Proceedings of Meetings on Acoustics

OCTOBER 07 2026

Building a real-time convolution system for assessing vocal effort of teachers

Bethany Wu; Brian E. Anderson ; Brian A. Mazzeo

Check for updates

Proc. Mtgs. Acoust. 54, 060010 (2024)
https://doi.org/10.1121/2.0002408

View Online
Export Citation

Connected Content

This is a companion to: Building a real-time convolution system for assessing vocal health of teachers

Articles You May Be Interested In

An acoustic analysis of vocal effort and speaking style

Proc. Mtgs. Acoust. (April 2022)

Dichotic listening performance and effort as a function of spectral resolution and interaural symmetry

J. Acoust. Soc. Am. (August 2021)

Effects of speech style, room acoustics, and vocal fatigue on vocal effort

J. Acoust. Soc. Am. (May 2016)

12 October 2026 23:32:43

ASA
LEARN MORE
Advance your science and career as a member of the
Acoustical Society of America

ASA
ACOUSTICAL SOCIETY OF AMERICA

Volume 54 http://acousticalsociety.org/

Acoustics Week in Canada
Joint Meeting
186th Meeting of the Acoustical Society of America
and the Canadian Acoustical Association

Ottawa, Ontario, Canada
13-17 May 2024

Speech Communication: Paper 3aSC3

Building a real-time convolution system for assessing vocal effort of teachers

Bethany Wu and Brian E. Anderson
Department of Physics & Astronomy, Brigham Young University, Provo, UT, 84602, USA; bethanywu5@gmail.com; bea@byu.edu

Brian A. Mazzeo
Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT, 84602, USA; bmazzeo@byu.edu

12 October 2026 23:32:43

Due to prevalent vocal health issues in teachers, the acoustics of K-12 classrooms has become a common topic of study in acoustics. One way to understand the impact of a classroom's acoustics on speech is through real-time convolution of speech with a binaural room impulse response (BRIR). This is done by having a talker seated in an anechoic chamber and their vocal effort can be assessed while using the real-time convolution system to simulate the acoustics of a variety of classroom conditions. Keeping the talker in one physical space provides more control over the testing environment. A system that can successfully execute convolution in real-time requires parameters to be fine-tuned, an optimized algorithm, and appropriate hardware. Current efforts and lessons learned during the development of this system are shared. Goals for a finished real-time convolution system include specific testing to determine the effects of background noise and reverberation on a teacher's vocal effort.

Proceedings of Meetings on Acoustics, Vol. 54, 060010 (2026); https://doi.org/10.1121/2.0002408.

*©2026 Author(s). All article content, except where otherwise noted, is licensed under a Creative Commons Attribution 4.0 International (**CC BY**) License (https://creativecommons.org/licenses/by/4.0/).*

1. INTRODUCTION

Teachers are recognized as one of the largest groups of professional voice users. Because of this, their vocal health is at risk daily, which can cause lasting effects and impose significant financial costs for them. Elementary and high school teachers especially are three times more likely than individuals working other occupations to develop issues related to vocal health.[1] A study relating classroom acoustics to teacher vocal health found that changes in voice symptoms correlated with the teachers' average noise exposure during the workday.[2] In the study, noise exposure was quantified by measuring background noise levels and the reverberation time (RT60) of the classrooms that the teachers were tested in.

Previous studies with this same goal conducted tests by placing teachers inside different classrooms and repeating test procedures to obtain results. However, physically bringing teachers to multiple locations to achieve a diverse survey of classrooms is troublesome and can introduce unwanted variables into testing. For this reason, a real-time convolution system (RTCS) is being created to virtually bring teachers to multiple classrooms, as they physically stay inside an anechoic chamber. This RTCS would produce binaural auralizations of any classroom environment, which would be created through convolution of real-time speech with prerecorded oral binaural room impulse responses (OBRIRs).

To both ensure the accuracy of the system's output and have the ability to state all the assumptions and shortcuts used, the RTCS was coded from scratch. In order to create realistic auralizations with an RTCS, the latency of the real-time convolution computation must be as low as possible and at a shorter time than the earliest sound reflections might arrive, which was estimated to be about 6 ms. This paper highlights the various efforts made to minimize latency by optimizing both the software and hardware used for the RTCS. The real-time convolution algorithm was optimized to further decrease total system latency (TSL); this process included finding optimal system parameters to ensure both an accurate system output and low latency. Overall, the final implementation and parameters chosen for this RTCS resulted in ideal latency for the purposes of this research. Problems with noise in the RTCS still exist but this paper describes progress being made.

Section 2 of the paper will describe convolution processing for digital signals to minimize the processing time. Convolution is the mathematical process needed to modify the speech signal with the room's response so that it sounds like the talker is in that room. Section 3 will describe various implementation strategies that we explored, along with the hardware and software needs for those strategies. It also describes the approach we decided to take and the reasons for our decision. Section 4 describes the latency issues with trying to make the processing happen fast enough that the talker does not experience any delays such that the processing is happening in real time. Section 5 discusses how settings were optimized for the boards and issues that remain with their use that are currently being worked on. Finally we provide concluding remarks in Section 6.

2. CONVOLUTION IN REAL TIME

An auralization is created by computing the convolution of a room's recorded finite impulse response (FIR) with a source signal (such as a speech recording having L samples). Convolution, which commonly implies using signals in the time domain, computes a summation of time-shifted impulse responses. Specifically, the convolution of two signals is formed by taking each point in the source signal and multiplying it by each point in the FIR signal and preserving the relative time delays of the FIR. Each of these L multiplication results, which are just scaled versions of the FIR, are then time-staggered so that they start at the same time as the individual source signal sample. The resulting time-staggered and scaled FIR results are then summed together to form the overall convolution output.

Another important thing to note about convolution is the length of each signal involved. For an input $x[n]$ consisting of L samples and FIR $h[n]$ consisting of P samples (where n is an index), the output signal

12 October 2026 23:32:43

$y[n]$ (from the linear convolution) will be of length N=L+P-1 samples long. Discrete time convolution is expressed mathematically in the time domain in Eq. (1), where $x[n]$ represents the input signal, $h[n]$ is an FIR, and N is the total number of samples in the output; this equation also illustrates the commutative property of convolution.

$$y[n] = x[n] * h[n] = \sum_{m=0}^{N-1} x[m]\, h[n-m] = h[n] * x[n] \tag{1}$$

Convolution can also be seen as a modification of the input $x[n]$ by the FIR $h[n]$. For the purposes of this RTCS, $x[n]$ represents blocks of real-time speech ($x(t)$) from a talker, and $h[n]$ is a FIR recorded by one ear microphone (left or right) of a classroom's OBRIR. The resulting signal, $y[n]$, is the auralization created, sounding as if the talker was speaking in the room where the OBRIR, $h[n]$, was measured.

For long input signals, the number of computations can become very large, which can potentially increase latency for the RTCS. To combat this, convolution can be done in the frequency domain by using Fast Fourier Transform (FFT) methods. This changes the order of the complexity of convolution for an input signal of length N from $\mathcal{O}(N^2)$ in the time domain to $\mathcal{O}(N \log_2 N)$ in the frequency domain, which is only strictly true when N is a power of two.[3] Convolution in the frequency domain is computed by Eq. (2),

$$y[n] = x[n] * h[n] = \mathcal{F}^{-1}\{\mathcal{F}\{x[n]\} \cdot \mathcal{F}\{h[n]\}\} = \mathcal{F}^{-1}\{X(\omega) \cdot H(\omega)\} \tag{2}$$

where $\mathcal{F}$ denotes the Discrete Fourier Transform in this case; $X(\omega)$ is the source spectrum while $H(\omega)$ is the frequency response function and ω represents angular frequency. As mentioned before, for long input signals or a long FIR, convolution can become very computationally expensive. To further combat this, a few different methods were used to decrease the computational complexity (and therefore latency) of the RTCS, which are discussed in the sections below.

12 October 2026 23:32:43

A. BLOCK CONVOLUTION AND THE OVERLAP-SAVE METHOD

When looking at the case when the input signal $x(t)$ is significantly long, convolution with an FIR is commonly performed in blocks through block convolution.[4] This process suggests dividing the input signal into smaller blocks $x_i[n]$ (where i is an index representing the block number) and then continuously performing the convolution of each block with the FIR $h[n]$. Then the convolution output for each block can be correctly combined to form the overall convolution result. The overlap-save method employing FFTs is one way to efficiently perform block convolution using less memory, direct processing, and avoiding redundant computations.

Using the overlap-save method starts by breaking the input signal into overlapping blocks; the amount of overlap depends on the length of the FIR chosen. Assume the input signal is initially divided into successive blocks of length L and the FIR is P samples long. The overlapping blocks of the input signal will be of length L+P-1, where each block overlaps the preceding block by P-1 samples. Next, the circular convolution result of each overlapping block is calculated using the FFT method.

As mentioned before, when performing the linear convolution of an input with L samples and FIR with P samples, the output will be L+P-1 samples long. Conversely, when performing the circular convolution of the same lengthed input signal and FIR, the length of the output will be the maximum of the two lengths L and P. Therefore, in the circular convolution of an overlapping block of length L+P-1 and FIR of length P, the output will be of length L+P-1. Due to the circular nature of the FFT-based convolution, the first P-1 samples are discarded leaving L samples from each convolution result. These L samples can then be concatenated directly, without overlap or addition, to form the correct convolution result of the whole input signal with the FIR. The overlap-save process is illustrated in Fig. 1.

Because the input signal used for convolution in this project is speech recorded in real time, it can be classified as infinitely long. Because of this, block convolution and the overlap-save method are used to compute the convolutions in the RTCS. To perform block convolution, the input signal is continuously recorded in blocks of chosen length. The input block size (in ms) is one of the RTCS parameters discussed in Section 4. To ensure that speech is recorded continuously (without gaps), a special structure called a ping-pong buffer is used.

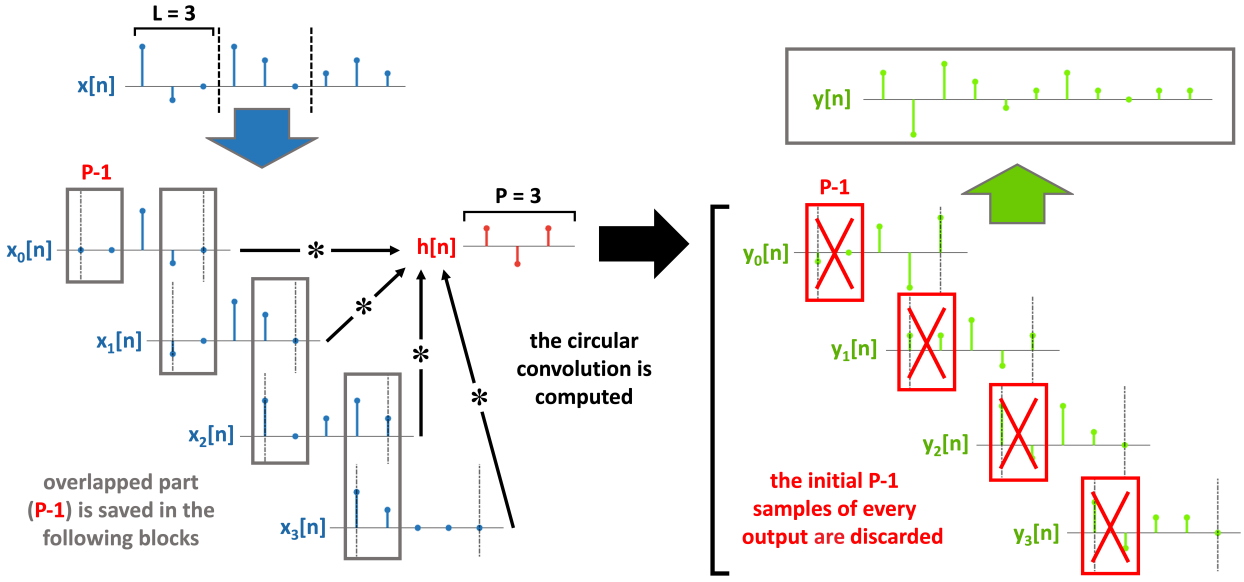


Figure 1: An illustration of using the overlap-save method for block convolution of an input signal x[n] with an FIR h[n] to obtain the overall result y[n]. (Note: The latency caused by the computational time of each convolution process is not shown in this figure.)

B. PING-PONG BUFFER

Ring buffers are a common data structure used in computer science and help in simplifying data management, optimizing latency, and reducing the number of possible operations per second.[5] They perform like a queue with a first in, first out (FIFO) process. Ping-pong buffers are memory blocks that operate using pointers to the current position within the buffer. These pointers keep track of where the processor is so that data can be simultaneously read and written while the RTCS runs.[6] Dynamic memory access routines allow peripherals to shuttle data into and out of these memory blocks.

Using the ping-pong buffer, speech data can be recorded and written into one block while the previously recorded block of speech can be read from the other block and used for convolution. The length of the input blocks stored must be chosen carefully so that the program has enough time to do the convolution before data is overwritten. If, for example, the convolution of a certain data block size (in time) took more computational time than the time needed to record the block size itself, multiple blocks would be needed to prevent corruption of data. For this project, the input block size was minimized to both reduce latency and to avoid the need for a large ring buffer.

3. IMPLEMENTATION

Although there are a variety of platforms (software and hardware) for implementing an RTCS, few can run the RTCS with minimal/zero latency. The sections below highlight the efforts made to minimize latency

12 October 2026 23:32:43

by researching and testing different software and hardware platforms in which to implement the RTCS. Through the guidance of multiple electrical engineers and other scientists, the decision was made for the implementation of the current RTCS as described in Section 3.A.iii.

A. CODE IMPLEMENTATION

i. MATLAB

Since the 1990s, the software system MATLAB has been commonly used for digital signal processing (DSP) projects.[7] What makes MATLAB ideal for DSP is its interactive environment, extensive libraries, and high-speed calculations. In addition, since MathWorks (the company behind MATLAB) offers educational licenses, MATLAB is easily accessible to universities and their students. At BYU, MATLAB is the main software program used for computational work in acoustics, as well as in a few electrical engineering courses. Because of this, the first version of the RTCS was implemented in MATLAB.

To perform the convolution of two signals in MATLAB, the '*conv*' function can be used with the signals as the function's input arguments. This function uses the direct method of computing the convolution, in the time domain, and can be computationally expensive for large signals. For real-time processing, the '*filter*' function can be used to compute the convolution efficiently, in the frequency domain, using digital filtering methods. The most efficient MATLAB function for computing the convolution of two long signals is '*fftfilt*'; this function performs the frequency domain convolution with block processing to leverage the efficiency of the FFT.

The MATLAB version of the RTCS was implemented using a ping-pong buffer, and all three functions mentioned above were used and tested. The minimum latency achieved was around 270 ms; this latency was attributed to an inherent delay in MATLAB when recording and immediately playing audio. Some potential sources of the inherent delay include the audio buffering (overflows/underflows), audio driver latency, and the operating system and hardware used. To combat this inherent delay, the MATLAB RTCS was also converted to a standard executable file (EXE), a MATLAB executable file (MEX), and a MATLAB audio plugin; all of these other implementations were expected to run with lower latency than a MATLAB script but resulted in the same measured latency.

ii. Field-Programmable Gate Array

To solve the problem of inherent delays, an RTCS implemented on a field-programmable gate array (FPGA) was considered. An FPGA is a type of configurable integrated circuit that can be reprogrammed after manufacturing to implement a large variety of systems. They consist of an array of programmable logic gates that can be configured to interconnect with other gates and perform various digital functions. These devices are known for their inherent parallel architecture, high clock speed, and, therefore, fast input/output (I/O) rates, which make them popular for implementing real-time systems.[8–10] To control the behavior of an FPGA, a user must implement a design in a hardware description language (HDL) like VHDL and Verilog.

In MATLAB and Simulink, an algorithm can be converted to HDL code and implemented in hardware by using MATLAB's HDL Coder package. As long as the algorithm is written with syntax and functions that are compatible with HDL code generation, its equivalent in an HDL can be created. Unfortunately, none of the convolution functions (i.e. '*filter*' or '*filtfilt*') used in the MATLAB version of the RTCS algorithm were compatible with the HDL code generator. Therefore, in order to implement an RTCS on an FPGA, a new algorithm would have to be written using functions compatible with the generator; these functions can be found in MATLAB's DSP Toolbox. We were advised that the learning curve for the HDL code and FPGA architecture was too steep for this RTCS application. However, FPGAs should be a viable option for future implementations of real-time systems like this.

12 October 2026 23:32:43

iii. STMicroelectronics Microcontroller

In between the efficiency and power of MATLAB and FPGAs are microcontrollers. The STM32F746 Discovery Board specifically is powered by the ARM Cortex-M7 core, which provides high processing power for a wide range of applications. The board includes a 4.3-inch capacitive touchscreen thin film transistor liquid-crystal display and extensive connectivity options to ease user interaction and communication. Using STM32 software development tools like STM32CubeIDE, the board is most commonly programmed in C or C++. Due to the ideal efficiency and power of the STM32F746, as well as its user-friendly architecture, the current version of the RTCS is implemented on these boards. Figure 2 shows the front and back of an STM32F746 Discovery Board used in the RTCS.

In the senior-level, Introduction to DSP course taught by BYU's Electrical and Computer Engineering Department, these discovery boards are programmed by students to solve problems in many different contexts such as radar, sonar, and communications. Using preexisting code, written by author Brian Mazzeo, that employs a ping-pong buffer to perform functions in close-to-real time, the RTCS was implemented on the STM32F746 Discovery Board. The system utilizes the board's line I/O ports to send a speech signal into the board for convolution and then send the auralization back out. To maximize the speed of the RTCS, a total of six boards (three for each channel and respective OBRIR) are used to perform the convolution of the left and right OBRIRs with input speech. The reasoning for using three boards per OBRIR convolution is explained in Section 4.B.

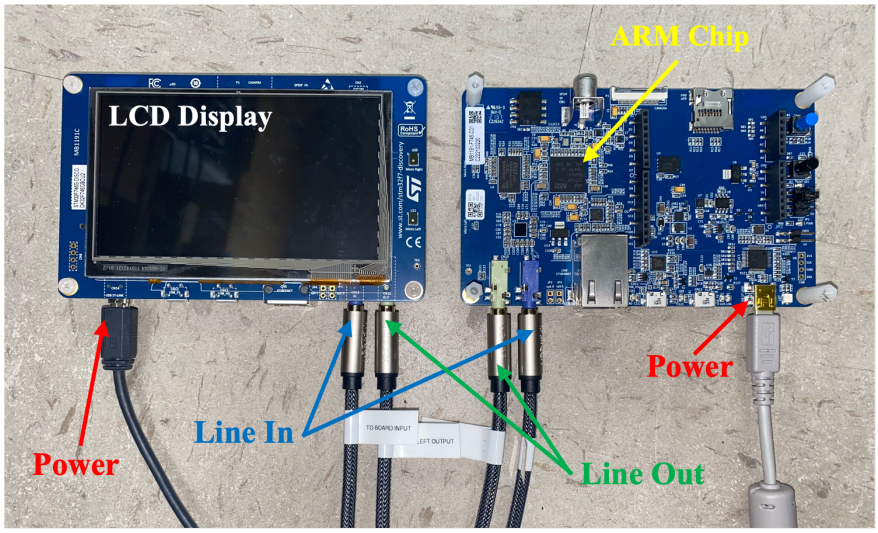


Figure 2: The front (left) and back (right) of an STM32F746 Discovery Board. The main components of the board are indicated with arrows.

B. SYSTEM HARDWARE

Many pieces of hardware are used in conjunction with the RTCS implemented on the STM32F746 boards. A diagram of the whole system's process is shown in Fig. 3, and the real setup is pictured in Fig. 4. Speech from a talker is recorded using a DPA 4066 headset microphone, which is provided 48V phantom power and preamplification by a Focusrite Scarlett 18i20 (2nd Generation) before being sent into the STM32F746 boards. After the real-time convolution processing, the resulting output signals from the STM32F746 boards are correctly summed and amplified by a second Focusrite (acting as a mixer). By routing the output signals directly to the Focusrite's headphone outputs (in both instances), the resulting signals can be output with "ultra-low latency". From the second Focusrite's headphone outputs, the signals are

12 October 2026 23:32:43

played through off-ear AKG K1000 headphones; these headphones were chosen to allow for the uninhibited sound transmission of the direct sound from the talker's mouth to each ear. Figure 5 shows KEMAR, a G.R.A.S. head and torso simulator, wearing both the headset microphone and off-ear headphones used in the RTCS.

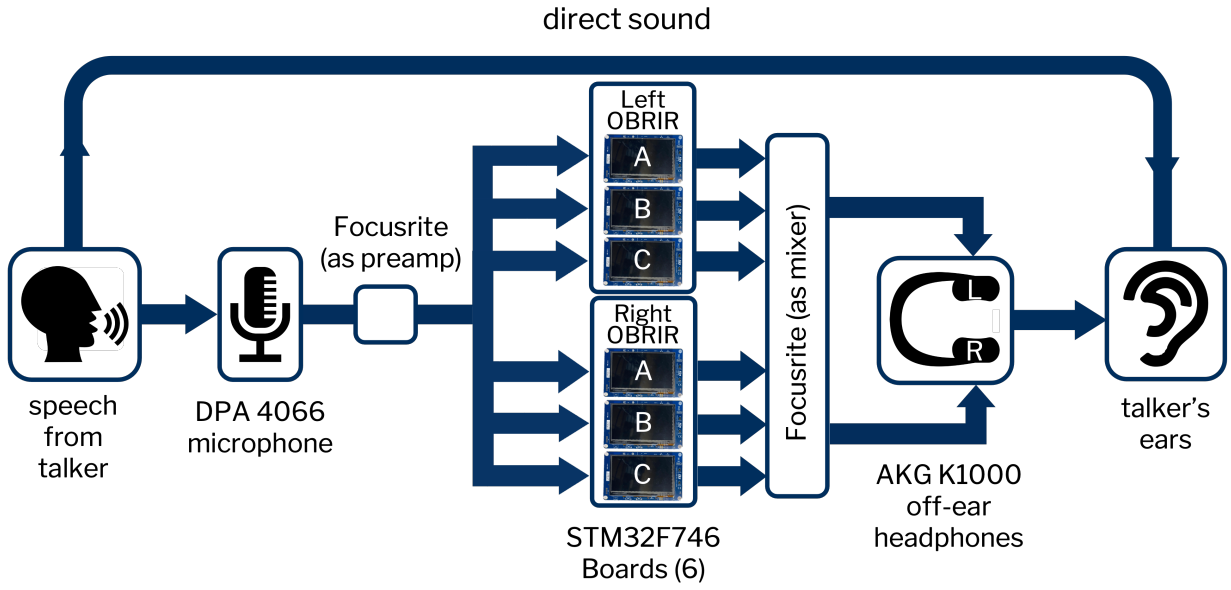


Figure 3: A diagram of the basic process and components of the whole RTCS. Different parts of the left and right OBRIRs are loaded onto each STM32F746 board.

12 October 2026 23:32:43

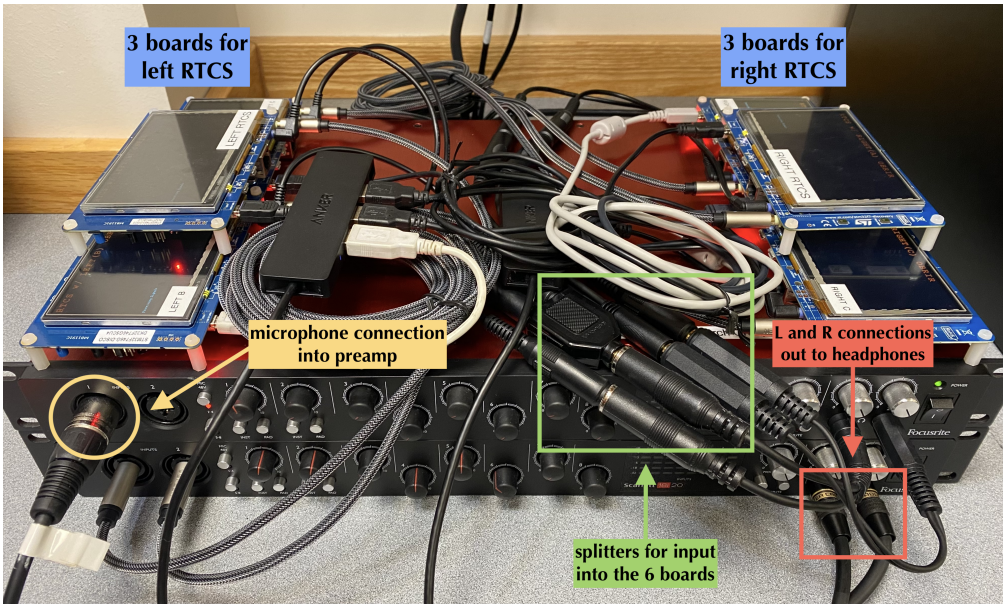


Figure 4: A picture of all the hardware involved in the RTCS which includes two Focusrite Scarletts (the top one acts as a preamp and the bottom one as a mixer) and six STM32F746 Discovery Boards (three for each left and right side).

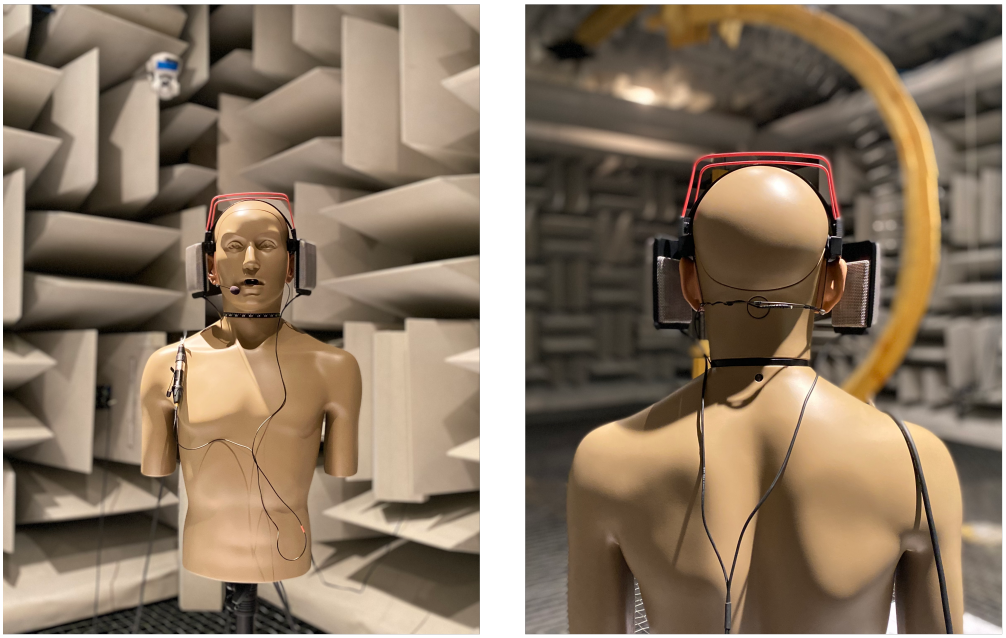


Figure 5: A G.R.A.S. head and torso simulator manikin (KEMAR 45BC) wearing AKG K1000 off-ear headphones and a DPA 4066 headset microphone used for the RTCS.

4. LATENCY

Total system latency (TSL) is defined as the time delay between the onset and the response of an event. In the context of a subject using the RTCS, TSL would be the time measured between the moment the

12 October 2026 23:32:43

microphone records a speech signal, to when the response signal is sent out to the headphones. Ideally, TSL should be minimized for any "real-time" system. However, a variety of unavoidable issues exist that increase TSL. In the case of the RTCS in Fig. 3, the minimum TSL is 3.6 ms. While a tiny fraction of this latency comes from passing analog signals through the Focusrites, the majority of this latency is due to the analog-to-digital and digital-to-analog conversion (ADC and DAC) on the STM32F746 boards. This portion of latency was measured by omitting the convolution computation on the boards and electrically looping audio data straight through the RTCS hardware; in other words, a chirp signal was input directly to the first Focusrite, this input was passed through the boards without any changes to the signal, output to the second Focusrite, and recorded after amplification. Custom in-house software named Easy Spectrum Time Reversal (ESTR), which was created by Kingsley *et al.*,[11] was used to both generate the chirp signal used and record the output signal after amplification. Using cross correlation (a substitute for deconvolution) of the chirp with the output signal to estimate the impulse response, the inherent TSL of 3.6 ms was determined.

Additional latency can be caused by the size of the buffer chosen to load in data to the ping-pong buffer. If a buffer size of x ms is chosen, it takes exactly x ms to record one block of speech data. This in turn adds an additional x ms to the inherent 3.6 ms latency resulting in a TSL of at least 3.6+x ms. As mentioned before, because convolution is a computationally expensive process it inevitably adds latency to any system. However, the processing time can be greatly reduced by choosing a shorter input signal and/or impulse response. The following subsections describe tests done to optimize the system's parameters, such as sampling frequency, input block size, and OBRIR length; these tests were done using one STM32F746 boards. After choosing the final values for these parameters, a method called partitioned convolution was used to finally achieve the right latency at output for this RTCS to function as a real-time system. To accomplish this with the limited hardware, more STM32F746 boards were added to help in the RTCS process.

12 October 2026 23:32:43

A. OPTIMIZING THE RTCS PARAMETERS

With block size kept constant, there are two system parameters that must be kept within a certain range in order to ensure a correct output signal from the system. These parameters are the OBRIR length and sampling frequency (f_s), which affect the time of the convolution computation. If these parameters are set to be outside their allowed range of values, the computational time for the convolution will exceed the block size causing increased latency and incorrect timing of the output signals. A longer OBRIR results in a more complex convolution computation and increases the computational time. Conversely, sampling rates that are too low result in slower data buffering for a fixed length OBRIR in time. Figure 6 is a plot of input block size versus latency of the RTCS when using a 100 ms long FIR for different f_s. It shows that the higher f_s is, the lower latency is for various speech block sizes. Additionally, the shorter the block length, the lower the latency.

By setting the sampling rate to 48 kHz and block size to 4 ms on a single STM32F746 board, the minimum TSL achieved is 8.7 ms; the maximum OBRIR length allowed with these two parameter values is a little longer than 100 ms. To allow for a longer OBRIR, a lower sampling rate was considered. Because there is not much critical speech information above 10 kHz, a sampling rate of 24 kHz was tested on an STM32F746 board. At this sampling rate, frequencies up to the 12 kHz Nyquist frequency would still be correctly sampled by the RTCS making this the ideal minimum sampling rate to use. At a sampling rate of 24 kHz and same block size of 4 ms, the minimum TSL achieved is 1 ms higher, but allows for a maximum OBRIR length that is twice as long (200 ms).

According to the ANSI S12.60-2002 standard for Classroom Acoustics, the recommended RT60 inside a classroom should not exceed 0.6 s. However, in a study by Bistafa and Bradley,[12] it was found that RT60s between 0.4 and 0.5 s are ideal for achieving 100% speech intelligibility in quiet classrooms. Furthermore, the Technical Committee on Architectural Acoustics of the Acoustical Society of America (ASA) stated that

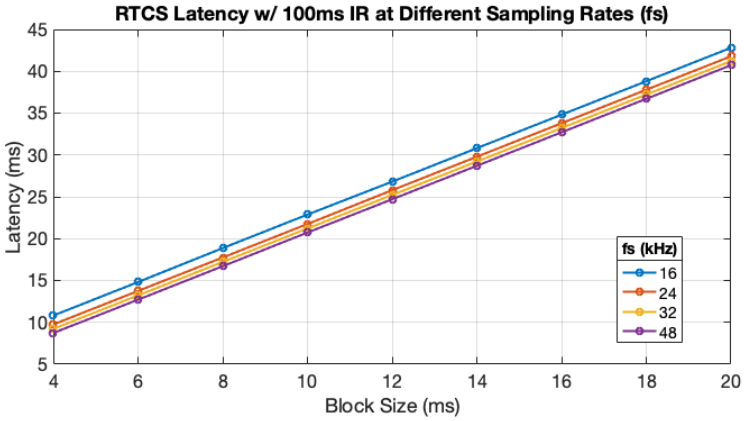


Figure 6: The measured RTCS latency for different input block sizes when using an IR of fixed length. The data shows that increasing the sampling rate decreases latency across all block sizes.

ideally classrooms should have RT60s in the range of 0.4-0.6 s.[13] Using these facts, Fig. 7 shows the decay of sound in classrooms in time for RT60s between 0.4 and 0.6 s. This plot illustrates that by using a 100 ms IR, only a maximum of 15 dB decay in initial sound levels can be achieved for a 0.4 s RT60. If a 200 ms IR is used, a 20 dB decay can be achieved for a 0.6 s RT60, a 25 dB decay for a 0.5 s RT60, and a 30 dB decay for a 0.4 s RT60; due to these higher decay levels achieved within the ideal RT60 range, the minimum IR/OBRIR length was chosen to be 200 ms.

Through these analyses, a sampling rate of 24 kHz and block size of 4 ms were ultimately chosen to allow for a 200 ms long OBRIR in the RTCS. The resulting TSL of the RTCS programmed on a single STM32F746 board with these parameter values is 9.75 ms.

A series of OBRIR measurements were made in elementary and secondary school classrooms.[14] During OBRIR measurements, KEMAR was kept at least 3 feet away from any large reflecting surface (walls, ceiling, floor, etc.). This stand off distance was to ensure that no reflections arrived at KEMAR's ear microphones earlier than 6 ms after the initial sound (using the approximation that sound travels around 1 ft/s). Because of this, the beginning of the OBRIRs were truncated by 6 ms without loss of important information; as a result of this, the OBRIR sections used would be the 200 ms from 6-206 ms of the original, full-length OBRIRs. Due to this truncation, the RTCS can only allow for 6 ms of TSL when using these OBRIRs. To achieve a TSL of 6 ms, the 200 ms long OBRIRs can be partitioned into nonuniform sections so that the convolution process can be parallelized. Correspondingly, parallelization achieved in the hardware by adding more STM32F746 boards to separately compute the convolution of each nonuniform sections.

B. NON-UNIFORM PARTITIONED CONVOLUTION

Non-uniform partitioned convolution is an efficient method for computing the real-time convolution of a long signal with a shorter signal by breaking the longer signal into smaller, manageable segments or partitions.[15] When the filter (in this case IR) is much longer than the input signal, the process of filter partitioning is used. Instead of performing a direct convolution that can be computationally intensive, the process involves dividing the IR into segments, convolving each segment separately with the input signal, and then combining the results. This approach reduces computational complexity and memory usage, making it particularly useful for real-time applications where quick processing is essential.

Because typical classroom IRs could have RT60s ranging from 0.3-0.7s, the length of IRs used in real-time room reverberation simulation systems like this could be long (a few seconds). Even if the input block size is chosen to be small (a few ms) and the FFT-based, overlap-save method is used to compute the

12 October 2026 23:32:43

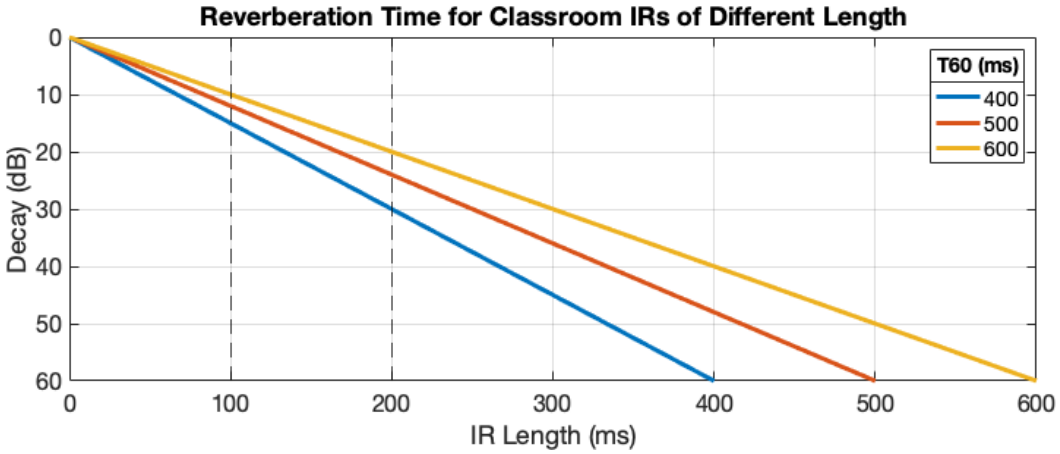


Figure 7: Decay of sound levels in time inside a classroom for different RT60 values. Shortening the IRs to 100 or 200 ms show that a maximum decay of 15 or 30 dB respectively can still be achieved.

convolutions, there will still be significant I/O delays due to the long IRs. In 1995, Gardner published a paper detailing the method of non-uniform partitioned convolution for low-latency filtering that is widely-known and used today.[16] The algorithm for non-uniform partitioned convolution used in this RTCS follows a similar approach to Gardner's. By first truncating the front of the IRs and then partitioning them into three separate IRs for convolution, the effects of the measured TSL (due to the convolution computations) can be eliminated at the system's output. Meaning, the RTCS can function as a real-time system to its users.

After accounting for the inherent TSL, only about 2.4 ms would be allowed for the convolution computation process; this was made possible by partitioning the OBRIR into three separate OBRIRs. As shown in Fig. 6, latency can be manipulated by changing the input block size. By making the first OBRIR be the first 1.7 ms of the original OBRIR and by changing the input block size to be 0.5 ms, a delay of 2.4 ms was imposed; adding the inherent delay (3.6 ms) to this results in a TSL of 6 ms to match the 6 ms truncation of the initial OBRIR. Then, since the second partitioned OBRIR would have to start 7.7 ms after the direct sound, it would need to have the appropriate length such that its convolution process would take 7.7 ms. The second OBRIR was chosen to be 3.3 ms long and the block size was kept at 0.5 ms to achieve this 7.7 ms delay. Lastly, the third OBRIR was set to be the remaining 195 ms, which started at 11 ms after the direct sound. Because the length of this third OBRIR was fixed, the block size was changed until a delay of 11 ms was achieved; this happened when the block size was set to be 3.8 ms. Figure 8 shows this partitioning visually on a fake OBRIR created. Since the first two OBRIRs are of shorter duration and the block size is also short, direct convolution was used in their computations. This was due to the fact that the overlap-save method is only efficient for larger block sizes. When the convolutions of these three partitioned OBRIRs are done in parallel and their results are concatenated in time, the overall result will output 6 ms after the direct sound. Therefore, by truncating and partitioning the OBRIRs in this specific manner, the inherent TSL and processing times do not delay the output of the RTCS. By using this method of partitioned convolution, this RTCS (using the specific OBRIRs measured) can function as a real-time system to its users and output responses 6 ms after the direct sound.

5. STM32F746 BOARD LIMITATIONS

After optimizing the RTCS parameters and using multiple STM32F746 boards to achieve a truly real-time system, the gains on each piece of hardware used were adjusted. This step was essential to ensure that the output levels of the RTCS corresponded correctly with the input speech levels of the user. The

12 October 2026 23:32:43

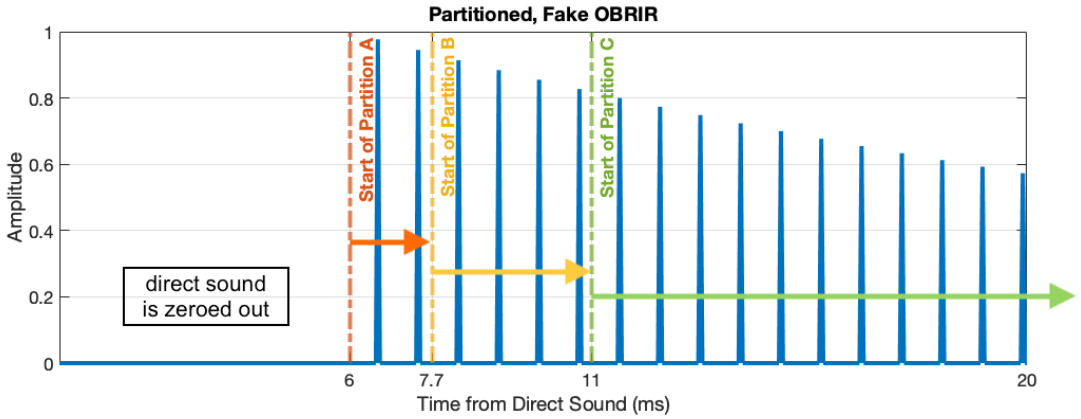


Figure 8: A plot of a fake oral binaural room impulse response (OBRIR) partitioned the same way as a real OBRIR that would be used in the RTCS. (Notice only the first 20 ms of an OBRIR are shown in this plot; in reality, the 6-206 ms portion of a real OBRIR is used in the RTCS.

different gains include that on the microphone preamp (to improve the signal-to-noise ratio (SNR)), the STM32F746 board's input (to avoid clipping) and output volume levels, and the headphone amplifier output level (to have correct output levels to the user's ears). To adjust all these gains to result in the correct input speech level to RTCS output level ratio, a head and torso simulator mentioned in Section 3.B, KEMAR, was used. Equipped with a precision microphone in each ear and a built-in loudspeaker at the mouth, KEMAR can measure the levels heard by a talker using the RTCS. Wearing the RTCS headphones and headset microphone, KEMAR was used to essentially remeasure the OBRIRs using feedback from the RTCS (with the real OBRIRs loaded in) rather than feedback from being placed in the physical classrooms. The OBRIRs were measured and calculated from KEMAR using a software program called EASERA (Electronic and Acoustic System Evaluation and Response Analysis).[17] The remeasured OBRIRs were then compared to the real OBRIRs, to ensure that the amplitude of the reflections (produced by the RTCS) had the same relative amplitudes as in the real OBRIRs with respect to the direct sound. To reach the correct output levels for both the left and right OBRIRs, the gains were adjusted and OBRIRs remeasured until they matched the amplitudes of the real OBRIRs.

After completing this step in getting the RTCS ready to use for vocal tests, background noise and perceptible buzzing that can be heard in the output speech was noticed. To try and combat these issues, a variety of tests were done. Each board was individually tested to make sure all produced the same output when given the same input and OBRIR. Although the same background noise and buzzing occurred in the output of each individual board, these issues worsened when multiple board outputs were added/mixed together. Regular room IRs were loaded into the boards, but had the same noise output; this was to test whether the OBRIRs were the source of the noise. Artificial OBRIRs were created using bandpass-filtered Delta functions and used in the system, but although the noise levels could be decreased dramatically, a robotic buzzing in the output speech remained. These artificial OBRIRs were created to have the same envelope as some real OBRIRs measured, but less reflection impulses were included to test whether multiple and densely-packed impulses were the problem. The background noise was found not to depend heavily on whether the OBRIRs were measured data or were artificially created, nor did the type of OBRIRs used change the buzzing quality of the output speech. Another possible noise source tested was whether the OBRIR peak amplitude values were too low, which was causing low SNR at output. The real OBRIRs contained reflections less than a tenth of the amplitude of the direct sound, so the OBRIRs were normalized and multiplied by larger numbers (from 1-10) to try and increase the SNR while performing convolution computations on the boards. However, doing this proportionately increased the noise levels at output. It was found that the left and right

12 October 2026 23:32:43

OBRIRs needed to be scaled correctly relative to each other, using the same normalization factor for both in order for their outputs to have the correct levels relative to each other (the use of independent normalization factors led to a noticeable increase in the background noise in the right ear compared to that in the left ear).

Issues remain in the current system and are currently being worked on. It may be that the lengths of input signals were not optimally chosen and that other settings were not optimized with these boards. We hope that current research efforts will overcome these issues.

6. CONCLUSION

The overall goal of this research is to measure the effects of classroom acoustics on the vocal strain of teachers. To reduce variables introduced by moving subjects from room to room, a real-time convolution system (RTCS) was developed to create binaural auralizations from any measured acoustic environment in real time. Unlike other existing systems, this RTCS was developed without the use of commercial plugins to perform convolution in real time, and thus the whole process can be clearly reported on and controlled. Due to the fact that the system must function in real time, system latency was prioritized and ultimately minimized by taking the measures explained in this paper. The algorithm performing convolution in real time was optimized by using an Fast Fourier Transform (FFT) based, overlap-save method for convolution along with filter partitioning. By using these methods, maximum speed and minimal error could be ensured during the data buffering process and convolution computation. System parameters that affected total system latency (TSL) the most were identified and optimized to make this RTCS function as a real-time system.

The RTCS was implemented on a total of six high-speed and high-efficiency microcontrollers developed by STMicroelectronics. Due to the limitations of these boards and primarily the buzzing quality of the speech that was produced by them, the complete RTCS was not ready to be used for subject vocal strain testing. Perhaps with a deeper dive into the architecture of these STM32F746 boards, the sources of these issues could potentially be found and fixed to make this RTCS still a viable tool for vocal strain testing in the future.

12 October 2026 23:32:43

ACKNOWLEDGMENTS

The authors gratefully acknowledge funding for this research from the National Institute of Health (NIH) and support from Brigham Young University (BYU). The authors also acknowledge Dr. Brian Jeffs of BYU's Electrical and Computer Engineering Department who played an important role in the development of the RTCS's architecture.

REFERENCES

[1] E. Smith, J. Lemke, M. Taylor, H. L. Kirchner, and H. Hoffman, "Frequency of voice problems among teachers and other occupations," J. Voice **12**(4), 480–488 (1998).

[2] J. Kristiansen, S. P. Lund, R. Persson, H. Shibuya, P. M. Nielsen, and M. Scholz, "A study of classroom acoustics and school teachers' noise exposure, voice load and speaking time during teaching, and the effects on vocal and mental fatigue development," Int. Arch. Occup. Environ. Health **87**, 851–860 (2014).

[3] R. Agarwal and J. Cooley, "New algorithms for digital convolution," IEEE Transactions on Acoustics, Speech, and Signal Processing **25**(5), 392–410 (1977).

[4] A. V. Oppenheim, *Discrete-time signal processing* (Pearson Education India, 1999).

[5] M. Orlikowski, "Single producer–multiple consumers ring buffer data distribution system with memory management," in *Trends and Innovations in Information Systems and Technologies*, Springer (2020), Vol. 1160, pp. 3–13.

[6] M. Malka, N. Amit, M. Ben-Yehuda, and D. Tsafrir, "riommu: Efficient iommu for i/o devices that employ ring buffers," ACM SIGPLAN Notices **50**(4), 355–368 (2015).

[7] J. H. McClellen, A. V. Oppenheim, and R. W. Schafer, *Computer-Based exercises for signal processing using MATLAB 5* (Prentice Hall PTR, 1997).

[8] Y. Chen and V. Dinavahi, "Fpga-based real-time emtp," IEEE Transactions on Power Delivery **24**(2), 892–902 (2008).

[9] I. S. Uzun, A. Amira, and A. Bouridane, "Fpga implementations of fast fourier transforms for real-time signal and image processing," IEE Proceedings-Vision, Image and Signal Processing **152**(3), 283–296 (2005).

[10] M. Psarakis, A. Dounis, A. Almabrok, S. Stavrinidis, and G. Gkekas, "An fpga-based accelerated optimization algorithm for real-time applications," Journal of Signal Processing Systems **92**(10), 1155–1176 (2020).

[11] A. D. Kingsley, J. M. Clift, B. E. Anderson, J. E. Ellsworth, T. J. Ulrich, and P.-Y. L. Bas, "Development of software for performing acoustic time reversal with multiple inputs and outputs," in *Proceedings of Meetings on Acoustics*, AIP Publishing (2022), Vol. 46.

[12] S. R. Bistafa and J. S. Bradley, "Reverberation time and maximum background-noise level for classrooms from a comparative study of speech intelligibility metrics," J. Acoust. Soc. Am. **107**(2), 861–875 (2000).

[13] B. Seep, R. Glosemeyer, E. Hulce, M. Linn, and P. Aytar, "Classroom acoustics: A resource for creating learning environments with desirable listening conditions" (2003).

[14] M. Robertson, "Survey of acoustic conditions of elementary and secondary schools in utah county and voice amplification systems," (2023).

[15] F. Wefers, *Partitioned convolution algorithms for real-time auralization*, Vol. 20 (Logos Verlag Berlin GmbH, 2015), pp. 68–71.

[16] W. G. Gardner, "Efficient convolution without input-output delay," J. Audio Eng. Soc. **43**, 127–136 (1995).

[17] W. Ahnert, S. Feistel, and W. Richert, "Merging room-acoustic and electro-acoustic measurement methods," in *Audio Engineering Society Convention 116*, Audio Engineering Society (2004).

12 October 2026 23:32:43