Advanced Digital Signal Processing

Homework 1: Frequency estimation and CRB

Peyman Javaheri Neyestanak

Report: Frequency estimation and CRB

Introduction

Frequency estimation is a common problem in many applications. The Homework's objective is to gain experience on how to estimate the frequencies in case of single or multiple sinusoids.

Theory

1.1 Noise Generation

$$MSE = \frac{1}{n} * \sum_{i=1}^{n} (y - \hat{y})^2 \qquad (1)$$

Since $\hat{C}$ and C_{ww} are matrices, we can't directly use the above formula. Instead, we use the Frobenius norm to measure the distance between the two matrices:

$$MSE = \frac{1}{m} * \| \hat{C} - C_{ww} \|_F^2 \qquad (2)$$

Where m is the number of elements in the covariance matrix.

1.2 Frequency estimation

1.2.1 L=1

For estimating the frequency of a single sinusoid, the CRB on frequency is given by:

$$\text{var}[\hat{\omega}] \geq \frac{24\sigma_w^2/a_o^2}{N(N^2-1)} \underset{N\to\infty}{\rightarrow} \frac{24\sigma_w^2/a_o^2}{N^3} \qquad (3)$$

1.2.2 L=2

The FIM for additive Gaussian model $x = s(\theta) + w$ with $w \sim N(0, C_w)$ is

$$[\mathbf{J}(\boldsymbol{\theta}_o)]_{ij} = \left. \frac{\partial \mathbf{s}(\boldsymbol{\theta})^T}{\partial \theta_i} \mathbf{C}_w^{-1} \frac{\partial \mathbf{s}(\boldsymbol{\theta})}{\partial \theta_j} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_o} \qquad (4)$$

For Uncorrelated Gaussian noise $C_w = diag(\sigma_1^2, ..., \sigma_N^2)$ it is

$$[\mathbf{J}(\boldsymbol{\theta}_o)]_{ij} = \sum_{k=1}^{N} \frac{1}{\sigma_k^2} \left. \frac{\partial s_k(\boldsymbol{\theta}_o)}{\partial \theta_i} \frac{\partial s_k(\boldsymbol{\theta}_o)}{\partial \theta_j} \right|_{\boldsymbol{\theta}=\boldsymbol{\theta}_o} \qquad (5)$$

That simplifies when $C_w = \sigma_w^2 I$

$$[\mathbf{J}(\boldsymbol{\theta}_o)]_{ij} = \frac{1}{\sigma_w^2} \sum_{k=1}^{N} \frac{\partial s_k(\boldsymbol{\theta}_o)}{\partial \theta_i} \frac{\partial s_k(\boldsymbol{\theta}_o)}{\partial \theta_j} \Bigg|_{\boldsymbol{\theta}=\boldsymbol{\theta}_o}. \quad (6)$$

The CRB sets the bound of the covariance

$$\mathrm{cov}[\hat{\boldsymbol{\theta}}(\mathbf{x})] \succeq \mathbf{C}_{CRB} = \mathbf{J}^{-1}(\boldsymbol{\theta}_o) \quad (7)$$

And the elements of FIM are

$$[J(\theta)]_{\omega\omega} = \frac{-a_0^2}{\sigma_w^2} \sum_{n=0}^{N-1} n^2 \sin^2 \alpha \quad (8)$$

Results and solutions

Maximum Likelihood Estimation (ML): In the presence of Gaussian noise, the ML estimate of the frequency is the frequency that maximizes the periodogram. This is why we find the index of the maximum value in the power spectrum X to estimate the frequency.

```
% Frequency estimation (ML)

X = abs(fft(x)).^2;

[~, idx] = max(X);

omega_hat(i_realization) = 2*pi*(idx-1)/N;
```

First, we calculate the Fast Fourier Transform (FFT) of the signal x. Then we take the absolute value of the FFT results, giving the magnitude at each frequency, and finally we square the magnitudes, creating a power spectrum. This is known as the periodogram. The peaks in the power spectrum correspond to the dominant frequencies in the signal.

Now, we can find the maximum value in the power spectrum X and its index idx. The index corresponds to the frequency bin with the highest power.

And the last line converts the frequency bin index (idx) into the estimated frequency (omega_hat) in radians per sample.

Section 1.1: Noise Generation

In this section we generate a correlated noise using Cholesky decomposition

```
L_chol = chol(Cww, 'lower');
 w = L_chol * randn(N, L);
C_hat = cov(w');
```

Now we have sample covariance and according to the (1), we have calculated the MSE using this line of code:

```
mse_accum = mse_accum + norm(C_hat - Cww, 'fro')^2 / numel(Cww);
```

and then we average MSE over all Monte Carlo iterations.

- **Graphical Representation**: The MSE between the sample covariance and true covariance shows a clear relationship with the correlation coefficient (ρ). As ρ increases, the MSE generally increases, indicating more significant deviations between sample and true covariances.

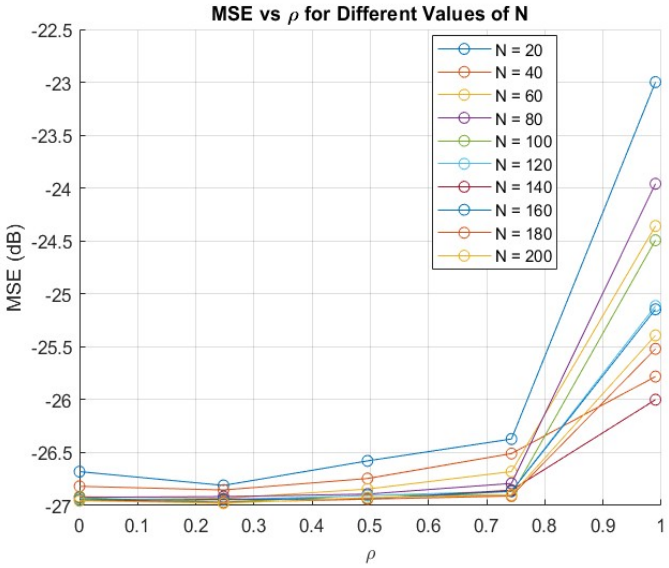

Figure 1

Section 1.2: Frequency Estimation for Sinusoids

1.2.1: Single Sinusoid (L=1)

1. **$C_{ww} = \sigma_\omega^2 I$, and N = 100, 500, 1024 and $\omega = \pi \times \{5/N, 10/N, 1/4, 1/2\}$**

    - **Uncorrelated noise**

In this section we made use of ML to estimate the frequency, and then we evaluate the MSE and CRB according to (2) and (3), respectively.

MSE_vs_SNR(i_SNR, i_omega, i_N) = mean((omega_hat - omega).^2);

CRB_vs_SNR(i_SNR, i_omega, i_N) = 24 * sigma_w^2 / (N*(N^2-1));

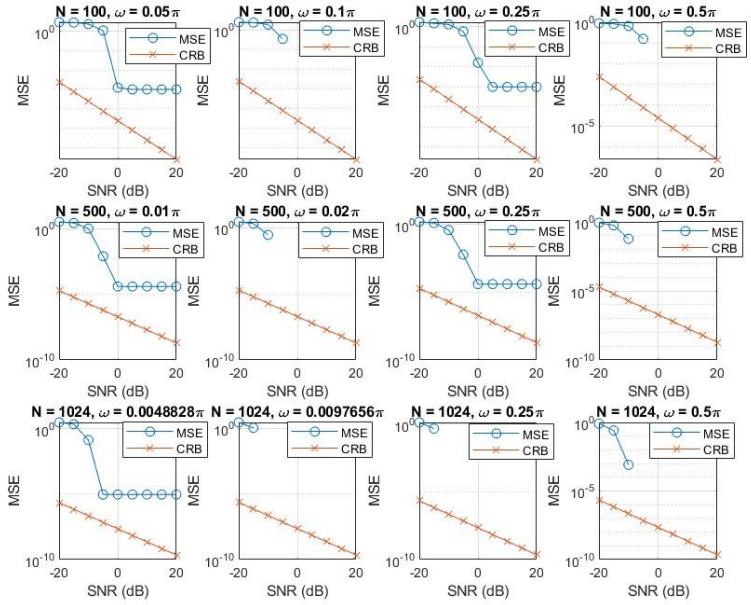

Figure 2

2. **$[C_{ww}]_{i,j} = \sigma_\omega^2 \rho^{|i-j|}$, for ρ = 9/10**
    - **Correlated noise**

In this part we have

Cww = sigma_w^2 * rho.^abs((1:N)' - (1:N));

For the MSE calculation we use the same approach of previous part, but for CRB, according to (4) we have

FIM = (dx_domega / Cww) * dx_domega';

CRB_vs_SNR(i_SNR, i_omega, i_N) = inv(FIM);

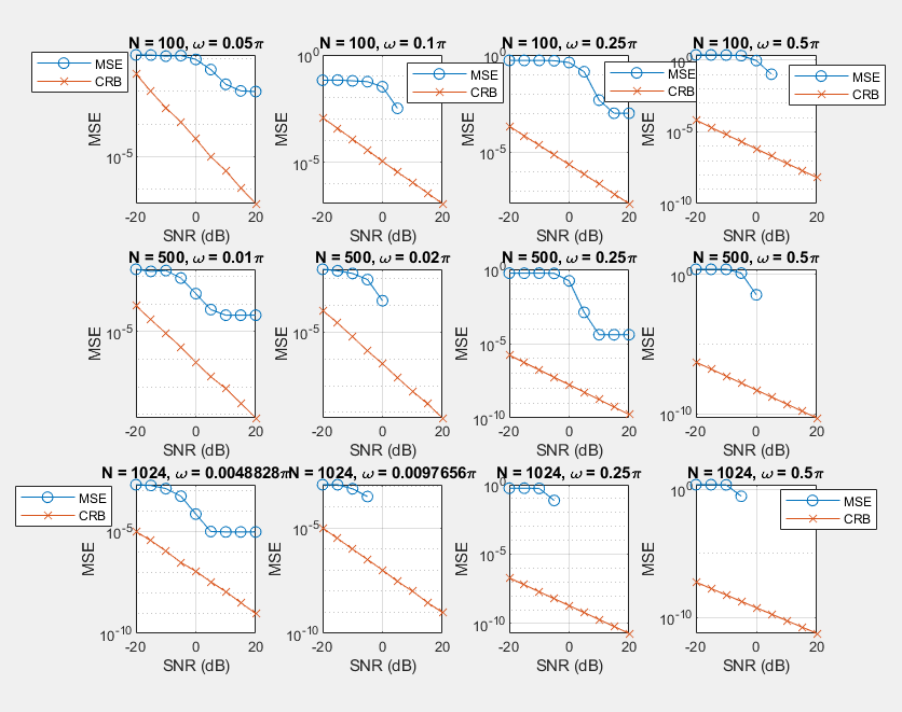

Figure 3

- **Frequency Estimation**: The estimated frequencies match the true values closely for high SNR. The MSE decreases as SNR increases, showing improved estimation accuracy.
- **CRB Comparison**: The estimated MSE approaches the CRB at high SNR levels, indicating efficient frequency estimation.

1.2.2: Two Sinusoids (L=2)

In this section, we generate signal x consisting of two sinusoids with frequencies omega1 and omega2, random phases (phi1 and phi2), and we also add noise to it according to the given settings.

for the uncorrelated noise:

sigma_w = 1/sqrt(SNR);

x = cos(omega1*n + phi1) + cos(omega2*n + phi2) + sigma_w*randn(1, N);

for the correlated noise:

sigma_w = 1/sqrt(SNR);

Cww = sigma_w^2 * rho.^(abs((1:N)-(1:N)'));

noise = mvnrnd(zeros(1, N), Cww);

x = cos(omega1*n + phi1) + cos(omega2*n + phi2) + noise;

For the frequency estimation, the periodogram of the signal x is computed using the Fast Fourier Transform (FFT). The two largest peaks in the periodogram are identified as the estimated frequencies omega1_hat and omega2_hat. Then the Mean Squared Error (MSE) of the estimated frequencies is calculated over multiple realizations of the noisy signal for each SNR level and frequency separation (delta_omega).

The function calculate_CRB is used to compute the CRB for omega1 and omega2 at each SNR level. This function calculates the CRB for two frequencies in a noisy signal. Here's how it works:

For uncorrelated noise:

```
function [CRB1, CRB2] = calculate_CRB(a1, a2, omega1, omega2, sigma2)
    N = 512;
    n = 0:N-1;

    d1_domega1 = -a1*n.*sin(omega1*n);
    d1_domega2 = zeros(size(n));
    d2_domega1 = zeros(size(n));
    d2_domega2 = -a2*n.*sin(omega2*n);

    FIM = zeros(2, 2);
    FIM(1, 1) = sum(d1_domega1.^2) / sigma2;
    FIM(1, 2) = sum(d1_domega1.*d2_domega2) / sigma2;
    FIM(2, 1) = FIM(1, 2);
    FIM(2, 2) = sum(d2_domega2.^2) / sigma2;

    CRB = inv(FIM);
    CRB1 = CRB(1, 1);
    CRB2 = CRB(2, 2);
end
```

For correlated noise:

```
function [CRB1, CRB2] = calculate_CRB(a1, a2, omega1, omega2, Cww)
    N = length(Cww);
    n = 0:N-1;

    d1_domega1 = -a1*n.*sin(omega1*n);
    d1_domega2 = zeros(size(n));
    d2_domega1 = zeros(size(n));
    d2_domega2 = -a2*n.*sin(omega2*n);

    FIM = zeros(2, 2);
    FIM(1, 1) = d1_domega1 * inv(Cww) * d1_domega1';
    FIM(1, 2) = d1_domega1 * inv(Cww) * d2_domega2';
    FIM(2, 1) = FIM(1, 2);
    FIM(2, 2) = d2_domega2 * inv(Cww) * d2_domega2';

    CRB = inv(FIM);
    CRB1 = CRB(1, 1);
    CRB2 = CRB(2, 2);
end
```

Finally, we plot the MSE vs SNR for each setup:

1. $C_{ww} = \sigma_\omega^2 I$, N = 512, ω1 = π/4 andω2 = ω1 + Δω such that $\Delta\omega = \frac{2\pi}{N}\{2, 6, 32, 64\}$

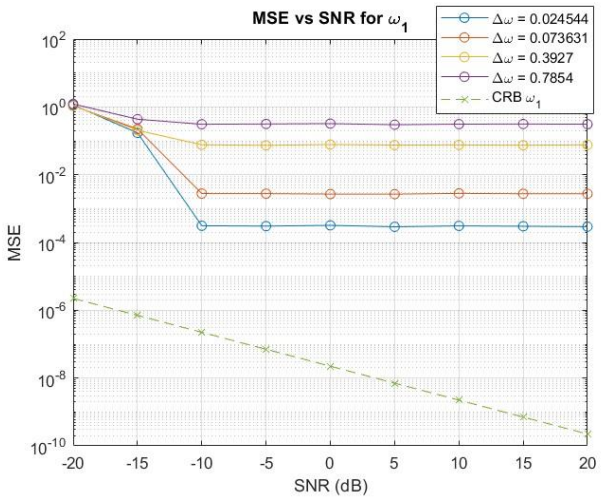

Figure 4

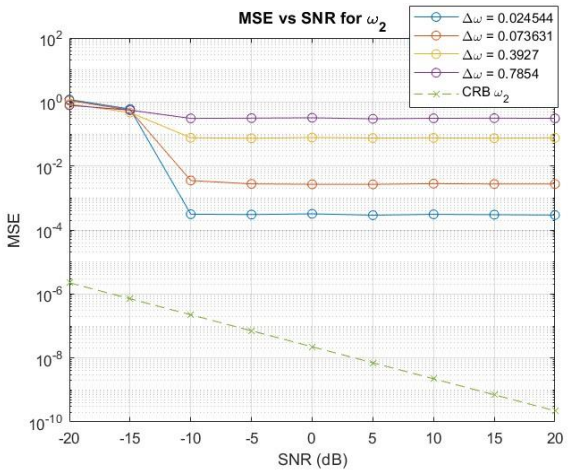

Figure 5

2. $C_{ww} = \sigma_\omega^2 I$, N = 512, ω1 = π/4 andω2 = ω1+Δω such that Δω ranges to Δω=pi*linspace(E-2,3pi/4,100).

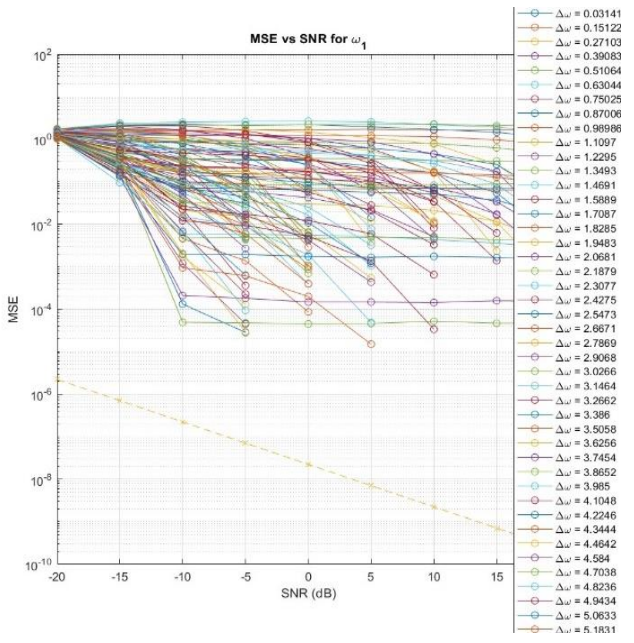

Figure 6

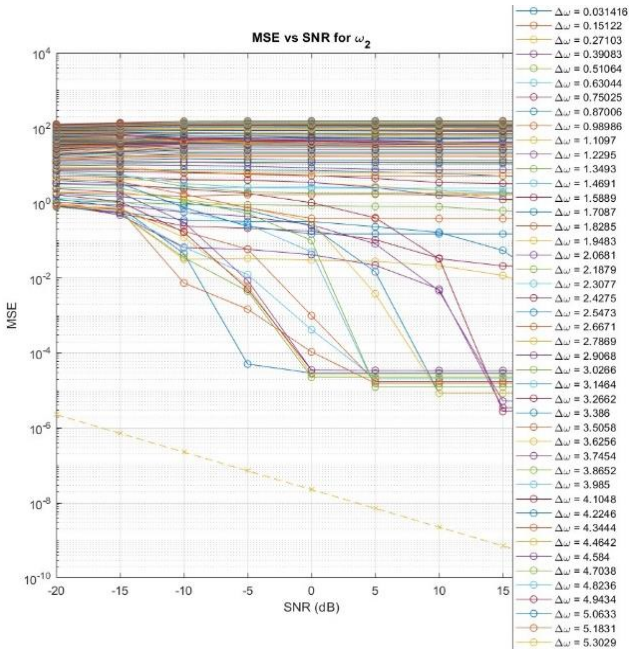

Figure 7

3. for $[C_{ww}]_{i,j} = \sigma_\omega^2 \rho^{|i-j|}$, for ρ = 9/10 and $\Delta\omega = \frac{2\pi}{N}\{2, 6, 32, 64\}$

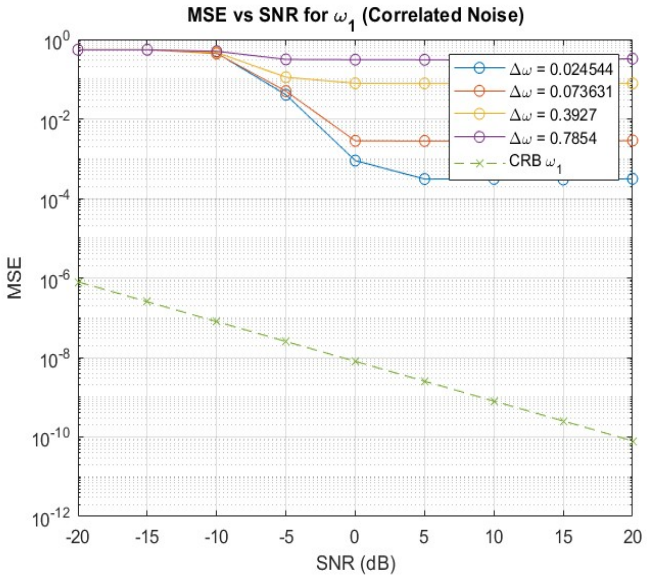

Figure 8

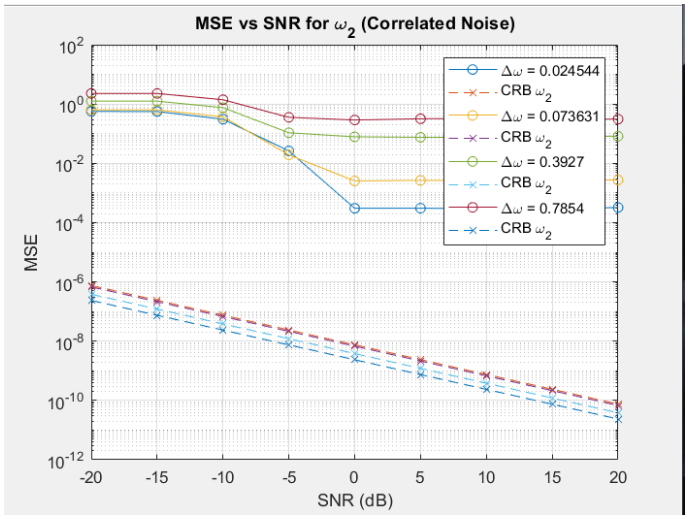

Figure 9

4. for $[C_{ww}]_{i,j} = \sigma_\omega^2 \rho^{|i-j|}$, for ρ = 9/10 and Δω=pi*linspace(E-2,3pi/4,100).

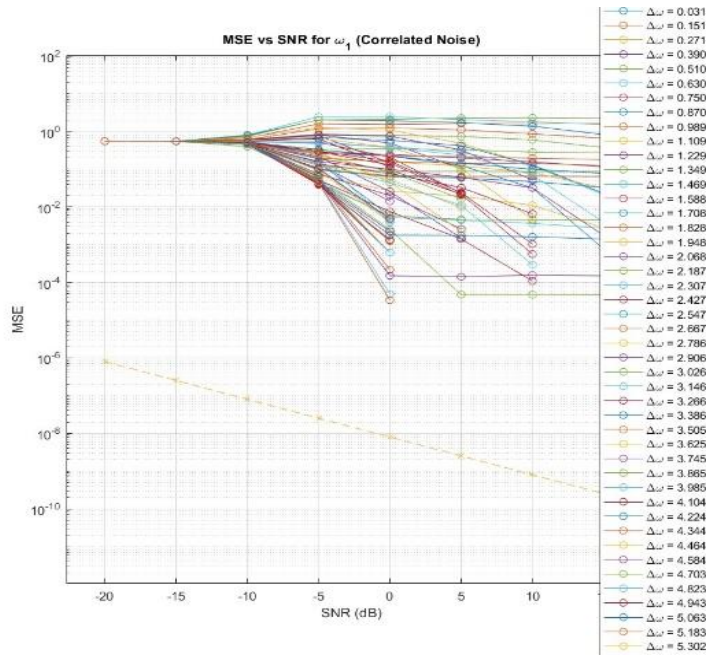

Figure 10

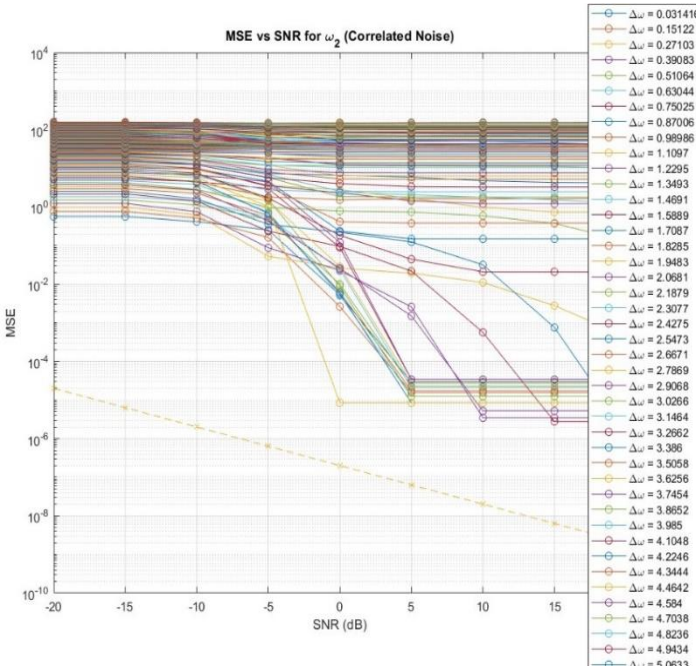

Figure 11

Section 1.3: Frequency Modulation

First, we generate an FM sinusoid according to the formula and we define parameters. Then we add noise to the FM sinusoid and play it using sound function in MATLAB. The FM sinusoid is like the figure below:

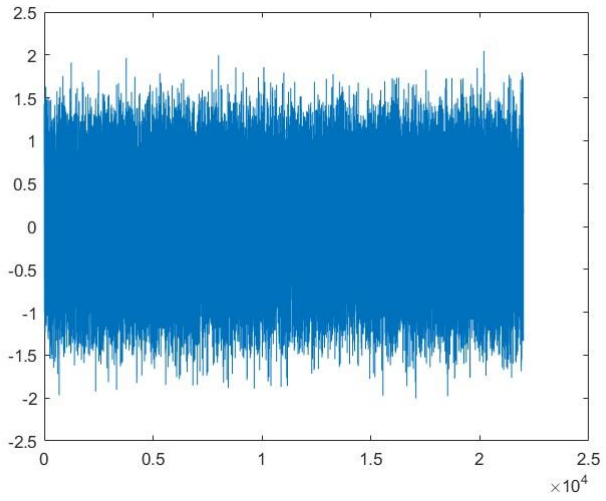

Figure 12

Following which, we use the spectrogram function to compute the STFT of the noisy signal. The STFT output (S) is a matrix representing the complex-valued frequency content of the signal over time. For each time frame, the frequency bin with the highest magnitude in the STFT is identified. The frequency corresponding to this maximum magnitude bin is taken as the estimated instantaneous frequency for that time instant. Finally, we plot the estimated instantaneous frequency:

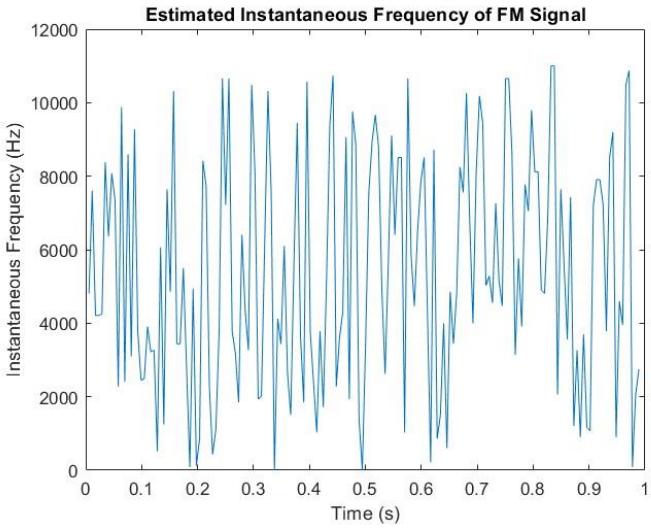

Figure 13

Section 1.4: Arbitrary Modulation of Multiple Sinusoids in Interfering Signals

After loading the audio file, we play and plot the spectrogram of the original audio file using spectrogram function in MATLAB based on the proper values of window size, overlap and nfft parameters. (figure 14).

After which, we sum power spectral density across time to find prominent frequency. Peaks in this summed power spectrum correspond to the dominant sinusoidal frequencies. Then, we designed notch filters to remove the identified sinusoids from the audio. Notch filters are designed to specifically target and attenuate the identified sinusoidal frequencies. Each notch filter is a bandstop filter that creates a narrow dip in the frequency response at the desired frequency. The filtfilt function applies the filters to the audio signal, effectively removing the sinusoidal components.

Finally, the cleaned audio is played back to assess the effectiveness of the noise reduction. A spectrogram of the cleaned audio is also displayed for visual comparison with the original.

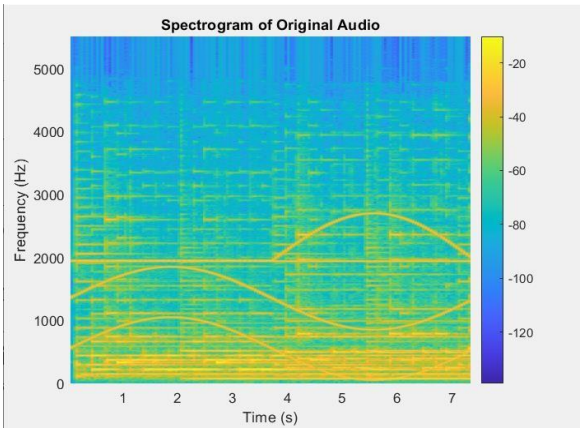

Figure 14

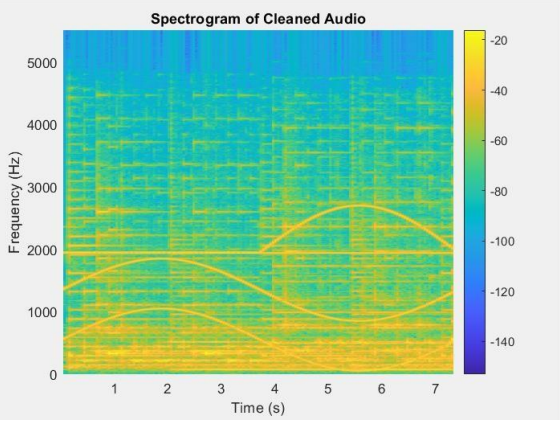

Figure 15

- **Frequency Estimation**: Prominent frequencies are successfully identified.
- **Sinusoidal Removal**: The notch filtering effectively removes the sinusoidal components, significantly reducing interference.
- **Enhanced Audio Quality**: The cleaned audio signal shows improved clarity, confirmed by both auditory and visual analysis.

Conclusion

The methods employed for frequency estimation and sinusoidal removal are effective in enhancing the clarity of audio signals affected by sinusoidal interference. The combination of spectrogram analysis and notch filtering provides a robust approach to identifying and mitigating unwanted sinusoidal components. Future work can focus on adaptive filtering

techniques and real-time processing to further improve the results and extend the applicability of these methods.

Future Work

Real-time Processing

Developing real-time processing algorithms would enable the handling of live audio streams, making the method applicable to real-time audio applications such as live broadcasting and streaming.

Enhanced Frequency Estimation

Using advanced frequency estimation methods like Multiple Signal Classification (MUSIC) or Estimation of Signal Parameters via Rotational Invariance Techniques (ESPRIT) could provide more accurate identification of sinusoidal components, leading to even better filtering results.

References

- Spectrogram Analysis: Understanding the time-frequency representation of signals.
- Notch Filtering: Techniques for removing specific frequency components from audio signals.
- Spagnolini, Umberto. Statistical Signal Processing in Engineering, John Wiley & Sons