AN ANALYSIS OF OSCILLATOR-BASED COMPUTATIONS FOR IMAGE PROCESSING

by

Brandon Bernard Jennings

B.S. Computer Engineering, University of Maryland, Baltimore County, 2013

Submitted to the Graduate Faculty of

Swanson School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science

University of Pittsburgh

2016

UNIVERSITY OF PITTSBURGH

SWANSON SCHOOL OF ENGINEERING

This thesis was presented

by

Brandon Bernard Jennings

It was defended on

April 26, 2016

and approved by

Amro El-Jaroudi, Ph.D, Associate Professor, Department of Electrical and Computer Engineering

Yrian Chen, Ph.D, Associate Professor, Department of Electrical and Computer Engineering

Bruce R. Childers, Ph.D, Professor, Department of Computer Science

Thesis Advisor: Donald M. Chiarulli, Ph.D, Professor, Department of Computer Science

Copyright © by Brandon Bernard Jennings

2016

AN ANALYSIS OF OSCILLATOR-BASED COMPUTATIONS FOR IMAGE PROCESSING

Brandon Bernard Jennings, M.S.

University of Pittsburgh, 2016

Although there has been a vast amount of research into improving CMOS technology and computer architecture to make more powerful and efficient systems, the trends of decreasing sizes and energy and increasing power and speed are plateauing. Major roadblocks hindering the progression of Boolean logic based computing are transistor size, heat dissipation, clock speed, and computation power. This has inspired investigation into new methods for performing, complex operations not based on logic gates, or non-Boolean computations.

One such method is coupled oscillator arrays. Instead of a logic gates to compute complex functions, the intrinsic physical properties of the oscillators can be used for computation making them more efficient for non-Boolean computations. This thesis will explore the use of coupled oscillator arrays to perform convolution, a primitive operation that plays a central role in many signal and image processing algorithms. Real-world circuit model parameters will be discussed and their impact on the circuit will be analyzed. In addition, this thesis will show the use of oscillators in Degree of Match (template matching), discrete cosine transform, discrete Fourier transform, Gabor filtering, and image compression. The effects of the model parameters on the will be examined on these implementations.

TABLE OF CONTENTS

LIST OF FIGURES

PREFACE

I would like to thank my advisors Dr. Steven P. Levitan and Dr. Donald M. Chiarulli for supporting my work and challenging me to work harder and think creatively. I would like to also thank Dr. Sylvanus N. Wosu for his guidance and mentorship throughout my graduate studies. Lastly, I would like to thank my family and friends for their encouragement and emotional support.

1.0 INTRODUCTION

Complementary metal–oxide–semiconductor (CMOS) is the current standard technology used in designing integrated circuits, composed of tiny devices known as transistors. These transistors dictate the flow of electricity through circuits. This current control property of the transistors enables the design of logic gates such as AND, OR, and inverter gates to perform Boolean algebra. In Boolean algebra, the values are denoted by 1's and 0's represented by the on (1) and off (0) switching of the transistors. Multiple and a wide variety of logic gates can be arranged in different fashions to perform different kinds of operations. This is essentially the framework of modern computers.

Although there has been a vast amount of research into improving CMOS technology and computer architecture to make more powerful and efficient systems, the trends of decreasing sizes and energy and increasing power and speed are plateauing. Major roadblocks hindering the progression of Boolean logic based computing are transistor size, heat dissipation, clock speed, and computation power. This has inspired investigation into new methods for performing complex operations not based on logic gates, also called non-Boolean computations.

One such method is coupled oscillator arrays. Instead of a logic gates to compute complex functions, the intrinsic physical properties of the coupled oscillators can be used for computation. This thesis will explore the use of coupled oscillator arrays to perform convolution, a primitive operation that plays a central role in many signal and image processing algorithms. Real-world circuit model parameters will be discussed and their impact on the circuit will be analyzed. In addition, this thesis will show the use of coupled oscillator arrays in Degree of Match (template matching), discrete cosine transform (DCT), discrete Fourier transform (DFT), Gabor filtering, and image compression. The effects of the model parameters on the will be examined on these implementations.

1.1 GOALS

The first goal of this thesis is to validate the feasibility of performing mathematical convolution using coupled oscillator arrays. This will be accomplished by exhaustive analysis of the parameters of coupled oscillator arrays and observing how they impact the fidelity of the model. The second goal of this master's thesis is to demonstrate the use of oscillator-based convolution in image processing functions, such as image filtering, discrete cosine transform, image compression, and discrete Fourier transform. The parameters of the oscillator model and their effect on the accuracy of the image processing functions will be analyzed.

1.2 STATEMENT OF WORK

The following steps were taken to accomplish the goals mentioned in the previous section:

1. Show that an oscillator-based Degree of Match (DOM) circuit can be characterized by a squared Euclidean distance equation, and therefore used for template matching.
2. Using this characterization, build a real-world circuit model with parameters for coupling asymmetry, locking region, input noise, and output noise.
3. Define how the DOM model can be used to compute convolution.
4. Using convolution, show how oscillator-based DCT can be implemented and how the convolution can be further optimized to improve design and mitigate error.
5. Using convolution, show how oscillator-based DFT can be implemented and how the convolution can be further optimized to improve design and mitigate error.
6. Analyze the impact of the parameter variation on Degree of Match using randomly generated input vectors of various sizes and test over a range of parameter values.
7. Analyze the impact of the parameter variation on convolution using randomly generated input vectors of various sizes and test over a range of parameter values.
8. Show how convolution can be used to implement Gabor filtering and analyze the impact of the parameter variation on filtering a 120x120 greyscale image and test over a range of parameter values.
9. Analyze the impact of the parameter variation on DCT using randomly generated input vectors of various sizes and frequencies and test over a range of parameter values.
10. Show how DCT can be implemented in image compression and analyze the impact of the parameter variation on compressing a 256x256 greyscale image and test over a range of parameter values.

11. Analyze the impact of the parameter variation on DFT using randomly generated input vectors of various sizes and frequencies and test over a range of parameter values.

1.3 CONTRIBUTION

This thesis provides an exhaustive analysis demonstrating the use of coupled oscillators to perform non-Boolean operations such as Degree of Match (template matching), convolution, DCT, and DFT. It presents a thorough investigation of the circuit model parameters, showing that their impact on the circuit will vary depending not only on the parameters themselves but also on additional factors such as oscillator array size, input vectors values, and type of image processing primitive. This thesis is organized as follows: Section 2 gives the background, Section 3 describes the approach, Section 4 lays out the experimental design, Section 5 presents the results and analysis, and Section 6 lists the conclusion.

2.0 BACKGROUND

Inspired by the work of Christian Huygens [1], in which he observed the behavior of coupled pendulums spontaneously synchronizing, there has been much work in studying models of coupled oscillators. Oscillators have shown promise for use in various non-Boolean computations in different domains such as magnetic, electric, and biological. Horvath [2] models the interaction of spin torque coupled oscillators via their magnetic field and demonstrate the use of this dynamic in an edge detection application. Shibata [3] emulated the behavior of oscillators using a CMOS ring oscillators and supporting CMOS circuitry to produce associative memory function. Hoppensteadt [4] proposed that coupled microelectromechanical oscillators can be useful to efficiently process analog information and theorize that they can function as a neurocomputer having oscillatory autocorrelative associative memory.

Nikonov et al. [5] proposed using weakly coupled voltage controlled oscillators to approximate convolution. They demonstrate the use of oscillators through Gabor filtering. Unlike most of the previous schemes based on Phase-Shift Keying, this work focuses on Frequency-Shift Keying, where the frequency shifts of the oscillators are used to encode image. The authors show that coupled oscillator arrays can be used for an approximation of convolution and that this analog method is a more energy efficient alternative to the digital algorithm.

Building on this work, Chiarulli et al. [6] demonstrated using oscillators to perform an exact convolution based on a coupled oscillator degree of match (DOM) metric. Simulation of the DOM circuit showed that the behavior of the coupled oscillators was similar to that of a squared Euclidean distance metric (L_2^2), which alone can be a computational primitive for template matching and distance metrics, like the ones used in image processing pipelines such as HMAX [7]. But as this thesis shows, it can also be used to directly compute convolution. Parameters were added to the model to analyze real-world circuit characteristics: the relative strength of each oscillator to other oscillators in the cluster (coupling asymmetry), the range of frequencies over which the oscillators synchronize and lock (locking region), noise on the two input channels to the oscillators, and noise on the DOM circuit output. It was shown the circuit has a relatively strong robustness to the parameters and their impact on the circuit vary depending on size of the array and DOM values between the two input vectors.

Part of this thesis will also explore other convolution-based primitives. Gabor filters [8][9] are linear filters resembling the human visual system used for feature extraction in image processing, most commonly used for edge detection, by convolving the filters with an image. The discrete cosine transform [10] is an extremely useful transform that is used in many applications such as image processing, as it segregates the low and high frequencies. It is a Fourier analysis that performs a convolution operation on a discrete set of data points and a series of cosine functions that oscillate at increasing frequencies. An important use of DCT is in lossy image compression [11], where sections of an image are transformed so that less important information can be removed in the frequency domain which reduces the size of the file. DFT is similar to the DCT, but instead of cosines it utilizes complex sinusoids [12]. There is current work that seeks to improve algorithms and hardware implementations of these primitives [13] – [17], however oscillators show a potential to be an efficient method to compute them.

This thesis extends the work from [6] even further by exploring the use of the DOM circuit to compute not only standalone convolution, but convolution involved in primitives such as Gabor filtering, discrete cosine transform, image compression, and discrete Fourier transform. The same parameters are swept and an in-depth analysis is done.

3.0 APPROACH

In this section, the approach will begin with the characterization of the DOM circuit. Next, the parameterized model will be discussed. Then, convolution will be defined given the DOM model. A discussion of oscillator-based discrete cosine transform will follow. Finally, there is a discussion of oscillator-based discrete Fourier transform.

3.1 DEGREE OF MATCH

The coupled oscillator array model used in this thesis is based on an array of electrically coupled nano-oscillators, depicted in Figure 1[6]. The input to this circuit is two vectors of analog (pixel) voltages ($v_1 \ldots v_n$) and ($v_1' \ldots v_n'$). The oscillators are simple 2-port voltage controlled oscillators with an input control port and bidirectional-output coupling port. Each oscillator is driven by the pairwise difference of the individual voltages ($v_i' - v_i$). The oscillator outputs are directly coupled though a resistor network and the voltage at the common node is integrated as a measure of the relative synchronization of the oscillators and hence the degree of match of the input vectors.

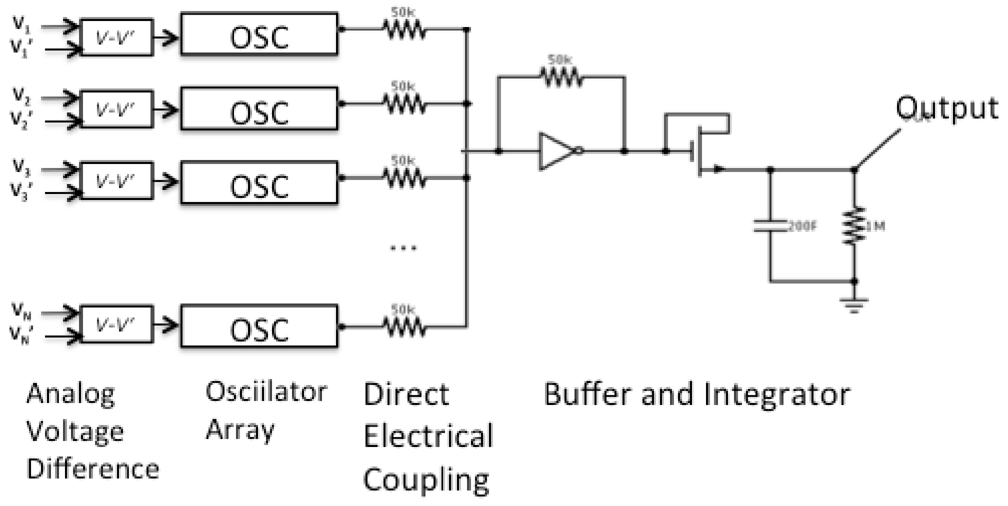


Figure 1: DOM Circuit Model

In order to observe the behavior of the oscillators, a simulation was conducted of a cluster of three oscillators. Two of the oscillators were held at a voltage such that their frequency matched the synchronization frequency *F*, while the third oscillator's input is swept. The blue dots are subtracted from the maximum output to give inverted simulation results from the DOM circuit output. As the input of the third oscillators swept, the oscillators begin to synchronize and then unsynchronize, causing the parabolic shape on the DOM output shown in Figure 2[6].

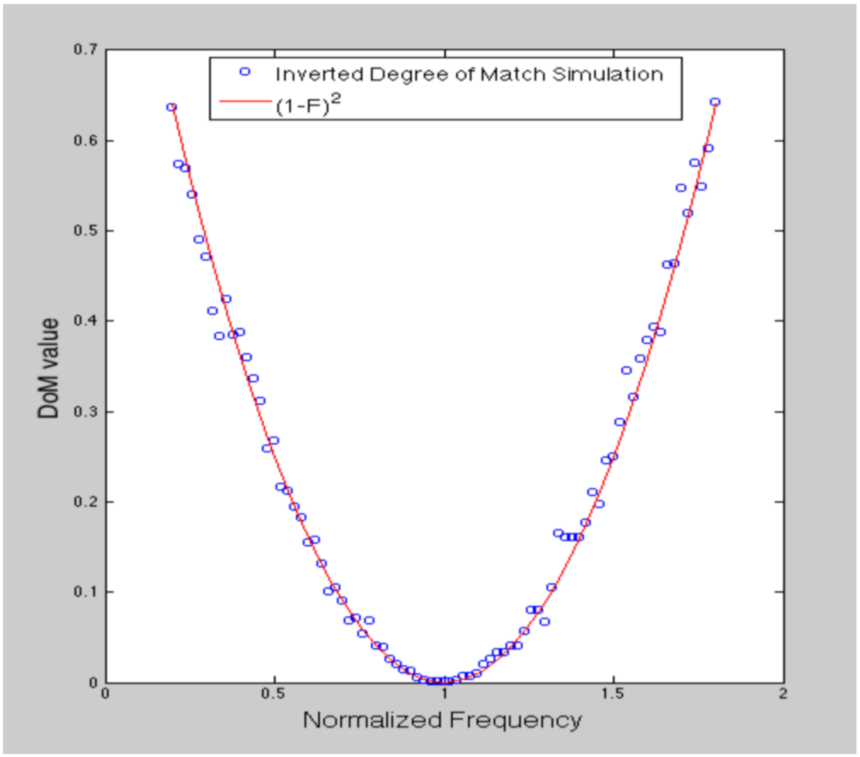


Figure 2: Simulation of 3-oscillator array, with two oscillators synchronized while sweeping the third.

Fitting a curve to the data yields a polynomial equation similar to x^2, where x is a vector of the differences between the two vectors. Thus, the output of the circuit can be modeled as the squared Euclidean distance metric:

$$\boldsymbol{L_2(A,B)} = \sqrt{\boldsymbol{(a_1\text{-}b_1)^2 + \cdots + (a_n\text{-}b_n)^2}} = \sqrt{\sum_{i=1}^{n} \boldsymbol{(A\text{-}B)^2}}$$

Equation 1: Euclidean distance metric

$$\boldsymbol{DOM(A,B) = L_2{}^2(A,B) = (a_1\text{-}b_1)^2 + \cdots + (a_n\text{-}b_n)^2} = \sum_{i=1}^{n} \boldsymbol{(A\text{-}B)^2}$$

Equation 2: Degree of match with oscillators.

The Euclidean distance metric is a commonly used method for template matching as it quantifies how similar two vectors (images) are. Thus, the DOM model can be used as a distance metric for template matching.

3.2 PARAMETERIZED DOM MODEL

Beginning with Equation 2 on page 10, this provides a base case that assumes ideal conditions. However, there are real-world parameters that can have an impact on the DOM circuit. Given a particular circuit and its implementation, these circuit parameters can cause variability in the output of the circuit. There are four parameters modeled and analyzed in this thesis: coupling asymmetry (CA), locking region (LR), input noise (IN), and output noise (ON). Figure 3 is a simulation of the DOM model with the parameters.

1: Coupling Asymmetry (*CA*)

$$\boldsymbol{DOM(A,B)} = \sum_{i=1}^{n} \boldsymbol{ca_i} \times (\boldsymbol{a_i} - \boldsymbol{b_i})^2$$

2: Locking Region (*LR*)

$$\boldsymbol{DOM(A,B)} = \sum_{i=1}^{n} (\boldsymbol{a_i} - \boldsymbol{b_i})^2 ,$$

$$\textit{\textbf{when}}\ \boldsymbol{DOM} < \boldsymbol{LR} \rightarrow \boldsymbol{DOM} = \boldsymbol{0}$$

3: Input Noise (*IN*)

$$\boldsymbol{DOM(A,B)} = \sum_{i=1}^{n} \left((\boldsymbol{a_i} + \boldsymbol{n^A}_i) - (\boldsymbol{b_i} + \boldsymbol{n^B}_i) \right)^2$$

4: Output Noise (*ON*)

$$\boldsymbol{DOM(A,B)} = \sum_{i=1}^{n} (\boldsymbol{a_i} - \boldsymbol{b_i})^2 + \boldsymbol{N_o}$$

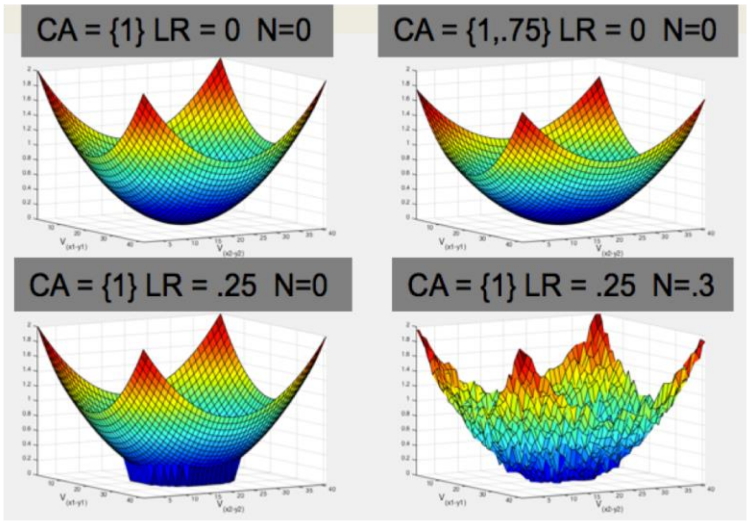


Figure 3: Simulation of 3-oscillator array with parameters.

Coupling asymmetry (CA) captures the difference in coupling strength between the individual oscillators. Conditions can exist where not all of the oscillators in a cluster have the same relative strength; one oscillator may be stronger than another by a fraction. The CA parameter models this variance in the relative strength of the oscillators as an array of coupling asymmetry coefficients.

Locking region (LR) represents the range of the frequencies over which a cluster of oscillators will synchronize. In ideal conditions, the oscillators will synchronize at a single frequency as shown in the Figure 2. However, based on the strength of the coupling network (the resistors connecting the oscillators depicted in the Figure 1) the oscillators will synchronize over a small range of frequencies. Larger resistors mean smaller amounts of current, making the oscillator less likely to couple. Conversely, smaller resistors yield larger amounts of current, making the oscillator more likely to couple. Oscillators in a cluster exhibiting frequency values within the locking region would synchronize since the input values become indistinguishable on the output of the circuit. The *LR* parameter models this locking region.

Input and output noise are common in any circuit, causing variation in the circuit behavior. Random noise can appear on any input or output channel. To model this Gaussian white random noise was added to the channels. Taking these parameters into consideration, the new parameterized DOM becomes

$$\boldsymbol{DOM}(\boldsymbol{A},\boldsymbol{B}) = \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \boldsymbol{ca}_{\boldsymbol{i}} \times \left((\boldsymbol{a}_{\boldsymbol{i}} + \boldsymbol{n^A}_{\boldsymbol{i}})\text{-}(\boldsymbol{b}_{\boldsymbol{i}} + \boldsymbol{n^B}_{\boldsymbol{i}})\right)^2 + \boldsymbol{N_O}$$

$$\boldsymbol{when\ DoM < LR \ \rightarrow DoM = 0}$$

Equation 3: Parameterized DOM model

where ca_i is a value from a coupling asymmetry vector, $n^A{}_i$ and $n^B{}_i$ are elements from an input noise vector added to input vectors A and B respectively, N_O is output noise and LR is locking region. For future reference, CA refers to coupling asymmetry, LR refers to locking region, IN refers to input noise, and ON refers to output noise.

3.3 CONVOLUTION

In this section, convolution will be defined in terms of the DOM model. The first subsection is an overview of convolution. The second subsection describes oscillator based convolution.

3.3.1 Convolution

Convolution is a widely used mathematical operation that is merely a summation of the dot products of two vectors. It is expressed by the generic equation

$$\boldsymbol{c = \sum_{i=1}^{n} a_i \times b_i}$$

Equation 4: 1-D Convolution

where a_i and b_i are elements of input vectors A and B of size n. Convolution iteratively pairwise multiplies the values of the input vectors indexed at i and then sums the products. In some signal processing applications, input vector B is inverted though it is not the case for image processing. An example is shown in Figure 4.

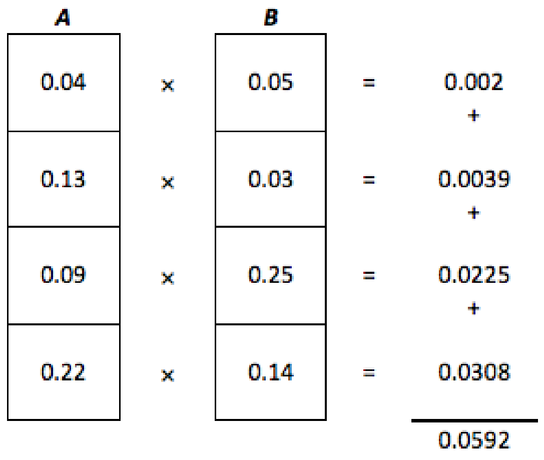


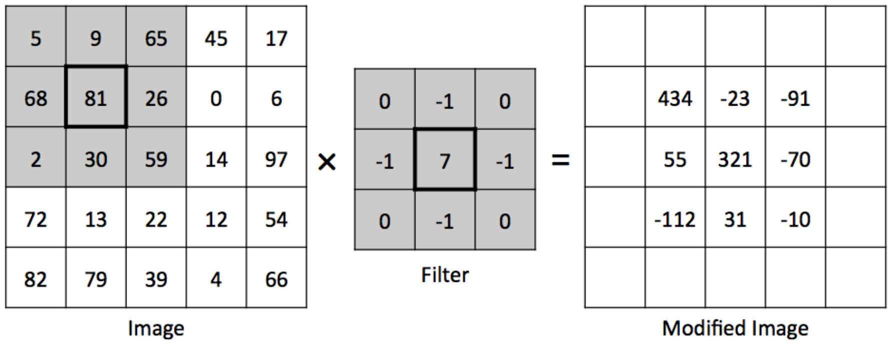
Figure 4: Example of 1-D convolution

Convolution operations are used in many different fields such as statistics, differential equations, image and signal processing and detection. A common example of convolution in image processing is filtering. Filtering, in this context, is a two dimensional convolution operation between an input array and a special type of kernel chosen depending on the kind of filtering to be performed (edge detection, sharpening, blurring, etc.). It can be expressed by

$$c = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{i,j} \times b_{i,j}$$

Equation 5: 2-D convolution.

where $a_{i,j}$ and $b_{i,j}$ are elements of input arrays A and B of size $m \times n$. The filter kernel is laid on top of the image such that the middle of the filter is centered upon a particular pixel. Each value of the filter is then multiplied with its corresponding value in the image and summed together to result in a new pixel value. Figure 5 shows an example of two dimensional filtering. Though this example fits the filter within the bounds of the image, there are various techniques one can use to handle edge cases, where the filter is centered such that the edge of the filter is off the image.



5	9	65	45	17
68	81	26	0	6
2	30	59	14	97
72	13	22	12	54
82	79	39	4	66

Image

×

0	-1	0
-1	7	-1
0	-1	0

Filter

=

	434	-23	-91	
	55	321	-70	
	-112	31	-10	

Modified Image

Figure 5: Example of 2-D convolution.

3.3.2 Oscillator-Based Convolution

Equation 6 is an algebraic expansion of the DOM model. There are two observations. The first is that there are two DOM terms in the form of $\sum A^2$ and $\sum B^2$, which are the equivalent of two DOM models where one accepts A and a zero vector as its inputs and the other one accepts B

and a zero vector. The second observation is that a convolution of input vectors A and B is present in the form of $2\sum AB$.

$$\boldsymbol{DOM(A,B)} = \sum \boldsymbol{(A - B)^2} = \sum \boldsymbol{(A^2 - 2AB + B^2)} = \sum \boldsymbol{A^2} - \boldsymbol{2} \sum \boldsymbol{AB} + \sum \boldsymbol{B^2}$$
$$= \boldsymbol{DOM(A,0) - 2{*}conv(A,B) + DOM(B,0)}$$

Equation 6: Expanded DOM equation.

Therefore, convolution can be computed using three oscillator clusters. In addition to an oscillator cluster that takes A and B as inputs to compute DOM(A, B), two more clusters are needed that only take either A or B and a zero vector as inputs, to compute DOM(A, 0) and DOM(B, 0). Hence, the convolution becomes

$$\boldsymbol{conv(A,B)} = \frac{\boldsymbol{DOM(A,B) - DOM(A,0) - DOM(B,0)}}{\boldsymbol{-2}}$$

Equation 7: Convolution with oscillators.

Unlike the previous work in [2], which only approximated convolution, this convolution is precise provided that the DOM model fits to a squared Euclidean distance metric.

3.4 DISCRETE COSINE TRANSFORM

This section describes oscillator-based DCT. The first section discusses how oscillators can be used to compute DCT. The second section discusses how to improve the computation.

3.4.1 Oscillator-Based DCT

The discrete cosine transform is an extremely useful transform that is used in many applications such as audio and image processing. It is a Fourier analysis that performs a convolution operation on a discrete set of data points and a series of cosine functions that oscillate at increasing frequencies. It is expressed as

$$\boldsymbol{y(k) = w(k) \sum_{n=1}^{N} x(n)\, cos\left(\frac{\pi}{2N}(2n - 1)(k - 1)\right), \qquad k = 1, 2, \ldots, N}$$

Equation 8: DCT convolution.

where

$$\boldsymbol{w(k) = \begin{cases} \dfrac{1}{\sqrt{N}}, & k = 1 \\ \sqrt{\dfrac{2}{N}}, & 2 \le k \le N \end{cases}}$$

and where *y* is a vector of same length as *x.* Because oscillators can be used to perform convolution, then by extension they can be used to perform discrete cosine transforms, with an input vector and and computer cosines.

As shown in Equation 8, the DCT involves multiple convolutions. In fact, on a one dimensional vector, a DCT requires n convolutions, where n is the size of the transform vector. Each of the n elements in output vector *y* is a convolution of input vector *x* and different vectors of cosine values. To design a full hardware DCT implementation, 3n oscillator clusters of size n are required, for a total of $3n^2$ oscillators. However, the next section describes a way to significantly reduce this number.

3.4.2 DCT Optimization

There are a couple of ways oscillator-based DCT can be improved. First, mathematically the DOM(0, B) = 1 because each of the cosine terms is multiplied by *w(k)* and then squared before summed. Since it is a constant, this term can be removed and replaced with a pre-computed value as shown in Equation 9, which will in turn reduce the amount of error in the convolution and the number of oscillators required is reduced to $2n^2$ since only two clusters are required.

$$\boldsymbol{conv(A,B)_{dct} = \frac{DOM(A,B) - DOM(A,0) - 1}{-2}}$$

Equation 9: Oscillator-based DCT convolution.

Next, because input vector *A* never changes, the DOM(A, 0) term can be reused in each of the convolutions. Since there is only one DOM(A, 0) cluster, this further reduces the number of oscillators to $n^2 + n$. This is a significant decrease in size from the original $3n^2$.

3.5 DISCRETE FOURIER TRANSFORM

This section describes oscillator-based DFT. The first section discusses how oscillators can be used to compute DFT. The second section discusses how to improve the efficiency of the computation.

3.5.1 Oscillator-Based DFT

The DFT, similar to the DCT, characterizes data in the frequency domain, however the DFT represents the data using complex sinusoids of various frequencies. It is characterized as

$$\boldsymbol{y(k) = \sum_{j=1}^{N} x(j) W_n{}^{(j-1)/(k-1)}, \qquad k = 1, 2, \ldots, N}$$

Equation 10: DFT convolution

where

$$\boldsymbol{W_n = e^{(-2\pi i)/n}}$$

Oscillators can be used to perform convolution between a dynamic input vector and pre-computed complex sinusoids. Because the DFT involves complex sinusoids, the complex terms need to be decomposed into real and imaginary components, treated in separate convolutions, then constructed back into a complex term, shown in Equation 11

$$\boldsymbol{conv(A,B) = conv(A_{real}, B_{real}) - conv(A_{imag}, B_{imag})}$$
$$\boldsymbol{+ j[conv(A_{real}, B_{imag}) + conv(A_{imag}, B_{real})]}$$

Equation 11: Oscillator-based DFT convolution.

Just as in DCT, there are n convolutions in one DFT operation, each of size n. Because of the complex values in the DFT, more oscillator clusters are needed since each convolution requires four pairwise convolutions, introducing more error in the computations and requiring a greater number of oscillators. Using four convolutions, the DFT needs 12 oscillator clusters per convolution, for a total of 12n clusters for a DFT operation and 12n^2 oscillators. However, this number can be further reduced.

3.5.2 DFT Optimization

Like DCT, the DFT convolution operation can be improved to reduce error and size. Typically, the input vector used with the DFT is a real signal and does not contain any imaginary components. This means that the convolutions involving the imaginary component of the input vector, considered input vector *A* for example, will be zero.

$$\boldsymbol{conv(A_{imag}, B_{imag}) = conv(A_{imag}, B_{real}) = 0}$$

Equation 12: *A* is not complex and has no imaginary parts.

Therefore, these two convolutions can be removed, resulting in Equation 13 which reduces the effects from parameter variations and the number of oscillators to 6n^2.

$$\boldsymbol{conv(A,B)_{dft} = conv(A_{real}, B_{real}) + conv(A_{real}, B_{imag})}$$

Equation 13: Optimized DFT equation, for when *A* is real.

For DFT, mathematically DOM(B_{real},0) and DOM(B_{imag},0) are also constants. The first frequency of B_{real} equal to the size of B and the others equal to half the size of B. The first frequency of B_{imag} is equal to 0 and the rest are equal to half the size of B. DOM(B_{real},0) and DOM($B_{imaginary}$,0) can be replaced with pre-computed values to further reduce the impact of parameters on the convolution.

$$\boldsymbol{conv(A_{real}, B_{real}) = \frac{DOM(A_{real}, B_{real}) - DOM(A_{real}) - DOM(\sum B_{real})}{-2}}$$

Equation 14: Optimized convolution between real parts of input vectors.

$$\boldsymbol{conv(A_{real}, B_{imag}) = \frac{DOM(A_{real}, B_{real}) - DOM(A_{real}) - DOM(\sum B_{imag})}{-2}}$$

Equation 15: Optimized convolution between real part of *A* and imaginary part of *B*.

Again, the same DOM(A, 0) cluster can be used in both convolutions since input vector A does not change. The total number of oscillators has now been reduced to $2n^2 + n$.

4.0 EXPERIMENTAL DESIGN

In this section the experimental design of the studies is discussed. All simulations were done in MATLAB. The first section defines the parameters and their values. The next section outlines the design of the template matching and convolution studies. The last section outlines the design of the DCT and DFT studies.

4.1 PARAMETERS AND OTHER STUDY CONFIGURATIONS

In this section, the model parameters are defined and the inputs and oscillator array sizes are determined.

4.1.1 Model Parameters

This section begins with a discussion of how the model parameters *CA*, *LR*, *IN*, and *ON* are encoded in the study simulations. In each case the magnitude of the parameter is expressed as a fraction p of the dynamic range of the input/output. The base case is when $p = 0$ and it ranges from 0 to 0.25. *CA* is a vector of the same size as input vectors *A* and *B* with values of $1 \pm p$, where the values are from a Gaussian distribution centered about 1. *LR* is a scalar ranging from 0 to p times the maximum output. *IN* is a vector of the same size as input vectors *A* and *B* with values 0 0±p, where the values are from a Gaussian distribution centered about 0. *ON* is a scalar ranging from 0 to p times the maximum output and is generated from a Gaussian distribution centered about 0.

4.1.2 Array Sizes and Inputs Tested

For all studies in this thesis, the oscillator array sizes used are squared values ranging from 4 to 169 and all input values range from 0 to 1, unless otherwise specified. This means that the maximum input value is 1 and the maximum output ranges from 4 to 169 (corresponding with each array size), since it is possible for every input difference is 1.

4.2 SIMULATION STUDY: DOM/CONVOLUTION

This section outlines the experimental design for DOM and convolution. The first subsection describes the input vectors generated for these studies. The second subsection discusses the error metric used. The same input vectors and error metric were used in both studies.

4.2.1 Input Generation

Since the input values range from 0 to 1, the DOM values for a given size range from 0 to N, the size of the array. To test the full range of possible DOM values between the input vectors, random input vector pairs were generated such that for each size there are 101 input vector pairs that have DOM values ranging from 0 to N. Each size is divided by 100 so that there is an equal number of input vectors generated for each size. This was accomplished by first generating a vector of squared difference values whose sum is the desired DOM value. In other words, a vector v is generated such that $v_i = (a_i - b_i)^2$ and $\sum v_i$ = DOM(A, B). This is done using a MATLAB function *randfixedsum(n, m, s, a, b), based on the Randfixedsum Algorithm* by Roger Stafford [18][19], that returns an $n \times m$ array where each m column has n random values on the interval $[a, b]$ that sum to s. This algorithm generates only the possible random values given the initial conditions.

With this vector v of squared difference terms, input vectors A and B can be derived. The square root of v_i is taken resulting in $\sqrt{v_i} = (a_i - b_i)$. The signs of these values ($\sqrt{v_i}$) are randomized (it is possible for $a_i > b_i$ or $a_i < b_i$) and values are assigned to a_i and b_i. If v_i is negative, then $b_i > a_i$. The value of b_i is then randomly chosen between $-v_i$ and 1 and a_i becomes $b_i + v_i$. If v_i is positive, then $b_i < a_i$. The value of b_i is then randomly chosen between 0 and $(1 - v_i)$ and a_i becomes $b_i + v_i$.

4.2.2 Output and Error Metric

The error metric used in these studies is a normalized root mean squared error that compares values computed from the base-case model to values computed from the parameterized model.

$$\boldsymbol{NRMSE} = \frac{\|\boldsymbol{xref}\text{-}\boldsymbol{x}\|}{\|\boldsymbol{xref}\text{-}\mathbf{mean}(\boldsymbol{xref})\|}$$

Equation 16: Normalized root mean square error.

where *xref* is a vector of DOM/convolution values computed from the base-case model, *x* is a vector of DOM/convolution values computed from oscillator-based DOM for various parameter values, and || indicates the 2-norm, or Euclidean distance. *NRMSE* was used in these studies because, given our input constraints, they test over the entire range of inputs. And because that range changes between sizes, the error needs to be normalized.

4.3 SIMULATION STUDY: DISCRETE COSINE TRANSFORM/DISCRETE FOURIER TRANSFORM

This section outlines the experimental design for DCT and DFT. The first subsection describes the input vectors generated for these studies. The second subsection discusses the error metric used. The same input vectors and error metric were used in both studies.

4.3.1 Input Generation

The DCT and DFT only take one input vector, as the other is predetermined because it is a vector of cosines/complex sinusoids of various frequencies. Similar to the DOM and convolution studies, multiple random input vectors of various sizes are generated. However, unlike the DOM and convolution studies, the input vectors for this study are not based on particular DOM values.

4.3.2 Output and Error Metric

The error metric used in these studies is a mean squared error that compares values computed from the base-case model to values computed from the parameterized model.

$$\boldsymbol{MSE} = \frac{\|\boldsymbol{x}\text{-}\boldsymbol{xref}\|}{\boldsymbol{N}}$$

Equation 17: Mean squared error.

where *xref* is a vector of DCT/DFT computed from the base-case model, *x* is a vector of DCT/DFT values computed from the parameterized model, *N* is the number of values, and || indicates the 2-norm, or Euclidean distance. *MSE* was used in this study each element of the DCT/DFT is a separate convolution and are not related to one another. The *MSE* is calculated for each of the DCT/DFT input vectors for each size and then the average of MSE values over all of the frequencies are taken. This was done so that a comparison can be made between different input vector sizes without having to consider a particular frequency, since different sizes have a different number of possible frequencies. It also mitigates the case where different frequencies could be affected differently by the parameters, though in practice it is not the case.

5.0 RESULTS AND ANALYSES

This section presents the results and analyses. It shows histograms and error plots to demonstrate the impact of the parameters on the accuracy of the circuit model performing DOM, convolution, DCT, and DFT, as well as Gabor filtering and image compression implementations. The subsections present results in the following order: DOM, convolution, Gabor filtering, DCT, image compression, and DFT.

5.1 DOM (TEMPLATE MATCHING)

In this section, the effects of the model parameters on DOM are analyzed. Each of the following subsections analyzes a different parameter in the following order: coupling asymmetry, locking region, input noise, and output noise.

5.1.1 Coupling Asymmetry

Figure 6 shows the response of the DOM circuit with varying *CA*. The plot shows the polynomial shape that characterizes the circuit. The variation in the general shape of the curve, specifically why it is stair-stepped as opposed to smooth, is caused because there are multiple possible combinations of values for both input vectors that correspond to a particular DOM value. The minimum and maximum points on the x-axis will always remain 0 and maximum DOM value, respectively, though the points in between can vary, causing the stair-stepped shape in the response.

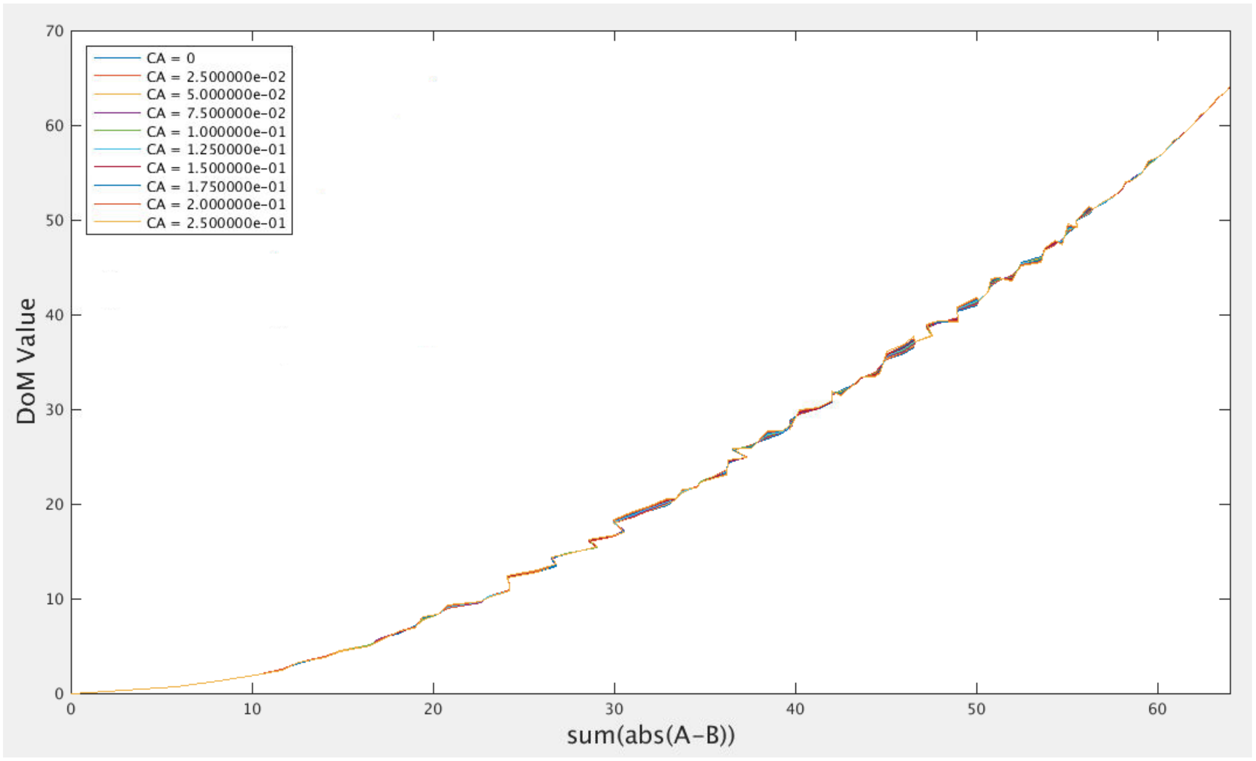


Figure 6: DOM(A, B) circuit response with *CA* variation.

There is also more variation for larger DOM values for particular *CA*. This is explained in Figure 7, which shows a histogram of the $(a_i - b_i)^2$ terms to which the *CA* parameter is applied according to Equation 3 for an array size of 64. Larger DOM values obviously have larger $(a_i - b_i)^2$ terms thus there is a larger difference between the base case values and the parameterized values. When DOM = 0, all of the $(a_i - b_i)^2$ terms are 0 and when DOM = 64 they are 1.

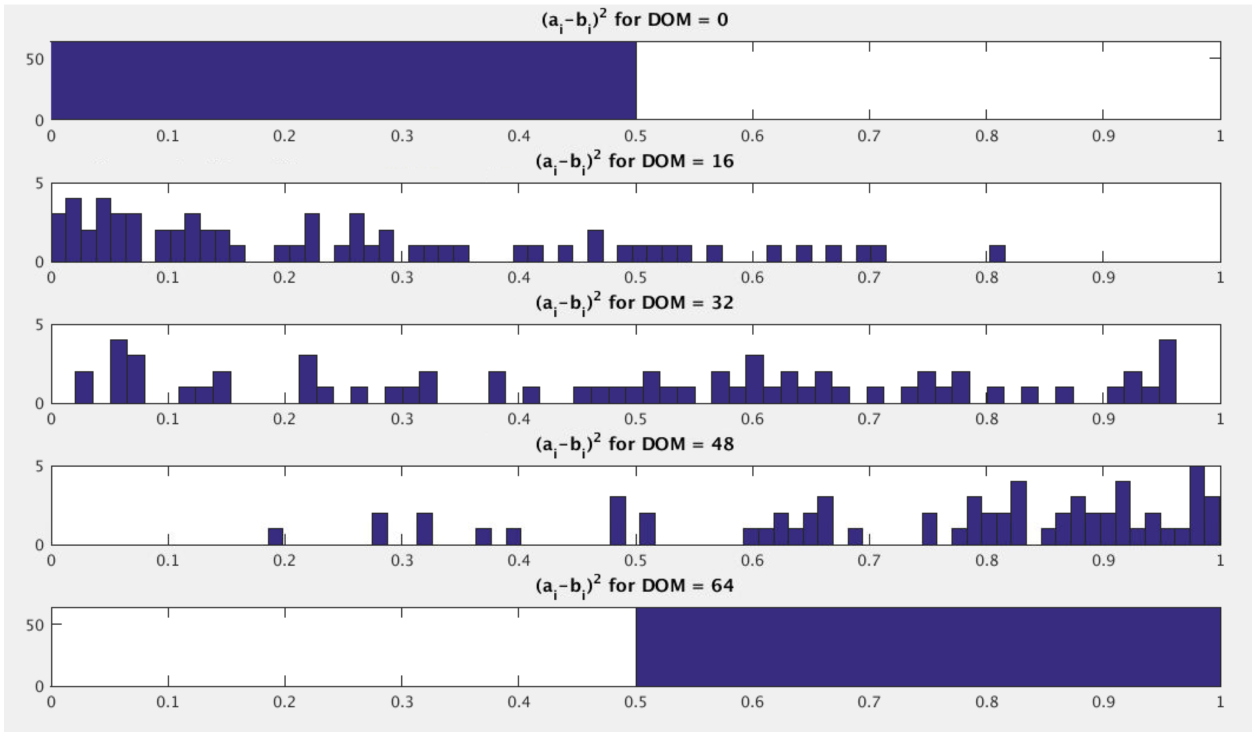


Figure 7: Histogram of $(a_i - b_i)^2$ values for particular DOM(A,B) values for an array size of 64.

Figure 8 shows the *NRMSE* of *CA* variations on DOM(A, B) for different oscillator array sizes. It is observed that the smaller oscillator array sizes are more sensitive than larger array sizes. This is because the average of the random *CA* approaches 1 as the number of oscillators increase. However, the *CA* variation has minor impact on the DOM output for all array sizes tested, with smaller sizes having error scores better than 0.2 in the extreme case where oscillators can experience a 25% relative strength.

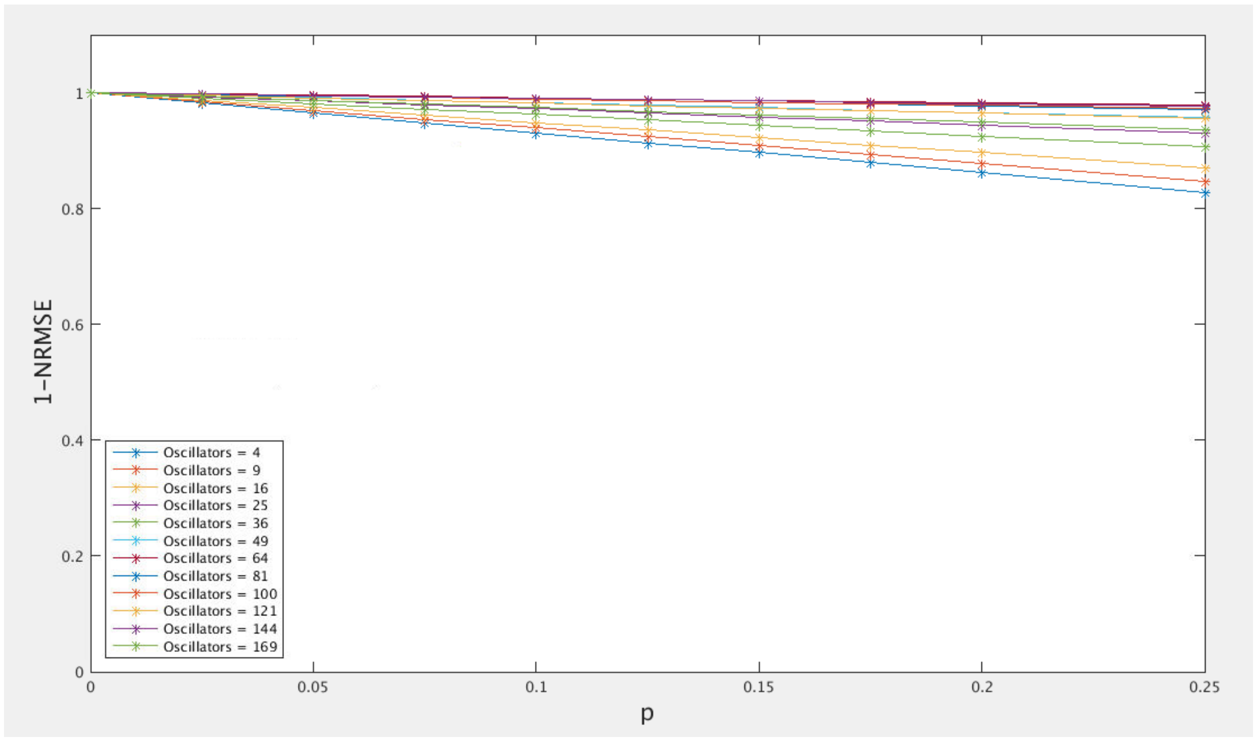


Figure 8: NRMSE of DOM(A, B) with *CA* variations for different array sizes.

5.1.2 Locking Region

Figure 9 shows the response of the DOM circuit for varying *LR*. When a DOM value is determined to be below the threshold *LR*, it is considered a match, or 0. As the locking region parameter increases, it results a step shape seen in the response as more values are set to 0.

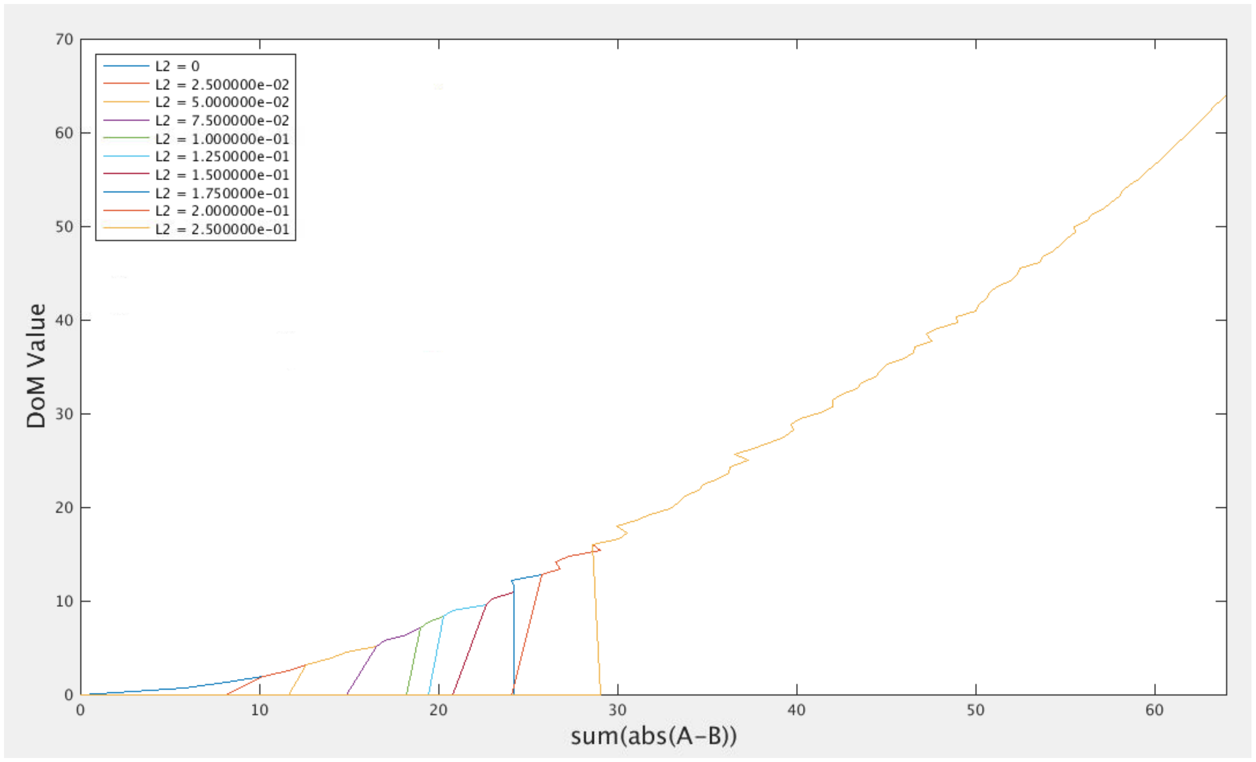


Figure 9: DOM(A, B) circuit response with *LR* variation.

Figure 10 shows the *NRMSE* of *LR* variation on DOM(A, B) for different oscillator array sizes. There is virtually no difference in the amount of error between array sizes. This is because *LR* is relevant outside of the summation of the convolution and its value is proportional to the size of the array. Each array sizes loses the same number of values, in this case 25%.

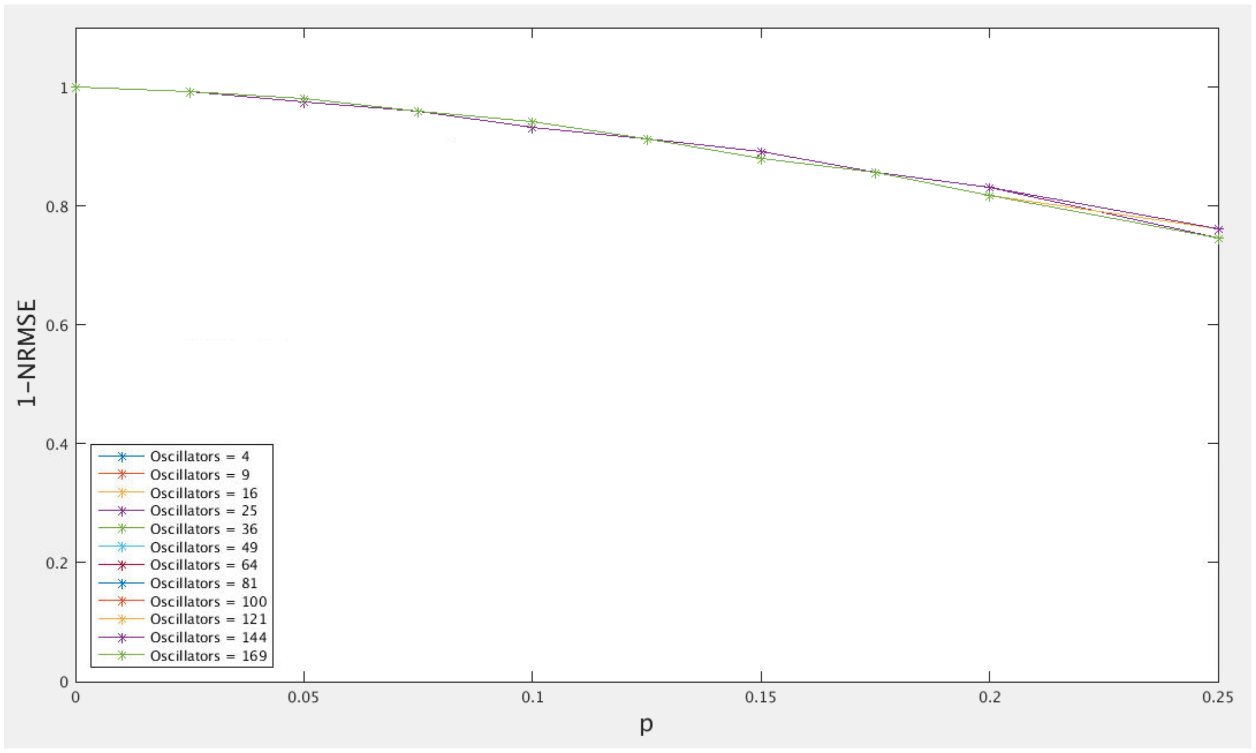


Figure 10: NRMSE of DOM(A, B) with *LR* variations for different array sizes.

5.1.3 Input Noise

Figure 11 shows the response of the DOM circuit for varying *IN*. Like the *CA* response in Figure 6, the response shows more variation for larger DOM values and similarly, as the $(a_i - b_i)$ terms increase for the same input noise value, there is a larger difference between the outputs from the base case model and the parameterized model.

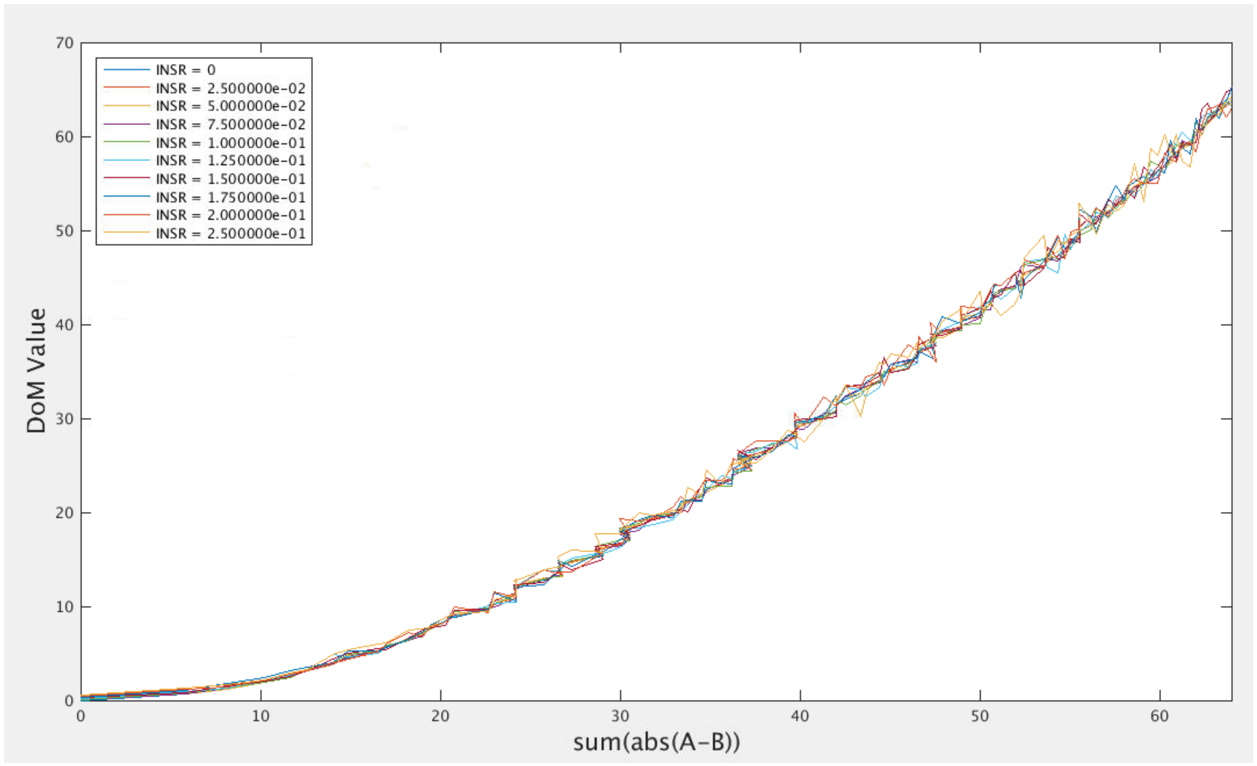


Figure 11: DOM(A, B) circuit response with *IN* variation.

Considering the DOM model with only the input noise parameter, Equation 3 becomes

$$\boldsymbol{DOM}(\boldsymbol{A},\boldsymbol{B}) = ([\boldsymbol{A}+\boldsymbol{N_A}]\text{-}[\boldsymbol{B}+\boldsymbol{N_B}])^{\mathbf{2}}$$

Equation 18: DOM with input noise.

Expanding this new equation gives

$$\boldsymbol{DOM}(\boldsymbol{A}\text{-}\boldsymbol{B}) = (\boldsymbol{A}\text{-}\boldsymbol{B}+\boldsymbol{N_A}\text{-}\boldsymbol{N_B})^{\mathbf{2}} = \left((\boldsymbol{A}\text{-}\boldsymbol{B})\text{-}\boldsymbol{N_{A\text{-}B}}\right)^{\mathbf{2}} = (\boldsymbol{A}\text{-}\boldsymbol{B})^{\mathbf{2}}\text{-}\mathbf{2}\boldsymbol{N_{A\text{-}B}}(\boldsymbol{A}\text{-}\boldsymbol{B})+\boldsymbol{N_{A\text{-}B}}^{\mathbf{2}}$$

Equation 19: Expanded DOM with input noise.

where $N_{A\text{-}B}$ is the total added noise $N_A - N_B$. The range of values presented by $N_{A\text{-}B}$ increases as $(a_i - b_i)$ increases.

Figure 12 shows the *NRMSE* of *IN variations* on DOM(A, B) for different oscillator array sizes. It shows that similarly to *CA,* smaller array sizes are more sensitive to input noise than the larger array sizes because as the array size increases, the average of the input noise approaches 0. However, *IN* variation has slightly more impact on the DOM circuit output than *CA* variation.

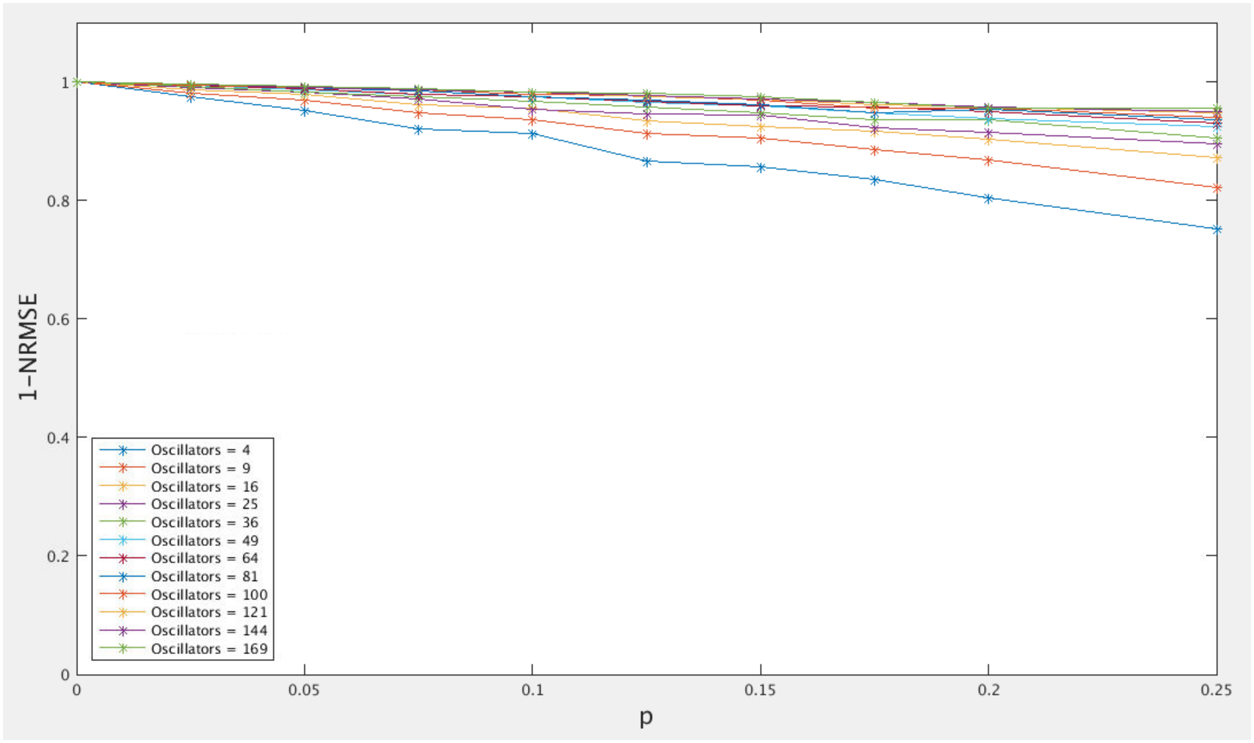


Figure 12: NRMSE of DOM(A, B) with *IN* variations for different array sizes.

5.1.4 Output Noise

Figure 13 shows the response of the DOM circuit for varying *ON*. Output noise inflicts greater variation on the DOM output than input noise because it is added at the end of the summation and is a percentage of the maximum DOM value.

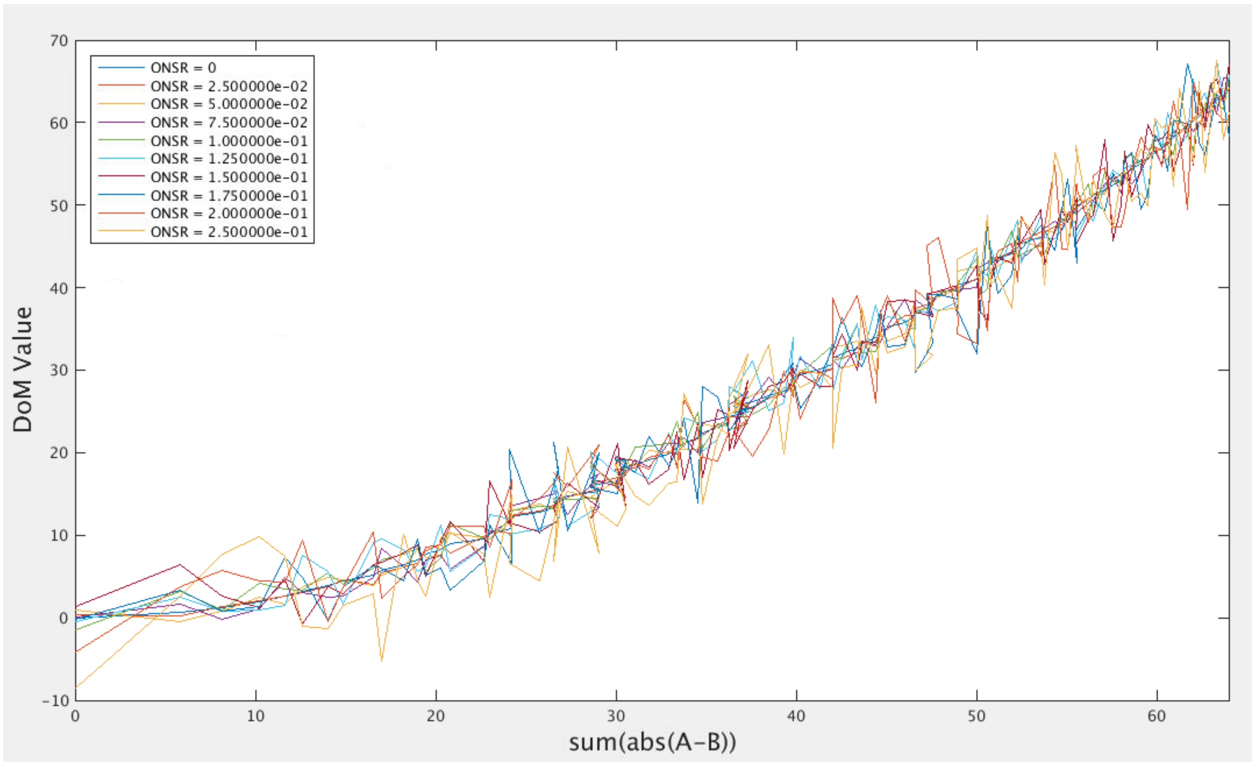


Figure 13: DOM circuit response with *ON* variation.

Figure 14 shows the *NRMSE* of *ON* variation on DOM for different oscillator array sizes. Because the noise is a percentage of the maximum output, which is proportional to array size, different array sizes experience relatively the same amount of error. Overall, for all sizes tested it has a greater impact than *IN* because it is a fraction of the dynamic output range and does not average.

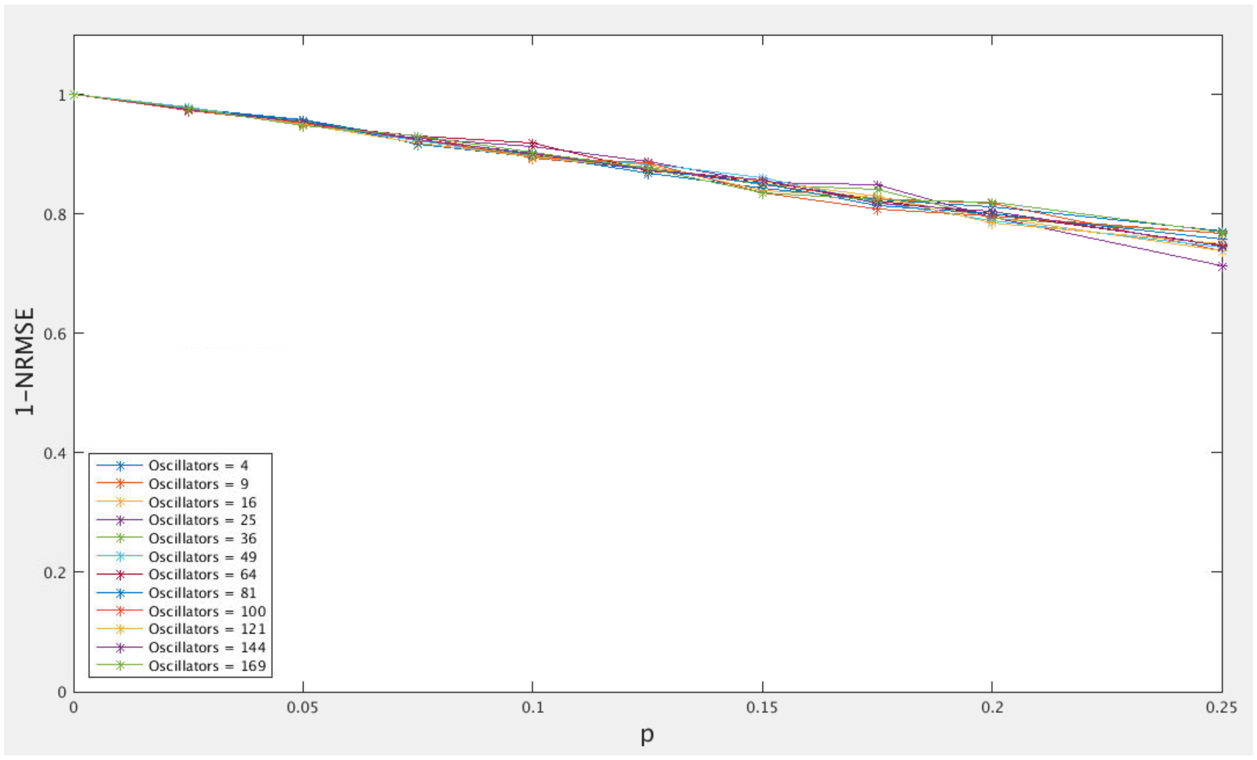


Figure 14: NRMSE of DOM(A, B) with *ON* variation for different array sizes.

5.2 CONVOLUTION

In this section, the effects of the parameters on convolution are analyzed. Each of the following subsections analyzes a different parameter in the following order: coupling asymmetry, locking region, input noise, and output noise.

5.2.1 Coupling Asymmetry

Figure 15 shows the *NRMSE* of *CA* variation on convolution for different oscillator array sizes. Just as in DOM, it shows that smaller array sizes are more sensitive than larger array sizes. As the number of oscillators increase, the average of the asymmetry approaches 1. Because there are three DOM clusters, it might seem intuitive that convolution should have greater error than DOM. However, according to Equation 7, the error attributed by DOM(A, 0) and DOM(0, B) gets subtracted out. Figure 16 shows a comparison of the amount of error in all three DOM clusters. The amount of error from each cluster is similar, with more error in DOM(A, 0) and DOM(0, B).

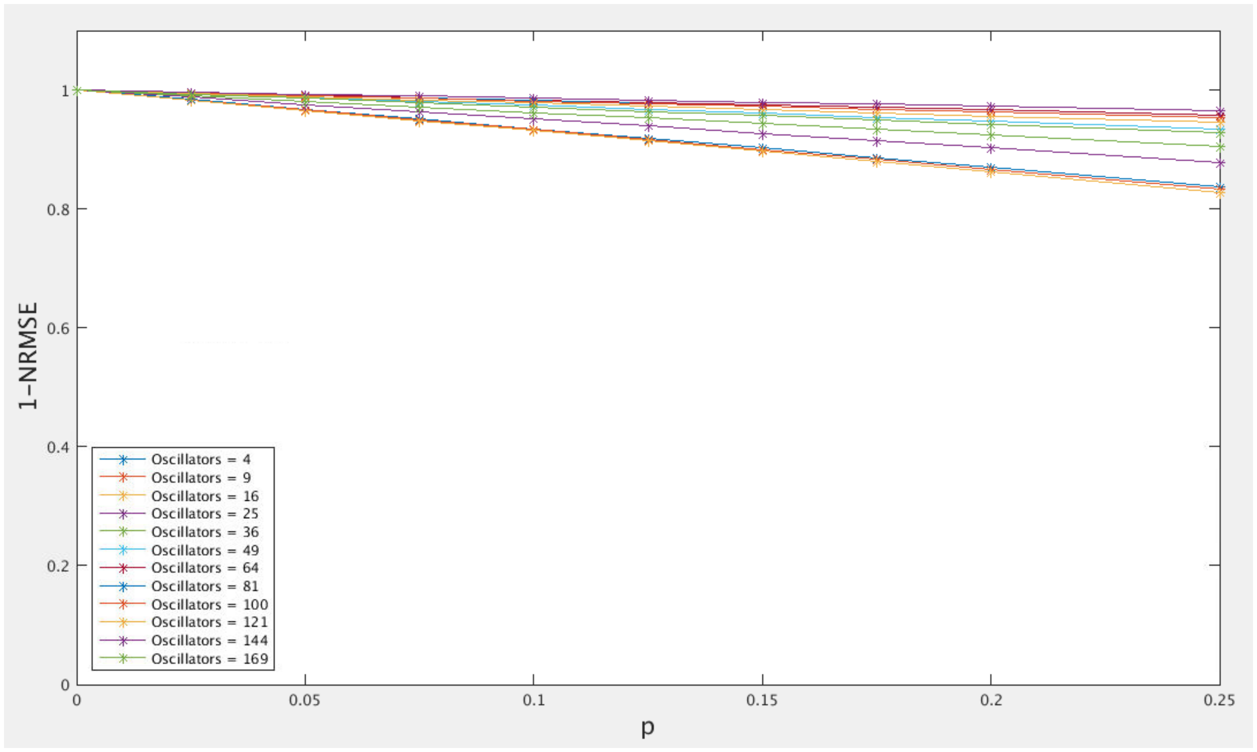


Figure 15: NRMSE of convolution with *CA* variation for different array sizes.

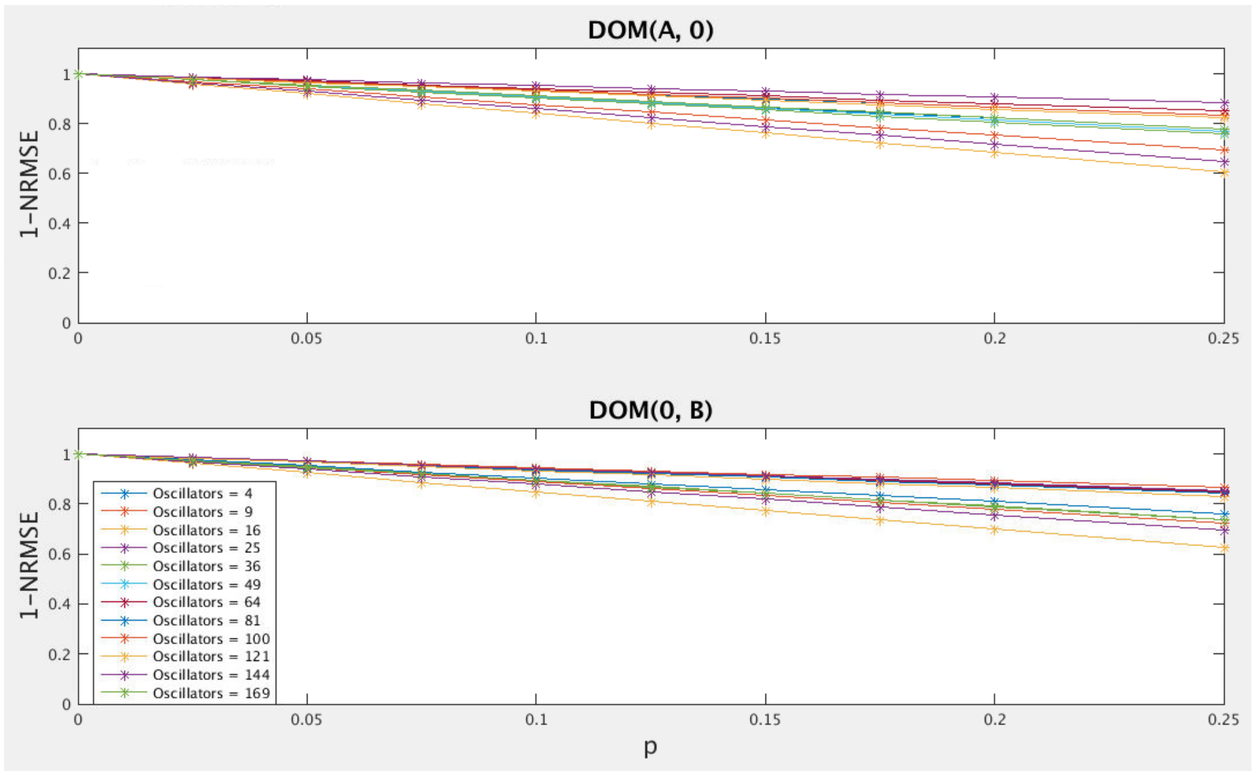


Figure 16: NRMSE of other DOM clusters in convolution with *CA* variation for different array sizes.

Figure 17 shows a histogram of the squared difference values from DOM(A, 0) and DOM(0, B) clusters for an array size of 64. The other array sizes tested, though not shown, have the same distribution. Unlike Figure 7, the distribution for these values is relatively uniform for small DOM values and skew in opposite directions as the DOM values increase. This means that there is less variation between different DOM values for these clusters.

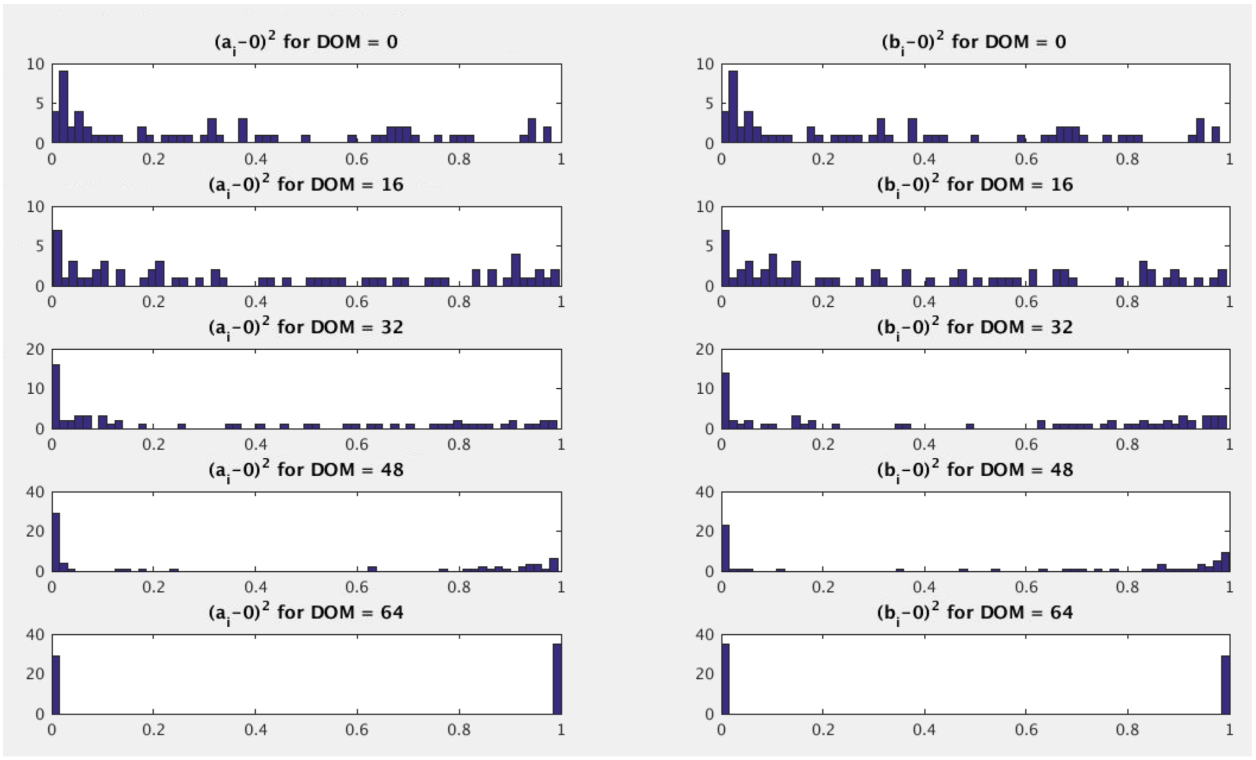


Figure 17: Histogram of $(a_i – 0)^2$ and $(b_i – 0)^2$ values for particular DOM values.

5.2.2 Locking Region

Figure 18 shows the *NRMSE* of *LR* variation on convolution for different oscillator array sizes. It shows that for smaller *LR* parameter values, array size and locking region are independent. However, as the parameter increases in value, locking region has more effect on convolution for smaller sizes. This is because for smaller array sizes, there are more DOM(A, 0) and DOM(0, B) values below the threshold, driving them to 0 and introducing even more error to the convolution, as shown in Figure 19 and Figure 20.

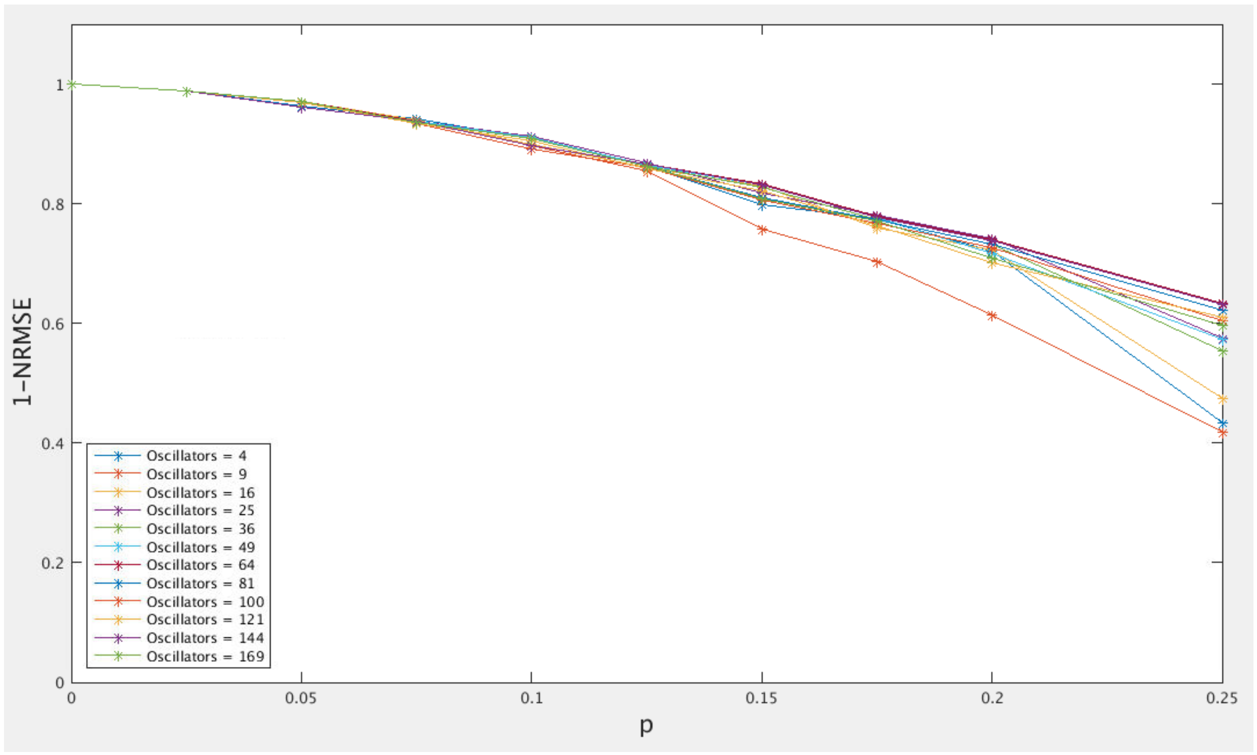


Figure 18: NRMSE of convolution with *LR* variation for different array sizes.

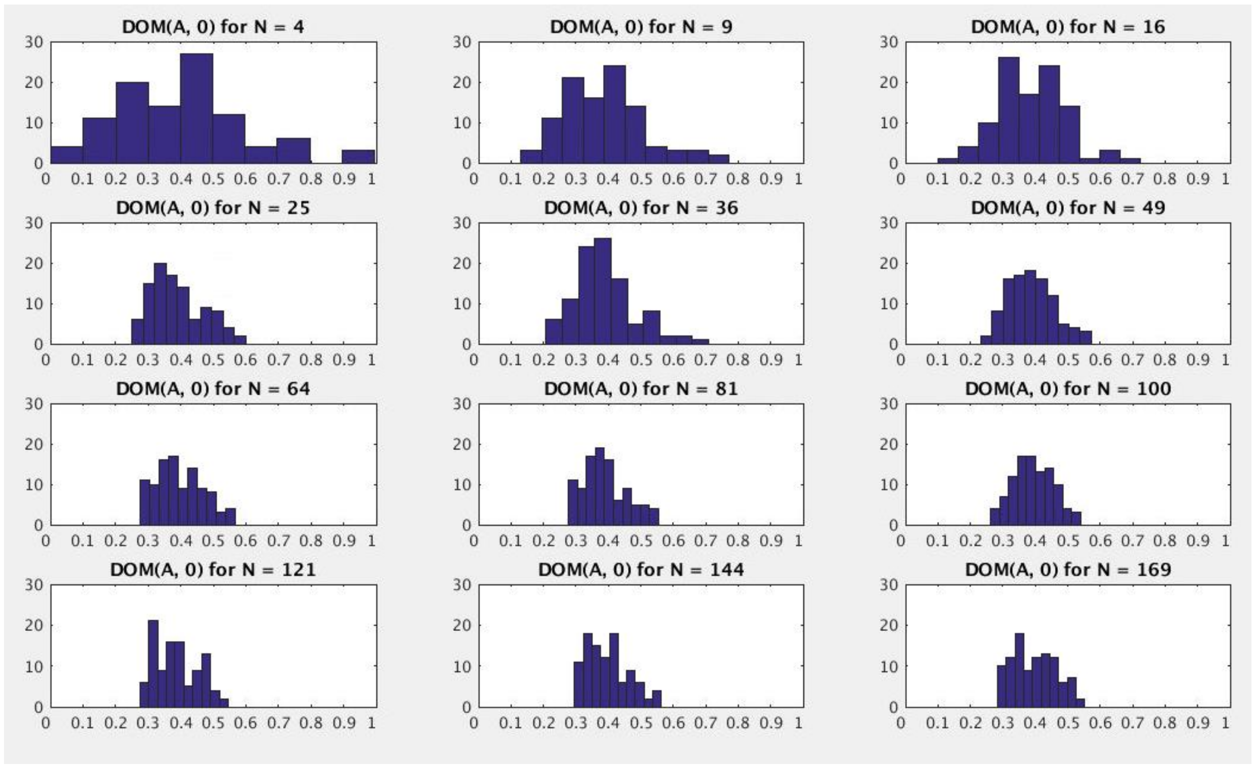


Figure 19: Histogram of DOM(A, 0) values in convolution for various sizes, where the x-axis is percentage of dynamic range.

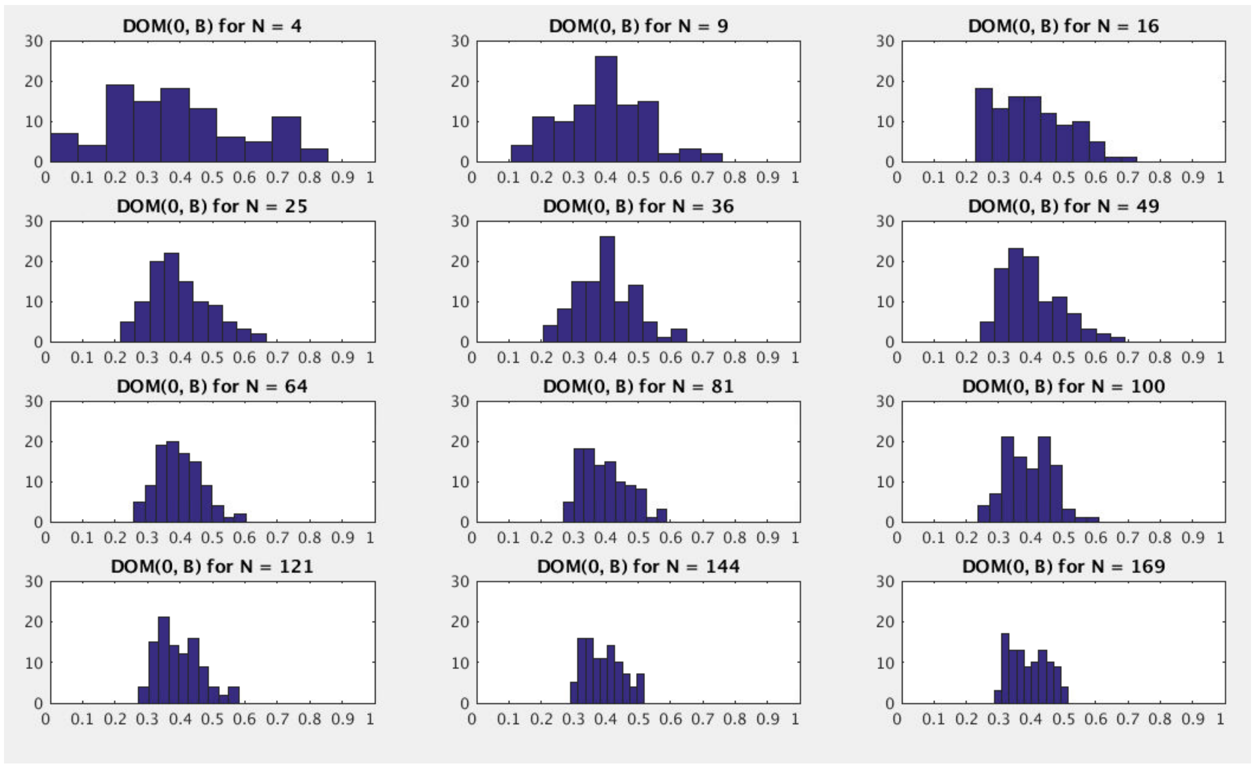


Figure 20: Histogram of DOM(0, B) values in convolution for various sizes, where the x-axis is percentage of dynamic range.

5.2.3 Input Noise

Figure 21 shows the *NRMSE* of *IN* variation on convolution for different oscillator array sizes. There is less discernable trend between oscillator array sizes for convolution because according to Figure 22, larger array sizes are more sensitive to *IN* for DOM(A, 0) and DOM(0, B). Where as for *CA*, the same proportional amount of error was being subtracted out, for *IN* this is not the case. Because the noise in larger array sizes for DOM(A, B) average out, the noise from the 0 input in DOM(A, 0) and DOM(B, 0), is not subtracted out in the DOM model.

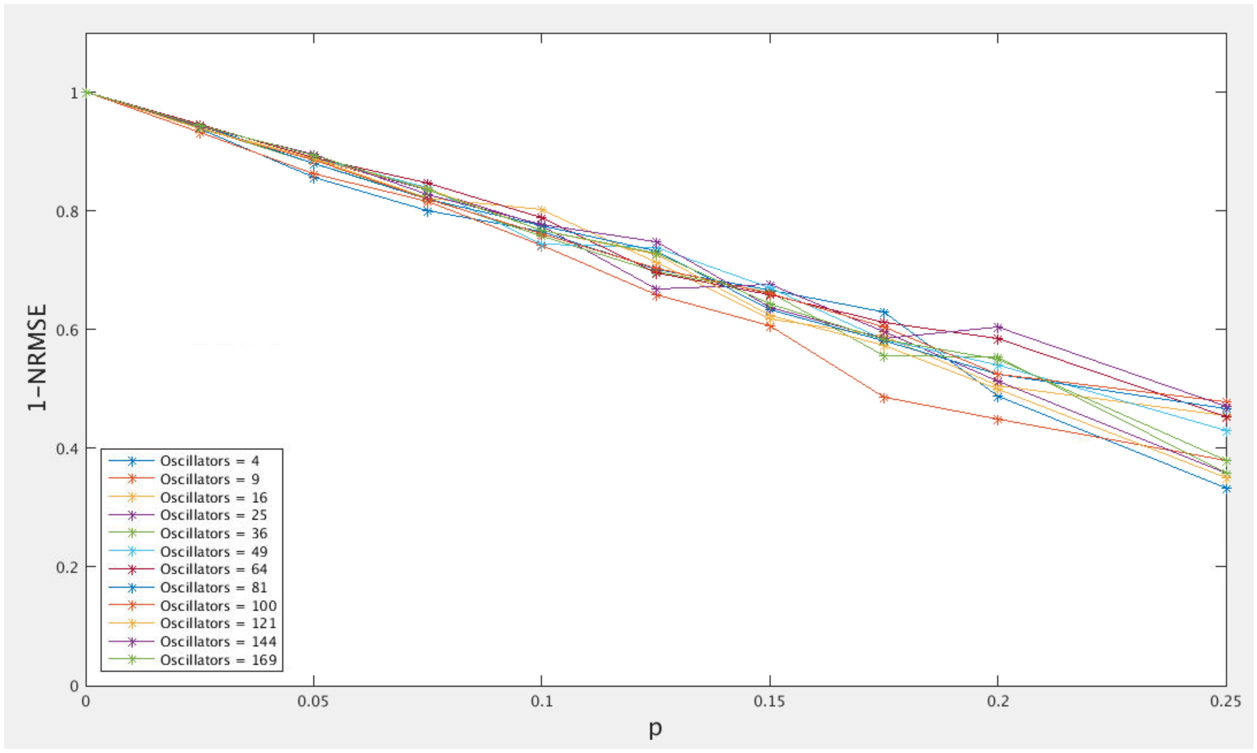


Figure 21: NRMSE of convolution with *IN* variation for different array sizes.

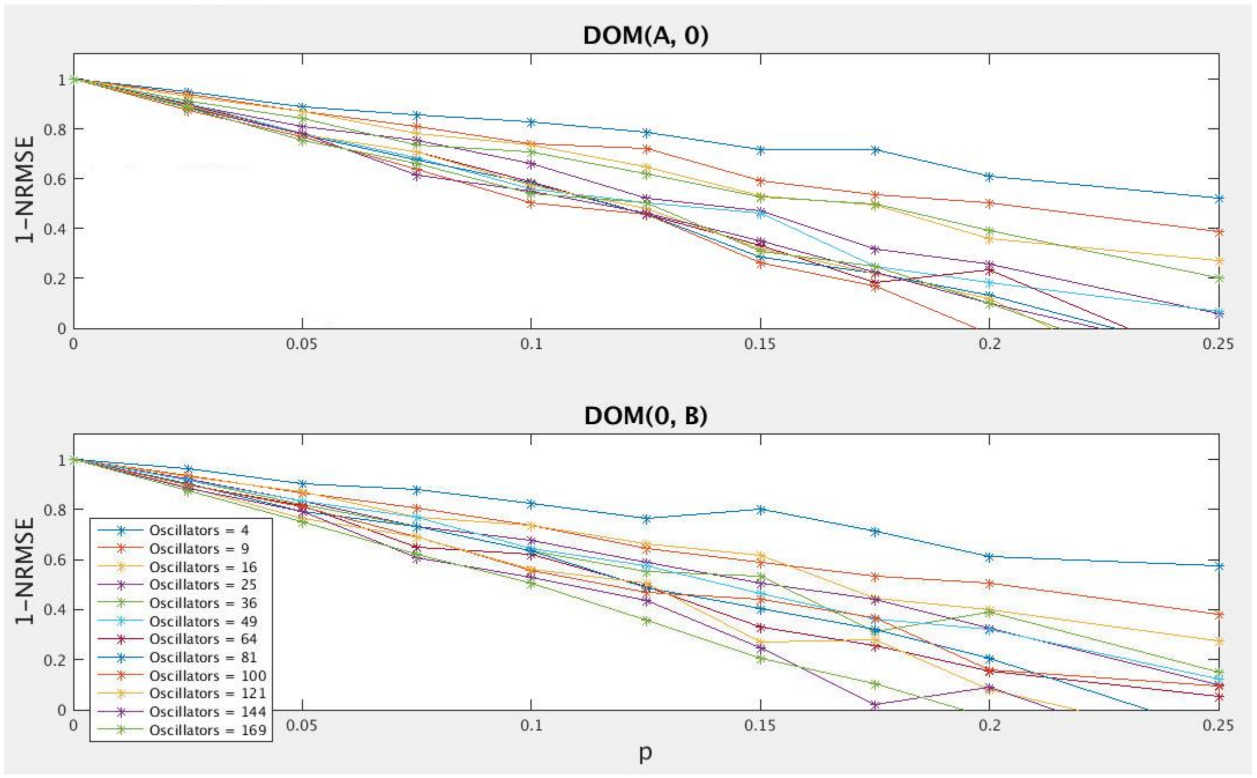


Figure 22: NRMSE of other DOM clusters in convolution with *IN* variation for different array sizes.

5.2.4 Output Noise

Figure 23 shows the NRMSE plot of *ON* on convolution for different array sizes. It is shown that, similar to *IN*, there is less discernibility between oscillator array sizes. However, the smallest array size, 4, generally performs better than the others. Figure 24 shows that also similar to *IN*, DOM(A, 0) and DOM(0, B) are more sensitive for larger array sizes. This is because according to Figure 19 and 20, the range of values for DOM(A, 0) and DOM(0, B) is similar to DOM(A, B). Even though the fraction of noise being added is relative, larger array sizes have a smaller range of DOM values, so the noise added to the DOM values for those clusters is relatively higher.

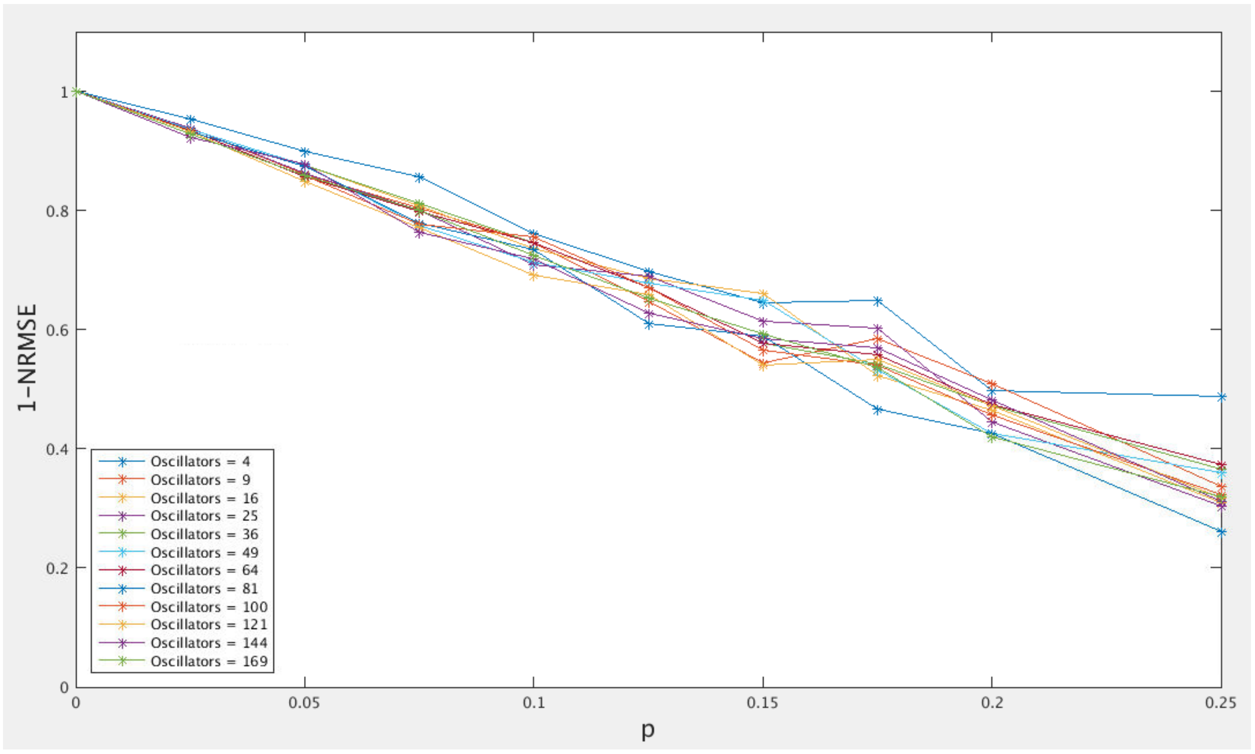


Figure 23: NRMSE of convolution with *ON* variation for different array sizes.

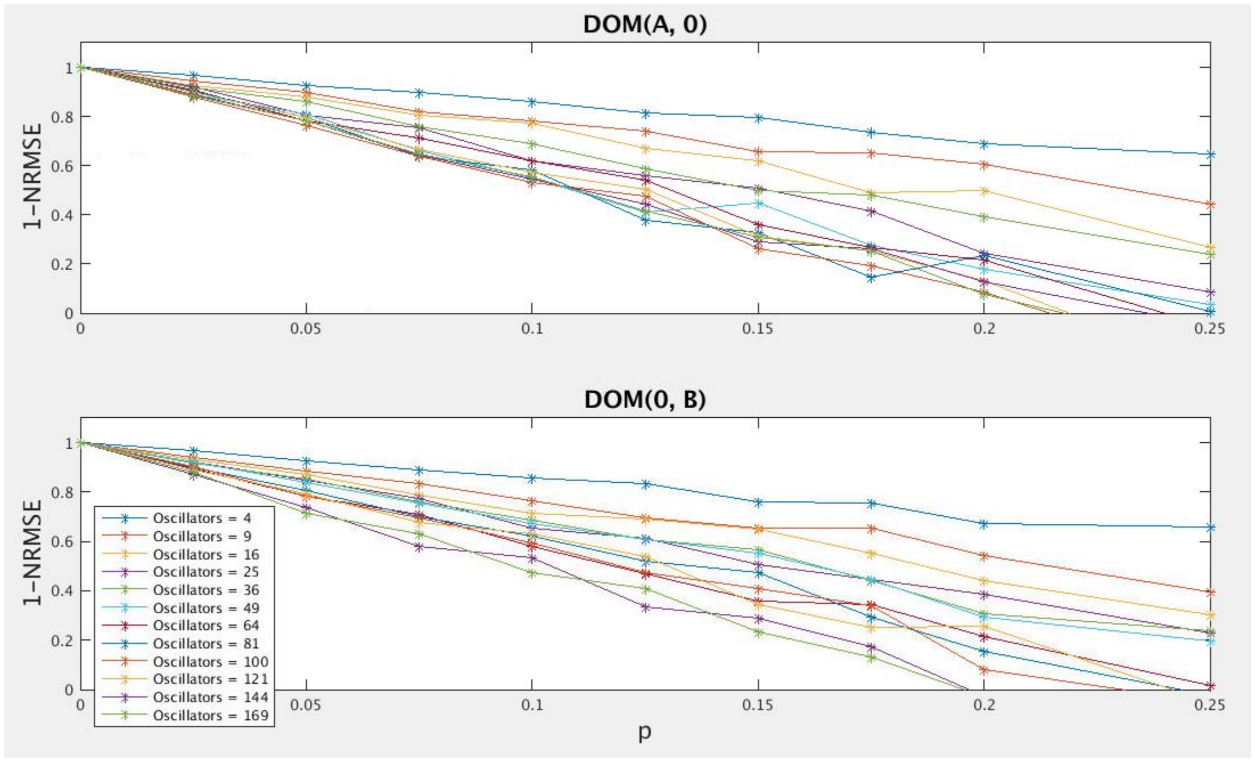


Figure 24: NRMSE of DOM clusters in convolution with *ON* variation for different array sizes.

5.3 GABOR FILTERING

To further demonstrate oscillator-based convolution, in this section it is implemented in Gabor filtering. The effects of the parameters on Gabor filtering are analyzed.

5.3.1 Implementation Design

. Figure 25 shows examples of Gabor filters of various orientations used in edge detection. The 0° filter can be used to detect horizontal edges, the 90° filter can be used to detect vertical edges and the 45° and 135° filters can be used to detect diagonal edges.

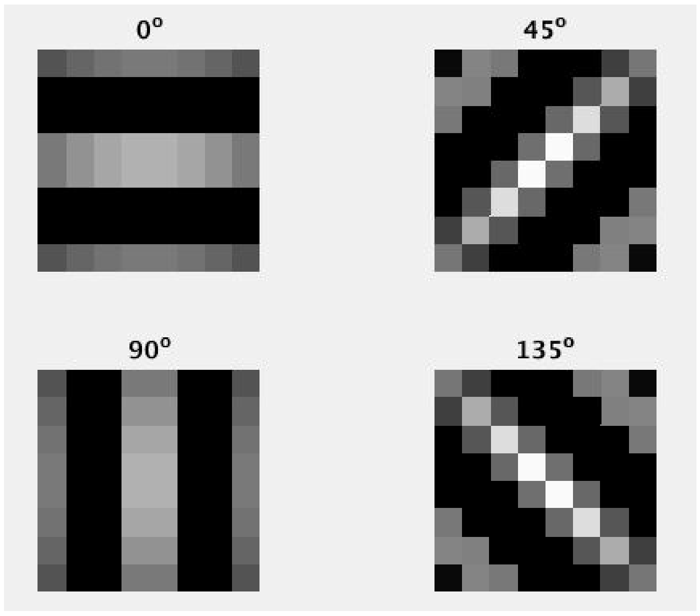


Figure 25: Example Gabor filters for edge detection.

Gabor filtering was implemented by applying an 8x8 45° Gabor filter to the 120x120 greyscale image of a vehicle in Figure 26.

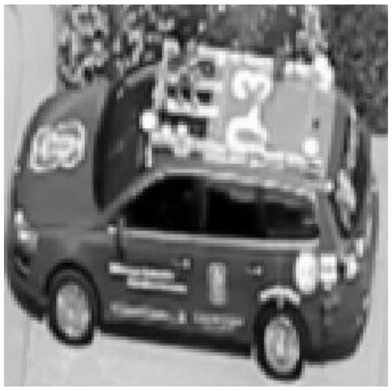


Figure 26: Input image used for Gabor filtering.

5.3.2 Filtering Output

Figure 27 shows the result of the image filtered with parameter variations. The image is severely washed out by input and output noise after only increasing *p* to 0.05, however it is still useful with *CA* variation where *p* = 0.25. Although *LR* distorts the image by making more pixels black, the image may still be useful for some filtering applications as much of the shape of the image is still visible.

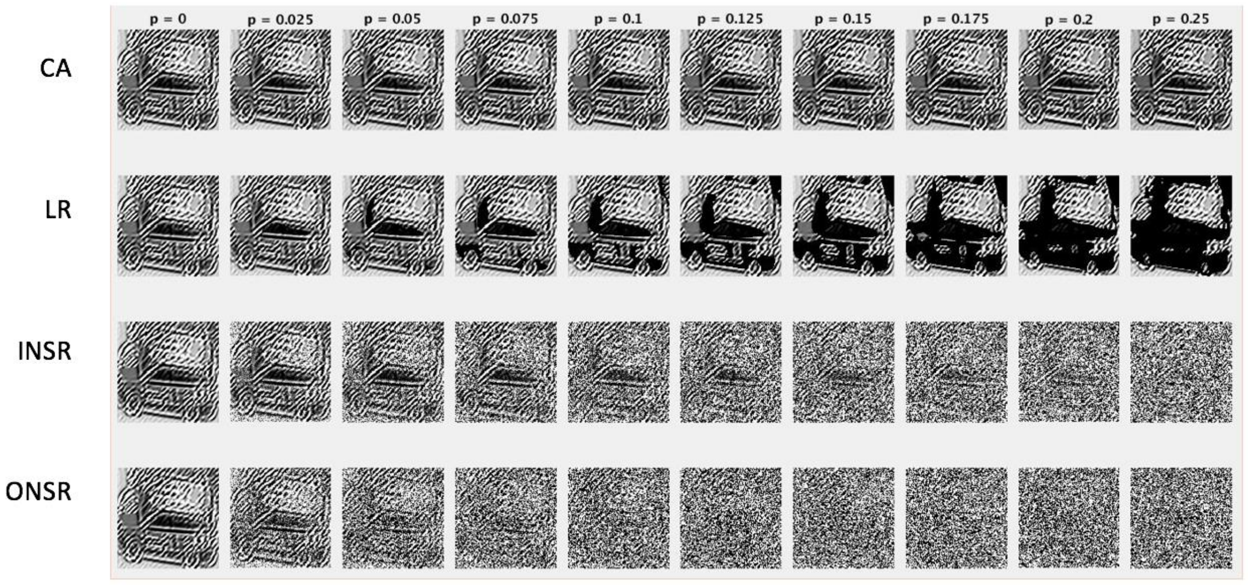


Figure 27: Image results from filtering with model parameter variation.

Figure 28 shows a histogram the DOM values from each cluster of the convolution in the Gabor filter study. Although *LR* and *ON* are fractions of 64, the dynamic range based on 8x8 convolutions, the DOM values computed in this study exceed 64 because the inputs are not bounded and some filter values are less than 0. The DOM(A, 0) cluster is the most impacted by *LR* because many of the DOM values fall below the threshold values, where DOM(A, B) and DOM(0, B) values remain above the threshold.

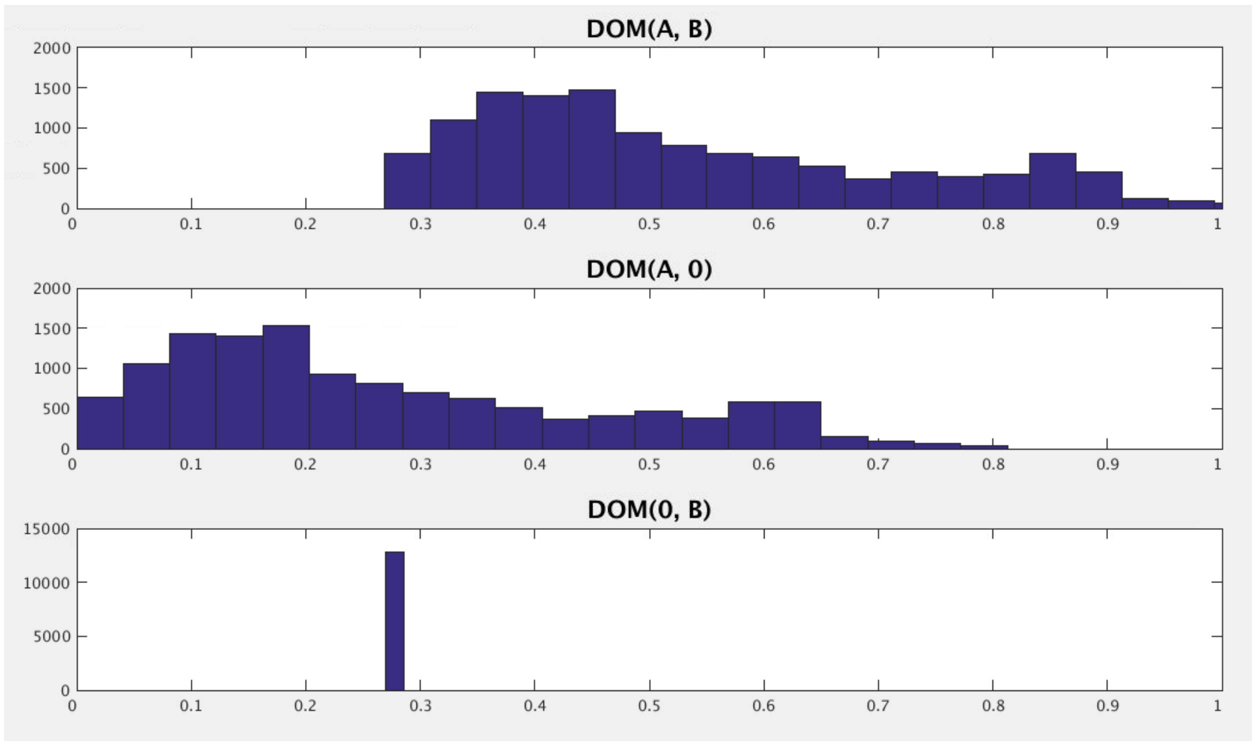


Figure 28: Histogram of DOM values from each cluster in the convolution for Gabor filtering, where the x-axis is percentage of dynamic range.

5.4 DISCRETE COSINE TRANSFORM

In this section, the effects of the parameters on DCT are analyzed. Each of the following subsections analyzes a different parameter in the following order: coupling asymmetry, locking region, input noise, and output noise.

5.4.1 Coupling Asymmetry

Figure 29 shows the *MSE* plot of *CA* variation on DCT for different array sizes. It shows that *CA* variation has little impact on DCT because the squared difference values in the DCT are small for DCT because one of the input vectors contains cosine values.

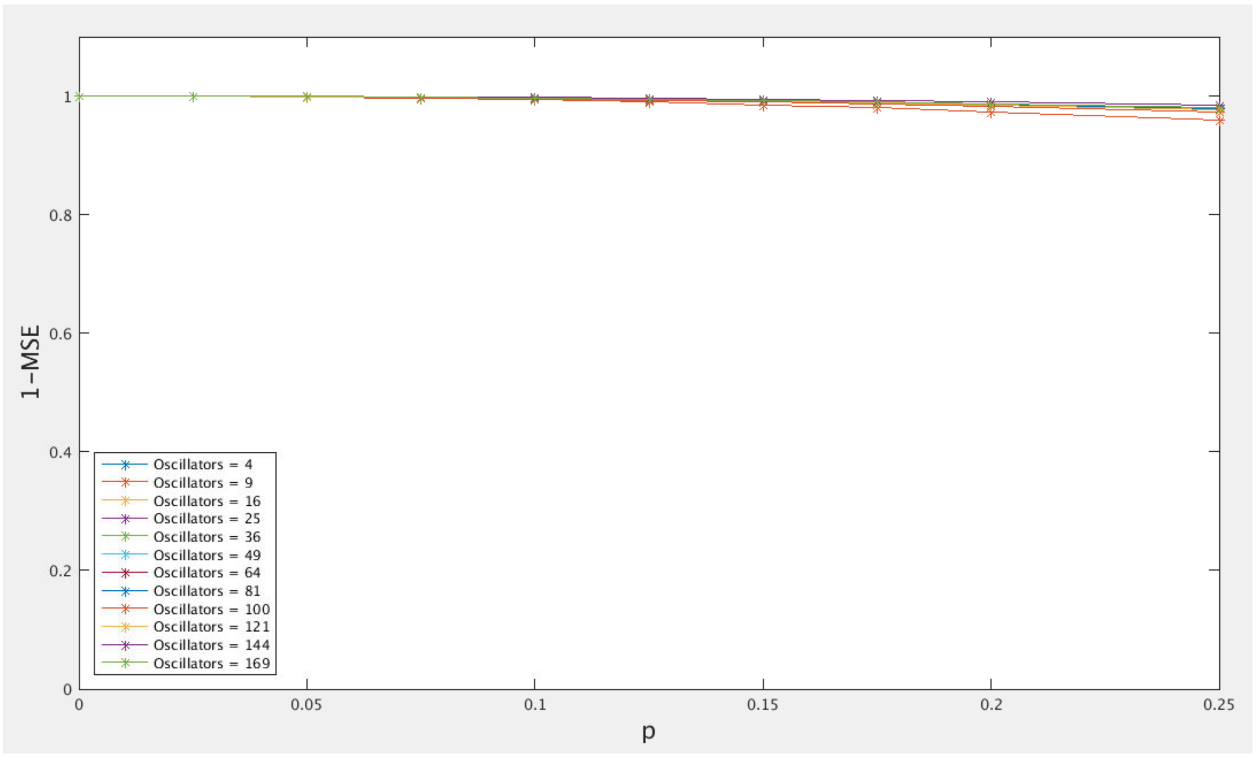


Figure 29: MSE of DCT with *CA* variation for different array sizes.

5.4.2 Locking Region

Figure 30 shows MSE for *LR* variation on DCT for different array sizes. It shows DCT is fairly resistant to *LR* variation for smaller values of p and becomes more sensitive as p increases. Figure 31 shows that a majority of the DOM(A, B) values for most of the sizes are around 30%

of the dynamic range. Any sizes with values below the threshold only have one. Figure 32 shows that DOM(A, 0) does not have any values below the threshold for this particular vector. Given the uniform randomness of the input vectors, it is extremely unlikely to have a vector that is similar to cosines.

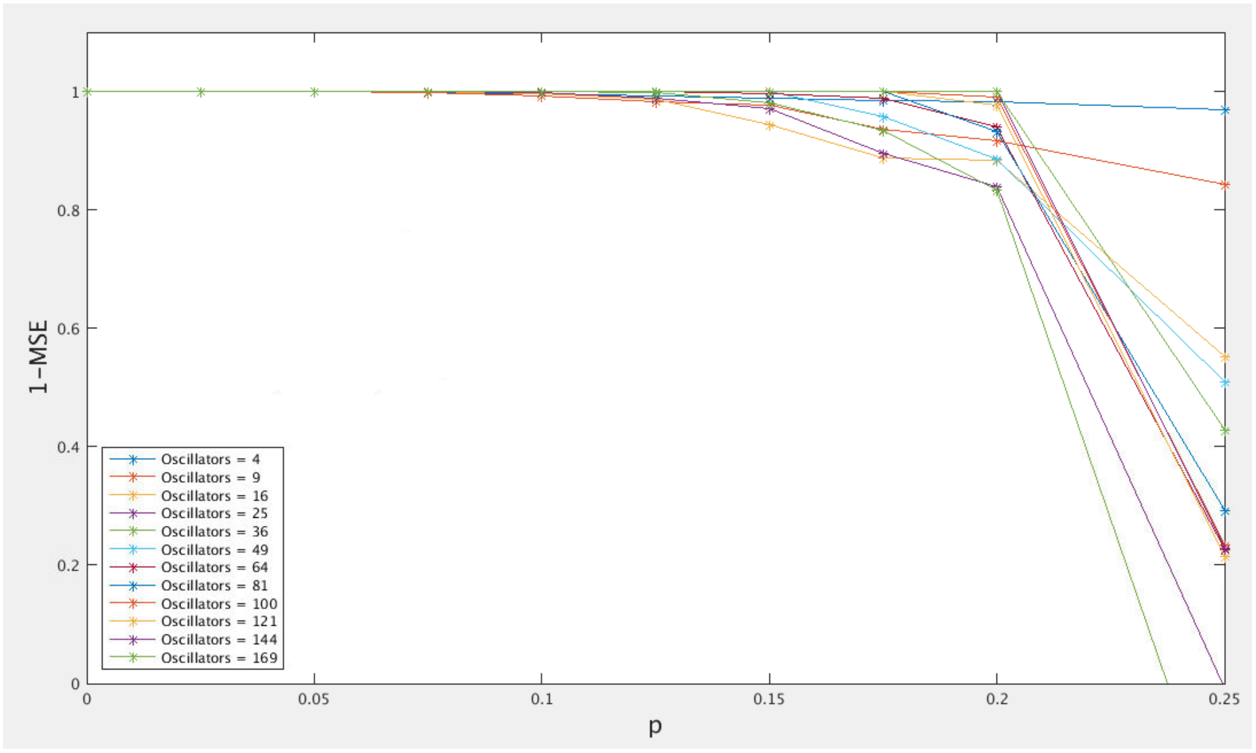


Figure 30: MSE of DCT with *LR* variation for different array sizes.

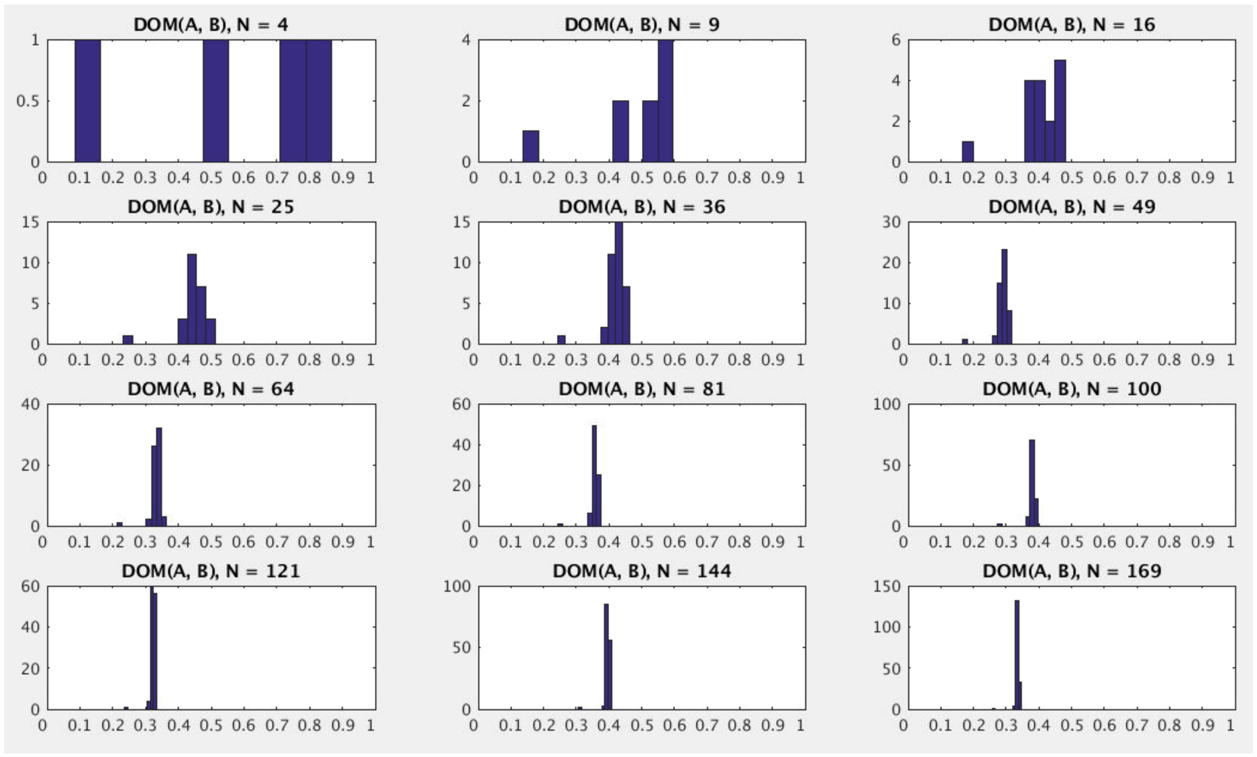


Figure 31: Histogram of DOM(A, B) values in the convolution for DCT of one vector for different image sizes, where the x-axis is percentage of dynamic range.

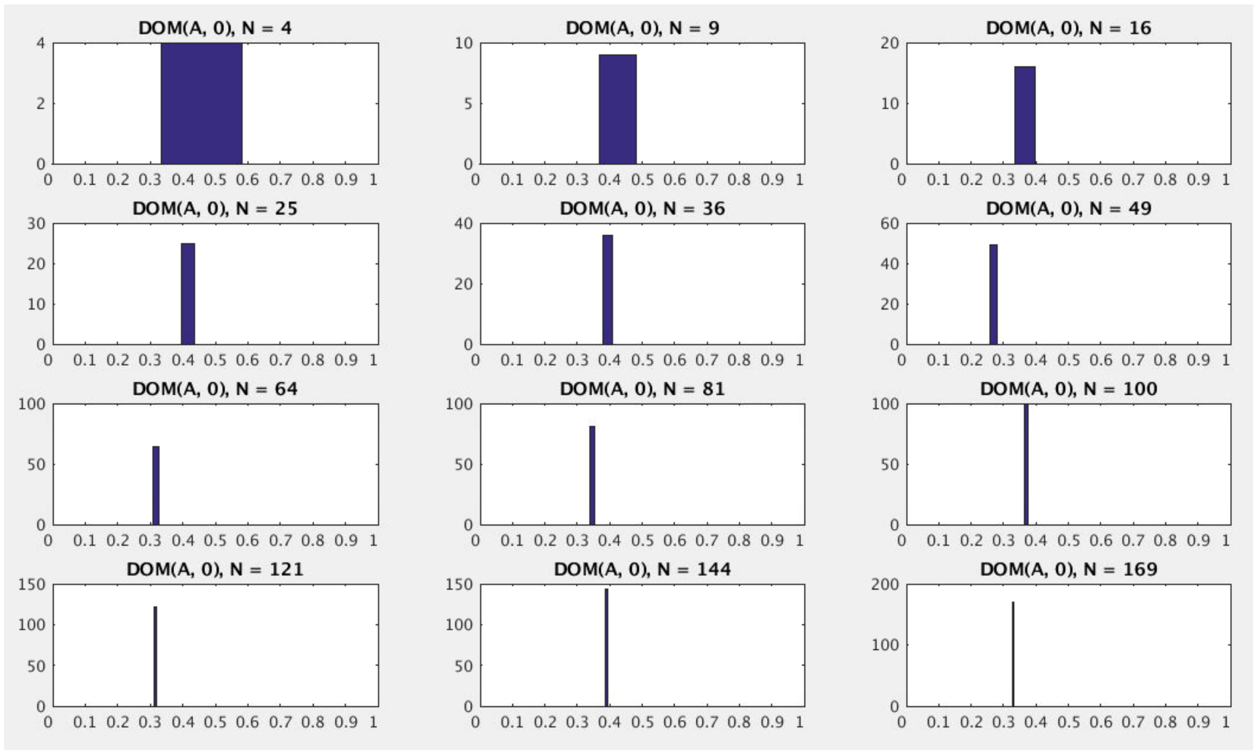


Figure 32: Histogram of DOM(A, 0) values in the convolution for DCT of one vector for different image sizes, where the x-axis is percentage of dynamic range.

5.4.3 Input Noise

Figure 33 shows *IN* on DCT for various oscillator array sizes. It shows that for DCT, larger array sizes experience more error. This is because the cosine values get smaller as the array size increases, increasing the difference between input vectors *A* and *B*. This increases the variation in the DCT, as discussed in Section 5.1.3. In addition, larger array sizes involve more convolutions in DCT, since there are *n* DCT operations for any problem size.

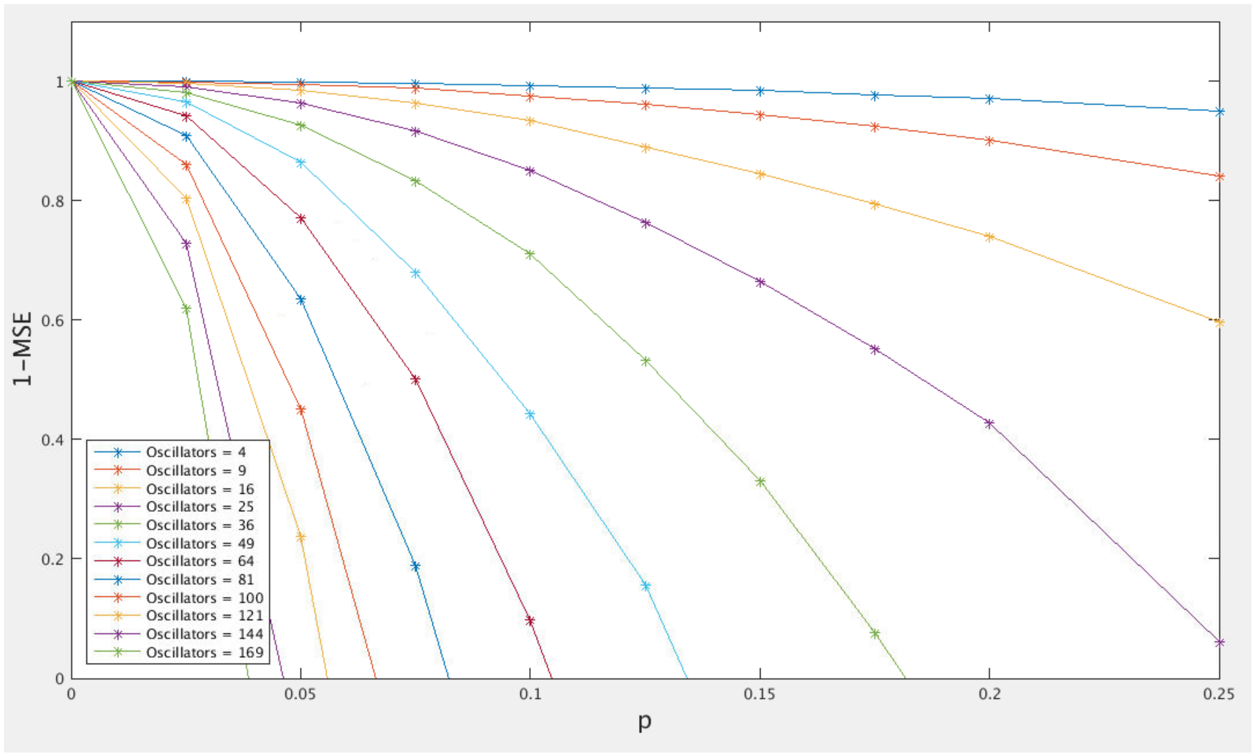


Figure 33: MSE of DCT with *IN* variation for different array sizes.

5.4.4 Output Noise

Figure 34 shows *ON* on DCT for various sizes. Again, it is shown that there is more error experienced by the larger array sizes. This is because there is more noise added to larger array sizes because the noise is a fraction of a larger dynamic range. As mentioned before, there are also more operations for larger array sizes. Unlike the DOM and convolution studies, the error for this study was measured with *MSE* which is is not normalized.

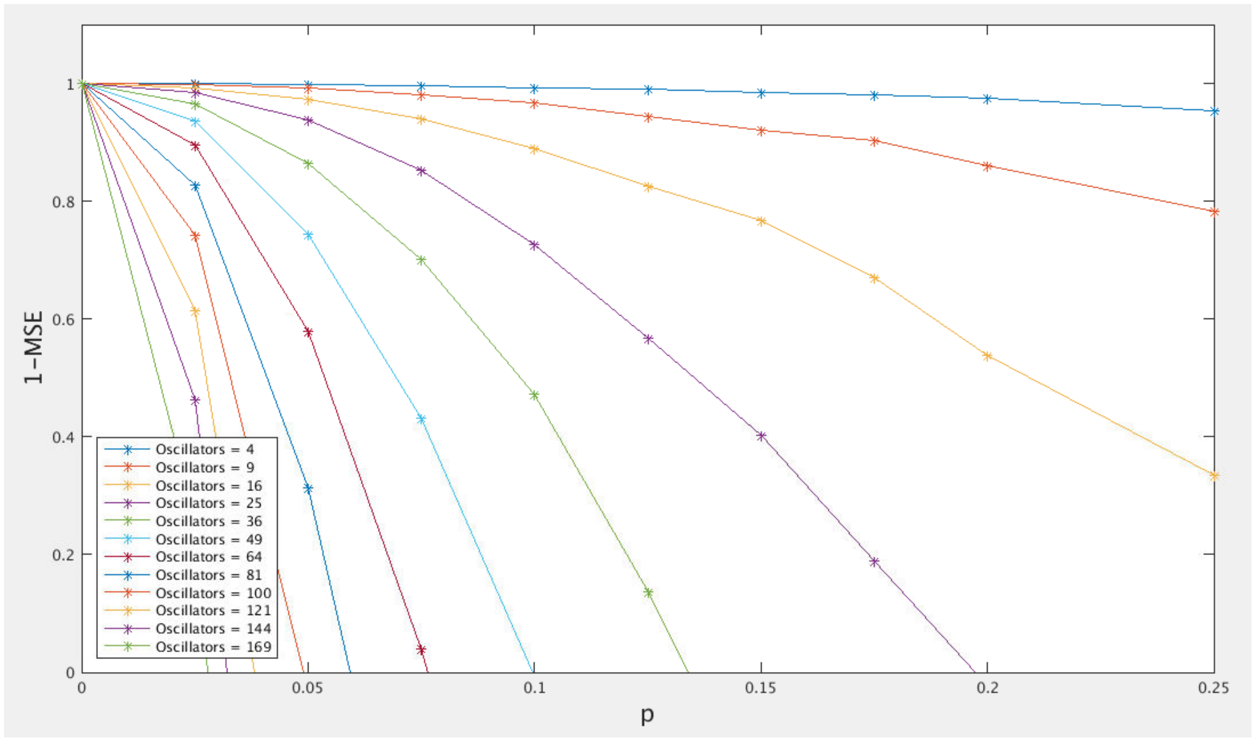


Figure 34: MSE of DCT with *ON* variation for different array sizes.

5.5 IMAGE COMPRESSION

To further demonstrate oscillator-based DCT, in this section DCT is implemented in an image compression algorithm. DCT is used in many image compression algorithms. It can be used to remove the high frequencies from an image and reduce the density of the image, in turn reducing the number of bits to store it.

5.5.1 Implementation Design

The compression algorithm used for this implementation performs 2-D DCT on the original image and removes the high frequencies by zeroing the DCT values in the lower righthand side of the DCT matrix. Then the 2-D inverse DCT (IDCT) returns the image without the information from those frequencies. The image is encoded using a technique known as run-length encoding in which sequential data of the same value is stored as a single value and a count. Removing the higher frequencies reduces the length of the encoding.

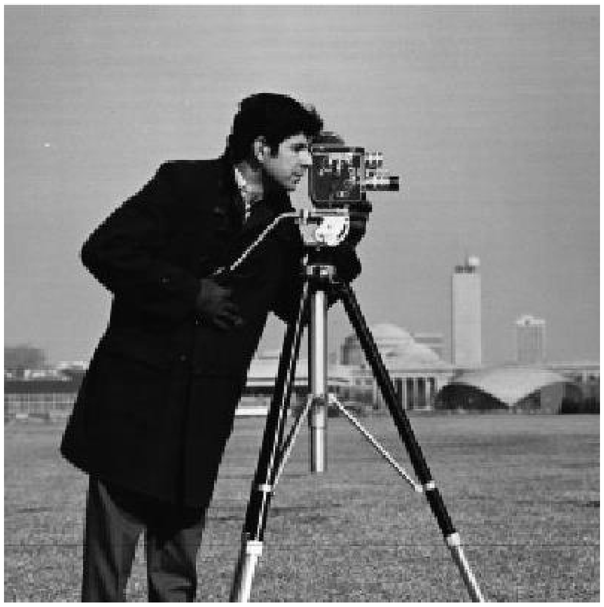


Figure 35: Image used for DCT compression

For this oscillator-based compression implementation, the 256x256 image pictured in Figure 35 was used. The image is 64KB. MSE was used to measure error. 2-D DCT/IDCT are implemented by computing two 1-D DCT/IDCT operations.

5.5.2 Results

Figure 36 shows the images resulting from the compression with paramater variations. *CA* variation shows minor impact on the image quality because of the small squared difference values. *LR* does degrade the image, though not as much as Gabor filtering because as seen in Figure 37 many of the DOM values are close to 0 before they are cutoff by *LR*, and the images may still be useful depending on the application. In compression, *IN* and *ON* do not washout the image as much as Gabor filtering because of the definition of oscillator-based DCT in Equation 9; some of the impact of noise is subtrated between the two DOM clusters.

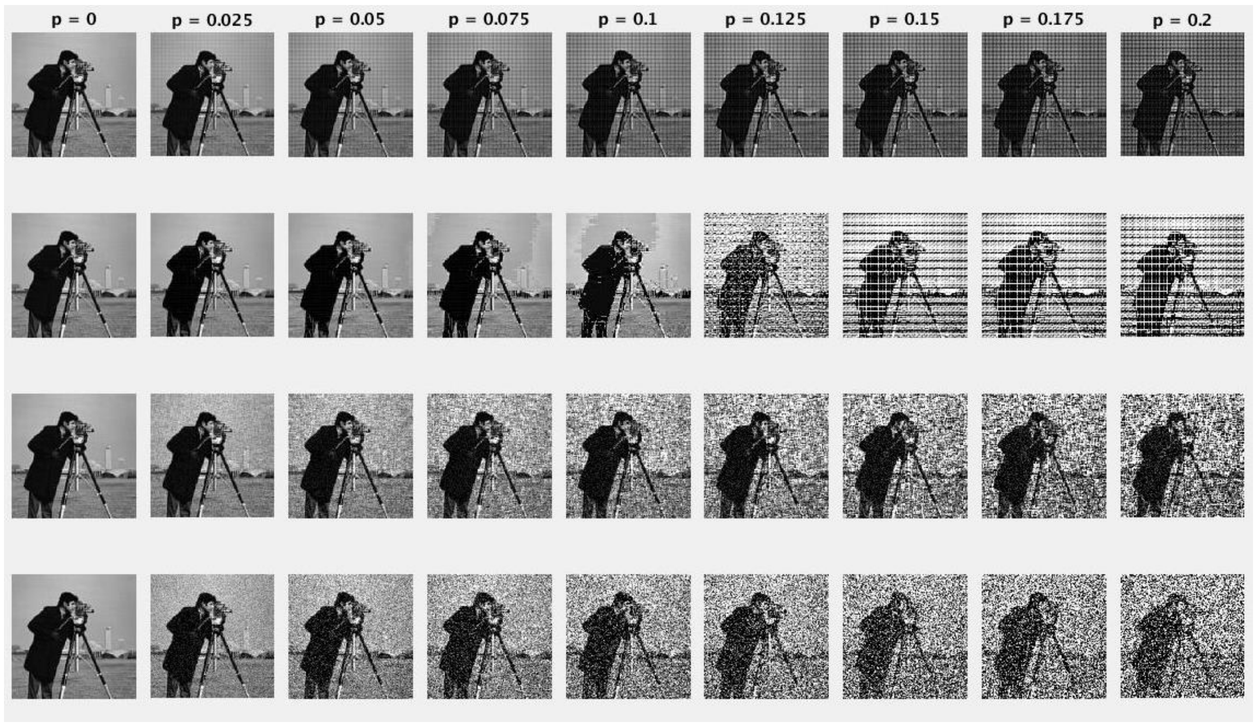


Figure 36: Image results from compression with model parameter variation.

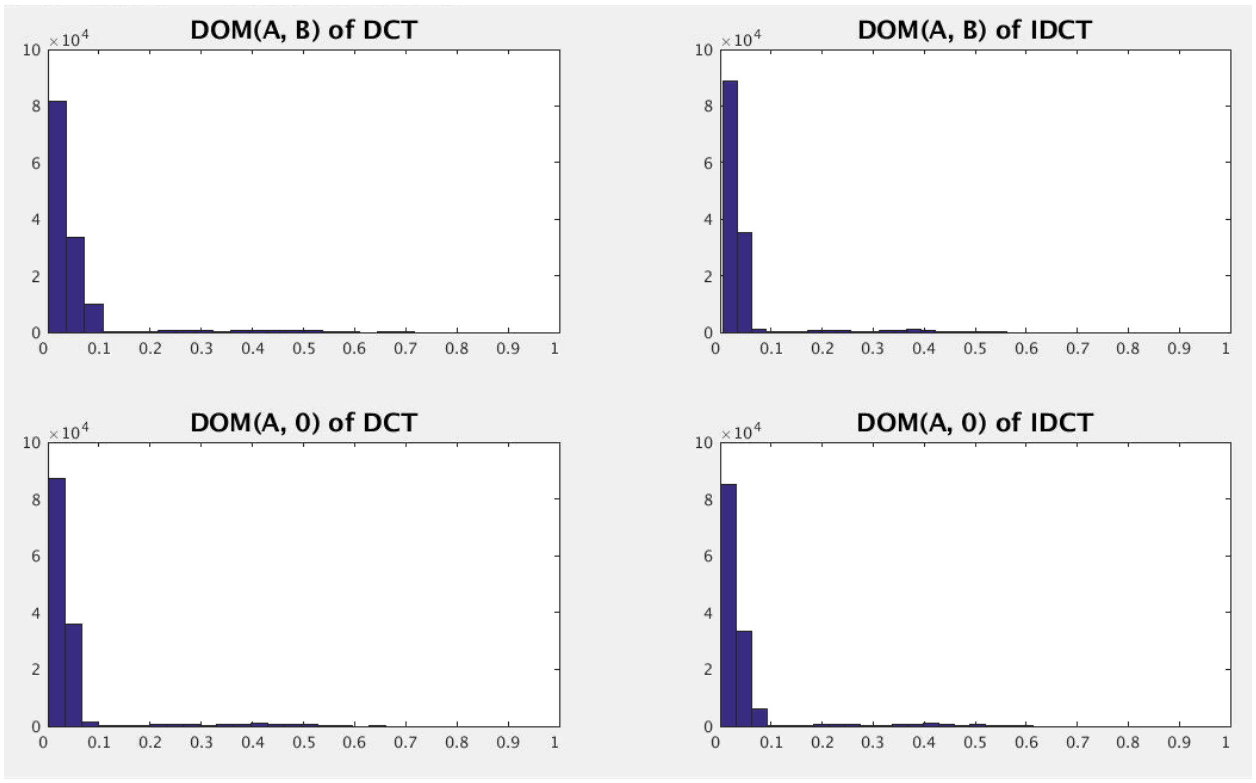


Figure 37: Histogram of DOM values from each cluster in the convolution for compression, where the x-axis is percentage of dynamic range.

Figure 38 shows how the file size of the compression was impacted by the model parameter variations. *CA, IN,* and *ON* change the pixel values and adds more data to the image, thus, more bits are required to represent the image. The opposite if true for *LR*, which removes data in the DCT, reducing the bits required to represent the image. This figure shows that despite the relatively strong visual integrity of compression, methods would be necessary to mitigate the increase in file size from parameter variation.

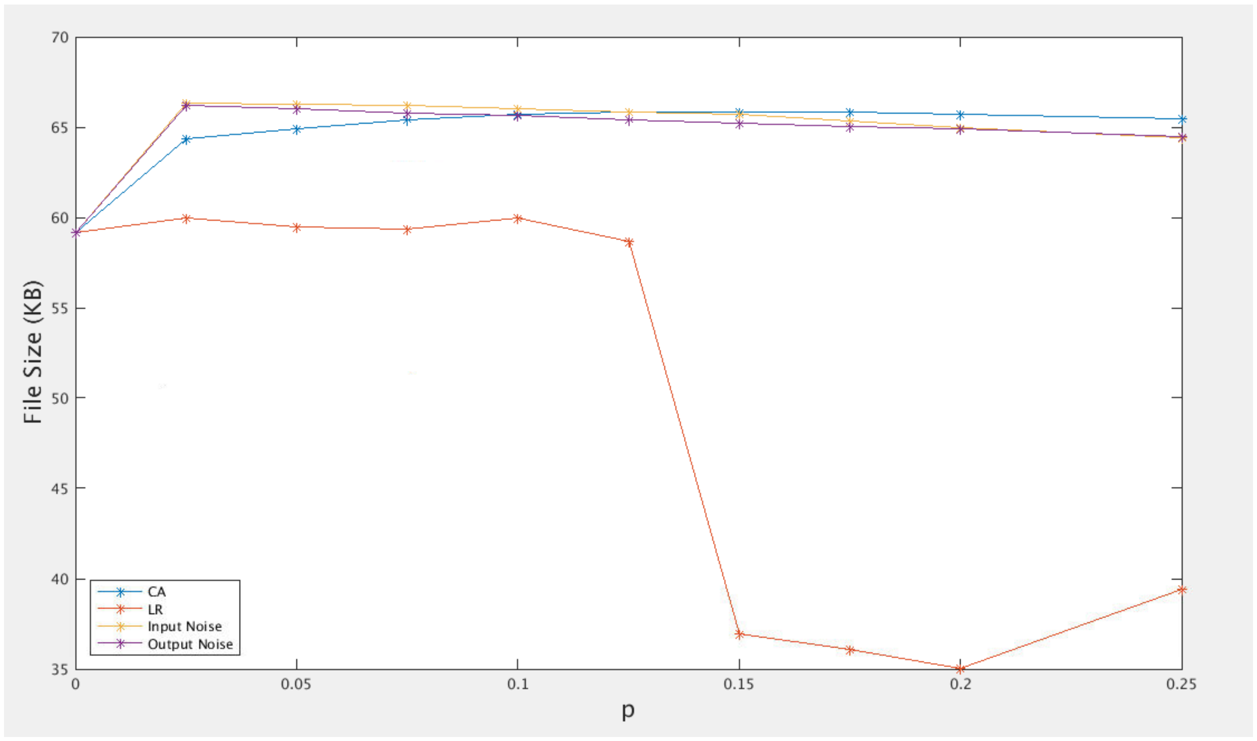


Figure 38: Image file size with model parameter variatoin.

5.6 DISCRETE FOURIER TRANSFORM

In this section, the effects of the parameters on DFT are analyzed. Each of the following subsections analyzes a different parameter in the following order: coupling asymmetry, locking region, input noise, and output noise.

5.6.1 Coupling Asymmetry

Figure 39 shows *CA* variation on DFT for various oscillator array sizes. There is notably more error present in DFT operations compared to DCT because each DFT operation involves two convolutions according to Equation 13 and the error becomes additive. This is also because of the larger difference squared values from the complex sinusoids that range on the interval [-1 1], as opposed to the smaller intervals of the cosines in DCT. Like DCT, larger array sizes experience more error because larger array sizes involve more DFT operations.

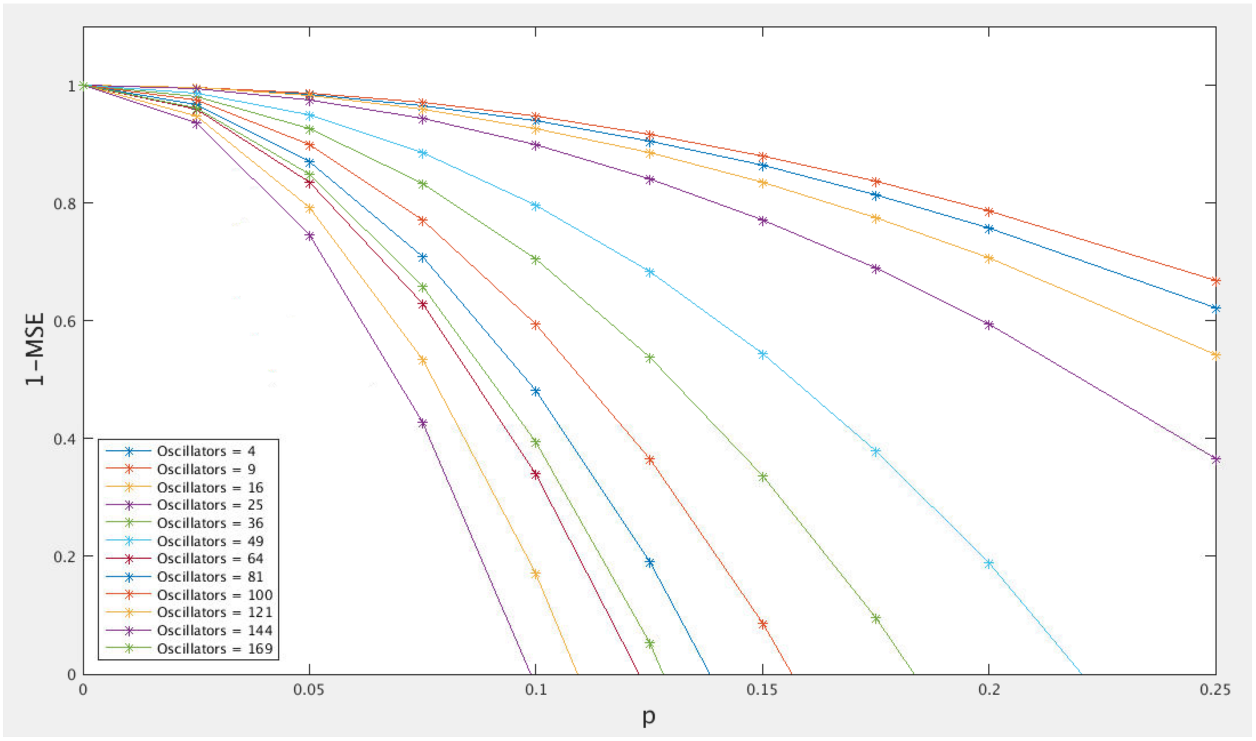


Figure 39: MSE of DFT with *CA* variation for different array sizes.

5.6.2 Locking Region

Figure 40 shows *LR* variation on DFT for various oscillator array sizes. It shows that DFT is less sensitive to *LR* for smaller *p* values and more sensitive for larger *p* values. Because of the greater difference between the input vectors and the complex sinusoids, the DFT experiences greater DOM values as shown in Figure 41 – Figure 42 and, just as in DCT, it becomes unlikely to have DOM values below 25% threshold.

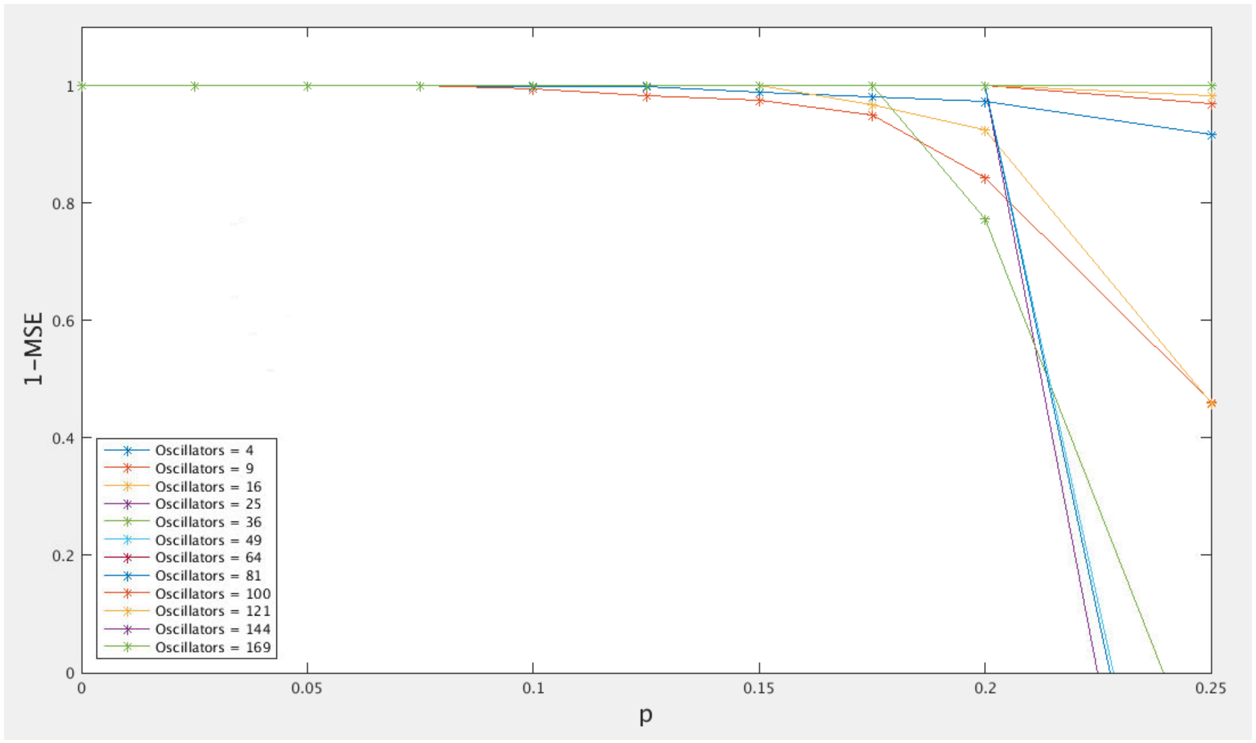


Figure 40: MSE of DFT with *LR* variation for different array sizes.

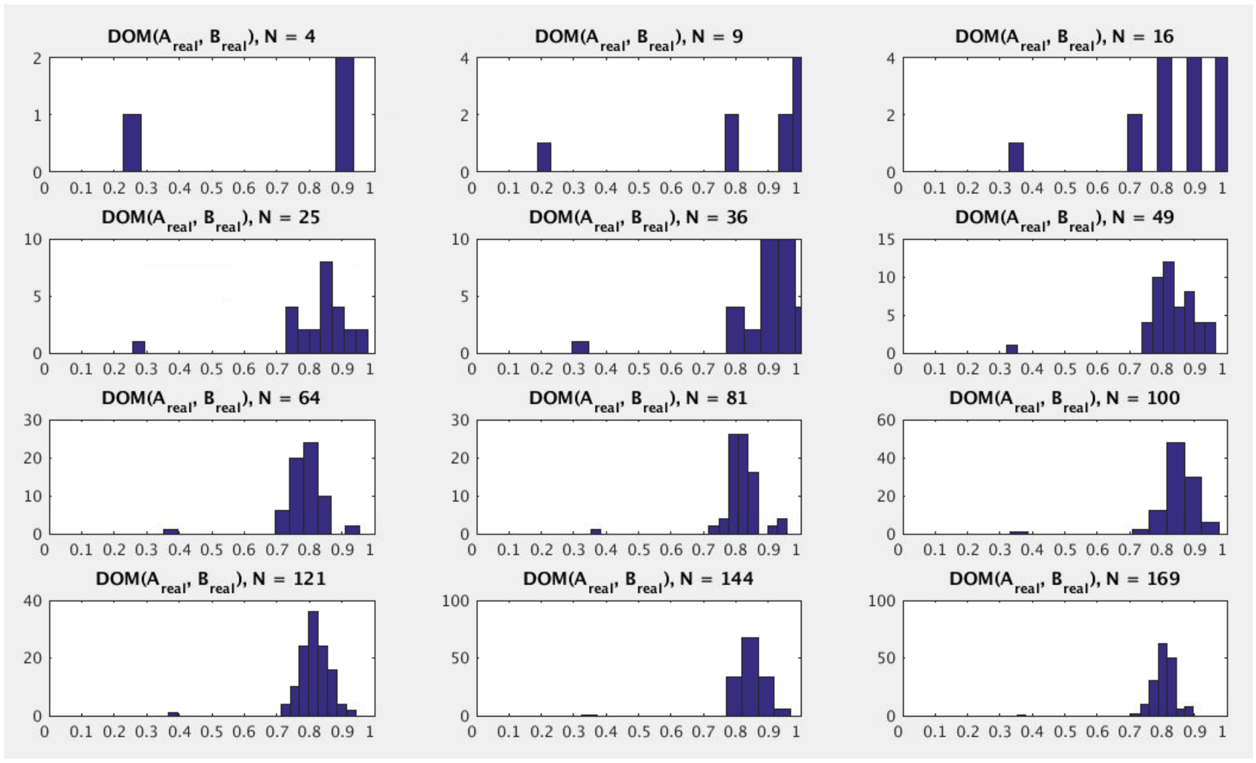


Figure 41: Histogram of DOM(A_{real}, B_{real}) values in the convolution for DFT for different image sizes, where the x-axis is percentage of dynamic range.

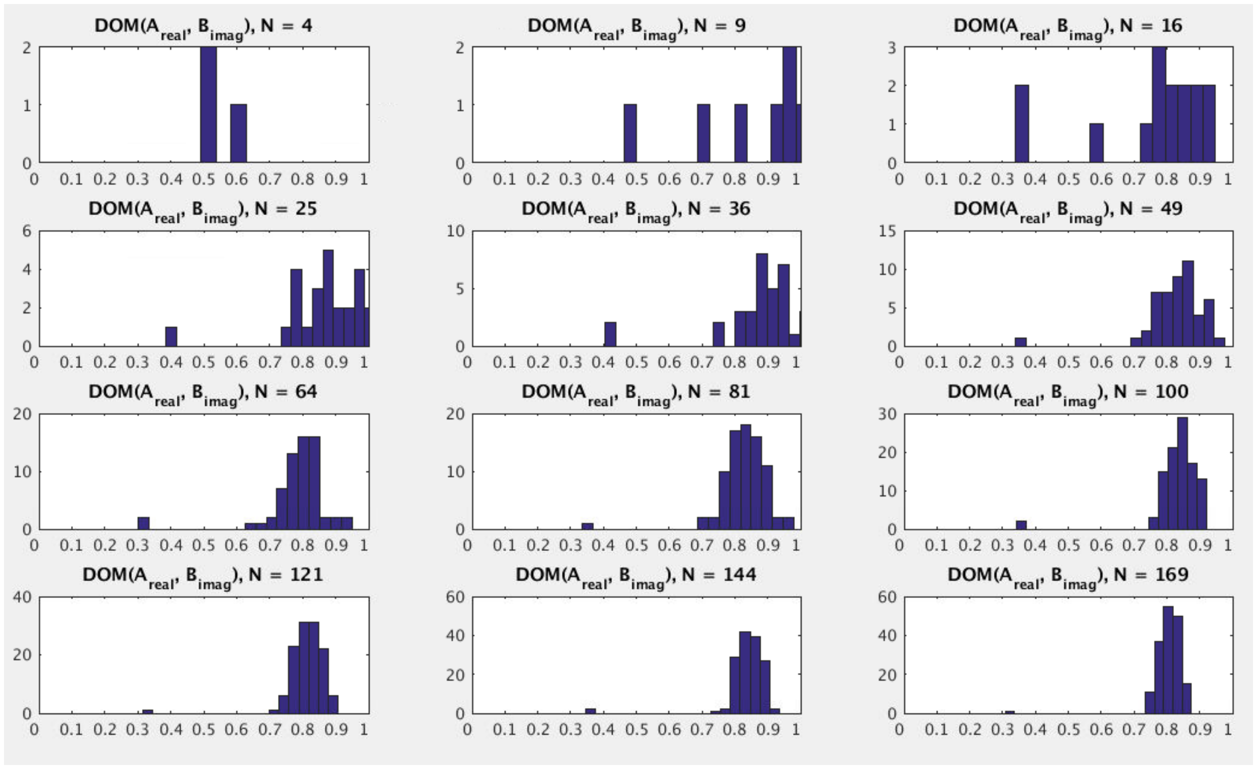


Figure 42: Histogram of DOM(A_{real}, B_{imag}) values in the convolution for DFT for different image sizes, where the x-axis is percentage of dynamic range.

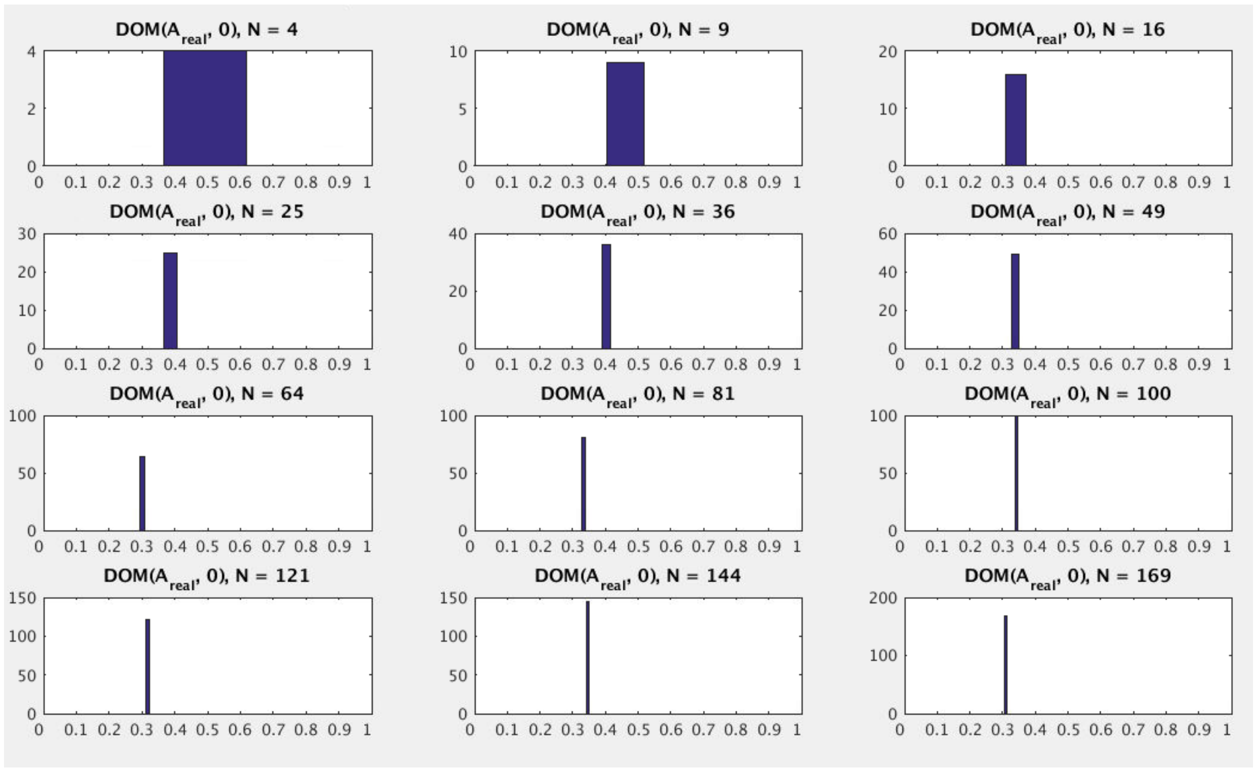


Figure 43: Histogram of DOM(A_{real}, 0) values in the convolution for DFT for different image sizes, where the x-axis is percentage of dynamic range.

5.6.3 Input Noise

Figure 44 shows *IN* variation on DFT for various oscillator array sizes. Again, as the array sizes increase, the differences between the input vector and the complex sinusoids increase, causing more variation in the DFT. In addition, there are more DFT operations for larger sizes.

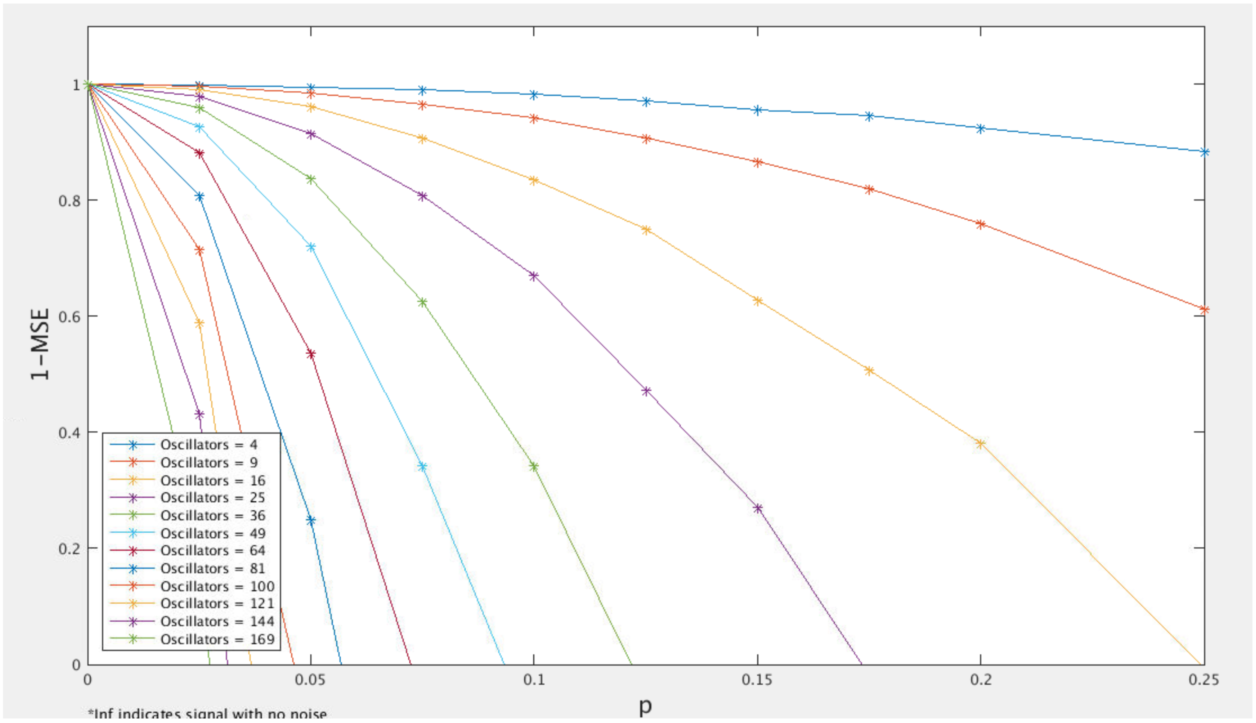


Figure 44: MSE of DFT with *IN* variation for different array sizes.

5.6.4 Output Noise

Figure 45 shows *ON* variation on DFT for various oscillator array sizes. More noise is added to the larger array sizes because of a larger dynamic range and thus experience a greater difference. As in DCT, the DFT is measured with a *MSE* and is not normalized and larger array sizes involve more operations.

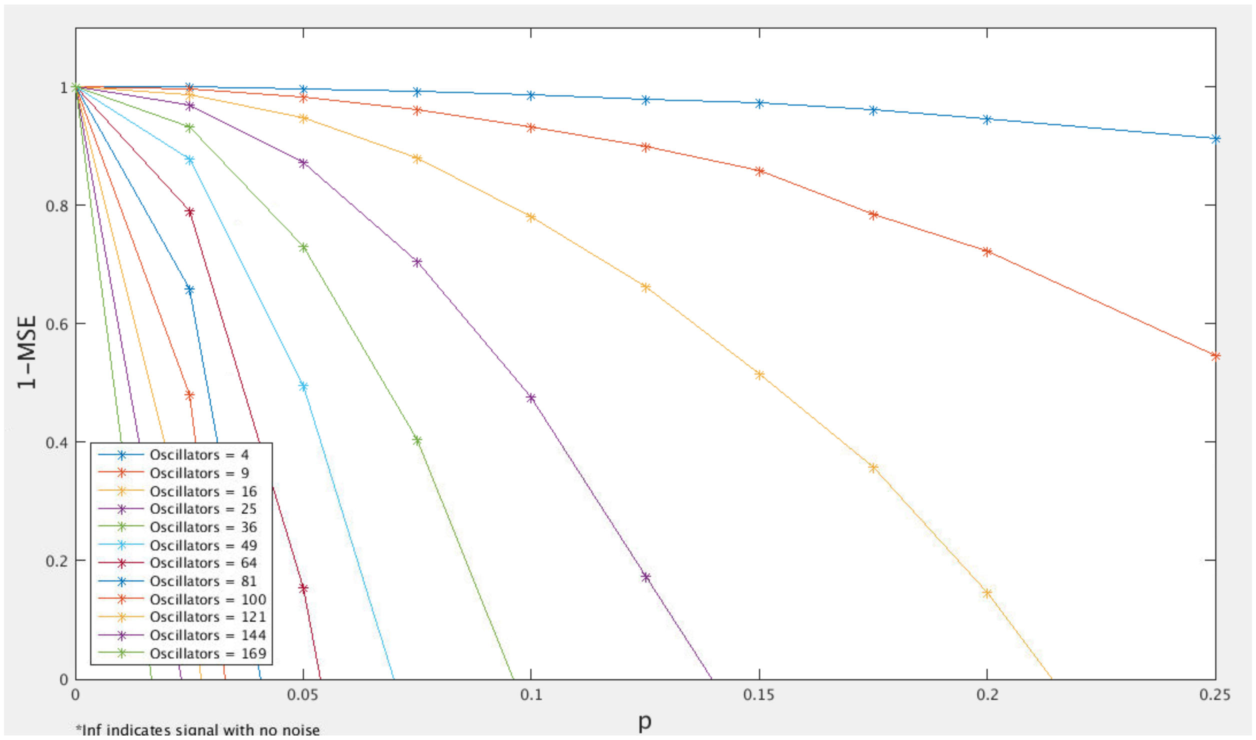


Figure 45: MSE of DFT with *ON* variation for different array sizes.

6.0 CONCLUSION

In this thesis, it has been shown that a Degree of Match circuit built using oscillator arrays can be characterized by a squared Euclidean distance equation, a commonly used distance metric, and therefore can be used for template matching. A real-world circuit model was then built with parameters for coupling asymmetry, locking region, input noise, and output noise. It was shown that using this DOM model, an exact convolution can be computed using three DOM oscillator clusters. From here, convolution-based signal processing primitives discrete cosine transform and discrete Fourier transform can be computed; it was shown how these primitives can be optimized to reduce the number of required oscillators and the amount of error cause from model parameter variation.

The impact of these variations on the DOM circuit was analyzed and it was determined that *CA* and *IN* become less important as the array size increases, where as *LR* and *ON* are parameters are unrelated to array size. Parameter variation impact was then analyzed for convolution and it was found that, again, *CA* and *IN* impact is greater for smaller array sizes, however *LR* variation also has greater impact for smaller array sizes for greater *p*. Convolution was then implemented in a Gabor filtering application where it was shown that *CA* variation had negligible impact on filtering, and that though *LR* variation degraded the image, visually the filtered images were could still usable depending on the application. *IN* and *ON* both rendered the image unrecognizable with relatively small values of *p*.

Convolution was used to perform DCT and the results showed that *CA* variation had little impact and *LR* had minor impact for low parameter variation values. Input and output noise impacted DCT the most and showed that larger sizes performed worse because more variation is added. DCT was then implemented in a compression application and it was shown that although the output images from the compression were still viable even with high levels of parameter variations, *CA, IN,* and *ON* variation added more information to the images and caused their file sizes to increase during compression. *LR* variation in fact significant reduced the file size. Convolution was lastly used to compute DFT and the results were similar to that of DCT, only worse because more convolutions were involved with the operation.

The viability of computing image processing primitives using coupled oscillator arrays has been proven, provided that the oscillator arrays behaves as the squared L_2^2 norm. In the future, it is hoped that hardware implementations these of coupled oscillators will be designed and analyzed, verifying this distance metric model. This will allow for more opportunities to exploit the benefits of the technology by integrating them into full systems such as image processing pipelines and classifiers and building high power, low energy nano-devices.

BIBLIOGRAPHY

[1] Christian Huygens, “Horologium Oscillatorium”, 1673.

[2] Horvath, A., Synchronization in cellular spin torque oscillator arrays. Cellular Nanoscale Networks and their applications (CNNA), 13th International Workshop on. IEEE, 2012.

[3] Shibata, T., et al., (2012). CMOS supporting circuitries for nano-oscillator-based associative memories. 13th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA), 1-5. Turin, Italy, August 29-31, 2012.

[4] Hoppensteadt F.C. and Izhikevich E.M. (2001) Synchronization of MEMS Resonators and Mechanical Neurocomputing. IEEE Transactions On Circuits and Systems I, 48:133-138

[5] Nikonov, Dmitri E., Ian A. Young, and George I. Bourianoff. "Convolutional Networks for Image Processing by Coupled Oscillator Arrays." arXiv preprint arXiv:1409.4469 (2014).

[6] D. M. Chiarulli, B. Jennings, Y. Fang, and A. Steel, and S. P. Levitan, “A Computational Primitive for Convolution based on Coupled Oscillator Arrays”, ISVLSI 2015

[7] B. B. Jennings *et al.*, "HMAX image processing pipeline with coupled oscillator acceleration," *2014 IEEE Workshop on Signal Processing Systems (SiPS)*, Belfast, 2014, pp. 1-6.

[8] D. J. Gabor, “Theory of communication,” IEE, vol. 93, no. 26, pp. 429–457, 1946.

[9] A. C. Bovik, M. Clark, and W. S. Geisler, “Multichannel texture analysis using localized spatial filters,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 12, no. 1, pp. 57–73, 1990.

[10] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete cosine transform. IEEE Trans. on Computers, 23:88–93, January 1974.

[11] Wallace, Gregory K. "The JPEG still picture compression standard." Consumer Electronics, IEEE Transactions on 38.1 (1992): xviii-xxxiv.

[12] Cochran, William T., et al. "What is the fast Fourier transform?" Proceedings of the IEEE 55.10 (1967): 1664-1674.

[13] K. Ramya, K. K. Manoj, and S. Mithunraj, “FPGA Implementation of Gabor Filter Design for Image Processing Application”, International Journal of Emerging Technology and Advanced Engineering, vol. 4, issue 1, January 2014

[14] A. K. Jha, A. Kumar, G. Schaefer, and M. A. R. Ahad, “An efficient edge preserving image interpolation algorithm,” in International Conference on Informatics, Electronics & Vision, 2014.

[15] Do, A. "An efficient low area implementation of 2-D DCT on FPGA." 2015 9th International Conference on Electrical and Electronics Engineering (ELECO). IEEE, 2015.

[16] S. Koryciak, A. Dabrowska-Boruch, and K. Wiatr, “Hardware Implementation of IDCT Fast Algorithms for Still Images Decompression in the JPEG Standard”, Image Processing & Communication, vol. 17, no. 4, pp. 103-108

[17] Mankar, Abhishek, Narayan Prasad, and Sukadev Meher. "FPGA implementation of discrete Fourier transform core using NEDA." Communication Systems and Network Technologies (CSNT), 2013 International Conference on. IEEE, 2013.

[18] R. Stafford. (2006) Random vectors with fixed sum. [Online]. Available: http://www.mathworks.com/matlabcentral/fileexchange/9700

[19] Emberson, Paul, Roger Stafford, and Robert I. Davis. "Techniques for the synthesis of multiprocessor tasksets." proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010). 2010.