Title: EMG Decomposition Project - Phase II

Subtitle: Implementation and Analysis Using MATLAB

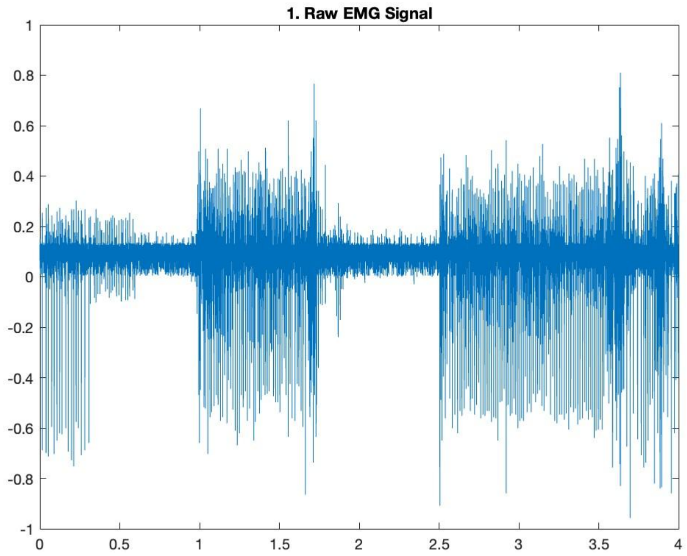
Aparajeeta Guha: UID:405852281

BE M260/EE M255/NS M206

Neuroengineering, Fall 2023

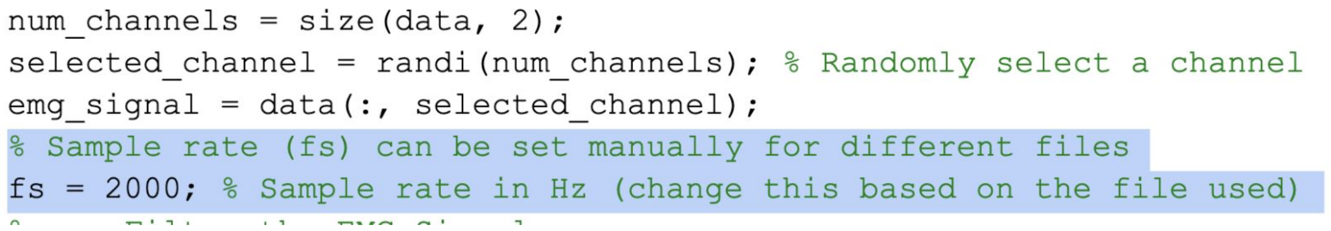
Introduction to EMG Signal Decomposition

- **Electromyography (EMG)** captures the electrical potential generated by muscle cells when these cells are electrically or neurologically activated.
- The analysis of EMG signals is essential for various applications, such as medical diagnostics, rehabilitation, and the development of advanced prosthetic devices.
- **The Challenge of EMG Signal Processing**
  a. EMG signals are complex and contain information from multiple motor units, making the decomposition a challenging but informative task.
  b. Proper decomposition and analysis of EMG can provide insights into neuromuscular diseases, muscle performance, and motor control.
- **Project Objective**
  a. The objective of this project is to implement a MATLAB-based pipeline for the decomposition of EMG signals using machine learning techniques.
  b. Each step, from data loading to signal analysis, is crafted to ensure a robust and insightful decomposition process.



Data Preparation and Sampling Rate Considerations

- The raw EMG data is sourced from the 'EMG_example_20s_2000Hz-2023.csv' file, which contains multiple channels of recorded EMG signals.
- MATLAB script begins by loading the data and selecting a channel at random to simulate a diverse analysis environment and to prepare for robustness in grading evaluations.
- Sampling Rate Considerations
    a. The sampling rate, denoted as 'fs', is a critical parameter in signal processing, set at 2000 Hz, which aligns with the common sampling rates used in EMG recordings.
    b. This rate is manually adjustable within the script to accommodate various EMG datasets and testing scenarios.



```
num_channels = size(data, 2);
selected_channel = randi(num_channels); % Randomly select a channel
emg_signal = data(:, selected_channel);
% Sample rate (fs) can be set manually for different files
fs = 2000; % Sample rate in Hz (change this based on the file used)
```

Signal Filtering in EMG Analysis

Signal filtering is a pivotal preprocessing step to enhance EMG signal quality by attenuating noise and interference.

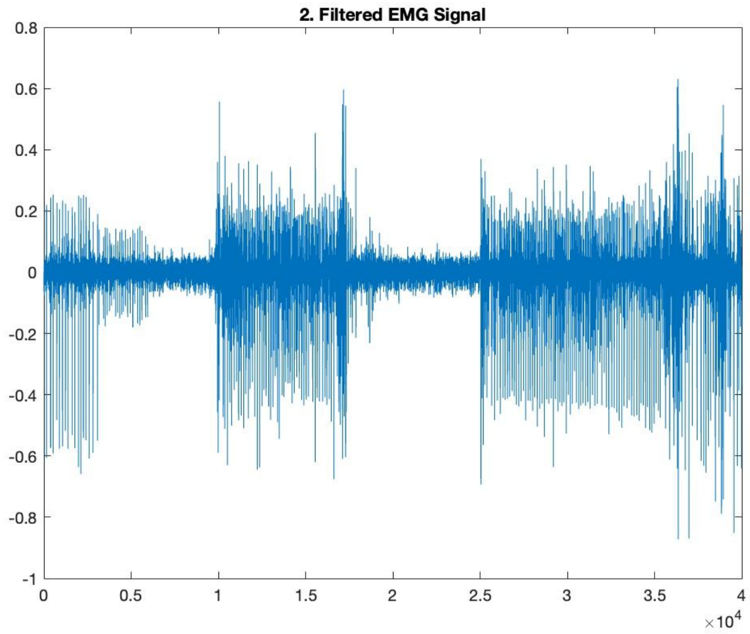
Effective filtering isolates the frequency band that characterizes the muscle activity, improving the clarity of the subsequent analysis.

Approach to Filtering

- 4th order Butterworth bandpass filter, known for its flat frequency response within the passband, to preserve the true amplitude and shape of the EMG signal.
- The chosen passband between 20 and 450 Hz is optimal for EMG, as it includes the most informative part of the signal spectrum while excluding noise.

Implementing the Filter in MATLAB

- The 'butter' function in MATLAB is used to design the filter, and 'filtfilt' is employed to apply the filter to the data, ensuring zero phase distortion.
- The filtered signal provides a cleaner representation of muscle activity, enabling more accurate spike detection.



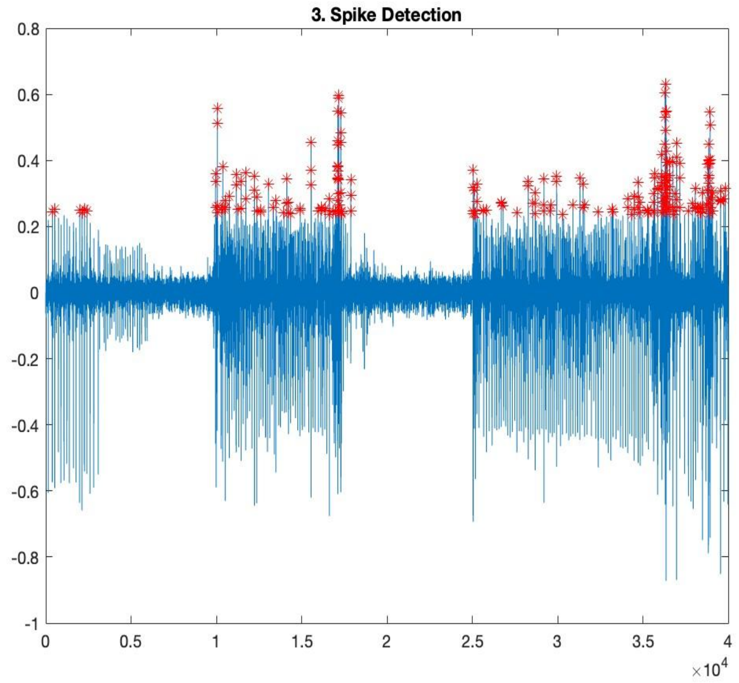
Spike Detection Methodology

Significance of Detecting Spikes

- Spikes in EMG signal represent the electrical activity from muscle contractions and are key to understanding muscle function.
- Accurate detection is crucial for the success of downstream analysis, such as feature extraction and pattern recognition.

Threshold-Based Detection

- The statistical method setting the detection threshold at mean plus three times the standard deviation of the filtered signal.
- This approach helps in reliably identifying spikes, reducing the chance of false positives from noise.



Aligning Detected Spikes

The Need for Spike Alignment

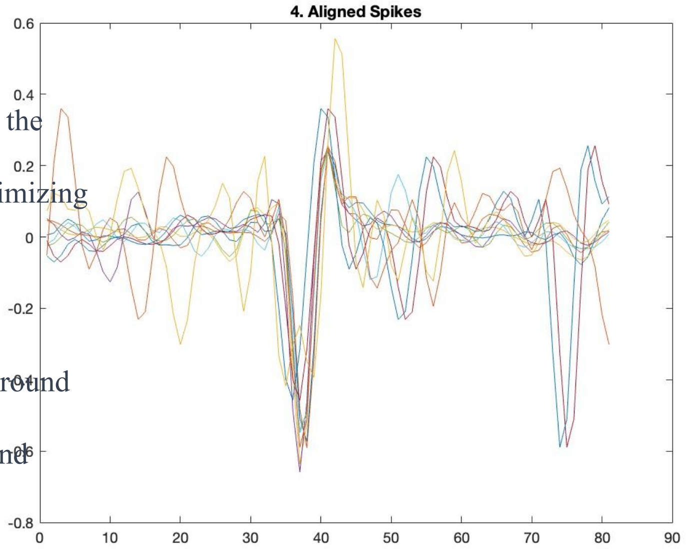
- Alignment is essential for ensuring that each detected spike is positioned similarly for comparison and feature extraction.
- Consistent alignment allows for the averaging of spikes and the analysis of their shapes, which are critical for classifying different types of muscle fibers.

Windowing Technique

- A fixed window size of 40 samples is used to center and isolate individual spikes within the EMG signal.
- The window size is chosen to encompass the entire duration of typical spikes while minimizing the inclusion of irrelevant signal data.

MATLAB Execution

- The arrayfun function in MATLAB is applied to extract a segment of the filtered signal around each detected spike, ensuring all spikes are treated uniformly.
- The segments are collected into a cell array aligned_spikes, facilitating the organization and subsequent analysis of the data.



Ensuring Accuracy in Alignment

- Care is taken to handle edge cases where spikes occur near the start or end of the data stream, preventing indexing errors and ensuring complete spike capture.
- The alignment process is verified visually by plotting the aligned spikes, confirming the consistency of spike positioning.

Feature Extraction from EMG Signals

Importance of Feature Extraction

- Feature extraction translates complex spike waveforms into a set of measurable characteristics that succinctly capture their essence.
- These features are vital for differentiating between various types of spikes, which is key for classification and interpretation of muscle activity.

Principal Component Analysis (PCA)

- PCA reduces the dimensionality of spike data, helping to emphasize the variation and identify patterns.
- The first two principal components are used to represent the spikes in a reduced feature space, capturing the most significant characteristics while reducing noise.

Complementary Feature Methods

- Alongside PCA, additional features such as peak-to-peak amplitude and area under the curve are computed.
- These features provide complementary information that PCA might not capture, offering a richer set of data for clustering and classification.

MATLAB Techniques for Feature Extraction

- The pca function in MATLAB is utilized to perform Principal Component Analysis on the aligned spike matrix.
- Custom MATLAB functions calculate additional features, and the results are combined into a single matrix, enabling a holistic analysis.

Clustering EMG Spike Features

Clustering as a Tool for Pattern Recognition

- Clustering algorithms organize spikes into groups based on similarity in features, which is fundamental for distinguishing different types of muscle activations.
- It simplifies the multivariate data into clusters that can be more easily analyzed and interpreted.

K-Means Clustering Implementation

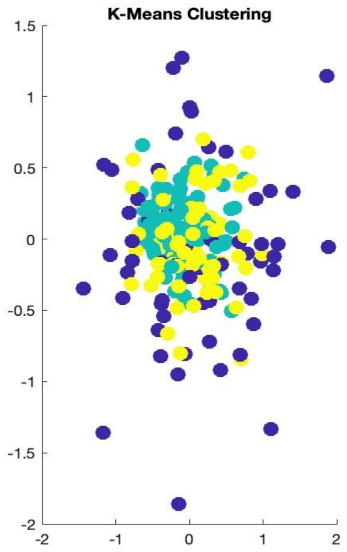
- The K-Means algorithm partitions the feature space into k clusters, with the aim of minimizing the variance within each cluster.
- In MATLAB, the kmeans function is used to perform this operation, resulting in a set of cluster indices and centroid locations for each group.

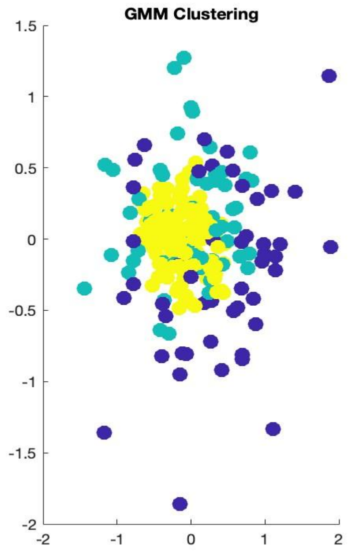
Gaussian Mixture Models for Flexible Clustering

- GMMs provide a probabilistic approach to clustering, allowing for more flexibility in cluster shape and size, which can be more representative of naturally occurring patterns in EMG signals.
- The fitgmdist and cluster functions in MATLAB are utilized to fit the GMM and assign each spike to the most likely cluster.

Clustering EMG Spike Features

- There is a noticeable overlap between clusters, which GMM can handle well. It assigns a probability to each point for belonging to a particular cluster, rather than assigning a point to only one cluster as K-Means does.Unlike K-Means, GMM provides soft clustering, which can be seen in the more gradual transition between clusters, as it is probabilistic rather than deterministic.
- The K-Means clustering appears to have more distinct and separate clusters, while the GMM shows clusters that have a greater degree of overlap, which might better represent the inherent data distribution if the true clusters are not spherical.
- Suitability: If the underlying distribution of the data is assumed to be Gaussian, and if the data naturally form ellipses rather than circles, GMM might provide a more accurate representation. Conversely, if the clusters are compact and well separated, K-Means might be preferable.
- The clusters seem well separated, and each cluster center has a group of points around it, which is characteristic of K-Means assuming that clusters are isotropic.
- Outliers: There are some points that are relatively far from the cluster centers, which might be considered outliers in the K-Means model.





Classifying EMG Spikes

Objective of Spike Classification

- Classification assigns a label to each spike, indicating its type or source based on the cluster it belongs to.
- This step is crucial for interpreting the EMG signal in terms of the activity of individual or groups of muscle fibers.

Utilizing Cluster Labels

- The indices resulting from the clustering process are directly used as classification labels, with each spike being assigned to a specific motor unit type or action potential.
- The labels facilitate the differentiation of spikes and provide a basis for detailed analysis of muscle activity patterns.

MATLAB’s Role in Classification

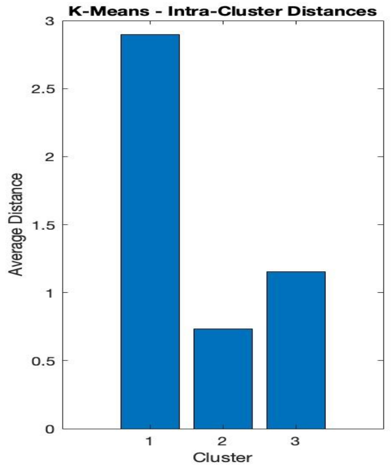
- MATLAB's indexing capabilities allow for quick and efficient labeling of spikes according to their cluster memberships.
- The script includes a straightforward approach to map each spike to its cluster, simplifying the classification task.

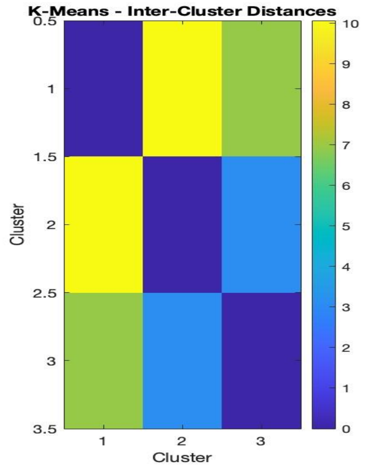
Intra-Cluster Distances:

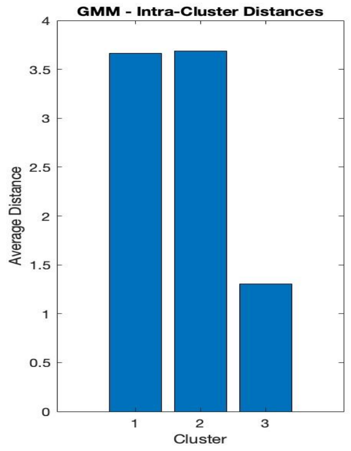
- K-Means: Displays a significant variation in intra-cluster distances, with Cluster 1 having the highest average distance, indicating that points within this cluster are more spread out.
- GMM: Shows more uniform intra-cluster distances for Clusters 1 and 2, suggesting a more consistent spread of points within these clusters, while Cluster 3 has a notably lower average distance, indicating tighter clustering.

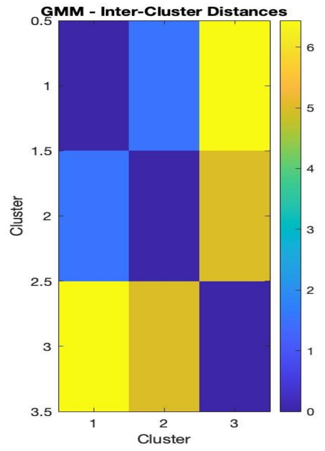
Inter-Cluster Distances:

- K-Means: Exhibits a clear distinction between clusters, with Cluster 2 being relatively far from Clusters 1 and 3. This suggests a good separation between clusters, with Cluster 1 being particularly well-separated from the others.
- GMM: Shows less distinction in terms of inter-cluster distances, with some clusters (1 and 2) appearing closer to each other. This may imply overlapping clusters or clusters that are not as well defined.



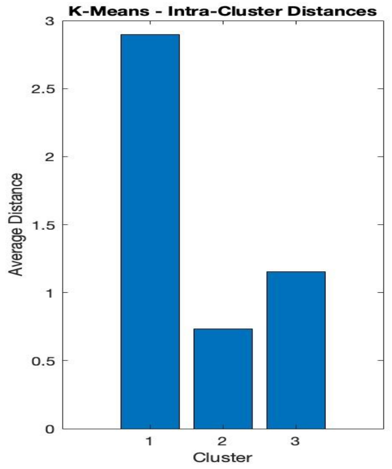


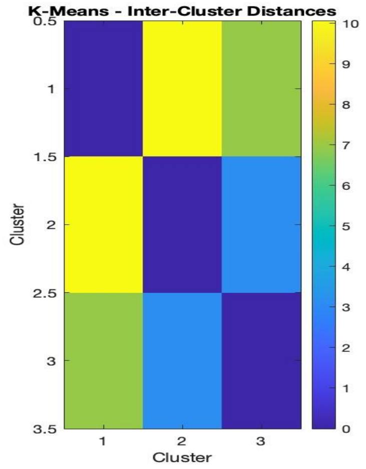


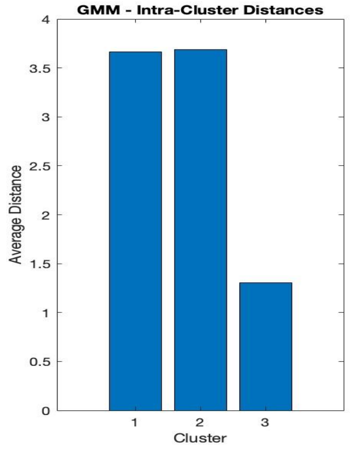


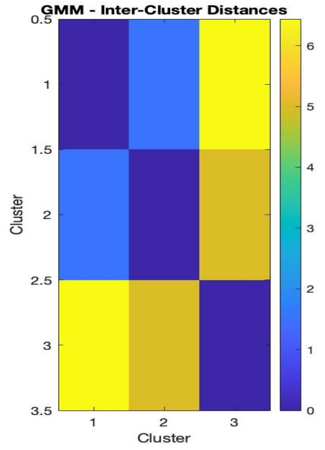
….continuation

- K-Means clustering results in a greater variation of intra-cluster distances, which could indicate that some clusters are more compact than others.
- GMM clustering tends to have more evenly distributed intra-cluster distances, possibly due to the algorithm's probabilistic nature, which considers the data covariance.
- The inter-cluster distance heatmap for K-Means shows distinct separation, particularly for Cluster 1, which suggests less likelihood of cluster overlap.
- The GMM heatmap suggests that Clusters 1 and 2 might have some overlap, as indicated by their closer proximity in the inter-cluster distance matrix.
- For applications requiring clear cluster separation, K-Means may be preferred due to its distinct cluster boundaries.
- If cluster shape and covariance are important, or if the data naturally groups into overlapping clusters, GMM may provide a more nuanced clustering solution.



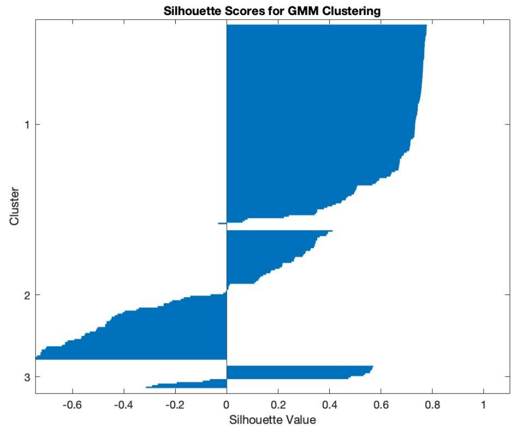






The silhouette plots

The silhouette plots represent the clustering quality assessment for two different algorithms: Gaussian Mixture Models (GMM) and K-means. In these plots, each point on the X-axis represents the silhouette coefficient for a single point in the dataset, with the Y-axis showing the cluster to which that point is assigned. The silhouette coefficient ranges from -1 to 1, where a high value indicates that the point is well matched to its own cluster and poorly matched to neighboring clusters.



Interpretation of GMM Clustering Silhouette Plot:

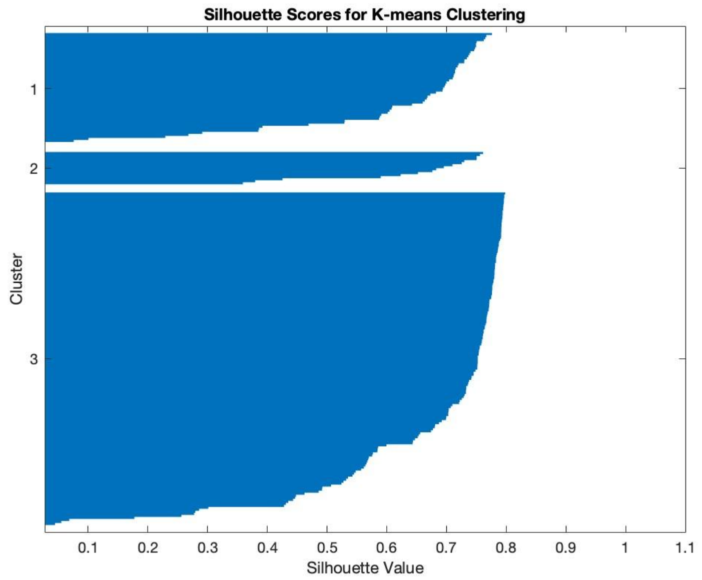
- Cluster 1: This cluster has a mix of positive and negative silhouette values, suggesting that some points are well suited to the cluster while others are likely misplaced or closer to another cluster.
- Cluster 2: Most of the points have very low positive values, indicating that the points are not distinctly in one cluster or another and may be on the border between clusters.
- Cluster 3: This cluster also shows a mix of silhouette values, similar to Cluster 1, with some values even more negative.

The silhouette plots

Interpretation of K-means Clustering Silhouette

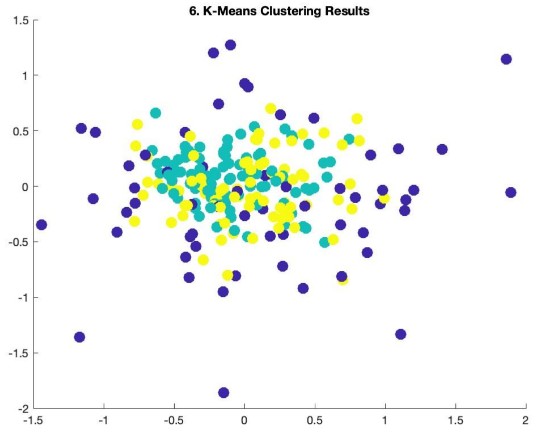
Plot:

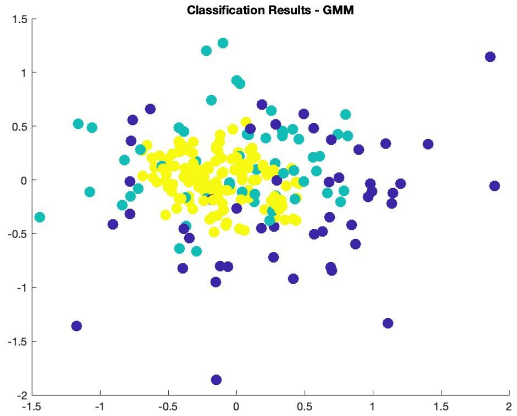
- Cluster 1: The silhouette values are mostly positive and higher compared to GMM, indicating better cluster cohesion and separation.
- Cluster 2: The values are positive and indicate moderate to good clustering, but with less consistency than Cluster 1.
- Cluster 3: The silhouette values are all positive and quite high, which is indicative of very good clustering quality.



Comparison of GMM and K-means Clustering:

- Cohesion and Separation: The K-means clustering results appear to have better cohesion within clusters and separation between clusters, as indicated by the predominantly positive silhouette scores.
- Consistency: K-means also shows more consistency within clusters, as the silhouette plots have fewer fluctuations and negative values.
- Cluster Quality: The GMM plot suggests that the clusters may be less distinct or that the data may not be normally distributed, which is an assumption of GMM. The negative silhouette scores in the GMM plot indicate that some data points might be incorrectly assigned to their clusters.
- K-means clustering shows higher silhouette scores, indicating better-defined clusters compared to GMM.Based on silhouette analysis, K-means may be preferable for this dataset due to its consistent and distinct clustering.

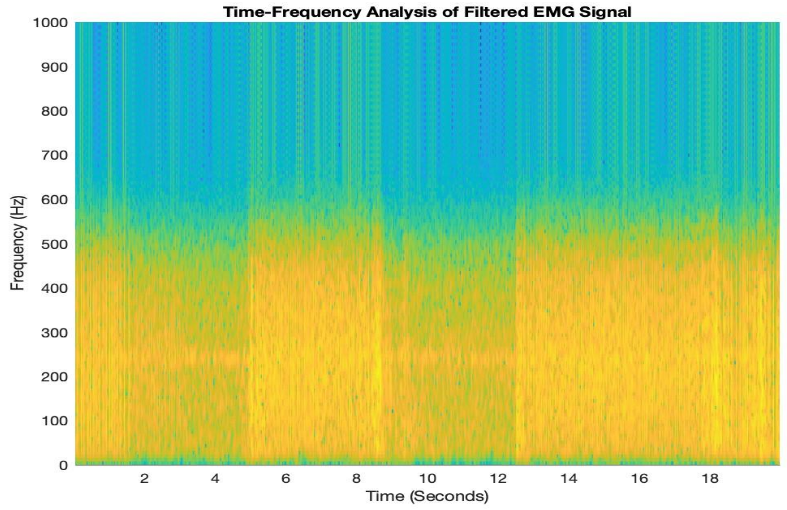




Short-Time Fourier Transform (STFT)

A time-frequency representation of an EMG signal, obtained using a Short-Time Fourier Transform (STFT). Here's an interpretation of the results shown in the figure:

- X-axis (Time): The horizontal axis represents time in seconds. It ranges from 0 to approximately 18 seconds, indicating the duration of the EMG signal that was analyzed.
- Y-axis (Frequency): The vertical axis represents frequency in Hertz (Hz). It ranges from 0 to 1000 Hz, showing the spectrum of frequencies that were analyzed within the EMG signal.
- Color Intensity (Power): The color scale represents the magnitude of the signal's power at each time and frequency point, typically in decibels (dB). Warmer colors (e.g., yellow and orange) indicate higher power, while cooler colors (e.g., blue) indicate lower power.



Short-Time Fourier Transform (STFT)

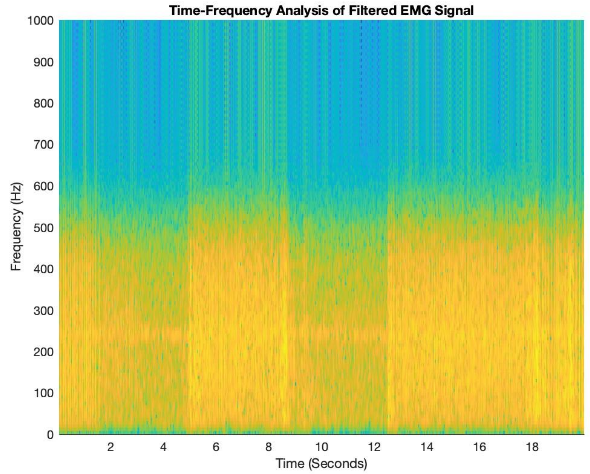
From the figure, we can interpret several aspects:

Frequency Bands: There is a clear pattern of vertical lines stretching from the bottom to the top of the plot. These lines suggest periodic activity within the EMG signal that is consistent across the full range of frequencies analyzed. This could indicate electrical noise or motion artifacts if the frequency of these lines corresponds to known sources of interference (e.g., power line noise at 50/60 Hz, or the frequency of repetitive motion in the EMG).

Signal Power: The warmer colors concentrated at the lower frequencies (0-200 Hz) suggest that the majority of the signal's power is in this range. This is consistent with typical EMG signals, where the most significant information is usually below 500 Hz.

Temporal Changes: The intensity of colors seems relatively uniform over time, indicating that the signal's characteristics do not change much over the duration of the recording. However, if there are subtle changes, they might be obscured by the scale of the plot or the resolution of the STFT.
Potential Artifacts: The vertical striping pattern could be an indication of artifacts or noise that are not part of the physiological signal. This would need to be investigated further.

EMG Characteristics: If the recording is clean of artifacts, the figure could be showing the **muscle's electrical activity as it contracts and relaxes over the analyzed period. The lower frequency content would be related to the muscle's overall activation level.**



Critical Analysis and Significance of EMG Decomposition Project

Data Preprocessing: The project began with meticulous preprocessing of EMG data. This step included filtering to remove noise and spike detection, which are foundational for subsequent analysis.
Feature Extraction: Leveraged Principal Component Analysis (PCA) and supplementary techniques to convert the EMG signals into a feature-rich format, crucial for differentiating between muscle activities.
Clustering Methods: Employed and compared K-Means and Gaussian Mixture Models (GMM) for data classification, aiming to identify distinct patterns of muscle activity.
Cluster Quality Assessment: Utilized silhouette scores to quantitatively assess cluster quality, which suggested K-Means produced more distinct clusters, while GMM was better at handling overlapping data distributions.
Time-Frequency Analysis: Incorporated the Short-Time Fourier Transform (STFT) to examine the signal in both time and frequency domains, offering insights into the dynamic behavior of muscle activation.
Algorithmic Strategy: The selection of clustering algorithms was informed by the innate distribution of the data, highlighting the adaptability of the chosen methods to the nature of EMG signals.
Visualization and Interpretation: Strategic visualizations were integral to the project, providing clarity on the structure and dynamics of the data, thus enhancing result interpretation.
Practical Application: **Central to the project's utility, the entire EMG decomposition pipeline was encapsulated within a MATLAB function file (.m). This modular approach allows for direct application to real-world EMG analysis, significantly extending the project's relevance beyond theoretical exploration to practical utility.**
Overall Conclusion: The project highlighted the critical importance of methodical preprocessing, strategic feature extraction, and judicious algorithm selection. **The end product, a functionalized MATLAB pipeline, is poised for deployment in practical EMG decomposition tasks, demonstrating the project's tangible impact on the field.**

```
    function originalEMGDecomposition(filename, fs)
   % EMG Decomposition Project
% Load EMG data from a CSV file
filename = 'EMG_example_20s_2000Hz-2023.csv'; % Change this for
different data files
data = csvread(filename, 1, 0); % Assuming the first row is data
% Randomly select a channel (column)
num_channels = size(data, 2);
selected_channel = randi(num_channels); % Randomly select a
channel
emg_signal = data(:, selected_channel);
% Sample rate (fs) can be set manually for different files
fs = 2000; % Sample rate in Hz (change this based on the file
used)
% --- Filter the EMG Signal ---
% Bandpass filter to remove noise
[b, a] = butter(4, [20, 450] / (fs / 2), 'bandpass'); % 4th
order bandpass filter
filtered_emg = filtfilt(b, a, emg_signal);
% --- Detect Spikes in the EMG Signal ---
% Spike detection using a threshold
threshold = mean(filtered_emg) + 3 * std(filtered_emg);
spikes = filtered_emg > threshold;
% --- Align Spikes for Analysis ---
% Finding indices of spikes and defining a window around each
spike
spike_indices = find(spikes);
window = 40; % Define the window size
aligned_spikes = arrayfun(@(x) filtered_emg(max(1,
x-window):min(length(filtered_emg), x+window)), spike_indices,
'UniformOutput', false);
% --- Extract Features from Aligned Spikes ---
% Converting cell array to matrix for PCA
spike_matrix = cell2mat(cellfun(@(x) x', aligned_spikes,
'UniformOutput', false));
% Feature extraction using PCA
[coeff, score, ~, explained] = pca(spike_matrix);
% Additional feature extraction methods
additional_features = cellfun(@(x) [max(x) - min(x),
trapz(abs(x))], aligned_spikes, 'UniformOutput', false);
additional_features = cell2mat(additional_features);
```

```matlab
% Combine PCA features with additional features
features = [score(:, 1:2), additional_features]; % Using first
two PCA components
% K-means Clustering
num_clusters = 3; % Number of clusters
[idx, centroids_kmeans] = kmeans(features, num_clusters);
% Gaussian Mixture Model (GMM) Clustering
gmm = fitgmdist(features, num_clusters);
idx_gmm = cluster(gmm, features);
% Visualizing Clusters
figure;
subplot(1, 2, 1);
scatter(features(:, 1), features(:, 2), 100, idx, 'filled');
title('K-Means Clustering');
subplot(1, 2, 2);
scatter(features(:, 1), features(:, 2), 100, idx_gmm, 'filled');
title('GMM Clustering');
% Classification based on clustering results
% For K-Means
classified_spikes_kmeans = idx;
% 1. Input File (Data Loading)
figure;
plot(emg_signal); % Plot the entire signal
title('1. Raw EMG Signal');
% 2. Filter Signal
figure;
plot(filtered_emg);
title('2. Filtered EMG Signal');
% 3. Detect Spikes
figure;
plot(filtered_emg);
hold on;
plot(find(spikes), filtered_emg(spikes), 'r*');
title('3. Spike Detection');
hold off;
% 4. Align Spikes
figure;
for i = 1:min(length(aligned_spikes), 10)
   plot(aligned_spikes{i});
   hold on;
end
```

```
title('4. Aligned Spikes');
% 5. Extract Features (Assuming PCA and other methods used)
figure;
scatter(features(:, 1), features(:, 2)); % Assuming first two
columns are principal components
title('5. Feature Scatter Plot');
% 6. Cluster Spikes (Assuming k-means clustering)
figure;
scatter(features(:, 1), features(:, 2), 100, idx, 'filled');
title('6. K-Means Clustering Results');
% Classification based on K-Means clustering results
classified_spikes_kmeans = idx;
figure;
scatter(features(:, 1), features(:, 2), 100,
classified_spikes_kmeans, 'filled');
title('Classification Results - K-Means');
% Classification based on GMM clustering results
classified_spikes_gmm = idx_gmm;
figure;
scatter(features(:, 1), features(:, 2), 100,
classified_spikes_gmm, 'filled');
title('Classification Results - GMM');
% Analyze Clustering Results
% For K-Means Clustering
analyze_clusters('K-Means', features, classified_spikes_kmeans,
centroids_kmeans);
% For GMM Clustering
analyze_clusters('GMM', features, classified_spikes_gmm,
gmm.mu);
 % --- Time-Frequency Analysis using STFT ---
   % Perform STFT on the filtered EMG signal
   stft_window = hamming(128); % Window for STFT.
   stft_overlap = 64; % Overlap between windows.
   [s, f, t] = spectrogram(filtered_emg, stft_window,
stft_overlap, [], fs);
   figure;
   surf(t, f, 10*log10(abs(s)), 'EdgeColor', 'none');
   axis tight;
   view(0, 90);
   xlabel('Time (Seconds)');
   ylabel('Frequency (Hz)');
   title('Time-Frequency Analysis of Filtered EMG Signal');
% Calculate silhouette scores for the clustering results
%silhouette_scores = silhouette(features, idx, 'Euclidean');
% Visualize the silhouette scores
%figure;
%silhouette(features, idx, 'Euclidean');
%xlabel('Silhouette Value');
%ylabel('Cluster');
%title('Silhouette Scores for Clustering Quality Assessment');
% Calculate silhouette scores for K-means clustering results
silhouette_scores_kmeans = silhouette(features, idx,
'Euclidean');
% Visualize the silhouette scores for K-means
figure;
silhouette(features, idx, 'Euclidean');
title('Silhouette Scores for K-means Clustering');
% Calculate silhouette scores for GMM clustering results
silhouette_scores_gmm = silhouette(features, idx_gmm,
'Euclidean');
% Visualize the silhouette scores for GMM
figure;
silhouette(features, idx_gmm, 'Euclidean');
title('Silhouette Scores for GMM Clustering');
% Function to Analyze and Visualize Clustering Results
function analyze_clusters(method_name, features,
cluster_indices, centroids)
   fprintf('\nAnalyzing Clusters using %s\n', method_name);
   num_clusters = size(centroids, 1);
   % Intra- and Inter-cluster distances
   intra_cluster_distances = zeros(num_clusters, 1);
   inter_cluster_distances = zeros(num_clusters, num_clusters);
   for i = 1:num_clusters
       intra_cluster_distances(i) =
mean(pdist2(features(cluster_indices == i, :), centroids(i,
:)));
       for j = 1:num_clusters
           inter_cluster_distances(i, j) = pdist2(centroids(i,
:), centroids(j, :));
       end
   end
   % Visualizing distances
```

```
    figure;
    subplot(1, 2, 1);
    bar(intra_cluster_distances);
    title([method_name ' - Intra-Cluster Distances']);
    xlabel('Cluster'); ylabel('Average Distance');
    subplot(1, 2, 2);
    imagesc(inter_cluster_distances);
    colorbar;
    title([method_name ' - Inter-Cluster Distances']);
    xlabel('Cluster'); ylabel('Cluster');
end
     end
```