ΣΔ Quantization with the Hexagon Norm in $\mathbb{C}$

Michael Andrew Zichy

A Thesis Submitted to
University North Carolina Wilmington in Partial Fulfillment
Of the Requirements for the Degree of
Master of Arts

Department of Mathematics and Statistics

University North Carolina Wilmington

2006

Approved by

Advisory Committee

Chair

Accepted by

Dean, Graduate School

This thesis has been prepared in the style and format

consistent with the journal

American Mathematical Monthly.

TABLE OF CONTENTS

ABSTRACT vi

ACKNOWLEDGMENTS vii

1 Introduction 1

2 Frames 3

3 Quantization 8

3.1 PCM Quantization 8

3.2 $\Sigma\Delta$ Quantization 13

4 $\Sigma\Delta$ in $\mathbb{C}$ 15

5 Conclusion 25

APPENDIX 26

REFERENCES 41

LIST OF FIGURES

1 PCM Mean Square Error 11

2 PCM Mean Square Error 11

3 PCM Mean Square Error 12

4 $\Sigma\Delta$ Mean Square Error 14

5 Euclidean Norm Quantization 15

6 Box Norm Quantization 16

7 Diamond Norm Quantization 18

8 Hexagon Norm Quantization 19

9 Box Norm Quantization in $\mathbb{C}$ 22

10 Box Norm Quantization in $\mathbb{C}$ (odd roots of unity) 22

11 Box Norm Quantization in $\mathbb{C}$ (even roots of unity) 23

12 Diamond Norm Quantization in $\mathbb{C}$ 23

13 Hexagon Norm Quantization in $\mathbb{C}$ 24

LIST OF TABLES

1 1-Bit PCM: Mean Square Error 34

2 2-Bit PCM: Mean Square Error 35

3 3-Bit PCM: Mean Square Error 36

4 1-Bit $\Sigma\Delta$: Mean Square Error 37

5 $\Sigma\Delta$ Box Quantization in $\mathbb{C}$: Mean Square Error 38

6 $\Sigma\Delta$ Diamond Quantization in $\mathbb{C}$: Mean Square Error 39

7 $\Sigma\Delta$ Hexgon Quantization in $\mathbb{C}$: Mean Square Error 40

ABSTRACT

It has been shown that Pulse Code Modulation (PCM) Quantizer Schemes and $\Sigma\Delta$ Quantizer Schemes can be used to quantize one dimensional data for transmission and recovery with each scheme having its own advantages and disadvantages. This work will discuss the viability of a two dimensional $\Sigma\Delta$ Quantization scheme for use in transmission and recovery. Three distinct two dimensional quantizer norms, the Box, the Diamond, and the Hexagon norm, will be compared using the Mean Square Error to show that the Hexagon Quantizer norm out performs the others and serves the purpose for a practical two dimensional quantizer.

ACKNOWLEDGMENTS

I would like to thank Dr. Michael Freeze for teaching me how to write in a mathematics paper and Dr. Mark Lammers for teaching me what to write in a mathematics paper. Without their guidance and support I could never have come as far as I have, and for that they have my respect and my gratitude.

1 Introduction

There is a lot of signal information in a thirty minute symphony. A myriad of instruments are playing different notes while cavatinas, concertos, glissandos, trills, and a host of other musical terms that my high school band years failed to prepare me for are occurring at each and every moment. Every moment is full of information that is invaluable to the symphony. How can one hope to put a seemingly infinite amount of information onto a CD so that the symphony can be enjoyed days later at one's leisure?

This question in signal processing was answered with the introduction of the Nyquist-Shannon Sampling Theorem which showed that Fourier transforms offer a way of taking continuous data and "sampling" it for the most important pieces. This sampling creates a discrete set, similar to the way the entirety of a symphony can be stored in a few black dots and marks on sheet music, that can be used for storage, transmission, and recovery . In this paper, the signals of interest will be the analog signals created by having already sampled a continuous signal.

However, this analog signal itself will not suffice for the purpose of real signal transmission and recovery. Computers are naturally restricted to a set number of bits in which to process information. In other words, although one may succeed in reducing an entire thirty minute symphony's information into a mere 100 irrational numbers, this will be worthless to a computer that only understands the language of 0's and 1's.

A method will be necessary to convert analog signals over to "digital" signals, i.e. to convert the finite numbers that make up the analog signal into a smaller set of numbers from a specifically chosen alphabet. This conversion will be taken care of by a process called "quantization". Of course, this quantization will introduce error and it will be important to both measure and minimize this error. This paper will compare two different methods of quantization, PCM Quantization and ΣΔ Quantization, and compare their error using a new Quantizer in $\mathbb{R}^2$.

The first step toward a digital representation of an analog signal x is to expand the signal over a given dictionary e_n such that

$$x = \sum c_n e_n$$

where the c_n are real or complex numbers. This step can be achieved through use of the concept of Frames developed by Duffin and Schaeffer in 1952. However, while this representation is certainly discrete, it is not "digital" since the coefficients are real or complex valued. The second step in the process, the quantization, reduces the continuous range of this sequence to a discrete set. We create a new signal

$$\tilde{x} = \sum q_n e_n$$

where q_n are elements of a discrete, finite set called the quantization alphabet. The performance of the quantizer can be measured using the approximation error $\|x - \tilde{x}\|$ where $\|\cdot\|$ is a suitable norm.

2 Frames

When working with applications it is often convenient to assume that the signals of interest are elements of a Hilbert space (i.e. an inner product space with a defined norm). So a signal x can be thought of as a vector in a Hilbert space such as $H = \mathbb{R}^n$ or $H = \mathbb{C}^n$. Frames are a special type of dictionary $\{e_n\}$ which can be used to give stable redundant decompositions of a signal. Informally, a frame can be thought of as a spanning set of the given vector space with a few minor restrictions, but to be technical the following definition is used.

Definition 2.1 (Frame) *A collection $F = \{e_n\}$ in a Hilbert Space H is a **frame** for H if there exists $0 < A \leq B < \infty$ such that*

$$\forall x \in H, A\|x\|^2 \leq \sum_n |\langle x, e_n \rangle|^2 \leq B\|x\|^2$$

The constants A and B are called the frame bounds, and if $A = B$, then F is called a tight frame.

I will restrict discussion in this paper to tight frames as they have nice properties and the results could be generalized to non-tight frames if one were so inclined. Also, because it will be necessary in this paper to create frames of arbitrary size it will be convenient to use roots of unity.

Definition 2.2 (N^{th} Roots of Unity) *The N^{th} roots of unity are given by*

$$e^{\frac{2\pi i}{N}} = \cos(\frac{2\pi n}{N}) + i \cdot \sin(\frac{2\pi n}{N})$$

for $0 \leq n < N$

When working over $\mathbb{C}$, the roots of unity create a frame. However, the roots of unity can also be generalized to higher dimensions using harmonic frames. Thus, for example, if working over $\mathbb{R}^2$ and a frame with 4 vectors is desired, then the frame vectors can be generated by $\{e_1, e_2, e_3, e_4\} =$

$$\left\{ \begin{bmatrix} \cos(\frac{0\pi}{4}) \\ \sin(\frac{0\pi}{4}) \end{bmatrix}, \begin{bmatrix} \cos(\frac{4\pi}{4}) \\ \sin(\frac{4\pi}{4}) \end{bmatrix}, \begin{bmatrix} \cos(\frac{8\pi}{4}) \\ \sin(\frac{8\pi}{4}) \end{bmatrix}, \begin{bmatrix} \cos(\frac{12\pi}{4}) \\ \sin(\frac{12\pi}{4}) \end{bmatrix} \right\}$$

or simply

$$e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, e_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, e_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, e_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

The following shows how a frame will be used in signal processing.

Definition 2.3 (Analysis Operator) *Let $\{e_n\}$ be a frame for a Hilbert Space H with frame bounds A and B. The **analysis operator***

$$F : H \longmapsto \ell^2$$

*is defined by $(Fx)_k = \langle x, e_k \rangle$. The operator $S = F^*F$ is called the **frame operator** and satisfies*

$$AI \le S \le BI$$

*where I is the identity operator on H. The inverse of S, S^{-1}, is called the **dual frame operator** and satisfies*

$$B^{-1}I \le S^{-1} \le A^{-1}I$$

Frames are useful in signal processing because of the following theorem.

Theorem 2.1 *Let $\{e_n\}$ be a frame for H with frame bounds A and B, and let S be the corresponding frame operator. Then $\{S^{-1}e_n\}$ is a frame for H with frame bounds*

B^{-1} and A^{-1}. Further, for all $x \in H$

$$x = \sum_n \langle x, e_n \rangle (S^{-1} e_n)$$

$$= \sum_n \langle x, (S^{-1} e_n) \rangle e_n$$

These atomic decompositions are the first step towards a digital representation. If the frame is tight with frame bound A, then both frame expansions are equivalent and thus

$$\forall x \in H, x = A^{-1} \sum_n \langle x, e_n \rangle e_n$$

Recall that all the frames in this paper will be tight frames and so the above equality will hold.

To illustrate the process of deconstructing and reconstructing a signal, it will be helpful to see a worked out example. Suppose the Hilbert Space chosen is $H = \mathbb{R}^2$ and the vector to be transmitted is $x = \begin{bmatrix} a \\ b \end{bmatrix}$. Further suppose that the frame chosen is $F = \{e_1, e_2, e_3, e_4\}$ as above (i.e $e_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, $e_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $e_3 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$, $e_4 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}$). Thus, $F = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix}$ and so $F \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \\ -a \\ -b \end{bmatrix}$. In other words, to transmit the vector $x = \begin{bmatrix} a \\ b \end{bmatrix}$, four numbers $\{a, b, -a, -b\}$ are sent instead.

Computing the frame operator:

$$S = F^*F = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & 0 \\ 0 & -1 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

Thus,

$$S^{-1} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix}$$

After the transmission of a, b, $-a$ and $-b$, the receiver can use the above theorem to reconstruct the original vector:

$$x = \sum_{1}^{3} \langle x, e_n \rangle (S^{-1})$$

$$= aS^{-1} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + bS^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + (-a)S^{-1} \begin{bmatrix} -1 \\ 0 \end{bmatrix} + (-b)S^{-1} \begin{bmatrix} 0 \\ -1 \end{bmatrix}$$

$$= a \begin{bmatrix} \frac{1}{2} \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ \frac{1}{2} \end{bmatrix} + (-a) \begin{bmatrix} -\frac{1}{2} \\ 0 \end{bmatrix} + (-b) \begin{bmatrix} 0 \\ -\frac{1}{2} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{1}{2}a + 0 + -\frac{1}{2}(a) + 0 \\ 0 + \frac{1}{2}b + 0 + -\frac{1}{2}(-b) \end{bmatrix}$$

$$= \begin{bmatrix} a \\ b \end{bmatrix}$$

Note that if the frame chosen had been a basis, only two numbers would have been sent. In $\mathbb{R}^2$, only two numbers are actually necessary (and in general, in $\mathbb{R}^d$ only d numbers are needed). Choosing only d vectors for a frame in $\mathbb{R}^d$ is called "critically sampling." However, there are many benefits to "oversampling" the vector. A frame that sends more information than necessary is said to be redundant and will be vital to signal processing as shall be shown in the next section.

3 Quantization

In the previous example, the vector $x = \begin{bmatrix} a \\ b \end{bmatrix}$ was broken down by the frame and transmitted as the numbers a, b, $-a$ and $-b$. While this is certainly analog, it is not digital. Suppose, for example, that $a = \cos(1)$ and $b = \sin(1)$. Unless there is a way to convert this discrete data to digital data, frames will be useless for signal processing. Naively, binary could be offered as a solution; however, binary proves to be computationally inefficient and expensive to transmit. If the fraction $\frac{41}{64}$ is to be transmitted in binary, the number .101001 must be sent. Suppose that, unfortunately, unforeseen circumstances cause the signal to be disturbed and instead the information received is .001001 (a mere one digit switch up) or the fraction $\frac{9}{64}$. Binary lacks any efficient means to deal with lost or corrupted information (a situation that may be more common then previously realized when one considers that information is often rounded or truncated in the transmission process). A more efficient scheme that converts discrete data to digital would, therefore, be preferred.

3.1 PCM Quantization

Because it is necessary to take the $x_n = \langle x, e_n \rangle$ and approximate it with a quantized number q_n from a set digital alphabet (depending on whether one were allotted 2-bits, 4-bits, 8-bits, etc.), a natural choice would be to use a quantization technique known as the $2\lceil 1/\delta \rceil$-level PCM quantizer with step size δ given by replacing $x_n = \langle x, e_n \rangle$ with $q_n = \delta(\lceil x_n/\delta \rceil - 1/2)$ from the alphabet defined by $\mathcal{A} = \{(-K + 1/2)\delta, (-K + 3/2)\delta, ..., (-1/2)\delta, (1/2)\delta, ..., (K - 1/2)\delta\}$ where $K \in \mathbb{N}$. In other words, the q_n in the alphabet that is closest to x_n will be what is used to approximate x_n. This idea of "closeness" will require that a suitable norm on the given Hilbert Space be chosen. Again, an example will help clarify exactly how the process works:

Suppose the vector $x = \begin{bmatrix} \cos(1) \\ \sin(1) \end{bmatrix}$ is to be transmitted. Here the space $H = \mathbb{R}^2$ will be convenient to work over. It is assumed that the given alphabet is $\mathcal{A} = \{1, -1\}$, thus the step size $\delta = 2$. Further suppose that the frame size is $N = 3$. Thus,

$$x = \begin{bmatrix} 0.54030230586814 \\ 0.84147098480790 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 0 \\ -0.5 & 0.86602540378444 \\ -0.5 & -0.86602540378444 \end{bmatrix}$$

Computing the frame coefficients gives the sequence:

$$x_n = \{0.54030230586814,\ 0.45858409645708,\ -0.99888640232522\}$$

PCM simply rounds each frame coefficient to the nearest member of the alphabet. Since the given alphabet is $\mathcal{A} = \{-1, 1\}$, the frame coefficients compute to:

$$q_n = \{1, 1, -1\}$$

These frame coefficients are what get transmitted to the receiver who then uses the theorem to reconstruct an approximation of the original signal:

$$x \approx A^{-1} \sum_{n=1}^{3} q_n e_n$$

where

$$S = F^*F = \begin{bmatrix} 1 & -0.5 & -0.5 \\ 0 & 0.86602540378444 & -0.86602540378444 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -0.5 & 0.86602540378444 \\ -0.5 & -0.86602540378444 \end{bmatrix}$$

and thus,

$$S^{-1} = \begin{bmatrix} \frac{2}{3} & 0 \\ 0 & \frac{2}{3} \end{bmatrix}$$

The receiver reconstructs the original signal:

$$x \approx \left(\frac{3}{2}\right)^{-1} \sum_{n=1}^{3} q_n e_n$$

$$x \approx \frac{2}{3}\left(1 \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 1 \cdot \begin{bmatrix} -0.5 \\ 0.86602540378444 \end{bmatrix} - 1 \cdot \begin{bmatrix} -0.5 \\ -0.86602540378444 \end{bmatrix}\right)$$

$$x \approx \begin{bmatrix} .66666666666667 \\ 1.1547005383793 \end{bmatrix}$$

Computing the error $|x - \tilde{x}|$ yields

$$\sqrt{(0.54030230586814 - .66666666666667)^2 + (0.84147098480790 - 1.1547005383793)^2}$$

Thus $|x - \tilde{x}| = .337758352$. It has been shown that the Mean Square Error of the PCM scheme is approximately on the order of $\frac{1}{N}$ [1] and since $N = 3$ for this example, the error is not unreasonable. One would expect that as N increases, the error would decrease to acceptable levels. In Figure 1, the Mean Square Error of five hundred different trials for each $2 \leq N \leq 64$ using the N^{th}-roots of unity and the alphabet $\mathcal{A} = \{-1, 1\}$ is shown.

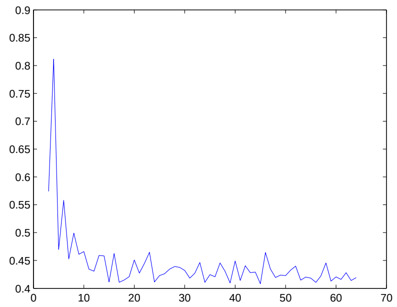

Figure 1: PCM Mean Square Error

Notice that as N increases, the error does not seem to be uniformly decreasing. This oscillation occurs because the PCM scheme does not consistently operate well with a small alphabet. In the above example, the smallest possible alphabet was chosen for simplicity. Thus, while some of the results where approximately on the order of $\frac{1}{N}$, others became unstable and were not. Increasing the size of the alphabet will yield the expected results. If the alphabet is increased in size to $\mathcal{A} = \{-1, -0.5, 0.5, 1\}$, and we take the Mean Square Error of five hundred trials again, then the results are as shown in Figure 2.

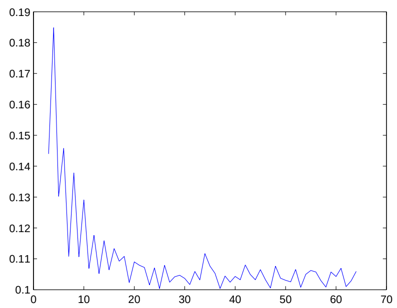

Figure 2: PCM Mean Square Error

If the alphabet were increased even more to $\mathcal{A} = \{-1, -0.75, -0.5, -0.25, 0.25, 0.5, 0.75, 1\}$, the results are even better as shown in Figure 3.

It should be noted that as the size of N increases, the PCM scheme does improve,

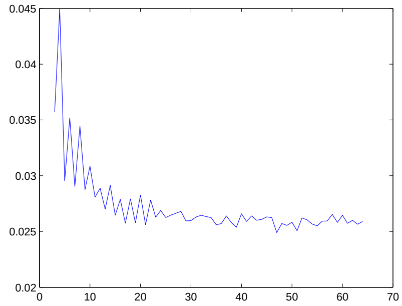

Figure 3: PCM Mean Square Error

but it appears to improve only up to a point. It appears that the error function is asymptotic. This result is explained by the fact that the Mean Square Error can be shown to have an additive constant on the order of δ^2. Thus, improving the size of the alphabet is necessary to improve the MSE past a certain size frame vector.

3.2 $\Sigma\Delta$ Quantization

While the PCM scheme is easy to implement, it fails to take full advantage of the redundancy of the frame. Also, if the alphabet must be restricted to a small size, then the scheme has a chance of becoming unstable and not producing accurate reconstructions. Another scheme, called $\Sigma\Delta$ Quantization, better handles small alphabets and utilizes the redundancy of the frame.

In the $\Sigma\Delta$ scheme, a running tab on the errors created as x_n is approximated by q_n is kept so that the scheme can compensate. Put simply, if the scheme recognizes that it has rounded the last few x_n's down, it may round the next x_n up to even out the error and vice versa. $\Sigma\Delta$ Quantization works similarly to the PCM case in that an alphabet $\mathcal{A}$, a step size δ, and a frame size N are chosen. The difference occurs in how the quantizer operates.

Definition 3.1 ($\Sigma\Delta$ Quantizer) *The first order $\Sigma\Delta$ Quantizer is defined by the iteration*

$$u_n = u_{n-1} + x_n - q_n$$

$$q_n = Q(u_{n-1} + x_n)$$

where $u_0 = 0$ and Q(u) is defined by:

$$Q(u) = arg\ min_{q\in\mathcal{A}}|u - q|$$

In other words, $Q(u)$ is the element of the alphabet closest to u. If two members of the alphabet are equally close, the larger number is chosen.

Looking back at the example from before. Suppose the vector $x = \begin{bmatrix} \cos(1) \\ sin(1) \end{bmatrix}$ is to be transmitted. Again, working over the space $H = \mathbb{R}^2$ using the N^{th}-roots of unity and the alphabet $\mathcal{A} = \{-1, 1\}$ the frame coefficients are calculated:

$$x_n = \{0.54030230586814,\ 0.45858409645708,\ -0.99888640232522\}$$

Quantizing gives the following:

$$q_1 = Q(u_0 + x_1) = Q(0 + 0.54030230586814) = 1$$

$$u_1 = u_0 + x_1 - q_1 = 0 + 0.54030230586814 - 1 = -.45969769413186$$

$$q_2 = Q(u_1 + x_2) = Q(-.45969769413186 + 0.45858409645708) = -1$$

$$u_2 = u_1 + x_2 - q_2 = -.45969769413186 + 0.45858409645708 - (-1) = .99888640232522$$

$$q_3 = Q(u_2 + x_3) = Q(.99888640232522 + (-0.99888640232522)) = 1$$

Thus, the appropriate quantized sequence $q_n = \{1,\ -1,\ 1\}$ would be used to reconstruct an approximation of the original signal.

In Figure 4, five hundred random vectors of appropriate norm were used to compute the Mean Square Error where $2 \leq N \leq 64$ using the N^{th}-roots of unity and the alphabet $\mathcal{A} = \{-1, 1\}$.

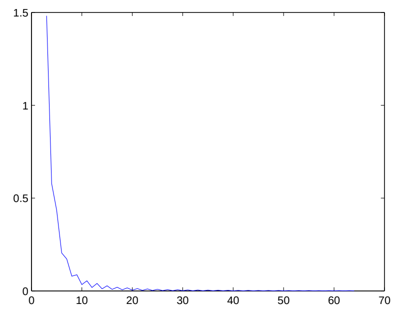

Figure 4: $\Sigma\Delta$ Mean Square Error

4 $\Sigma\Delta$ in $\mathbb{C}$

In $\mathbb{R}$, the quantization amounts to dividing the number line into a set number of regions. Working in $\mathbb{R}^d$, the quantization problem becomes a problem of "tiling" the space. For example in $\mathbb{R}^2$, notice that the Euclidean norm $|\vec{x}| = \sqrt{x^2 + y^2}$ is not a reasonable norm since it creates areas of overlap.

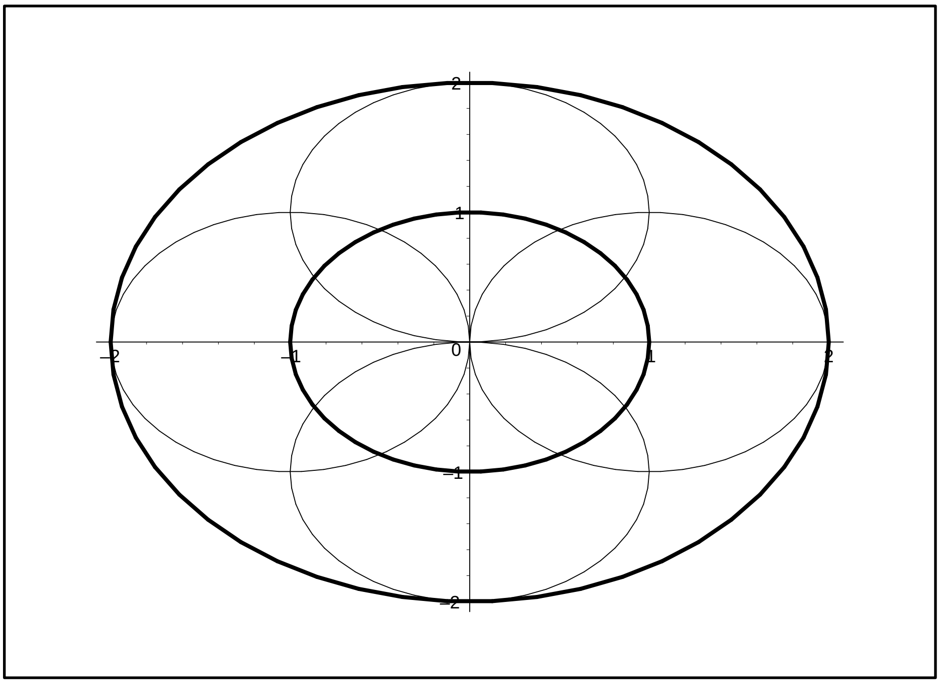

Figure 5: Euclidean Norm Quantization

In the figure above, all the possible vectors are inside the bold circle of radius 1 about the origin. All the calculated x_n's are therefore in the circle of radius 2 about the origin and need to be rounded to one of the points $(1, 0)$, $(0, 1)$, $(-1, 0)$, $(0, -1)$ (i.e. q_n's represented by the centers of the four inner circles). So the point $x_n = (1.5, 0.5)$ would be quantized to the point $q_n = (1, 0)$ since it falls in that circle. Note, however, that some areas are overlapped and thus it becomes difficult to determine which q_n to round to (such as the point $(0.5, 0.5)$), while other regions are not represented in the circles at all (such as the point$(1.3, 1.3)$) and thus cannot be rounded to any of the q_n's.

The problem with the Euclidean norm is that it does not "tile" the space. It creates regions of overlap and misses regions entirely. The goal will be to tessellate the plane so as to have a clear quantization scheme, but to tessellate using regular shapes so that the quantization is not overly complex. One such norm that would be suitable would be the **Box norm** $|\cdot| = \sup|x, y|$:

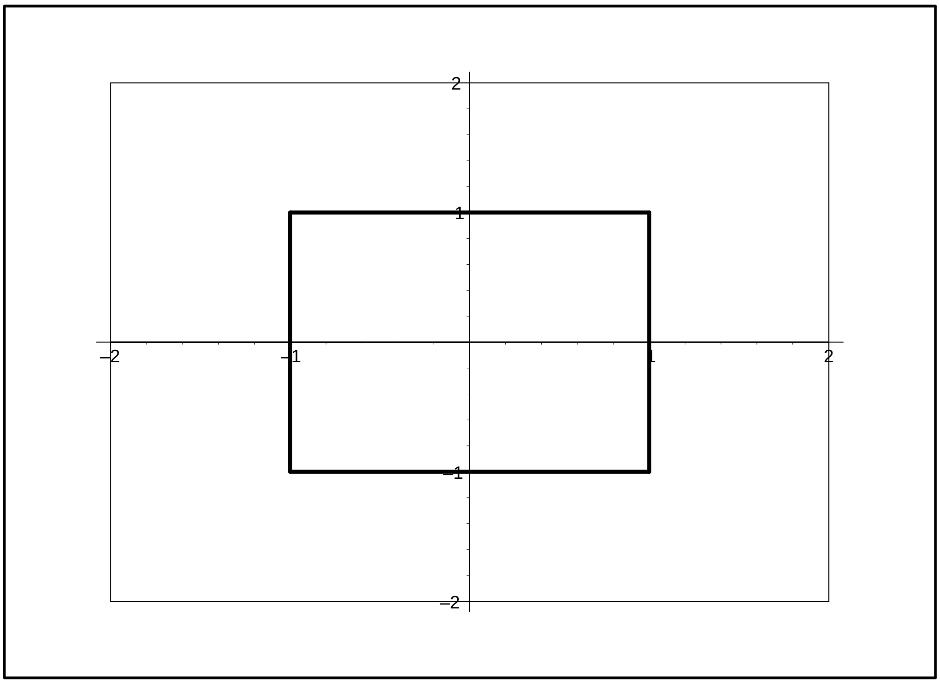

Figure 6: Box Norm Quantization

It might first be beneficial to prove that the Box norm is, in fact, an actual norm.

Definition 4.1 (Norm) *Given an n-dimensional vector $\overrightarrow{v}$, $|\cdot|$ is a norm given that:*

1) $|\overrightarrow{v}| \geq 0$ *and* $|\overrightarrow{v}| = 0$ *iff* $\overrightarrow{v} = \overrightarrow{0}$

2) $|k\overrightarrow{v}| = |k||\overrightarrow{v}|$ *for any scalar* k

3) $|\overrightarrow{v_1} + \overrightarrow{v_2}| \leq |\overrightarrow{v_1}| + |\overrightarrow{v_2}|$

The Box Norm clearly satisfies the first criterion since it is defined to be the absolute value of the largest of the vector's elements. To prove it satisfies the second condition, note:

$$|k\overrightarrow{v}| = |\begin{bmatrix} kx \\ ky \end{bmatrix}| = \sup|kx, ky| = |k|\sup|x, y| = |k||\overrightarrow{v}|$$

For the third condition, note that:

$$|\overrightarrow{v_1} + \overrightarrow{v_2}| = |\begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}| = \sup|x_1 + x_2, y_1 + y_2|$$

If $x_1 + x_2 > y_1 + y_2$, then

$$\sup|x_1 + x_2, y_1 + y_2| = |x_1 + x_2| \leq |x_1| + |x_2|$$

by the triangle inequality and thus

$$|\overrightarrow{v_1} + \overrightarrow{v_2}| \leq |\overrightarrow{v_1}| + |\overrightarrow{v_2}|$$

The proof is similar if $y_1 + y_2 > x_1 + x_2$. Thus, the Box Norm is a norm.

Note that the boxes completely tessellate the plane (and more specifically the region of norm 2 that is of immediate concern) and the box is also a regular shape so the norm is not overly complicated to compute. In the figure above, all the points x_n will fall within the large two by two square, and each will be quantized to one of the points $(1, 1)$, $(1, -1)$, $(-1, 1)$, $(-1, -1)$ depending on which of the four smaller squares the point falls in.

Another suitable norm would be the **Diamond norm**: $|\cdot| = |x| + |y|$:

Again, to show that the Diamond Norm is in fact a norm is not difficult. The first condition is clear since the sum of the absolute value of two numbers will always

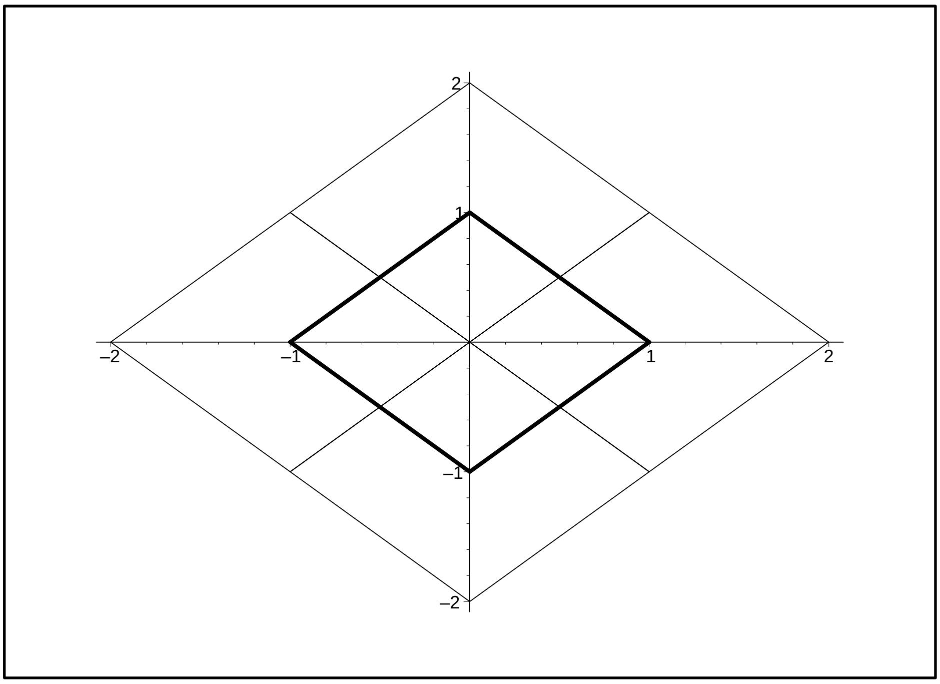

Figure 7: Diamond Norm Quantization

be positive or zero and zero just in the case that those two numbers were both zero. The second condition can be shown to be satisfied:

$$|k\overrightarrow{v}| = |\begin{bmatrix} kx \\ ky \end{bmatrix}| = |kx| + |ky| = |k||x| + |k||y| = |k|\left(|x| + |y|\right) = |k||\overrightarrow{v}|$$

And the third can be shown as well given that:

$$|\overrightarrow{v_1} + \overrightarrow{v_2}| = |\begin{bmatrix} x_1 + x_2 \\ y_1 + y_2 \end{bmatrix}| = |x_1 + x_2| + |y_1 + y_2|$$

and by the triangle inequality used twice

$$|x_1 + x_2| + |y_1 + y_2| \leq |x_1| + |x_2| + |y_1| + |y_2| = |x_1| + |y_1| + |x_2| + |y_2| = |\overrightarrow{v_1}| + |\overrightarrow{v_2}|$$

Thus, the Diamond Norm is a norm.

This is basically a modified version of the Box norm. Note that all the x_n's will fall within the large diamond, and each will get mapped to the point $(1,0)$, $(0,1)$, $(-1,0)$, or $(0,-1)$ depending on which smaller diamond the x_n resides in.

The reason it is so important for $|\overrightarrow{x}| \leq 1$ is because falling outside this region will create instability in the $\Sigma\Delta$ Quantization scheme. Any vector $\overrightarrow{x}$ such that $|\overrightarrow{x}| \leq 1$ is at most one unit away from any quantizer (by creation), so $|u_n| \leq 1$ thus the computations never involve going outside the region $|\cdot| \leq 2$. The $\Sigma\Delta$ Quantization scheme is such that all x_n's that are outside the region $|\cdot| \leq 1$ will get shifted back inside the by the corresponding u_n's. This is why it is so important for $|\overrightarrow{x}| \leq 1$. Otherwise, computations may send calculations outside of the $|\cdot| \leq 2$ region that the u_n's will not be able to shift back. Situations such as this could involve having to round an incredibly large x_n to a much smaller set number of the alphabet creating large errors.

A third norm considered is the **Hexagon norm**: $|\cdot| = \frac{1}{\sqrt{3}}\left[|y| + |\frac{\sqrt{3}}{2}x - \frac{1}{2}y| + |\frac{-\sqrt{3}}{2}x - \frac{1}{2}y|\right]$

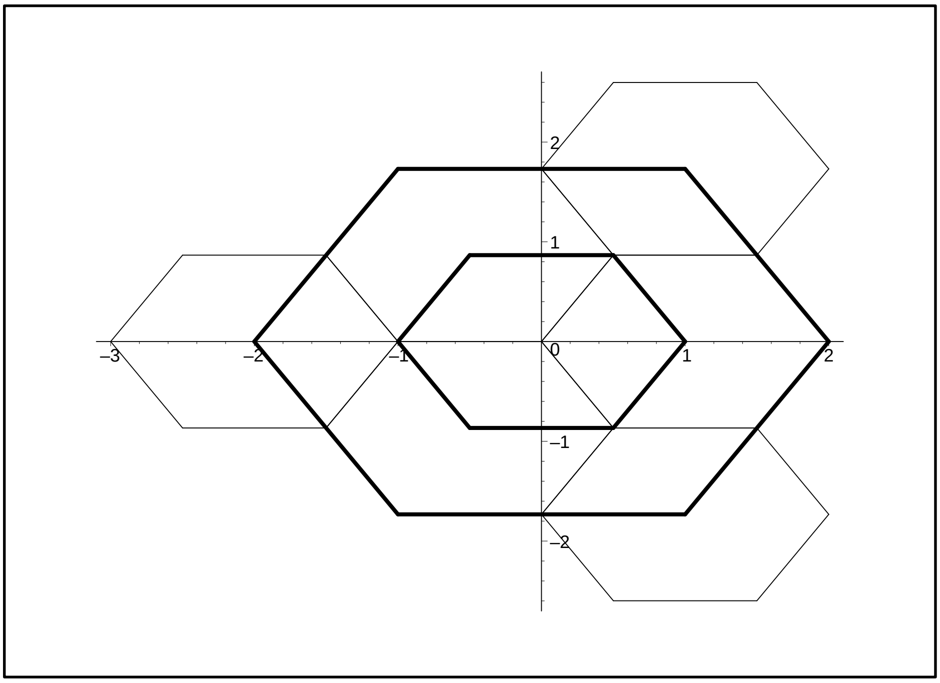

Figure 8: Hexagon Norm Quantization

Again, it needs to be shown that the Hexagon Norm is a true norm. It is easy to see that it satisfies the first condition as it is defined to be a positive number multiplied by the sum of three absolute values. This quantity will always be positive except in the case that both x and y were zero. For the second condition, note that:

$$|k\vec{x}| = \left| \begin{bmatrix} kx \\ ky \end{bmatrix} \right| = \frac{1}{\sqrt{3}} \left[|ky| + |\frac{\sqrt{3}}{2}kx - \frac{1}{2}ky| + |\frac{-\sqrt{3}}{2}kx - \frac{1}{2}ky| \right]$$

$$= \frac{1}{\sqrt{3}} \left[|k||y| + |k||\frac{\sqrt{3}}{2}x - \frac{1}{2}y| + |k||\frac{-\sqrt{3}}{2}x - \frac{1}{2}y| \right]$$

$$= |k|\frac{1}{\sqrt{3}} \left[|y| + |\frac{\sqrt{3}}{2}x - \frac{1}{2}y| + |\frac{-\sqrt{3}}{2}x - \frac{1}{2}y| \right] = |k||\vec{x}|$$

The proof of the third condition relies on using the triangle inequality three times:

$$|\vec{v_1}+\vec{v_2}| = \frac{1}{\sqrt{3}} \left[|y_1 + y_2| + |\frac{\sqrt{3}}{2}(x_1 + x_2) - \frac{1}{2}(y_1 + y_2)| + |\frac{-\sqrt{3}}{2}(x_1 + x_2) - \frac{1}{2}(y_1 + y_2)| \right]$$

and since $|y_1 + y_2| \leq |y_1| + |y_2|$

$$\leq \frac{1}{\sqrt{3}} \left[|y_1| + |y_2| + |\frac{\sqrt{3}}{2}x_1 + \frac{\sqrt{3}}{2}x_2 - \frac{1}{2}y_1 - \frac{1}{2}y_2| + |\frac{-\sqrt{3}}{2}x_1 + \frac{-\sqrt{3}}{2}x_2 - \frac{1}{2}y_1 - \frac{1}{2}y_2| \right]$$

and similarly,

$$\leq \frac{1}{\sqrt{3}} \left[|y_1| + |y_2| + |\frac{\sqrt{3}}{2}x_1 - \frac{1}{2}y_1| + |\frac{\sqrt{3}}{2}x_2 - \frac{1}{2}y_2| + |\frac{-\sqrt{3}}{2}x_1 - \frac{1}{2}y_1| + |\frac{-\sqrt{3}}{2}x_2 - \frac{1}{2}y_2| \right]$$

which can be shown using the associative property to be

$$= |\vec{v_1}| + |\vec{v_2}|$$

Thus the Hexagon Norm is a norm.

Here, all the x_n's will fall within the large hexagon and get mapped to the point $(1,0)$, $(1,\sqrt{3})$, $(1,-\sqrt{3})$, $(-\frac{1}{2},\frac{\sqrt{3}}{2})$, $(-\frac{1}{2},-\frac{\sqrt{3}}{2})$, or $(-2,0)$. Note that the hexagons completely tessellate the space, and while they actually cover more space than is needed, this does not cause any problems. As it turns out, the Hexagon norm can be shown to be the best at minimizing the mean square error in $\mathbb{R}^2$ [4].

In $\mathbb{R}^3$, it has been shown that the truncated octahedron best minimizes the mean square error, and for higher dimensions the problem is still unsolved.

One might note that $\mathbb{C}$ behaves very much like the space $\mathbb{R}^2$ and can be be "tiled" in a similar fashion using the Box, Diamond, and Hexagon norms. So instead of sending vectors from $\mathbb{R}^d$ to $\mathbb{R}$ to quantize, vectors in $\mathbb{R}^d$ could be sent to $\mathbb{C}$. Since the $\Sigma\Delta$ Quantizer performs better than the PCM Quantizer given the same δ and N in the one dimensional quantizer case, it would be suspected that the $\Sigma\Delta$ Quantizer should perform better than the PCM Quantizer in the two dimensional case as well.

Suppose a vector in $\mathbb{C}^2$ represents the signal to be transmitted. Using the Harmonic Frame $e_N = \begin{bmatrix} \cos(\frac{2\pi(k-1)}{N}) \\ \sin(\frac{2\pi(k-1)}{N}) \end{bmatrix}$ it will be possible to convert the original signal into an analog signal composed of a finite number of elements in $\mathbb{C}$. These complex numbers can the be quantized similar to the example above using either the PCM, the box, the diamond, or the hexagon quantizer.

Let $\begin{bmatrix} a_1 + b_1 i \\ a_2 + b_2 i \end{bmatrix}$ be a vector in $\mathbb{C}^2$. Using the box quantizer norm $|\cdot| = \sup|x,y|$ using the N^{th}-roots of unity and comparing the log log plot of the mean square error of 500 random trials yields the following results: The two distinct linear patterns developing are an expected result from the choice of odd roots of unity versus even roots of unity. Comparing the two separately yields:

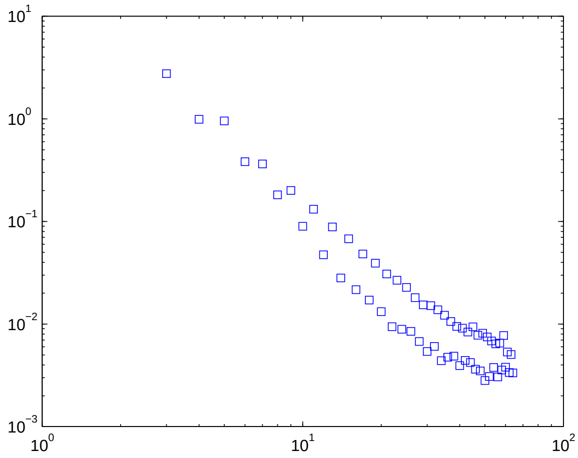

Figure 9: Box Norm Quantization in $\mathbb{C}$

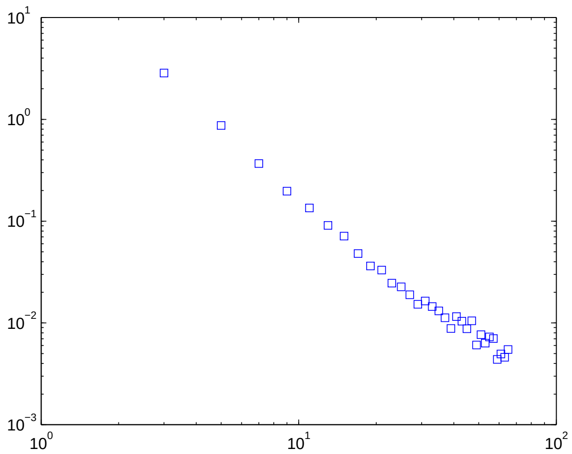

Figure 10: Box Norm Quantization in $\mathbb{C}$ (odd roots of unity)

Using the diamond quantizer norm $|\cdot| = |x| + |y|$ with the N^{th}-roots of unity $3 \leq N \leq 64$ and comparing the log log plot of the mean square error of 500 random

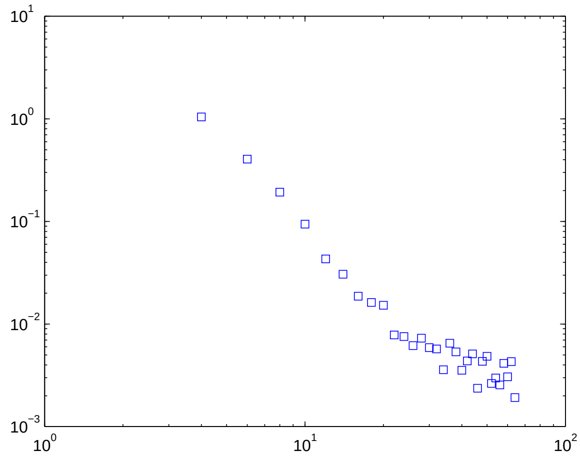

Figure 11: Box Norm Quantization in $\mathbb{C}$ (even roots of unity)

trials yields the following results: Again a sharp division can be seen between the

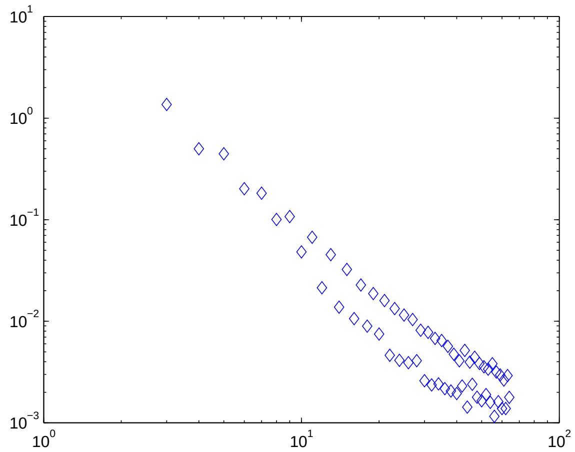

Figure 12: Diamond Norm Quantization in $\mathbb{C}$

choice of even and odd roots of unity.

Using the hexagon quantizer norm $|\cdot| = |x| + |y|$ with the N^{th}-roots of unity $3 \leq N \leq 64$ and comparing the log log plot of the mean square error of 500 random trials yields the following results: In this case, two distinct lines form where every

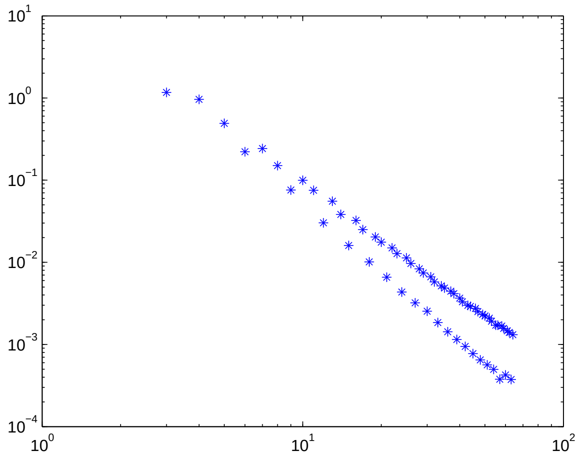

Figure 13: Hexagon Norm Quantization in $\mathbb{C}$

third root of unity is chosen. This arises from the properties of the hexagon norm.

5 Conclusion

It has been shown that any vector in $\mathbb{R}^d$ can be mapped to a sequence in R using a suitable frame, and that this mapping can be quantized into a digital signal suitable for transmission and recovery in signal processing. Both the PCM Quantizer and the $\Sigma\Delta$ Quantizer work, but the PCM scheme was shown to be heavily dependent on the size of δ because increasing the size of the frame N will only improve the MSE up to a point asymptotically. Since the size of the alphabet cannot always feasibly be increased in practice, the PCM scheme may not be the best scheme to implement. Since the $\Sigma\Delta$ Quantization Scheme does not necessarily need to use a larger alphabet to be comparable in practice to the PCM scheme, it may be more useful in certain applications. Also, since it has also been shown that quantizing in higher orders (i.e. $\mathbb{R}^2$ versus $\mathbb{R}$) reduces the mean square error and that, in particular, the hexagon quantizing scheme minimizes mean square error in $\mathbb{R}^2$, it would be prudent to test the value of a frame that maps $\mathbb{R}^d$ to $\mathbb{R}^2$ (or $\mathbb{C}$).

In this paper, three distinct quantizer norms, the Box, the Diamond, and the Hexagon, were compared using the mean square error of each. The Hexagon proved to be noticeably better at minimizing the mean square error. Thus, if the goal is to quantize a signal by mapping $\mathbb{R}^d$ to a sequence in two dimensional space $\mathbb{C}$, the Hexagon Quantizer is a good choice for minimizing error.

APPENDIX

1-Bit PCM in $\mathbb{R}^2$

```
clear for g=2:64
for L=1:500
%%%%%%%%%%Vector Creation%%%%%%%%%%
test=1; while test==1
    O=[(2*rand-1); (2*rand-1)];
    if sqrt(O(1)^2+O(2)^2)<=1
        test=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Frame%%%%%%%%%%%
N=g; for k=1:N
   e(k,1)=[cos(2*pi*(k-1)/N)];
   e(k,2)=[sin(2*pi*(k-1)/N)];
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Atomic Decompostion%%%%%%%%%%
for n=1:N
    x(n)=[e(n,1),e(n,2)]*conj(O);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%PCM Quanitzation%%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j); R=real(w(j));
%%%%%%%%%%%%%%%
if R>0
    Q(j)=1;
else
    Q(j)=-1;
end
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%%
A=e'*e; F=(conj(Q)*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2; end error(g)=mean(MSE);
end

X=3:g;
plot(X,error(X));
```

2-Bit PCM in $\mathbb{R}^2$

```
clear for g=2:64
for L=1:500
%%%%%%%%%Vector Creation%%%%%%%%%
test=1; while test==1
    O=[(2*rand-1); (2*rand-1)];
    if sqrt(O(1)^2+O(2)^2)<=1
        test=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%Frame%%%%%%%%%
N=g; for k=1:N
   e(k,1)=[cos(2*pi*(k-1)/N)];
   e(k,2)=[sin(2*pi*(k-1)/N)];
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%Atomic Decompostion%%%%%%%%%
for n=1:N
    x(n)=[e(n,1),e(n,2)]*conj(O);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%PCM Quanitzation%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j); R=real(w(j));
%%%%%%%%%%%%%%%
if R>0
    if R>1/2
        Q(j)=1;
    else
        Q(j)=1/2;
    end
else
    if R<-1/2
        Q(j)=-1;
    else
        Q(j)=-1/2;
    end
end
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%
A=e'*e; F=(conj(Q)*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2; end error(g)=mean(MSE);
end

X=3:g; plot(X,error(X));
```

1-Bit $\Sigma\Delta$ in $\mathbb{R}^2$

```
clear for g=2:64
for L=1:500
%%%%%%%%%%Vector Creation%%%%%%%%%%
test=1; while test==1
    O=[(2*rand-1); (2*rand-1)];
    if sqrt(O(1)^2+O(2)^2)<=1
        test=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Frame%%%%%%%%%%
N=g; for k=1:N
   e(k,1)=[cos(2*pi*(k-1)/N)];
   e(k,2)=[sin(2*pi*(k-1)/N)];
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Atomic Decompostion%%%%%%%%%%
for n=1:N
    x(n)=[e(n,1),e(n,2)]*conj(O);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Sigma-Delta Quanitzation%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j)+u(j); R=real(w(j));
%%%%%%%%%%%%%%%
if R>0
    Q(j)=1;
else
    Q(j)=-1;
end u(j+1)=u(j)+x(j)-Q(j);
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%
A=e'*e; F=(conj(Q)*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2; end error(g)=mean(MSE);
end

X=3:g; plot(X,error(X));
```

ΣΔ Box Quantization in ℂ

```
clear
for g=2:64
for L=1:500
%%%%%%%%%%Vector Creation%%%%%%%%%%
O=[(2*rand-1)+(2*rand-1)*i; (2*rand-1)+(2*rand-1)*i];
    x=real(O);
    y=imag(O);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Frame%%%%%%%%%%
N=g; for k=1:N
   e(k,1)=cos(2*pi*(k-1)/N);
   e(k,2)=sin(2*pi*(k-1)/N);
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Atomic Decompostion%%%%%%%%%%
x=(e*O)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Sigma-Delta Quanitzation%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j)+u(j); R=real(w(j));
I=imag(w(j));
%%%%%%%%%%%%%%%
if I>0
    if R>0
        Q(j)=1+i;
    else
        Q(j)=-1+i;
    end
else
    if R<0
        Q(j)=-1-i;
    else
        Q(j)=1-i;
    end
end u(j+1)=u(j)+x(j)-Q(j);
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%
A=e'*e; F=(Q*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2; end error(g)=mean(MSE);
end

X=3:g;
loglog(X,error(X),'s');
```

```
clear
for g=2:64
for L=1:500
%%%%%%%%%%Vector Creation%%%%%%%%%%
test=1; while test==1
    O=[(2*rand-1)+(2*rand-1)*i; (2*rand-1)+(2*rand-1)*i];
    x=real(O);
    y=imag(O);
    if abs(x)+abs(y)<=1
        test=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Frame%%%%%%%%%%
N=g; for k=1:N
   e(k,1)=cos(2*pi*(k-1)/N);
   e(k,2)=sin(2*pi*(k-1)/N)*i;
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Atomic Decompostion%%%%%%%%%%
x=(e*O)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Sigma-Delta Quanitzation%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j)+u(j); R=real(w(j));
I=imag(w(j));
%%%%%%%%%%%%%%%
if I>R
    if I>-R
        Q(j)=i;
    else
        Q(j)=-1;
    end
else
    if I>-R
        Q(j)=1;
    else
        Q(j)=-i;
    end
end u(j+1)=u(j)+x(j)-Q(j);
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%
A=e'*e; F=(Q*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2;
end
error(g)=mean(MSE);
end
```

```
X=3:g;
loglog(X,error(X),'d');
```

ΣΔ Hex Quantization in ℂ

```
clear
for g=2:64
for L=1:500
%%%%%%%%%%Vector Creation%%%%%%%%%%
test=1; while test==1
    O=[(2*rand-1)+(2*rand-1)*i; (2*rand-1)+(2*rand-1)*i];
    x=real(O);
    y=imag(O);
    if 1/sqrt(3)*(abs(y)+abs(sqrt(3)/2*x-1/2*y)+abs(-sqrt(3)/2*x-1/2*y))<=1
        test=0;
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Frame%%%%%%%%%%
N=g; for k=1:N
   e(k,1)=cos(2*pi*(k-1)/N);
   e(k,2)=sin(2*pi*(k-1)/N);
end
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Atomic Decompostion%%%%%%%%%%
x=(e*O)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Sigma-Delta Quanitzation%%%%%%%%%%
u=zeros(1,N+1); for j=1:N w(j)=x(j)+u(j); R=real(w(j));
I=imag(w(j));
%%%%%%%%%%%%%%%
if I>0
    if R>0
        if I>sqrt(3)/2
            if I>(-sqrt(3)*R+sqrt(3))
                Q(j)=1+sqrt(3)*i;
            else
                Q(j)=-1/2+sqrt(3)/2*i;
            end
        else
            if I>sqrt(3)*R
                Q(j)=-1/2+sqrt(3)/2*i;
            else
                Q(j)=1;
            end
        end
    else
        if I<-sqrt(3)*R-sqrt(3)
            Q(j)=-2;
        else
            Q(j)=-1/2+sqrt(3)/2*i;
        end
    end
else
    if R>0
```

```
        if I>-sqrt(3)/2
            if I<-sqrt(3)*R
                Q(j)=-1/2-sqrt(3)/2*i;
            else
                Q(j)=1;
            end
        else
            if I<sqrt(3)*R-sqrt(3)
                Q(j)=1-sqrt(3)*i;
            else
                Q(j)=-1/2-sqrt(3)/2*i;
            end
        end
    else
        if I>sqrt(3)*R+sqrt(3)
            Q(j)=-2;
        else
            Q(j)=-1/2-sqrt(3)/2*i;
        end
    end
end u(j+1)=u(j)+x(j)-Q(j);
%%%%%%%%%%%%%%%
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%Reconstruction%%%%%%%%%%
A=e'*e; F=(Q*e*A(1)^-1)';
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
MSE(L)=abs(O(1)-F(1))^2+abs(O(2)-F(2))^2;
end
error(g)=mean(MSE);
end

X=3:g;
loglog(X,error(X),'*');
```

Table 1: 1-Bit PCM: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.648528248		34	0.410736701
3	0.574001293		35	0.424579709
4	0.811630553		36	0.420722031
5	0.469701899		37	0.44557524
6	0.557813196		38	0.430658705
7	0.452630873		39	0.409684012
8	0.499225796		40	0.449015146
9	0.461099592		41	0.414092245
10	0.465944299		42	0.44046093
11	0.434443826		43	0.428079372
12	0.43078891		44	0.429065429
13	0.459050334		45	0.408205121
14	0.458178024		46	0.464250227
15	0.411389744		47	0.434710816
16	0.46250938		48	0.419387548
17	0.410761618		49	0.423828961
18	0.415023041		50	0.422846122
19	0.421012195		51	0.432714168
20	0.450910526		52	0.439917516
21	0.427359557		53	0.414665513
22	0.4446631		54	0.420230816
23	0.464722386		55	0.418325637
24	0.411485437		56	0.410625521
25	0.42283856		57	0.422219382
26	0.426195756		58	0.445510031
27	0.434408817		59	0.412881979
28	0.439197678		60	0.420506076
29	0.437540964		61	0.415907798
30	0.432011104		62	0.427941463
31	0.418296366		63	0.414117599
32	0.427217959		64	0.419153875
33	0.446251476			

Table 2: 2-Bit PCM: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.332805323		34	0.111697611
3	0.143970898		35	0.107671586
4	0.184823736		36	0.105298586
5	0.130257553		37	0.100403565
6	0.145783377		38	0.10443728
7	0.110815116		39	0.102429272
8	0.137809965		40	0.104313066
9	0.11068989		41	0.103228197
10	0.129049286		42	0.108014067
11	0.106920306		43	0.104945844
12	0.117585723		44	0.103220852
13	0.1052316		45	0.106511164
14	0.115839299		46	0.103231011
15	0.106435848		47	0.100578285
16	0.113321051		48	0.107613603
17	0.109250062		49	0.103715278
18	0.11079712		50	0.103045101
19	0.102299366		51	0.102554119
20	0.109022393		52	0.106547184
21	0.107940285		53	0.100768101
22	0.107214351		54	0.10497011
23	0.101572884		55	0.106246516
24	0.107043184		56	0.105752576
25	0.100394887		57	0.10295952
26	0.107899915		58	0.100862261
27	0.102431448		59	0.105740038
28	0.10419473		60	0.104266403
29	0.104687348		61	0.106932566
30	0.103667659		62	0.101012643
31	0.101687223		63	0.102886489
32	0.105911959		64	0.105937041
33	0.103155276			

Table 3: 3-Bit PCM: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.274722006		34	0.026243285
3	0.035740169		35	0.025599742
4	0.044928512		36	0.025710023
5	0.029569574		37	0.026401684
6	0.035171576		38	0.0258084
7	0.029041616		39	0.025385973
8	0.034422432		40	0.026592808
9	0.028765878		41	0.025903501
10	0.030843766		42	0.026401994
11	0.028082765		43	0.026006964
12	0.028870793		44	0.026080654
13	0.02701129		45	0.02630949
14	0.029133612		46	0.02623817
15	0.026465422		47	0.024909117
16	0.027871659		48	0.02571832
17	0.025747015		49	0.025547506
18	0.027924764		50	0.025828734
19	0.025785243		51	0.025070287
20	0.028267967		52	0.026221782
21	0.025609438		53	0.026040602
22	0.027822555		54	0.02567042
23	0.026287345		55	0.025514798
24	0.026885939		56	0.025930967
25	0.026250375		57	0.025939971
26	0.026463009		58	0.02653471
27	0.026630383		59	0.025810525
28	0.026800096		60	0.026465421
29	0.025954505		61	0.025732863
30	0.025978082		62	0.025998212
31	0.026301094		63	0.025649199
32	0.026456498		64	0.025895889
33	0.026337403			

Table 4: 1-Bit $\Sigma\Delta$: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.620337749		34	0.000627
3	1.481652393		35	0.004108
4	0.577886597		36	0.000582
5	0.435272281		37	0.003621
6	0.204452036		38	0.000489
7	0.171424772		39	0.00314
8	0.078983911		40	0.000462
9	0.086884557		41	0.002676
10	0.033713668		42	0.00032
11	0.05451842		43	0.002598
12	0.018475699		44	0.000298
13	0.039977654		45	0.002339
14	0.010789488		46	0.000259
15	0.027653237		47	0.002166
16	0.007856289		48	0.000233
17	0.019499681		49	0.001948
18	0.005407689		50	0.000225
19	0.016295705		51	0.001771
20	0.003889141		52	0.000183
21	0.012355784		53	0.001617
22	0.002565235		54	0.000141
23	0.010310468		55	0.001522
24	0.002034552		56	0.000132
25	0.00835463		57	0.001408
26	0.001701006		58	0.000126
27	0.006756405		59	0.001316
28	0.001353012		60	0.000125
29	0.00599411		61	0.001204
30	0.001130268		62	9.25E-05
31	0.005006218		63	0.001131
32	0.000897055		64	9.14E-05
33	0.004699891			

Table 5: $\Sigma\Delta$ Box Quantization in $\mathbb{C}$: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	1.338607906		34	0.00440137
3	2.75931564		35	0.01220089
4	0.990437065		36	0.004752211
5	0.954220783		37	0.010581858
6	0.382229778		38	0.004864637
7	0.363888721		39	0.009472614
8	0.181601927		40	0.003934411
9	0.2004324		41	0.00913089
10	0.089897283		42	0.00442896
11	0.131818469		43	0.008370071
12	0.047363899		44	0.004223266
13	0.08854045		45	0.009390505
14	0.028164598		46	0.003631785
15	0.067782779		47	0.007791177
16	0.021638835		48	0.003494275
17	0.048224397		49	0.008135321
18	0.01715245		50	0.002818651
19	0.039210666		51	0.007488333
20	0.013196406		52	0.003081322
21	0.030835791		53	0.006839986
22	0.009430021		54	0.003771295
23	0.02674824		55	0.006408415
24	0.008893955		56	0.003051065
25	0.022787516		57	0.006524748
26	0.008487247		58	0.003572032
27	0.018106343		59	0.007728195
28	0.006760919		60	0.003795593
29	0.015408719		61	0.005341693
30	0.005408784		62	0.003369402
31	0.01513251		63	0.00504539
32	0.006057833		64	0.003336606
33	0.013796187			

Table 6: $\Sigma\Delta$ Diamond Quantization in $\mathbb{C}$: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.631269255		34	0.002420049
3	1.363916459		35	0.006472463
4	0.499390981		36	0.002171014
5	0.44547029		37	0.005663609
6	0.201989346		38	0.002057077
7	0.182383143		39	0.004734409
8	0.100543653		40	0.001947249
9	0.10733345		41	0.004080599
10	0.048205278		42	0.002302635
11	0.067148864		43	0.005164168
12	0.021385435		44	0.001429697
13	0.045284008		45	0.003956065
14	0.013776672		46	0.002397833
15	0.032401088		47	0.004424651
16	0.010638195		48	0.001787132
17	0.022762885		49	0.00384094
18	0.008953481		50	0.001647673
19	0.018758099		51	0.003567393
20	0.00748326		52	0.001897542
21	0.016022281		53	0.003382196
22	0.004631731		54	0.001590657
23	0.013346379		55	0.003820959
24	0.004127449		56	0.001158107
25	0.011513521		57	0.003163394
26	0.00391933		58	0.001614724
27	0.010405501		59	0.002962785
28	0.004092723		60	0.00138221
29	0.008172954		61	0.002639189
30	0.002597238		62	0.001383741
31	0.007795052		63	0.002920447
32	0.002360181		64	0.001784979
33	0.006778393			

Table 7: $\Sigma\Delta$ Hexgon Quantization in $\mathbb{C}$: Mean Square Error

Frame Size	MSE		Frame Size	MSE
2	0.863553796		34	0.00516455
3	1.166452999		35	0.004860178
4	0.963260884		36	0.001428072
5	0.491057453		37	0.004446481
6	0.221938646		38	0.004156592
7	0.242062856		39	0.001150329
8	0.150115578		40	0.003663515
9	0.075851192		41	0.003312087
10	0.099516949		42	0.000946732
11	0.0751925		43	0.002996299
12	0.030222923		44	0.002905167
13	0.055425447		45	0.000773193
14	0.038257163		46	0.002744111
15	0.016001339		47	0.002513353
16	0.032351894		48	0.000648661
17	0.024958083		49	0.002319411
18	0.010118304		50	0.00222959
19	0.020327227		51	0.000564329
20	0.017593496		52	0.002088716
21	0.006575223		53	0.001914808
22	0.014955578		54	0.000497343
23	0.012772635		55	0.001711905
24	0.004342819		56	0.001717233
25	0.011301876		57	0.000377398
26	0.009690305		58	0.001675956
27	0.003217666		59	0.00156751
28	0.008315828		60	0.000425089
29	0.007411958		61	0.001467484
30	0.002539001		62	0.001403218
31	0.006668663		63	0.000374892
32	0.005768432		64	0.001313284
33	0.00184459			

REFERENCES

[1] John J. Benendetto, Alexander M. Powell, and Özgür Yilmaz, "Sigma-Delta ($\Sigma\Delta$) Quantization and Finite Frames," IEEE, to appear

[2] Mark Lammers, Alexander M. Powell, and Özgür Yilmaz, "Optimal Alternate Dual Frames for Digital to Analog Conversion," preprint

[3] Ingrid Daubechies and Ron DeVore, "Approximating a bandlimited function using very coarsely quantized data: A family of stable sigma-delta modulators of arbitrary order," *Annals of Mathematics*, 158 (2003), pp. 679-710

[4] J.H. Conway and N.J.A. Sloane, *Sphere Packings, Lattices, and Groups*, New York: Springer-Verlag, 1988, pp. 56-61

[5] Antti-Veikko Rosti. Tampere University of Technology. Jun 22 07:50:46 BST 2004. Mar 24 2006. <http://www.cs.tut.fi/sgn/arg/rosti/1-bit/>

[6] David Jimenez, Long Wang, and Yang Wang, "PCM Quantization Errors and The White Noise Hypothesis," preprint