Pulse Sequence Programming using XML and JavaScript

W. Overall[1], and J. Pauly[1]
[1]Electrical Engineering, Stanford University, Stanford, CA, United States

Introduction: Pulse-sequence programming is typically performed in low-level compiled languages, principally C and C++[1,2]. These languages are extremely flexible, but the complexity of the resulting code makes it hard to maintain. Creating new sequences is time-consuming, and the resulting compiled code is often not portable across platforms. We propose the use of higher-level XML and interpreted scripting to simplify the sequence programming process, resulting in human-readable code that is portable across scanners and more easily maintained. This approach has analogy in web programming, where HTML files provide a lightweight format for describing web pages, and interpreted scripting languages (JavaScript, perl, etc.) are made available on top of that to enable more complex and dynamic content.

Methods: Our approach uses a standard XML template at its core for maximum portability across platforms. Pulse sequences are specified through three classes of tags: pulse tags, which specify individual pulses; layout tags, which specify global constraints; and sequencing tags, which specify sequence changes that occur across TR boundaries. As an example, a simple RF-spoiled gradient-echo pulse sequence might be specified as shown in Figure 1. For this sequence, four pulse tags are required to specify the readout, slice selection, RF, and TE interval, and three sequencing tags are used to set up the number of repetitions, RF spoiling, and imaging plane. Pulse positions can be specified relative to the other pulses in the sequence or as an absolute time, so that the pulses move as expected in response to sequence changes. Pulse tags exist for many common purposes, and primitive binary pulse tags exist if unsupported or arbitrary waveforms are desired. Similarly, sequencing tags exist for common operations such as magnetization preparation, interleaving of sequences, etc.

While this framework is sufficient for fully specifying the majority of common pulse sequences, some constraints do not easily fit within the XML paradigm. As an example, if one wanted to set TE as an arbitrary algebraic combination of other sequence parameters, the XML file format alone would not be adequate. For these situations, we also provide the ability to include inline JavaScript code. Through the JavaScript, arbitrary sequence parameters can be accessed and set, and flexible sequence control is available with a minimum of code.

We have implemented this XML+JavaScript sequence interpreter within the SpinBench[3] sequence-programming and spin simulation tool, which is available as a free download online. In addition to the ability to read and write these lightweight sequence files, the program is also able to generate and modify sequences through a graphical interface. The sequence-rendering engine (Fig. 2) interprets each XML tag as a separate object, each with its own unique set of properties and rendering rules. A sequencing pipeline creates and renders each TR as requested by the output, caching intermediate results at each stage of the pipeline. This approach permits fast rendering without the large memory requirements associated with keeping an entire sequence rendered in memory at once. JavaScript code is interpreted as it is encountered, and results are passed along to the appropriate object.

Results: We have applied this approach for programming a wide variety of common pulse sequences, including those shown in Figure 3. New sequences are straightforward to implement, and many can be completed using the graphical interface in less than an hour. The JavaScript interpreter is useful in specifying dynamic logical constraints, but most sequences require less than 20 lines of JavaScript in total.

Discussion: We have developed and demonstrated a flexible, platform-independent MR pulse-programming language. Our graphical environment for creating pulse sequences using this language, along with a library of sample sequences, is available for download over the web[3].

References:
1. Magland, J, *et al*. Proc. 14th ISMRM: 2365, 2006.
2. Benoit-Cattin, H, *et al*. JMR 173(1): 97-115, 2005.
3. http://www.SpinBench.com/

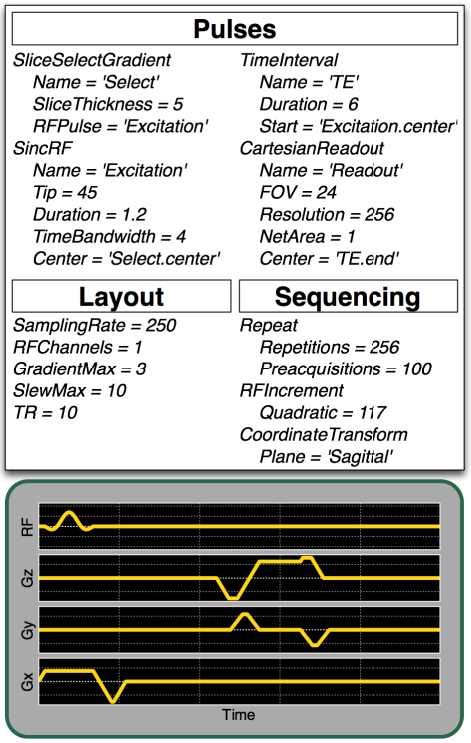

Figure 1: *Pseudo-XML (top) shows the settings required for a simple SPGR pulse sequence. A specialized XML interpreter can then parse and render the sequence as shown (bottom).*

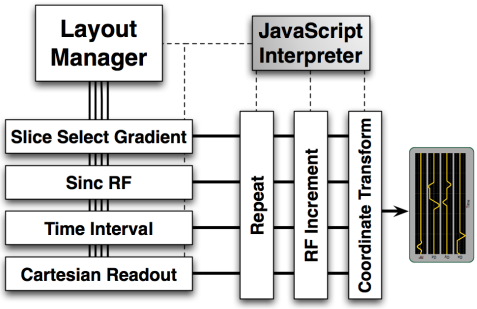

Figure 2: *Block diagram of the rendering process. Pulses are rendered according to layout constraints, then TRs are sequenced as specified.*

Figure 3: *Pulse sequences described in XML and rendered via SpinBench. (a): Fast Spin Echo (b): Refocused SSFP (c): Spiral GRE.*