Akademia Górniczo-Hutnicza

Faculty of Mechanical Engineering and Robotics

Mechatronic Engineering

Kinematics and Dynamics of Mechatronic Systems

Lab Report

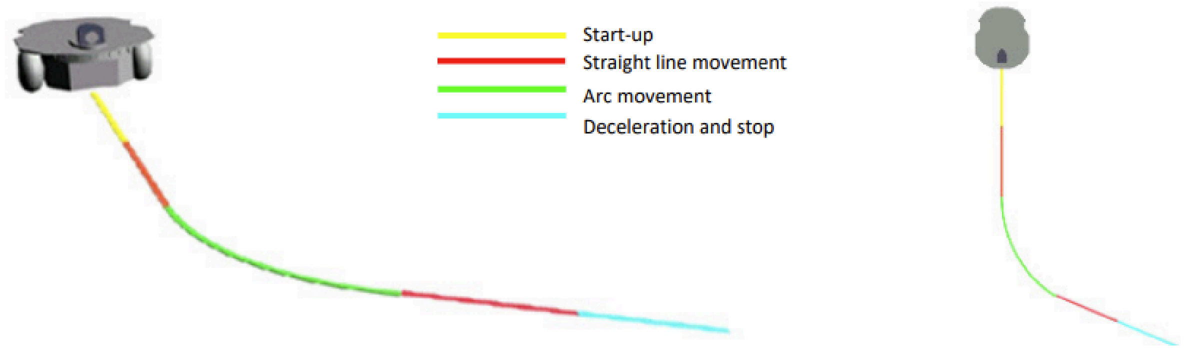
Kinematics

Author: Szymon Świątek

Kraków, 04 June 2025

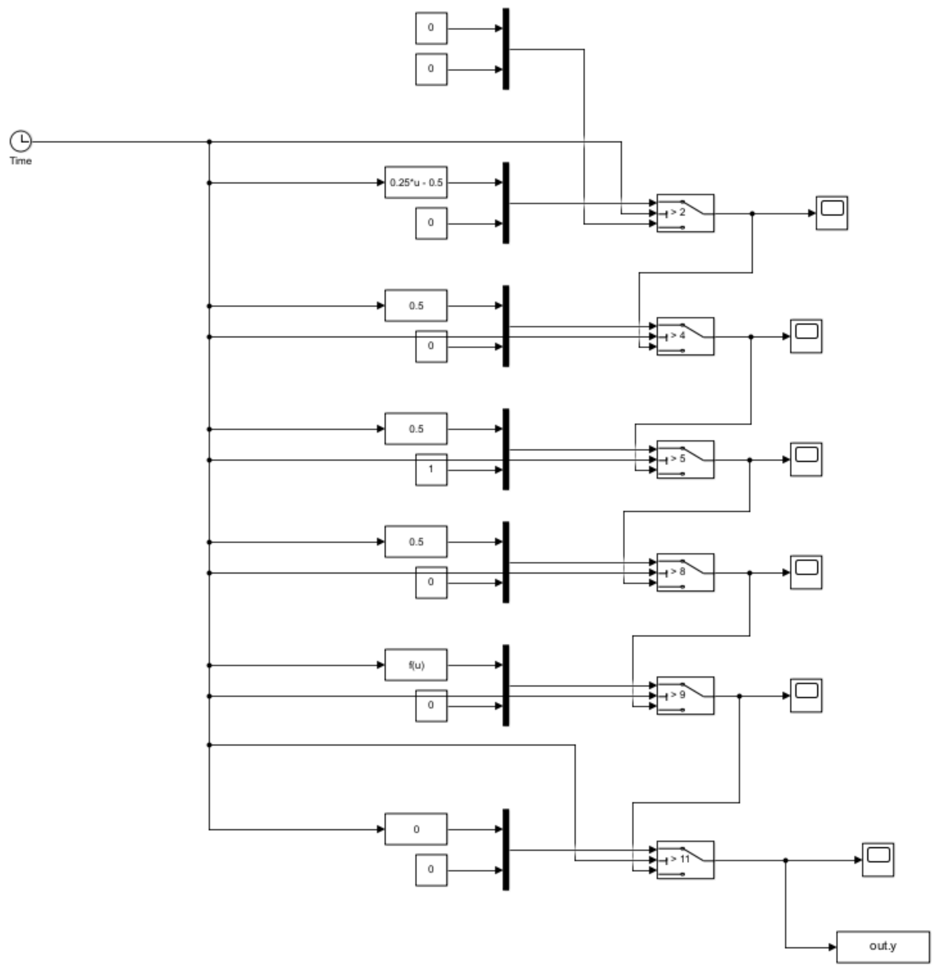
Trajectory Definition

Trajectory was created based on the following diagram. The different time intervals and characteristics defining the trajectory are summarized in the table below. To calculate the velocity a Simulink trajectory generator model was created. A diagram for the Simulink model can be found in the next section. After generating the initial trajectory, the results were further modified to eliminate sharp changes of velocity.

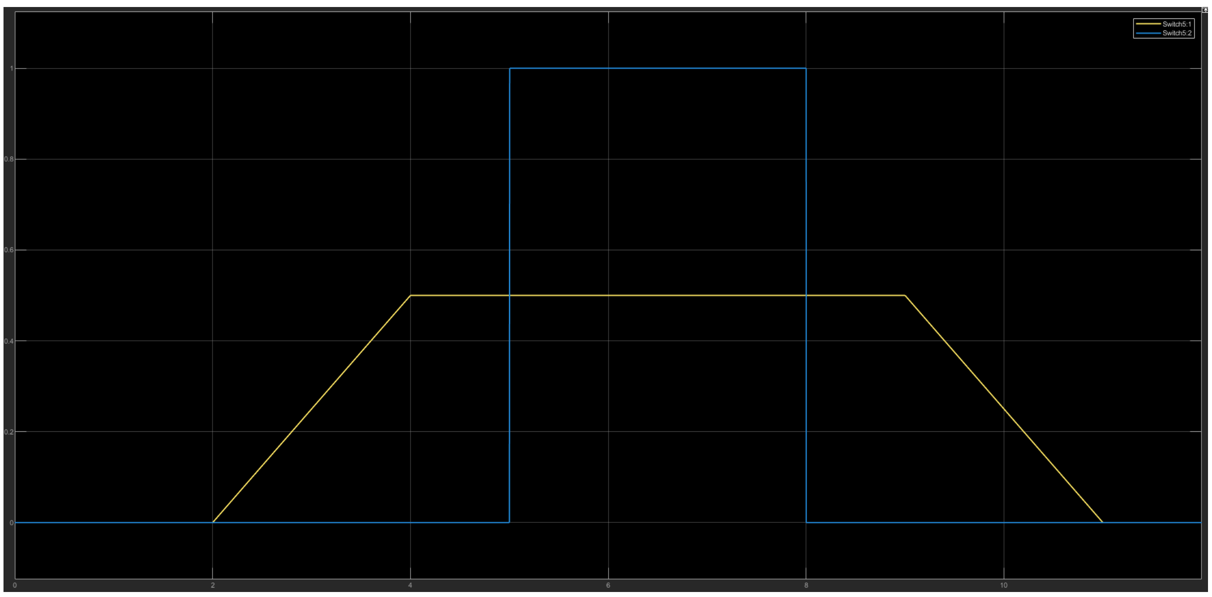


Trajectory plan	
Point A velocity	$v_a = 0.5\ [m/s]$
Start-up at	$2\ [s]$
Acceleration	$2\ [s]$
Straight line movement No.1	$1\ [s]$
Arc movement	$3\ [s]$ with $R = 3\ [m]$
Straight line movement No.2	$1\ [s]$
Deceleration and stop	$2\ [s]$

Simulink Model



Simulink model



Velocity and trajectory signal output

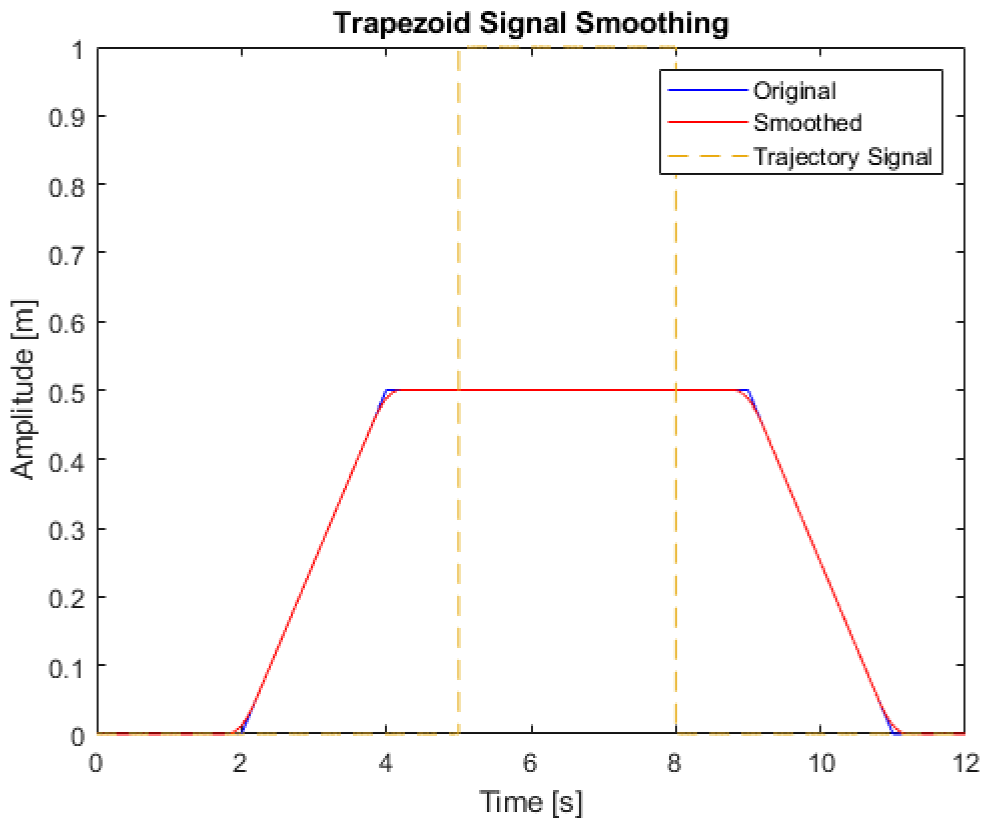
Trajectory Smoothing

Sharp changes in velocity are undesirable for several reasons. In the original velocity signal, four sections can be observed where sudden changes in the direction occur. To resolve this issue, the signal can be smoothed. In this case, a moving average filter was applied. This filter smooths the corners of the graph, and with additional checks it ensures that the velocity remains within the proper range, from 0 to the $v_a = 0.5$. Also this filter does not introduce oscillatory or vibration-like characteristics into the signal unlike a low-pass filter.

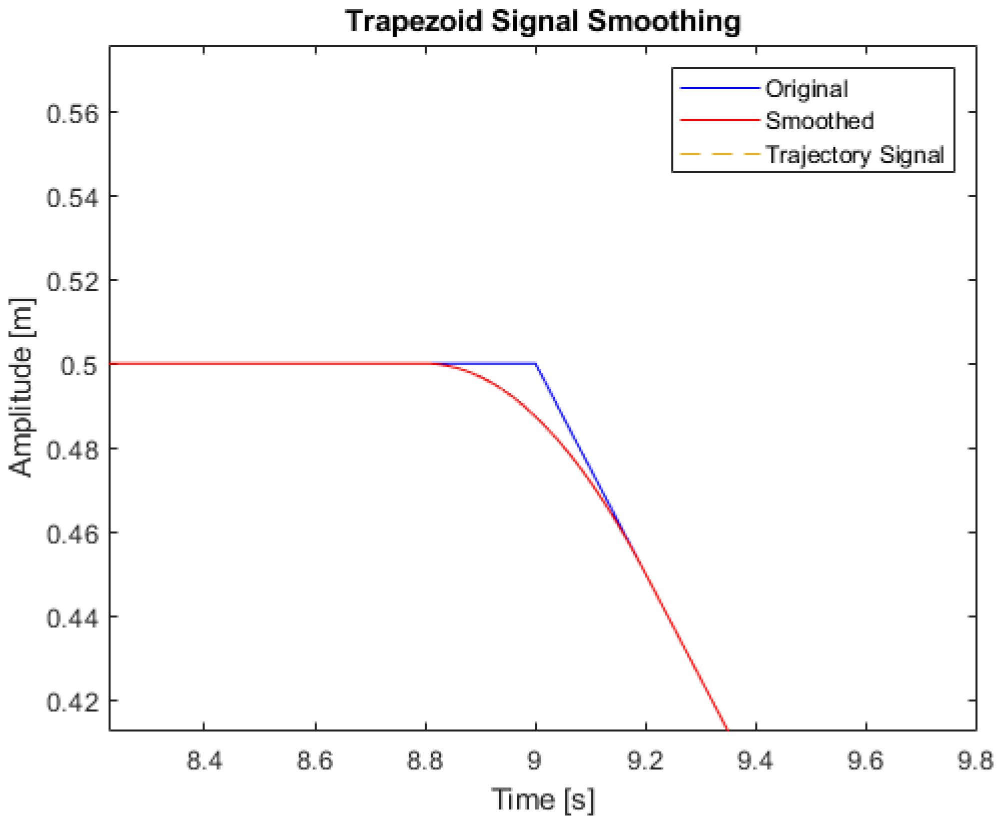
The signal was exported to the MATLAB workspace, where smoothing was applied. Below is the code used:

```
% Define smoothing window size
windowSize = 401; % should be odd and small enough to preserve shape
% Ensure y is a column vector
y = y(:);
% Find min and max to clamp the values
y_min = min(y);
y_max = max(y);
% Create a normalized moving average filter
w = ones(windowSize, 1) / windowSize;
% Pad the signal to avoid edge effects
padSize = floor(windowSize / 2);
y_padded = [y(1)*ones(padSize,1); y; y(end)*ones(padSize,1)];
% Apply convolution (same size as original signal)
y_smoothed = conv(y_padded, w, 'valid');
% Clamp values to original signal bounds
y_smoothed = max(min(y_smoothed, y_max), y_min);
% Plot original and smoothed signals
figure(2)
plot(t,y, 'b-', 'DisplayName', 'Original');
hold on;
plot(t,y_smoothed, 'r-', 'DisplayName', 'Smoothed');
hold on;
plot(t,y_t, '--', 'DisplayName', 'Trajectory Signal')
legend;
xlabel('Time [s]');
ylabel('Amplitude [m]');
title('Trapezoid Signal Smoothing');
```

The figures below show the original and smoothed signal, as well as the trajectory signal indicating when the arc movement occurs. In the second figure the effects of smoothing are clearly visible.



Trapezoid Signal Smoothing
Original
Smoothed
Trajectory Signal
Amplitude [m]
Time [s]
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
2
4
6
8
10
12



Trapezoid Signal Smoothing
Original
Smoothed
Trajectory Signal
Amplitude [m]
Time [s]
0.42
0.44
0.46
0.48
0.5
0.52
0.54
0.56
8.4
8.6
8.8
9
9.2
9.4
9.6
9.8

Inverse Kinematics

Equations

Beta dot

$$\dot{\beta} = \frac{V_a}{l_3}\, tan(\varphi - \beta)$$

Phi dot

$$\dot{\varphi} = \frac{\sqrt{V_a^2 + l_3^2 \dot{\beta}^2}}{R}$$

In the model, Phi is also multiplied by Trajectory Signal.

Alpha 1 dot

$$\dot{\alpha}_1 = \frac{V_a cos(\beta) + l_1 \dot{\beta}\, cos(\beta)}{r_1 cos(\beta)}$$

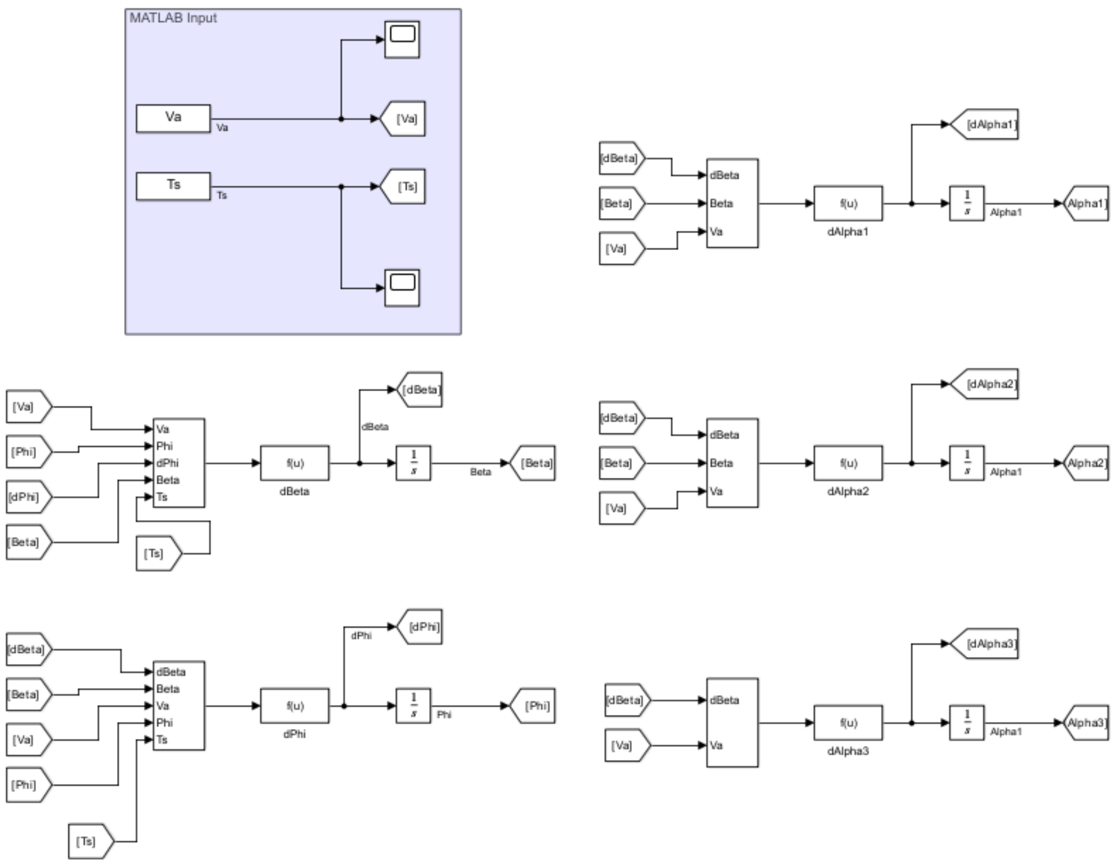
Alpha 2 dot

$$\dot{\alpha}_2 = \frac{V_a cos(\beta) - l_1 \dot{\beta}\, cos(\beta)}{r_2 cos(\beta)}$$

Alpha 3 dot

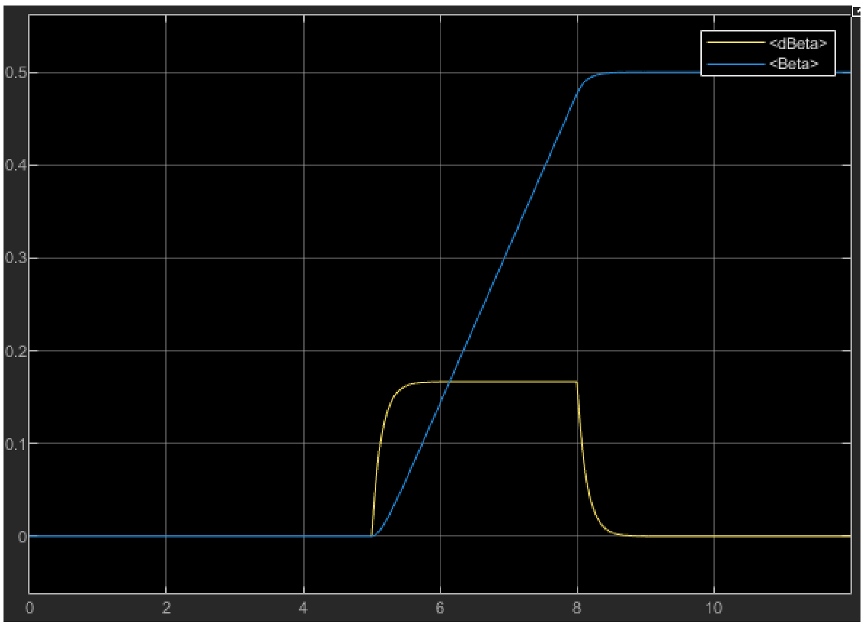
$$\dot{\alpha}_3 = \frac{\sqrt{l_5^2 \dot{\beta}^2 + V_a^2}}{r_3}$$

Simulink model

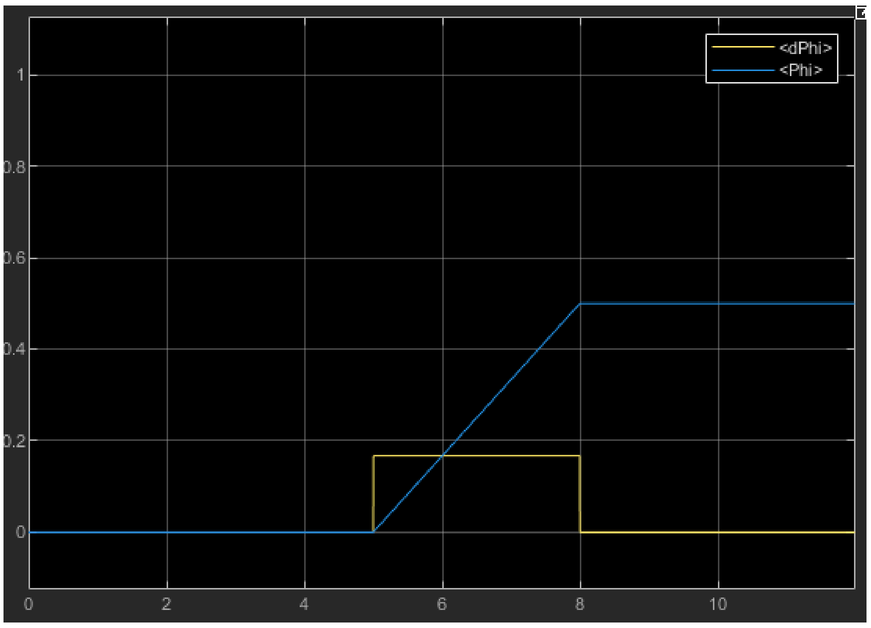


Results

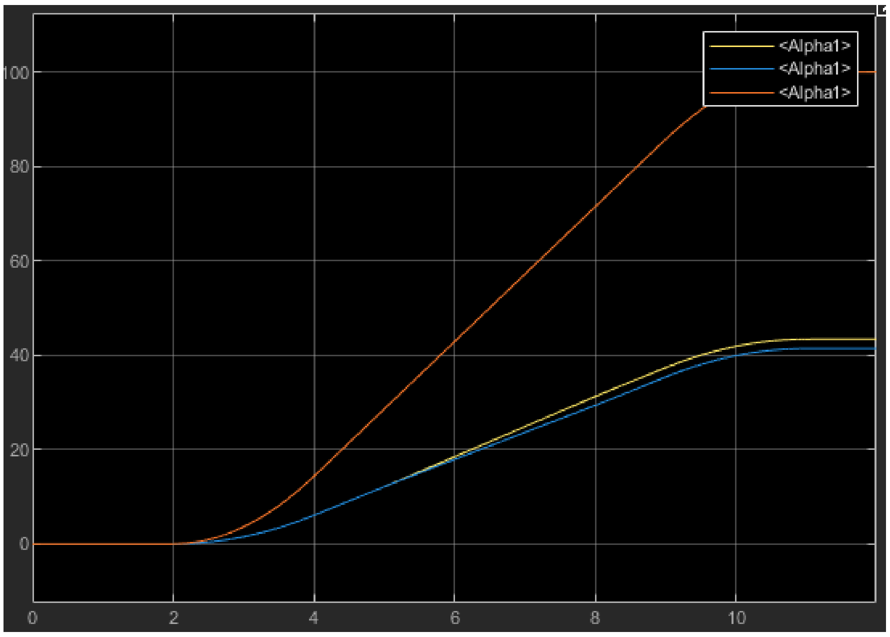
Beta comparison

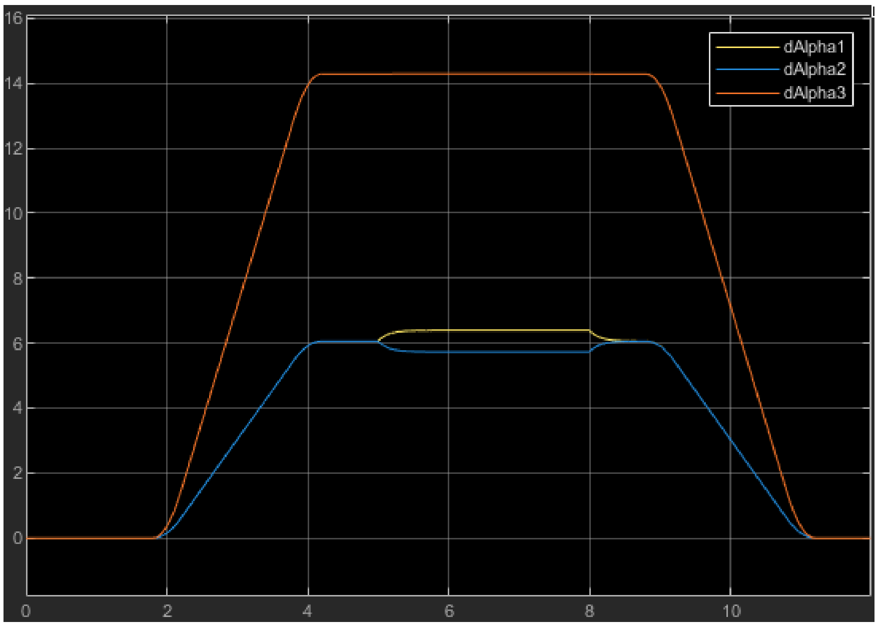


Phi comparison

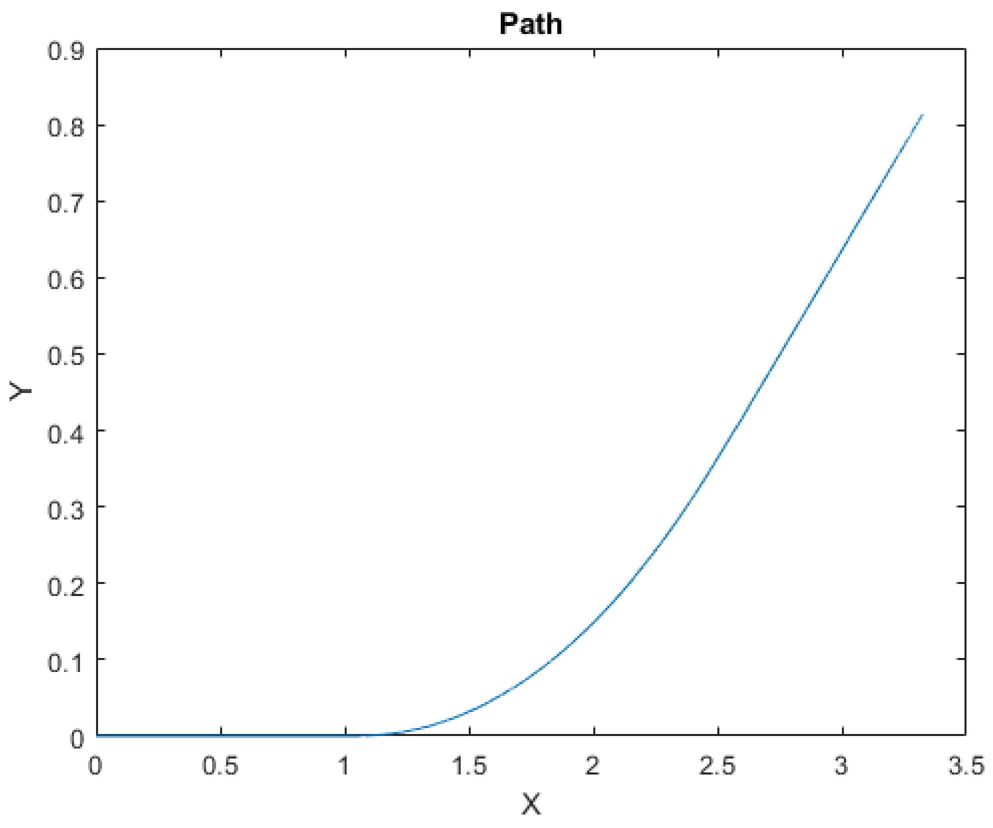


Alpha comparison





Path


Path
Y
X