SWEETENING SIGNALS AND SYSTEMS WITH SIMULINK

G.D. Cain, M.A. Mughal, A. Yardim and D. Barjamovic

University of Westminster, Department of Electronic Systems, London W1M 8JS, UK

ABSTRACT

Block diagram assembly of simulation experiments is particularly attractive to students of signals and systems. SIMULINK 3, when augmented by specialist DSP/comms blocks that enhance operator interaction at run time, offers a uniquely valuable setting for flexible and powerful "instrument-based" experimentation. We describe a range of student laboratory exercises which we have rapidly commissioned and point out the hurdles developers face when expanding blockset holdings. New real-time audio capabilities promise experiments with special motivating features.

SIMULINK is the block diagram tool in the MATLAB family of software products. It permits nearly effortless visual assembly of complicated signal generators, processing system elements and measuring "instruments" into runnable behavioural system simulation configurations while still retaining access to the MATLAB Workspace, renowned for its ease of ready calculations, friendly graph plotting and rapid development capability for high-level algorithms. Thus SIMULINK's potential for accelerating learning of topics in signals and systems areas is unparalleled.

Once suitable subsystem blocks are created (often from "component" functional blocks already supplied in the SIMULINK collection of "blocksets") this great promise can start to be realized. Our experience - once this watershed is reached - is that SIMULINK does indeed provide a superb learning vehicle: every bit as good as the expectations this tool raises. Students can quickly master the physical manipulation of blocks and carry through effective learning programmes in a surprisingly short time.

The problem comes behind the scenes, in placing the ready-made infrastructure at the disposal of the students to begin with. Arriving "at critical mass" in operation away from SIMULINK's traditional centre of strength in control systems did not prove easy for us. Productive usage in areas such as communications and DSP requires very much more (in our view) than the elements currently furnished in MATHWORKS' Communication Toolbox or its DSP Blockset. Fortunately, SIMULINK 3 (which became available to us at the midpoint of the timeframe covered in this paper) is a massive improvement on SIMULINK 2.2, and goes a long way toward eradicating the difficulties (such as awkward complex signal handling) that we experienced when we first started up the SIMULINK learning curve.

Nevertheless, the investment of effort needed for effective development capability is non-trivial and there is much learning required on the part of the development team to achieve what is needed. Regrettably, the spectacular ease of development that mainline MATLAB enjoys is currently not immediately transferable into the SIMULINK domain. We could have benefited greatly from knowledge of vital "tricks" and interaction with expert courseware developers in this arena, but did not have such luxuries in the time available for our project. This paper relates some of our tribulations and successes while launching a number of laboratory exercises over a 7-month segment of an academic year.

Our project initially focussed on providing a suite of easy-to-grasp experiments to support the study of basic signal representation, filtering, modulation and sampling/reconstruction of signals. Although our initial deadline (about 6 weeks' preparation window) centred on a one-week concentrated offering of this material as part of an introductory MSc module, the same material was, from the outset, intended to be utilized (at a more leisurely pace) at undergraduate level as well. As a first development plateau, we settled on creation of these seven (two-hour) experiments to be performed via SIMULINK:

(1) **Signal-Space Representation of Signals** (highlighting orthogonal basis functions and vector components in signal space)

(2) **Adding Two Signals** (showing various superposition and disjointness cases in both time and frequency domains)

(3) **Experimenting with Periodic Signals and Bandwidth** (moving from Fourier transform to the special case of Fourier Series; also instrumenting for several alternative bandwidth declaration strategies)

(4) **Filtering Fun** (separating nearby sinusoids and suppressing bandpass noise)

(5) **Modulation Mania** (simple AM and envelope demodulation; phase modulation; OOK, PSK and FSK)

(6) **A Sampler of Sampling** (essence of spectral replicating and filtering to reconstruct; audible aliasing; offset sampling)

(7) **A Hilbert Transform Tour** (what these odd transforms look like; use in SSB-AM and lowpass filter structures)

Experiments (2)-(5) were the most successful, resulting in most tasks being achieved by most students in the session time available. Not surprisingly, the first and last experiments proved least satisfactory due to the inherent difficulty of their subject matter, with Experiment (6) falling somewhere in between. In each experiment there were either 3 or 4 tasks requiring construction or modification of a SIMULINK model. Each task took form from a colour-coded library (where the given component blocks resided), so it was easy to see at a glance how far each group of students had progressed.

Figure 1 shows a specimen solution to the second task in Experiment (4). Notice that, apart from the summer, every component is a non-standard block. The burden of massive augmentation of the standard library offering was undertaken because we felt each signal and noise generator needed to have interactive parameter control at runtime if we were to obtain the "feel" of true equipment-based laboratory experiments. (Sadly, SIMULINK 2.2.1 - and later SIMULINK 3 -provided us with only a single block - "slider gain" - with non-static interaction capability.)

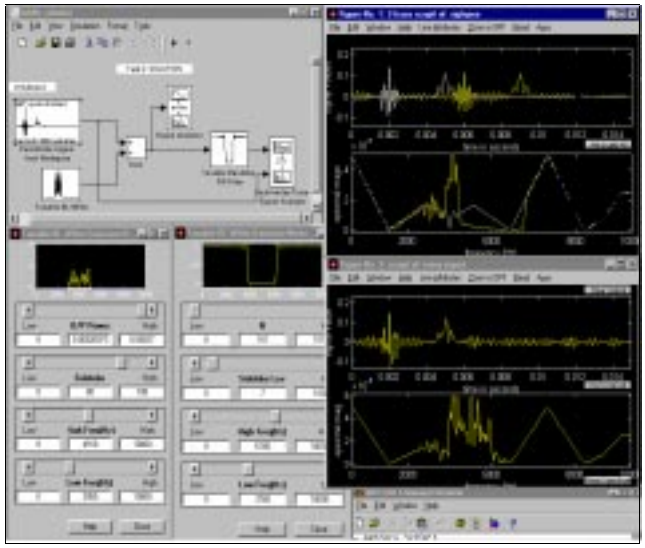

Figure 1. Filtering a Signal Embedded in Bandpass Gaussian Noise

Figure 2 displays a solution encountered in Experiment (6), where replicated spectral shapes can be observed on the second signal analyzer trace. Again, almost all blocks were newly

constructed so as to obtain the experimental control we wanted. Our main work, then, was fabrication of specialist blocks. Whenever possible, these featured sliders and edit boxes for parameter control. For instance, our rectangular pulse generator has 4 sliders (amplitude, pulse width, pulse repetition period and pulse offset). We produced a library of some 52 blocks (over a very hectic 6-week period) that could deliver the sort of interactivity we felt was necessary. We found that icon plots of gains and signal shapes were highly useful since these provided students with valuable visual cues as to which block was undergoing parameter adjustment. To our amazement, all this stood up to some 420 student-hours of usage without faltering! This just goes to show the inherent solidity of the SIMULINK environment. If you provide a palette of system components that meet your desires, the SIMULINK framework will deliver the performance sought, most admirably!

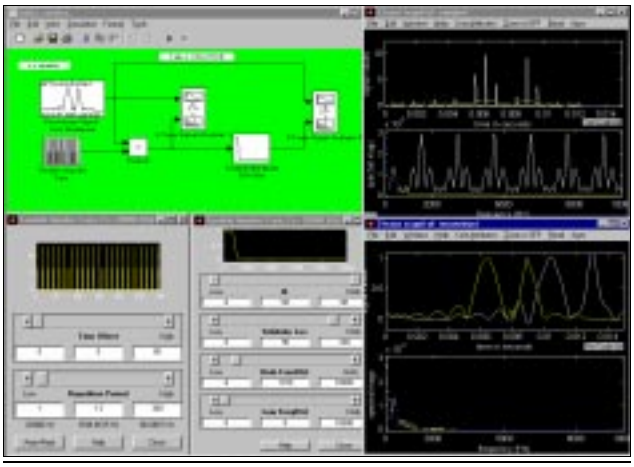

Figure 2. Specimen Experimental Task of Performing Sampling and Reconstruction

The challenge in working with SIMULINK is to move from the block-oriented DSP approach used in MATLAB working to a stream processing approach. The tack we adopted was to employ buffering so that all our signals were (apart from parameter-tuning transients and purposely injected noise) periodic. This gave the feel of stably-triggered oscilloscope operation and a general re-enforcement of standard equipment-realized experimentation that students had experienced elsewhere. All this necessitated two categories of block development: buffered (like in signal generators and analyzers) and instantaneous (such as rectifiers and filters). This distinction is a bit subtle and not too natural, yet strangely no student appeared to have met difficulty in smoothly invoking the two classes of blocks whenever needed.

Following our first frantically-prepared (but gratifying) SIMULINK laboratories, we settled down to weaving this powerful capability into all our DSP offerings. From October 1998 to mid-March 1999 we carried through a fairly ambitious programme:

- Baseband Data Communications - 2 labs at PG/ Industry level: Pulse shaping, Nyquist filters & noisefree data patterns; Nyquist, Root-Nyquist and optimal filters fighting noisy data patterns (40 student-hours)
- Baseband Signal Shaping - 1 lab at UG level: Eye diagrams on holding scopes (87 student-hours)
- TOUCHTONE (DTMF) Detection - 1 lab at PG/ Industry level (30 student-hours)
- TOUCHTONE (DTMF) Detection - 1 lab at UG level (84 student-hours)

Thus, a total of twelve laboratory sessions (many of them run multiply to allow small-group work) were commissioned, clocking up over 650 student-hours of operation. All laboratory assignments were paperless, with negligible prior preparation on the part of the students. In addition, we gained and utilized a repertoire of SIMULINK vignettes for spicing up in-class demonstrations.

In building all this experimental framework our team has expanded the early specialist block library to well over 160 blocks. It is felt that now a comfortably wide range of tasks can be supported and that many useful new experiments could be devised within the present resources.

Figure 3 provides a preview of our newest sorts of experiments. Here we take an audio signal from a CD soundtrack, process it and play it out in real time using the on-board PC sound card. Although the processing possible at this early juncture is severely constrained (here we shape our noise contribution with a two-pole resonator and later combat it with a 31-coefficient FIR bandstop filter), the new audio handling blocks and the power of frame-based filtering permit (barely) real-time deadlines to be met. We plan to launch a number of short, sharp experimental tasks centred around such appealing sound effects.

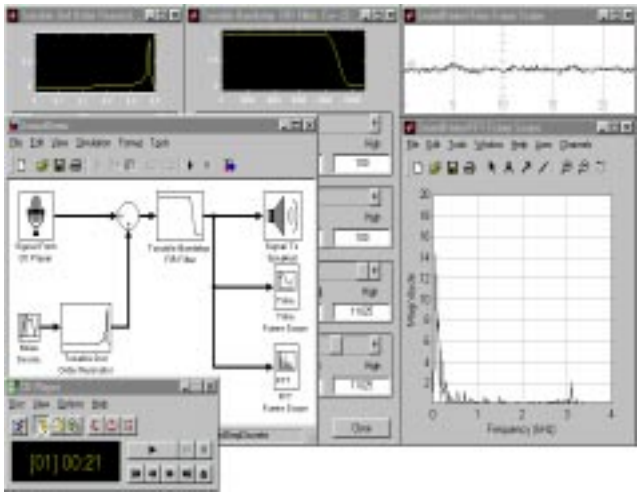

Figure 3. Real-Time Audio Enhancement

Our experience in developing blocks has highlighted these issues:

- What should the "granularity" of a good block be? (Too all-embracing functionality erodes user confidence, while too low-level constructs make for models that are unwieldy and burdensome to wire and maintain)
- How much "detail hiding" in masked subsystems is healthy?
- How deep should a user need to push in setting parameters of a block (shielding users from excessive control obligations)
- How to enhance model clarity/self-documentability?
- How can powerful GUIs be smoothly linked in to masked block control?
- What guides buffering tradeoffs when audio signals are handled in experiments?

Library navigation emerged as a common annoyance. It is all too easy (even with the new block browser) to get diverted to a lengthy search for blocks in various remote blocksets locations. Our approach to easing this problem for our students was to provide a palette of block types exactly adequate to permit task solution, distancing our students from the wider body of blocks available as standards or indeed from our full specialist library. This greatly promoted focus and doubtless helped maintain a brisk pace of experiment performance.

Many are the difficulties that face developers of specialist blocks. But, based on the work we have carried out, our conclusions are clear: when development hurdles are overcome and powerful, proven blockset elements are available for use, a diverse portfolio of laboratories can rapidly be built up. In the space of a couple of hours students can move from basic jobs to very challenging undertakings. When all the supporting infrastructure has all been done right, SIMULINK offers an excellent and exciting experimentation environment that students enjoy using.