# Music 422 Project Report

Jatin Chowdhury, Arda Sahiner, and Abhipray Sahoo

October 10, 2020

## 1 Introduction

We present an audio coder that seeks to improve upon traditional coding methods by implementing block switching, spectral band replication, and gain-shape quantization. Block switching improves on coding of transient sounds. Spectral band replication exploits low sensitivity of human hearing towards high frequencies. It fills in any missing coded high frequency content with information from the low frequencies. We chose to do gain-shape quantization in order to explicitly code the energy of each band, preserving the perceptually important spectral envelope more accurately.

## 2 Implementation

### 2.1 Block Switching

Block switching, as introduced by Edler [1], allows the audio coder to adaptively change the size of the block being coded. Modern audio coders typically encode frequency lines obtained using the Modified Discrete Cosine Transform (MDCT). Using longer blocks for the MDCT allows for better spectral resolution, while shorter blocks have better time resolution, due to the time-frequency trade-off known as the Fourier uncertainty principle [2]. Due to their poor time resolution, using long MDCT blocks for an audio coder can result in "pre-echo" when the coder encodes transient sounds. Block switching allows the coder to use long MDCT blocks for normal signals being encoded and use shorter blocks for transient parts of the signal.

In our coder, we implement the block switching algorithm introduced in [1]. This algorithm consists of four types of block windows: long, short, start, and stop. Long and short block windows are simple sine windows, as defined in [3]:

$$w[n] = \sin\left(\left(n + \frac{1}{2}\right)\frac{\pi}{N}\right) \tag{1}$$

where $N$ is the length of the block. In our coder, we use $N_{long} = 2048$ and $N_{short} = 256$. The start window is defined as follows:

$$w[n] = \begin{cases} \sin\left(\left(n+\frac{1}{2}\right)\frac{\pi}{N_{long}}\right) & n < \frac{N_{long}}{2} \\ 1 & \frac{N_{long}}{2} \le n < \frac{3N_{long}}{4} - \frac{N_{short}}{4} \\ \sin\left(\left(n - \left(\frac{3N_{long}}{4} - \frac{N_{short}}{4}\right) + \frac{1}{2}\right)\frac{\pi}{N_{short}}\right) & \frac{3N_{long}}{4} - \frac{N_{short}}{4} \le n < \frac{3N_{long}}{4} + \frac{N_{short}}{4} \\ 0 & n \ge \frac{3N_{long}}{4} + \frac{N_{short}}{4} \end{cases} \tag{2}$$

The stop window is the exact same as the start window, reversed in time. Figure 1 shows all of the windows used for the block switching algorithm. Our coder implements a full block length

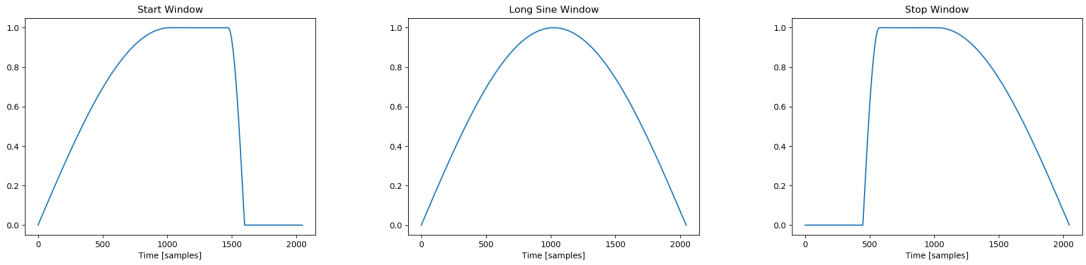


Figure 1: Start (left), sine (middle), and stop (right) windows.

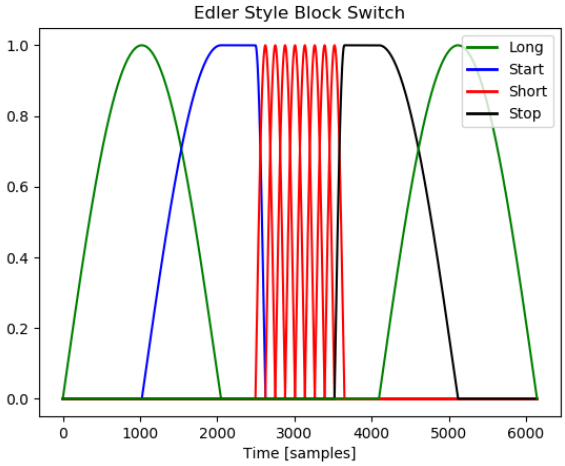


Figure 2: Full block switching cycle.

of lookahead to find transients. If the coder detects a transient in the next block, it processes the current block using a "start" window. The block with the transient is then processed using a series of short blocks, and the block after is processed using a "stop" window. All other blocks are processed using long blocks. One full cycle of block switching from long blocks, to short, and back to long is shown in fig. 2.

### 2.1.1 Transient Detection

In order to know when to use short blocks, our coder implements a transient detector to detect transients in blocks of the signal. The transient detector calculates a modified peak-to-average ratio (PAR) of the signal in the block and compares the PAR value to a threshold. If the PAR is above the threshold, the detector determines that there is a transient present in that block. An example of the transient detector's efficacy can be seen in fig. 3.

A typical peak to average ratio computes the maximum and average signal power over the entire block and divides the two values. Since our goal is primarily to eliminated pre-echo, instead of using the average power over the whole block, we compute the average of the signal power of the subset of the block that occurs before the maximum value.

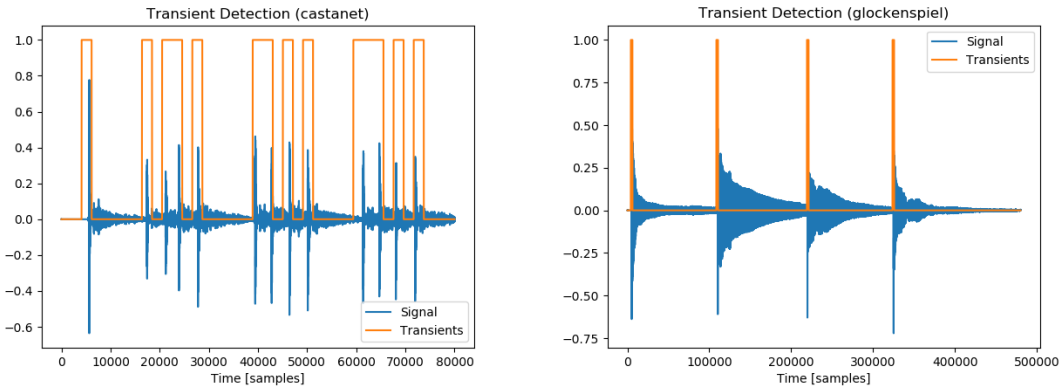


Figure 3: Example of the PAR transient detector detecting transients from sample recordings of castanets (left) and glockenspiel (right).

### 2.1.2 Results

The results of including block switching in the encoder can be seen in fig. 4. Note that the pre-echo in the coded signal lasts for $\sim$ 1000 samples, while in the block switched coded signal, the pre-echo only lasts for $\sim$ 100 samples, which is expected based on the lengths of the long and short blocks used. The main drawback of the current block switching is that the transient detector is not quite perfect: on highly transient signals, it may miss a couple of transients, and on signals without transients but with large amplitude swings (e.g. vibrato), it sometimes finds false positives.

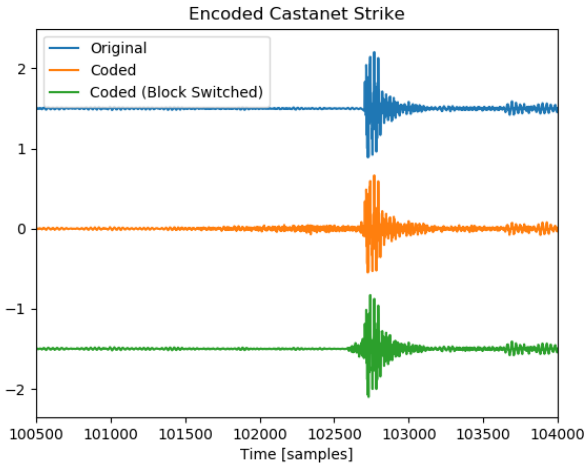


Figure 4: Example of castanet strike encoded with and without block switching. Note the lengths of the pre-echoes in the two coded files.

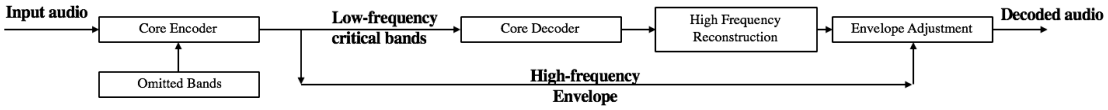


Figure 5: Block Diagram for Spectral Band Replication

## 2.2 Spectral Band Replication

### 2.2.1 Motivation

Spectral Band Replication (SBR) for audio coding was a much later innovation, most popularized by Dietz et al in 2002 [4]. The underlying principle which motivates spectral band replication is that at low bit-rates, the existing baseline coder will often not allocate any bits to the high-frequency components of the signal, because of higher signal-to-mask ratios (SMR) at lower frequency bands which take higher priority. As a result, the baseline coder is not satisfactory at a low bit rate, sounding as if it were a low-pass filtered version of the input. Therefore, in order to have an audio coder which is practical to use at low bit rates, as is often demanded by many applications, it is important to have a method to represent high frequency components better than the baseline approach. Moreover, especially in harmonic signals, we know that higher frequencies are often roughly decayed versions of lower frequency components, representing higher-order harmonics of lower fundamental frequencies. With SBR, we hope to exploit this redundancy for a better sounding decoded signal at the same bit rate.

### 2.2.2 Implementation

SBR is an approach which allows us to coarsely reconstruct the high frequencies of the signal while requiring fewer bits in the compressed format than the baseline method of encoding each MDCT line. This is accomplished by:

1. In the encoder, transmitting only an estimate of the envelope of the signal at higher frequencies, at a coarser frequency interval than the usual MDCT line spacing.

2. In the decoder, transposing the lower frequency MDCT lines to the higher frequency portion, adjusting this transposition by the encoded envelope in order to prevent the higher frequencies from dominating the signal.

Practically speaking, this is implemented by first establishing the proportion of total MDCT lines which we hope to omit in the encoder and reconstruct in the decoder, $p_{SBR}$, which we for the purposes of simplicity keep fixed at a value of 1/2. Then, the critical bands corresponding to these MDCT lines to be omitted are stored in a coding parameter, which both the encoder and decoder will use. A block diagram of the procedure is shown in Figure 5, while in Figure 6 we show an idealized version of the transposition and envelope adjustment.

**Encoding**

Next, we build our spectral envelope transmission for these omitted bands on top of our existing framework for bit allocation and transmission, representing the envelope of the signal as the mean absolute value of the FFT over all frequency lines in each omitted critical band. Thus, instead of transmitting $nLines[iBand]$ values with only a few mantissa bits, we instead transmit one value with the optimal number of bits for that singular value, using our bit allocation subroutine. Naturally, the benefits of this method can only be seen at lower bit rates, when the high-frequency critical bands would almost always be allocated 0 bits. As a result, we only make this feature "active" when

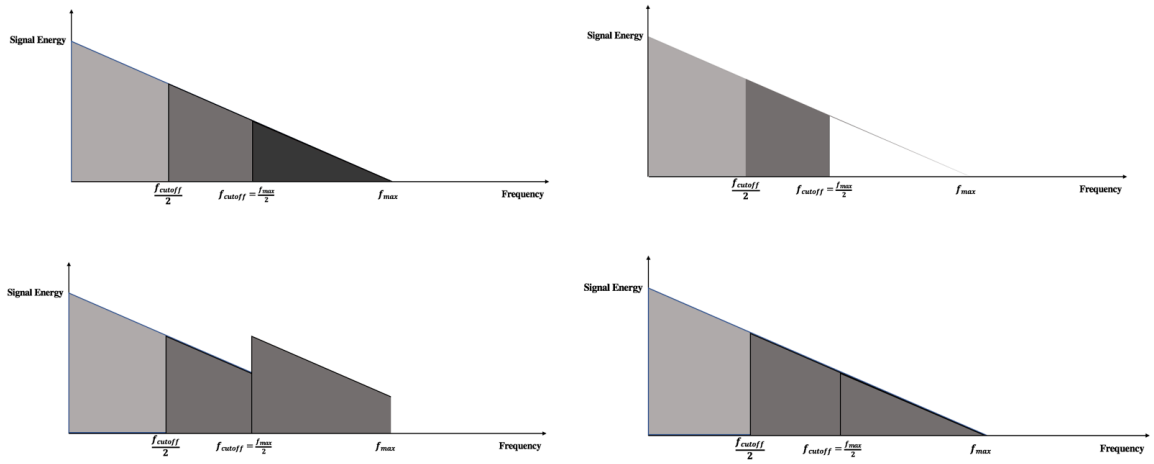


Figure 6: Ideal model of Spectral Band Replication. (Upper Left) Frequency representation of original signal. (Upper Right) Frequency representation of signal with higher frequencies omitted–what is transmitted by the encoder. (Lower Left) Frequency representation after transposing lower frequencies onto higher frequencies. (Lower Right) Frequency representation after envelope adjustment is made to higher frequencies.

our bit rate is under 128 kbps.

**Decoding**
Upon decoding, we first transpose the lower frequencies to their corresponding multiple of $1/p_{SBR}$ (i.e. 2) in the omitted higher frequencies. Because we use the MDCT to represent our frequency lines, we note that the frequency represented at MDCT line $k$ is not identically half of the frequency at $2k$, since the MDCT lines are shifted by 1/2 the line spacing relative to FFT frequency lines. To resolve this issue, we use a linear spline interpolation of lower frequency values to estimate the signal strength at frequencies exactly $p_{SBR}$ times each higher frequency, and transpose these interpolated values. Then, we smooth our received envelope with a Gaussian filter in order to prevent distinct jumps in signal energy, and adjust the amplitude of the transposed frequencies by this smoothed envelope.

### 2.2.3 Results

We find empirically that our implemented Spectral Band Replication approach significantly improves fidelity with respect to high-frequency components of coded audio, especially at lower bit rates such as 96 kbps. An example of this improvement is seen in Figure 7.

If we had more time, we could do more to improve this approach. In particular, one value to represent the envelope in each critical band is likely too coarse to represent the signal, especially where the signal values at higher frequencies is sparse, such as for harmonic signals. Instead, we can make our approach more general by transmitting $k$ such values per critical band to represent the envelope, where $k$ is a parameter to be tuned. This may allow for a more refined higher frequency representation of harmonic signal.

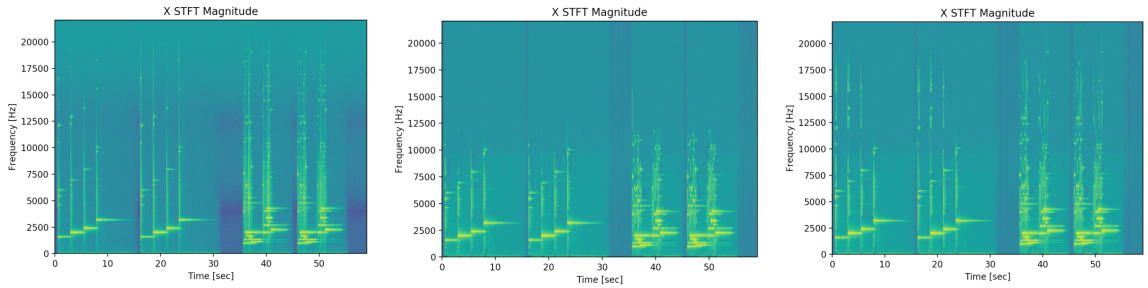


Figure 7: Example of SBR for glockenspiel signal. (Left) STFT of original, uncompressed glockenspiel 16-bit PCM version. (Middle) STFT of baseline coded version of glockenspiel signal at 96 kbps. (Right) STFT of SBR-coded version of glockenspiel signal at 96 kbps. We can see that the lower frequencies are identical to the baseline coder, while the higher frequencies are much better represented than the baseline coder. There is still room for improvement!

## 2.3 Gain-shape Quantization

A perceptually important characteristic of an audio signal is the spectral envelope. We use a gain-shape vector quantization scheme to encode each critical band of MDCT frequency lines. Given a vector of MDCT coefficients $x$, the gain-shape scheme explicitly encodes the "gain" $G$ or the energy in each band using scalar quantization and the "shape" $S$ or the energy-normalized unit vector using vector quantization.

$$G = ||x||_2 \tag{3}$$

$$S = \frac{x}{||x||_2} = \frac{x}{G} \tag{4}$$

### 2.3.1 Scalar quantization of the gain

The gain is normalized by the length of the vector, mu-law companded and then uniform-quantized. The mu-law companding function, shown below, is chosen to allocate more bits to small-level gain values.

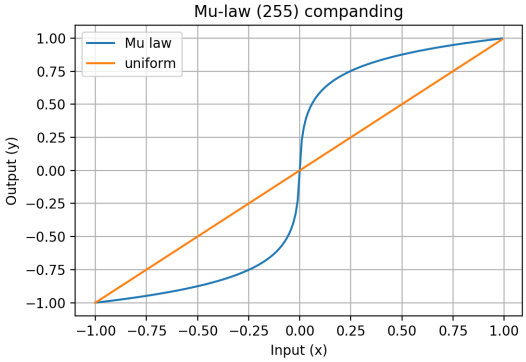


Figure 8: Mu-law companding function

### 2.3.2 Bit allocation for the gain and shape

For a given band of $L$ MDCT coefficients and bit allocation $b$, the number of bits allocated for the shape is given by. This is the same equation as used in CELT [5].

$$b_{\text{shape}} = \frac{b}{L} + 0.5 * \log_2(L) - k_{\text{fine}} \tag{5}$$

$$b_{\text{gain}} = b - b_{\text{shape}} \tag{6}$$

### 2.3.3 Pyramid vector quantization of the shape

The normalized shape vector $S$ lies on a hypersphere. Like CELT [5], we use an algebraic codebook called pyramid vector quantization; the codebook vectors are normalized to project them from the hyperpyramid onto the hypersphere. The codebook points are roughly regularly spaced on the hypersphere. This regularity allows for fast lookup to find the closest codebook point to an input vector, as well as for encoding into an integer index for transmission. The inverse operations to convert the encoded index back to the codebook point is also fast and memory-efficient. Moreover, we can choose the appropriately sized codebook depending on the bit-allocation for a band.

For contrast, a data-trained VQ scheme that supports variable bit-allocations would require a codebook for every possible bit-allocation as well as large number of codewords for large dimensional vectors.

A normalized PVQ codebook $P(L, K)$ consists of the set of integer vectors whose L1 norm is equal to K.

$$P(L, K) = \left\{ \frac{x}{||x||_2} : x \in \{Z^n : \sum |x_i| = K\} \right\} \tag{7}$$

The codebook size $N(L, K)$ obeys the recurrence $N(L, K) = N(L-1, K) + N(L-1, K-1) + N(L, K-1)$. For a given band of MDCT coefficients of known length $L$ and bit allocation $R$, we compute the K that meets the condition $\log_2 N(L, K) \leq R$. We then project the shape vector $S$ to the closest codebook point using the algorithm described in [6]. This gives us an integer index that we store. The decoder reconstructs from the integer index the codebook vector again following the decoding algorithm in [6].

As an example, the figure below shows the PVQ codebook vectors as round orange points. After normalizing by gain, we end up with the blue points. An input marked by the green cross near (1,0) is projected to the pyramid point near (2,0). The index corresponding to (2,0) is then transmitted/stored.

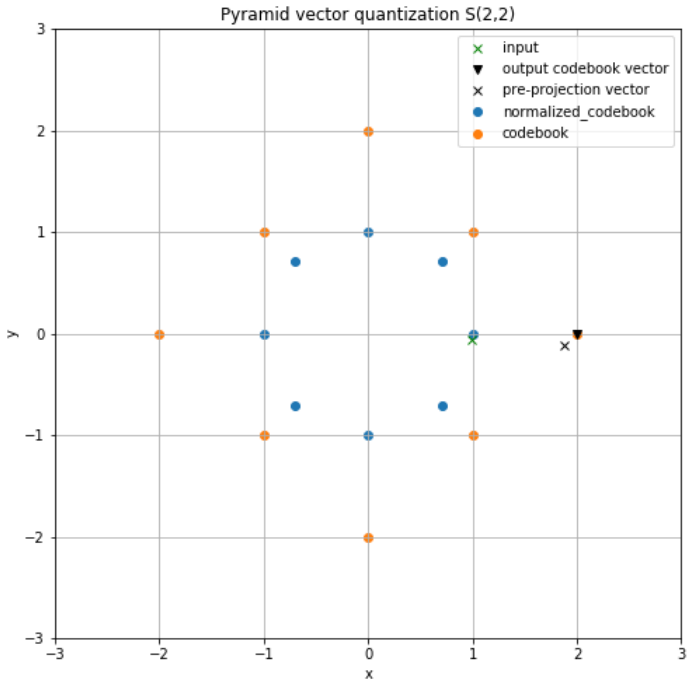


Figure 9: Example of pyramid vector quantization using codebook with L=2, K=2

### 2.3.4 Recursive band splitting

For bands that are allocated more than 30 bits, we divide the band into two subbands. If one of the two subbands still has more than 30 bits allocated, we recursively split it. This is done to avoid overflow issues with index computation in the PVQ step; it can also take a longer time to compute the index for large number of bits. The two halves are then jointly encoded using a mid-stereo encoding scheme. If vector $x$ is split into left and right halves $x_l$ and $x_r$, then the mid vector $M$ and $S$ vectors are calculated as follows:

$$M = \frac{(x_l + x_r)}{2} \tag{8}$$

$$S = \frac{(x_l - x_r)}{2} \tag{9}$$

We normalize $M$ to $m = M/||M||$ and $S$ to $s = S/||S||$. The energy distribution between $M$ and $S$ is encoded in a $\theta$ variable:

$$\theta = \arctan \frac{||S||}{||M||}$$

.

For quantization, the number of bits allocated to $\theta$ is calculated using equations (6) and $\theta$ taking the role of shape. $m$ and $s$ are calculated using equation 11 of [5]:

$$a_m = 0.5 * (a - (L-1)\log_2 \tan\theta) \tag{10}$$

$$a_s = a - a_m \tag{11}$$

$\theta$ is uniformly quantized. $m$ and $s$ are quantized using PVQ.

#### 2.3.5 Vector Quantization results

At low bitrate of 96kbps, critical material like castanet sounds much better than our baseline. However, there appear to be some artifacts introduced in material like the quartet singing. They appear to be caused by bands in the higher frequencies "collapsing" if few bits are allocated to large bands.

# 3 Results

We test our coder using five audio samples from the EBU Sound Quality Assessment Material (SQAM)[1], castanets, glockenspiel, harpsichord, vocal quarter, and male German speech. Subjectively, our coder performs about the same as the baseline coder at 128 kbps, and better for transient signals, as is to be expected given the implementation of block switching. At 96 kbps, our coder performs about the same as the baseline coder, particularly in preserving energy with gain-shape quantization and recreating the high frequency components of a signal, aided by the implementation of Spectral Band Replication and vector quantization.

## 3.1 Listening Tests

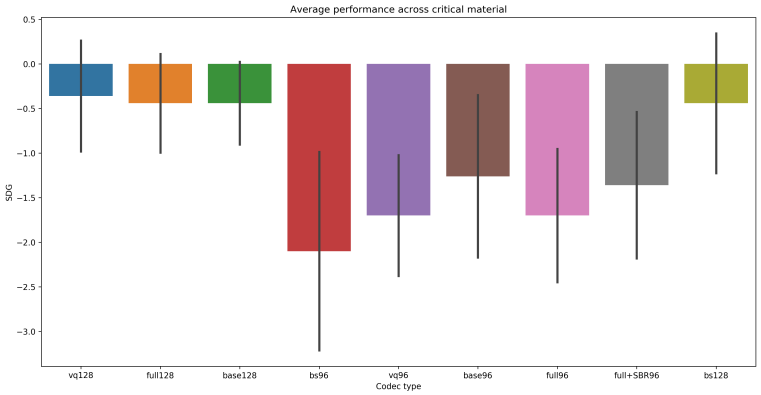


Figure 10: Average listening test results of various codecs on critical material

[1] https://tech.ebu.ch/publications/sqamcd

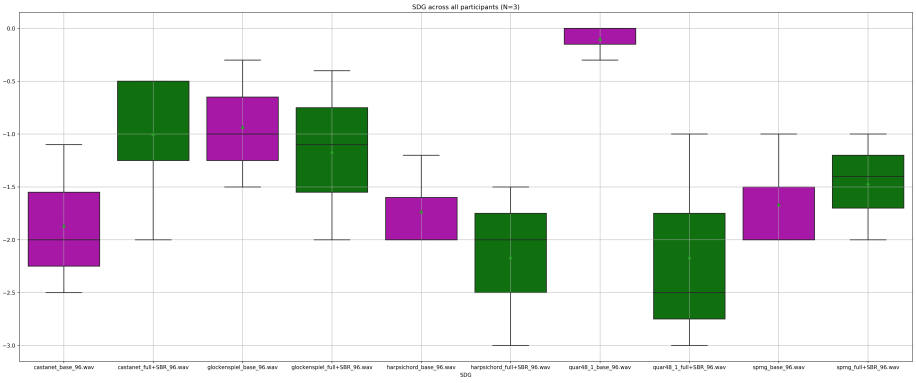


Figure 11: Piece by piece comparison of our method with the baseline coder at 96 kbps. Purple is the baseline coder, while green is ours.

From the plot in fig. 10, we can see that our final encoder performs equally as well as the baseline coder at a bit rate of 128 kbps. However, at 96 kbps bit rates, we see that if we just implement block switching, we perform much worse than the baseline. Adding vector quantization improves upon this, while adding spectral band replication further reduces the gap to make our final coder about identical to the baseline coder at 96 kbps. In fig. 11, we can see that for certain critical material, our coder performs much better, whereas for some it performs much worse than the baseline.

# 4 Conclusion

We present an audio coding format that uses spectral band replication, vector quantization, and block switching to improve upon the performance of traditional coding methods. However, we are unable to see much improvement in our current iteration. There are several areas of future work that could improve our existing coder. For block switching, having a more sophisticated transient detection algorithm could greatly improve performance; it also might be worthwhile to test against a Dolby AC-2 style block switching algorithm [7]. For spectral band replication, an improvement could be to use a finer resolution for the spectral envelope being transmitted (e.g. 4 values instead of 1 per critical band). For vector quantization, we could better handle edge cases of large band sizes and small bit allocations.

# References

[1] B. Edler, "Coding of audio signals with overlapping transform and adaptive window shape," *Frequenz*, vol. 43, no. 9, pp. 252–256, September 1989.

[2] Louis de Broglie, *On the Theory of Quanta*, Annales de la Fondation Louis de Broglie, Paris, France, 2004, English translation of doctoral thesis [**?**].

[3] Julius O. Smith, *Spectral Audio Signal Processing*, W3K Publishing, http://www.w3k.org/books/, 2011.

[4] Martin Dietz, Lars Liljeryd, Kristofer Kjorling, and Oliver Kunz, "Spectral band replication, a novel approach in audio coding," in *Audio Engineering Society Convention 112*. Audio Engineering Society, 2002.

[5] Jean-Marc Valin, Gregory Maxwell, Timothy B Terriberry, and Koen Vos, "High-quality, low-delay music coding in the opus codec," *arXiv preprint arXiv:1602.04845*, 2016.

[6] Thomas Fischer, "A pyramid vector quantizer," *IEEE transactions on information theory*, vol. 32, no. 4, pp. 568–583, 1986.

[7] Robert L. Andersen, Brett G. Crockett, Grant A. Davidson, Louis D. Davis, Mark F. andFielder, Stephen C. Turner, Mark S. Vinton, and Phillip A. Williams, "Introduction to dolby digital plus, an enhancement to the dolby digital coding system," in *Audio Engineering Society Convention 117*, Oct 2004.

[8] C. R. Helmrich and B. Edler, "Audio coding using overlap and kernel adaptation," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 590–594, May 2016.

[9] "jmvalin | recent entries," https://jmvalin.dreamwidth.org/, (Accessed on 03/12/2020).