RADAR SIGNAL PROCESSING USING COMPRESSED SENSING

Internship Report

AUTHORS

Abhinav M Balakrishnan

Arun Ramesh

INSTITUTE

School of Engineering, CUSAT

ACKNOWLEDGEMENT

This acknowledgment is a testament to the intensive drive and technical competence of many individuals who have contributed to the success of our project.

Special thanks to Shri B.S. Teza, Scientist ‘E’, ASL DRDO, for not only selecting us for this internship, but also for his consistent guidance, encouragement, and valuable insights throughout the course of our project.

We also thank our professors for their constant support and inspiration. Special thanks to our HoD, Dr. Deepa Shankar and our class co-ordinator and department faculty, Dr. Mridula S.

ABSTRACT

This project explores the application of compressed sensing techniques to radar signal processing, aiming to overcome the limitations imposed by traditional Nyquist sampling. It also presents the theoretical foundations of compressed sensing, details key reconstruction algorithms such as Orthogonal Matching Pursuit (OMP), Iterative Shrinkage Thresholding Algorithm (ISTA), and Coordinate Descent (CoD), and evaluates their performance through simulations and Monte Carlo trials. The results demonstrate the effectiveness and trade-offs of these algorithms in both noiseless and noisy environments, highlighting their potential for efficient and robust radar signal acquisition and processing. The project also explores the future possibilities and modifications for these algorithms for executing real time RADAR processing.

The GitHub repository for all the code and results can be accessed: CS Algorithms

TABLE OF CONTENTS

CHAPTER 1: NYQUIST CRITERIA AND ITS LIMITATIONS

1.1 INTRODUCTION

The Nyquist–Shannon sampling theorem is a theorem in the field of signal processing which serves as a fundamental bridge between continuous-time signals and discrete-time signals. It establishes a sufficient condition for a sample rate that permits a discrete sequence of samples to capture all the information from a continuous-time signal of finite bandwidth.

For a signal of frequency f_{signal}, the minimum sampling rate required to avoid aliasing, according to the Nyquist criterion is,

Nyquist-Shannon Sampling Criteria

$$f_{\text{s}} \geq 2f_{\text{signal}} \tag{1}$$

This means that the sampling frequency must be at least twice the highest frequency present in the signal to ensure perfect reconstruction from its samples.

1.2 LIMITATIONS

One of the main limitations of the Nyquist sampling theorem is the requirement for high sampling rates when dealing with signals that contain high-frequency components, which can be challenging to achieve in practice due to several reasons:

- **Hardware Limitations:** Analog-to-digital converters (ADCs), capable of very high sampling rates are expensive and may not be readily available. The speed and resolution of ADCs are often limited by current technology.
- **Data Storage and Processing:** High sampling rates generate large volumes of data, which require significant storage capacity and processing power. This can make real-time processing and analysis difficult or costly.
- **Power Consumption:** Systems operating at high sampling rates typically consume more power, which is a critical concern in portable or battery-powered devices.
- **Noise Sensitivity:** At higher frequencies, electronic components are more susceptible to noise and interference, which can degrade the quality of the sampled signal.

These limitations motivate the development of alternative sampling techniques, such as **Compressed Sensing**, which aim to reconstruct signals accurately from fewer samples than required by the traditional Nyquist criterion, especially when the signal is sparse or compressible in some domain.

CHAPTER 2: COMPRESSED SENSING

2.1 INTRODUCTION

The limitations of the Nyquist criterion, especially in applications requiring high data rates or operating under hardware constraints, have led to the exploration of new signal acquisition paradigms. Compressed Sensing (CS) is one such approach that leverages the sparsity of signals in some domain to enable accurate reconstruction from far fewer samples than traditionally required.

2.2 MOTIVATIONS FOR COMPRESSED SENSING

Key motivations for using compressed sensing include:

- **Efficient Data Acquisition:** CS allows for the collection of only the most informative measurements, reducing the burden on data acquisition systems.
- **Reduced Storage and Transmission Costs:** By acquiring fewer samples, CS minimizes the amount of data that needs to be stored or transmitted, which is particularly beneficial in bandwidth-limited or remote sensing scenarios.
- **Lower Power Consumption:** Fewer samples mean less processing and lower power requirements, which is advantageous for battery-powered and embedded systems.
- **Enabling New Applications:** CS opens up possibilities for applications where traditional sampling is impractical, such as medical imaging, wireless communications, and radar signal processing.

In the following chapters, we explore the principles of compressed sensing and its application to radar signal processing.

2.3 FUNDAMENTAL TERMS

Before delving deeper into compressed sensing, it is important to understand some fundamental terms:

- **Sparsity:** A signal is said to be sparse if most of its coefficients are zero or close to zero. Sparsity is a key assumption in compressed sensing.

- **Basis:** In compressed sensing, a basis is a set of vectors (such as Fourier, wavelet, or DCT bases) in which the signal can be represented as a linear combination. The choice of basis is crucial, as it determines the sparsity and thus the effectiveness of compressed sensing for a given signal. It is also called the **dictionary matrix**.
- **Measurement Matrix:** In compressed sensing, the measurement matrix is used to acquire linear projections of the original signal. It is also known as the dictionary matrix or sampling matrix.
- **Reconstruction Algorithm:** Algorithms such as Basis Pursuit, Orthogonal Matching Pursuit (OMP), and LASSO are used to recover the original sparse signal from the compressed measurements.

Understanding these terms is essential for grasping the principles and practical implementation of compressed sensing.

2.4 MATHEMATICAL MODEL

In compressed sensing, the measurement process can be mathematically modeled as:

$$\mathbf{y} = \phi \mathbf{x} \tag{2}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$ is the **original signal** (which is assumed to be sparse or compressible in some basis)
- $\phi \in \mathbb{R}^{m \times n}$ is the **measurement matrix** (with $m < n$)
- $\mathbf{y} \in \mathbb{R}^m$ is the **compressed (measurement) vector**.

If the signal **x** is not sparse in its original domain but is sparse in some transform domain (e.g., DCT, DFT, or wavelet), we can write $\mathbf{x} = \Psi \mathbf{s}$, where Ψ is the **basis matrix** and **s** is the **sparse coefficient vector**. The measurement model then becomes:

$$\mathbf{y} = \phi \Psi \mathbf{s} = \Theta \mathbf{s} \tag{3}$$

where $\Theta = \phi\Psi$ is the **sensing matrix**.

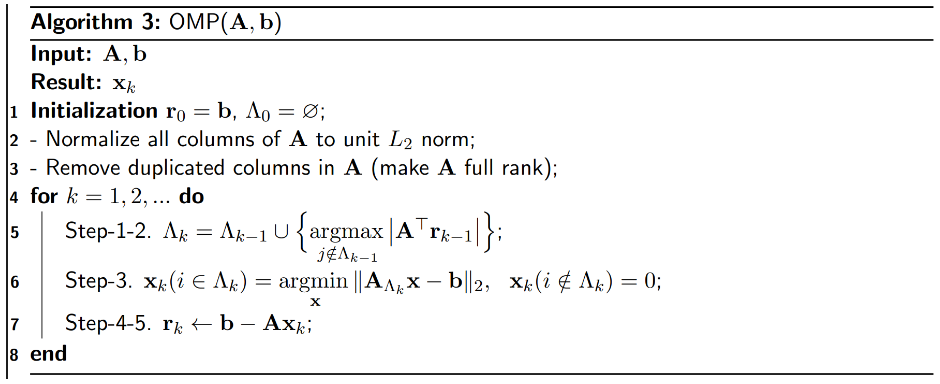
The goal of compressed sensing is to recover **x** (or **s**) from the measurements **y**, given knowledge of ϕ (and Ψ if applicable), by exploiting the sparsity of the signal.

CHAPTER 3: RECONSTRUCTION ALGORITHMS

The various algorithms are used for reconstructing back the original signal that was initially compressed by the process as shown previously.

3.1 ORTHOGONAL MATCHING PURSUIT (OMP)

The OMP algorithm is an iterative greedy algorithm used to recover sparse signals from compressed measurements. At each iteration, it selects the column of the measurement matrix that is most correlated with the current residual and updates the solution accordingly. The process continues until a sufficiently small residual is met. The steps are listed below, as shown below



```
Algorithm 3: OMP($\mathbf{A}, \mathbf{b}$)
Input: $\mathbf{A}, \mathbf{b}$
Result: $\mathbf{x}_k$
Initialization $\mathbf{r}_0 = \mathbf{b}$, $\Lambda_0 = \varnothing$;
- Normalize all columns of $\mathbf{A}$ to unit $L_2$ norm;
- Remove duplicated columns in $\mathbf{A}$ (make $\mathbf{A}$ full rank);
for $k = 1, 2, \ldots$ do
    Step-1-2. $\Lambda_k = \Lambda_{k-1} \cup \left\{ \underset{j \notin \Lambda_{k-1}}{\operatorname{argmax}} \left| \mathbf{A}^\top \mathbf{r}_{k-1} \right| \right\}$;
    Step-3. $\mathbf{x}_k(i \in \Lambda_k) = \underset{\mathbf{x}}{\operatorname{argmin}} \|\mathbf{A}_{\Lambda_k}\mathbf{x} - \mathbf{b}\|_2, \quad \mathbf{x}_k(i \notin \Lambda_k) = 0$;
    Step-4-5. $\mathbf{r}_k \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x}_k$;
end
```

Figure 1: OMP Algorithm[**omp-intro**]

This algorithm can be implemented in MATLAB and Python with necessary toolboxes and libraries.

3.1.1 Algorithm Implementation

- **MATLAB**

Here, The code is divided into two main parts: the OMP function definition and the signal generation/reconstruction workflow. The omp function implements the OMP algorithm. It takes as input a measurement matrix A, a measurement vector b, and the sparsity level K. The function normalizes the columns of A and initializes the residual r as the measurement vector. It iteratively selects the column of A most correlated with the current residual, adds its index to the support set Lambda, and solves a least-squares problem to update the estimated sparse vector x. The residual is updated accordingly. This process repeats for K iterations, corresponding to the assumed sparsity of the original signal. For ideal omp implementation, n,m,k are defined and a random gaussian sensing matrix is created. then a sparse signal is created in frequency domain and is multiplied with the sensing matrix to create the measurements(output). Now, the omp algorithm is used to recover the frequency domain signal back and the rreconstructed and original signals are plotted together to check for errors. For noisy implementation, an additional gaussian noise is generated and added to the output measurement vector and this noisy output is given to the omp algorithm. Then a sum of random time domain sinusoidal input was given with Ψ as a random gaussian matrix and ϕ as a inbuilt function for dft, dftmtx(). dft was taken instead of idft since the omp was calculated in the frequency domain. the ouput was then converted to time domain using ifft() and was plotted.

- **Python**

  Libraries like **numpy** and **matplotlib** are imported for mathematical operations and plotting results respectively.

  - **Stage 1:** The basic implementation was done by taking length of signal (n), number of measurements (m) and non-zero values or sparsity (k) as input. The sensing matrix was assumed to be filled with random gaussian values.
  - **Stage 2:** The next stage involved taking a sum of three sinusoidal signals as input signal (k = 3) and it is converted to a more sparser domain with **Discrete Cosine Transform (DCT)**. The function is used by importing the **scipy** library. While initially k was fixed, it is then taken as an input from user. DCT was initially tested for a single sine wave as well as for sum of sine waves of different frequencies, as shown in the figure below.
  - **Stage 3:** In the above stages, reconstruction was observed for pure signals. So, a noise (in dB) was introduced before the reconstruction process.

All these stages were plotted and the error was calculated and observed.

3.1.2 Monte-Carlo Trials

Monte Carlo trials are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The is used to understand the behaviour of the algorithm under change in parameters, including sparsity, noise and number of measurements taken. It is useful for analysis, understanding its performance for various values of multiple inputs. In other words, this acts like a testbench for the algorithm.

The input values were stored in a list and these were fed to the trial algorithm. The error was calculated for a number of trials for the same input values, and only the average error is plotted to prevent unwanted variations in reconstruction.

3.1.3 Observations & Results

- **MATLAB**

The algorithm was initially tested directly in frequncy domain. In its ideal form(ie. without noise), for a low enough sparsity, the algorithm perfectly reconstructed the frequency and the amplitude values of compressed signal. Then, two values of noise was given(SNR=0dB and SNR=20dB). The Algorithm was able to reconstruct the signal near-perfectly for an SNR of 20 dB. For an SNR of 0 dB(signal power=noise power), the results were more inaccurate, both in terms of position on the graph(frequency) and the amplitude values.The algorithm was able to reconstruct some parts of the signal with a fair amount of accuracy.

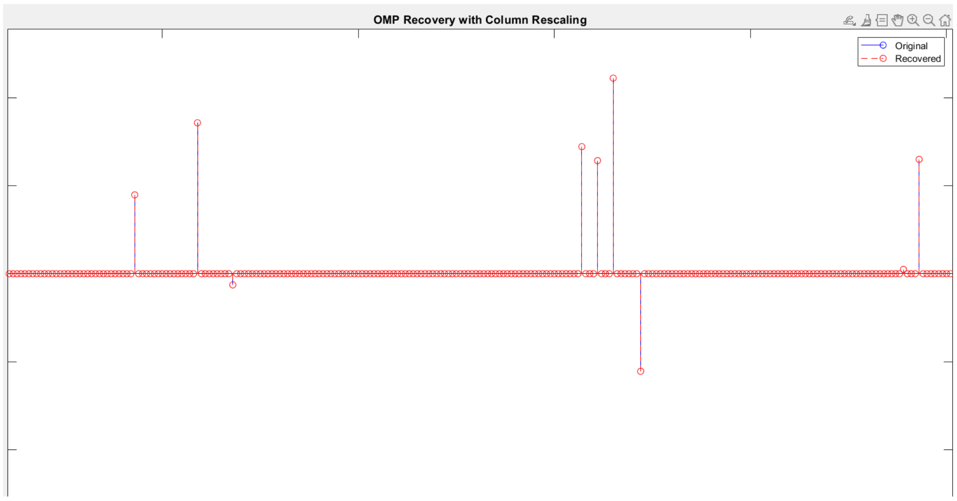


Figure 2: OMP Signal Reconstruction:Ideal (No noise added)

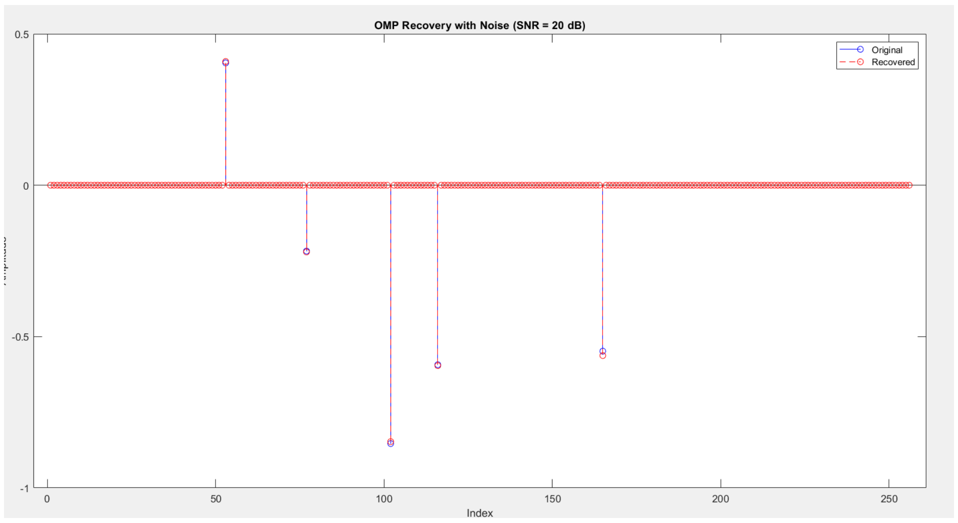


Figure 3: OMP Signal Reconstruction: 20dB

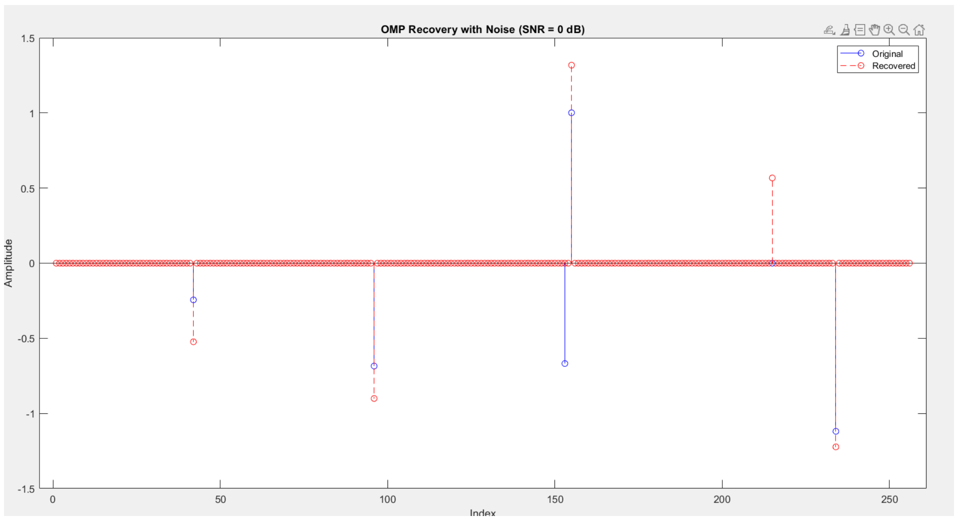


Figure 4: OMP Signal Reconstruction: 0dB

Next, the code was used to implement sinusoids in time domain. A sum of 5 real sinusoids was given as input to the algorithm. Then the output is plotted along with the original signal to compare them. Initially, the algorithm gave an output which had its amplitude greatly decreased w.r.t the original signal.

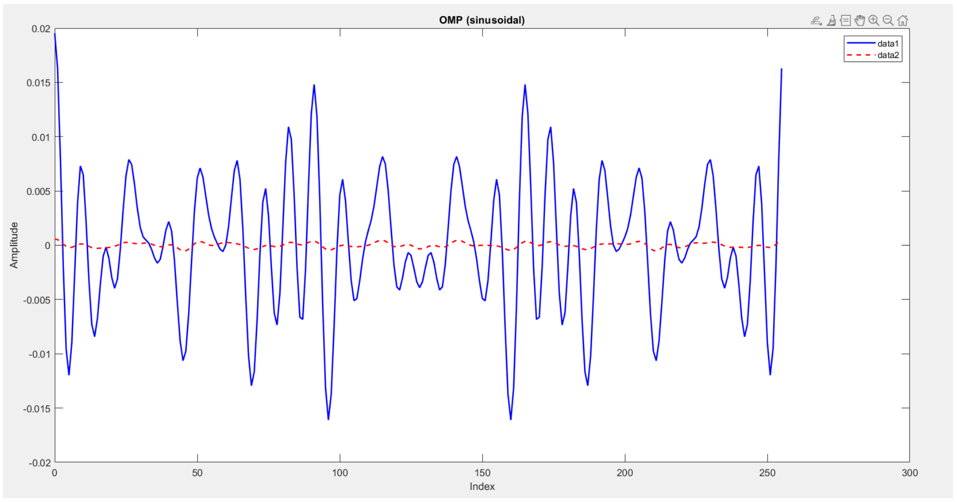


Figure 5: OMP Signal Reconstruction: 0dB

Further testing of the code showed the importance of normalising the theta matrix. The normalised theta matrix showed the correct output. After this was resolved, the algorithm was able to reconstruct the signal fairly accurately. Smaller peaks of the input signal was harder to reconstruct for the algorithm, and also, there was a reduction in the amplitude of the reconstructed signal w.r.t the original signal.A slight phase shift was observed in some outputs when the code is run for different random inputs.

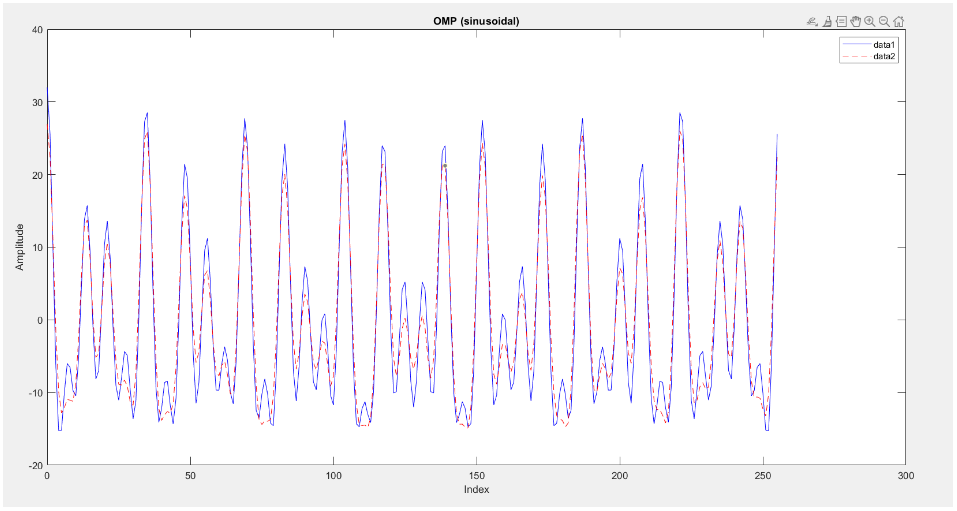


Figure 6: OMP Signal Reconstruction: sinusoidal input

- **Python**

For **Stage 1** implementation, the sparse matrix is already created by specifying k. So, the compressed matrix (y) is generated by just multiplying sensing matrix (Θ) and the generated sparse matrix (s). The results are plotted as shown below,

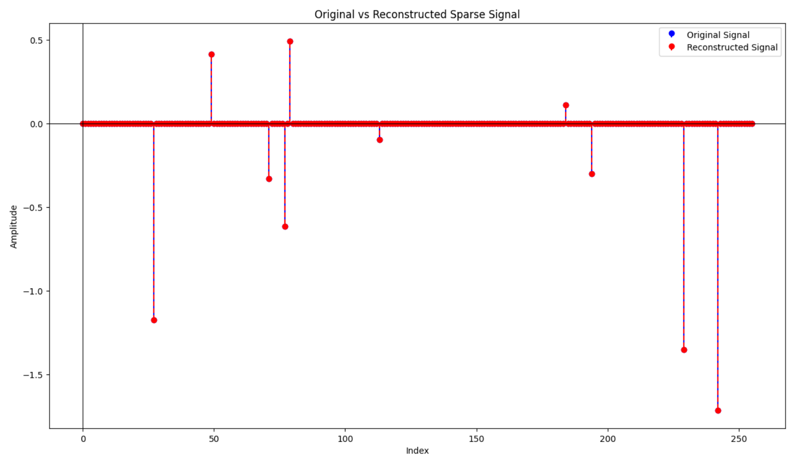


Figure 7: OMP Algorithm Stage 1 Implementation: Perfect Reconstruction

While the reconstruction as shown above is very accurate, it is not always the case. As sparsity increases, the measurements to be taken also increases. Hence, there are some necessary conditions for perfect recovery of a signal. As mentioned in [**rani-cs**], the relation between n, m and k is:-

$$\boxed{m \geq C \cdot k \cdot \log\left(\frac{n}{k}\right)} \tag{4}$$

where **C** is a constant almost equal to 2.

Hence, if the above equation is not satisfied, then reconstruction is very difficult. The failed reconstruction is shown below.

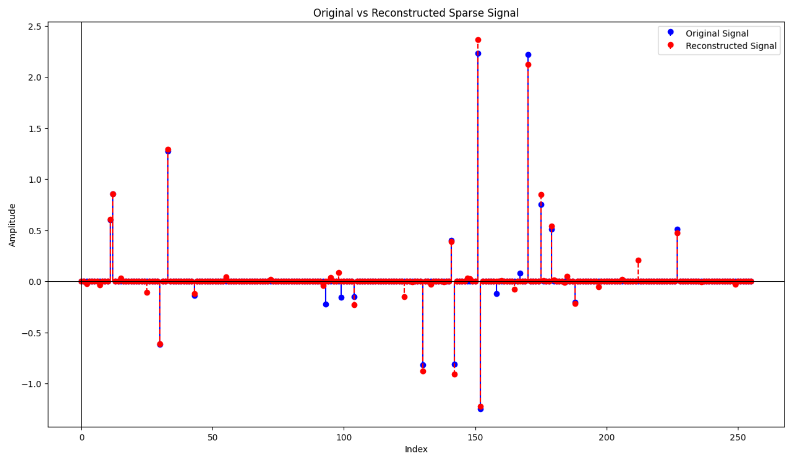


Figure 8: OMP Algorithm Stage 1 Implementation: Failed Reconstruction

When it comes to **Stage 2** implementation, the sensing matrix is divided into a basis matrix and measurement matrix. The basis matrix is used to convert our input signal to a sparser signal. For sinusoidal inputs, it is best to represent the signals in its frequency domain. So, FFT or DCT can be used. Since, all sinusoids are real signals, DCT was possible. The sum of sinusoids were converted to DCT and the results are being plotted to check its sparsity.

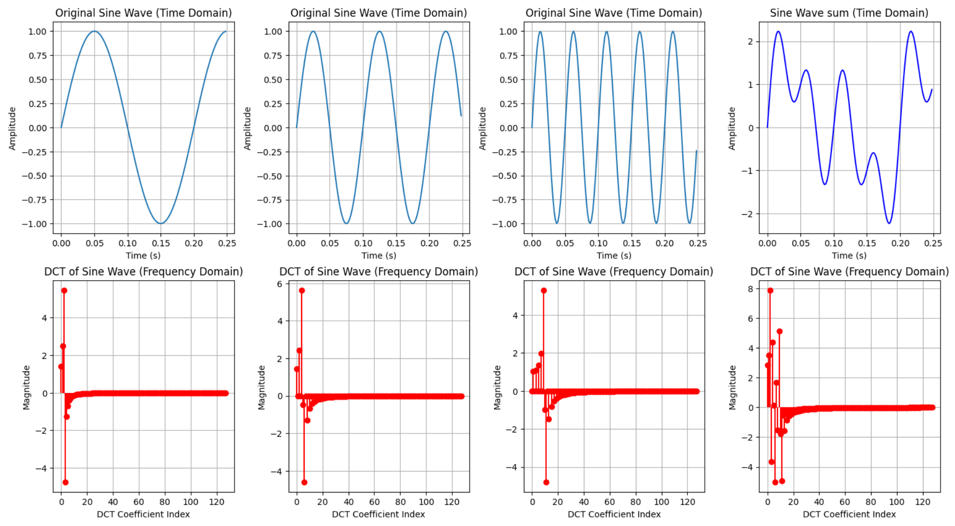


Figure 9: OMP Algorithm Stage 2 Implementation: DCT Basis on Sinusoidal signals

The sinusoidal signal is initially tested for various values of n and m, keeping k = 3. Some of the results are plotted as shown,

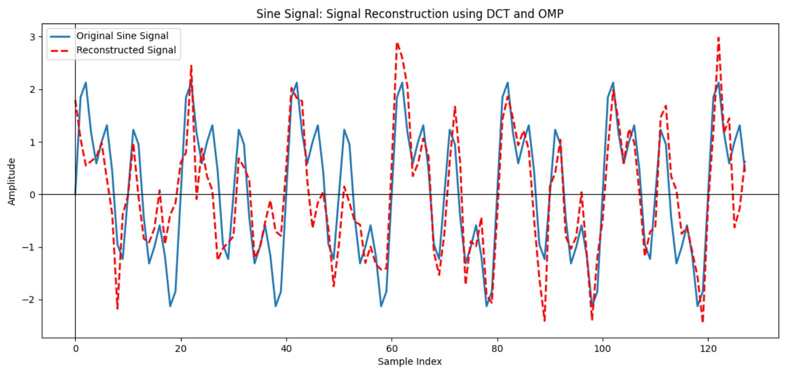


Figure 10: OMP Algorithm Stage 2 Implementation: For n = 128, m = 60

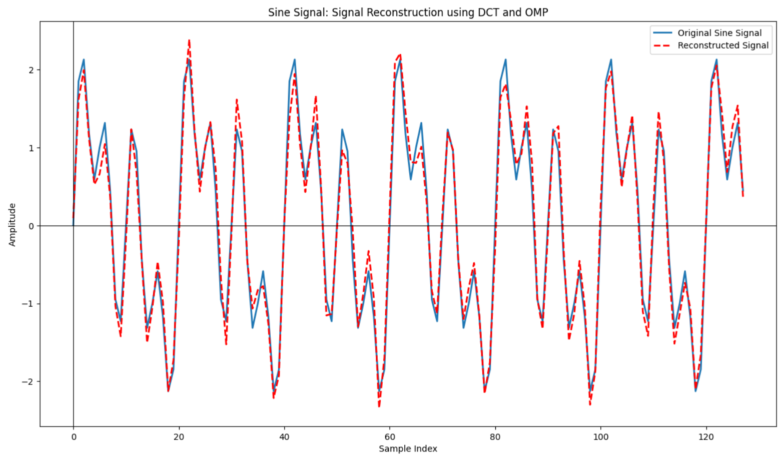


Figure 11: OMP Algorithm Stage 2 Implementation: For n = 128, m = 100

So, generally we can say as **number of measurements increases, the reconstruction error decreases**. Till now, no noise has been considered during the reconstruction. To analyse the algorithm for each value of n, m, k and even noise, it is difficult for us to understand the trend of error. So, a Monte Carlo trial has been implemented on the OMP algorithm for three variable parameters, **measurements**, **sparsity** and **noise**. So, all three parameters are compared and the results are plotted.

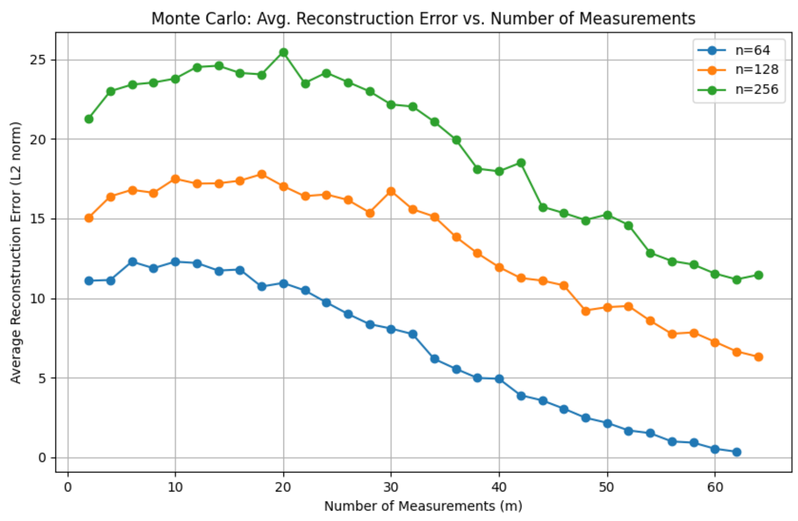


Figure 12: Monte Carlo Trial: Measurements (m)

The analysis above is for noiseless, fixed sparsity (k = 3) reconstruction.

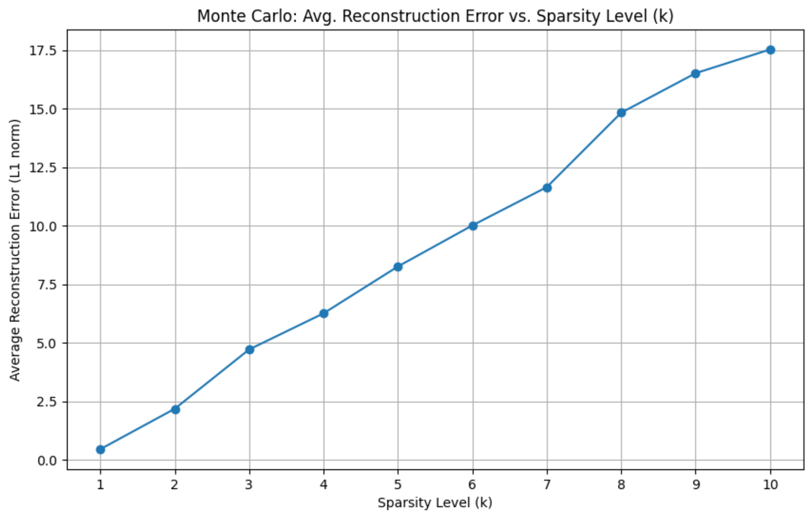


Figure 13: Monte Carlo Trial: Sparsity (k)