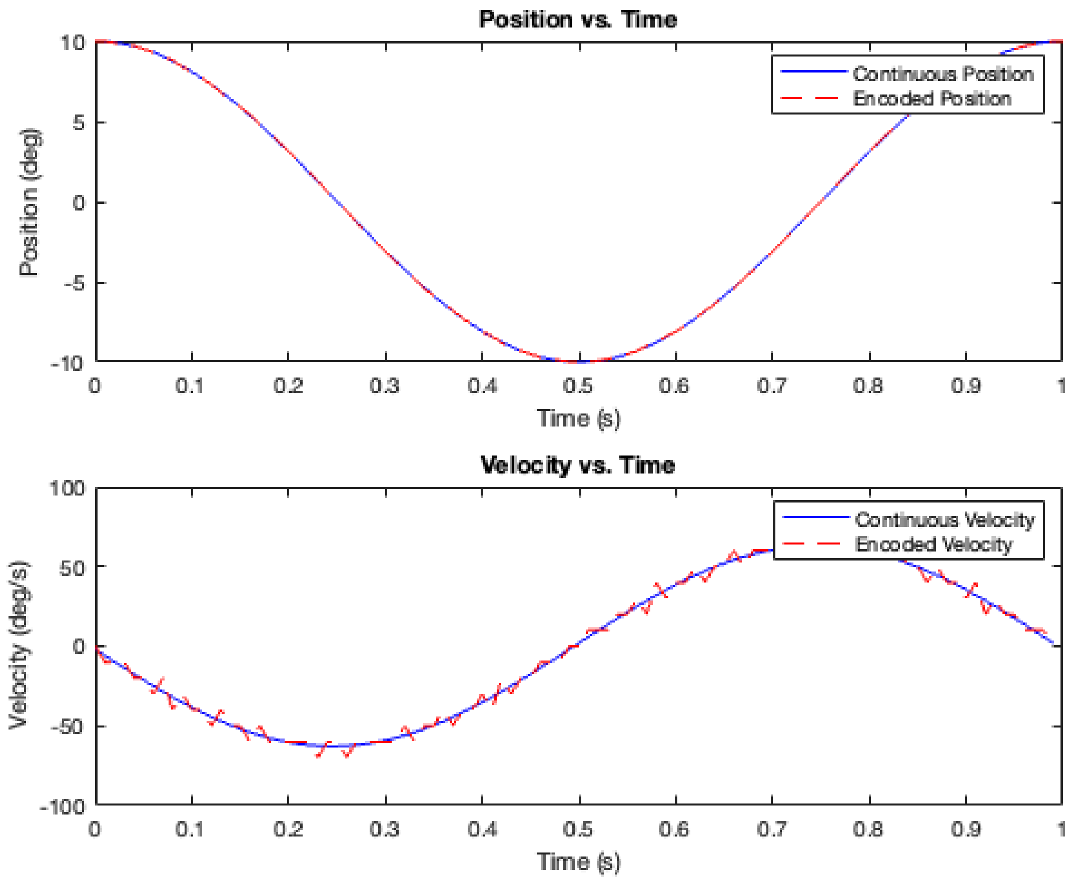
```
clc, clear, close all;
```

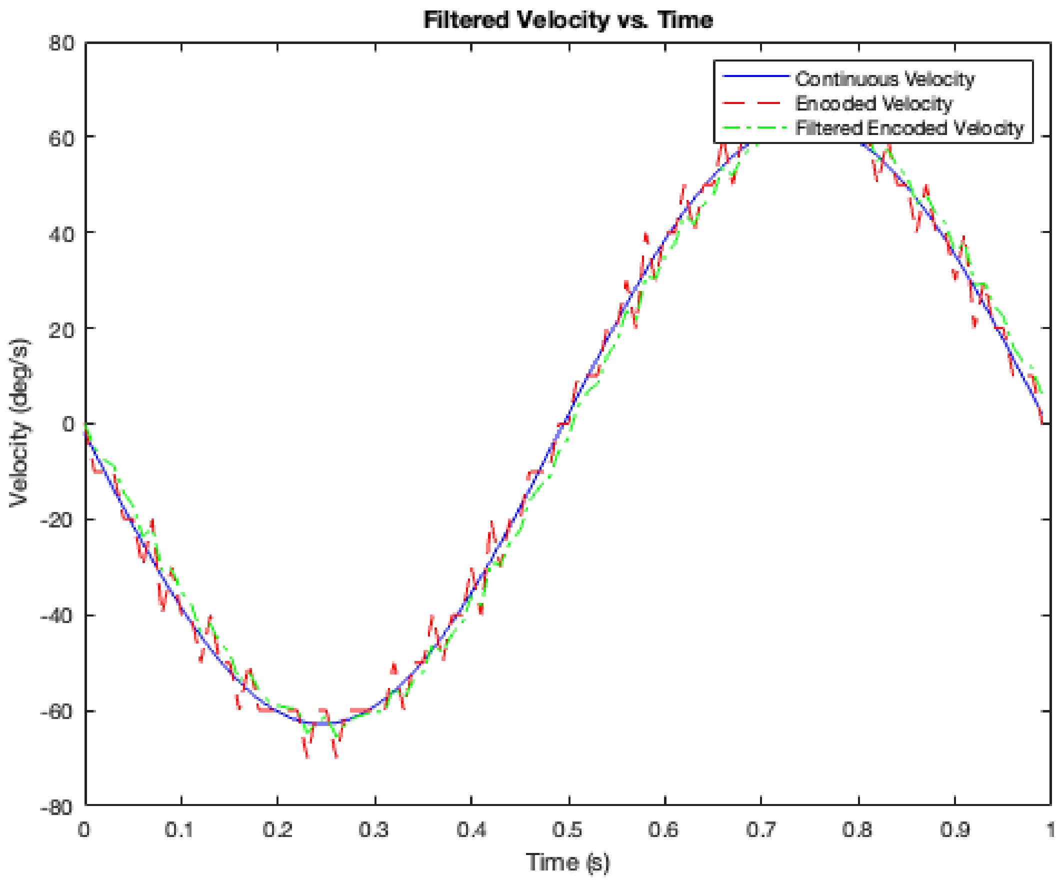
Question 3

```
% Part a: Plot pos vs. time and enc_pos vs. time
t = 0:0.01:1; % Time array with 0.01 sample time
pos = 10*cos(2*pi*t); % Continuous angular position (deg) vs. time
enc_pos = round(pos,1); % Simulated encoder reading with resolution of 0.1

figure();
subplot(2,1,1); % First subplot
plot(t, pos, 'b-', t, enc_pos, 'r--');
legend('Continuous Position', 'Encoded Position');
xlabel('Time (s)');
ylabel('Position (deg)');
title('Position vs. Time');

% Part b: Compute and plot the numerical derivative for velocity
delta_t = 0.01; % Time step
vel = diff(pos) / delta_t; % Numerical derivative of continuous position
enc_vel = diff(enc_pos) / delta_t; % Numerical derivative of encoded position

subplot(2,1,2); % Second subplot for velocity
plot(t(1:end-1), vel, 'b-', t(1:end-1), enc_vel, 'r--');
legend('Continuous Velocity', 'Encoded Velocity');
xlabel('Time (s)');
ylabel('Velocity (deg/s)');
title('Velocity vs. Time');
comment = ['The effect of quantization noise becomes more pronounced in the
 velocity estimation, ' ...
           'introducing significant noise in the derivative of the quantized
 signal.'];

% Part c: Filtering to reduce quantization noise
% Example using a simple low-pass IIR filter.
alpha = 0.5; % Filter weight, adjust this based on experimentation
filtered_enc_vel = IIR_WA(enc_vel, alpha); % Basic IIR filter

figure();
plot(t(1:end-1), vel, 'b-', t(1:end-1), enc_vel, 'r--', t(1:end-1),
 filtered_enc_vel, 'g-.');
legend('Continuous Velocity', 'Encoded Velocity', 'Filtered Encoded
 Velocity');
xlabel('Time (s)');
ylabel('Velocity (deg/s)');
title('Filtered Velocity vs. Time');
comment2 = ['Applying a filter reduces the noise in the velocity signal
 derived from the encoded position. ' ...
            'The choice of filter weight (alpha) is crucial to balance between
 noise reduction and phase lag. ' ...
            'Filtering before vs. after taking the derivative can affect the
 results; ' ...
```

```
           'it’s generally better to filter the position signal before
 differentiating to smooth out noise.'];

disp(comment);
disp(comment2);
```

```
The effect of quantization noise becomes more pronounced in the velocity
 estimation, introducing significant noise in the derivative of the quantized
 signal.
Applying a filter reduces the noise in the velocity signal derived from the
 encoded position. The choice of filter weight (alpha) is crucial to balance
 between noise reduction and phase lag. Filtering before vs. after taking
 the derivative can affect the results; it’s generally better to filter the
 position signal before differentiating to smooth out noise.
```

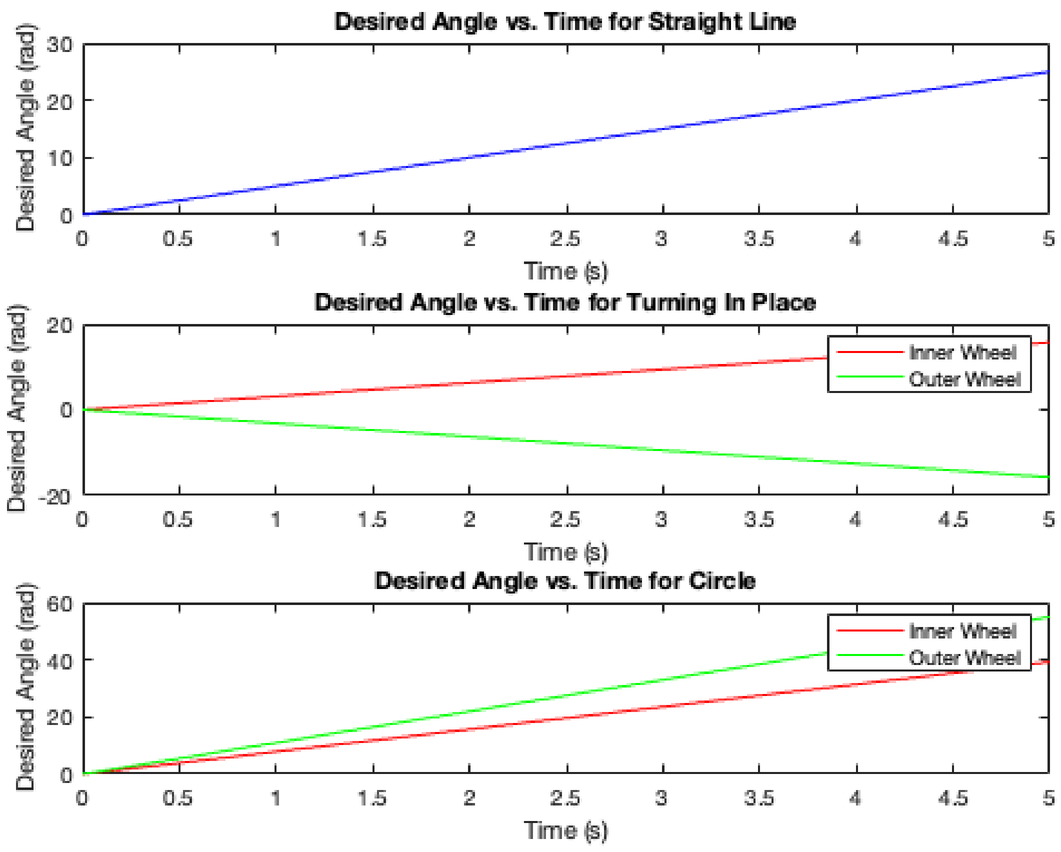




Question 4

```
% Common Parameters
rw = 0.02; % wheel radius in meters
D = 0.20; % distance between wheels in meters
T = 5; % time to complete the maneuver in seconds
dt = 0.01; % time step for simulation

% Part a: Drive in a straight line for 50cm
xf = 0.50; % final distance in meters
theta_f_straight = xf / rw; % final desired wheel angle in radians
omega_d_straight = theta_f_straight / T; % constant angular velocity

% Time array
t_straight = 0:dt:T;
theta_d_straight = omega_d_straight * t_straight; % Desired angle trajectory

% Part b: Turn in place by 180 degrees
phi = pi; % 180 degrees in radians
theta_f_turn = (phi * D) / (2 * rw); % final desired wheel angle in radians
omega_d_turn = theta_f_turn / T; % constant angular velocity
```

```
% Time array
t_turn = 0:dt:T;
theta_d_turn_inner = omega_d_turn * t_turn;
theta_d_turn_outer = -omega_d_turn * t_turn; % Opposite sign for turning in
 place

% Part c: Drive in a circle of radius 50cm for a 90 degree arc
R = 0.50; % radius of the circle in meters
theta_arc = pi/2; % 90 degrees in radians
theta_f_circle = (theta_arc * R) / rw; % final desired wheel angle in radians
 for inner wheel
omega_d_circle = theta_f_circle / T; % constant angular velocity

% Ratio of velocities for outer and inner wheels
velocity_ratio = (R + D) / R;
omega_d_outer = omega_d_circle * velocity_ratio;

% Time array
t_circle = 0:dt:T;
theta_d_circle_inner = omega_d_circle * t_circle;
theta_d_circle_outer = omega_d_outer * t_circle;

% Plotting the desired angles vs. time for all maneuvers
figure();

% Plot for straight line
subplot(3,1,1);
plot(t_straight, theta_d_straight, 'b');
title('Desired Angle vs. Time for Straight Line');
xlabel('Time (s)');
ylabel('Desired Angle (rad)');

% Plot for turning in place
subplot(3,1,2);
plot(t_turn, theta_d_turn_inner, 'r', t_turn, theta_d_turn_outer, 'g');
title('Desired Angle vs. Time for Turning In Place');
xlabel('Time (s)');
ylabel('Desired Angle (rad)');
legend('Inner Wheel', 'Outer Wheel');

% Plot for driving in a circle
subplot(3,1,3);
plot(t_circle, theta_d_circle_inner, 'r', t_circle,
 theta_d_circle_outer, 'g');
title('Desired Angle vs. Time for Circle');
xlabel('Time (s)');
ylabel('Desired Angle (rad)');
legend('Inner Wheel', 'Outer Wheel');
```



Published with MATLAB® R2023a