Sound Lab: A real-time, software-based system for the study of spatial hearing.

Elizabeth M. Wenzel / NASA-Ames Research Center [1]
Joel D. Miller / Raytheon STX Corporation
Jonathan S. Abel / San Jose State University Foundation

ABSTRACT

This paper describes Sound Lab (SLAB): a software-based, real-time virtual audio rendering system designed to work in the personal computer environment using a standard signal-processing library. SLAB is being developed as a tool for the study of spatial hearing. The system takes advantage of the low-cost PC platform while providing a flexible, maintainable, and extensible architecture to enable the quick development of experiments. The current capabilities and dynamic behavior of the SLAB system are described from the point of view of the psychoacoustician user.

1. INTRODUCTION

Interest in the simulation of acoustic environments has prompted a number of technology development efforts over the years for applications such as auralization of concert halls and listening rooms, virtual reality, spatial information displays in aviation, and better sound effects for video games. Each of these applications imply different task requirements or emphasize different aspects of the listening experience that, in turn, require different approaches in the development of rendering software and hardware. For example, the auralization of a concert hall or listening room requires accurate synthesis of the room response in order to create what may be perceived as an authentic experience. Information displays that rely on spatial hearing, on the other hand, are more often concerned with localization accuracy than the subjective authenticity of the experience. The former requires computationally intensive synthesis of the entire binaural room response that typically must be done off-line and/or with specialized hardware. A simpler simulation that emphasizes accurate control of the direct path, and perhaps a limited number of early reflections, may better achieve the latter goal. The fact that such a simulation does not sound "real" may have little to do with the quality of directional information provided.

Virtual reality applications such as astronaut training environments, where both good directional information and a sense of presence in the environment are

[1] Corresponding Author: bwenzel@mail.arc.nasa.gov

desired, may have requirements for both accuracy and some degree of authenticity or realism. Achieving these two aspects of the simulation requires that head tracking be enabled with special attention devoted to the dynamic response of the system. For example, a relatively high update rate (about 60 Hz) and low latency (less than about 100 ms) may be required in order to optimize localization cues from head motion as well as provide a smooth and responsive simulation of a moving listener and/or sound source [1-6]. Implementing a perceptually adequate dynamic response is computationally intensive and typically requires an array of dedicated digital signal processors and one or more host computers.

One solution for synthesizing interactive virtual audio has been the development of hybrid systems [e.g., 5-8]. These systems attempt to reconcile these goals by implementing real-time processing of the direct path and early reflections using a model (e.g., the image model) combined with measured or modeled representations of late reflections and reverberation that have been pre-computed for a limited set of listener-source positions. During dynamic, real-time synthesis, only the direct path and early reflections can be readily updated in response to changes in listener or source position. A densely measured or interpolated HRTF database is needed to avoid artifacts during updates. Late portions of the room response typically remain static in response to head or source motion, or given enough computational power, could be updated in response to head tracking using a database of pre-computed impulse responses. Model-based synthesis is computationally expensive but has minimal memory requirements, while data-based rendering is much less computationally expensive and has large memory requirements [6].

Both approaches could benefit from further research that specifies the perceptual fidelity required for adequate synthesis [e.g., 9, 10]. For example, it is commonly assumed that only the direct-path head-related transfer functions (HRTFs) need to be rendered at the highest possible fidelity while early reflections may be rendered with less fidelity, i.e., fewer filter coefficients [8]. However, the number of coefficients actually used is often based on a designer's best guess and the limitations of a particular system, rather than on the outcome of perceptual studies. Such studies could give the designers of such systems better guidance regarding where to devote computational resources with better assurance that perceptual validity is not being sacrificed.

The goal of the system described here, Sound Lab (SLAB), is to provide an experimental platform with low-level control of a variety of signal-processing parameters for conducting such studies. For example, some of the parameters that can be examined in future psychoacoustic studies include the number, fidelity (number of filter taps) and positioning (correct vs. incorrect) of reflections. System latency and update rate can also be manipulated. The project is also an

attempt to provide the basis of a low-cost, software-based system for dynamic synthesis of virtual audio over headphones that does not require an array of special purpose signal processing hardware. Because it has been designed for the Windows platform and relies on a standard signal-processing library, it can more readily take advantage of improvements in processing power without extensive software revisions.

The development of SLAB is currently a work-in-progress. This paper outlines the overall goals and architecture of the system as well as describes progress to date in implementation. It should be emphasized that not all aspects of the system described in Section 2 have been fully implemented.

2. DESIGN CHARACTERISTICS OF SOUND LAB

The SLAB system was designed to allow control over the kinds of spatialization parameters studied in psychoacoustic studies while providing a quick experiment development cycle. The system features a modular, object-oriented design that provides the flexibility and extensibility needed to accommodate a wide range of experiments. This design approach has the additional benefit of easing maintenance requirements.

Because of its modularity, SLAB can support a wide variety of signal flow structures without extensive software revisions. Currently, the "auralization unit" (see below) has a fixed architecture, consisting of a set of parallel signal paths from each source to the listener. Parameters determining the processing (delay line indices, filter coefficients, and the like) are computed based on the experiment state (defined by quantities such as source and listener position), and applied to the signal flow. In this way, the signal flow may be optimized for processing efficiency and latency, independent of the experiment. Since only certain aspects of the translation of experimental state to signal processing parameters change from study to study, development time is minimized. In contrast to an extremely flexible but computationally intensive system like the Spatialisateur [12] that uses a graphical signal processing software environment, SLAB's fixed signal flow architecture can be thought of as a compromise solution that optimizes efficiency at the expense of complete flexibility. Such a compromise is necessary given the design goal of providing a software solution for a low-cost host platform (e.g., Intel Pentium).

2.1 SLAB System Architecture

As shown in Fig. 1, the SLAB system is composed of five conceptual software layers: Application Programming Interface (API), 3D Projection, Signal Flow Translation, Signal Routing and Processing, and Digital Signal Processing Library. The acoustic scene, including listener, source, and environment

characteristics, is specified in the API layer and translated in the 3D Projection layer to geometric quantities, such as range and arrival angle for each path rendered between the source and listener. Physical effects, such as the head-related transfer function and air absorption, are rendered using a set of parallel signal paths built on Intel's Signal Processing Library.

2.2 SLAB Signal Processing

Physical scenario. The target scenario to be rendered is shown in Fig. 2. There are three domains in the physical scenario: the source, the environment, and the listener. A source, characterized by its waveform, level, radiation pattern, size, and dynamic quantities including position and orientation, radiates into an environment. Propagation of acoustic energy in the environment is specified by the speed of sound, spherical spreading loss, and air absorption; the environment is further specified by the location and characteristics of reflecting and transmitting objects. The source signal propagates through the environment, arriving at a listener characterized by a head-related impulse response (HRIR) and interaural time delay (ITD), as well as a dynamically changing position and orientation.

Physical signal flow. A signal path may be modeled according to the scenario of Fig. 2 using the signal flow architecture shown in Fig. 3. A set of P paths from the source to the listener (including the direct path) is separately rendered. The filter r(z) imposes the source radiation pattern on the source signal to take the signal from the source to a point in the vicinity of the source along a particular radiation direction. The filter $z^{-\tau_\alpha}$ a(z) applies the propagation delay, spherical spreading loss, and air absorption experienced as the source signal propagates from near the source to near the listener; the filter m(z) imposes transmission or reflection characteristics of any objects encountered. The filter $z^{-\tau_h}$ h(z) represents the head-related impulse response and interaural time delay, and takes any arriving signal from the vicinity of the listener along a particular direction to the listener's ear canals.

SLAB signal flow. The SLAB signal flow shown in Fig. 4 was designed to implement the physical effects discussed above in an easily maintained, efficient architecture. It consists of a set of parallel signal paths, one for each rendered path from the source to a listener's ears. The propagation delay and interaural time delay for each source-to-ear path are combined, and implemented via an interpolated delay line. Static effects along each path, such as materials reflection filtering are combined and implemented as an infinite impulse response (IIR) filter. A finite impulse response (FIR) filter is used to implement dynamic effects such as the head-related transfer function and the source radiation pattern.

Interpolated delay line. The interpolated delay line provides a fractionally indexed delay representing propagation delay and interaural time delay. The fractional delay line is implemented by linearly interpolating between adjacent samples of a two-times, up-sampled source signal.

Linear interpolation is used for its computational simplicity; each fractional sample evaluated requires only two multiply-accumulates. However, linear interpolation of the input source signal samples as described in [8] results in a low-pass filtering for delays near odd half integers. This effect is illustrated in Fig. 5, which plots the magnitude transfer function and phase delay for a linearly interpolated delay. By contrast, as illustrated in Fig. 6, the linearly interpolated, up-sampled delay exhibits noticeably less high-frequency attenuation. Since the input signal needs to be up-sampled only once, the additional computational cost is fixed and small compared with the cost of linearly interpolating indices for even a small number of paths. The factor of two up-sampling was used because it provided a sufficiently flat magnitude spectrum, was inexpensive to compute, and fit within the memory budget. A larger up-sampling filter would provide a flatter magnitude spectrum, but would require additional memory and multiply-accumulates. Finally, it should be pointed out that while other fractional delay methods are available, including all-pass interpolators, sinc interpolators, and the like [11], they have a much higher per-tap computational cost.

IIR filter (static scenario properties). The SLAB signal flow uses an IIR filter in each path to implement static filtering, such as that imposed by a materials reflection. The idea is that while IIR filters are often difficult to change over time without audible artifacts [12], many effects are more efficiently implemented via an IIR filter. For instance, many materials are characterized by reflections that are attenuated above a transition frequency, often below 1000 Hz. A low-order IIR filter is sufficient to model such materials, but because the transition frequency is such a small fraction of the sampling rate, the corresponding FIR implementation would require a large number of taps to adequately represent the required low-frequency detail.

Note that the SLAB signal flow architecture with its fixed IIR materials filter is limited to specular reflections (angle of incidence equals angle of reflection) in which the filtering is independent of the direction of arrival. In this way, separate image model paths are rendered, and the filtering resulting from interactions with reflecting and transmitting features in the environment is fixed (as the cascade of reflection and transmission filters) and independent of the path geometry. Diffuse reflections are not modeled in the SLAB auralization unit.

FIR filter (dynamic scenario properties). The SLAB FIR filter implements dynamically changing effects, including the source radiation pattern, air absorption, spherical spreading loss, and head-related transfer function. The FIR

filter coefficients for a given path are formed by convolving the impulse responses of the effects rendered, and windowing the resulting impulse response to the FIR filter length for that path.

Having a single FIR filter allows any number of effects to be rendered using the same optimized signal processing structure. However, care should be taken that the cascade of effects does not result in an impulse response that is much longer than the computational constraints allow. For instance, separate head-related transfer function tables could be prepared for the direct path and the reflected paths if the reflected path FIR filters were shorter than those of the direct path. Alternatively as discussed in [8], raw HRTFs may be appropriately modeled or smoothed, resulting in lower-order FIR filters that consume less computational resources.

To compute the source directivity filter for a given path from the source to listener ear, a table of impulse responses for the source radiation pattern is indexed according to the radiation direction. Source radiation-pattern impulse responses are tabulated on a grid in azimuth and elevation relative to the source, having the same number of azimuths for each elevation tabulated. A four-way linear interpolation is used to form the radiation pattern impulse response, $h(t; \theta, \varphi)$, associated with a given radiation azimuth θ and elevation φ.

$$h(t; \theta, \varphi) = \alpha_\varphi \alpha_\theta \, g(t; \theta_2, \varphi_2) + \alpha_\varphi (1-\alpha_\theta) \, g(t; \theta_1, \varphi_2) + (1-\alpha_\varphi) \alpha_\theta \, g(t; \theta_2, \varphi_1) + (1-\alpha_\varphi)(1-\alpha_\theta) \, g(t; \theta_1, \varphi_1). \quad (1)$$

The $g(t; \theta_k, \varphi_k)$ represent tabulated impulse responses; θ_1 and θ_2 are the tabulated azimuths immediately smaller and immediately larger than the input azimuth θ; and φ_1 and φ_2 are the tabulated elevations immediately smaller and immediately larger than the input elevation φ. The factors α are fractions representing the distance between the indexed azimuth and elevation and the interpolated azimuth and elevation:

$$\alpha_\varphi = (\varphi - \varphi_1) / (\varphi_2 - \varphi_1).$$
$$\alpha_\theta = (\theta - \theta_1) / (\theta_2 - \theta_1). \quad (2)$$

Spherical spreading loss is computed as $(1 + \rho^2 / \sigma^2)^{-1/2}$, where ρ is the source distance from the listener, and σ is the source size. This characteristic closely approximates that of a planar baffled cylindrical piston of radius σ [13].

The head-related transfer function for a particular path's arrival angle is computed via table look-up in a manner similar to that of the source radiation pattern. Due to the greater complexity of HRTFs, however, the table measurement grid has many more azimuth entries. The HRIRs typically used

here are minimum-phase representations of the raw left and right-ear impulse responses measured for individual subjects using a blocked-meatus technique [modified Crystal River Engineering "Snapshot" system, e.g., see 10]. Interaural time delays (ITDs) are estimated from the raw left and right-ear impulse responses and represented as a separate table of location-dependent pure delays.

The current database, as described in Section 3, contains an HRIR pair and ITD measured every 30° in azimuth and every 18° in elevation. The measured elevations extended from -36° to +54°. (There is, however, no limitation to the density of the HRIR database; denser measurement grids will be used in the future. A recent modification of the Snaphot system now allows HRIRs to be measured for arbitrary densities.) Measured data are extended to the full sphere via biharmonic spline interpolation [14] to form the head-related impulse response table. To form the HRIR and ITD for an arbitrary sound source azimuth and elevation, the four nearest database neighbors are linearly interpolated in real time in a manner similar to equation (1). This results in a 128-tap FIR filter that is applied to the output of the materials filters. 128-tap filters are currently the maximum fidelity that can be reasonably achieved for the direct path with a single Pentium II processor rendering a direct path and six 32-tap reflections (Section 3). Again, there is no inherent limitation on the number of coefficients in SLAB, other than the processing power of the host.

Mixing and equalization. Prior to headphone presentation, the outputs of the direct path and early reflection signal processing paths are summed to form a stereo output pair by a mixer having a 32-bit integer accumulator.

It should be noted that, currently the SLAB system uses head-related transfer functions that have been equalized for Sennheisser 430 headphones prior to storage in the HRIR database. It turns out that this headphone equalization (as well as those of other Sennheisser and many Sony headphones) suppresses energy at frequencies below a few kilohertz compared to the diffuse field. Such suppression has the beneficial effect of shortening the head-related impulse response. As a result, the output equalization filter shown in Fig. 4 is currently a pass-through component.

2.3. Dynamic Behavior

Interactive virtual audio systems are necessarily time varying. As the scenario changes over time, different signal processing parameters are required to render the changing physical effects imposed on the source signal. The difficulty is that all signal processing structures available for implementing the changing scenario are inherently static, assuming fixed coefficients. As a result, care must be taken when updating signal processing parameters. Ideally, new parameters are

switched in sufficiently frequently that the change from one parameter set to the next is imperceptibly small. Certain parameters such as time delays need to be updated every sample to avoid artifacts; minimum-phase head-related transfer functions are somewhat more forgiving.

The main problem with this approach is that it is expensive to compute signal processing parameters from scenario information. There is also the additional issue that peripherals such as head trackers typically provide updates at rates in the range 30 Hz to 120 Hz, so that intermediate scenario data must be developed.

Two methods are used to accommodate a changing scenario: output crossfading, and parameter crossfading (described as commutation in Jot, Larcher and Warusfel [12]). In output crossfading (e.g., as in early versions of the Convolvotron that used non-minimum phase HRTFs [15]), the output is a blend of the input processed according to past parameters and according to present parameters. While the two processing paths use static coefficients, the blend is varied over time to achieve a transition between the parameter sets. Parameter crossfading, by contrast, processes the input according to a varying set of rendering parameters.

Overlap-add methods that operate in the frequency domain are, in effect, a type of output crossfade where the crossfade interval corresponds to the overlap-add interval. Undesirable artifacts when updating the scenario are mitigated by the use of frequent updates and densely measured HRTF databases and/or densely pre-computed binaural room impulse responses [16, 17]. Disadvantages of this method include large memory requirements and the fact that changes in the source, room and receiver characteristics require new measurements or simulations. Other systems utilizing convolution in the time-domain also appear to have used densely-interpolated HRTF databases (e.g., spatial resolution on the order of 2° after interpolation), perhaps combined with a short period of output crossfade, to mitigate possible artifacts due to switching between filters [1, 18].

As developed below, the output crossfade has the drawback of being computationally burdensome. In addition, the output is a mixture of two different systems and might not resemble that of a single system intermediate between the two. Accordingly, the SLAB system uses a variation of parameter crossfading that we term "parameter tracking." Since new scenario information may be available relatively infrequently and contains measurement noise, signal processing parameters computed with each new scenario update become target parameters that are tracked or smoothed. Currently in the SLAB system, the scenario is actually updated frequently compared to other systems (i.e., at an average interval of about 8 ms given a 120 Hz scenario update rate). In

parameter crossfading, there may be multiple update rates for various signal processing parameters. In SLAB, there are two parameter update rates. Every other input frame or 1.45 ms (64 samples), filter coefficients are replaced with ones slightly closer to the target coefficients, while path delays are updated every sample (22.7 μs) to preserve embedded Doppler shifts.

Output crossfade. The signal flow architecture for output crossfading is illustrated in Fig. 7. Periodically, scenario information, ζ, such as source and listener positions are updated and translated to signal processing parameters, η, such as time delays and filter coefficients. The source signal is filtered in two parallel processes, one according to the current parameters η_n and the other according to the past parameters η_{n-1}. The past and current processing outputs are weighted and combined to form the system output:

$$h_\lambda(z) = \lambda\, h(z; \eta_n) + (1-\lambda)\, h(z; \eta_{n-1}). \qquad (3)$$

To smoothly transition from one set of parameters to the next, the crossfade parameter λ is changed from zero to one.

An example of an output crossfade parameter trajectory is shown in Fig. 8. There is a period of time during which only the current parameter processing appears at the output, followed by a transition region during which the system crossfades to the new parameters.

There are two main drawbacks to the output crossfade approach. One is that twice the computation of a stationary system is required during any transition between scenario states. This motivates using a short transition region; however, transition regions shorter than a few milliseconds can create audible artifacts such as clicks and zippering.

The other drawback is that the crossfaded system response does not correspond to that of crossfaded scenario parameters. As illustrated in Fig. 9, the crossfaded response between a system and its delayed version is a pair of impulse responses, rather than an impulse response appearing at a time between the two.

Parameter crossfade. The parameter crossfade signal flow architecture is shown in Fig. 10. Periodically, scenario information is updated, and translated to signal processing parameters η. As illustrated in Fig. 11, a crossfade function is used to generate a sequence of signal processing parameters intermediate between past and current parameters, η_{n-1} and η_n, and the input signal filtered accordingly,

$$h_\lambda(z) = h(z; \lambda\eta_n + (1-\lambda)\eta_{n-1}). \quad (4)$$

Note that the parameter trajectory of Fig. 11 is piecewise linear. If desired, say for accurate Doppler shift rendering, other trajectories that interpolate or smooth the sequence of signal processing parameter values (e.g., see parameter tracking below) may be used.

Provided that the increments between successive intermediate parameter sets are sufficiently small, a smooth transition will result. In the case of updating the FIR filter coefficients of HRTFs, switching in new coefficients every couple of milliseconds appears to be sufficient. However, to avoid audible artifacts, and to maintain the Doppler cues embedded in changing time delays, it is necessary to update any propagation delays and interaural time delays every sample.

By comparison to the output crossfade approach above, the parameter crossfade is computationally less expensive and, as illustrated in Fig. 9, can provide intermediate responses more closely matching those of intermediate values of the scenario parameters. Note, however, that this might not be the case for certain signal processing parameters such as IIR filter coefficients, where crossfaded values between the coefficients of stable filters can result in unstable filters. Under these circumstances, it is best to compute signal processing parameters from crossfaded scenario parameters. Typically, this is not done, since computing signal processing parameters from scenario information is computationally costly.

Parameter tracking. Scenario information such as source position or subject orientation often contains measurement noise. Such information may be processed, trading increased smoothness (i.e., noise reduction) for longer latency, before use in rendering. Alternatively, the scenario information may be translated to a sequence of signal processing parameters, η_n, and smoothed or tracked, as shown in Fig. 12. Again, smoothness is being traded for latency, but in this case at the signal processing parameter level.

The notion of estimating parameter trajectories via tracking is a classic problem from control theory. The goal is to find the parameter trajectory that best fits the measurements, given some knowledge about the noise corrupting the measurements and the parameter dynamics. For instance, in SLAB we assume that the head-tracking measurements are corrupted by a modest amount of additive noise and that the signal processing parameters that correspond to changes in the listener's head position remain relatively constant over a short period of time (e.g., less than 50 ms). This assumption is supported by the observation that during localization, listeners' head motions do not exceed velocities of about 180°/s [4]. As a result, the SLAB parameter tracker approximates a running average of signal processing parameters (over a 15-ms

time period) computed from measured head orientation and other measured scenario parameters. In this way, the additive measurement noise in the computed signal processing parameters is reduced while maintaining the parameter dynamics.

Probably the simplest tracker is the so-called leaky integrator,

$$\eta_t = (1-\alpha)\,\eta_T(t) + \alpha\eta_{t-1}\,, \qquad (5)$$

which computes the next signal processing parameters η_t as a fraction, (1-α), of the target parameters η_T, and a fraction α of the previous parameters η_{t-1}. Note that the leaky integrator is similar in form to an integrator, $\eta_t = \eta_T(t) + \eta_{t-1}$, which simply sums the input target parameters $\eta_T(t)$. The leaky integrator, however, gives less weight to past parameter values, so that its output is an average of only recent target parameters– "recent" meaning several samples when the fraction α is close to zero, and many samples when α is close to one. For example, when the tracking parameter α is close to one, the target parameters have relatively little influence on the current parameters, and the parameters converge slowly to their target. Such a tracking parameter is appropriate in the presence of moderate, zero-mean additive measurement noise, i.e., head-tracker noise, which is suppressed by averaging.

The leaky integrator output decays to the target parameters by a portion (1-α) per update. Setting the fraction α according to

$$\alpha = \exp\{ -1 / n \} \qquad (6)$$

results in a decay time of n updates. Accordingly, with scenario information updated at a rate, f_U, and new signal processing parameters computed at a parameter update rate, f_F, the tracking parameter

$$\alpha = \exp\{ -f_U / f_F \} \qquad (7)$$

gives an approximation to the piecewise linear approach described above (see Fig. 11).

The SLAB system uses leaky integrators to track changes in target signal processing parameters computed from scenario updates that are received roughly every eight milliseconds (f_U = 120 Hz). Leaky integrators tracking changing time delays are updated every sample (22.7 μs; f_F = 44.1 kHz), so that Doppler cues are maintained. To minimize computational cost, the leaky

integrators tracking the FIR filter coefficients are updated less frequently, every other input frame or 1.45 ms (64 samples; f_F = 690 Hz),

3. THE CURRENT SLAB SYSTEM IMPLEMENTATION

As mentioned in the introduction, the SLAB system is a work-in-progress. While the signal flow architecture illustrated in Fig. 4 has not yet been fully implemented, significant progress has been made in developing the modular software architecture to support its full implementation. The sections below discuss the components of the signal processing architecture that have been implemented so far and provide preliminary data regarding the performance of the system as well as verification of the basic signal processing algorithm.

3. 1 Preliminary System Measurements

To date, the components of the signal processing architecture that have been implemented include the interpolated delay line and the spherical spreading loss and HRTF components of the FIR filter. As described in a previous presentation [19], these components have been utilized in an initial simulation scenario based on the image model with a direct path and six first-order reflections. Scenario specifications and current performance characteristics achievable with a single 450 MHz Pentium II processor are summarized in Table 1.

One of the major implementation hurdles overcome in developing such a system has been to achieve adequate dynamic performance in Windows, an environment that is not ideally suited to real-time performance. A measurement of the internal latency of SLAB for the simulation described above provides a preliminary assessment of its dynamic performance. The internal latency is the delay between acquisition of location data by the host rendering system and the rendered audio output. Total system latency, or end-to-end latency, on the other hand, refers to the time elapsed from the transduction of an event or action, such as movement of the head, until the consequences of that action cause the equivalent change in the virtual sound source location. Latencies are contributed by individual components of a virtual audio system, including tracking devices, signal processors, software to control these devices, and communications lines [2-4].

Here, internal latency was measured in the following manner. An I/O port of the SLAB host renderer and the SLAB headphone output were connected to the begin- and end-time inputs of an interval timer, respectively. The I/O port was used to indicate when new scenario information was received by the SLAB host rendering system. With this configuration, the internal latency measurement was 24 ms. A preliminary value of 24 ms is quite encouraging considering the inherent difficulties in managing low-latency Windows audio output, let alone

while performing a high cost simulation that consumes a significant portion of available system resources. The total system latency would necessarily be somewhat longer than 24 ms since it would include additional head-tracker, communication line, and client/server latencies. Implementation of the full signal path of Fig. 4 will no doubt also add to the latency of the system, although it is unclear how significant this increase might be, particularly since the signal processing algorithms have yet to be fully optimized.

3.2 Verification of Signal Processing Algorithms

In this section, the current SLAB signal processing verification procedure is discussed. The SLAB auralization unit is being developed in two environments. Modeling of the signal processing algorithms is being conducted in MATLAB (v. 5.2, The Mathworks) and Microsoft Visual C++ is used for real-time implementation. The anechoic impulse responses of both implementations were compared to verify equivalent behavior. To generate the C++ responses, a utility was written that outputs impulse responses for a range of source and listener positions. These responses were read into a MATLAB utility which graphically compares C++ responses, MATLAB responses, and HRTF database responses (when available). The verification graphs for a succession of source locations is shown in Fig. 13. The source was placed 10 cm from the listener at -30, -40, -50, and -60 degrees azimuth. The listener was placed at the origin. When an un-interpolated HRTF measurement was available (-60 and -30), it was also displayed. In this instance, the only processing performed was spherical spreading loss in order to align the magnitude response curves.

As can be seen from the verification graphs, the C++ implementation is consistent with both the MATLAB model and the HRTF database. The minor deviations that exist seem reasonable considering the lower precision of the C++ implementation. The C++ implementation uses 16-bit integer, floating point, and double-precision floating point as data types; the MATLAB model uses only double-precision floating point. Another source of deviation could be implementation differences in the C++, SPL (Intel Signal Processing Library), and MATLAB math libraries. C++ and SPL libraries emphasize performance; MATLAB libraries emphasize computational accuracy. But, as illustrated by the verification results, the C++ and SPL real-time implementation yielded minimal computational artifacts.

This verification procedure highlights the importance of HRTF database density. The current database format contains HRIRs measured at every 30° of azimuth. Thus, the graphs in Figures 13a and 13d correspond to measured database locations. The graphs in Figures 13b and 13c, on the other hand, correspond to linearly interpolated HRIRs. The magnitude notch seen in Fig. 13a at 1000 Hz does not gracefully migrate to the 880 Hz notch seen in Fig. 13d. It, instead,

disappears in Fig. 13b and reappears in Fig. 13c. This type of linear interpolation artifact will be ameliorated through the use of denser HRTF databases that are now being measured.

4. CONCLUSIONS

The goal of the system described here, Sound Lab (SLAB), is to provide an experimental platform with low-level control of a variety of signal-processing parameters for conducting psychoacoustic studies. For example, some of the parameters that will be examined in future experiments include the number, fidelity (number of filter taps) and positioning (correct vs. incorrect) of reflections. System latency and update rate can also be manipulated. The project is also an attempt to provide the basis of a low-cost, software-based system for dynamic synthesis of virtual audio over headphones that does not require an array of special purpose signal processing hardware.

The development of SLAB is currently a work-in-progress. To date, the components of the signal processing architecture that have been implemented include the interpolated delay line and the spherical spreading loss and HRTF components of the FIR filter. These components have been utilized in an interactive simulation scenario based on the image model with a direct path and six first-order reflections and implemented on a single 450 MHz Pentium II.

One of the major implementation hurdles overcome in developing such a system has been to achieve adequate dynamic performance in Windows, an environment not ideally suited to real-time processing. Measurement of the internal latency of the system provides a preliminary assessment of the dynamic performance of SLAB. This preliminary value of 24 ms is quite encouraging considering the inherent difficulties in managing low-latency Windows audio output, let alone while performing a high cost simulation that consumes a significant portion of available system resources. Implementation of the full signal path (Fig. 4) will add to the latency of the system, although it is unclear how significant this increase might be, particularly since the signal processing algorithms have yet to be fully optimized.

Informal listening tests indicate that the dynamic behavior of the system is both smooth and responsive. The smoothness is enhanced by the 120-Hz scenario update rate, as well as the parameter tracking method which produces rather high parameter update rates; i.e., time delays are updated at 44.1 kHz and the FIR filter coefficients are updated at 690 Hz. The responsiveness of the system is enhanced by the low latency of 24 ms. The scenario update rate, parameter update rates, and latency all compare favorably to other virtual audio systems [e.g., 1, 8, 16, 18]. While the current scenario implemented in SLAB is incomplete, we don't expect implementation of the full signal path to significantly

increase latency or decrease filter coefficient update rates. At this point, it is difficult to estimate the impact of simulating higher-order reflections or late reverberation on SLAB system performance.

In addition to implementing the full signal path and optimizing the signal processing algorithms, future development will include exploration of several systems and performance issues. A client-server architecture has recently been implemented so that SLAB is isolated from other system components, such as the head tracker and the experimental control software. In order to facilitate simulation of more complex room models in real time, SLAB could be implemented as a distributed system to further spread out the computational load over multiple workstations. Multiple processor support is also being researched as a method for increasing computational resources.

ACKNOWLEDGEMENTS

Work supported by NASA and by the Navy (SPAWARSYSCEN, San Diego). Thanks to Durand Begault and Bernard Adelstein for editorial comments on the paper.

REFERENCES

1. Sandvad, J. 1996. Dynamic aspects of auditory virtual environments. 100th Convention of the Audio Engineering Society, Copenhagen, preprint 4226.

2. Wenzel, E. M. (1997) Analysis of the role of update rate and system latency in interactive virtual acoustic environments. 103rd Convention of the Audio Engineering Society, New York, NY, 1997, preprint 4633.

3. Wenzel, E. M. (1998). The impact of system latency on dynamic performance in virtual acoustic environments. Proceedings of the 15th International Congress on Acoustics and 135th Meeting of the Acoustical Society of America, Seattle, WA, 2405-2406.

4. Wenzel, E. M (1999) Effect of increasing system latency on localization of virtual sounds. Proceedings of the Audio Engineering Society 16th International Conference on Spatial Sound Reproduction. (Rovaniemi, Finland). April 10-12, 1999. New York: Audio Engineering Society, 42-50.

5. Horbach, U., Karamustafaoglu, A., Pelligrini, R., Mackensen, P. & Thiele, G. (1999) Design and applications of a data-based auralization system for surround sound. 107th Convention of the Audio Engineering Society. (Munich, Germany).

6. Pelligrini, R. S. (1999) Comparison of data- and model-based simulation algorithms for auditory virtual environments. 107th Convention of the Audio Engineering Society. (Munich, Germany).

7. Takala, T., Hanninen, R., Valimaki, V., Savioja, L., Huopaniemi, J., Huotilainen, T. & Karjalainen, M. (1996) An integrated system for virtual audio reality. 100th Convention of the Audio Engineering Society, preprint 4229, 43-46.

8. Savioja, L., Huopaniemi, J., Lokki, T. & Väänänen, R. (1999) Creating interactive virtual acoustic environments. Journal of the Audio Engineering Society, 47, 675-705.

9. Begault, D. R. (1996) Audible and inaudible early reflections: Thresholds for auralization system design. 100th Convention of the Audio Engineering Society, Copenhagen, Denmark,preprint 4244.

10. Begault, D. R., Wenzel, E. M., Lee, A. S., and Anderson, M. R. (1999) Direct comparison of the impact of head tracking, reverberation, and individualized head-related transfer functions on the spatial perception of a virtual speech source. 108th Convention of the Audio Engineering Society, Paris, France.

11. Timo I. Laakso, T. I., Vesa Valimaki, V., Matti Karjalainen, M. and Unto K. Laine, U. K. (1996) Splitting the unit delay - tools for fractional delay filter design. IEEE Signal Processing Magazine, 13, .

12. Jot, J. M., Larcher, V. and Warusfel, O. (1995) Digital signal processing issues in the context of binaural and transaural stereophony. 98th Convention of the Audio Engineering Society, Paris, France, preprint 3980.

13. Pierce, A. (1989) Acoustics. Acoustical Society of America: New York, p. 219.

14. David T. Sandwell, D. T. (1987) Biharmonic spline interpolation of GEOS-3 and SEASAT altimeter data. Geophysical Research Letters, 2, 139-142.

15. Convolvotron manual, release 1.2 (1990) Crystal River Engineering, Inc., pp. 3.2-3.3.

16. Bronkhorst, A. W. (1995) Localization of real and virtual sources. Journal of the Acoustical Society of America, 98, 2542-2553.

17. Gardner, W. G. (1995) Efficient convolution without input-ouput delay. Journal of the Audio Engineering Society, 43, 127-136.

18. Sahrhage, J., Blauert, J. and Lehnert, H. (1996). Implementation of an auditory/tactile virtual environment. Proceedings of the 2nd FIVE International Conference, Palazzo dei Congressi, pp. 18-26.

19. Miller, J. D., Abel, J. S. and Wenzel, E. M (1999) Implementation issues in the development of a real-time, Windows-based system to study spatial hearing. Journal of the Acoustical Society of America, 105, 1193.

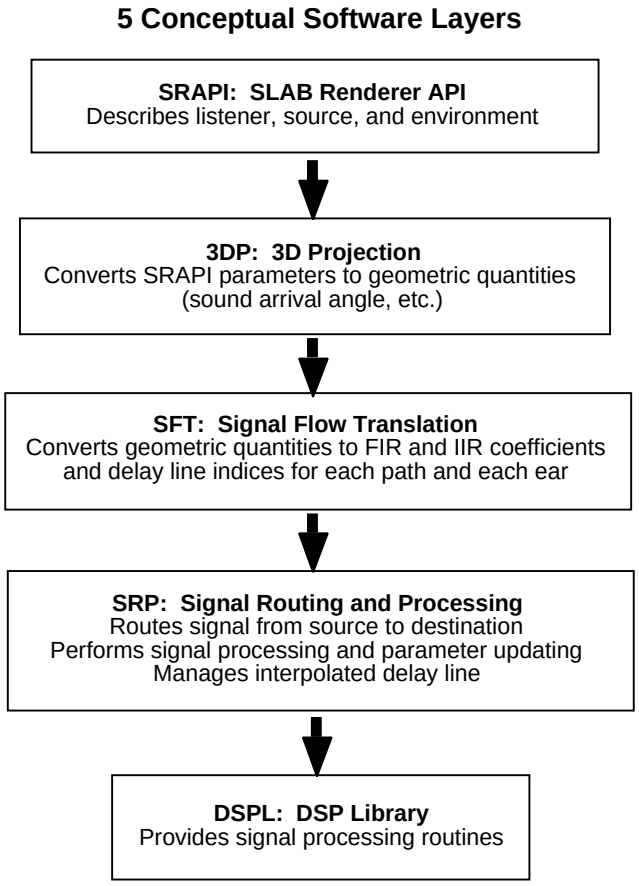
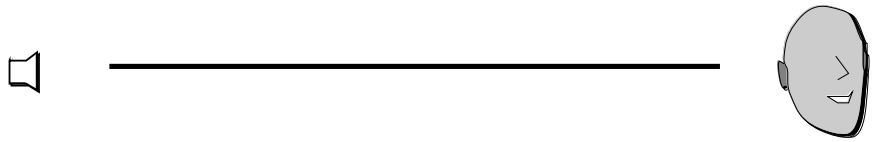


Fig. 1. Conceptual illustration of the five functional layers comprising the software architecture of the Sound Lab system.



SOURCE
- signal waveform
- radiation pattern
- position, orientation, velocity
- level
- size

ENVIRONMENT
- propagation medium
  - sound speed, spherical spreading loss, air absorption
- reflection characteristics
  - surface position, orientation, reflection / transmission filter
- reverberation characteristics
  - RT60, equalization

LISTENER
- head-related transfer function
- interaural time delay
- position, orientation, velocity

Fig. 2. Illustration of the physical scenario simulated in the Sound Lab system.

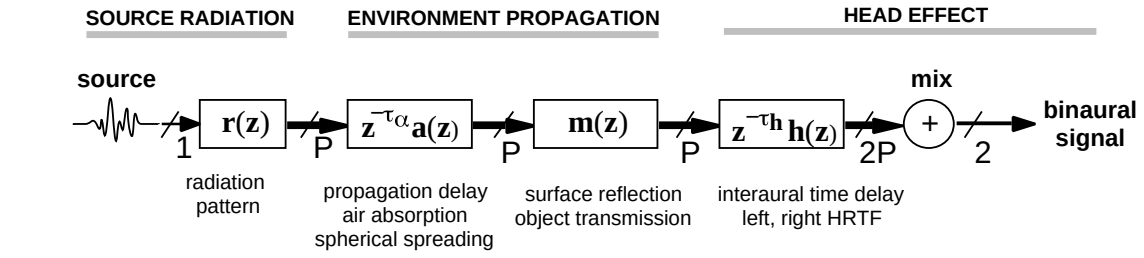


Fig. 3. The physical signal flow is shown as a block diagram that partitions the properties of the physical scenario into the relevant signal processing components.

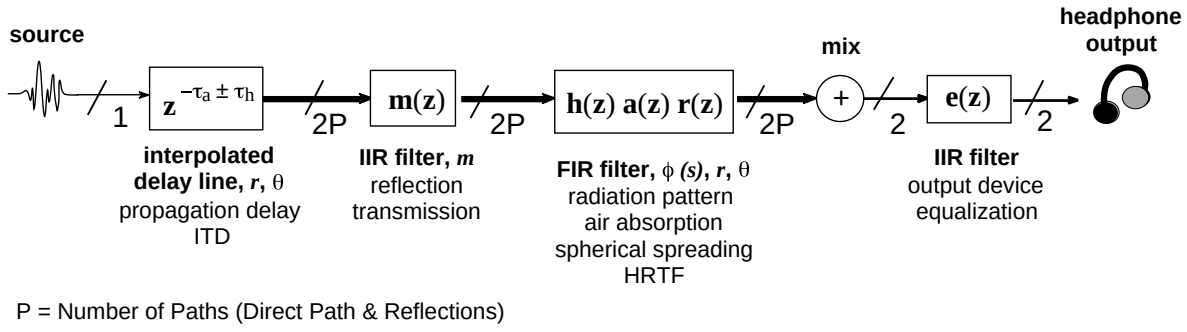


Fig. 4. The SLAB signal flow, or "auralization unit," is shown as a block diagram that partitions the properties of the physical scenario into the relevant signal processing components as they are implemented in the SLAB system architecture.

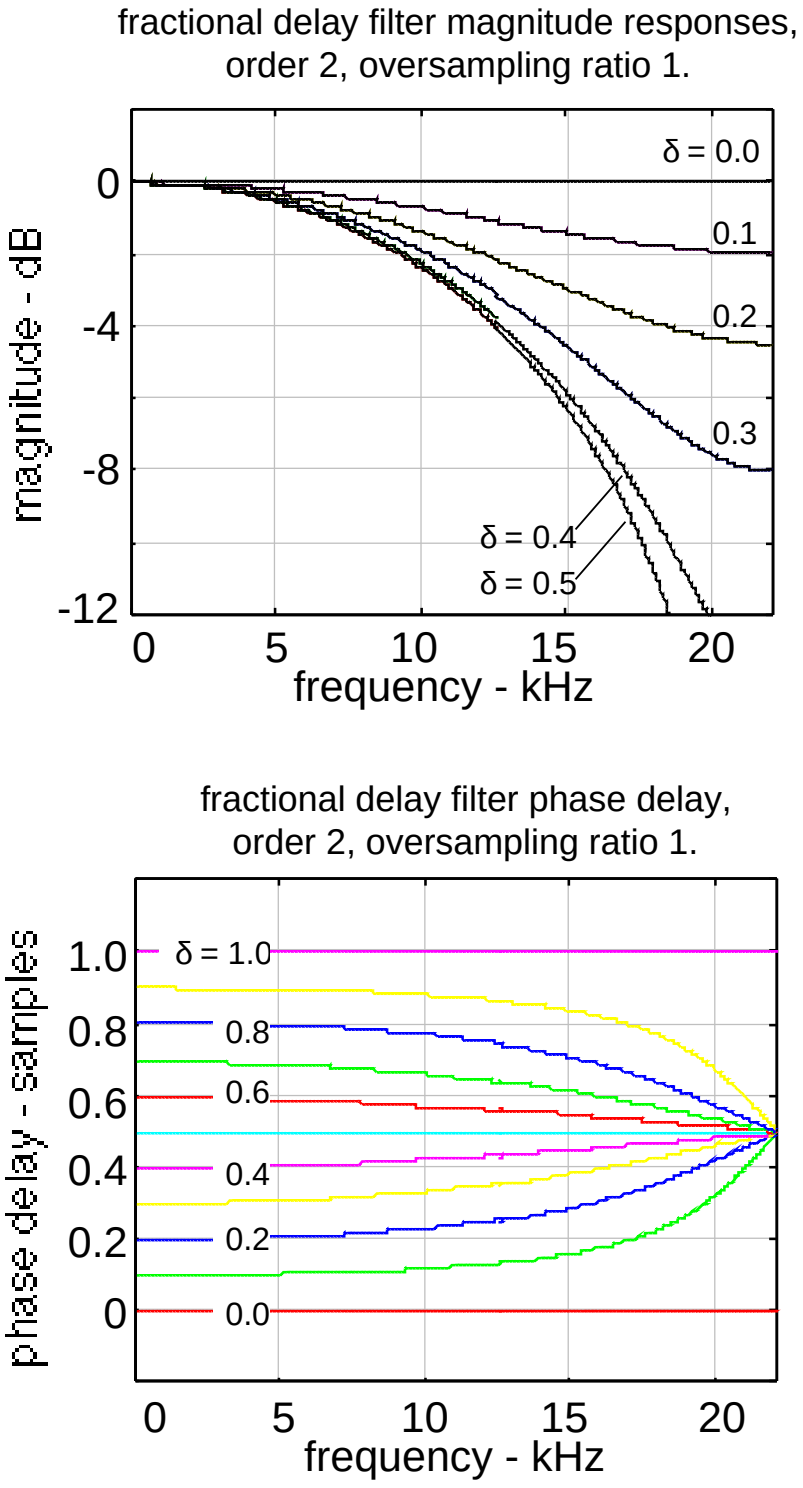


Fig. 5. The magnitude transfer function and phase delay is plotted for a linearly interpolated delay without up-sampling of the source signal. The parameter δ is the value of the delay expressed as fractions of a 44.1 kHz sample.

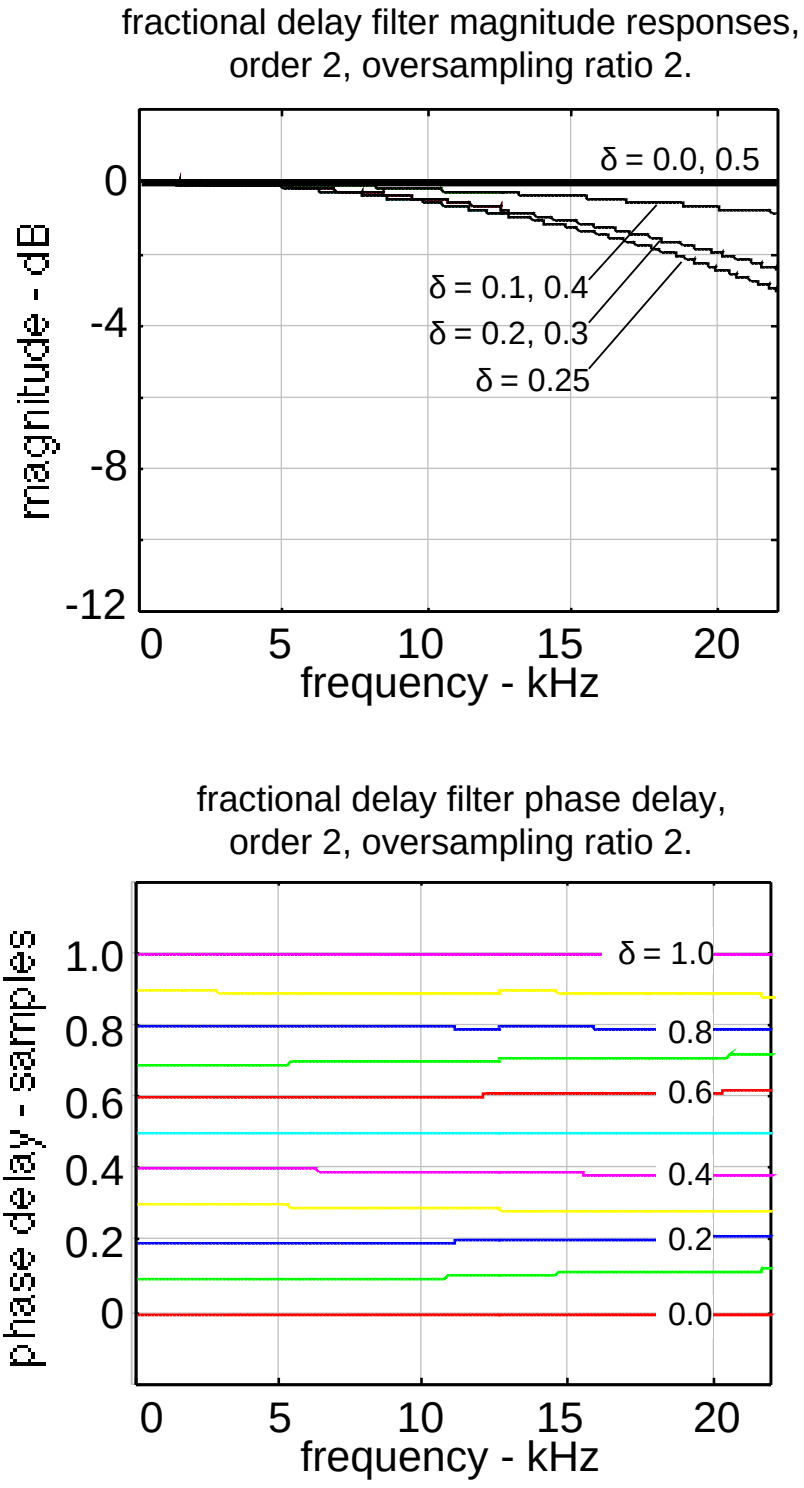


Fig. 6. The magnitude transfer function and phase delay are plotted for a linearly interpolated delay using a two-times up-sampled source signal. The parameter δ is the value of the delay expressed as fractions of a 44.1 kHz sample.

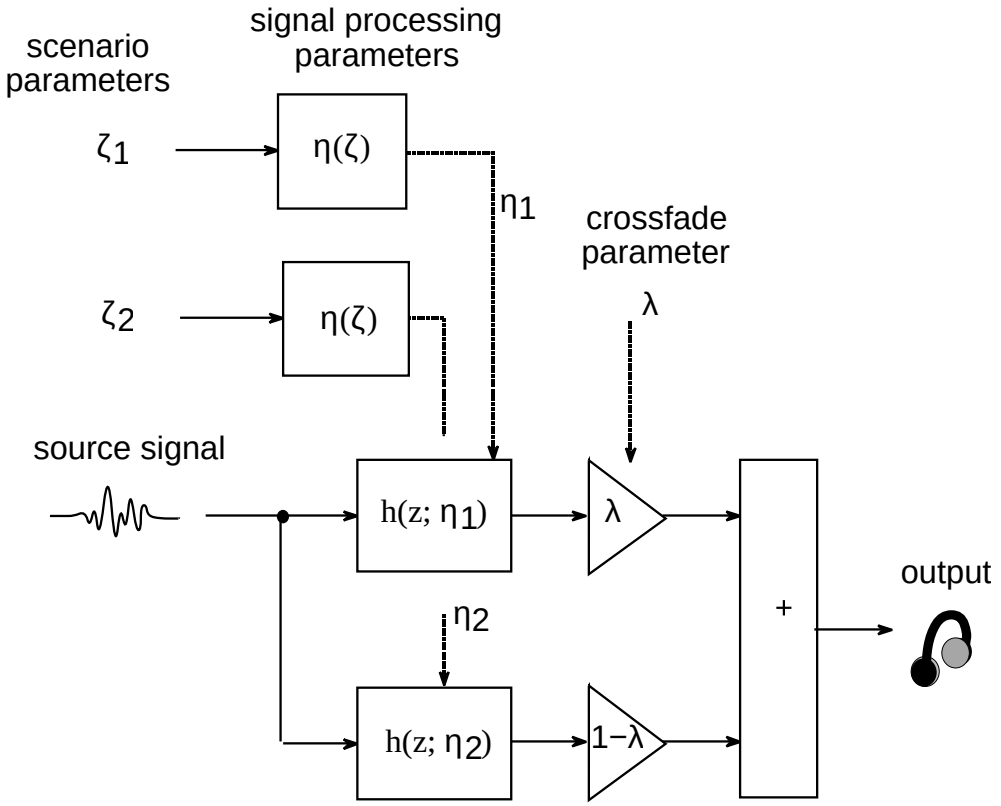


Fig. 7. Illustration of the signal flow architecture for output crossfading.

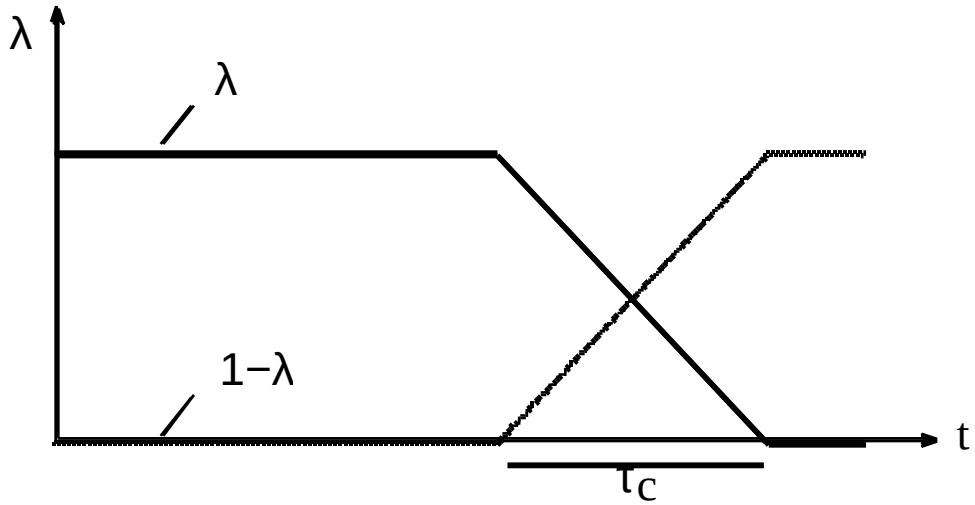


Fig. 8. Illustration of the parameter trajectory for output crossfading. The crossfade interval is length τ_c.

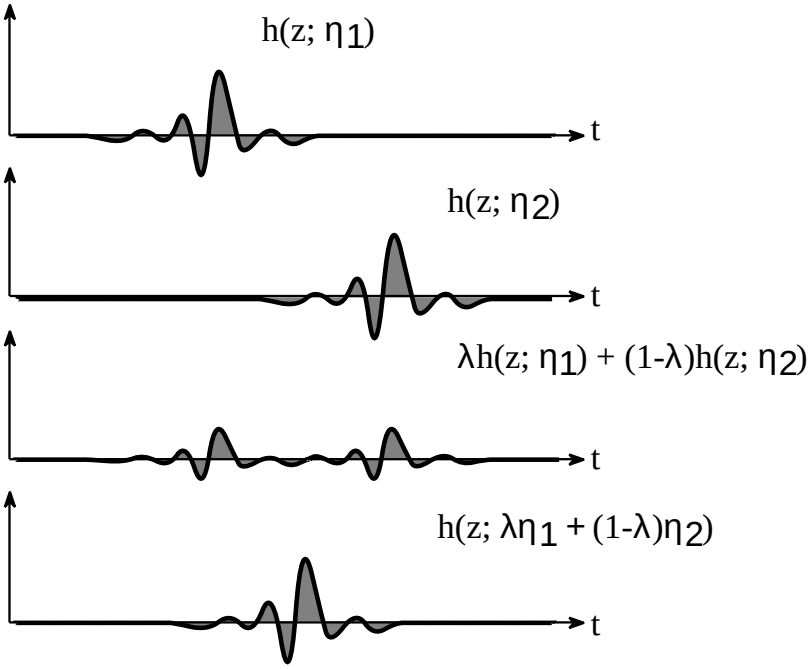


Fig. 9. Illustration of the system response resulting from output crossfading and parameter crossfading. The top panels represent the two signals to be crossfaded. The third panel shows the result of crossfading the output, which produces in two reduced-amplitude signals occurring at incorrect times. The last panel shows the output response for parameter crossfading, a single full-amplitude signal occurring at the correct intermediate time.

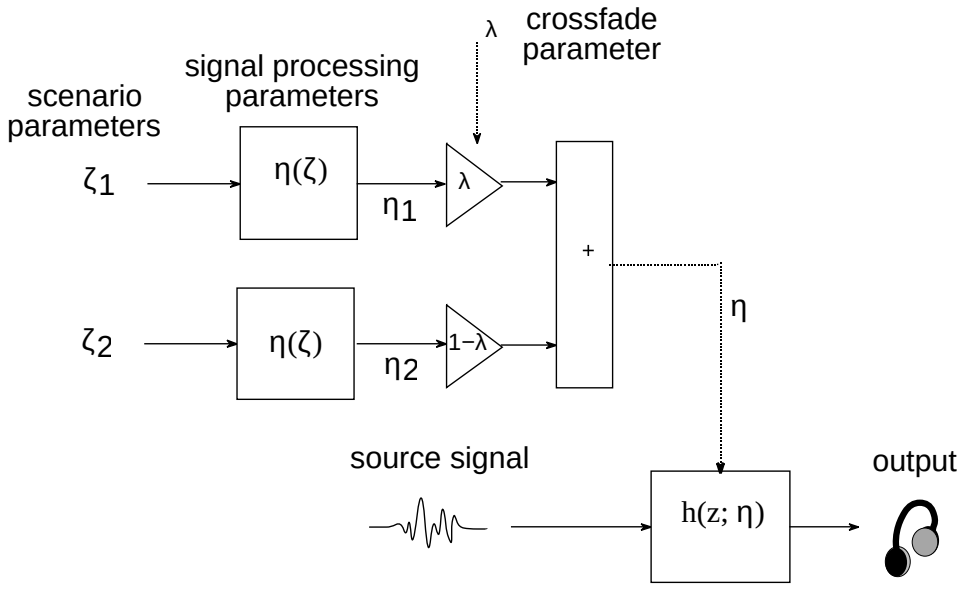


Fig. 10. Illustration of the signal flow architecture for parameter crossfading.

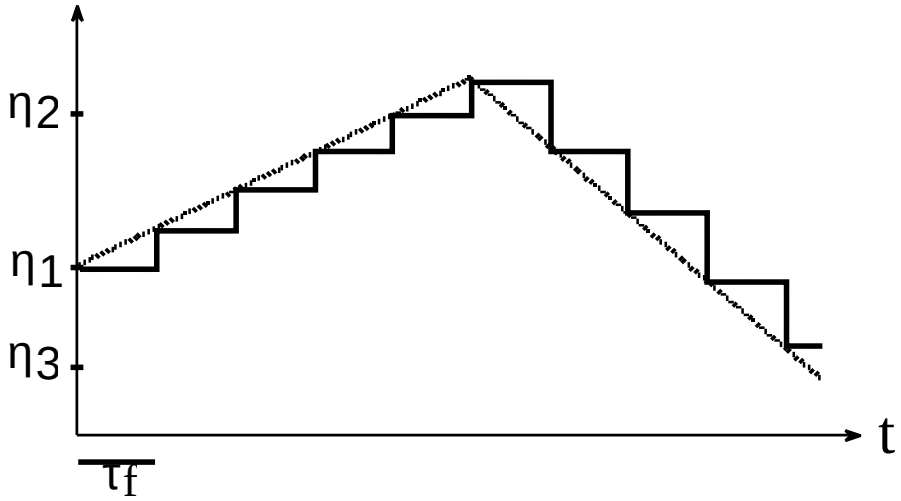


Fig. 11. Illustration of the parameter trajectory for parameter crossfading. Signal processing parameters η are linearly faded between successive values. The signal parameters implemented form a staircase having step width equal to the processing frame duration of τ_f.

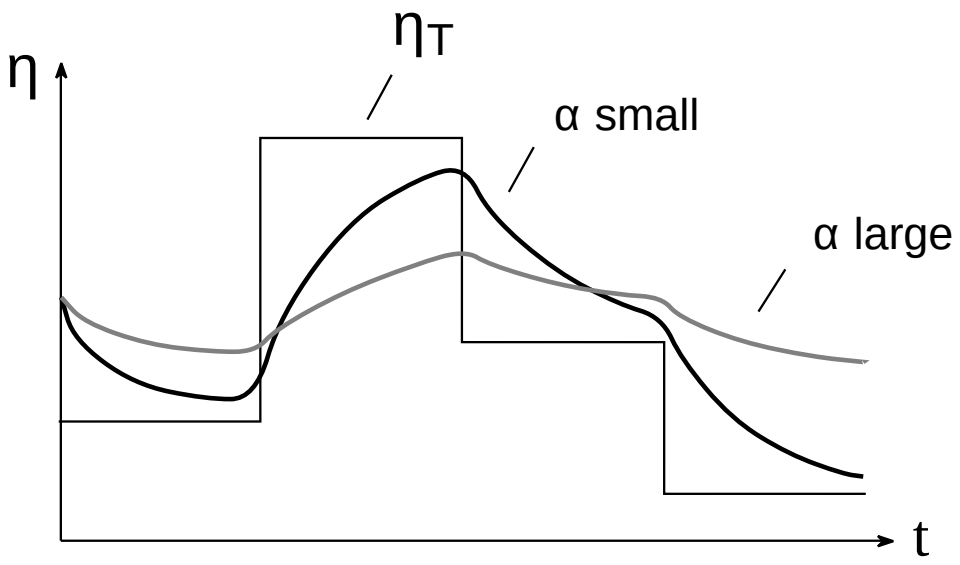
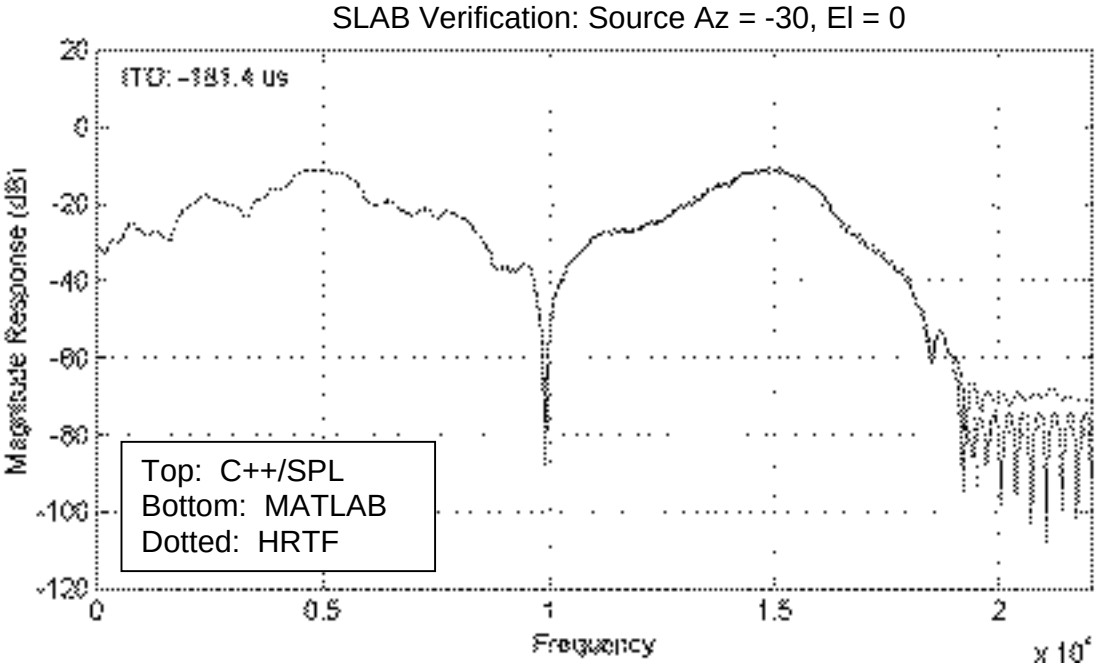


Fig. 12. Illustration of the parameter trajectory for the method of parameter tracking. Target parameters η_T are tracked using a leaky integrator in which each new parameter set is a constant proportion closer to the target set than was the previous set. When the tracking constant is close to one (α large), the parameters move slowly toward the targets. When the tracking constant is close to zero (α small), the tracked parameters move quickly toward their targets.

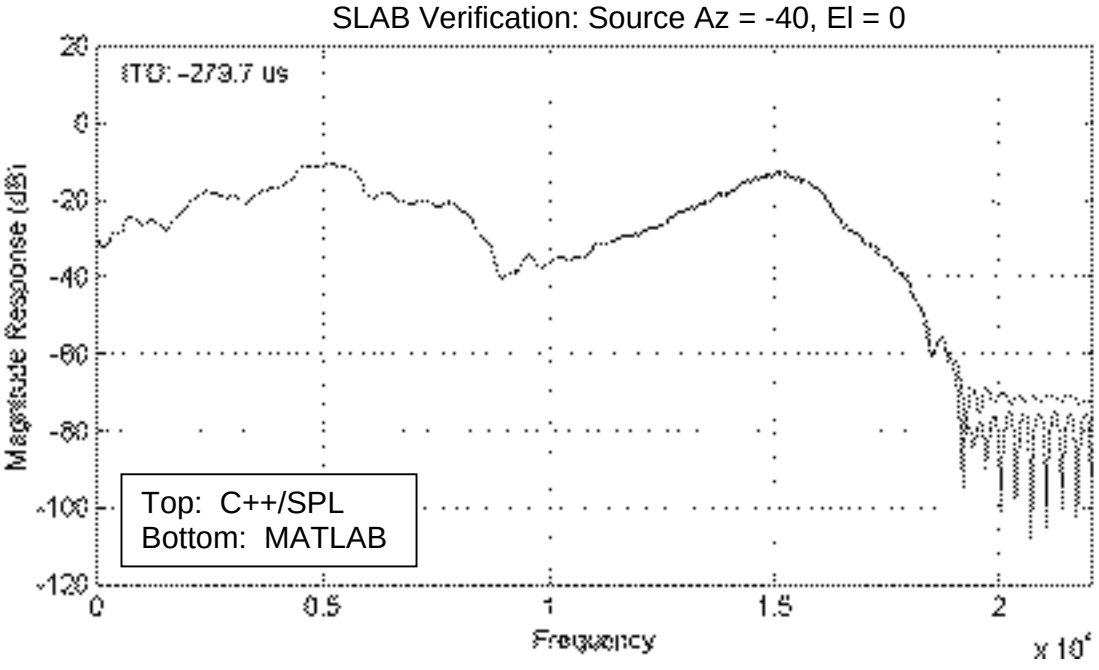
Simulation Scenario Specifications
Rectangular room; Image model
Number of first-order reflections: 6
Number of direct path FIR taps: 128
Number of reflection FIR taps: 32

System Dynamics
Sample Rate: 44.1 kHz
Internal system latency: 24 ms
Scenario update rate: 120 Hz
Delay line update: every sample (22.7 µs)
DSP coefficient update: every 64 samples (1.45 ms)

Table 1. Simulation scenario and performance characteristics for an initial implementation using the SLAB system.

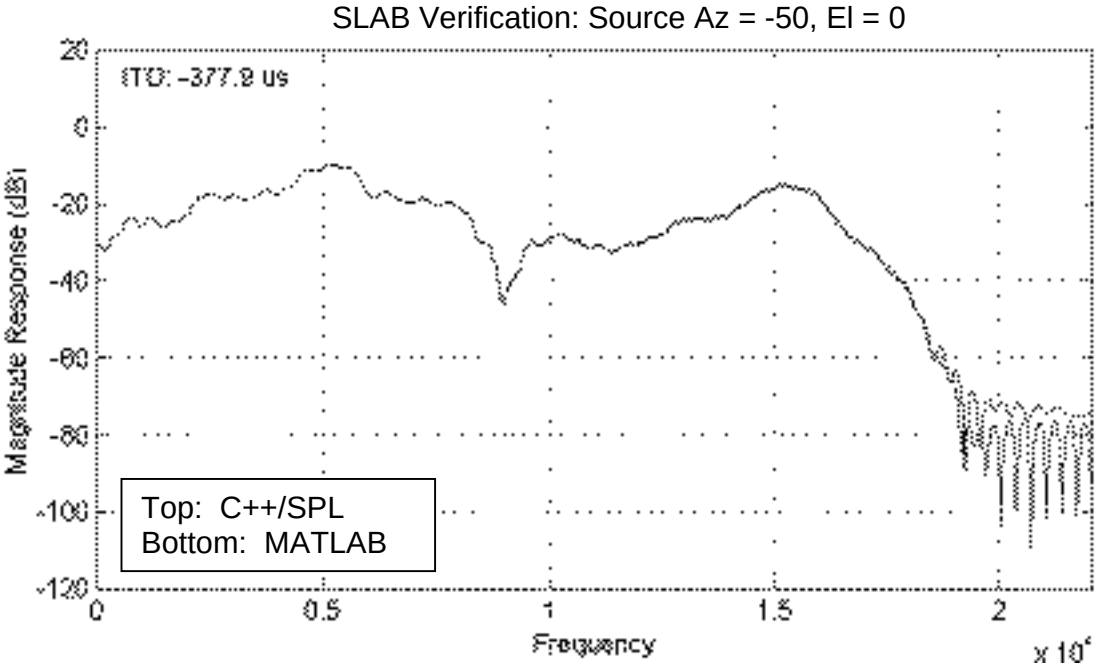


(a)

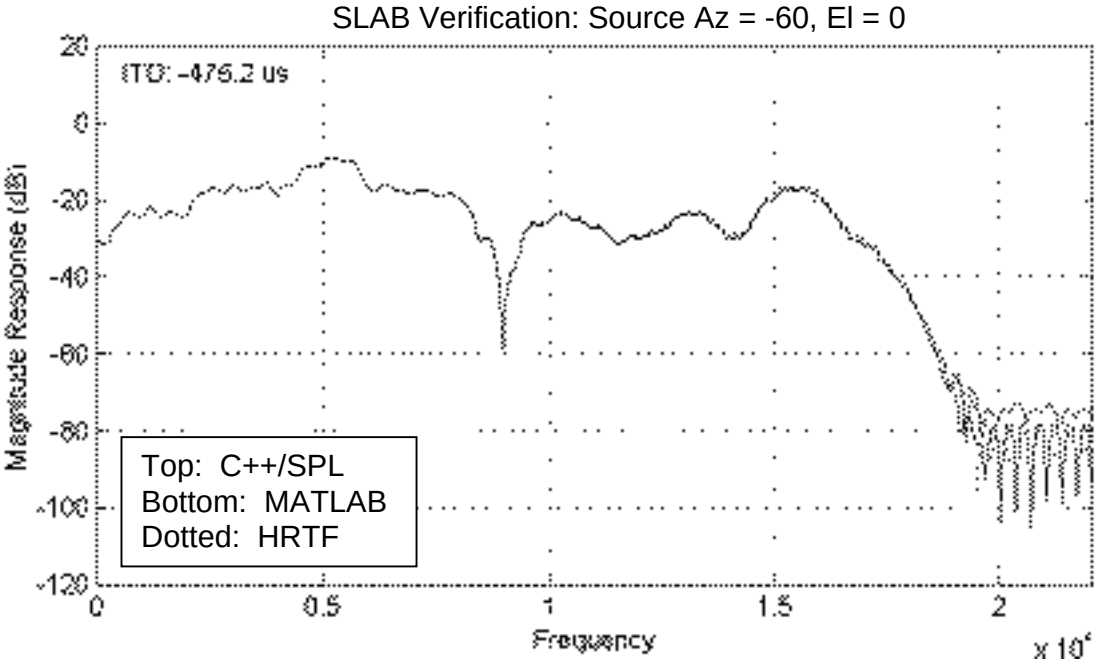


(b)

Fig. 13. Comparison of C++/SPL, MATLAB, and HRTF left-ear magnitude responses (anechoic). (a) - (d) illustrate that the C++/SPL implementation, the MATLAB model, and physical measurement all behave consistently across a range of source azimuth locations.



(c)



(d)

Fig. 13 continued. Comparison of C++/SPL, MATLAB, and HRTF left-ear magnitude responses (anechoic).