Warsaw University of Technology
Faculty of Electronics and Information Technology

ENUME 2025 – Assignment A #10

Estimation of lower limb joint angles

Tutor: Dr. Jakub Wagner

Authors: Zhongtian Dai Artem Zheldak

03.04.2025

Table of Contents

List of mathematical symbols

Mathematical Symbols:

- **αLA**: Left ankle dorsiflexion angle (in radians or degrees).
- **γLA**: Intermediate angle defined as

  $$\gamma_{LA} = \arcsin\left(\mathbf{x}_{LP}^{T}\mathbf{y}_{LT}\right)$$

- **xLT**: Components of the lower leg (LT) coordinate system.
- **xLP**: Components of the foot (LP) coordinate system.
- **MAE**: Mean Absolute Error, computed as

  $$MAE = \frac{1}{n}\sum_{i=1}^{n}|\alpha_{LA,\text{clean}}(i) - \alpha_{LA,\text{filtered}}(i)|$$

- **fc**: Cutoff frequency of the lowpass filter (in Hz).
- **fs**: Sampling frequency (in Hz).
- **n**: Total number of samples

- N -total number of samples (sample size).
- $\alpha_{\text{clean}}(i)$-i-th "clean" (true or reference) angle measurement.
- $\alpha\ filt$- the i-th "filtered" (or predicted/estimated) angle measurement.
- | | - the absolute value

- **Vicon** – Motion capture system used for collecting 3D marker data.
- **LT** – Lower Leg coordinate system.
- **LP** – Foot coordinate system.

- **Hz** – Hertz, a unit of frequency.
- · $Dotproduct$

- arcsin( ) Inverse sine function.

- cos( ) Cosine function.

- × Cross product operator.

- Time -Time vector (s).

- noise std - Standard deviation of the added noise, set to 15 mm.

$ConstructingtheLowerLeg(LT)CoordinateSystem$

$y_{LT} = \frac{\text{lank}-\text{ltio}}{|\text{lank}-\text{ltio}|}$ *Defines the vertical axis (from tibial origin to ankle).*

$z_{LT} = \frac{\text{lfeo}-\text{ltio}}{|\text{lfeo}-\text{ltio}|}$ *Defines the front-back axis (from tibial origin to femoral marker).*

$x_{LT} = \frac{y_{LT} \times z_{LT}}{|y_{LT} \times z_{LT}|}$ *Defines the side-to-side axis via the cross product, ensuring a right-handed system.*

$\boldsymbol{ConstructingtheFoot(LP)CoordinateSystem}$

$x_{LP} = \frac{\text{lfop}-\text{lfoo}}{|\text{lfop}-\text{lfoo}|}$ *Defines the forward axis (from foot origin to forefoot).*

$y_{LP} = \frac{\text{lfol}-\text{lfoo}}{|\text{lfol}-\text{lfoo}|}$ *Defines the lateral axis (from foot origin to lateral marker).*

$z_{LP} = \frac{y_{LP} \times x_{LP}}{|y_{LP} \times x_{LP}|}$ *Defines the vertical axis by the cross product, forming a right-handed system.*

$\boldsymbol{Estimating\ Left\ Ankle\ Dorsiflexion\ Angles}$

$\gamma_{LA} = \text{arcsi n}(x_{LP} \cdot y_{LP})$ *Computes an intermediate angle between the foot axes.*

$\alpha_{LA} = -\text{arcsi n}\left(\frac{x_{LP} \cdot z_{LT}}{\text{co s}(\gamma_{LA})}\right)$ Calculates the ankle dorsiflexion

Introduction

When people walk or run, their body joints move in specific ways. Understanding these movements helps doctors and trainers figure out if someone moves normally or has a problem. In this project, I'm focusing on one important movement: how the ankle joint moves.

To study this, I used special data collected by a system called Vicon, which tracks the exact position of different markers placed on the body during walking. These markers give us precise 3D coordinates, and from these coordinates, I can calculate how the joints are moving. The main idea here is to build two sets of coordinates: one for the lower leg (the calf) and one for the foot. Using these coordinate systems, I applied specific mathematical formulas (from our class materials) to figure out the ankle angle at each moment of walking.

But real-world measurements aren't perfect. In practice, the coordinates we get from motion-capture systems often have small errors or noise. To simulate this realistic situation, I added random "noise" (small errors) to the original data, similar to what you'd see from a less accurate tracking system. Then, I tried using a special mathematical tool called a lowpass Butterworth filter to smooth out these noisy measurements.

The main goal was to test different settings of this filter (specifically, different cutoff frequencies) to see which one gives the best result, meaning it reduces the noise effectively while still accurately showing how the ankle is moving. I evaluated this by comparing the results with and without the noise and calculating the average error (mean absolute error, MAE).

In the rest of this report, I'll explain exactly how I did the calculations, show the results clearly, and discuss what they mean.

Brief Procedure:

1. Data Loading

Load the time data and three-dimensional coordinates of all markers from the “typical_gait.mat” [4] file.

2. Processing Noise-Free Data

Build the lower leg (LT) and foot (LP) coordinate systems using the original data, then calculate the left ankle dorsiflexion angle according to the provided formulas. Convert the result to degrees and plot the time course.

3. Adding Noise

Add Gaussian noise with a standard deviation of 15 mm to all marker coordinates to simulate a lower-accuracy system. Compute and plot the unfiltered joint angles based on the noisy data.

4. Data Filtering

Apply the lowpass filter (using the lpfilt function) [5] to the noisy data for different cutoff frequencies. Recalculate and plot the left ankle dorsiflexion angle from the filtered data.

5. Error Evaluation

Compute the mean absolute error (MAE) between the angles obtained from the noise-free and filtered noisy data for each cutoff frequency, and identify the optimal cutoff frequency based on the lowest MAE.

Methodology and Results of Experiments

Computation of Left Ankle Dorsiflexion Angle from Noise-Free Data

1. **Data Loading:**
   Load the noise-free motion capture data from the "typical_gait.mat" file. This file provides the time vector and 3D coordinates of markers.
2. **Coordinate System Construction:**
   - For each time sample, construct the **Lower Leg (LT)** coordinate system using three specific markers (for example, using the tibial origin, ankle, and another reference point).
   - Similarly, construct the **Foot (LP)** coordinate system using the corresponding foot markers (such as the foot origin, forefoot, and lateral marker).
3. **Angle Calculation:**
   - Using the constructed LT and LP coordinate systems, compute the intermediate angle γ_{LA}(representing the orientation between foot axes).
   - Then calculate the left ankle dorsiflexion angle α_{LA} by projecting the foot's axis onto the lower leg's vertical axis, applying the provided mathematical formula.
   - Convert the computed angle from radians to degrees.
4. **Plotting:**
   Plot the resulting left ankle dorsiflexion angle over time, which serves as the baseline (noise-free) signal for further comparisons with noisy and filtered data.

This sequential process provides the reference angle data before any noise is introduced, ensuring a basis for evaluating filtering performance later on.

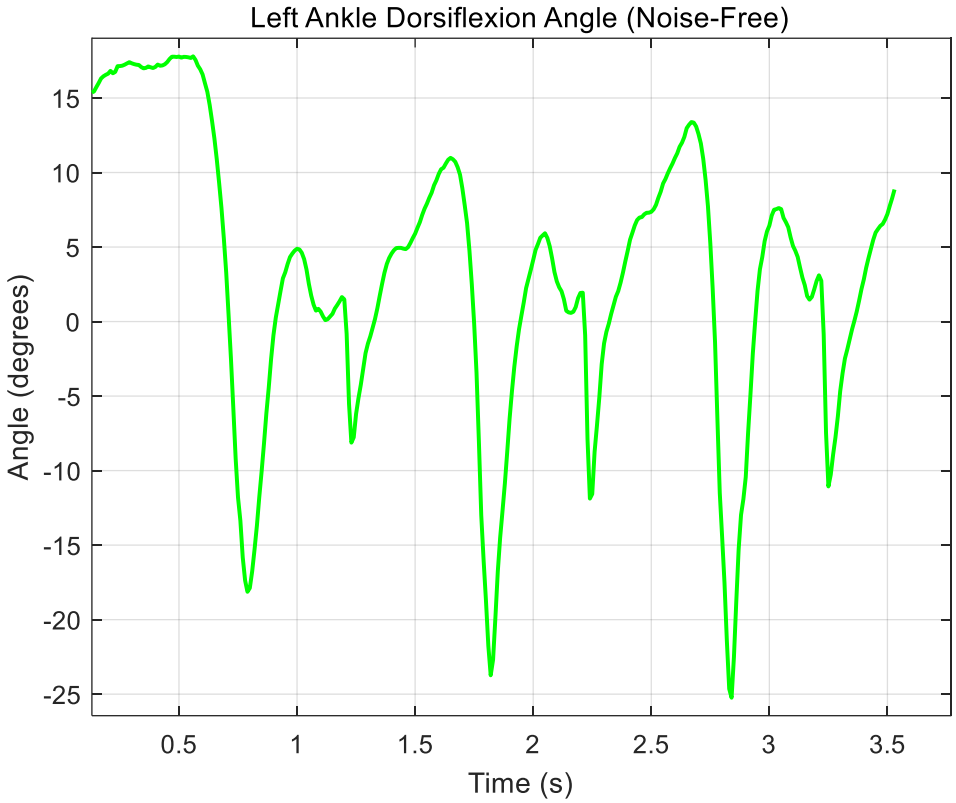

Figure 1 Left Ankle Dorsiflexion Angle (Noise-Free) based on data from "typical_gaint.mat"

Simulation of Measurement Noise

In real experiments, motion-capture systems often have some measurement errors, small random differences between the true positions and what the system reports.
To simulate these errors, I added random noise (Gaussian noise) to the original marker data. This noise had an average value of zero and a standard deviation of 15 mm. After adding this artificial noise to each marker coordinate, I repeated the calculation of the left ankle dorsiflexion angle using the noisy data.
By comparing the noisy angles to the clean angles from Step 1, it was easy to understand how much the noise affected our results. This step helps in figuring out how reliable our angle calculation method is under realistic conditions.

f_c_values = [0.1, 0.5, 1, 2, 2.5, 3, 3.5, 4,4.5 5, 5.5,6, 10, 15, 20] Hz

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|\alpha_{\text{clean}}(i) - \alpha_{\text{filt}}(i)|$$

$$M_{\text{noisy}} = M + 15 \cdot \text{randn}\big(\text{size}(M)\big)$$

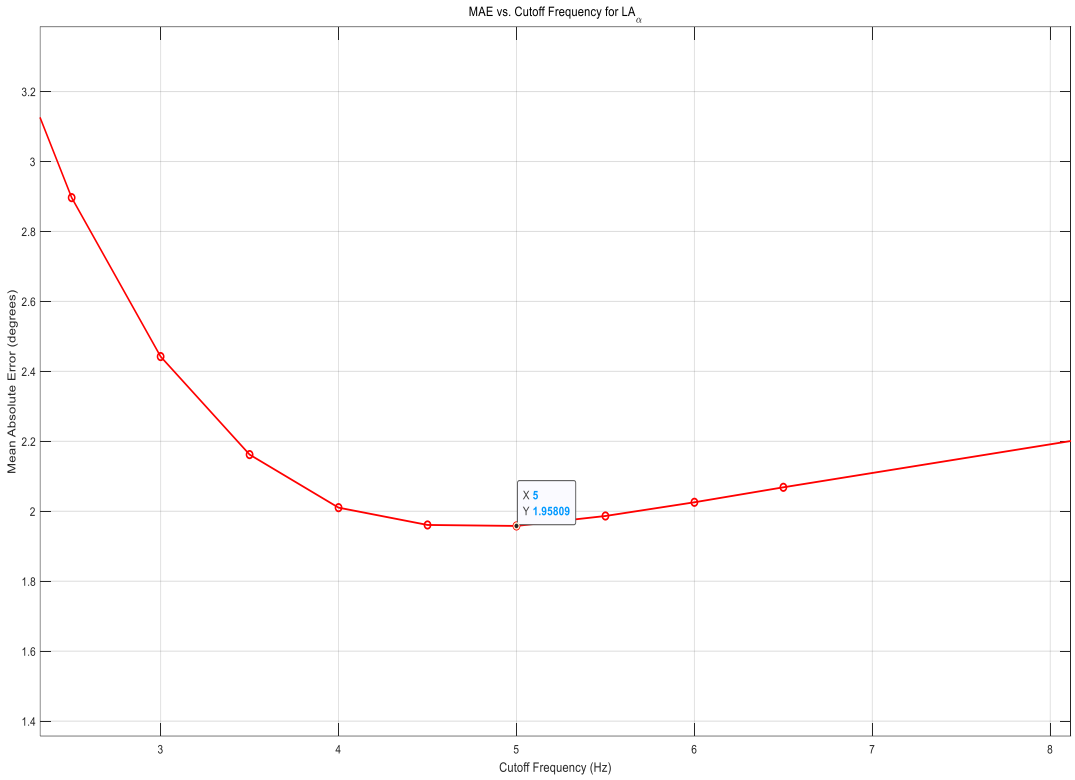

Figure 2

Due to the inherent randomness in the noise generation, each simulation run produces slightly different results. To obtain more reliable estimates, the simulation was repeated 100 times, and the average values were computed. The aggregated outcomes are presented in the following table.

Cutoff Frequency (Hz)	Average MAE (degrees)
0.1	8.182
0.5	6.0958
1.0	5.0938
2.0	3.4682
2.5	2.9207
3.0	2.5602
3.5	2.3205
4.0	2.1701
4.5	2.0863
5.0	2.0494
5.5	2.0431
6.0	2.0562
6.5	2.0814
10.0	2.387
15.0	2.8567
20.0	3.2947

Table 1

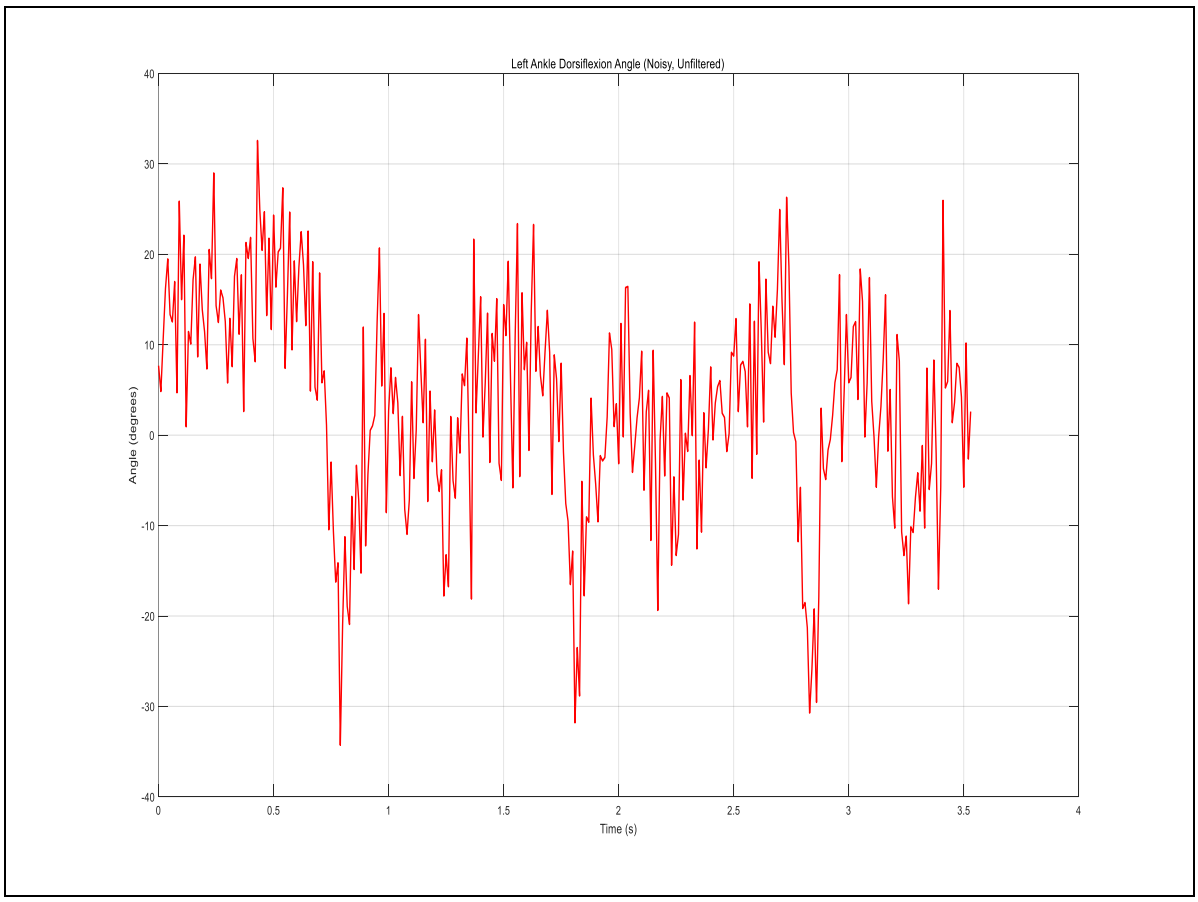
Left Ankle Dorsiflexion Angle (Noisy, Unfiltered)
Angle (degrees)
Time (s)
40
30
20
10
0
-10
-20
-30
-40
0
0.5
1
1.5
2
2.5
3
3.5
4

Figure 3

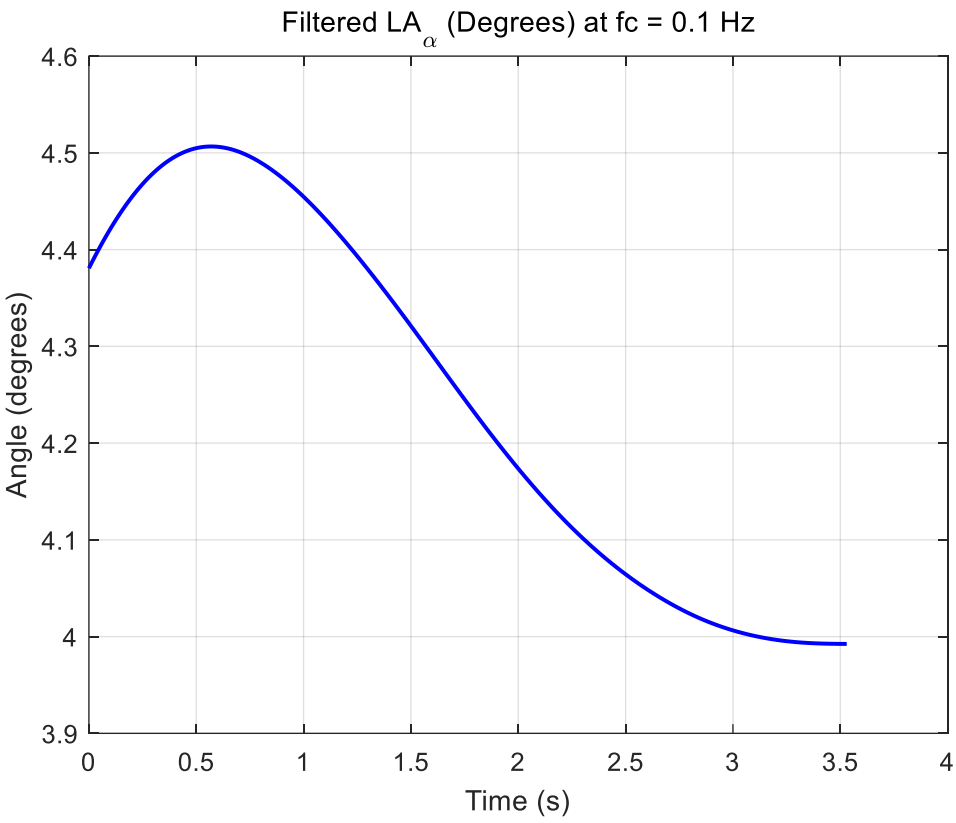

Figure 4

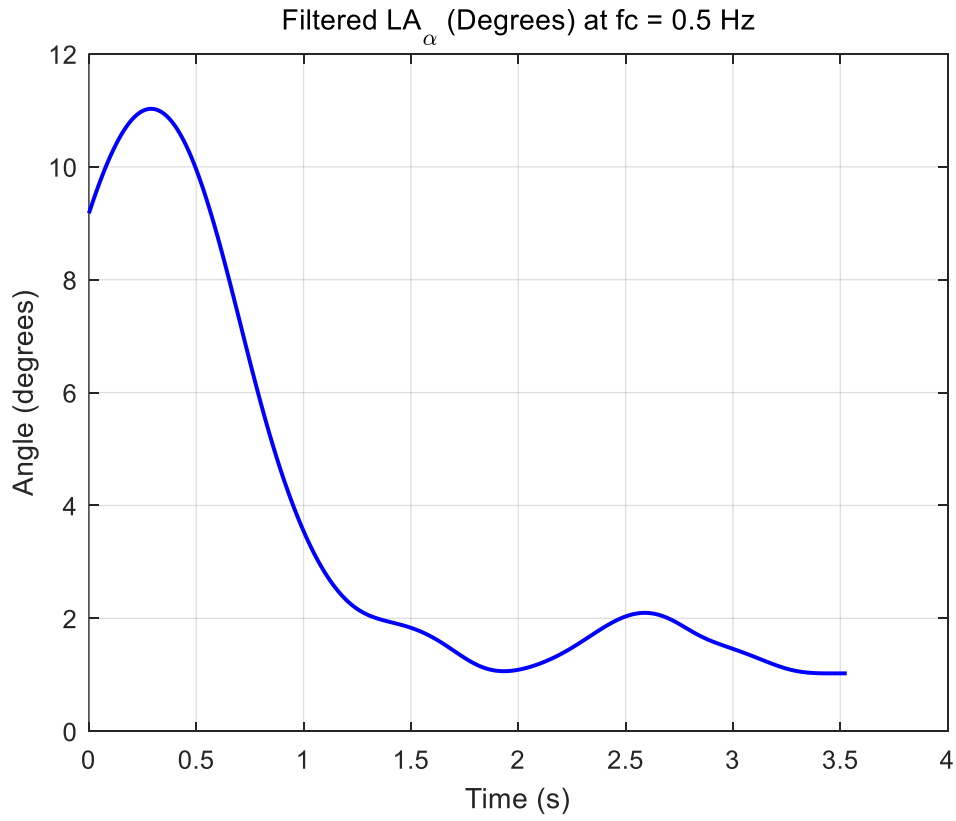

Figure 5

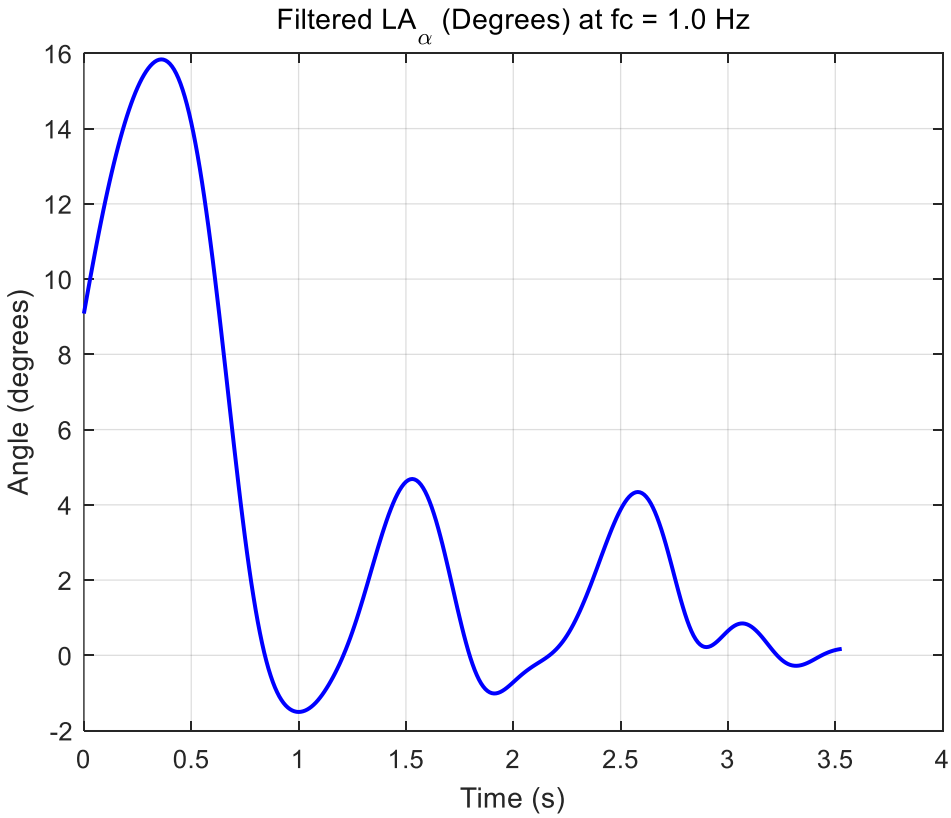
Filtered LA$_\alpha$ (Degrees) at fc = 1.0 Hz
Angle (degrees)
Time (s)
16
14
12
10
8
6
4
2
0
-2
0
0.5
1
1.5
2
2.5
3
3.5
4

Figure 6

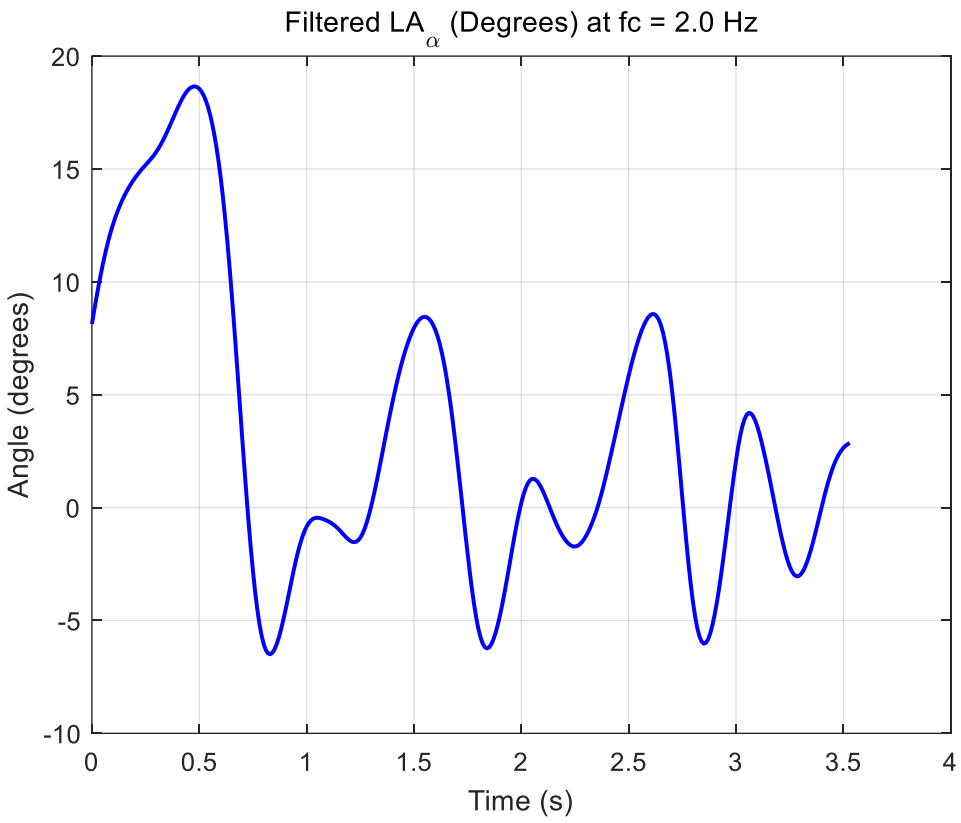

Figure 7

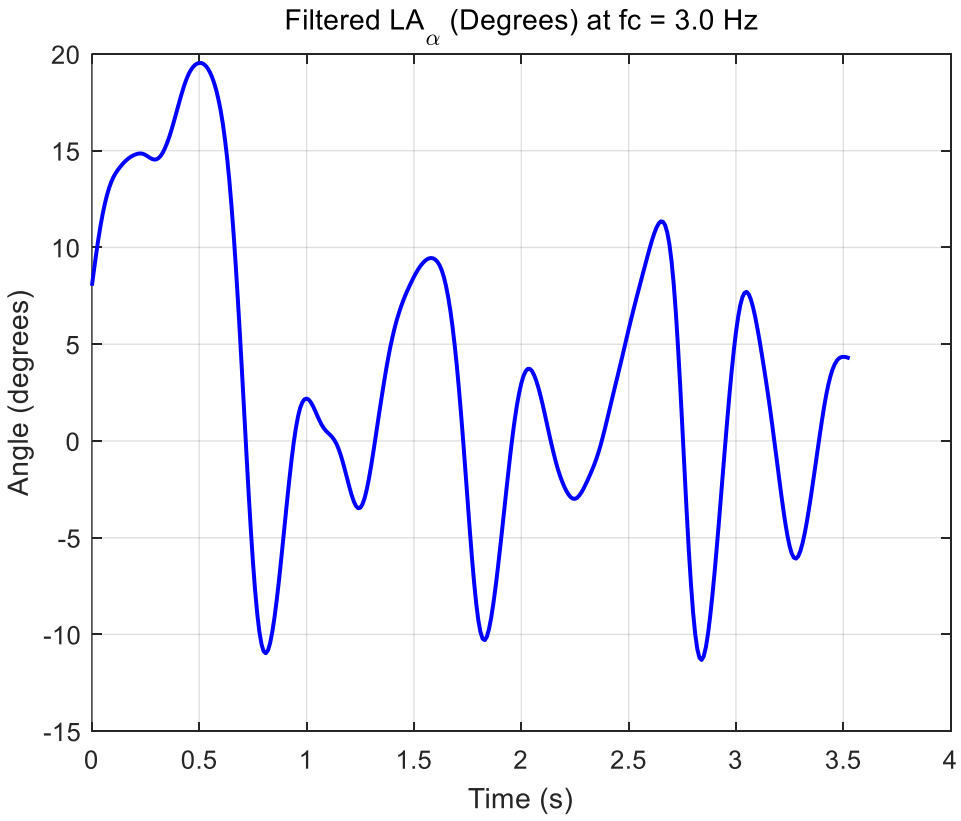

Figure 8

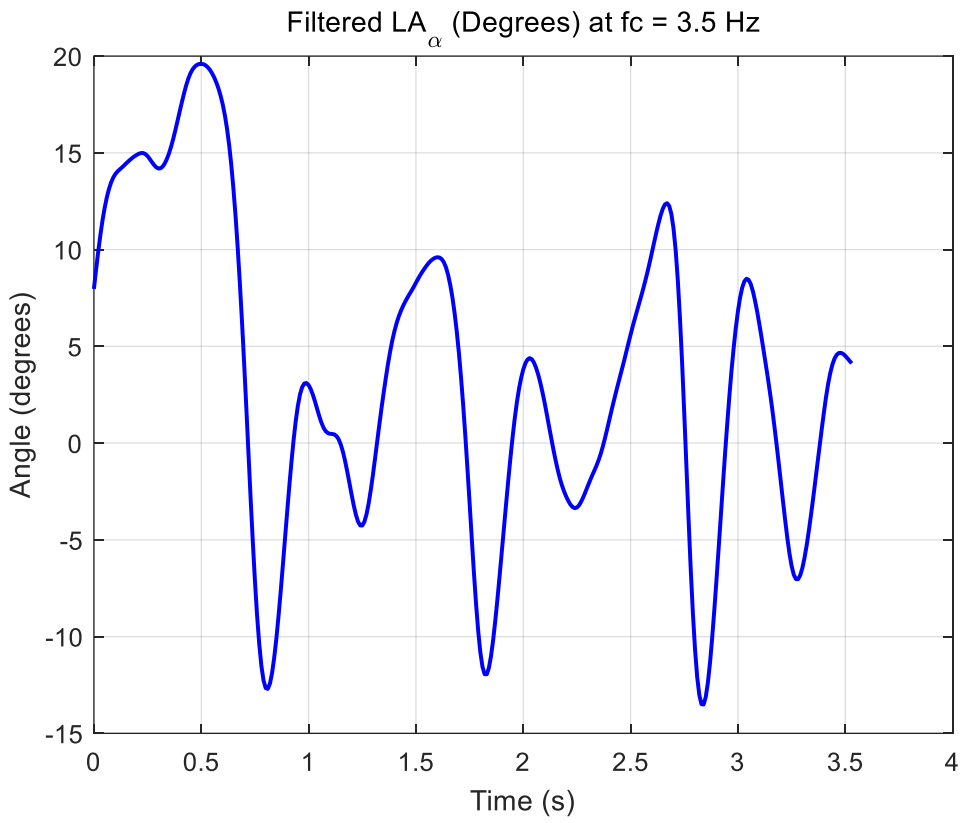

Figure 9

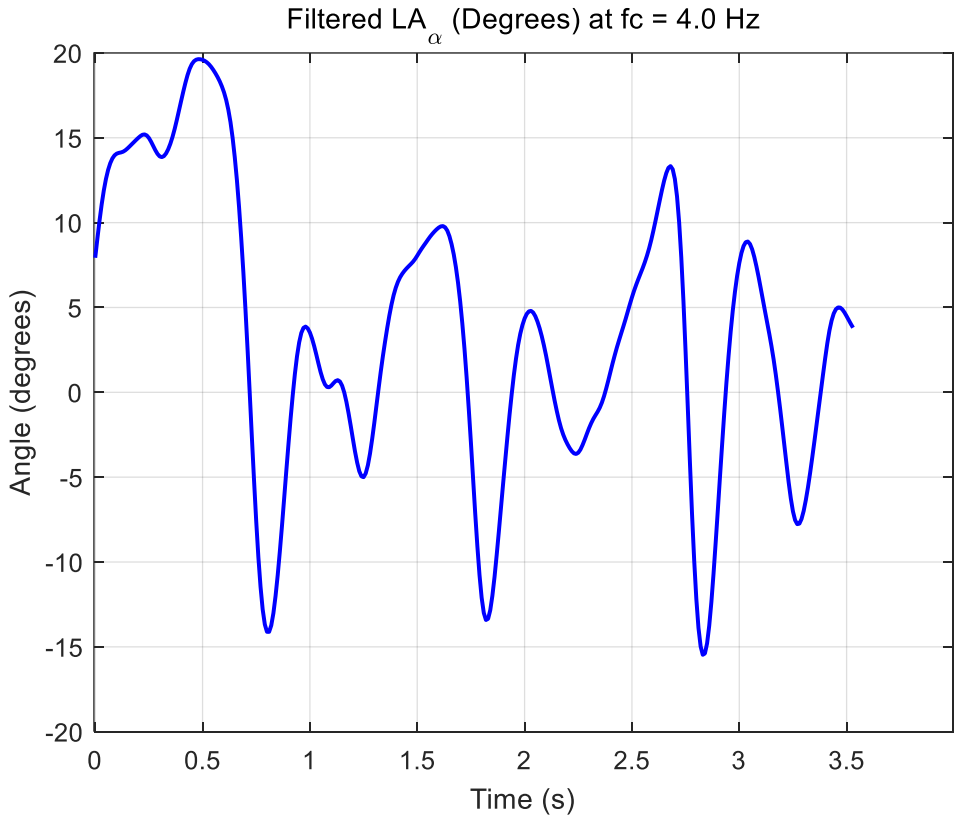

Figure 10

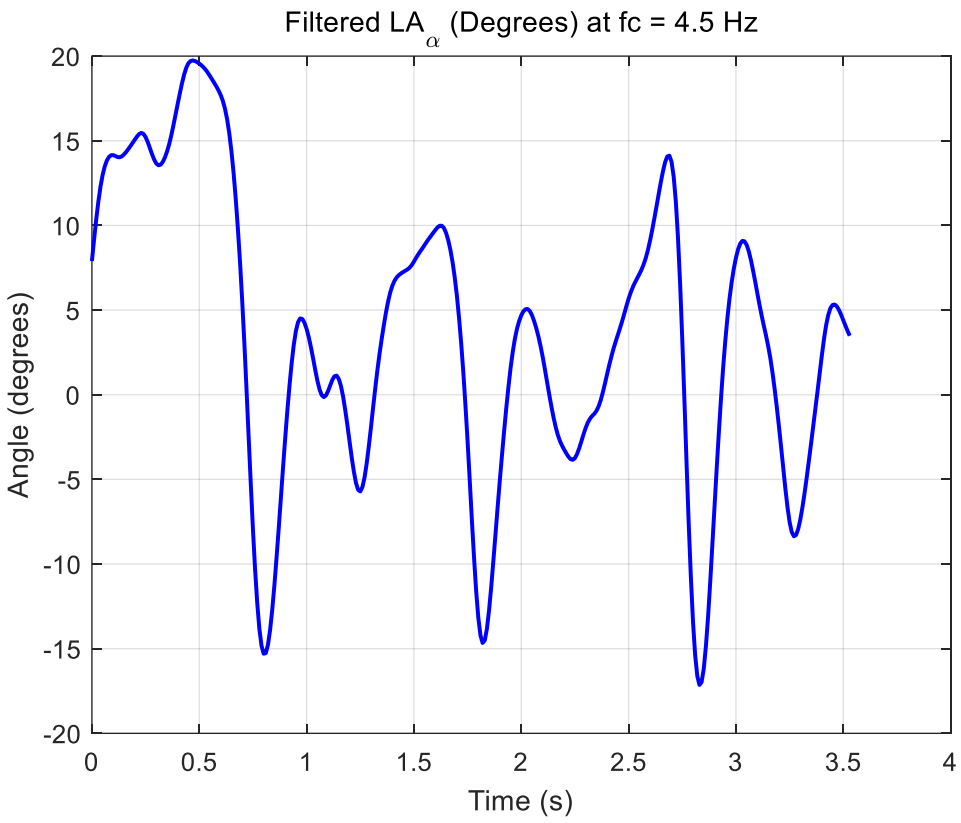

Figure 11

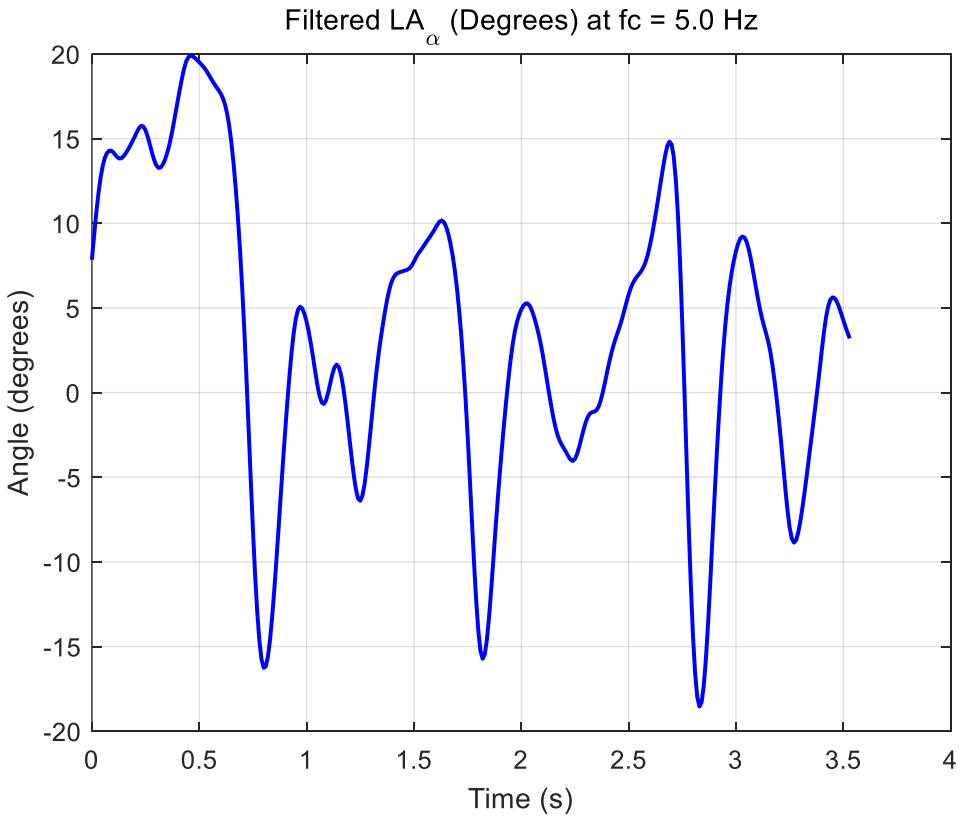

Figure 12

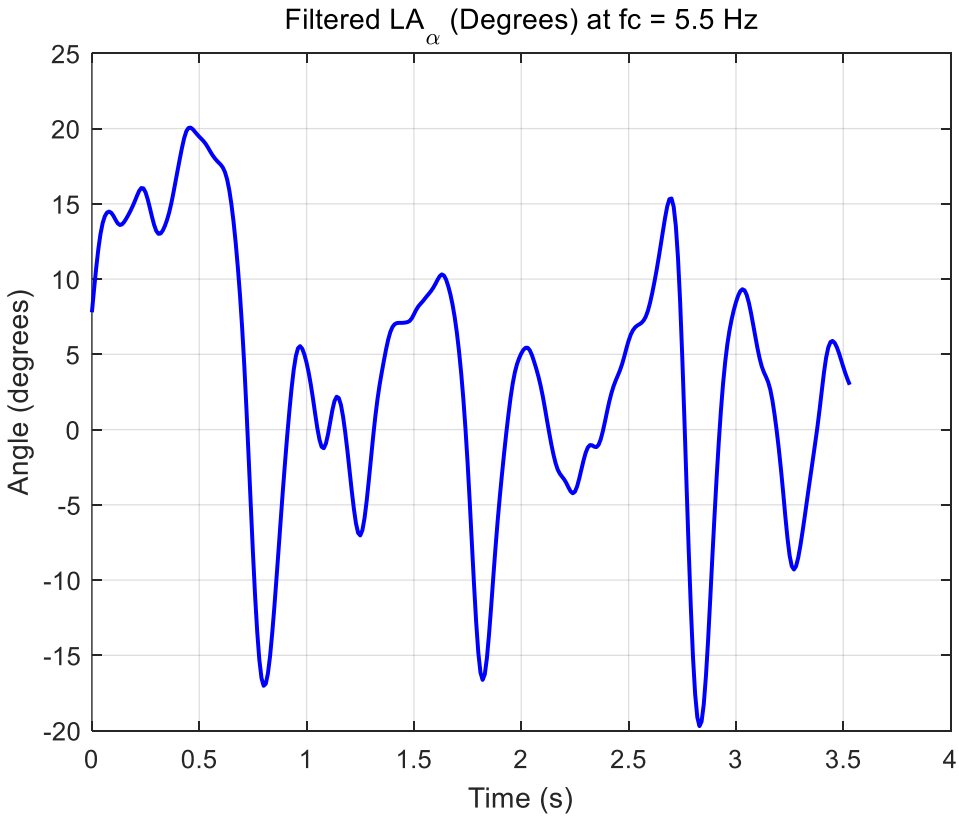

Figure 13

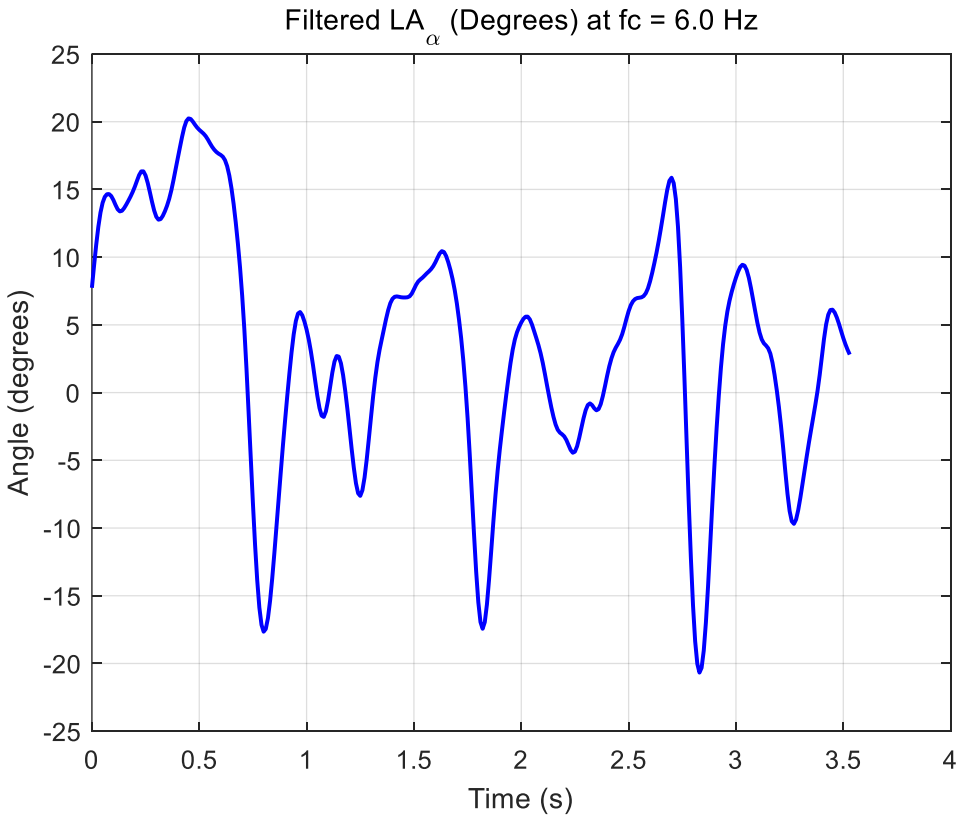

Figure 14

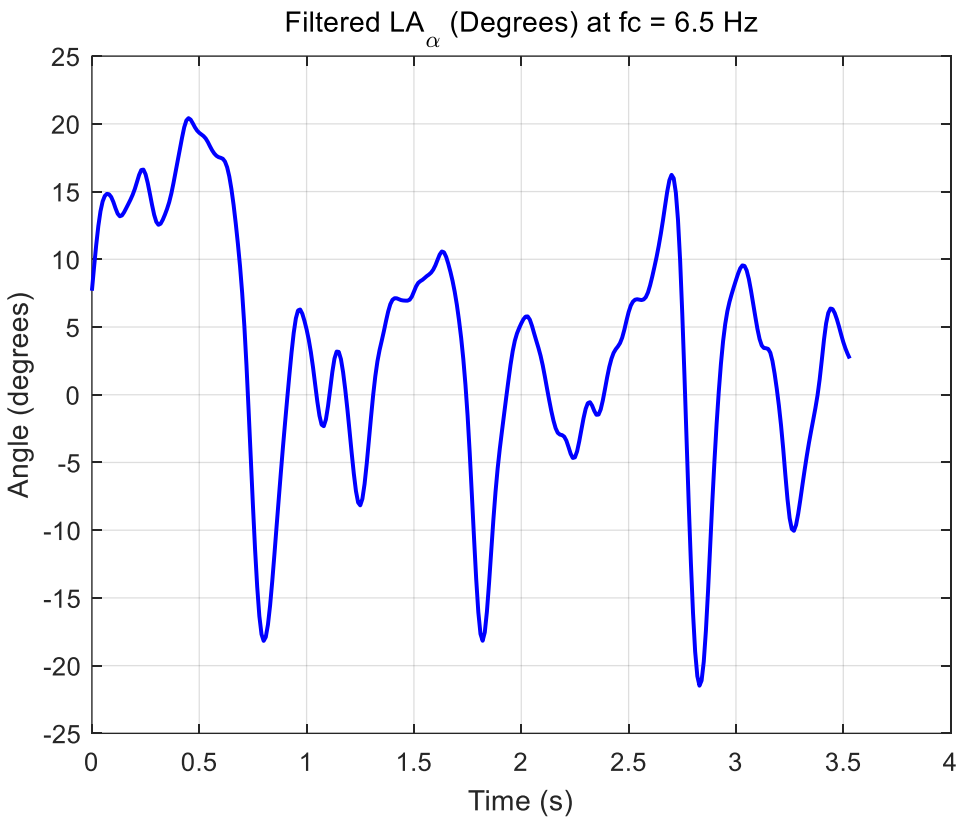

Figure 15

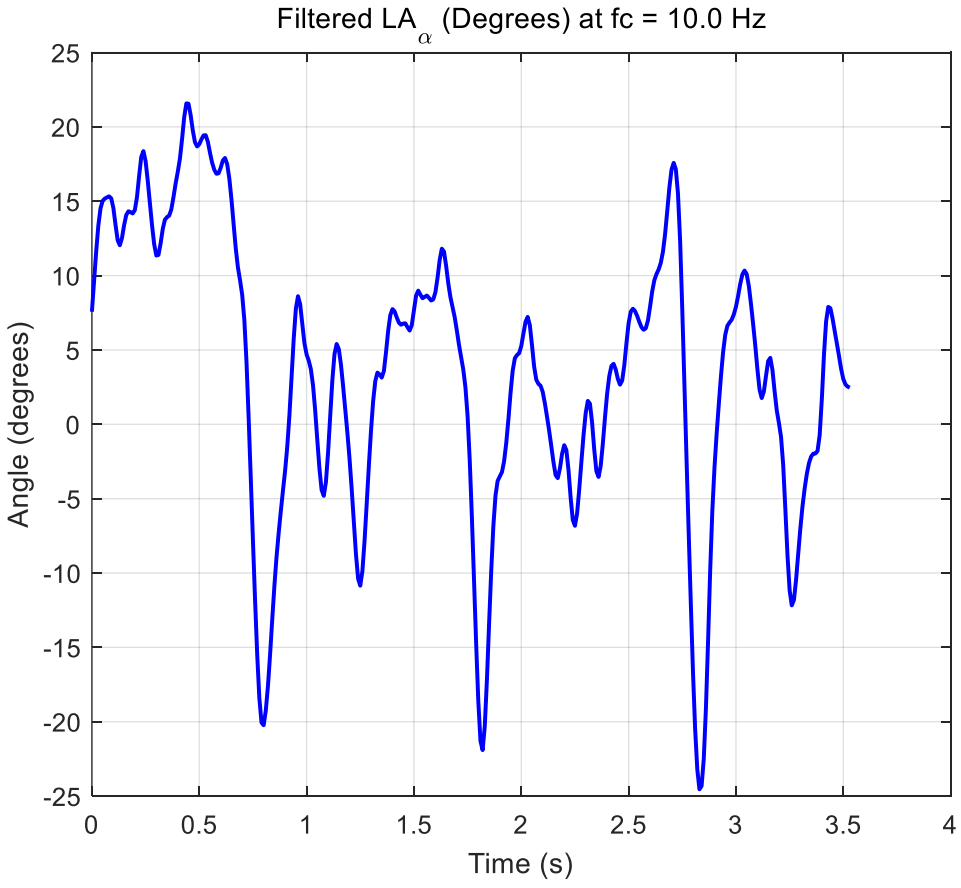

Figure 16

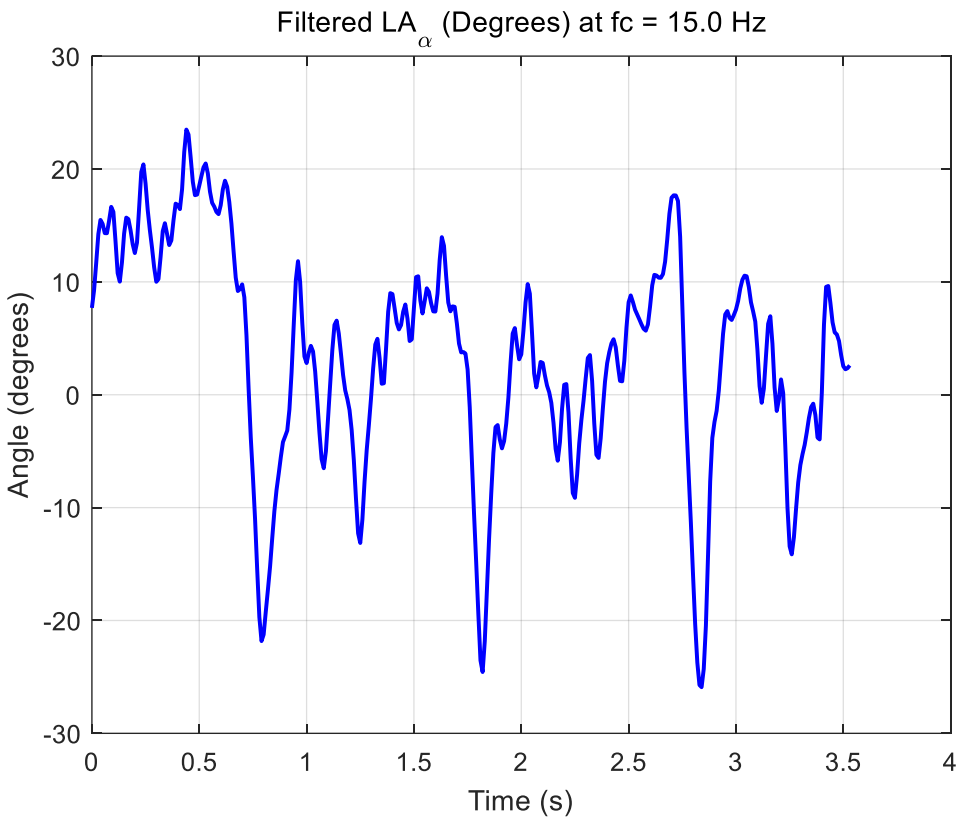

Figure 17

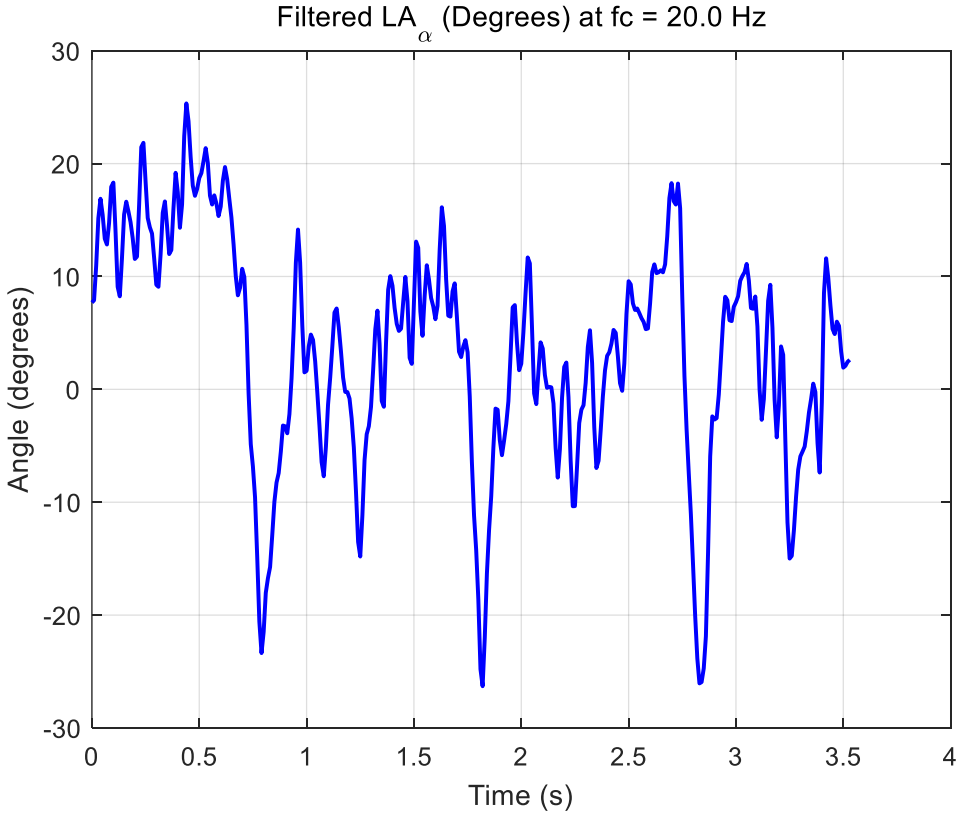

Figure 18

```
=== Step 5: Average MAE for each cutoff frequency after multiple trials ===
Cutoff Frequency (Hz)    Average MAE (degrees)
-------------------------------------------------
      0.1                    8.1820
      0.5                    6.0958
      1.0                    5.0938
      2.0                    3.4682
      2.5                    2.9207
      3.0                    2.5602
      3.5                    2.3205
      4.0                    2.1701
      4.5                    2.0863
      5.0                    2.0494
      5.5                    2.0431
      6.0                    2.0562
      6.5                    2.0814
      10.0                   2.3870
      15.0                   2.8567
      20.0                   3.2947
```

Figure 19

Discussion

Effect of Gaussian Noise

Adding Gaussian noise (figure 3) with zero mean and a standard deviation of 15 mm causes the originally smooth marker data to exhibit significant random fluctuations. When these perturbed coordinates are used to compute joint angles, the noise gets amplified, resulting in pronounced random peaks and valleys that obscure physiological characteristics and key events of normal gait. This increased variability can lead to misdiagnosis or erroneous biomechanical interpretations, highlighting the necessity of applying an appropriate lowpass filter to effectively remove the noise while preserving essential signal dynamics.

Role of Lowpass Filtering

Lowpass filtering effectively reduces the high-frequency noise present in the marker data, resulting in smoother joint angle estimates that better reflect the true underlying movements. By eliminating rapid fluctuations caused by Gaussian noise, the filter preserves the essential dynamics of the gait while removing unwanted artifacts. The

appropriate selection of the cutoff frequency is critical: too low a cutoff may excessively smooth the data and remove important signal details, whereas too high a cutoff may not adequately attenuate the noise. Overall, lowpass filtering plays a vital role in enhancing the reliability and clinical relevance of biomechanical analysis.

Optimization of Cutoff Frequency (fc)

When the cutoff frequency is less than 5.5-6 Hz, the filter is very restrictive (fig 4-12) , which means it removes a lot of high-frequency components. This results in an overly smooth signal that can lose important details and rapid changes in the gait dynamics. While the noise is reduced, some of the actual motion characteristics may also be attenuated, potentially distorting the true movement patterns.
On the other hand, when the cutoff frequency is more than 5.5-6 Hz, the filter becomes less aggressive (fig 15-18) , allowing more high-frequency components to pass through. Although this means that the essential gait dynamics are preserved, it also means that more of the unwanted noise remains in the signal. As a result, the estimated joint angles may still exhibit considerable variability, and the noise reduction may not be sufficient to improve the measurement accuracy.
Thus, around 5.5-6 Hz (fig 13-14) represents an optimal balance where enough noise is removed to reduce errors while still preserving the essential features of the motion.

Statistical Considerations

Although the simulation was repeated 100 times, the optimal cutoff frequency still fluctuated between 5.5 and 6 Hz. (fig 19) This variability reflects inherent uncertainties in the simulation process. The random nature of Gaussian noise, combined with small variations in the measurement data, leads to slight differences in the computed Mean Absolute Error (MAE) for each run. As a result, even with a large number of iterations, the optimal fc does not converge to a single fixed value but rather shows a range where the error is minimized.
This statistical variability highlights a potential limitation: the determination of the optimal cutoff frequency is sensitive to the specific noise realizations and experimental conditions. To address this, it may be beneficial to perform additional simulations or use more robust statistical methods to better estimate and confirm the optimal fc. Furthermore, reporting confidence intervals or standard deviations for the estimated cutoff frequency can provide a clearer picture of the uncertainty involved.

Conclusion

This report presented a method for estimating the left ankle dorsiflexion angle from motion-captured marker data by constructing the LT and LP coordinate systems and applying appropriate mathematical formulas. The method was validated by comparing noise-free data to data corrupted with simulated Gaussian noise. A lowpass Butterworth

filter was applied at various cutoff frequencies, and the optimal filter setting (5.5-6 Hz) was identified based on the minimum mean absolute error (MAE). These results provide valuable insight into accurately capturing joint kinematics in the presence of measurement noise, which is critical in clinical gait analysis and related applications.

References

[1] R. Z. Morawski, *Lecture Notes for the Course Numerical Methods*, Warsaw University of Technology, Faculty of Electronics and Information Technology, Spring Semester 2023/24.

[2] J. Wagner, *Algorithms for Calibration of Spectrophotometric Analyser of Edible Oils Mixtures*, master's Thesis, Warsaw University of Technology, Faculty of Electronics and Information Technology, 2013.

[3*] Enume 2025 Introduction to MATLAB.pdf Warsaw* University of Technology, Faculty of Electronics and Information Technology Numerical Methods (ENUME), Spring Semester 2025 Dr. Jakub Wagner Institute of Radioelectronics and Multimedia Technology

[4] *typical_gait.mat,* Vicon motion capture data, ENUME 2025 – Assignment Materials, Warsaw University of Technology, Spring Semester 2025.

[5] lpfilt.m, ENUME 2025 – Assignment A – Estimation of Lower Limb Joint Angles, Warsaw University of Technology, Spring Semester 2025.

Matlab code :

```
function complete_solution_no_simplifications()


    clear; close all; clc;


    load('typical_gait.mat');


    n = size(lfoo, 1);
    dt = mean(diff(time));
    fs = 1/dt;

    %% Compute Noise-Free Left Ankle Dorsiflexion Angle α_LA
    LA_alpha_clean_rad = zeros(n,1);
    for i = 1:n
        %  (28)~(30)
        [x_LT, y_LT, z_LT] = construct_LT_coords(ltio(i,:),
lank(i,:), lfeo(i,:));
        %  (34)~(36)
        [x_LP, y_LP, z_LP] = construct_LP_coords(lfoo(i,:),
lfop(i,:), lfol(i,:));
        % formulas (13) and (15)
        LA_alpha_clean_rad(i) = compute_alpha_LA_strict(x_LP, y_LP,
z_LT);
    end
    LA_alpha_clean_deg = rad2deg(LA_alpha_clean_rad);

    figure('Name','Noise-Free LA_\alpha (Degrees)');
    plot(time, LA_alpha_clean_deg, 'g', 'LineWidth', 1.5);
    grid on;
    xlabel('Time (s)');
    ylabel('Angle (degrees)');
    title('Left Ankle Dorsiflexion Angle (Noise-Free)');

   % Add Noise and Compute Unfiltered Angle
    noise_std = 15;
```

```
    ltio_noisy = ltio + noise_std * randn(size(ltio));
    lank_noisy = lank + noise_std * randn(size(lank));
    lfeo_noisy = lfeo + noise_std * randn(size(lfeo));
    lfoo_noisy = lfoo + noise_std * randn(size(lfoo));
    lfop_noisy = lfop + noise_std * randn(size(lfop));
    lfol_noisy = lfol + noise_std * randn(size(lfol));

    LA_alpha_noisy_deg = zeros(n,1);
    for i = 1:n
        [x_LT, y_LT, z_LT] = construct_LT_coords(ltio_noisy(i,:),
lank_noisy(i,:), lfeo_noisy(i,:));
        [x_LP, y_LP, z_LP] = construct_LP_coords(lfoo_noisy(i,:),
lfop_noisy(i,:), lfol_noisy(i,:));
        alpha_rad = compute_alpha_LA_strict(x_LP, y_LP, z_LT);
        LA_alpha_noisy_deg(i) = rad2deg(alpha_rad);
    end

    figure('Name','Noisy, Unfiltered LA_\alpha (Degrees)');
    plot(time, LA_alpha_noisy_deg, 'r', 'LineWidth', 1.5);
    grid on;
    xlabel('Time (s)');
    ylabel('Angle (degrees)');
    title('Left Ankle Dorsiflexion Angle (Noisy, Unfiltered)');

    %%  Single Filtering: Try Different Cutoff Frequencies and
Compute MAE
    fc_values = [0.1, 0.5, 1, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 ,6, 6.5,
10, 15, 20];
    MAE_values = zeros(size(fc_values));

    for iCutoff = 1:length(fc_values)
        fc = fc_values(iCutoff);

        ltio_filt = zeros(size(ltio_noisy));
        lank_filt = zeros(size(lank_noisy));
        lfeo_filt = zeros(size(lfeo_noisy));
        lfoo_filt = zeros(size(lfoo_noisy));
        lfop_filt = zeros(size(lfop_noisy));
        lfol_filt = zeros(size(lfol_noisy));
        for col = 1:3
            ltio_filt(:,col) = lpfilt(ltio_noisy(:,col), fc, fs);
            lank_filt(:,col) = lpfilt(lank_noisy(:,col), fc, fs);
            lfeo_filt(:,col) = lpfilt(lfeo_noisy(:,col), fc, fs);
            lfoo_filt(:,col) = lpfilt(lfoo_noisy(:,col), fc, fs);
```

```
            lfop_filt(:,col) = lpfilt(lfop_noisy(:,col), fc, fs);
            lfol_filt(:,col) = lpfilt(lfol_noisy(:,col), fc, fs);
        end

        LA_alpha_filt_deg = zeros(n,1);
        for k = 1:n
            [x_LT, y_LT, z_LT] = construct_LT_coords(ltio_filt(k,:),
lank_filt(k,:), lfeo_filt(k,:));
            [x_LP, y_LP, z_LP] = construct_LP_coords(lfoo_filt(k,:),
lfop_filt(k,:), lfol_filt(k,:));
            alpha_rad = compute_alpha_LA_strict(x_LP, y_LP, z_LT);
            LA_alpha_filt_deg(k) = rad2deg(alpha_rad);
        end

        figure('Name', sprintf('Filtered LA_\\alpha (Degrees) at fc
= %.1f Hz', fc));
        plot(time, LA_alpha_filt_deg, 'b', 'LineWidth', 1.5);
        grid on;
        xlabel('Time (s)');
        ylabel('Angle (degrees)');
        title(sprintf('Filtered LA_\\alpha (Degrees) at fc = %.1f
Hz', fc));

        MAE_values(iCutoff) = mean(abs(LA_alpha_clean_deg -
LA_alpha_filt_deg));
        fprintf('fc = %.1f Hz, MAE = %.4f degrees\n', fc,
MAE_values(iCutoff));
    end

    figure('Name','MAE vs. fc');
    plot(fc_values, MAE_values, 'r-o', 'LineWidth', 1.5,
'MarkerSize', 6);
    grid on;
    xlabel('Cutoff Frequency (Hz)');
    ylabel('Mean Absolute Error (degrees)');
    title('MAE vs. Cutoff Frequency for LA_\alpha');

    [bestMAE, bestIdx] = min(MAE_values);
    bestFc = fc_values(bestIdx);
    fprintf('\nBest cutoff frequency: fc = %.1f Hz, MAE = %.4f
degrees\n', bestFc, bestMAE);


    %% Multiple Trials
```

```
    nTrials = 100;
    MAE_trials = zeros(length(fc_values), nTrials);

    for trial = 1:nTrials
        ltio_noisy_trial = ltio + noise_std * randn(size(ltio));
        lank_noisy_trial = lank + noise_std * randn(size(lank));
        lfeo_noisy_trial = lfeo + noise_std * randn(size(lfeo));
        lfoo_noisy_trial = lfoo + noise_std * randn(size(lfoo));
        lfop_noisy_trial = lfop + noise_std * randn(size(lfop));
        lfol_noisy_trial = lfol + noise_std * randn(size(lfol));

        for iCutoff = 1:length(fc_values)
            fc = fc_values(iCutoff);

            ltio_filt_trial = zeros(size(ltio_noisy_trial));
            lank_filt_trial = zeros(size(lank_noisy_trial));
            lfeo_filt_trial = zeros(size(lfeo_noisy_trial));
            lfoo_filt_trial = zeros(size(lfoo_noisy_trial));
            lfop_filt_trial = zeros(size(lfop_noisy_trial));
            lfol_filt_trial = zeros(size(lfol_noisy_trial));
            for col = 1:3
                ltio_filt_trial(:,col) =
lpfilt(ltio_noisy_trial(:,col), fc, fs);
                lank_filt_trial(:,col) =
lpfilt(lank_noisy_trial(:,col), fc, fs);
                lfeo_filt_trial(:,col) =
lpfilt(lfeo_noisy_trial(:,col), fc, fs);
                lfoo_filt_trial(:,col) =
lpfilt(lfoo_noisy_trial(:,col), fc, fs);
                lfop_filt_trial(:,col) =
lpfilt(lfop_noisy_trial(:,col), fc, fs);
                lfol_filt_trial(:,col) =
lpfilt(lfol_noisy_trial(:,col), fc, fs);
            end

            LA_alpha_filt_trial_deg = zeros(n,1);
            for k = 1:n
                [x_LT, y_LT, z_LT] =
construct_LT_coords(ltio_filt_trial(k,:), lank_filt_trial(k,:),
lfeo_filt_trial(k,:));
                [x_LP, y_LP, z_LP] =
construct_LP_coords(lfoo_filt_trial(k,:), lfop_filt_trial(k,:),
lfol_filt_trial(k,:));
```

```
                alpha_rad = compute_alpha_LA_strict(x_LP, y_LP,
z_LT);
                LA_alpha_filt_trial_deg(k) = rad2deg(alpha_rad);
            end

            MAE_trials(iCutoff, trial) = mean(abs(LA_alpha_clean_deg
- LA_alpha_filt_trial_deg));
        end
    end

    avg_MAE = mean(MAE_trials, 2);

    fprintf('\n=== Step 5: Average MAE for each cutoff frequency
after multiple trials ===\n');
    fprintf('Cutoff Frequency (Hz)    Average MAE (degrees)\n');
    fprintf('-----------------------------------------------\n');
    for iCutoff = 1:length(fc_values)
        fprintf('      %.1f                    %.4f\n',
fc_values(iCutoff), avg_MAE(iCutoff));
    end
end


%%  other Function
 %(28)~(30)
function [x_LT, y_LT, z_LT] = construct_LT_coords(c_ltio, c_lank,
c_lfeo)
    % (29)
    vec_y = c_lank - c_ltio;
    y_LT = vec_y / norm(vec_y);

    % (30)
    vec_z = c_lfeo - c_ltio;
    z_LT = vec_z / norm(vec_z);

    % (28)
    cross_x = cross(y_LT, z_LT);
    x_LT = cross_x / norm(cross_x);
end

% (34)~(36)
function [x_LP, y_LP, z_LP] = construct_LP_coords(c_lfoo, c_lfop,
c_lfol)
    % (34)
```

```
    vec_x = c_lfop - c_lfoo;
    x_LP = vec_x / norm(vec_x);

    % (35)
    vec_y = c_lfol - c_lfoo;
    y_LP = vec_y / norm(vec_y);

    % (36)
    cross_z = cross(y_LP, x_LP);
    z_LP = cross_z / norm(cross_z);
end

%% (13) & (15)) ==========
function alpha_LA = compute_alpha_LA_strict(x_LP, y_LP, z_LT)
    % (15)
    gamma_LA = asin(dot(x_LP, y_LP));

    cos_gamma = cos(gamma_LA);
    if abs(cos_gamma) < 1e-8
        cos_gamma = 1e-8;  % Prevent division by zero
    end

    % (13)
    alpha_LA = -asin(dot(x_LP, z_LT) / cos_gamma);
end
```