface, and a sample simulation are shown below.





The iron-marker plugin consists of < 20 lines for computation, plus some housekeeping and setup. The plugin's interface is designed graphically.

Discussion

SpinBench is a flexible, fast, vendor-independent tool for concurrent pulse-sequence design and simulation. Designs can be easily saved and shared in the program's built-in format. Many common sequences and simulations can be defined using the graphical interface, while arbitrary functionality can be achieved through plugin APIs.

References

1. Summers RM, *et al.* MRM 3(3): 363-376, 1986.
2. Magland J, *et al.* Proc 14th ISMRM: 2365, 2006.
3. Benoit-Cattin H, *et al.* JMR 173(1): 97-115, 2005.

Download a free copy at:

http://www.SpinBench.com/