Ring Modulation Decomposition and Rn Encoding

Caleb Adams

Abstract

Ring Modulation Decomposition and Rn Encoding details a proposed method of lossy data compression, ring modulation decomposition, and a generalization of said method, Rn encoding. The method proposed operates directly off of Fourier-related transforms and is intended as an extension of these transforms rather than a replacement. This paper showcases the current state of my work, and hopefully, the basis of my future thesis. Currently, I am unable to access the expensive references and scientific journals, so I elected to skip a section on previous work in the subject and go straight into my deductions using common knowledge in the field.

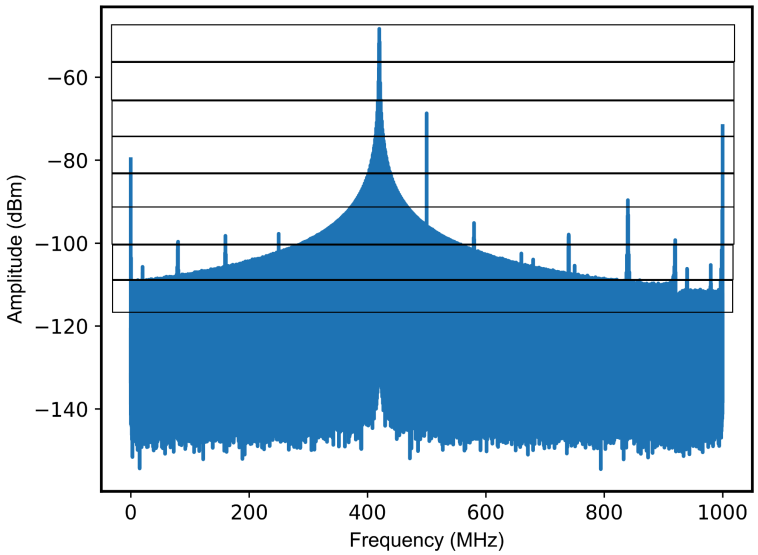
1 Introduction and Basic Concepts

(1.1) Digital audio data is defined as a vector of samples taken at a regular rate. A data compression algorithm is a pairing of two functions, an encoder, and a decoder, which is the inverse of the encoder. The encoding function's intention is to reduce the size of the input data, and the decoding function's intention is to undo the encoding as accurately as possible. The two overarching classes of compression algorithms are lossless and lossy. The difference between them is stated by the perfect or imperfect reversibility of the encoder function; if the decoder perfectly undoes the encoding operation, the algorithm is lossless, but if the decoder fails to perfectly undo the encoding operation, the algorithm is lossy.

A common method of audio data compression involves transform encoding, which uses integral transforms, many of which are based on the fourier transform. The Fourier transform takes a vector of samples in the time domain and transforms them into a vector of samples in the frequency domain. Formally, this process is the change of basis in a function space where each index of the fourier transformed vector corresponds with a basis function of the space.

Before introducing ring modulation, a change in perception must occur in the frequency domain. Consider drawing several horizontal lines on a frequency domain graph (see Figure 1).

Fig. 1



This generic example of a frequency domain plot is taken from a forum post on: https://dsp.stackexchange.com/

The space between one line and the next is an amplitude bin, and all amplitude values in the bin are considered as having the same amplitude as one another. As the bin size approaches 0, the assessment of amplitude similarity increases in accuracy, and it's logical to state that these horizontal lines are simply identifying frequencies of the same amplitude. Each amplitude bin is itself a vector containing the frequency values of the frequency domain data which are all of the same amplitude of the amplitude bin; another way to consider this transformation is as a quantization of the frequency domain data. All initial amplitude values are quantized to the nearest bin value, and each amplitude bin value is just a vector of the frequency values at said amplitude bin. This representation of

frequency domain data as a vector of amplitude bin vectors will be referred to as the amplitude bin representation.

(1.2) Ring Modulation is an operation where an audio signal, the carrier, is multiplied by a modulator signal. In the frequency domain, this operation results in two combination tones, the sum and the difference tone, which will be defined as:

Fig.2

$$RM_1(a,\ b) = a + b$$
$$RM_2(a,\ b) = |a - b|$$

with a being the frequency of the carrier signal and b being the frequency of the modulator signal. As an example, consider the amplitude bin vector $\begin{bmatrix} 400 \\ 1000 \end{bmatrix}$, using 200 as a modulator frequency and applying each RM operation to each index of the vector yields $\begin{bmatrix} 600 \\ 200 \\ 1200 \\ 800 \end{bmatrix}$; the full computation of this result is shown below.

Fig. 3

$$RM_1(400,\ 200) = 400 + 200 = 600$$
$$RM_2(400,\ 200) = |400 - 200| = 200$$
$$RM_1(1000,\ 200) = 1000 + 200 = 1200$$
$$RM_2(1000,\ 200) = |1000 - 200| = 800$$

$$RM\left(\begin{bmatrix} 400 \\ 1000 \end{bmatrix},\ 200\right) = \begin{bmatrix} 600 \\ 200 \\ 1200 \\ 800 \end{bmatrix}$$

The function $M(a)$ is an iterative function using repeated RM; it yields a vector of greater length than the input vector. Each iteration, $m(\boldsymbol{a}_{\mathbf{k}},\ a_{0,n})$, is defined as the result of the RM operation of the current scalar in the vector of initial inputs, $a_{0,n}$, with every scalar of the current vector of the iterative process, $\boldsymbol{a}_{\mathbf{k}}$. This itself yields the next vector of the process, $\boldsymbol{a}_{\mathbf{k}+1}$; the function is shown in its entirety below.

Fig. 4

$$m(\boldsymbol{a}_{\mathbf{k}},\ a_{0,n}) = iterate_{m=0}^{M-1}\left(RM_1(a_{k,m},\ a_{0,n}),\ RM_2(a_{k,m},\ a_{0,n})\right) = \boldsymbol{a}_{\mathbf{k}+1}$$

Now that $m(\boldsymbol{a}_{\mathbf{k}}, a_{0,n})$ has been defined, the full function of repeated RM operations, $M(a)$ is defined as:

Fig. 5

$$M(\boldsymbol{a}) = m_{N-1}(\ldots m_2(m_1(a_{0,0}, a_{0,1}), a_{0,2})\ldots)$$

Notice that $m_0 = \boldsymbol{a}_0 = a_{0,0}$, signifying that the initial input vector is only 1 scalar long.

For example, consider $\boldsymbol{a} = \begin{bmatrix} 4 \\ 8 \\ 13 \end{bmatrix}$, the expansion of which is in Figure 6.

Fig. 6

$$m_1 = 4+8,\ |4-8| = \boldsymbol{a}_1 = \begin{bmatrix} 12 \\ 4 \end{bmatrix}$$

$$m_2 = 12+13,\ |12-13|,\ 4+13,\ |4-13| = \boldsymbol{a}_2 = \begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}$$

$$M\left(\begin{bmatrix} 4 \\ 8 \\ 13 \end{bmatrix}\right) = \begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}$$

Following this, it can be seen that the relationship between the length of the input and output vector is given by:

Fig. 7

$$M(\boldsymbol{a})_{length} = 2^{a_{\text{length}}} - 1$$

2 RM Decomposition

(2.1) To define inverse repeated RM, consider now the inverse relationship implied by Figure 8.

Fig. 8

$$a_{length} = \log_2\left(M(a)_{Length} \right) - 1$$

In this scenario, $M(a)$, the repeated RM function, provides a useful method of expanding data, so it will become the decoder of the RM Decomposition algorithm, and it's notation will be changed to reflect this, becoming $R^{-1}(a)$. All that remains now, is to find the encoder function, or inverse repeated RM function, $R(a)$.

Assuming that $R(a)$ is sufficiently similar to $R^{-1}(a)$, it might be inferred that $R(a)$ consists of several recursive smaller functions, $r(a)$. The following figure shows the inputs and outputs of $R^{-1}(a)$, $r^{-1}(a)$, $R(a)$, $r(a)$ following the conditions in the figure.

$$a = \begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}, \quad b = \begin{bmatrix} 4 \\ 8 \\ 13 \end{bmatrix}$$

Fig. 9

$$R^{-1}(b) = a, \quad {r^{-1}}_1(4, 8) = \begin{bmatrix} 12 \\ 4 \end{bmatrix}, \quad {r^{-1}}_2\left(\begin{bmatrix} 12 \\ 4 \end{bmatrix}, 13 \right) = \begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}$$

$$R(a) = b, \quad r_1\left(\begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix} \right) = \left(\begin{bmatrix} 12 \\ 4 \end{bmatrix}, 13 \right), \quad r_2\left(\begin{bmatrix} 12 \\ 4 \end{bmatrix} \right) = (4, 8)$$

Using this approach, $r(a)$ yields a vector and a scalar. Knowing the solution to this process, the input and output can be abstracted to yield the following equations:

$$a = \begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}, \quad x = \begin{bmatrix} 12 \\ 4 \end{bmatrix}, \quad y = 13$$

Fig. 10

$$x_0 + y = 25 \Rightarrow y = 25 - x_0$$

$$\left| x_0 - y \right| = 1 \Rightarrow y = x_0 \pm 1$$

$$x_1 + y = 17 \Rightarrow y = 17 - x_1$$

$$\left| x_1 - y \right| = 9 \Rightarrow y = x_1 \pm 9$$

(2.2) Labeling these equations (1), (2), (3), (4) the solution to x_0, x_1 may be found by setting two of the four equations equal to each other, expressing one of the scalars solely in terms of the other scalar, then solving for said scalar using the available equations. The following figure shows this process for $(1) = (3)$.

$$(1)\ y = 25 - x_0,\ (3)\ y = 17 - x_1$$
$$25 - x_0 = 17 - x_1,\ x_0 = x_1 + 8 \Rightarrow x_1 = x_0 - 8$$

$$(2)\ \left|x_0 - y\right| = 1$$
$$\left|x_1 + 8 - \left(17 - x_1\right)\right| = 1$$
$$2x_1 - 9 = \pm 1$$
$$x_1 = 4,\ 5$$

Fig. 11

$$(4)\ \left|x_1 - y\right| = 9$$
$$\left|x_0 - 8 - \left(25 - x_0\right)\right| = 9$$
$$2x_0 - 33 = \pm 9$$
$$x_0 = 21,\ 12$$

Repeating this process for all 6 possibilities, the solutions for x_0, x_1 are shown below.

	x_0	x_1
$(1) = (2)$	12, 13	
$(1) = (3)$	12, 21	4, 5
$(1) = (4)$	12, 21	12, 13, 21, 22
$(2) = (3)$	3, 4, 5, 12, 13, 14	4, 5
$(2) = (4)$	3, 4, 5, 12, 13, 14	3, 4, 12, 13, 21, 22
$(3) = (4)$		4, 13

Fig. 12

The following figure shows the density of each solution.

	x_0	x_1
5 |	12	
4 |		4
3 |	13	13
2 |	3, 4, 5, 14, 21	5, 12, 21, 22
1 |		3

Fig. 13

Using the density plot, it is clear that the mode of each set of solutions is the correct solution for x_0, x_1. Given these solutions, it is straightforward to determine that $y = 13$. This method in **(2.2)** will be referred to as the mode of solutions method.

(2.3) To justify that this mode of solutions method works for all real values, the method of computation can be programmed and run a very large number of times for random real number vectors using the following test functions where $\boldsymbol{a}$ is some 3-dimensional vector consisting of real numbers and $\boldsymbol{b}$ is some 4-dimensional vector consisting of real numbers.

Fig. 14

$$\textit{Test } 1: \quad a = R\left(R^{-1}(a) \right)$$
$$\textit{Test } 2: \quad b = R^{-1}(R(b))$$

In written terms, the tests are as follows: a 3 dimensional vector, $\boldsymbol{a}$, can be expanded to 4 dimensions and reduced back to its original state, and a 4 dimensional vector, $\boldsymbol{b}$, can be reduced to 3 dimensions and expanded back to its original state. The code accompanying this paper runs both these tests, as "test1()" and "test2()", a large number of times to obtain justifiable assumptions about **Fig. 14**. Following these tests, it is observed that test 1 is true, but test 2 is false, failing to even apply the mode of solutions method. However, some careful rearranging shows another interesting development. $\boldsymbol{c}$ is the expansion of $\boldsymbol{a}$, $\boldsymbol{c} = R^{-1}(\boldsymbol{a})$, and $\boldsymbol{c}$ is a 4 dimensional vector. Where most random 4 dimensional vectors, $\boldsymbol{b}$, fail to apply the mode of solutions method $\boldsymbol{c}$ succeeds. This means that while all real number vectors may not apply RM Decomposition, a subset of all real numbers vectors do. Now, the goal must be to create a valid mapping between all real number vectors and the subset of valid real number vectors; this mapping from a continuous set to a discrete set resembles quantization, so I will label this process RMD Quantization (Q_{RM}).

To determine what this quantization process is, first, the properties of the subset of real numbers that successfully reduce must be determined. In **(2.1)** the 3d vector, $\begin{bmatrix} 4 \\ 8 \\ 13 \end{bmatrix}$, and its expansion $\begin{bmatrix} 25 \\ 1 \\ 17 \\ 9 \end{bmatrix}$ were considered; from the properties of RM expansion detailed in **(1.2)** and from the observed properties in the worked example before, it is clear that the expanded vector has the following property:

Fig. 15

$$a_0 + a_1 = a_2 + a_3$$
$$25 + 1 = 17 + 9 = 26$$

Therefore, to quantize any 4d vector to allow RMD to 3 dimensions, the frequency values of the vector must be approximated so this property becomes true. Since both sums should equal the same value, the equation in **Fig. 13** can be rewritten as shown in **Fig. 14,** and to make this notation more concise, c will be introduced as a combination of each scalar pair in $\boldsymbol{a}$.

Fig. 16

$$c_0 = a_0 + a_1$$
$$c_1 = a_2 + a_3$$
$$c_{error} = c_0 - c_1$$

The c_{error} gives the discrepancy between the sum of each scalar pair, and using it, a simple quantization algorithm where each scalar frequency is adjusted equally is obtained.

Fig. 17

$$\tilde{a}_0 = a_0 - \frac{c_{error}}{4}, \quad \tilde{a}_1 = a_1 - \frac{c_{error}}{4}$$
$$\tilde{a}_2 = a_2 + \frac{c_{error}}{4}, \quad \tilde{a}_3 = a_3 + \frac{c_{error}}{4}$$

When considering the audible change as frequency changes, it is worth remembering that in music theory, two times a specific frequency is the definition of an octave higher than said frequency. This definition is shown using scientific pitch notation and Hz in the following figure.

Fig. 18

$$A4 = 440\,Hz$$
$$A5 = 880\,Hz$$
$$2(A4) = A5 \Rightarrow 2(XN) = X(N+1)$$

Remembering this definition, a better quantizer can be obtained by changing higher frequencies more than lower frequencies. This results in a quantized version that is more audibly similar to the original signal than the equal change of each scalar presented in **Fig. 17**. This quantization is shown below:

Fig. 19

$$\tilde{a}_0 = a_0 - \frac{a_0}{\sum_{n=0}^{N-1} a_n}(c_{error}) \quad , \quad \tilde{a}_1 = a_1 - \frac{a_1}{\sum_{n=0}^{N-1} a_n}(c_{error})$$

$$\tilde{a}_2 = a_2 + \frac{a_2}{\sum_{n=0}^{N-1} a_n}(c_{error}) \quad , \quad \tilde{a}_3 = a_3 + \frac{a_3}{\sum_{n=0}^{N-1} a_n}(c_{error})$$

This formula and implementation works fine for a 4d to 3d reduction but it doesn't generalize to higher dimensions because of the previous definition of c_{error}. To improve this definition, first, consider an arbitrary 4d vector and its expansion shown below.

Fig. 20

$$a = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 7 \end{bmatrix}, \quad R^{-1}(a) = \begin{bmatrix} 12 \\ 2 \\ 8 \\ 6 \\ 11 \\ 3 \\ 9 \\ 5 \end{bmatrix}$$

Recall that c_{error} is the difference between the sum pairs of a vector, and a vector where $c_{error} = 0$, like the expanded vector shown above, contain sum pairs that are all equal to each other; in this case, each sum pair in the expanded vector is 14. Using the above vector in **Fig. 18,** the full generalized c_{error} is deduced below.

Fig. 21

$$c_0 = 12 + 2 = 14 \ , \ c_1 = 8 + 6 = 14$$

$$c_2 = 11 + 3 = 14 \ , \ c_3 = 9 + 5 = 14$$

$$\begin{aligned} c_{error,\,0} &= c_1 + c_2 + c_3 - c_0 - c_0 - c_0 \\ &= \left(c_1 + c_2 + c_3\right) - \left(c_0 + c_0 + c_0\right) \\ &= \left(c_0 + c_1 + c_2 + c_3\right) - \left(c_0 + c_0 + c_0 + c_0\right) \\ &= \left(c_0 + \ldots + c_{P-1}\right) - P\left(c_0\right) \\ &= \sum_{p=0}^{P-1} c_p - P\left(c_0\right) \end{aligned}$$

$$c_{error,\,s} = \sum_{p=0}^{P-1} c_p - P\left(c_s\right)$$

With this equation, the final generalized quantization process is given by the equation:

Fig. 22

$$\tilde{a}_n = a_n + \frac{a_n}{\sum_{k=0}^{K-1} a_k}\left(c_{error,\,s}\right)$$

With the quantizer, Q_{RM}, defined, a new test can be run using "rmdQuantizationTest()". As previously in test 2, vector b is some real valued 4-d vector.

Fig. 23

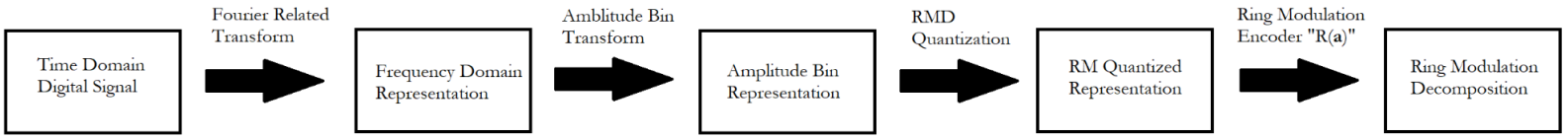
RMD Quantization Test:

$$\tilde{b} = Q_{RM}(b) \approx b$$

$$\tilde{b} = R^{-1}(R(\tilde{b}))$$

(2.4) Now that $R(\boldsymbol{a})$ is defined fully, the entire encoding process can be described fully from the time domain to the final vector of decomposed amplitude bin RM blocks. The block diagram of this is shown below.

Fig. 24



transform, such as the DFT, DCT, or DST. Then, the frequency domain signal is restructured as several amplitude bin vectors. Each one of these vectors is then reduced down to 2 scalars through Ring Modulation Decomposition. The decomposition function $R(\boldsymbol{a})$ has a depth of the number of iterations of $r(\boldsymbol{a})$, D. This process is shown below in its entirety as a composition of functions.

Fig. 25

$$\boldsymbol{k} = X[k] = \mathcal{F}(x[n])$$

$$\boldsymbol{a} = A(k)$$

$$\tilde{\boldsymbol{a}} = Q_{RM}(\boldsymbol{a})$$

$$r = R^{D}(\tilde{\boldsymbol{a}})$$

$$\boldsymbol{r} = \left(\mathcal{F} \circ A \circ Q_{RM} \circ R^{D}\right)(x[n])$$

3 Rn Encoding

(3.1) RM itself consists of two functions, and since, it is being considered here only as an operation of vectors, the question of generalizing RM may provide interest. To start, the notation of the encoder, R^D will be changed to R_N^D, where N is the number of functions in the decoding operation. Figure 16 demonstrates this development.

Fig. 26

$$RM = RM_1(a, b), RM_2(a, b)$$
$$R_N = R_1(a, b), \ldots, R_N(a, b)$$
$$RM = R_2$$

As shown above, Ring Modulation Decomposition is a R_2 encoder which compresses each amplitude bin vector down to 2 scalars. Working off of the definition of RM Decomposition in **(2.1 - 2.4)**, several properties of any feasible R_N encoder can be derived.

- Each R_N encoder consists of N functions
- Each function is $n: \mathbb{R} \to \mathbb{R}$
- Each function has at least two independent variables
- The R_N encoder also has a valid R_N quantizer.
- The Mode of Solutions method described in **2.2** is applicable for each iteration

To reinforce these properties, the below figure shows several other valid R_N encoders.

Fig. 27

$$F = R_2$$
$$F_1(a, b) = a + b + 1, \; F_2(a, b) = a + b + 2$$
$$G = R_3$$
$$G_1(a, b) = a + b, \; G_2(a, b) = |a - b|, \; G_3(a, b) = a \cdot b$$
$$H = R_2$$
$$H_1(a, b, c) = a + b + c, \; H_2(a, b, c) = |a + b - c|$$

4 Towards a Practical Encoder

(4.1) Moving towards a practical encoder, several problems, solutions, and general details must be covered. To keep the matter simple, it will be assumed that this encoder employs RMD, and psychoacoustic stages of compression will be excluded since I lack the expertise to discuss them. The following figure shows the potential stages of this theoretical encoder.

Stage 1. Data Preparation:

- Basic Sampling, Time Domain Data
- Fourier Related Transform, Frequency Domain Data
- Frequency Domain Quantization

Fig. 28 **Stage 2. Rn Preparation**

- Naive Dimension Quantization/c_{error} Calculation
- Dimension Promotion/Demotion

Stage 3. Rn Operations

- Dimension Quantization (DQ)
- Q_{RM}/RMD Quantization
- Ring Modulation Decomposition (RMD)
- Repeat 3.1 - 3.3 until 3 dimensions

Stage 1 was introduced in **(1.1)** and shown in **Fig. 24** in **(2.4)**. **Fig. 24** shows the encoding process given a 4-dimension to 3-dimension reduction, but it doesn't explain the problems at other dimensions. Recall the formula for dimension input and output in **Fig. 8**; Using RMD, dimensions that are a power of 2 can be reduced to a lower dimension, but dimensions that are not require another method of quantization, this is what I have labeled as Dimension Quantization (DQ).

Dimension Quantization is the process of adding redundant frequency information to an amplitude bin vector to increase the dimensions of a vector. Recall that each value in these vectors represents a frequency value, so the redundancy should have no effect on the perception of the vector. What may result in a change in perception is Q_{RM}. **Fig. 22** shows that the quantization process is directly proportional with the $c_{error,\ s}$ of the frequency pair, therefore to minimize the change caused by Q_{RM}, the redundant frequency chosen should be the one which results in the smallest c_{change} or sum of the absolute value of each $c_{error,\ s}$.

Fig. 29

$$c_{change} = \sum_{s=0}^{S-1} \left| c_{error,\ s} \right| = \sum_{s=0}^{S-1} \left| \sum_{p=0}^{P-1} c_p - P\left(c_s \right) \right|$$

To demonstrate this process, consider the 5-dimensional vector $\begin{bmatrix} 16 \\ 5 \\ 4 \\ 2 \\ 1 \end{bmatrix}$. To perform RMD on the vector, the dimension count should be padded with redundant frequencies to reach 8 dimensions. Padding the vector with 16 creates the frequency pairs (16, 1), (16, 2), (5, 4). Using the formula for c_{change}, the following figure demonstrates the computation when using 16 as the redundant frequency.

Fig. 30

$$\begin{bmatrix} 16 \\ 5 \\ 4 \\ 2 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} (16) \\ 16 \\ 5 \\ 4 \\ 2 \\ 1 \end{bmatrix}, \; c = \begin{bmatrix} 17 \\ 18 \\ 9 \end{bmatrix}, \; \bar{c} = 14.67$$

$$c_{change} = |14.67 - 17| + |14.67 - 18| + |14.67 - 9| = 11.33$$

Repeating this process for the other scalars gives the c_{change} for each padding frequency, revealing which frequency should be used for the smallest c_{change}.

Fig. 31

$$(16)\; c_{change} = 11.33\,, \; (5)\; c_{change} = 12$$
$$(4)\; c_{change} = 12.67\,, \quad (2)\; c_{change} = 14$$
$$(1)\; c_{change} = 14.67$$

From these calculations, it is clear that the best padding frequency to use in this case is 16. To reach higher dimensions, this process can be repeated by padding more than 1 frequency at once, and for certain dimension values, the quantization process must be repeated intermittently. For example, a 60-dimension vector undergoes the following process to be reduced to 3 dimensions. (Note that Q_{RM} is performed before every RMD if required).

Fig. 32

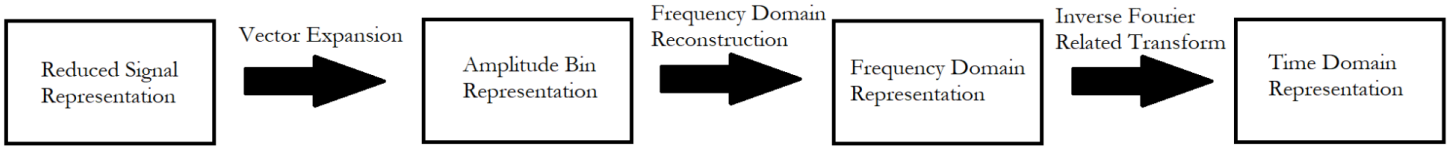
$$60 \rightarrow DQ \rightarrow 64 \rightarrow RMD \rightarrow 7 \rightarrow DQ \rightarrow 8 \rightarrow RMD \rightarrow 4 \rightarrow RMD \rightarrow 3$$

(4.2) With DQ defined, **Stage 3** is now defined in its entirety. Now, **Stage 2** and dimension promotion and demotion can be introduced as a method to reduce the potential quantization error in **Stage 3**. Once again, the problem of non-power of 2 dimensions should be considered. Instead of adding redundant frequency information to meet a dimension requirement, frequencies from a lower or higher amplitude bin can be moved to a target amplitude bin to meet the dimension requirement. As of this moment, I have yet to identify a practical method of determining which frequencies should be moved and how to calculate what will work best, but several points are being considered for an ideal method.

When promoting or demoting a frequency to a target band it should be considered whether to keep the frequency at the previous band or remove it. Depending on the spacing of each band, this operation may be perceptible or imperceptible. Frequencies should be carefully adjusted according to their psychoacoustic perceptibility. In general, this process should be weighed as a balance between the desired compression ratio and the acceptable change in perception. Like the other methods of quantization described before, the goal in simple terms should be to prepare the data for RMD while minimizing the $^{c}{}_{error}$.

(4.3) The decoding process is simply a reversal of the processes before except without the time consuming quantization. First, each vector is expanded according to the depth of the reduction to a higher dimension. From there, the vectors are reassembled as a frequency domain plot of data and the inverse of the fourier related transform applied is used to reconstruct the time domain audio data.

Fig. 33



The decoding process is much less complicated and straightforward than the encoding process, so much so that it might be possible to decode the data in real time for playback. Once a practical encoder has been implemented, the possibility of a real time decoder will be investigated thoroughly.

5 Future Consideration

(5.1) Most of the future research concerning this topic revolves around the quantization processes: amplitude bin quantization, R_N quantization, dimension quantization, and dimension promotion/demotion . Amplitude bin quantization may be done with different sizes of uniform horizontal slices, but it may also be done with variable size slices. It could be interesting to see how the size of the error makes an audible difference, and if it does, how could this quantization of the frequency domain be used to develop aurally interesting audio plugins and synthesizers?

R_N encoders and R_N quantization make up an extremely large portion of potential research. The quantization error is reduced by the minimizing the c_{error} for a target vector, so it makes sense to explore what R_N quantizers best minimize certain data sets. However, if R_N quantization doesn't affect the audibility of the data significantly, the problem becomes a more complicated optimization of perception and reduction. All of this is said without even addressing computational complexity and psychoacoustic compression, which will no doubt be important factors in the effectiveness and capabilities of any practical R_N encoder.

Dimension quantization and dimension promotion/demotion fall into a similar category with R_N quantization as they ideally aim to minimize c_{error}. However, unlike R_N quantization, manipulating the dimensions of a vector is required before reduction down to the minimum number of dimensions of an R_N encoder. A practical encoder will balance out acceptable c_{error} with dimension reduction and the aggression of dimension promotion/demotion. In general, these quantization tools allow any vector to be reduced to a minimum size, but at the moment, the trade off in quantization error isn't fully understood.

For the majority of the paper, most vectors shown are primarily understood as amplitude bin vectors where each scalar of the vector represents a frequency. What hasn't been addressed is the phase of each frequency, or in another phrasing, the imaginary component of each signal. The specifics of solving this problem come down to whichever fourier related transform may be employed for a practical algorithm, but if the imaginary frequency domain is considered itself, it may be reasonable to simply apply the same R_N encoder that was applied to the real domain to the imaginary domain. It may also be reasonable to attempt to reconstruct, with loss, the phase during decoding; much like some of the quantizers employed, phase could be a lossy component of the compression.

(5.2) R_N encoding could be incredibly useful for audio compression and audio tokenization. The method itself is attached to Fourier-related transforms, so it is safe to assume that with minimal error the encoding process could be used directly with the DCT to further compress data, and in the realm of audio processing and analysis, R_N encoders may be useful as a supplement to the standard DFT. R_N encoders that prioritize compression above all else could be extremely useful for deep learning models or other big data models; in specific, I only started exploring this topic after seeing the success

of residual vector quantization in audio data compression for audio generative models. Many methods of quantization could also be implemented for creative means as plugins for DAWs. For example, bitcrushing devices distort the signal through sampling rate reduction; employing a similar technique to dimension quantization and R_N quantization could result in unique auditory effects. In short, the potential of R_N encoding may be as deep as it is wide, but only time and further investigation will tell.