POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

2024-2025 SP&L Homework

APPOURCHAUX Léo, YE Yumeng

Advisor:
Prof. Umberto Spagnolini

1. Homework 1: Frequency Estimation and CRB

1.1. Sect. 1: Theoretical Tools and Methods Adopted

Homework 1 focuses on estimating the frequency of sinusoidal signals under various noise conditions, comparing performance to the Cramér-Rao Bound (CRB). The tasks involve noise generation, frequency estimation, and frequency modulation analysis, solved using the following methods:

- **Noise Generation:** We generate zero-mean Gaussian noise with an exponentially decaying correlation structure, defined by the covariance matrix $\mathbf{C}_{ww}$ where $[\mathbf{C}_{ww}]_{i,j} = \sigma_w^2 \cdot \rho^{|i-j|}$. We apply the Cholesky decomposition to factor the covariance matrix $\mathbf{C}_{ww}$ into $\mathbf{Q}\mathbf{Q}^T$, where $\mathbf{Q}$ is lower triangular. Using this, we generate correlated noise $\mathbf{w} = \mathbf{Q}\mathbf{g}$ from uncorrelated Gaussian noise $\mathbf{g} \sim \mathcal{N}(0, \mathbf{I})$, ensuring $\mathbf{w}$ matches the theoretical covariance $\mathbf{C}_{ww}$. The sample covariance $\hat{\mathbf{C}} = \frac{1}{L}\mathbf{w}\mathbf{w}^T$ is estimated and compared to the theoretical one $\mathbf{C}_{ww}$ using the Mean Squared Error (MSE) to assess accuracy.

- **Frequency Estimation:** For a noisy sinusoid $y[n] = A\cos(\omega_0 n + \phi_0) + w[n]$, we use:
  1. **FFT Peak Detection:** The Fast Fourier Transform (FFT) computes the periodogram, and the frequency at the maximum peak is selected.
  2. **Parabolic Interpolation:** Refines the estimate by fitting a parabola around the FFT peak, improving accuracy beyond FFT resolution.
  3. **Padding and Oversampling:** Zero-padding the signal before computing the FFT increases the frequency resolution by providing a finer grid of frequency bins, aiding in precise peak detection.
  4. **Monte Carlo**: We average the MSE over numerous realizations with varying noise, ensuring a reliable and smoothed evaluation of the frequency estimators.
  5. These methods are tested with or without filter ($h[n] = \delta[n]$ or $H(z) = \frac{2}{1+0.9z^{-1}}$), and with correlated or uncorelated noise ($\mathbf{C}_{ww} = \mathbf{I}$ or $[\mathbf{C}_{ww}]_{i,j} = \sigma_w^2 \cdot \rho^{|i-j|}$ with $\rho = 0.9$), with performance evaluated against the CRB.

- **Frequency Modulation:** For a signal $x[n] = a\cos(\gamma n^2 + \phi) + w[n]$, we estimate the time-varying frequency using:
  1. **Spectrograms:** Visualize the instantaneous frequency $\omega(n) = 2\gamma n$
  2. **Maximum Likelihood Estimation (MLE):** Estimates the modulation parameter γ by minimizing the residual error over a grid search.

1.2. Sect. 2: Solution, Kernel of MATLAB Code, and Results

1.2.1 Noise Generation

Code:

```
% Parameters
N_range = linspace(20,200,10);
sigma_w = 1;
rho_range = [0, 0.5, 0.99];
L = 100;
MONTE = 1000;

% Loop over rho and N
for j = 1:length(rho_range)
    rho = rho_range(j);
    for i = 1:length(N_range)
        dim = N_range(i);
        Cww = sigma_w^2 * rho.^abs((1:dim)' - (1:dim)); % Theoretical covariance
        Q = chol(Cww, 'lower'); % Cholesky decomposition
        for t = 1:MONTE
            w = Q * randn(dim, L); % Generate correlated noise
            C_hat = (w * w') / L; % Sample covariance
            MSE(j,i) = MSE(j,i) + norm(C_hat - Cww, 'fro')^2 / dim^2; % Accumulate MSE
        end
        MSE(j,i) = MSE(j,i) / MONTE; % Average MSE
    end
end
```

Explanation: We compute the average MSE over $MONTE$ noises. The Frobenius norm is the same as the Euclidean distance. We could have put `sum((C_hat - Cww).^2)` instead of `norm(C_hat - Cww, 'fro')^2`.

Results: The plot of MSE vs. N (Fig. 1) shows that MSE decreases with larger N, with a higher ρ producing larger errors due to increased correlation (more coefficients to estimate).

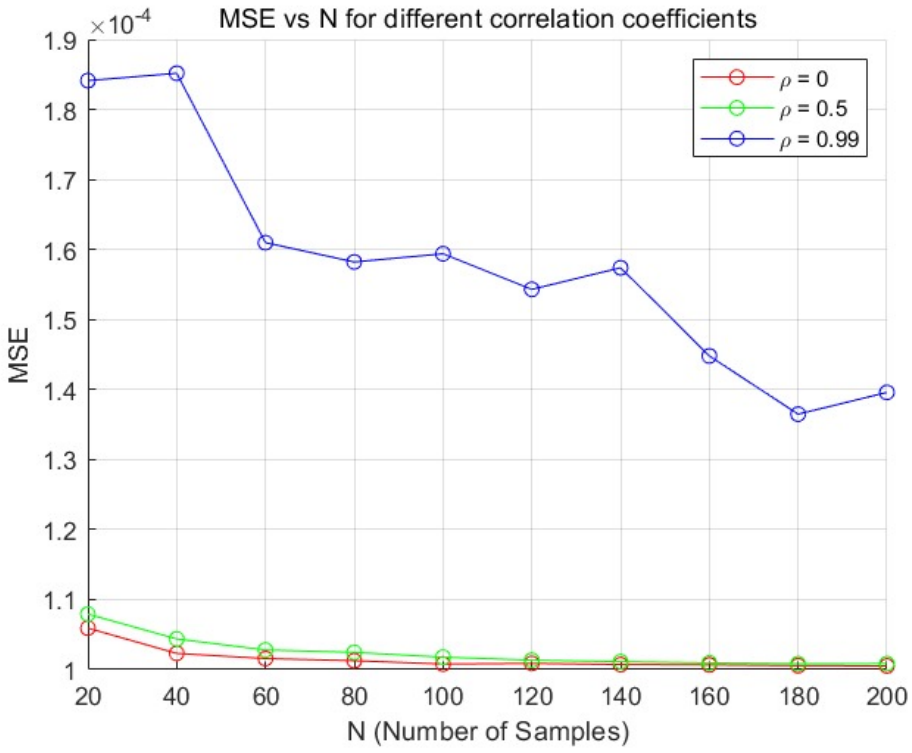


Figure 1: MSE vs N for different correlation coefficients

1.2.2 Frequency Estimation (Task 1.2.1, $h[n] = \delta[n]$)

Code:

```
Nrun = 200;
omega0 = [pi/4, pi/2, pi/sqrt(2), 0.81*pi];
N = [100, 500, 1024];
Oversampling = 1000;
A = 1;
snr = linspace(-20, 45, 28);

for i = 1:length(omega0)
    for j = 1:length(N)
        n_j = N(j);
        M = Oversampling * n_j;
        t = (1:n_j)';
        for k = 1:length(snr)
            for run = 1:Nrun
                w = sqrt((A^2/2)*10^(-snr(k)/10)) * randn(n_j,1);
                y = A * cos(omega0(i) * t + 2*pi*rand) + w;
                S = (abs(fft(y, M)).^2) / n_j; % Periodogram
                [~, idx_max] = max(S(2:M/2)); % FFT peak
                f_cent = idx_max + 1;
                Num = S(f_cent-1) - S(f_cent+1);
                Den = S(f_cent-1) + S(f_cent+1) - 2*S(f_cent);
                f_est2(run) = f_cent + 0.5 * Num / Den - 1; % Parabolic interpolation
            end
            MSE2(i,j,k) = mean((f_est2/M - omega0(i)/(2*pi)).^2); % MSE
        end
    end
end
```

Explanation: The first estimator f_cent is the index of the maximum value in the FFT periodogram (S), computed over $M = \text{Oversampling} \cdot N$ frequency bins, providing a coarse estimate of the signal's frequency. The second estimator f_est2 uses parabolic interpolation to refine this estimate. We fit a parabola to three periodogram points: $S(f_{cent} - 1)$, $S(f_{cent})$, and $S(f_{cent} + 1)$, and compute the maximum to estimate the true frequency peak with higher precision. The first and last frequency bins are ignored to ensure a valid three-point neighborhood for the parabolic fit, we account for this by incrementing the index of the maximum value f_cent.

The SNR is computed as follows for each N:
`CRB_N = 12/(N(j)*(N(j)^2-1))*(10.^(-snr/10));`
Then to display it on the graph:
`semilogy(snr, CRB_N/(4*pi^2), '--k', 'DisplayName', sprintf('CRB N=%d', N(j)));`

Results: MSE vs. SNR plots (Fig. 2) show that parabolic interpolation outperforms FFT peak detection, approaching the CRB on a wide range of the SNR, for all N. Note that the improvement is not that big since we already oversampled the FFT.

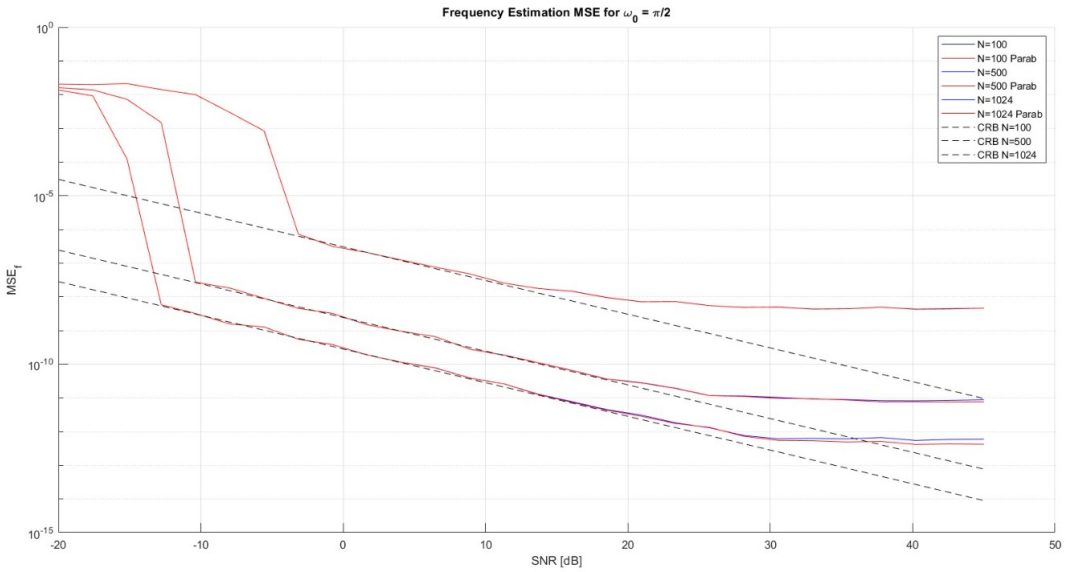


Figure 2: Frequency estimation MSE plot for $\omega_0 = \pi/2$

1.2.3 Correlated noise and filtering (Tasks 1.2.2 and 1.2.3

Code for filtered signal:

```
b=2;
a = [1 0.9];

% Compute filter magnitude squared for each omega0
H_mag2 = 4 ./ (1.81 + 1.8 * cos(omega0));

...

% Filter the signal before adding the noise
w = sqrt((A^2/2)*10^(-snr(k)/10)) * randn(n_j,1);
z = A * cos(omega0(i) * t + 2*pi*rand) + w;
y = filter(b, a, z) + w;

...

% Adapt the CRB
for i = 1:length(omega0)
    for j = 1:length(N)
        CRB_N = 12/(N(j)*(N(j)^2-1))*(10.^(-snr/10)) / H_mag2(i));
```

Explanation: To keep the CRB relevant, we divide it by the gain of the filter.

Code for correlated noise:

```
for i = 1:length(omega0)
    for j = 1:length(N)
        for k = 1:length(snr)

            % Compute FIM and CRB
            d_omega = -A * t .* sin(omega0_i * t);
            d_phi = -A * sin(omega0_i * t);
            I11 = d_omega' * (C_ww \ d_omega);
            I12 = d_omega' * (C_ww \ d_phi);
            I22 = d_phi' * (C_ww \ d_phi);
            FIM = [I11, I12; I12, I22];
            invFIM = inv(FIM);
            CRBs(i,j,k) = invFIM(1,1) / (4*pi^2);
```

Explanation: For the correlated noise generation, it's the same as in the Noise Generation section above with $\rho = 9$. The compute of the CRB is not trivial anymore since $\mathbf{C}_{ww}$ is no longer equal to the Identity matrix. The CRB is equal to the inverse of the Fisher information matrix, whose coefficients are computed as follows:
$I_{11} = \mathbf{d}_\omega^T C_{ww}^{-1} \mathbf{d}_\omega$
$I_{12} = I_{21} = \mathbf{d}_\omega^T C_{ww}^{-1} \mathbf{d}_\phi$
$I_{22} = \mathbf{d}_\phi^T C_{ww}^{-1} \mathbf{d}_\phi$

We compute all these with the same values of ω, $\mathbf{C}_{ww}$, and amplitude as each resulting signal. However, we pick $\phi_0 = 0$ as it does not change the mean value and simplifies the compute for us.

Results: We can see that despite all the stuff we added, both filtered signal and correlated noise for Task 1.2.3b (Fig. 3), Our estimators still perform well.

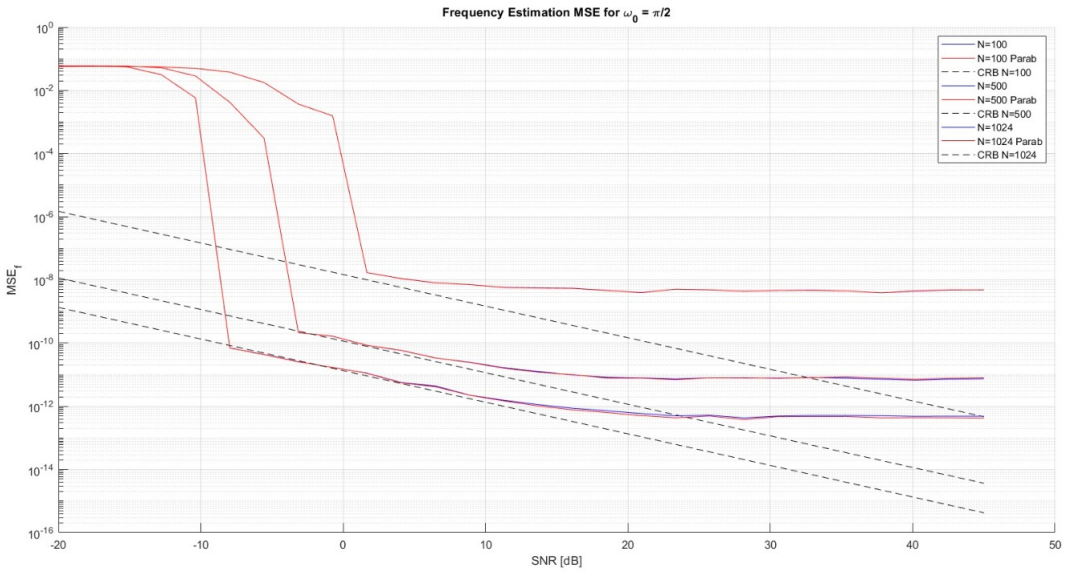


Figure 3: Frequency estimation MSE plot with correlated noise and filtered signal

1.2.4 Frequency Modulation

Code:

```
% Creating the sweep
fs = 22000;
N = 22000;
f0 = 1375;
fN = 8250;
sigma = 1;
omega0 = 2*pi*f0/fs;
omegaN = 2*pi*fN/fs;
gamma = (omegaN - omega0)/(2*N);
n = (0:N-1)';
phase = omega0*n + gamma*n.^2;
ref_signal = cos(phase) + sigma*randn(N,1);

% MLE
gamma_range = linspace(gamma*0.8, gamma*1.2, 50);
omega0_range = linspace(omega0*0.8, omega0*1.2, 50);

for i = 1:length(gamma_range)
    for j = 1:length(omega0_range)
        phase = omega0_range(j)*n + gamma_range(i)*n.^2;
        S = [cos(phase), sin(phase)];
        c_hat = pinv(S) * ref_signal;
        residual = ref_signal - S * c_hat; J_values(i,j) = sum(residual.^2);
    end
end
[~, idx] = min(J_values(:)); % Get the minimum
[i_opt, j_opt] = ind2sub(size(J_values), idx); % Get its index
gamma_hat = gamma_range(i_opt);
omega0_hat = omega0_range(j_opt);
```

Explanation: In the first part of the code, we create the frequency sweep using the γ and ω factors that we will have to estimate, so we can compare them with the true values.
To estimate γ and ω, we performed a grid search, we narrowed down the range arround the true values to save time, but we could have searched on a wider range, the results found were not pushed by the $\pm 20\%$ margin bounds.

Results: The spectrogram (Fig. 4) visualizes the frequency sweep, while MLE accurately estimates γ (Fig. 5), with the reconstructed signal closely matching the noisy input (Fig. 6). We can see that our γ estimation is very close.

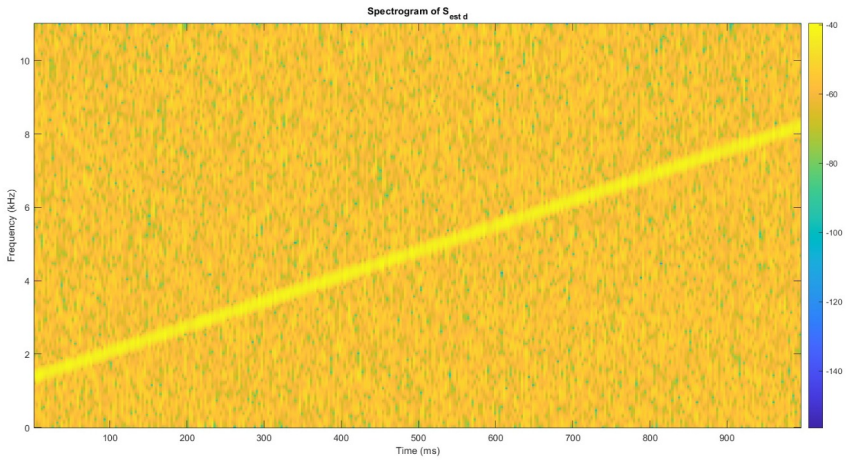


Figure 4: measured frequency sweep

```
True gamma: 4.462490e-05 rad/sample^2
Estimated gamma: 4.480704e-05 rad/sample^2
Estimated omega0: 0.387891 rad/sample
Estimated amplitude: 0.0093
Estimated phase: -0.2680 rad
```

Figure 5: true vs estimated gamma (and other parameters of the model)

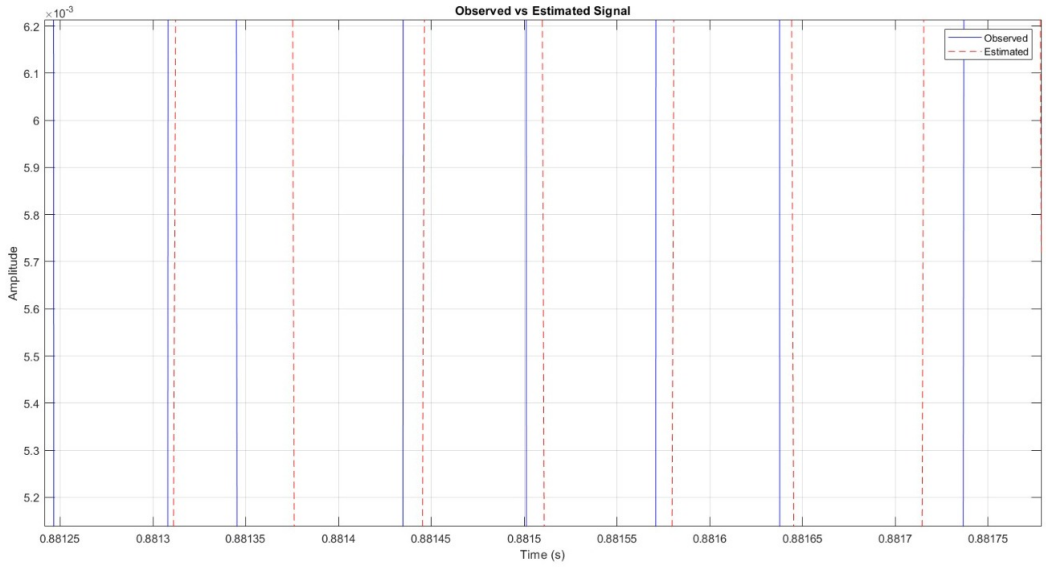


Figure 6: real and estimated superimposed

1.3. Conclusion

We learned that:

- Larger sample sizes and lower noise correlation improve the accuracy of covariance estimation.
- Parabolic interpolation enhances frequency estimation, closely approaching the CRB at high SNR.
- Spectrograms and MLE effectively analyze and estimate the parameters of frequency-modulated signals.

2. Homework 2: Adaptive Interference Cancellation

2.1. Sect. 1: Theoretical Tools and Methods Adopted

Homework 2 aims to extract clean audio signals from noisy stereo recordings using adaptive interference cancellation across five scenarios (a, b, c, d, e). The methods include:

- **Adaptive Filtering with LMS:** The Least Mean Squares (LMS) algorithm adjusts filter coefficients to minimize the error $\epsilon[n] = s_{\text{in}}[n] - \hat{w}[n]$, where $\hat{w}[n]$ is the estimated interference from the reference signal.
- **Pre-whitening:** Using Linear Predictive Coding (LPC), we decorrelate the reference signal, improving LMS convergence by flattening its spectrum.
- **Subband Decomposition:** For complex noise (b, c), we split the signal into frequency bands using Butterworth filters, applying LMS to each band for targeted noise suppression.
- **Spectrogram Analysis:** We use spectrograms to identify noise patterns, guiding reference signal construction (d) or subtraction (e).
- **Band-stop Filtering:** We played with Butterworth band-stop filter to try to remove specific harmonic interference.

2.2. Sect. 2: Solution, Kernel of MATLAB Code, and Results

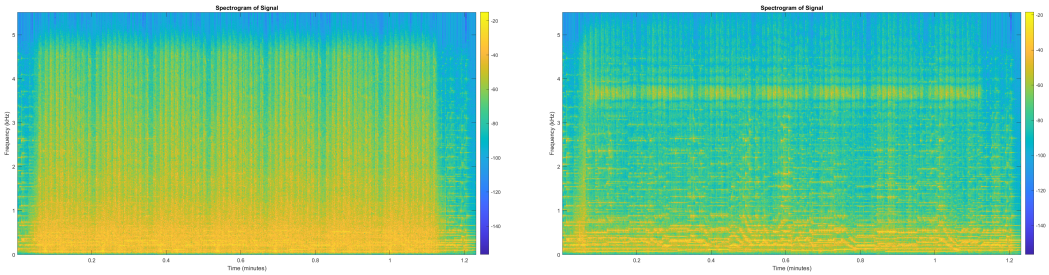
2.2.1 Scenario a

Code:

```
function S_est = WLMS(x, ref, mu, filter_length)
    [a_white, ~] = lpc(ref, 5); % Pre-whitening with LPC
    ref_white = filter(a_white*0.99, 1, ref); % Whiten reference
    ref_white = ref_white / max(abs(ref_white));
    x_normalized = x / max(abs(x));
    N = length(x);
    w = zeros(filter_length, 1);
    ref_buffer = zeros(filter_length, 1);

    for n = 1:N
        ref_buffer = [ref_white(n);
        ref_buffer(1:end-1)];
        S_est(n) = x_normalized(n) - w' * ref_buffer; % LMS
        w = w + mu * S_est(n) * ref_buffer; % Update coefficients
    end
    S_est = S_est * max(abs(x)); % Restore amplitude
end

S_est_a(:,1) = WLMS(Sin_a(:,1), Sn_ref_a(:,1), 0.06, 60);
S_est_a(:,1) = hardcut_filter(S_est_a(:,1), fs, 3300, 5200); % Band-stop filter
```

Explanation: We prewhiten the signal, then compute LMS. Since some of the noise was still present in the higher frequencies, we tried applying a bandstop filter. This succesfuly removed the noise, but also altered the signal, so we chose not to keep it.

Results: The spectrogram (Fig. 7b) shows a significant noise reduction compared to the original (Fig. 7a), with periodic interference mostly suppressed while preserving the integrity of the signal.



(a) The spectrogram of the original S_ref_a. (b) The spectrogram of S_est_a after whitening and LMS.

Figure 7: Whiten Sn_ref_a before applying LMS.

2.2.2 Scenario b

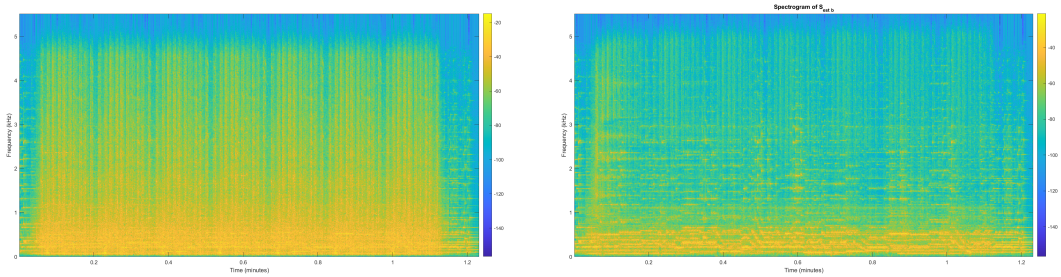
Code:

```
function S_est = SWLMS(x, ref, step_size, filter_length, fs)
    bands = [0,1000; 1000,2000; 2000,3000; 3000,4000; 4000,fs/2];
    S_est = zeros(length(x), 1);
    for i = 1:size(bands,1)
        if i==1,
            [b,a] = butter(6, bands(i,2)/(fs/2), 'low');
        elseif i==size(bands,1),
            [b,a] = butter(6, bands(i,1)/(fs/2), 'high');
        else,
            [b,a] = butter(6, [bands(i,1), bands(i,2)]/(fs/2), 'bandpass');
        end

        x_band = filtfilt(b, a, x);
        ref_band = filtfilt(b, a, ref);
        S_est = S_est + WLMS(x_band, ref_band, step_size, filter_length); % Subband LMS
    end
end

S_est_b(:,1) = SWLMS(Sin_b(:,1), Sn_ref_b(:,2), 0.03, 500, fs);
```

Explanation: This time, prewhitening wasn't enough, so we split the signal into frequency bands, filtered them independantly, each with custom LMS parameters, then recombined them.

Note that we did exactly the same for Scenario c.

Results: The spectrogram (Fig. 8b) shows effective noise suppression across bands compared to the original (Fig. 8a).



(a) Spectrogram of S_{ref_b} (b) Best Parameter Combination

Figure 8: The results of applying various noise suppression techniques.

2.2.3 Scenario d

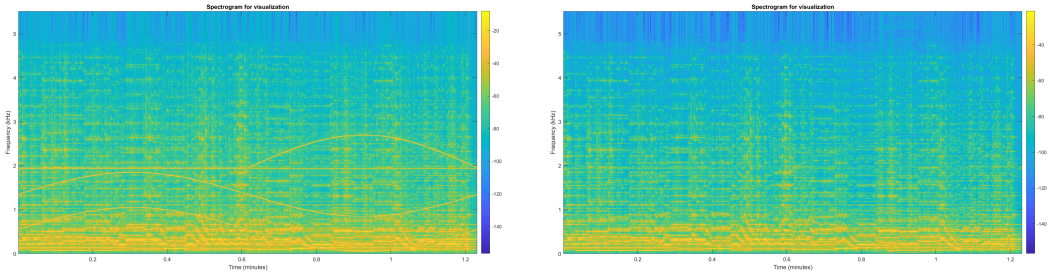
Code:

```
sin1 = create_stable_sin(1940, 816000, 11025, 0.02);
sin2 = create_stable_sin(1950, 408000, 11025, 0.02);
sin3 = create_variable_sin(1350, 500, 816000, 11025, 0.02);
sin4 = create_variable_sin(550, 500, 816000, 11025, 0.02);
sin5 = create_variable_sin(1950, 750, 816000, 11025, 0.02);
sin5 = sin5(1:408000);
Sn_ref_d = sin1+[sin2; sin5]+sin3+sin4; % Combine sinusoids

S_est_d(:,1) = WLMS(Sin_d(:,1), Sn_ref_d, 0.036, 42);
```

Explanation: In this code we recreate the frequency patterns that we observe in the original signal. We got the values by carefully inspecting the spectrogram of S_{in_d}. Then we just compute a prewhitened LMS with our reconstructed interference as the reference and voilà.

Note that we did exactly the same for Scenario e.

Results: The spectrogram (Fig. 9b) shows an almost complete removal of noise compared to the original signal (Fig. 9a).



(a) The spectrogram of the original $S_{\texttt{in\_d}}$.

(b) The spectrogram of $S_{\texttt{est\_d}}$

Figure 9: The spectrogram of $S_{\texttt{est\_d}}$

.

2.2.4 How did we get the right parameters

Code:

```
filter_lengths = [16, 32, 64, 128, 256]; % Pick any grid
forgetting_factors = 0.95:0.001:1;

for i = 1:length(filter_lengths)
    for j = 1:length(forgetting_factors)
        % Perform LMS or WLMS or any algo
        ...

        % Compute MSE
        mse(i,j) = mean((Sin_c_left - S_est).^2);
    end
end

% Find best parameters using MSE metric
[min_mse, idx] = min(mse(:));
```

Explanation: We simply perform iterative grid search for μ and N and pick the smallest MSE value. This optimizes for the reduction of noise while not losing valuable signal information.

Results: Here is an example of grid search for Scenario c. Note that we performed many of these for all Scenari.

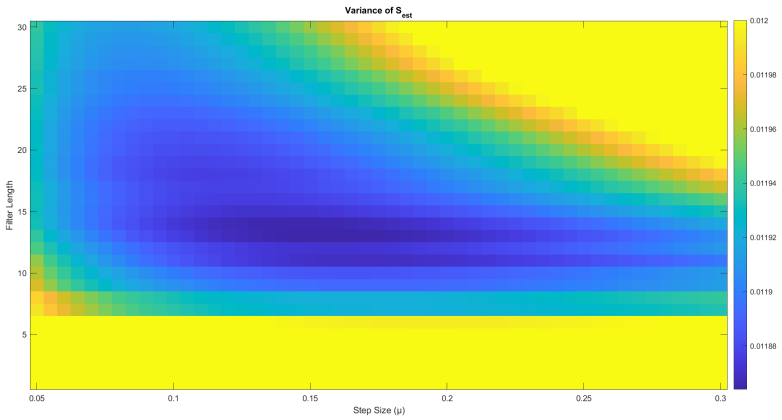


Figure 10: Optimal values of mu and N for c

2.3. Sect. 3: Conclusion

We learned that:

- Pre-whitening and subband decomposition enhance LMS performance for correlated and complex noise.
- Spectrogram-guided reference construction is crucial for time-varying interference.
- Parameter tuning balances noise suppression and signal fidelity.
- Direct bandstop is not always a wise option.

These techniques apply to speech enhancement and biomedical signal processing. Improvements could involve RLS for faster convergence or machine learning for noise pattern recognition.