DIPARTIMENTO
DI INGEGNERIA
DELL'INFORMAZIONE

1222 • 2022
800 ANNI

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

URBAN MOBILITY CHALLENGE

Alessandro Borgherini - 2123456
Riccardo Rettore - 2110512
Thomas Zamprogno - 2121463

Outline

- Introduction & Objectives
- Framework
- Obstacle avoidance procedure
- Trajectory tracking control
  - State error feedback controller
    - ❑ **Linear** control
    - ❑ **Nonlinear** control
  - Output error feedback controller
    - ❑ **B-point** control
    - ❑ **Further derivatives** control
- Parking
  - ❑ **Cartesian regulation**
  - ❑ **Posture regulation**

Introduction: the "urban mobility challenge"

Problem: An autonomous guided vehicle (AGV) has to travel along a closed circuit avoiding the other AGVs that are parked along the path; at the completion of one lap, the AGV has to get off the circuit and park at the box

PROJECT OBJECTIVES:

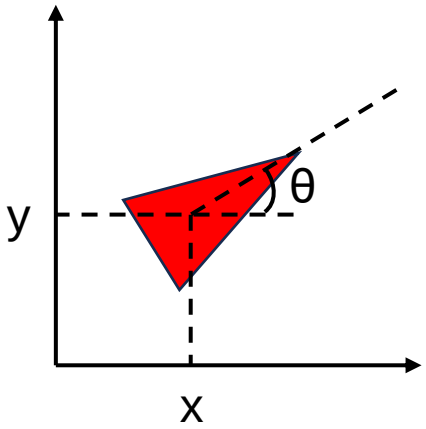
1) Track a closed-loop trajectory
2) Avoid obstacles on the trajectory
3) After performing one lap, reach an empty parking lot

PROPOSED SOLUTIONS:

1) State & output error feedback
2) Sensor for obstacle detection and avoidance
3) Cartesian & Posture regulation

Framework: unicycle

The AGV used in the project is a unicycle robot, a planar vehicle with a single orientable wheel



The unicyle pose is described by its state

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \in \mathbb{R}^2 \times \mathbb{S}^1$$

And its control inputs are the linear velocity v and the angular velocity ω.
The relation between the cartesian velocities and the control inputs is given by

$$\begin{aligned} \dot{x} &= v \cos\theta \\ \dot{y} &= v \sin\theta \\ \dot{\theta} &= \omega \end{aligned}$$

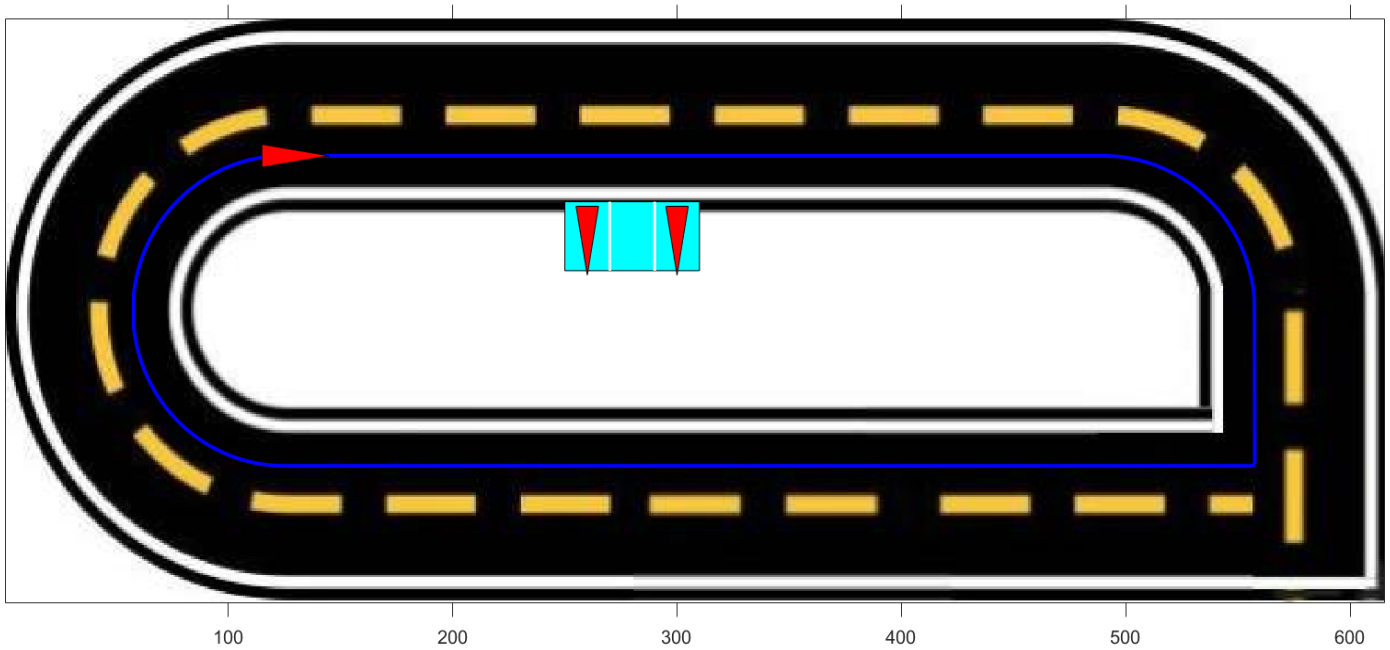
And we obtain the kinematics model

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Framework: trajectory

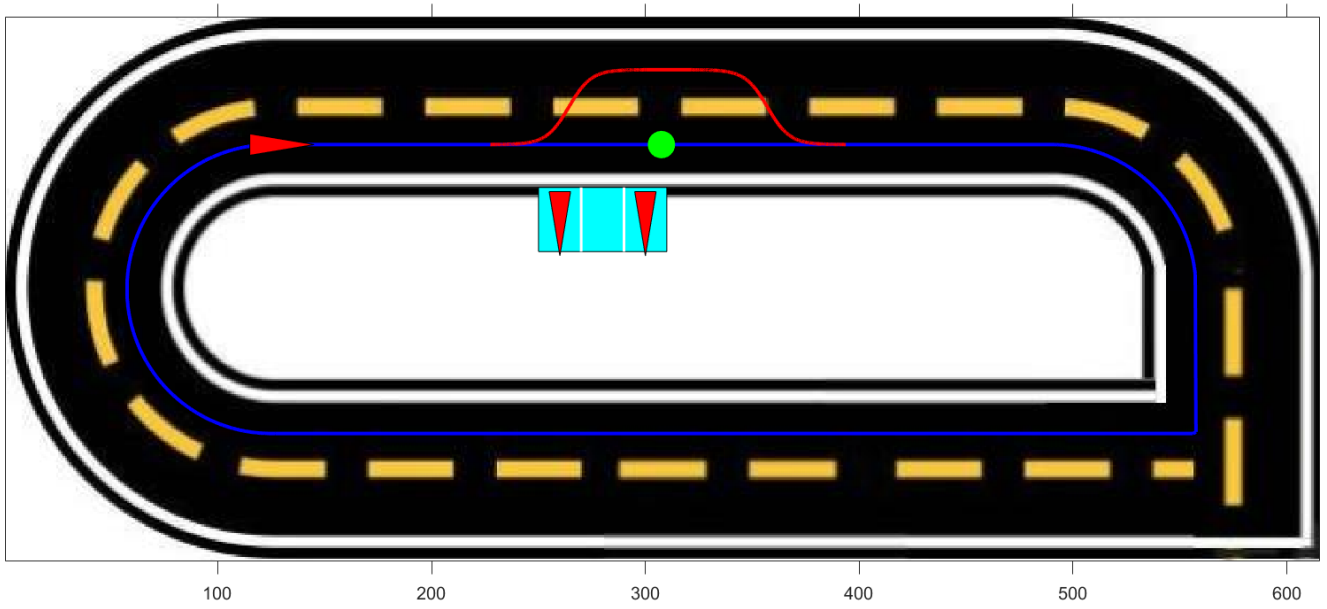
- To design the desired trajectory, a series of waypoints was defined. These waypoints were then interpolated using cubic spline interpolation to ensure a continuous path

- To test controllers' functionalities on a non-continuous path, a sharp square-angle turn is added to the path



Obstacle avoidance

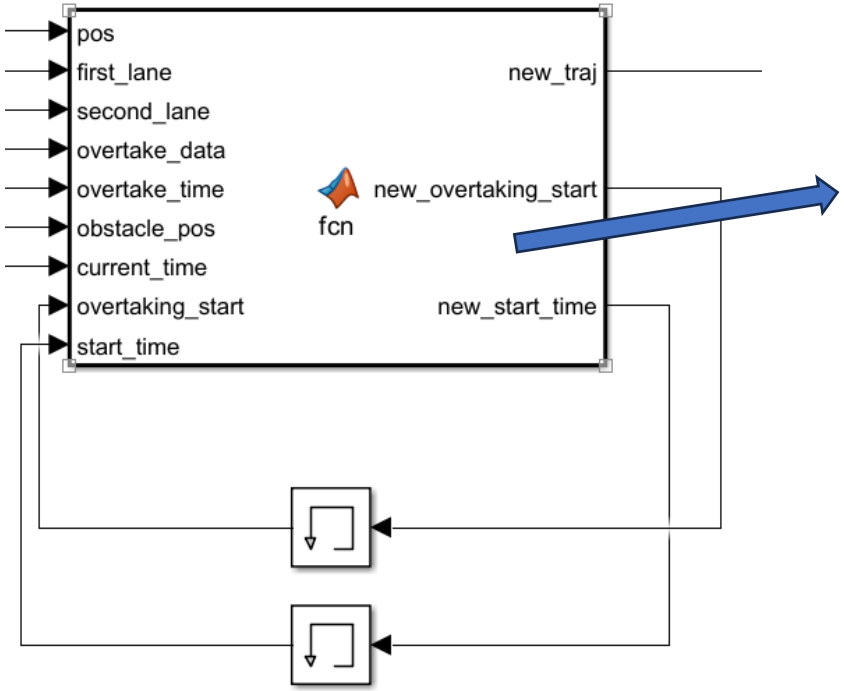
- The unicyle after detecting the obstacle progressively reaches the second lane, after overtaking the obstacle it progressively returns to the first lane



- To ensure a smooth and continuous trajectory during the changing lane procedure a sigmoid function has been used

```
function overtake_data = createOvertake(length)
    t = linspace(-7, 7, 3000); % Adjust range for smoothness
    x1 = 1 ./ (1 + exp(-t)); %Positive sigmoid
    x2 = exp(-t) ./ (1 + exp(-t)); %Negative sigmoid
    overtake_data = [0, x1, x1(end)*ones(1,length), x2];
end
```

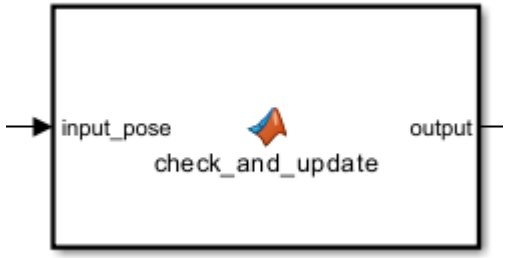
Obstacle avoidance



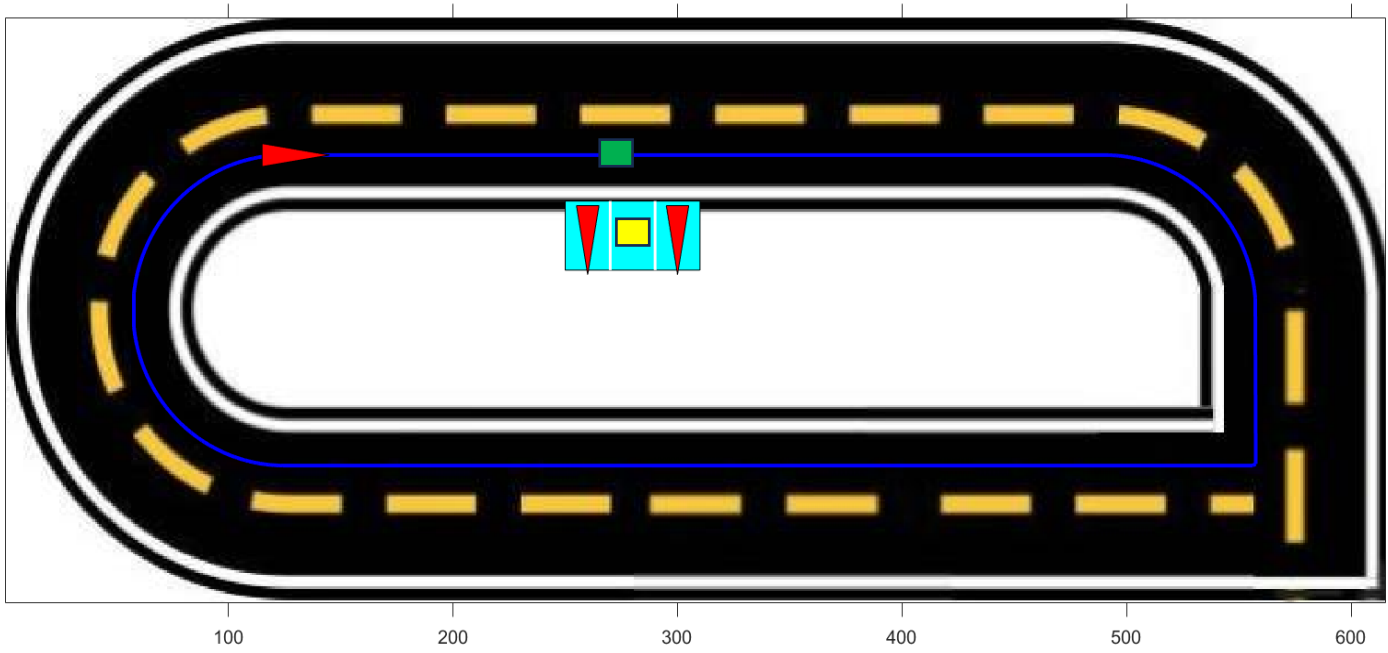

```
function [new_traj, new_overtaking_start, new_start_time]  = fcn(pos,first_lane, second_lane, overtake_data, overta
detection_margin_x = 80; % Detection range in x-direction
detection_margin_y = 50; % Detection range in y-direction
% Check if obstacle is within detection range and ahead of the obstacle
if ~overtaking_start
    if (abs(pos(1) - obstacle_pos(1)) < detection_margin_x && abs(pos(2) - obstacle_pos(2)) < detection_margin_y)
        new_overtaking_start = 1;
        new_start_time = current_time;
    else
        new_overtaking_start = overtaking_start;
        new_start_time = start_time;
    end
    idx = 1;
else
    idx = find(abs(overtake_time - (current_time - start_time)) <= 1e-3, 1);
    if ~isempty(idx)
        new_overtaking_start = overtaking_start;
    else
        new_overtaking_start = 0;
        idx = 1;
    end
    new_start_time = start_time;
end
new_traj = first_lane + (second_lane-first_lane)*overtake_data(idx);
```

Parking problem

```
% Initialize persistent variables
if isempty(target_y)
    target_y = 60; % Initial value
    target_x=270;
    target_thet=0;
end
if isempty(has_updated)
    has_updated = false; % Initialize the flag
end
% Extract x, y,theta from input_pose
x = input_pose(1);
y = input_pose(2);
theta=input_pose(3);
tolerance = 2;

% Check if the pose is near the target position
if abs(x - target_x) <= tolerance && abs(y - target_y)<= tolerance
    && ~has_updated  && abs(theta - target_thet) <= 0.2
    % Update
    target_y = 90;
    target_x=280;
    target_thet=pi/2;
    has_updated = true; % Set the flag to prevent future updates
end

% Output the updated or unchanged target coordinates
output = [target_x; target_y;target_thet];
```



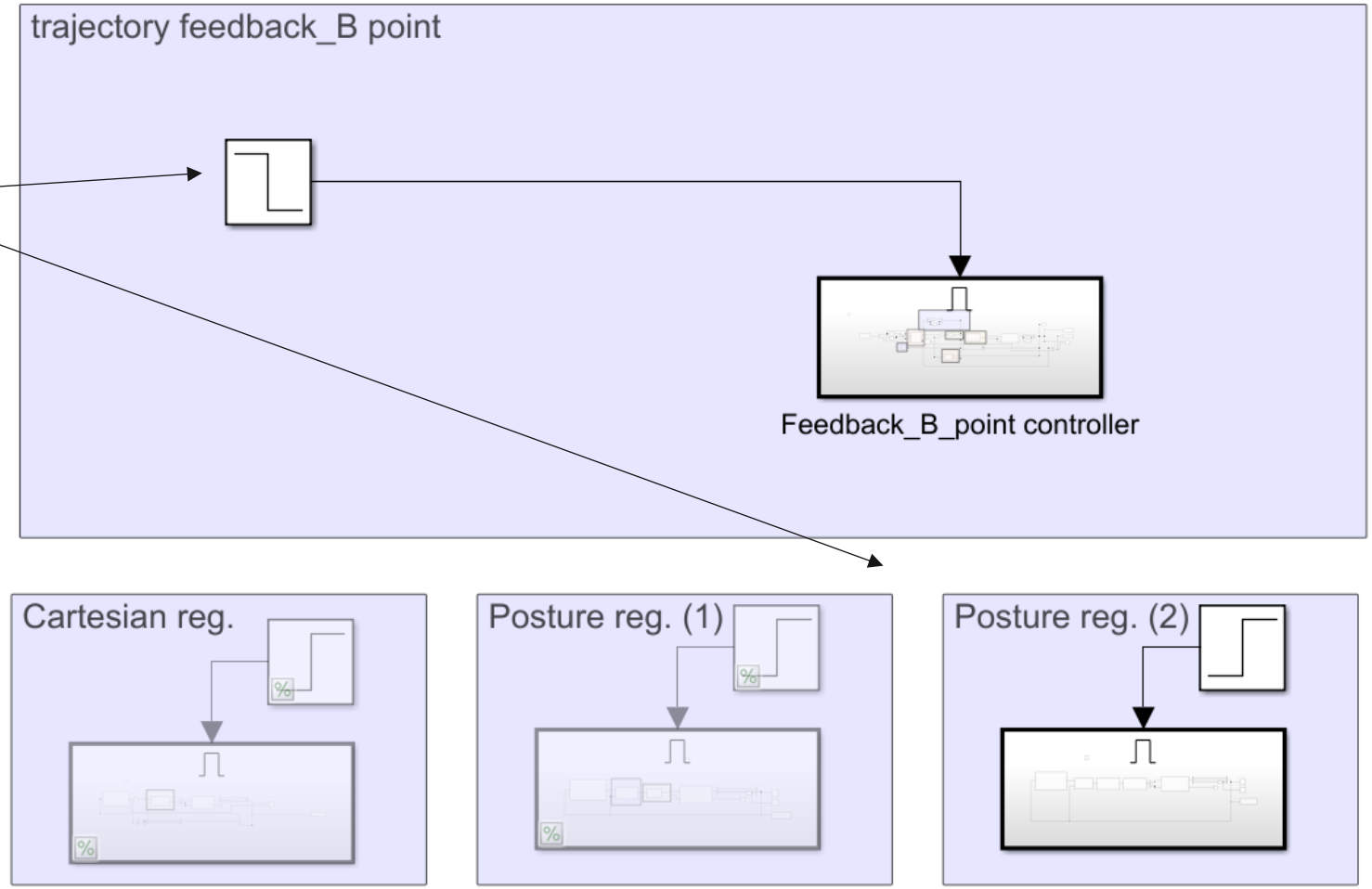
- Two waypoints for the parking
- Simulate a real parking condition



Merger between the tracking trajectory and the parking

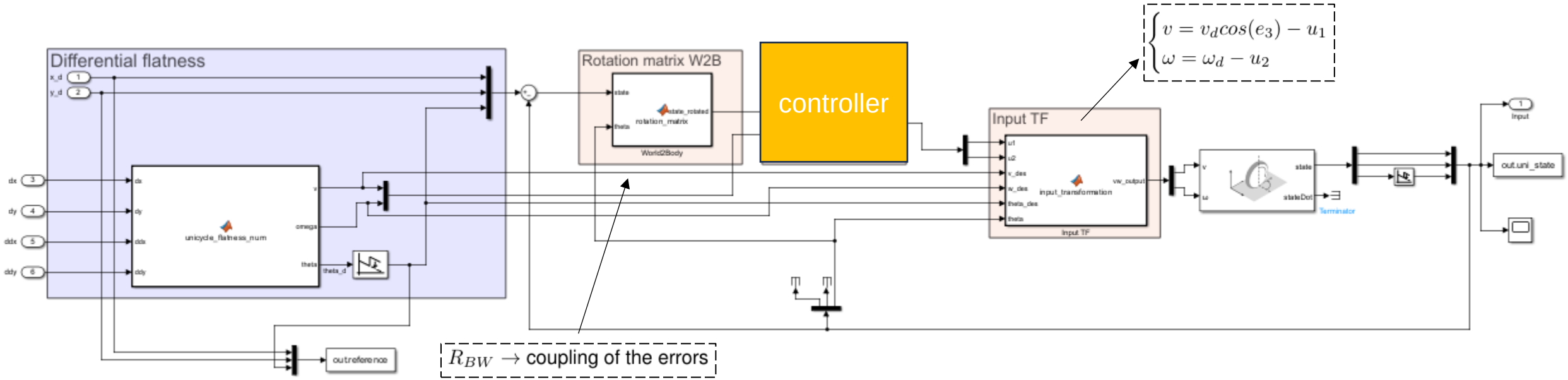
Enable to activate individual phases

Automatic switching



TRAJECTORY TRACKING - STATE ERROR FEEDBACK

Design



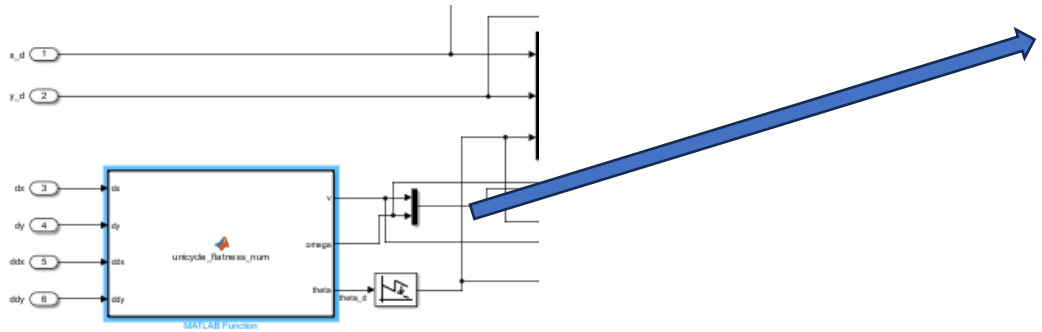
(x_d, y_d) twice differentiable and feasible for the unicycle

! Constraints on the trajectory !

The dynamic of the errors $(\dot{e}_1, \dot{e}_2, \dot{e}_3)$ is:

- Nonlinear
- Time-varying
- Coupled

Differential Flatness



The state variables x, y, and θ can be computed from the flat outputs $x_d(t)$ and $y_d(t)$ as follows:

$$x = x_d(t), \quad y = y_d(t), \quad \theta = \arctan 2\left(\dot{y}_d(t), \dot{x}_d(t)\right) + k\pi$$

The control inputs, linear velocity v and angular velocity ω, are derived from the flat outputs as:

$$v = \sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)}$$

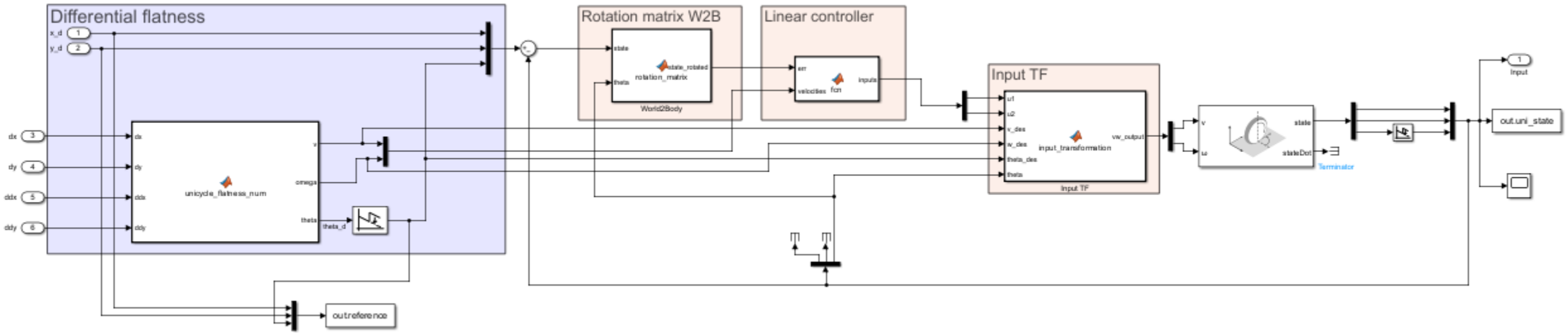
$$\omega = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)}$$

→ v_d must be different from zero

```
function [v, omega, theta] = unicycle_flatness_num(dx, dy, ddx, ddy)
    % Compute linear velocity safely
    velocity_magnitude = sqrt(dx^2 + dy^2);
    if velocity_magnitude < 1e-4
        v = sign(velocity_magnitude)*1e-4;
    else
        v = velocity_magnitude;
    end
    % Compute heading angle
    theta = atan2(dy, dx);

    % Compute angular velocity safely
    denominator = v^2;
    if denominator < 1e-4
        omega = (dx * ddy - dy * ddx) / 1e-4;
    else
        omega = (dx * ddy - dy * ddx) / denominator;
    end
end
```

avoiding division by zero

Linear Control

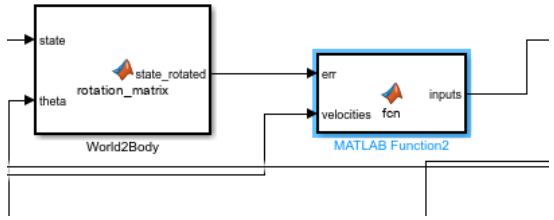


Idea:
Linearization around
the desired trajectory $\mathbf{e} = [0, 0, 0]^T$

Σ_L asymptotically stable $\longrightarrow$ Σ_{NL} locally asymptotically stable

Controller design

$$\begin{cases} u_1 = -k_1 e_1 \\ u_2 = -k_2 e_2 - k_3 e_3 \end{cases}$$



Gain Computation The feedback gains for the system are computed as follows:

$$k_1 = k_3 = 2\xi a \quad k_2 = \frac{a^2 - \omega_d^2}{v_d}$$

TI TV → persistent trajectories

where:

- ξ: Damping ratio, where $0 < \xi < 1$
- a: Natural frequency parameter, where $a > 0$

no guarantee of asymptotic stability, only for rectilinear and circular motion

Eigs allocation:

$$\lambda_1 = -2\xi a, \quad \lambda_2 = -\xi a + \sqrt{(\xi a)^2 - a^2}, \quad \lambda_3 = -\xi a - \sqrt{(\xi a)^2 - a^2}$$

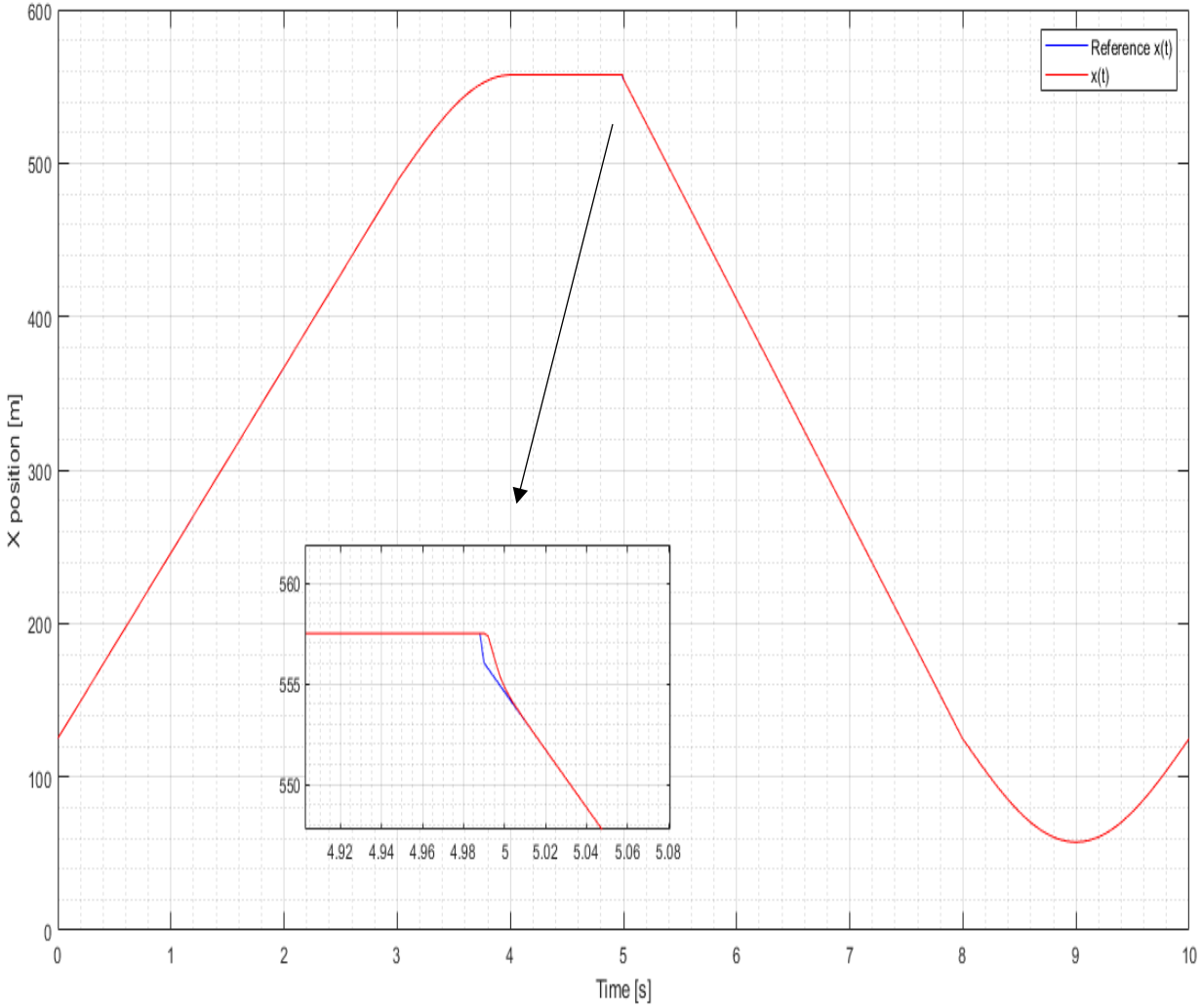
```
function inputs = fcn(err , velocities)
    d = 1 / (sqrt(2));
    a = 200;
    vd = velocities(1);
    wd = velocities(2);

    % Gains
    k1 = 2 * d * a;
    k2 = (a^2 - wd^2) / vd;
    k3 = k1;
```

saturation

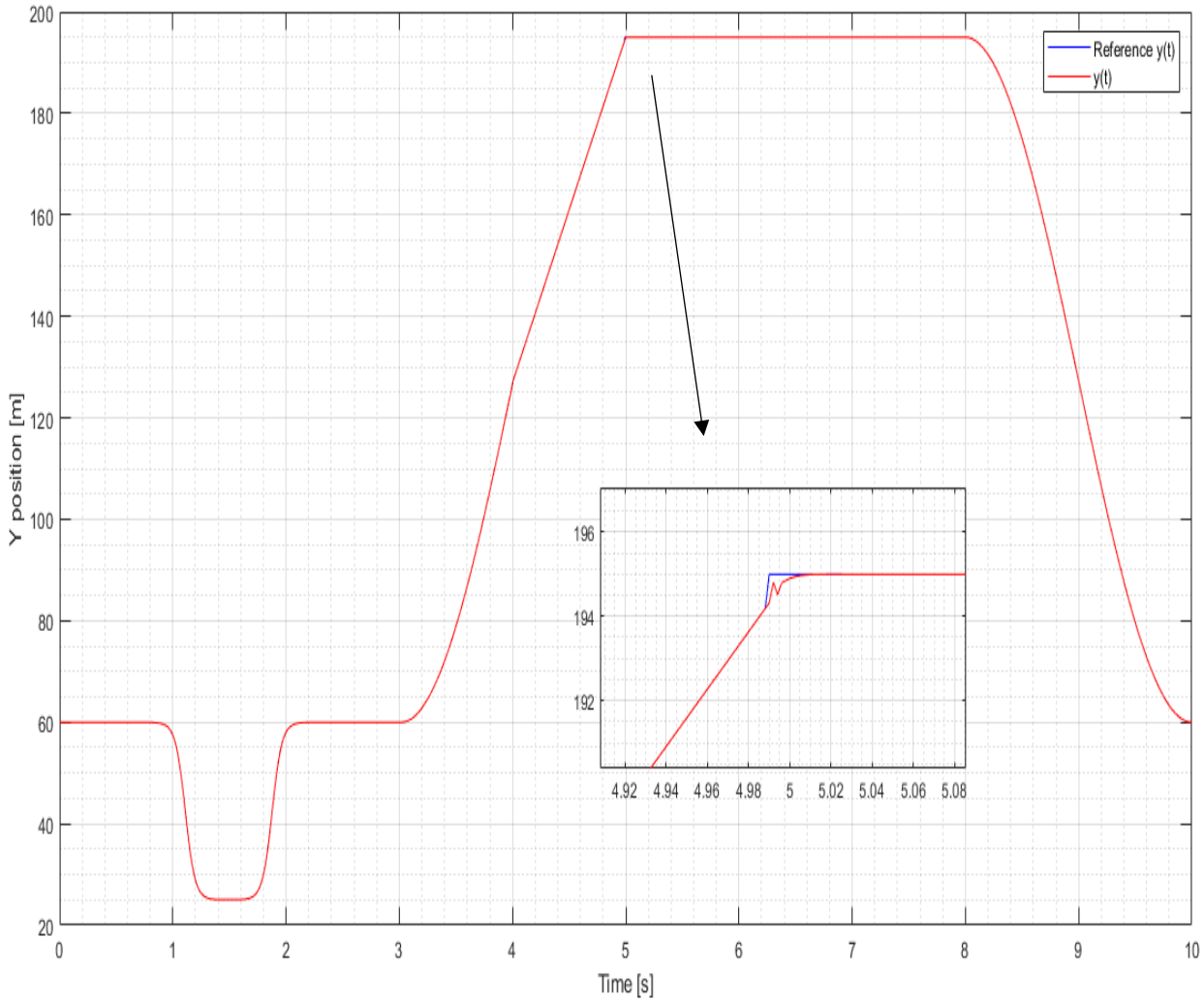
```
    % Clamp k2 to avoid large values
    max_k2 = 1000;
    k2 = min(k2, max_k2);

    % Gain matrix
    k = [-k1, 0, 0;
         0, -k2, -k3];

    % Compute inputs
    inputs = k * err;
end
```

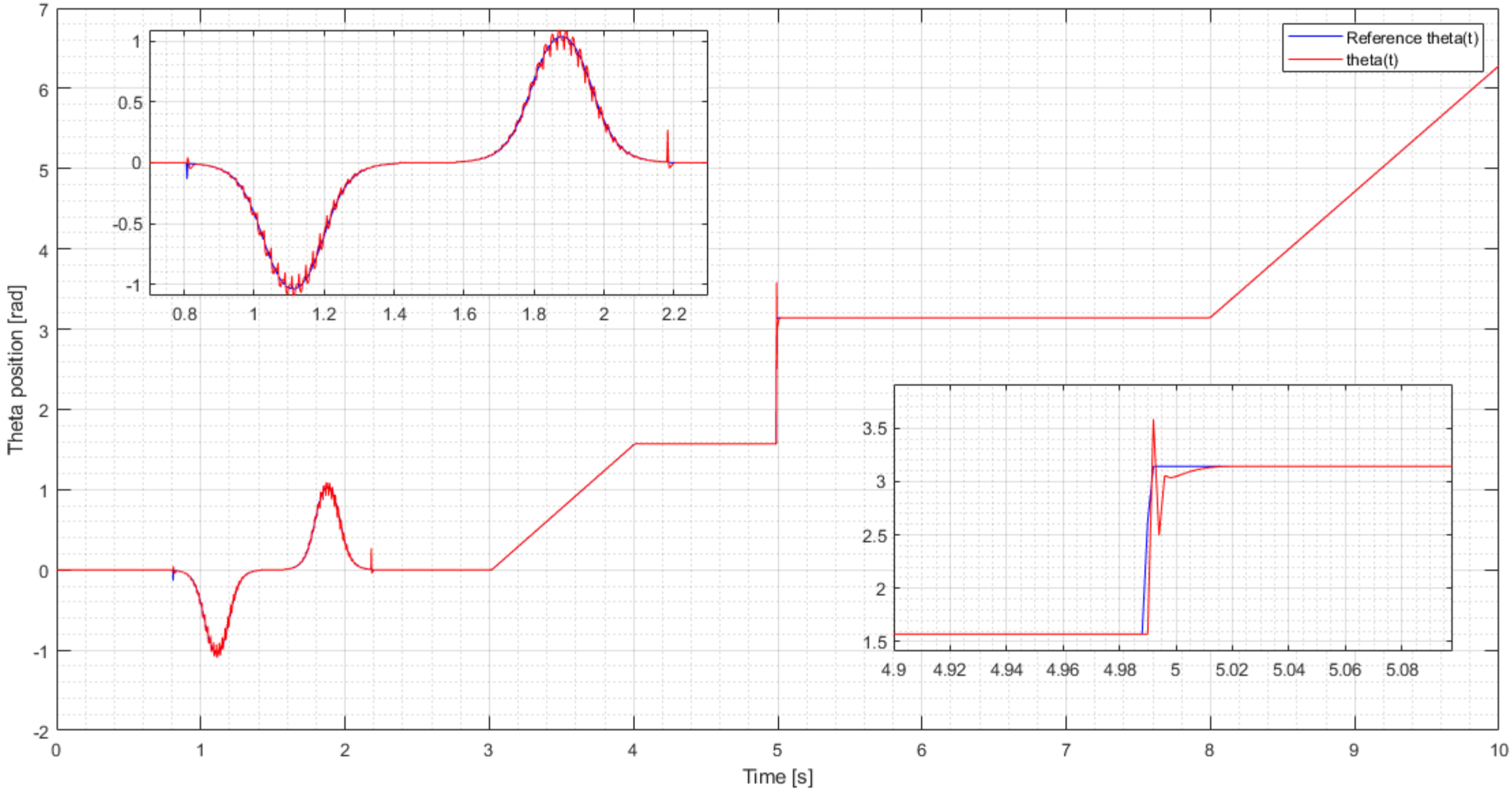
Plots

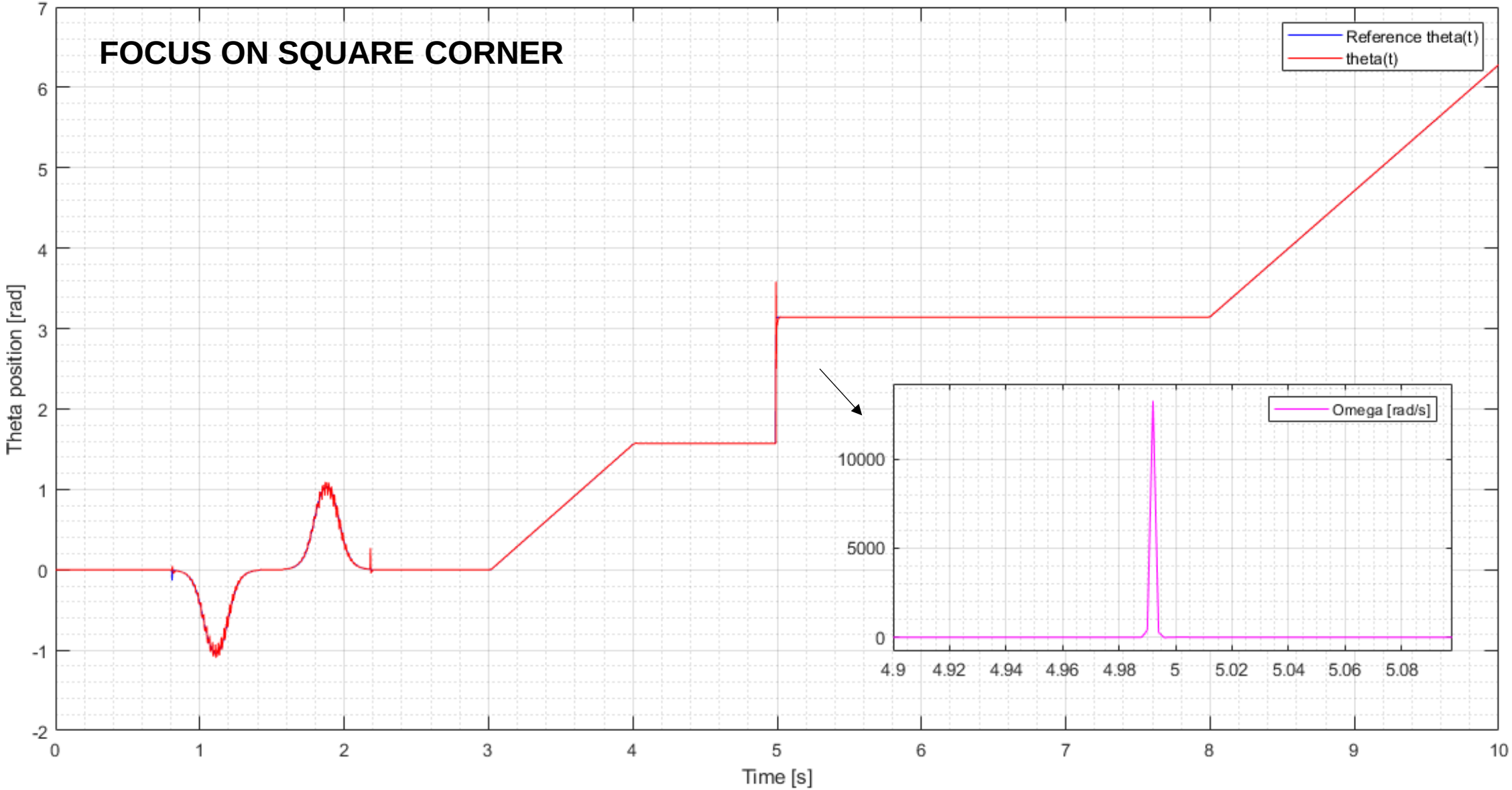
X TRACKING

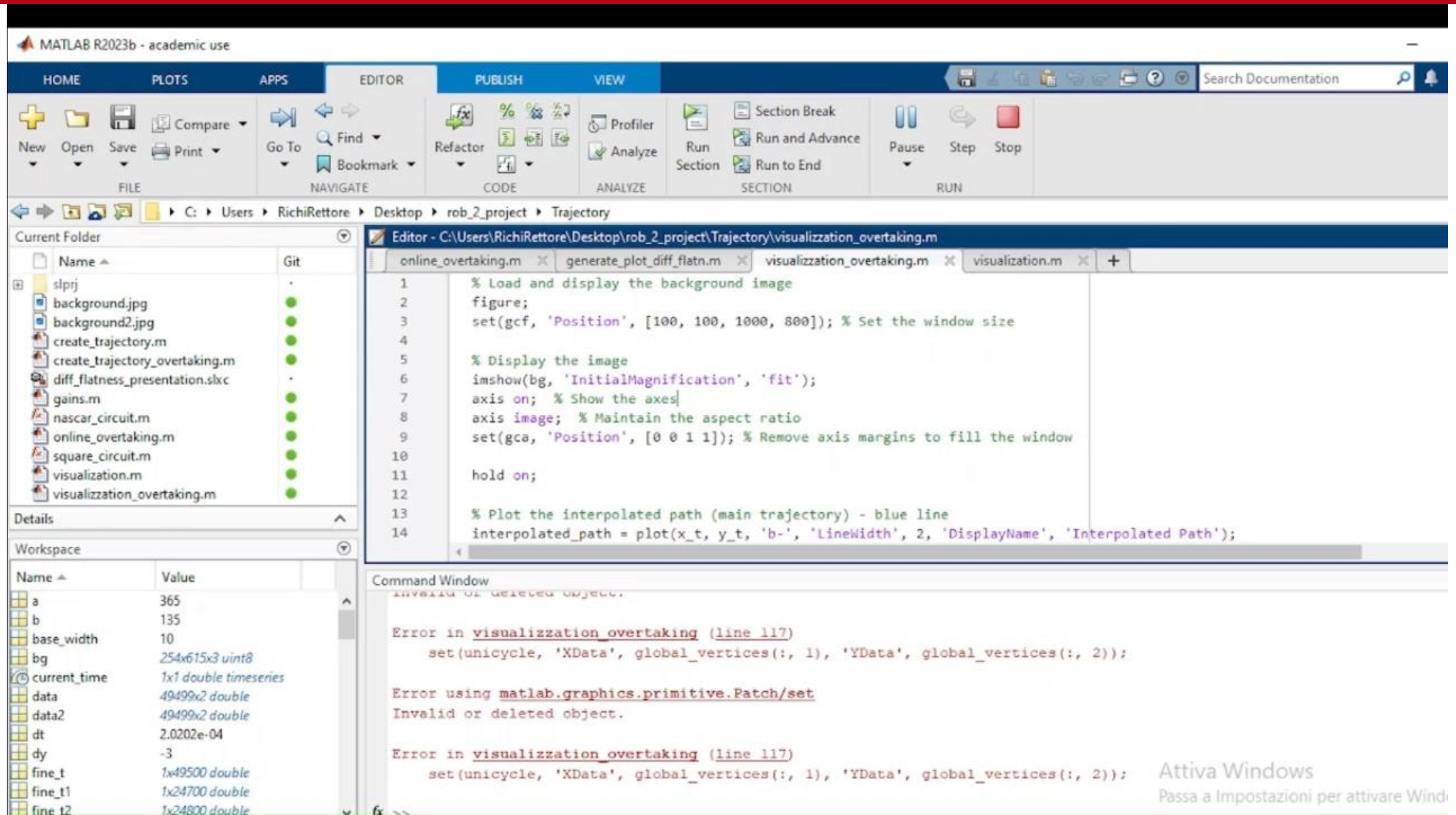


Y TRACKING

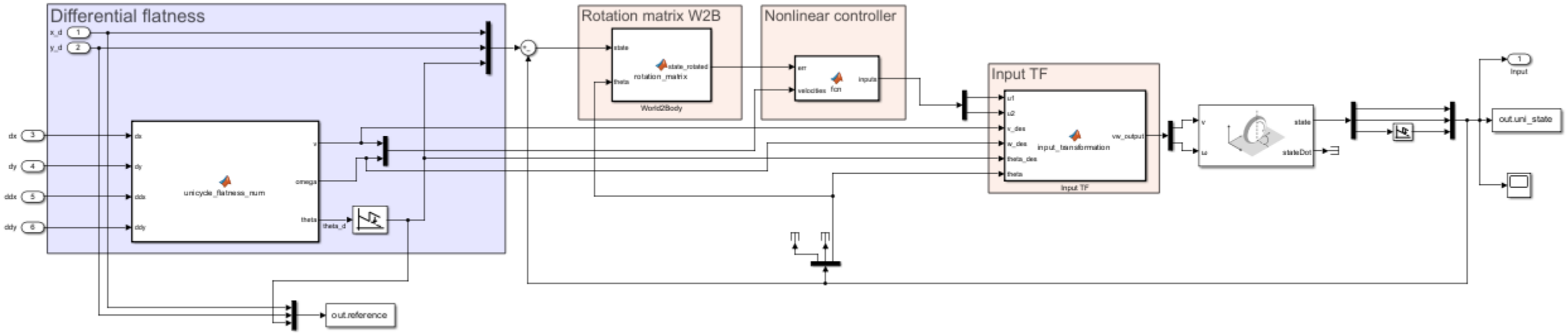



Reference theta(t)
theta(t)
Theta position [rad]
Time [s]


FOCUS ON SQUARE CORNER
Reference theta(t)
theta(t)
Omega [rad/s]
Theta position [rad]
Time [s]
7
6
5
4
3
2
1
0
-1
-2
0
1
2
3
4
5
6
7
8
9
10
10000
5000
0
4.9
4.92
4.94
4.96
4.98
5
5.02
5.04
5.06
5.08


MATLAB R2023b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Pause
Step
Stop
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
C: ▸ Users ▸ RichiRettore ▸ Desktop ▸ rob_2_project ▸ Trajectory
Current Folder
Name
Git
slprj
background.jpg
background2.jpg
create_trajectory.m
create_trajectory_overtaking.m
diff_flatness_presentation.slxc
gains.m
nascar_circuit.m
online_overtaking.m
square_circuit.m
visualization.m
visualizzation_overtaking.m
Details
Workspace
Name
Value
a 365
b 135
base_width 10
bg 254x615x3 uint8
current_time 1x1 double timeseries
data 49499x2 double
data2 49499x2 double
dt 2.0202e-04
dy -3
fine_t 1x49500 double
fine_t1 1x24700 double
fine_t2 1x24800 double
Editor - C:\Users\RichiRettore\Desktop\rob_2_project\Trajectory\visualizzation_overtaking.m
online_overtaking.m
generate_plot_diff_flatn.m
visualizzation_overtaking.m
visualization.m
1 % Load and display the background image
2 figure;
3 set(gcf, 'Position', [100, 100, 1000, 800]); % Set the window size
4
5 % Display the image
6 imshow(bg, 'InitialMagnification', 'fit');
7 axis on; % Show the axes
8 axis image; % Maintain the aspect ratio
9 set(gca, 'Position', [0 0 1 1]); % Remove axis margins to fill the window
10
11 hold on;
12
13 % Plot the interpolated path (main trajectory) - blue line
14 interpolated_path = plot(x_t, y_t, 'b-', 'LineWidth', 2, 'DisplayName', 'Interpolated Path');
Command Window
Error in visualizzation_overtaking (line 117)
set(unicycle, 'XData', global_vertices(:, 1), 'YData', global_vertices(:, 2));
Error using matlab.graphics.primitive.Patch/set
Invalid or deleted object.
Error in visualizzation_overtaking (line 117)
set(unicycle, 'XData', global_vertices(:, 1), 'YData', global_vertices(:, 2));
Attiva Windows
Passa a Impostazioni per attivare Wind

Nonlinear Control

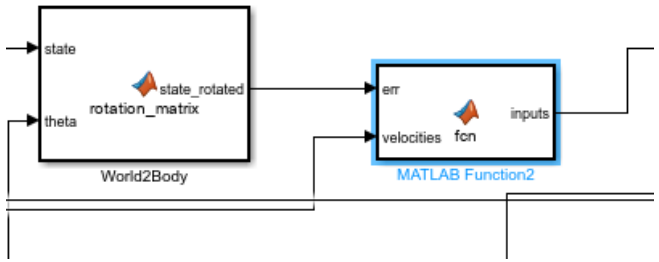


dynamic of the errors is in mixed form

Controller design

$$\begin{cases} u_1 = -k_1 e_1 \\ u_2 = -k_2 \left[v_d \frac{\sin e_3}{e_3} \right] e_2 - k_3 e_3 \end{cases}$$

nonlinear term



Nonlinear Gain Computation The nonlinear feedback gains are computed as:

$$k_1 = k_3 = 2\xi\sqrt{bv_d + \omega_d^2} \qquad k_2 = b$$

TV TI

no more singularity in vd=0

where:

- ξ: Damping ratio, where $0 < \xi < 1$
- b: Nonlinear system parameter, where $b > 0$

resembles k2 of the linear controller

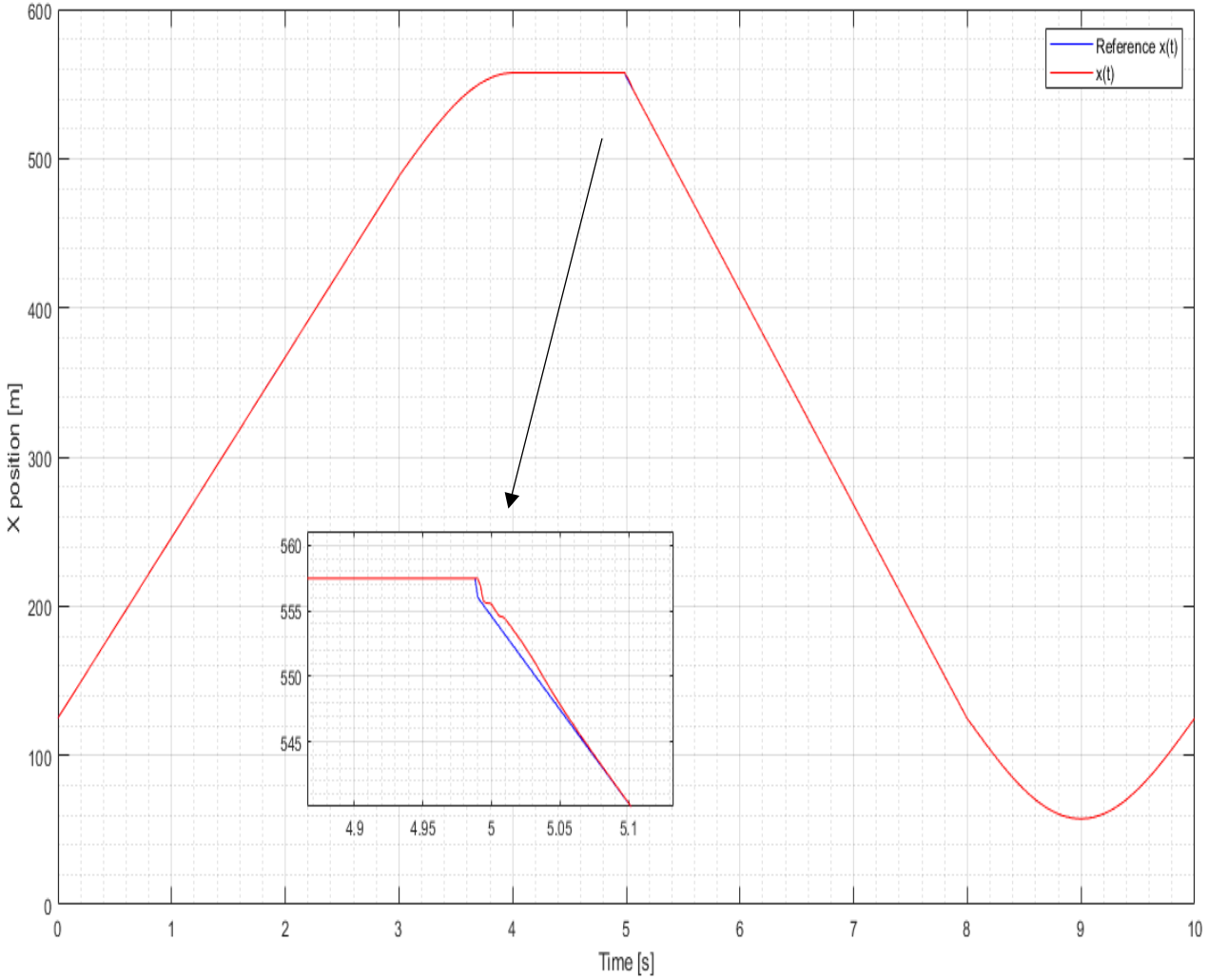
square curve

global asymptotic stability → assuming (vd,wd) bounded with bounded derivatives and (vd,wd) do not go concurrently to zero

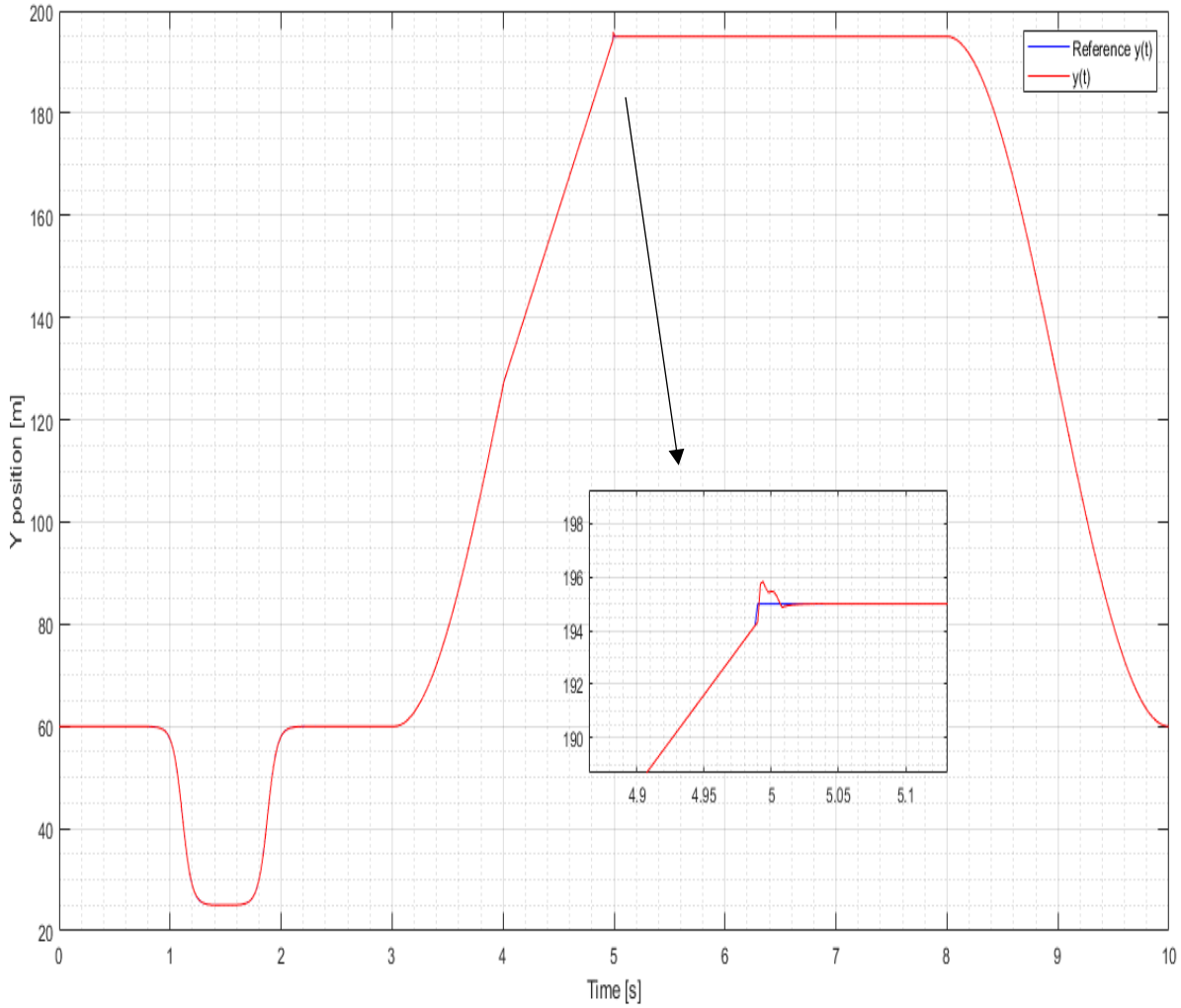
```
function inputs = fcn(err , velocities)
    d = 1 / sqrt(2);
    b = 10;
    vd = velocities(1);
    wd = velocities(2);
    e3 = err(3);

    % Safeguard for small ed
    e3 = sign(e3) * max(abs(e3), 1e-4);

    % Gains
    k1 = 2 * d * sqrt(b*vd+wd^2);
    k2 = b;
    k3 = k1;

    % Clamp kf to avoid large values
    kf = k2*vd*(sin(e3)/e3);
    max_kf = 800;
    kf = max(min(kf, max_kf), -max_kf);

    % Gain matrix
    k = [-k1, 0, 0;
         0, -kf, -k3];

    % Compute inputs
    inputs = k * err;
end
```

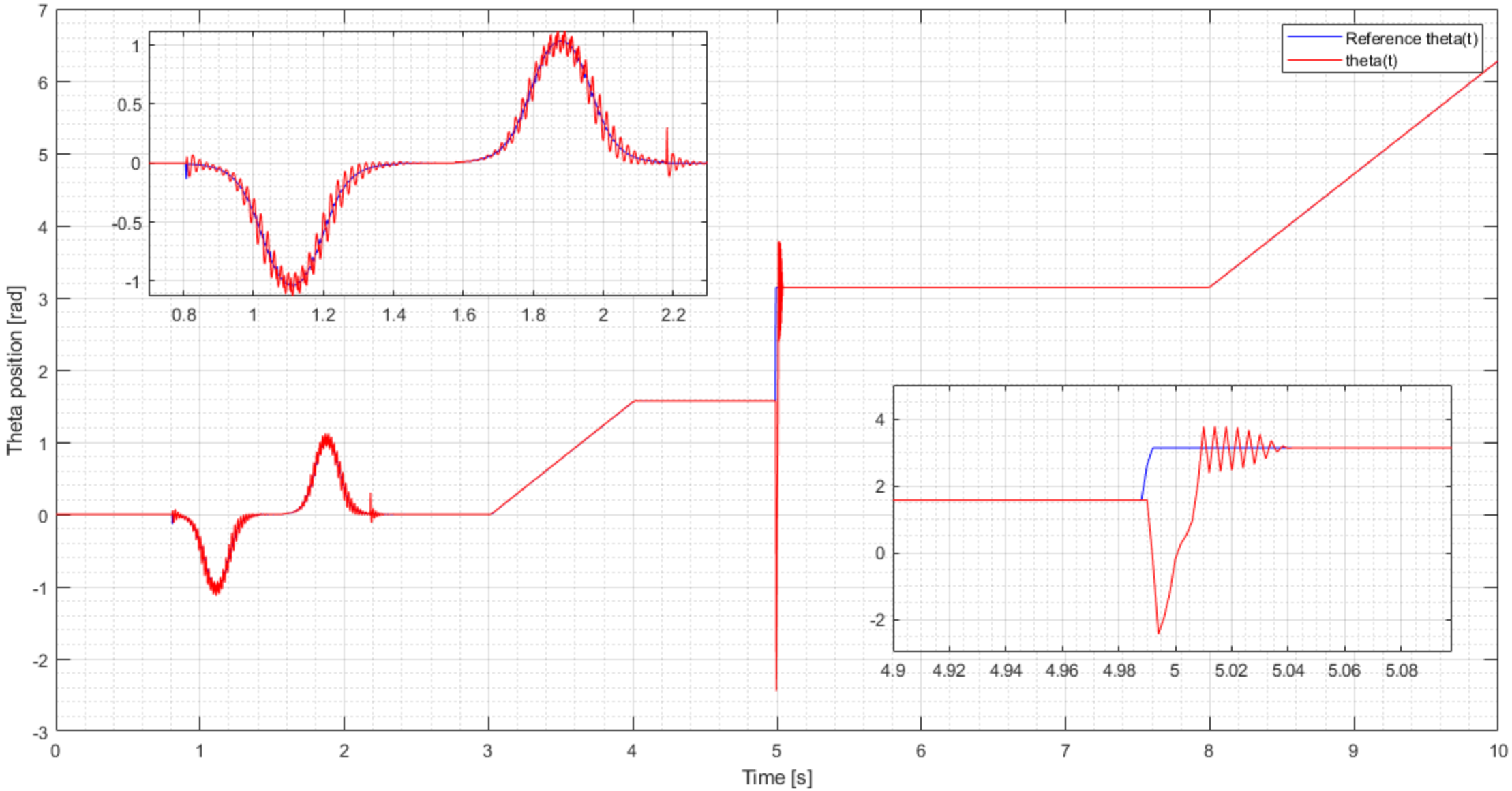
Plots

X TRACKING

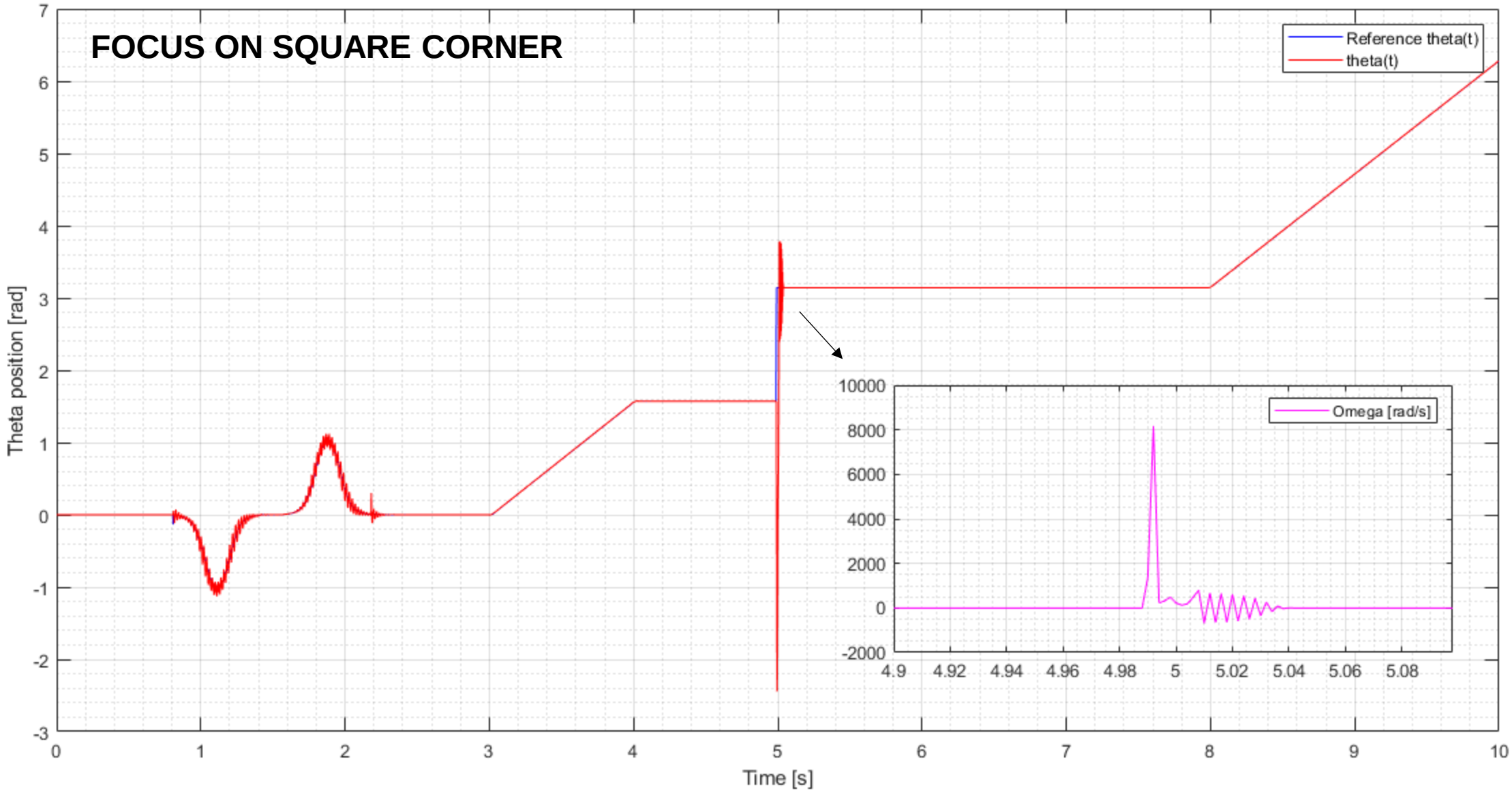


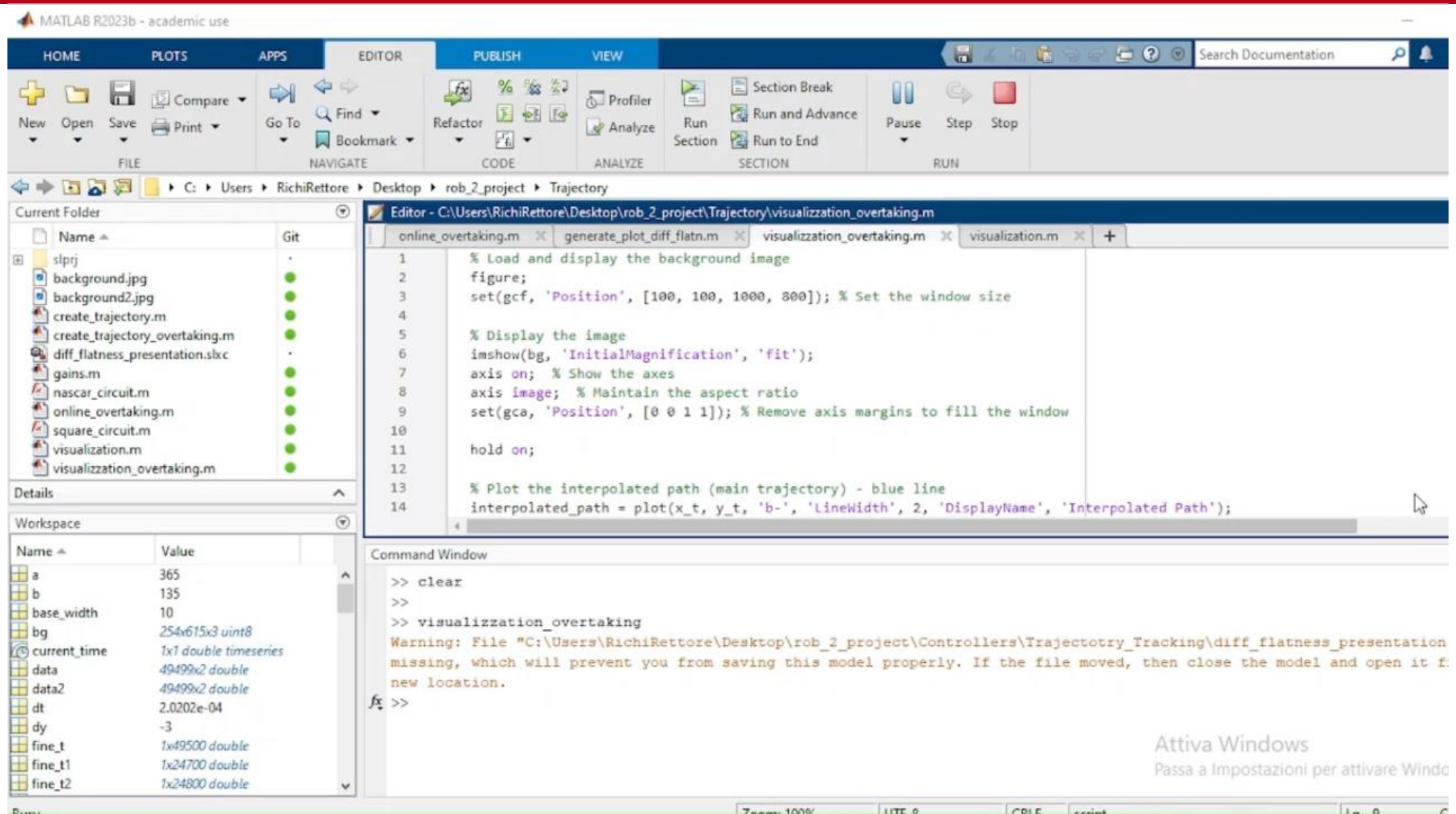
Y TRACKING



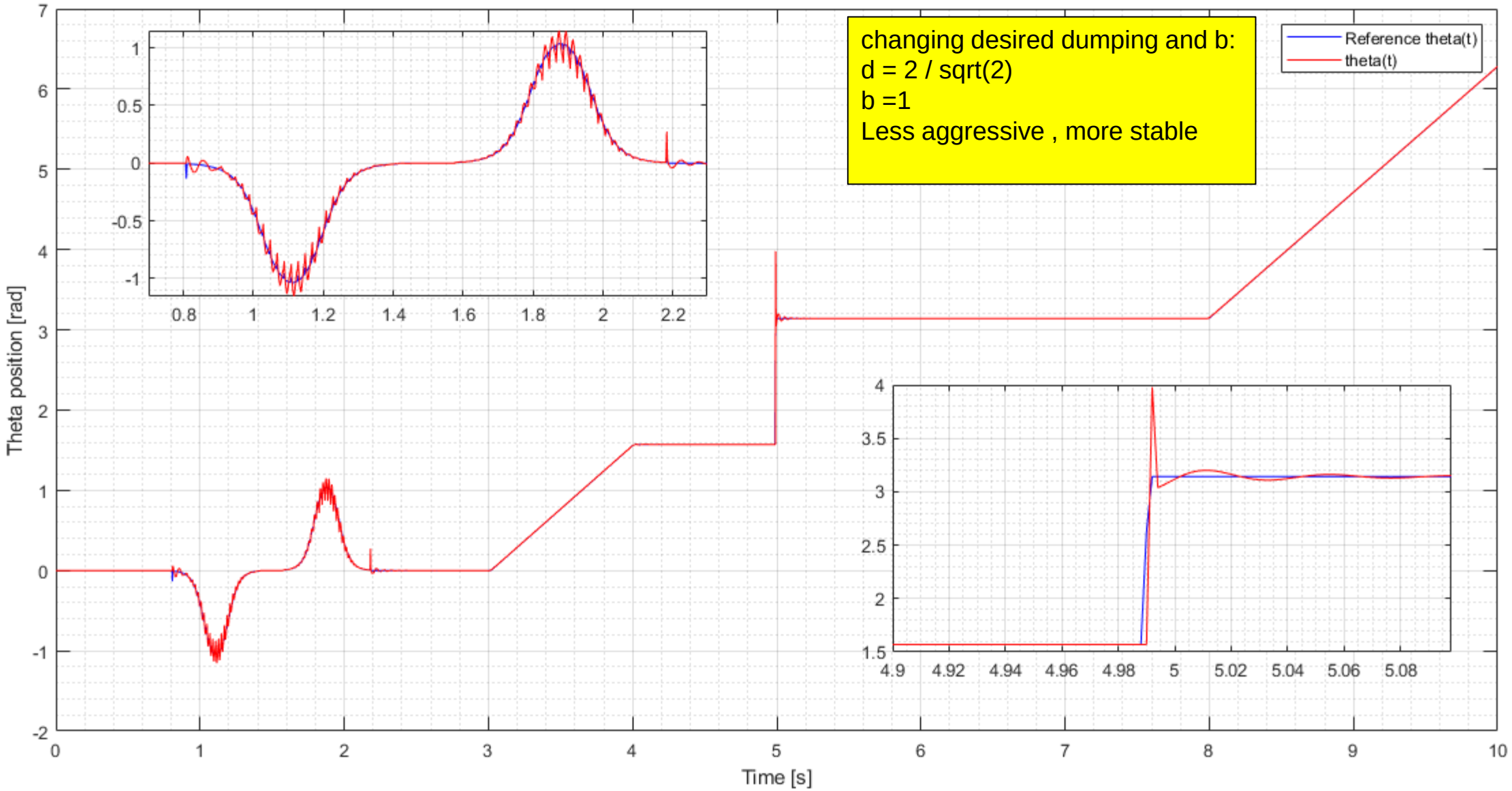

Reference theta(t)
theta(t)
Theta position [rad]
Time [s]

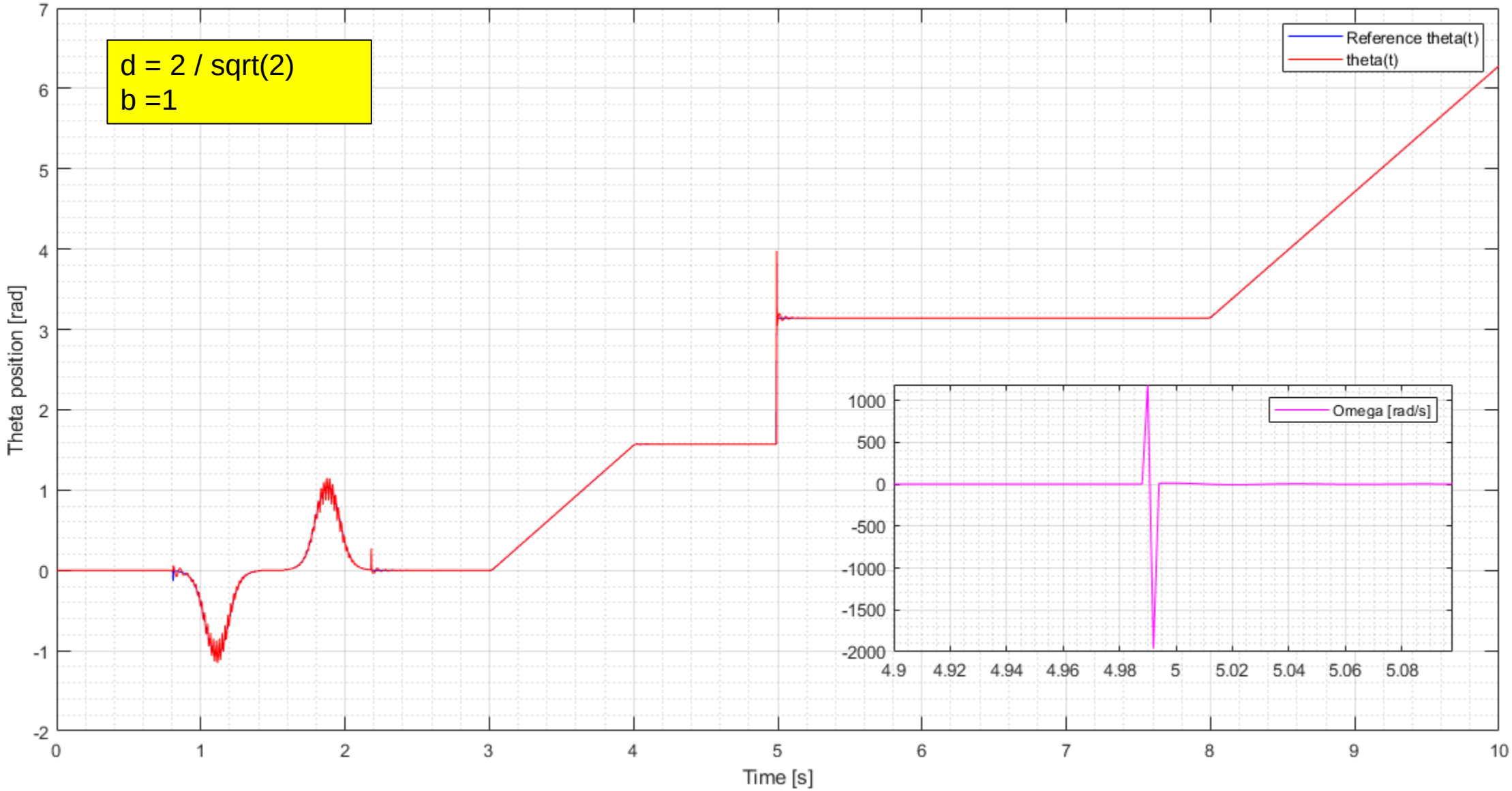
FOCUS ON SQUARE CORNER

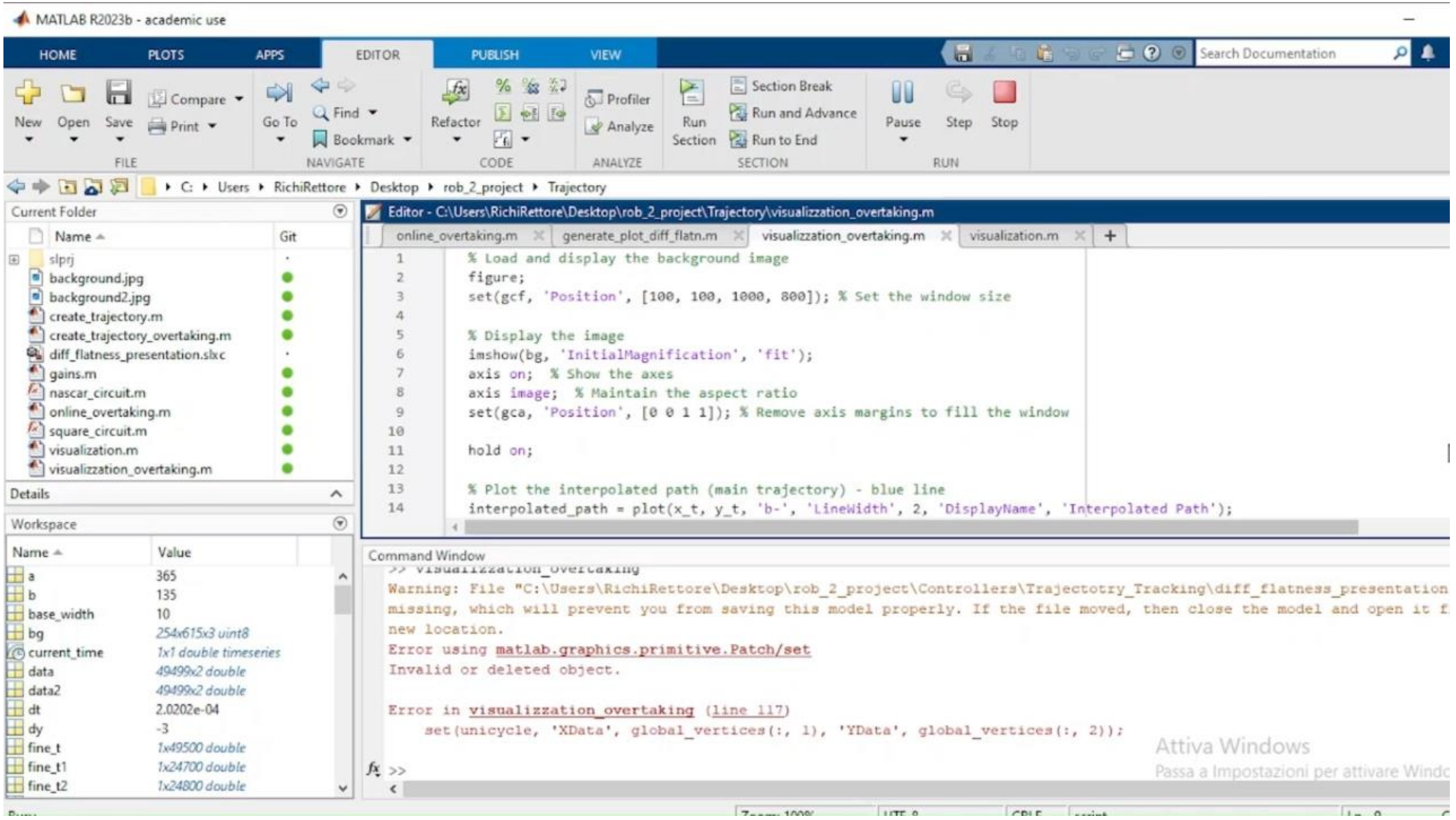


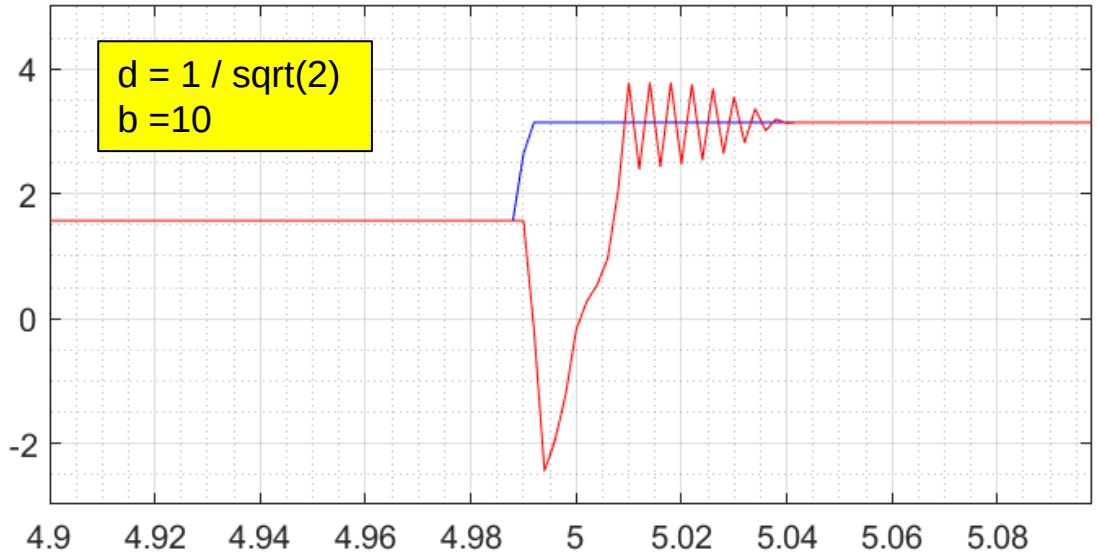

MATLAB R2023b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Pause
Step
Stop
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
C: ▸ Users ▸ RichiRettore ▸ Desktop ▸ rob_2_project ▸ Trajectory
Current Folder
Name
Git
slprj
background.jpg
background2.jpg
create_trajectory.m
create_trajectory_overtaking.m
diff_flatness_presentation.slxc
gains.m
nascar_circuit.m
online_overtaking.m
square_circuit.m
visualization.m
visualizzation_overtaking.m
Details
Workspace
Name
Value
a 365
b 135
base_width 10
bg 254x615x3 uint8
current_time 1x1 double timeseries
data 49499x2 double
data2 49499x2 double
dt 2.0202e-04
dy -3
fine_t 1x49500 double
fine_t1 1x24700 double
fine_t2 1x24800 double
Editor - C:\Users\RichiRettore\Desktop\rob_2_project\Trajectory\visualizzation_overtaking.m
online_overtaking.m
generate_plot_diff_flatn.m
visualizzation_overtaking.m
visualization.m
1 % Load and display the background image
2 figure;
3 set(gcf, 'Position', [100, 100, 1000, 800]); % Set the window size
4
5 % Display the image
6 imshow(bg, 'InitialMagnification', 'fit');
7 axis on; % Show the axes
8 axis image; % Maintain the aspect ratio
9 set(gca, 'Position', [0 0 1 1]); % Remove axis margins to fill the window
10
11 hold on;
12
13 % Plot the interpolated path (main trajectory) - blue line
14 interpolated_path = plot(x_t, y_t, 'b-', 'LineWidth', 2, 'DisplayName', 'Interpolated Path');
Command Window
>> clear
>>
>> visualizzation_overtaking
Warning: File "C:\Users\RichiRettore\Desktop\rob_2_project\Controllers\Trajectotry_Tracking\diff_flatness_presentation
missing, which will prevent you from saving this model properly. If the file moved, then close the model and open it f
new location.
>>
Attiva Windows
Passa a Impostazioni per attivare Windo

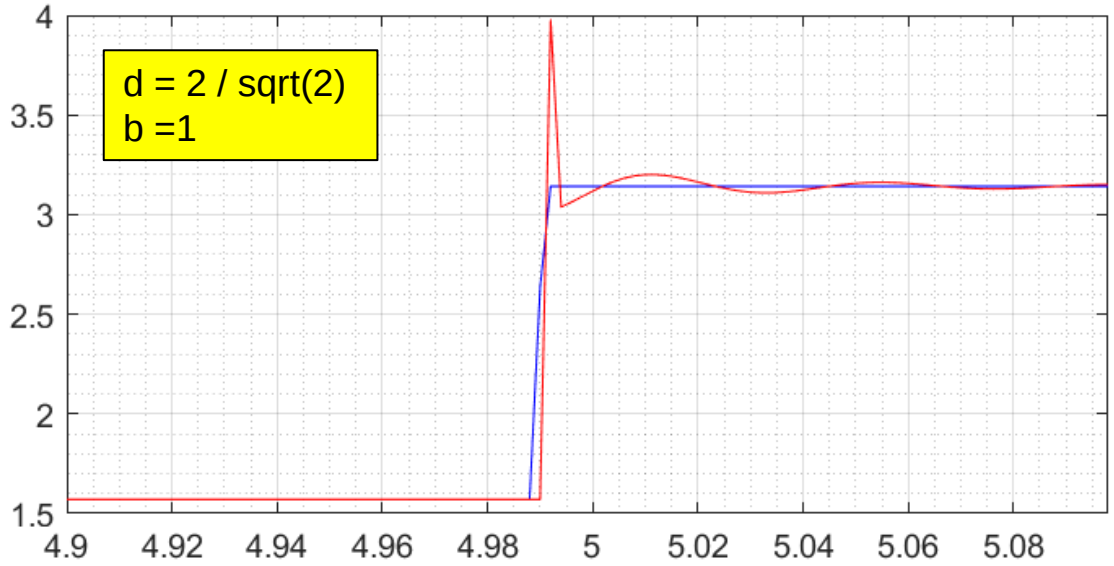
changing desired dumping and b:
d = 2 / sqrt(2)
b =1
Less aggressive , more stable

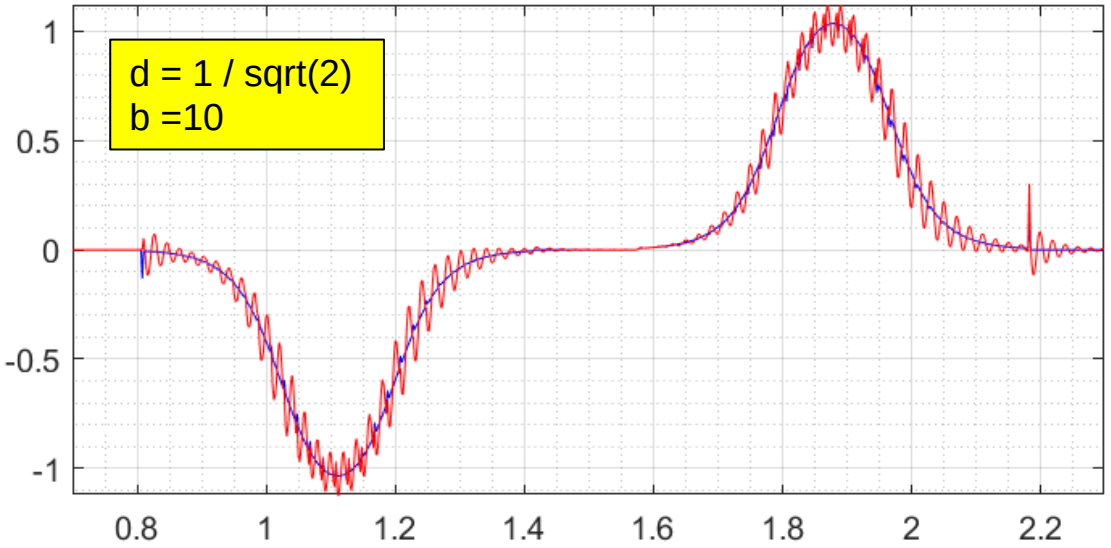


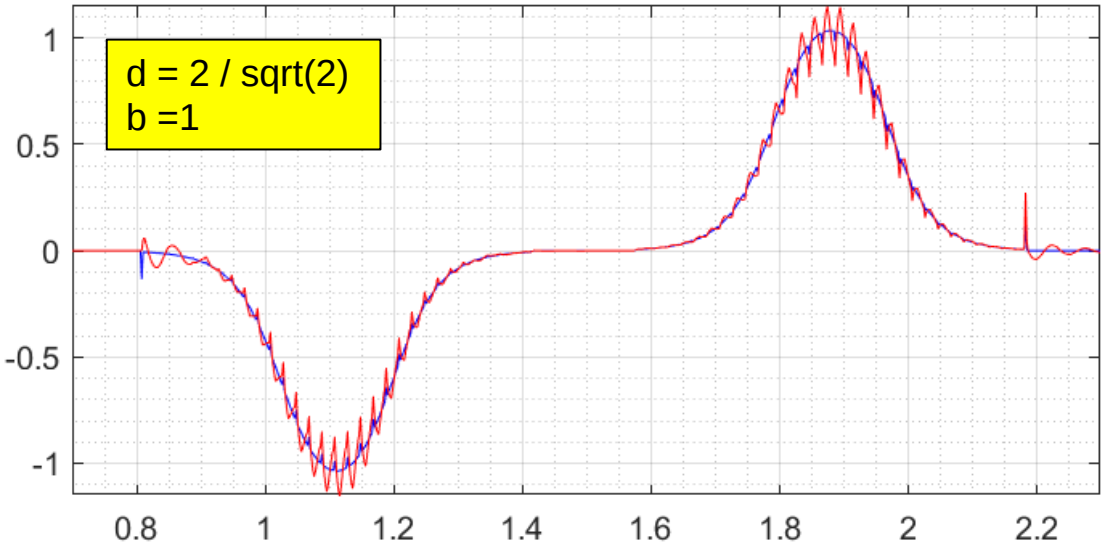

d = 2 / sqrt(2)
b =1
Reference theta(t)
theta(t)
Omega [rad/s]
Theta position [rad]
Time [s]


MATLAB R2023b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Pause
Step
Stop
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
C: ▸ Users ▸ RichiRettore ▸ Desktop ▸ rob_2_project ▸ Trajectory
Current Folder
Name
Git
slprj
background.jpg
background2.jpg
create_trajectory.m
create_trajectory_overtaking.m
diff_flatness_presentation.slxc
gains.m
nascar_circuit.m
online_overtaking.m
square_circuit.m
visualization.m
visualizzation_overtaking.m
Details
Workspace
Name
Value
a 365
b 135
base_width 10
bg 254x615x3 uint8
current_time 1x1 double timeseries
data 49499x2 double
data2 49499x2 double
dt 2.0202e-04
dy -3
fine_t 1x49500 double
fine_t1 1x24700 double
fine_t2 1x24800 double
Editor - C:\Users\RichiRettore\Desktop\rob_2_project\Trajectory\visualizzation_overtaking.m
online_overtaking.m
generate_plot_diff_flatn.m
visualizzation_overtaking.m
visualization.m
1 % Load and display the background image
2 figure;
3 set(gcf, 'Position', [100, 100, 1000, 800]); % Set the window size
4
5 % Display the image
6 imshow(bg, 'InitialMagnification', 'fit');
7 axis on; % Show the axes
8 axis image; % Maintain the aspect ratio
9 set(gca, 'Position', [0 0 1 1]); % Remove axis margins to fill the window
10
11 hold on;
12
13 % Plot the interpolated path (main trajectory) - blue line
14 interpolated_path = plot(x_t, y_t, 'b-', 'LineWidth', 2, 'DisplayName', 'Interpolated Path');
Command Window
Warning: File "C:\Users\RichiRettore\Desktop\rob_2_project\Controllers\Trajectotry_Tracking\diff_flatness_presentation
missing, which will prevent you from saving this model properly. If the file moved, then close the model and open it f
new location.
Error using matlab.graphics.primitive.Patch/set
Invalid or deleted object.
Error in visualizzation_overtaking (line 117)
set(unicycle, 'XData', global_vertices(:, 1), 'YData', global_vertices(:, 2));
>>
Attiva Windows
Passa a Impostazioni per attivare Windo


d = 1 / sqrt(2)
b =10

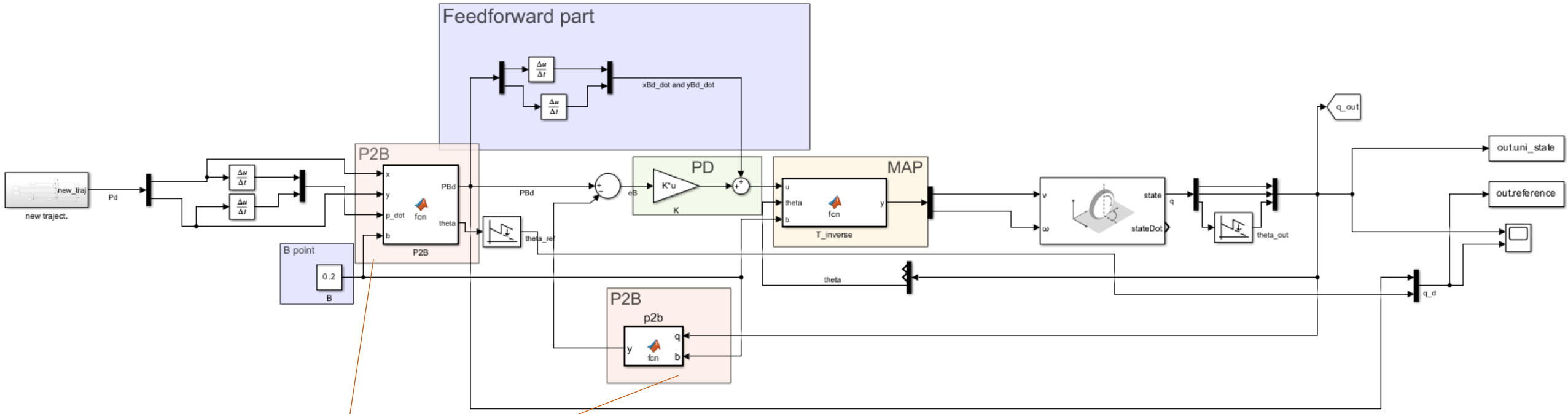

d = 2 / sqrt(2)
b =1


d = 1 / sqrt(2)
b =10

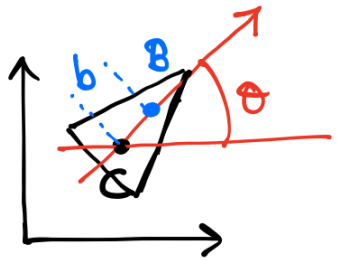

d = 2 / sqrt(2)
b =1

TRAJECTORY TRACKING - OUTPUT ERROR FEEDBACK

Feedback linearization based on a reference point on the sagittal axis



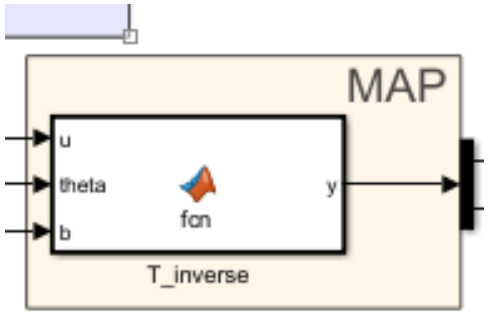
Change of coordinates: $x, y \rightarrow x_B, y_B$



Consider a point B along the sagittal axis with:

$$x_B = x + b\cos(\theta) \quad \text{and} \quad y_B = y + b\sin(\theta)$$

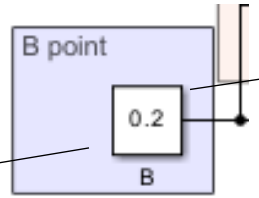
Map




```
function y = fcn(u,theta,b)
T_inverse=[ cos(theta),sin(theta);
            -sin(theta)/b,cos(theta)/b ];
vet=T_inverse*u;
v=vet(1);
w=vet(2);
y = vet;
```

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = T \begin{bmatrix} v \\ \omega \end{bmatrix}$$

$$\det = \mathrm{b}$$



Input-output linearized system:

$$\dot{x}_B = u_1, \quad \dot{y}_B = u_2, \quad \dot{\theta} = \frac{u_2 \cos\theta - u_1 \sin\theta}{b}$$

Controller design

Controller:

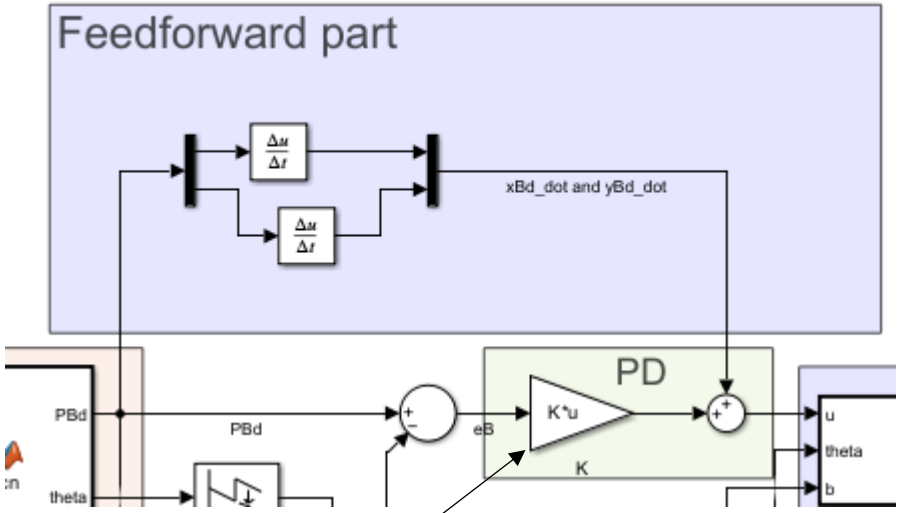
$$u_1 = k_{p1}(x_{Bd} - x_B) + \dot{x}_{Bd}$$
$$u_2 = k_{p2}(y_{Bd} - y_B) + \dot{y}_{Bd}$$

θ is not controlled

$$\dot{x}_{Bd} = u_1 \quad \text{and} \quad \dot{y}_{Bd} = u_2$$

Single Integrator , Linear Dynamics , Decoupled

$\Rightarrow$ SISO Controller



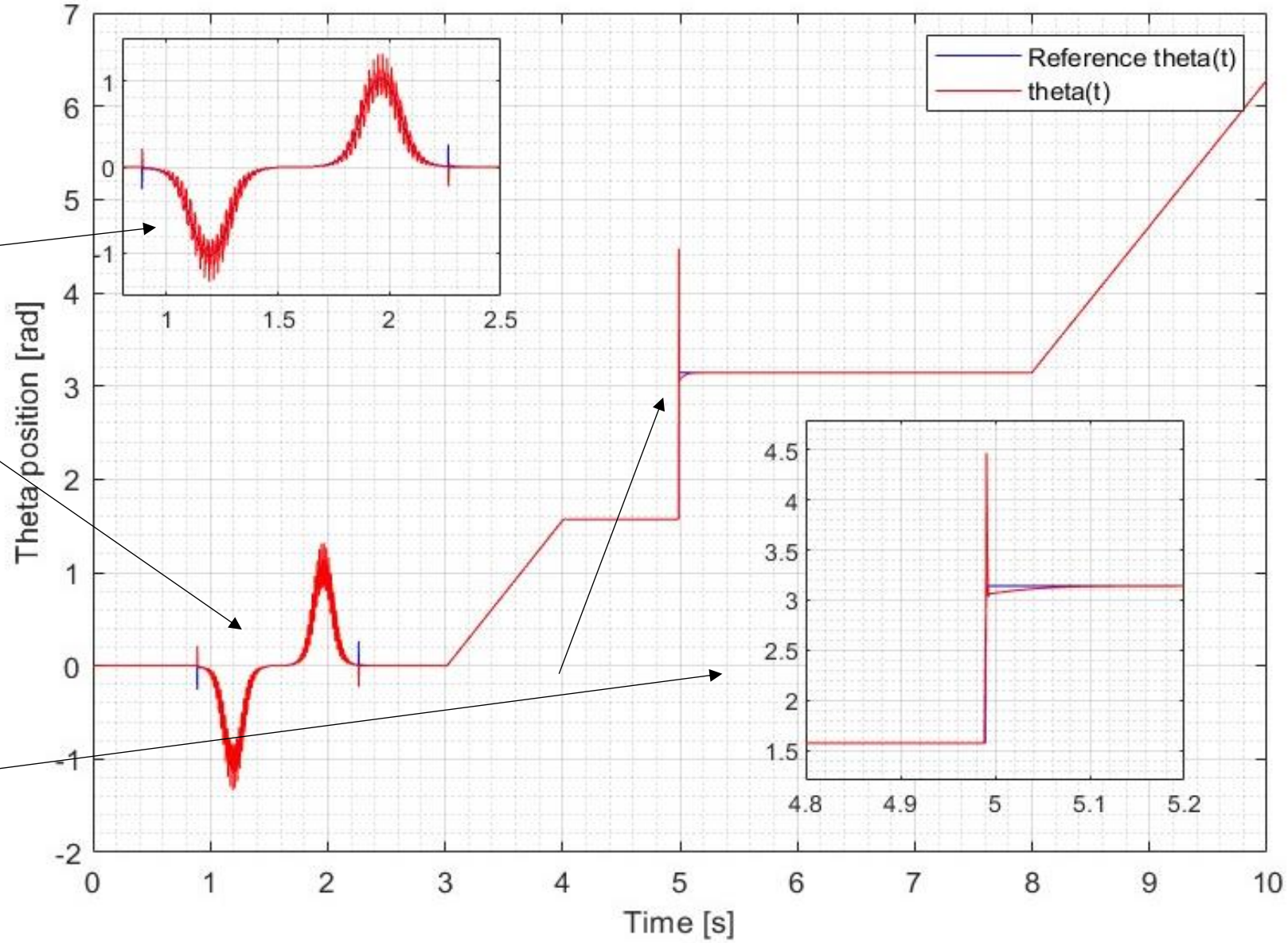

```
k_p1 = 45;
k_p2 = 30;
K=[ k_p1, 0;
    0, k_p2  ];
```

Pro:

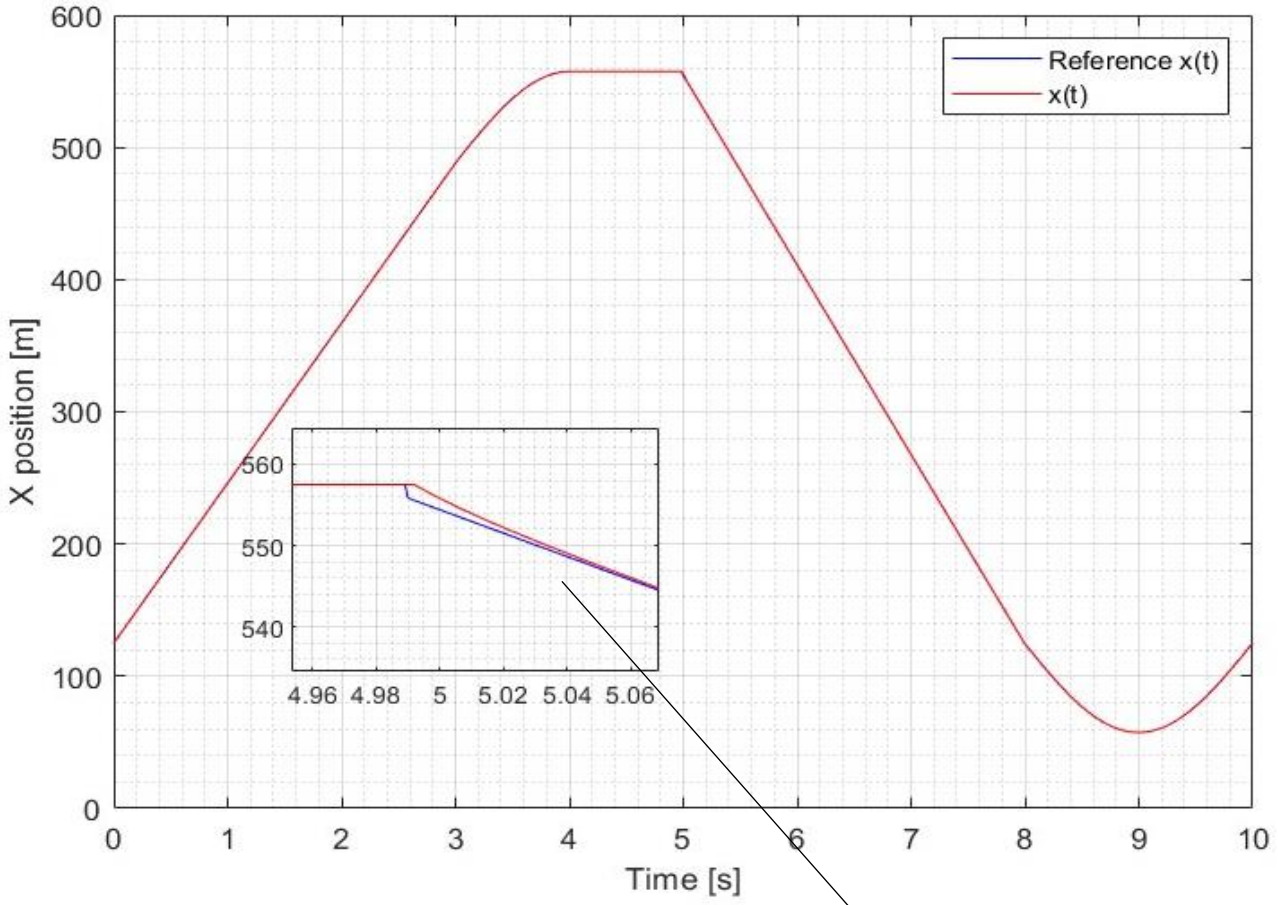
trajectory for B can be arbitrary

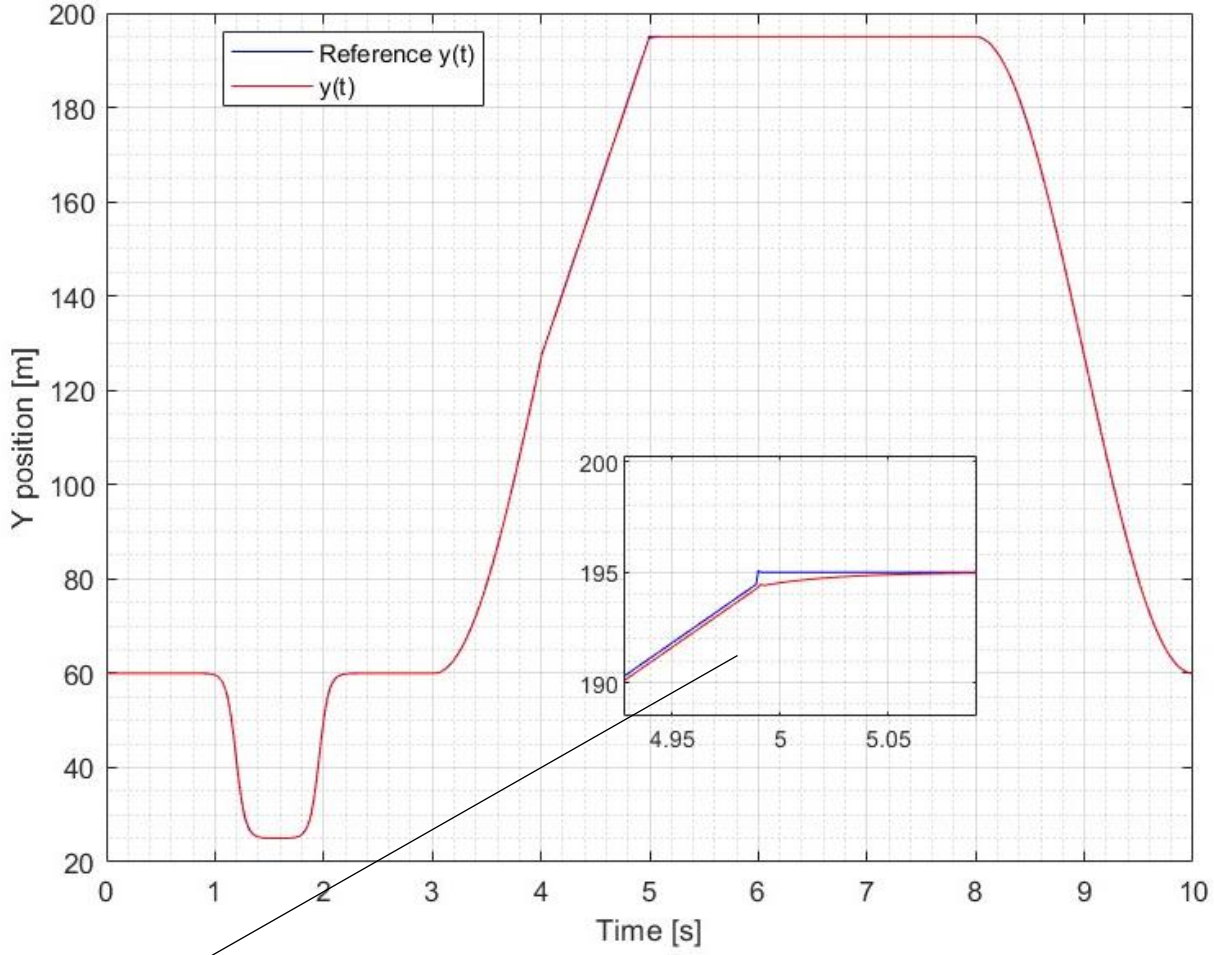
Plot

Overtaking procedure



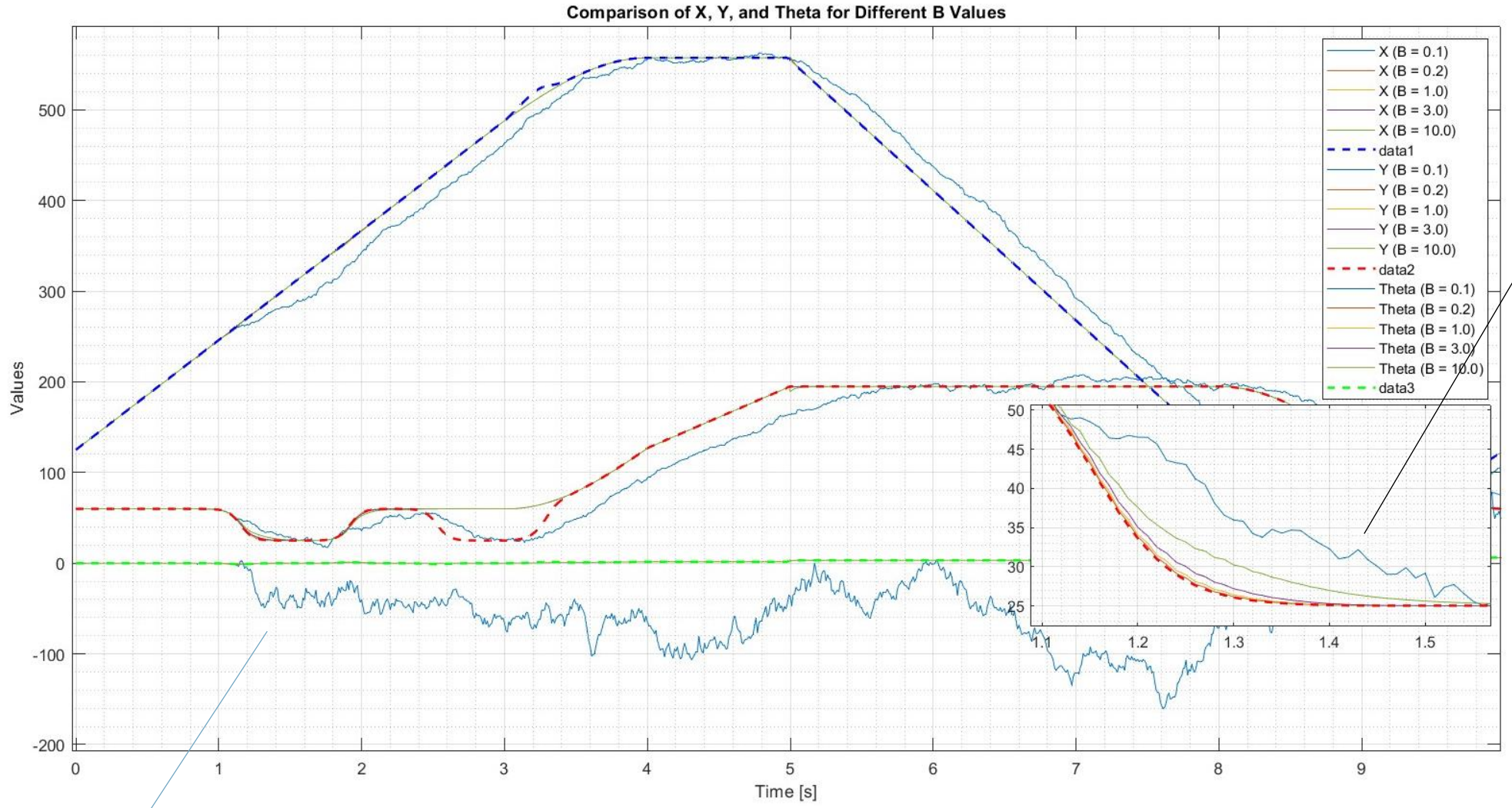
No problem with square corner





square corner

Plot

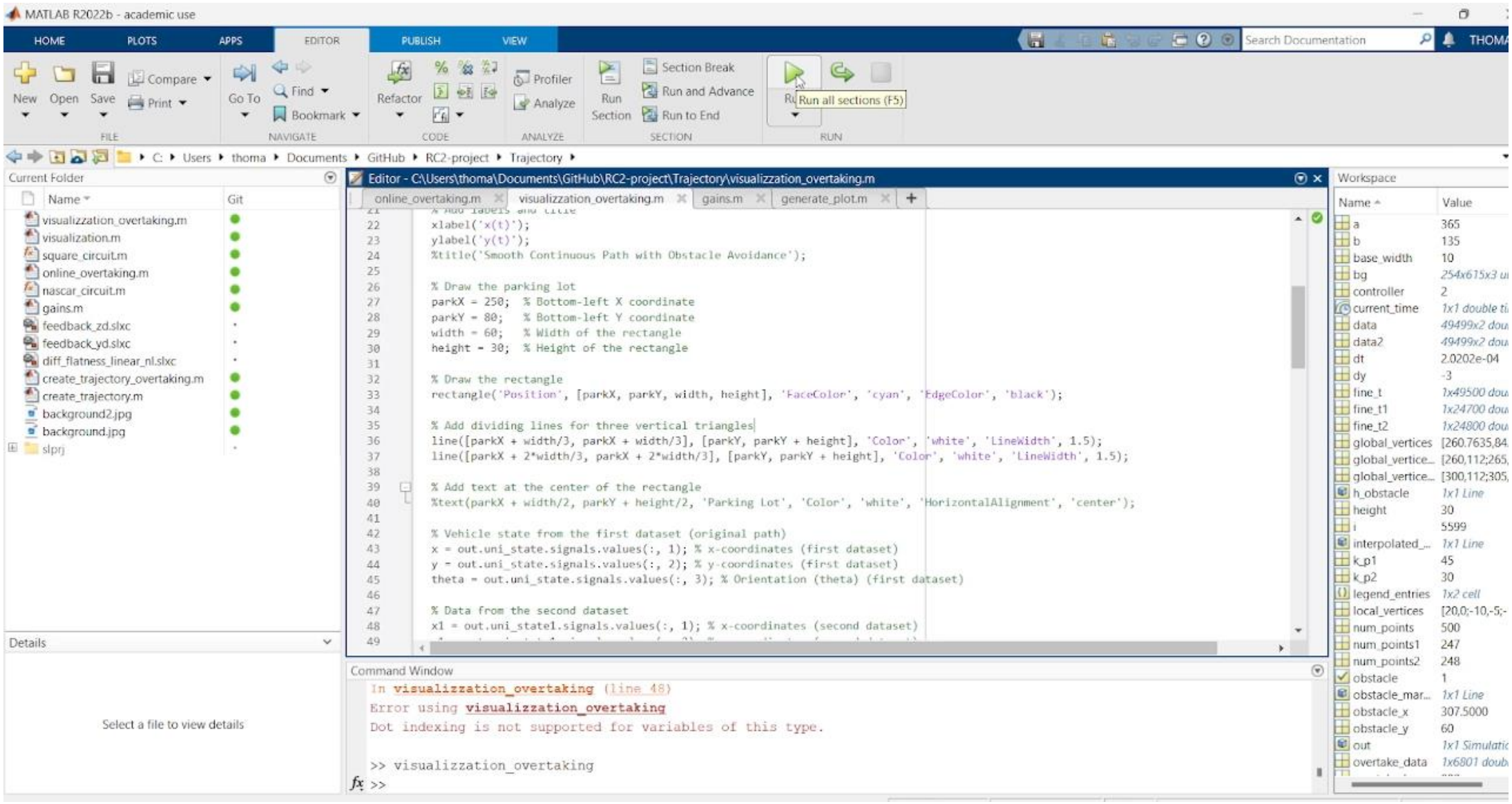


Overtaking procedure

b small:
Better trajectory
Higher rotational speed

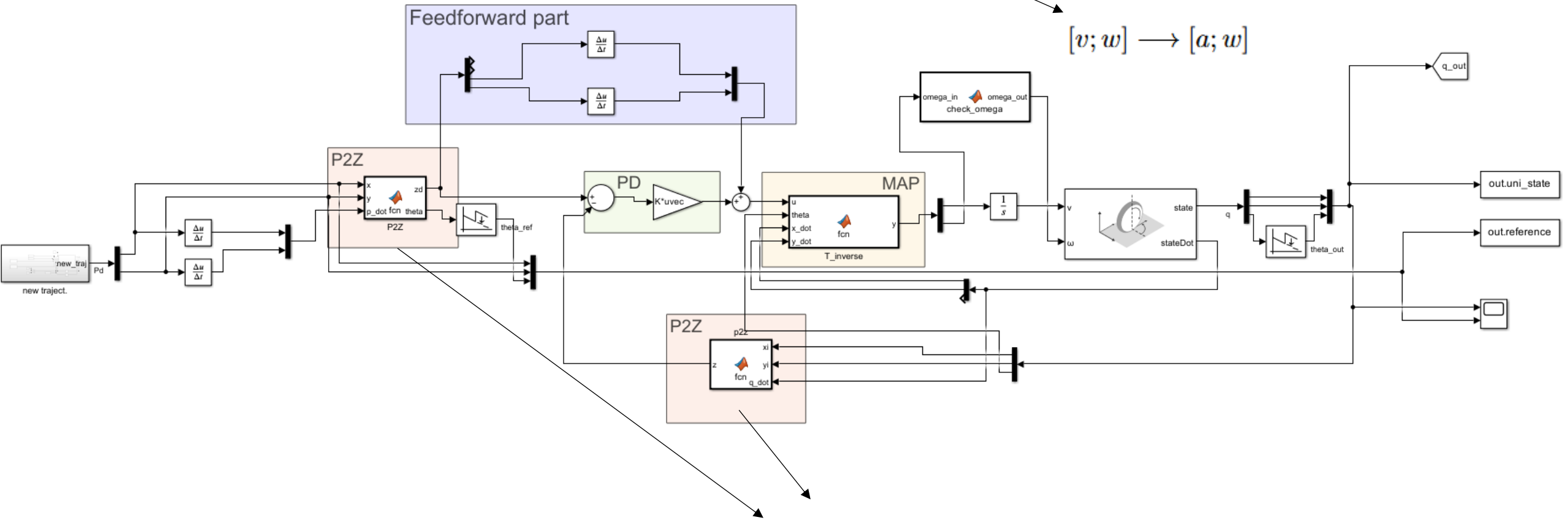
b large:
Less rotational speed
A smoother trajectory

b close to 0 $\rightarrow$ $\dot{\theta} = \frac{u_2 \cos\theta - u_1 \sin\theta}{b}$


MATLAB R2022b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Run all sections (F5)
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
C: ▸ Users ▸ thoma ▸ Documents ▸ GitHub ▸ RC2-project ▸ Trajectory ▸
Current Folder
Name
Git
visualizzation_overtaking.m
visualization.m
square_circuit.m
online_overtaking.m
nascar_circuit.m
gains.m
feedback_zd.slxc
feedback_yd.slxc
diff_flatness_linear_nl.slxc
create_trajectory_overtaking.m
create_trajectory.m
background2.jpg
background.jpg
slprj
Details
Select a file to view details
Editor - C:\Users\thoma\Documents\GitHub\RC2-project\Trajectory\visualizzation_overtaking.m
online_overtaking.m
visualizzation_overtaking.m
gains.m
generate_plot.m
xlabel('x(t)');
ylabel('y(t)');
%title('Smooth Continuous Path with Obstacle Avoidance');
% Draw the parking lot
parkX = 250; % Bottom-left X coordinate
parkY = 80; % Bottom-left Y coordinate
width = 60; % Width of the rectangle
height = 30; % Height of the rectangle
% Draw the rectangle
rectangle('Position', [parkX, parkY, width, height], 'FaceColor', 'cyan', 'EdgeColor', 'black');
% Add dividing lines for three vertical triangles
line([parkX + width/3, parkX + width/3], [parkY, parkY + height], 'Color', 'white', 'LineWidth', 1.5);
line([parkX + 2*width/3, parkX + 2*width/3], [parkY, parkY + height], 'Color', 'white', 'LineWidth', 1.5);
% Add text at the center of the rectangle
%text(parkX + width/2, parkY + height/2, 'Parking Lot', 'Color', 'white', 'HorizontalAlignment', 'center');
% Vehicle state from the first dataset (original path)
x = out.uni_state.signals.values(:, 1); % x-coordinates (first dataset)
y = out.uni_state.signals.values(:, 2); % y-coordinates (first dataset)
theta = out.uni_state.signals.values(:, 3); % Orientation (theta) (first dataset)
% Data from the second dataset
x1 = out.uni_state1.signals.values(:, 1); % x-coordinates (second dataset)
Command Window
In visualizzation_overtaking (line 48)
Error using visualizzation_overtaking
Dot indexing is not supported for variables of this type.
>> visualizzation_overtaking
>>
Workspace
Name
Value
a 365
b 135
base_width 10
bg 254x615x3 u
controller 2
current_time 1x1 double ti
data 49499x2 dou
data2 49499x2 dou
dt 2.0202e-04
dy -3
fine_t 1x49500 dou
fine_t1 1x24700 dou
fine_t2 1x24800 dou
global_vertices [260.7635,84.
global_vertice... [260,112;265,
global_vertice... [300,112;305,
h_obstacle 1x1 Line
height 30
i 5599
interpolated_... 1x1 Line
k_p1 45
k_p2 30
legend_entries 1x2 cell
local_vertices [20,0;-10,-5;-
num_points 500
num_points1 247
num_points2 248
obstacle 1
obstacle_mar... 1x1 Line
obstacle_x 307.5000
obstacle_y 60
out 1x1 Simulatic
overtake_data 1x6801 doub

Feedback linearization based on second order derivatives

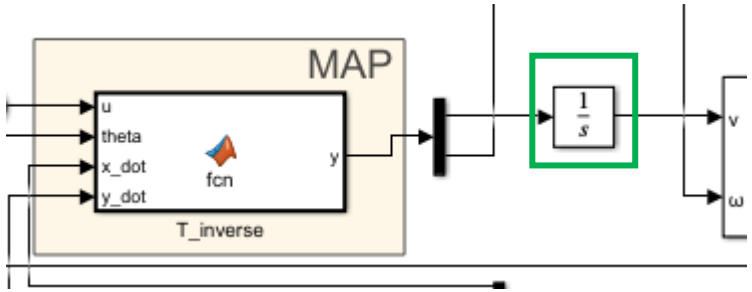
$$[v; w] \longrightarrow [a; w]$$




```
function z = fcn(x,y, q_dot)
x_dot = q_dot(1);%=v*cos(theta);
y_dot = q_dot(2);%=v*sin(theta);
z = [x; y; x_dot; y_dot];  % z is a 4x1 vector [z1;z2;z3;z4]
```

Map

```
function y = fcn(u,theta,x_dot,y_dot)
v=sqrt(x_dot^2+y_dot^2);
if abs(v) < 1e-4
   v=1e-4;
   T_inverse=[cos(theta),sin(theta);-sin(theta)/v,cos(theta)/v];
else
   T_inverse=[cos(theta),sin(theta);-sin(theta)/v,cos(theta)/v];
end
vet=T_inverse*u;
omega_in=vet(2);
if omega_in>700
   vet(2) = 700;  % Set limit omega
end
y = vet;
```



$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = T(\theta, v) \begin{bmatrix} a \\ w \end{bmatrix}$$

det v

Ensure that ω is finite

It prevents singularity due to the angle at 90° with v=0

Linearized system:

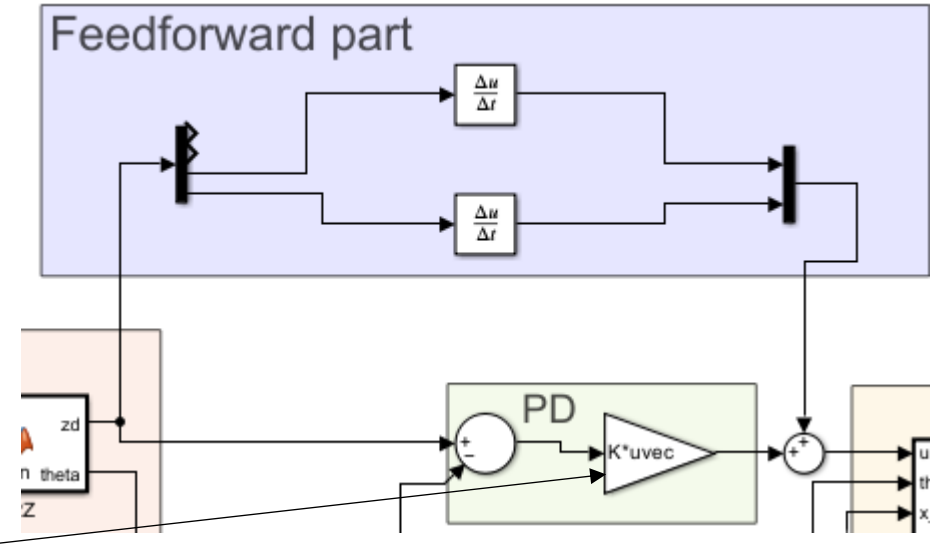
$$\ddot{x} = u_1, \quad \ddot{y} = u_2$$

Controller design

$$u_1 = \ddot{x}_d + k_{p1}(x_d - x) + k_{d1}(\dot{x}_d - \dot{x})$$
$$u_2 = \ddot{y}_d + k_{p2}(y_d - y) + k_{d2}(\dot{y}_d - \dot{y})$$

PD

```
k_p1 = 110;
k_p2 = 110;
k_d1 = 90;
k_d2 = 90;

K=[ k_p1, 0, k_d1, 0;
    0, k_p2, 0, k_d2  ];
```



~~$\dot{x}_{Bd} = u_1$ and $\dot{y}_{Bd} = u_2$~~ → $\ddot{x}_{Bd} = u_1, \quad \ddot{y}_{Bd} = u_2$

~~Single Integrator~~ , Linear Dynamics , Decoupled

$\Rightarrow$ SISO Controller

double integrator

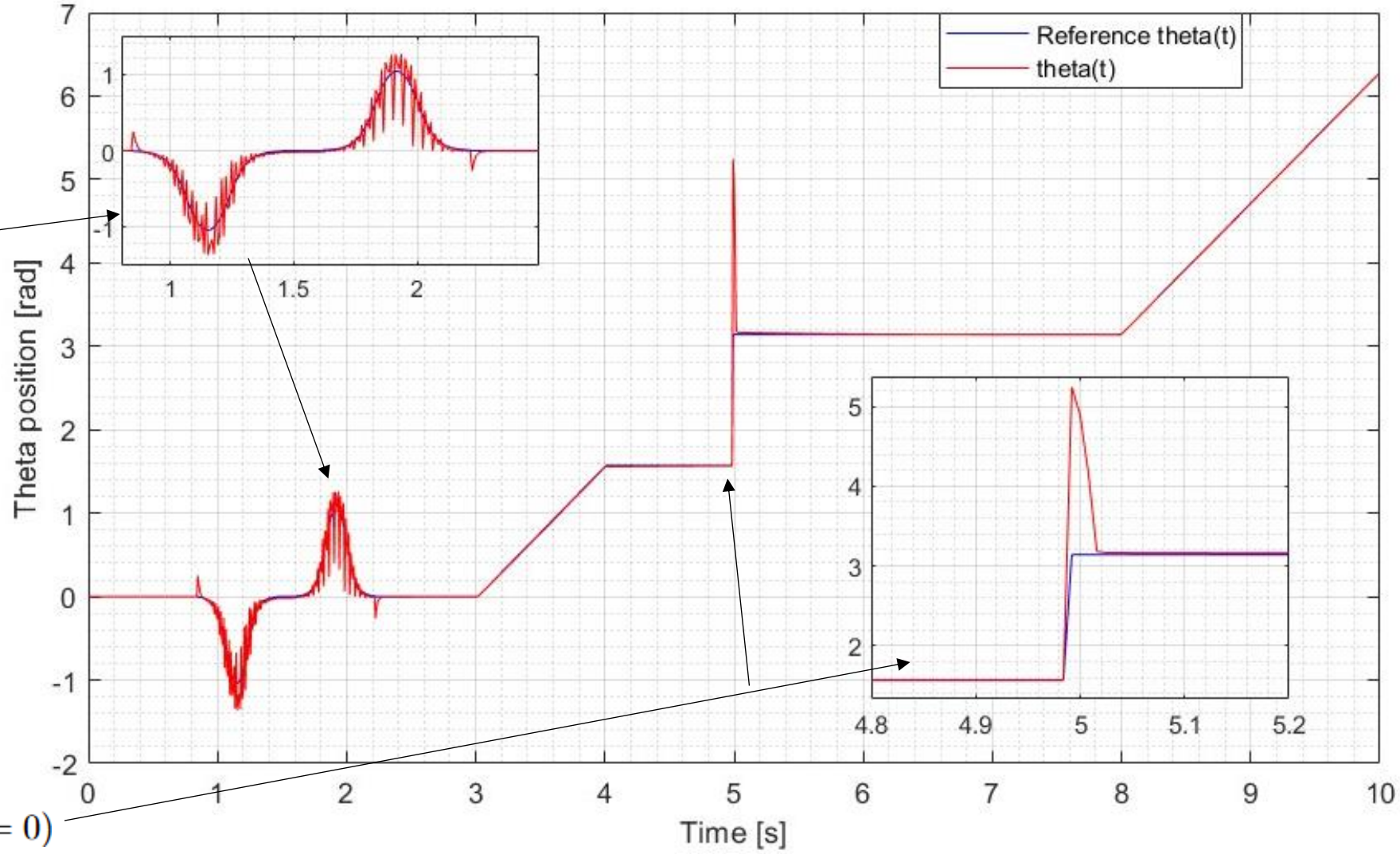
Need:

trajectory must be twice differentiable; $v_d \neq 0$

Plot

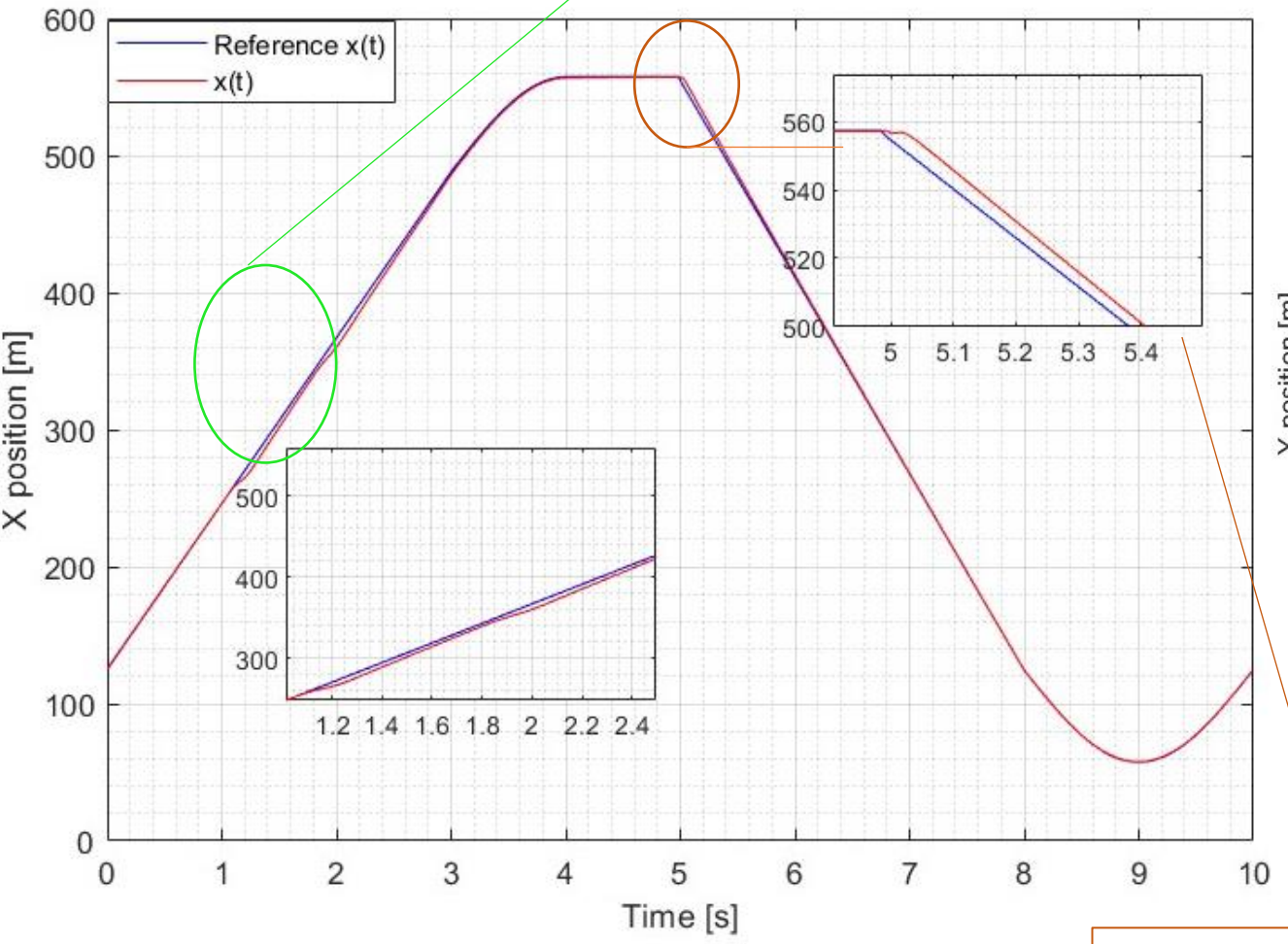
Overtaking procedure

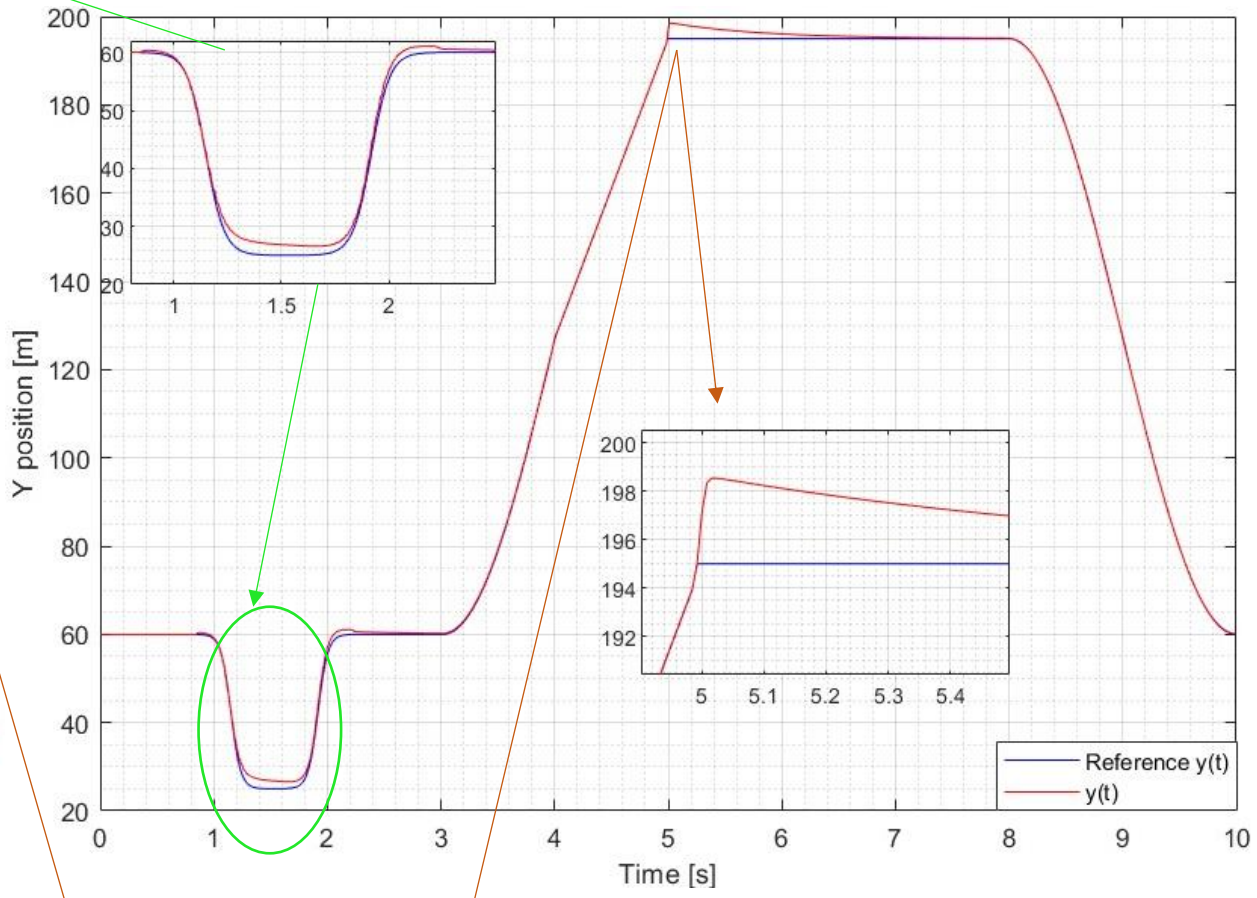
problem with square corner ($v_d = 0$)



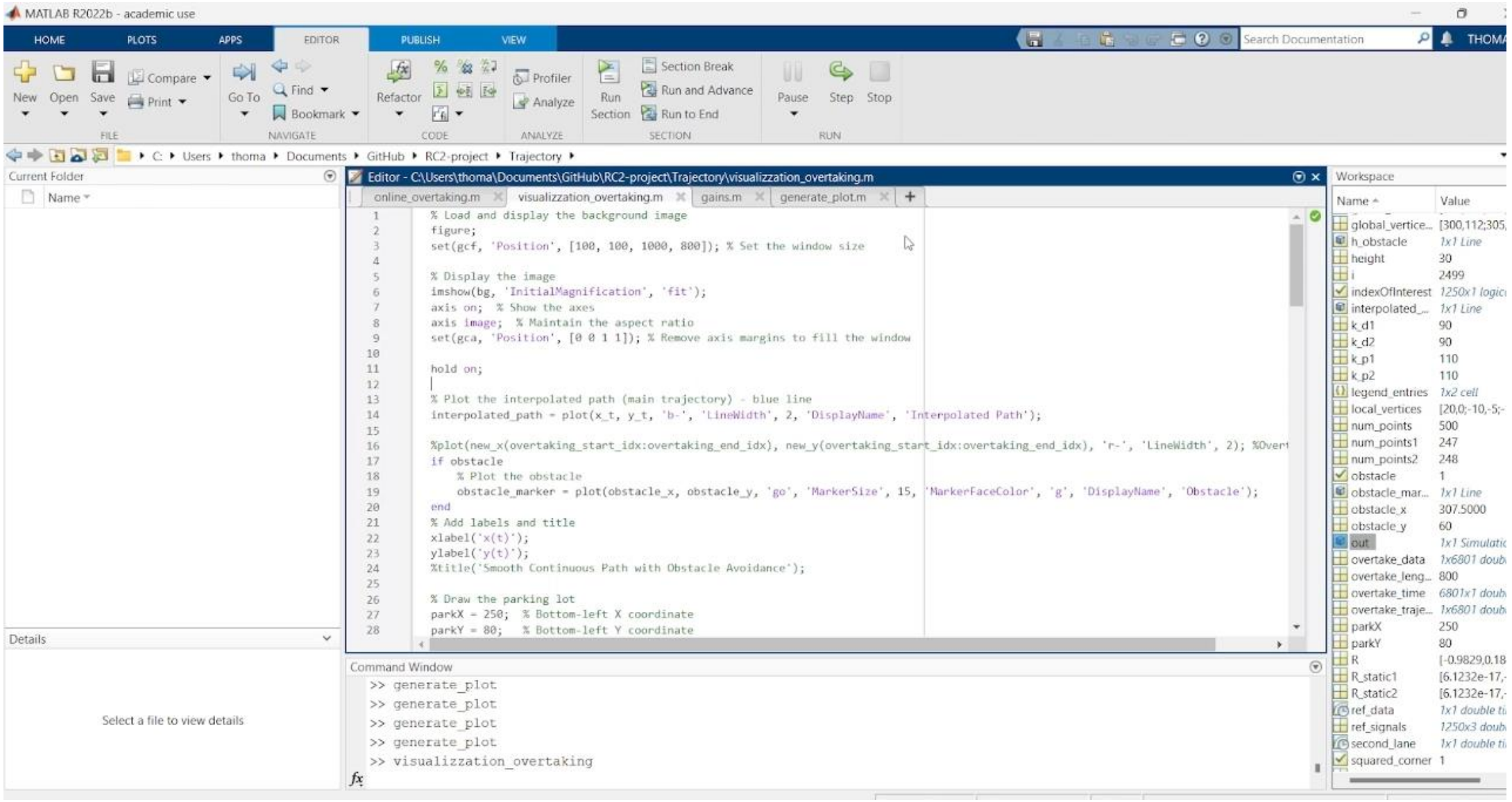
Plot

Due to the overtaking





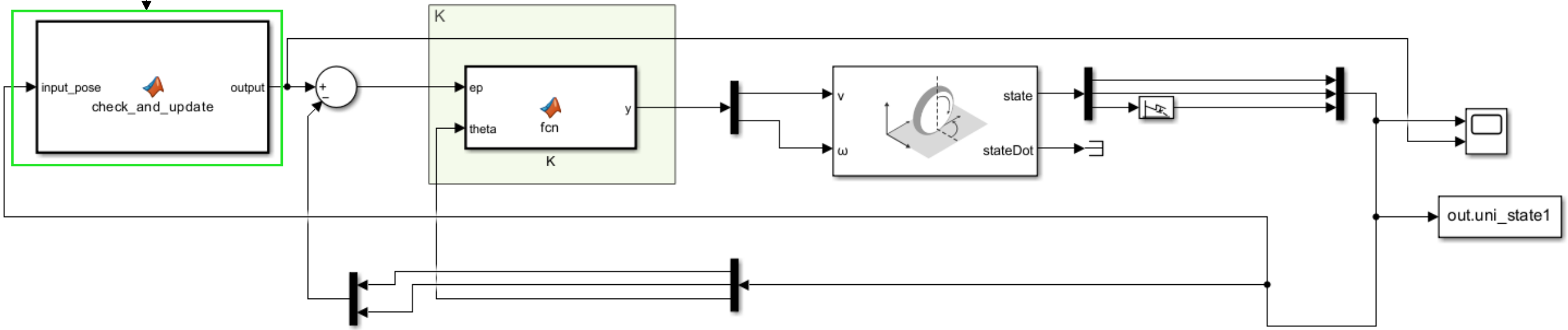
problem with square corner ($v_d = 0$)


MATLAB R2022b - academic use
HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW
Search Documentation
New
Open
Save
Compare
Print
Go To
Find
Bookmark
Refactor
Profiler
Analyze
Run Section
Section Break
Run and Advance
Run to End
Pause
Step
Stop
FILE
NAVIGATE
CODE
ANALYZE
SECTION
RUN
C: ▸ Users ▸ thoma ▸ Documents ▸ GitHub ▸ RC2-project ▸ Trajectory ▸
Current Folder
Name
Details
Select a file to view details
Editor - C:\Users\thoma\Documents\GitHub\RC2-project\Trajectory\visualizzation_overtaking.m
online_overtaking.m
visualizzation_overtaking.m
gains.m
generate_plot.m
% Load and display the background image
figure;
set(gcf, 'Position', [100, 100, 1000, 800]); % Set the window size
% Display the image
imshow(bg, 'InitialMagnification', 'fit');
axis on; % Show the axes
axis image; % Maintain the aspect ratio
set(gca, 'Position', [0 0 1 1]); % Remove axis margins to fill the window
hold on;
% Plot the interpolated path (main trajectory) - blue line
interpolated_path = plot(x_t, y_t, 'b-', 'LineWidth', 2, 'DisplayName', 'Interpolated Path');
%plot(new_x(overtaking_start_idx:overtaking_end_idx), new_y(overtaking_start_idx:overtaking_end_idx), 'r-', 'LineWidth', 2); %Overt
if obstacle
% Plot the obstacle
obstacle_marker = plot(obstacle_x, obstacle_y, 'go', 'MarkerSize', 15, 'MarkerFaceColor', 'g', 'DisplayName', 'Obstacle');
end
% Add labels and title
xlabel('x(t)');
ylabel('y(t)');
%title('Smooth Continuous Path with Obstacle Avoidance');
% Draw the parking lot
parkX = 250; % Bottom-left X coordinate
parkY = 80; % Bottom-left Y coordinate
Command Window
>> generate_plot
>> generate_plot
>> generate_plot
>> generate_plot
>> visualizzation_overtaking
Workspace
Name
Value
global_vertice... [300,112;305,
h_obstacle 1x1 Line
height 30
i 2499
indexOfInterest 1250x1 logic
interpolated_... 1x1 Line
k_d1 90
k_d2 90
k_p1 110
k_p2 110
legend_entries 1x2 cell
local_vertices [20,0;-10,-5;-
num_points 500
num_points1 247
num_points2 248
obstacle 1
obstacle_mar... 1x1 Line
obstacle_x 307.5000
obstacle_y 60
out 1x1 Simulatic
overtake_data 1x6801 doub
overtake_leng... 800
overtake_time 6801x1 doub
overtake_traje... 1x6801 doub
parkX 250
parkY 80
R [-0.9829,0.18
R_static1 [6.1232e-17,-
R_static2 [6.1232e-17,-
ref_data 1x1 double ti
ref_signals 1250x3 doub
second_lane 1x1 double ti
squared_corner 1

REGULATION

Cartesian regulation

Provides the two waypoints



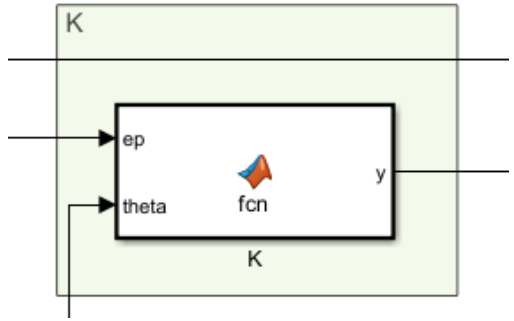
Drive the unicycle to a given Cartesian position

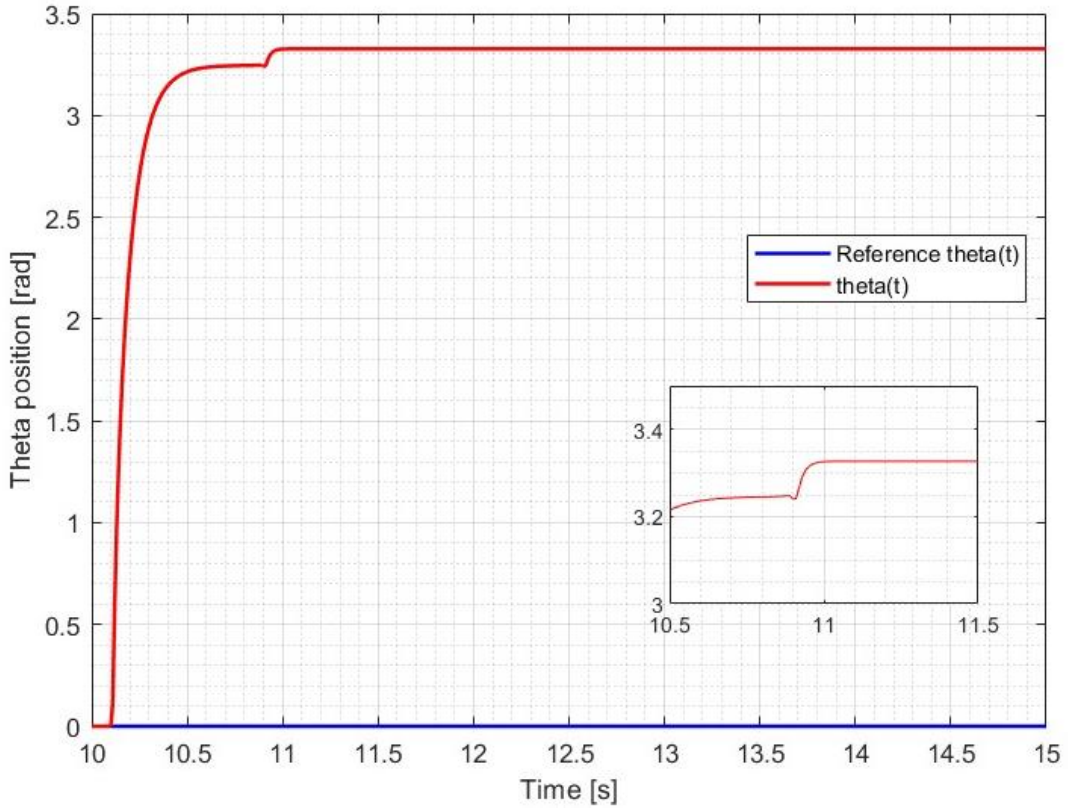
$$q_{des} = \begin{bmatrix} P_{des} \\ \forall \end{bmatrix}$$

The final orientation is not controlled

Control law

```
function y = fcn(ep,theta)
kv=40;
kw=50;
v=kv*(ep(1)*cos(theta)+ep(2)*sin(theta));
w=kw*(atan2(-ep(2),-ep(1))+pi-theta);
y = [v;w];
```





$$\begin{aligned} v &= k_v((x_{des} - x)\cos(\theta) + (y_{des} - y)\sin(\theta)) \\ \omega &= k_\omega(\text{atan2}(y, x) + \pi - \theta) \end{aligned}$$

γ

θ is not controlled

v is proportional to ep on the sagittal axis

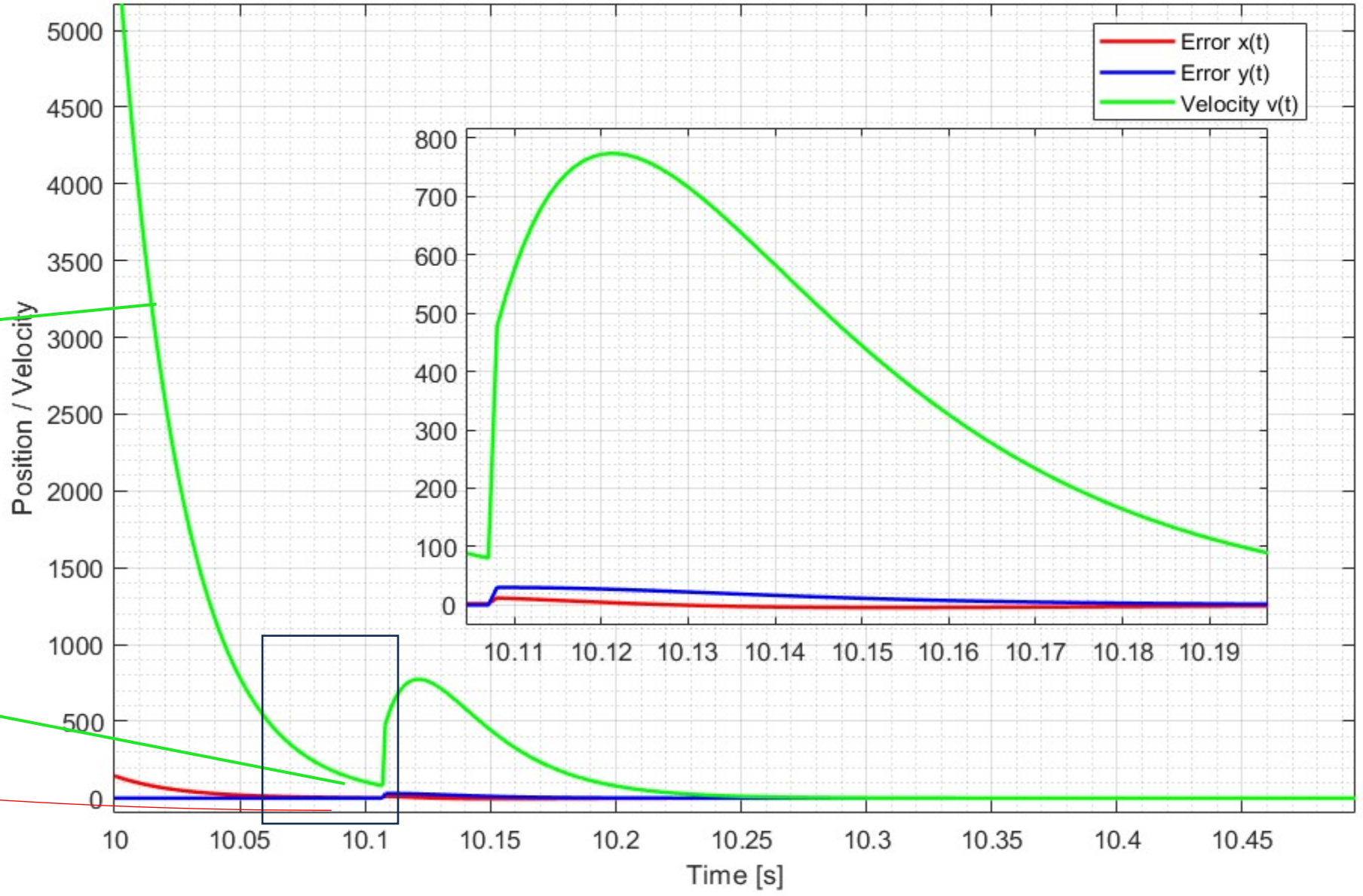
$\omega \propto$ Pointing Error

Plot

Proportional trend

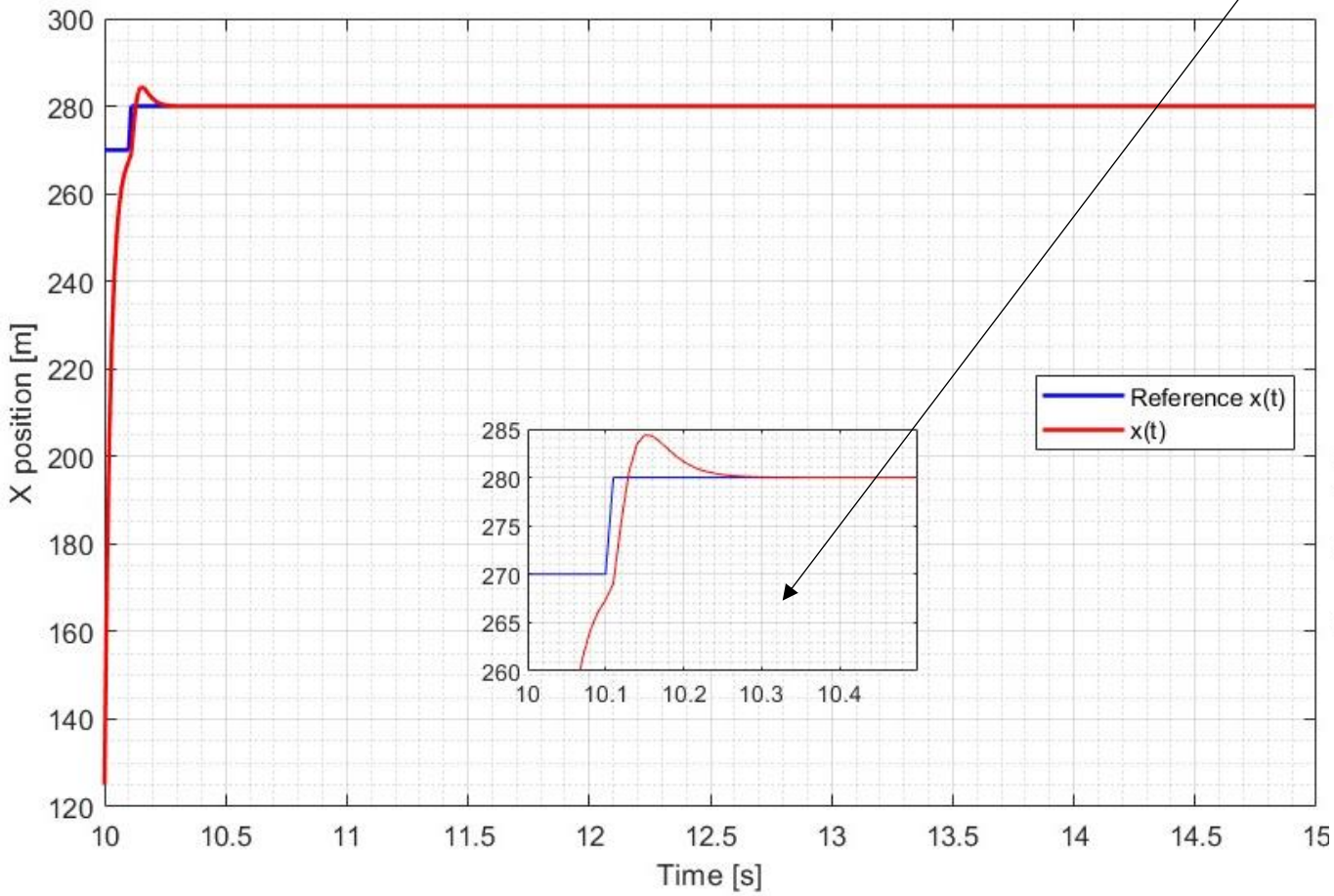
The greater the error, the greater the speed action

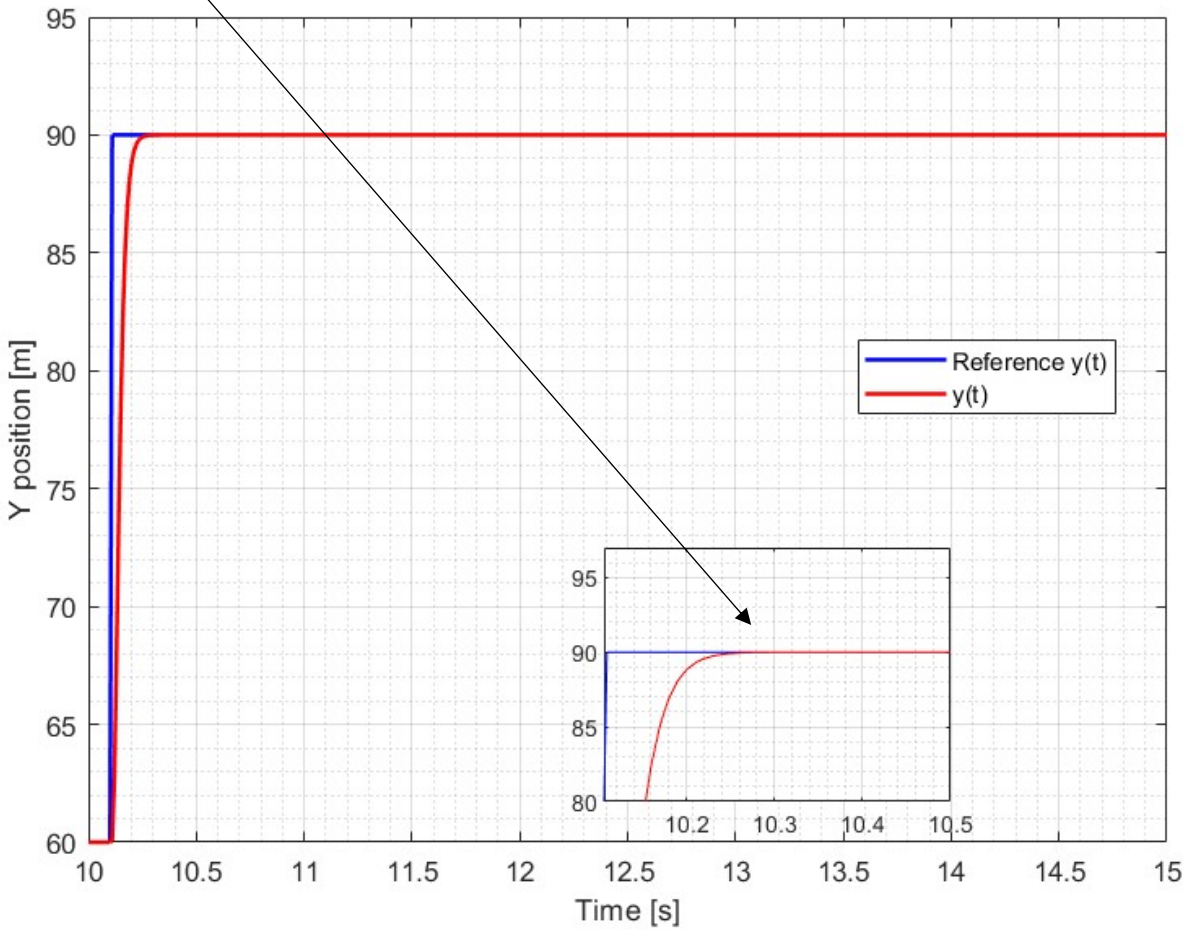
The error tends to zero, resulting in lower speed

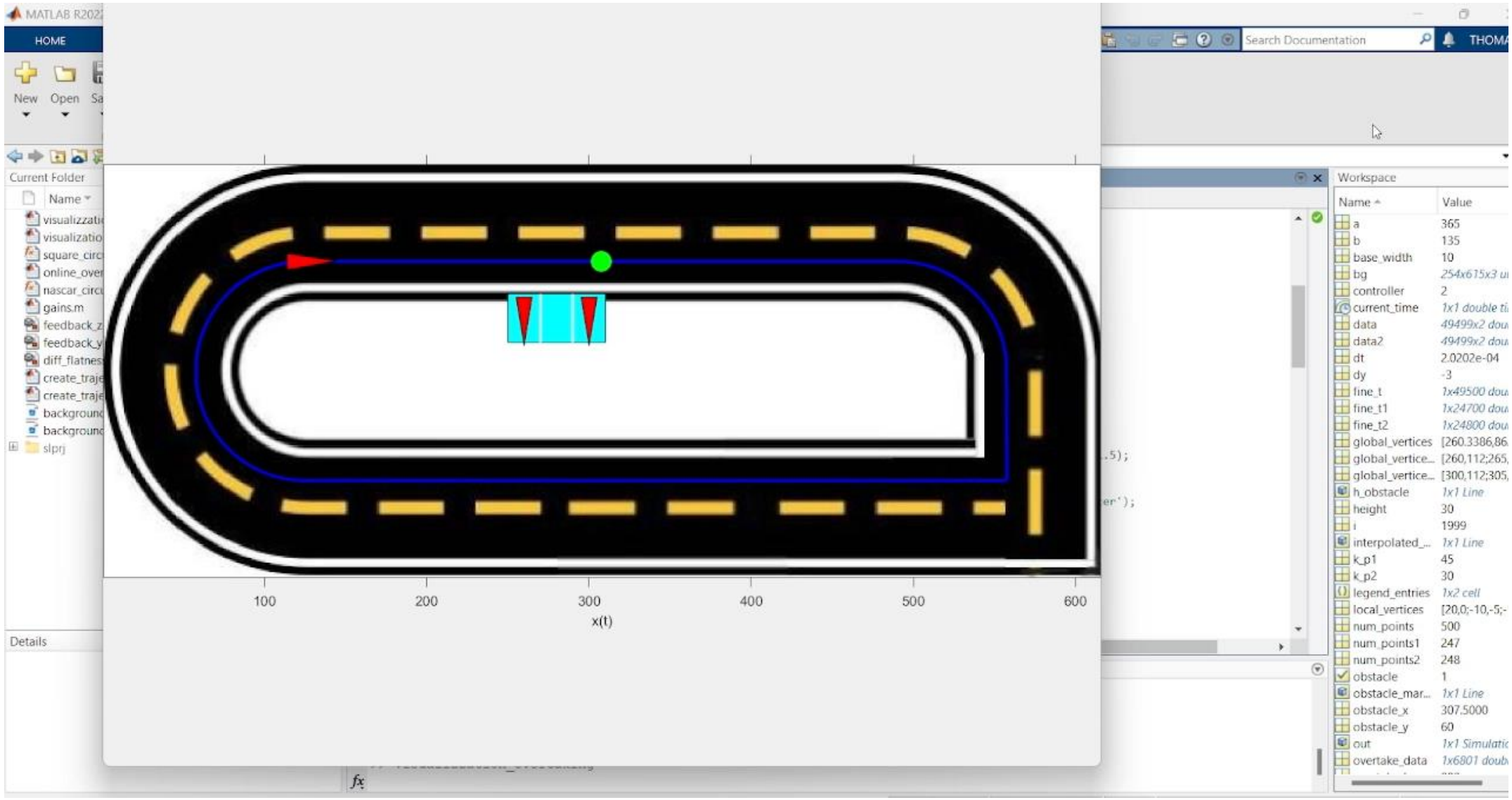


Plot

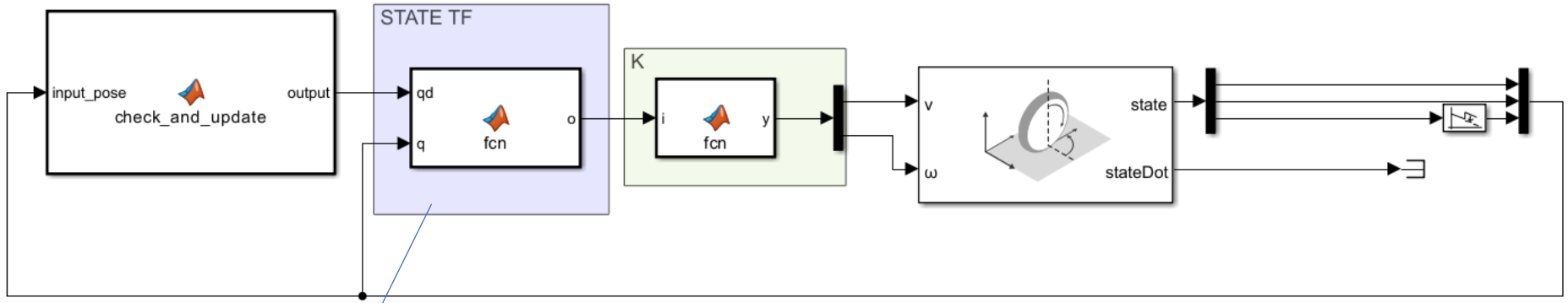
final position in 0.3s






MATLAB R2022
HOME
New
Open
Current Folder
Name
Details
100
200
300
400
500
600
x(t)
Search Documentation
Workspace
Name
Value
a 365
b 135
base_width 10
controller 2
dt 2.0202e-04
dy -3
fine_t
fine_t1
fine_t2
global_vertices
h_obstacle
height 30
i 1999
k_p1 45
k_p2 30
legend_entries
local_vertices
num_points 500
num_points1 247
num_points2 248
obstacle 1
obstacle_x 307.5000
obstacle_y 60
out
overtake_data

Posture regulation (1° version)



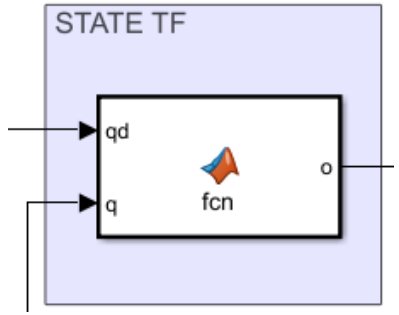
Drive the unicycle to a full pose $\quad \mathbf{q}_{\text{des}} = \begin{bmatrix} P_{\text{des}} \\ \theta_{\text{des}} \end{bmatrix}$

Convert to polar coordinates

The final orientation is not controlled

State trasformation

```
function o = fcn(qd,q)
    xi=q(1);
    yi=q(2);
    thetai=q(3);

    R = [cos(qd(3)), sin(qd(3));
        -sin(qd(3)), cos(qd(3))];


    x = (xi - qd(1));
    y = (yi - qd(2));
    theta = (thetai - qd(3));

    xy_new = (R * [x; y])';
    x = xy_new(1);
    y = xy_new(2);


    theta=mod(theta+pi,2*pi)-pi;

    rho=sqrt(x^2+y^2);
    gamma=atan2(y,x)+pi-theta;
    delta=gamma+theta;

    delta=mod(delta+pi,2*pi)-pi;
    gamma=mod(gamma+pi,2*pi)-pi;

    o = [rho;gamma;delta];
```



Before the state transformation we apply the rotation matrix $R = \begin{bmatrix} \cos\theta_d & \sin\theta_d \\ -\sin\theta_d & \cos\theta_d \end{bmatrix}$ in order to obtain the position error w.r.t. the final desired body frame

$$\rho = \sqrt{x^2 + y^2}$$

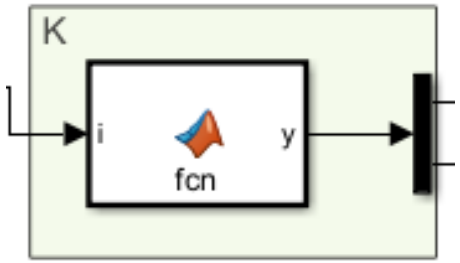
$$\gamma = \text{atan2}(y, x) + \pi - \theta$$

$$\delta = \gamma + \theta$$

polar coordinates

Control law

```
function y = fcn(i)
k1=20;
k2=30;
k3=4;
rho=i(1);
gamma=i(2);
delta=i(3);
v=k1*rho*cos(gamma);
w=k2*gamma+(k1* ((sin(gamma)*cos(gamma))/gamma) * (gamma+k3*delta));
if ~isfinite(w)
    w = 0;  % Set omega to a default value
end
y = [v;w];
```



$$\omega = k_2\gamma + \left(k_1 \cdot \frac{\sin(\gamma)\cos(\gamma)}{\gamma}\right) \cdot (\gamma + k_3\delta)$$

New term

Ensure that ω is finite

At starting time γ is 0

$$\dot{\gamma} = \frac{\sin\gamma}{\rho}v - \omega$$

$$\dot{\delta} = \frac{\sin\gamma}{\rho}v$$

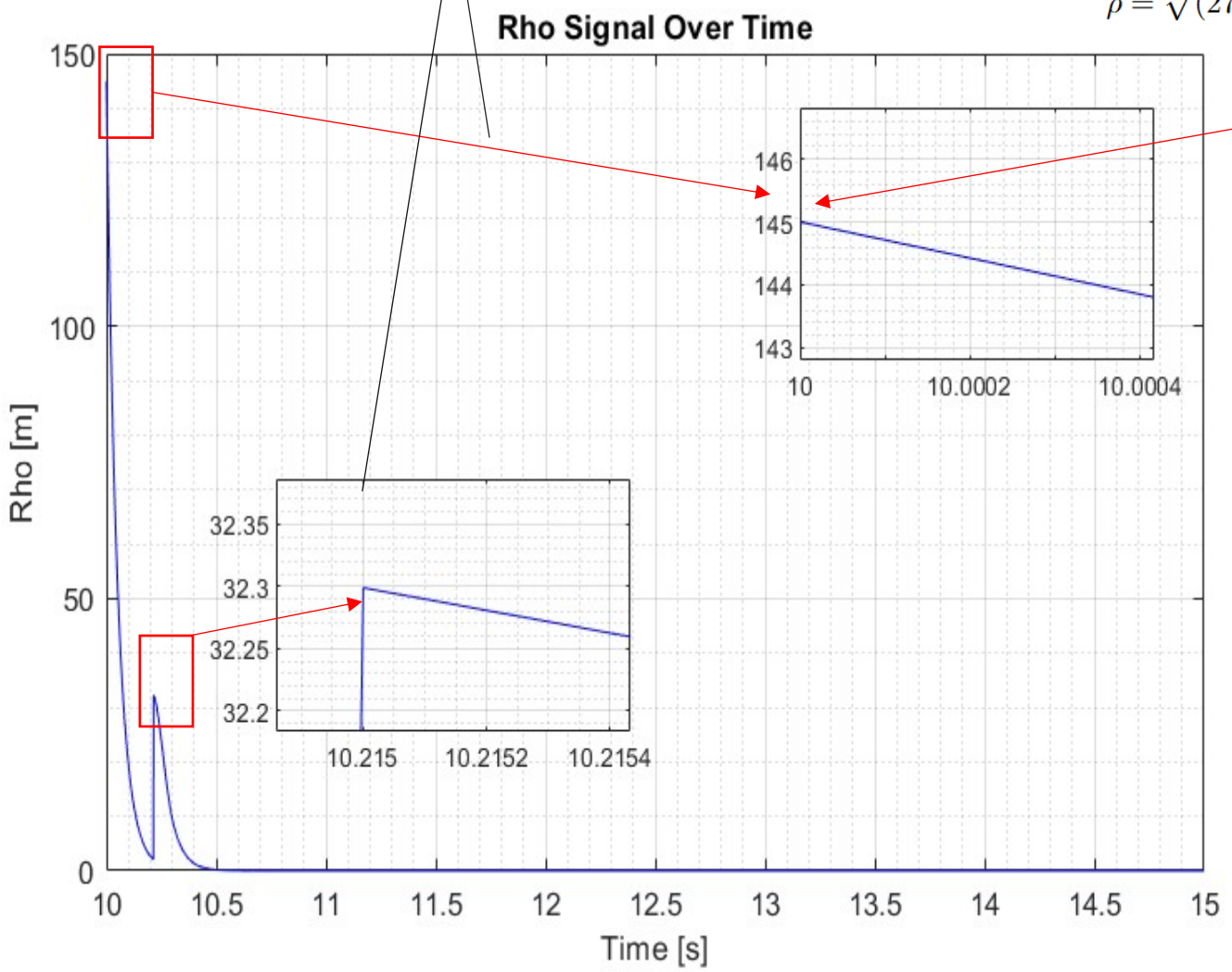
Problem: potential singularity when ρ=0

Plot

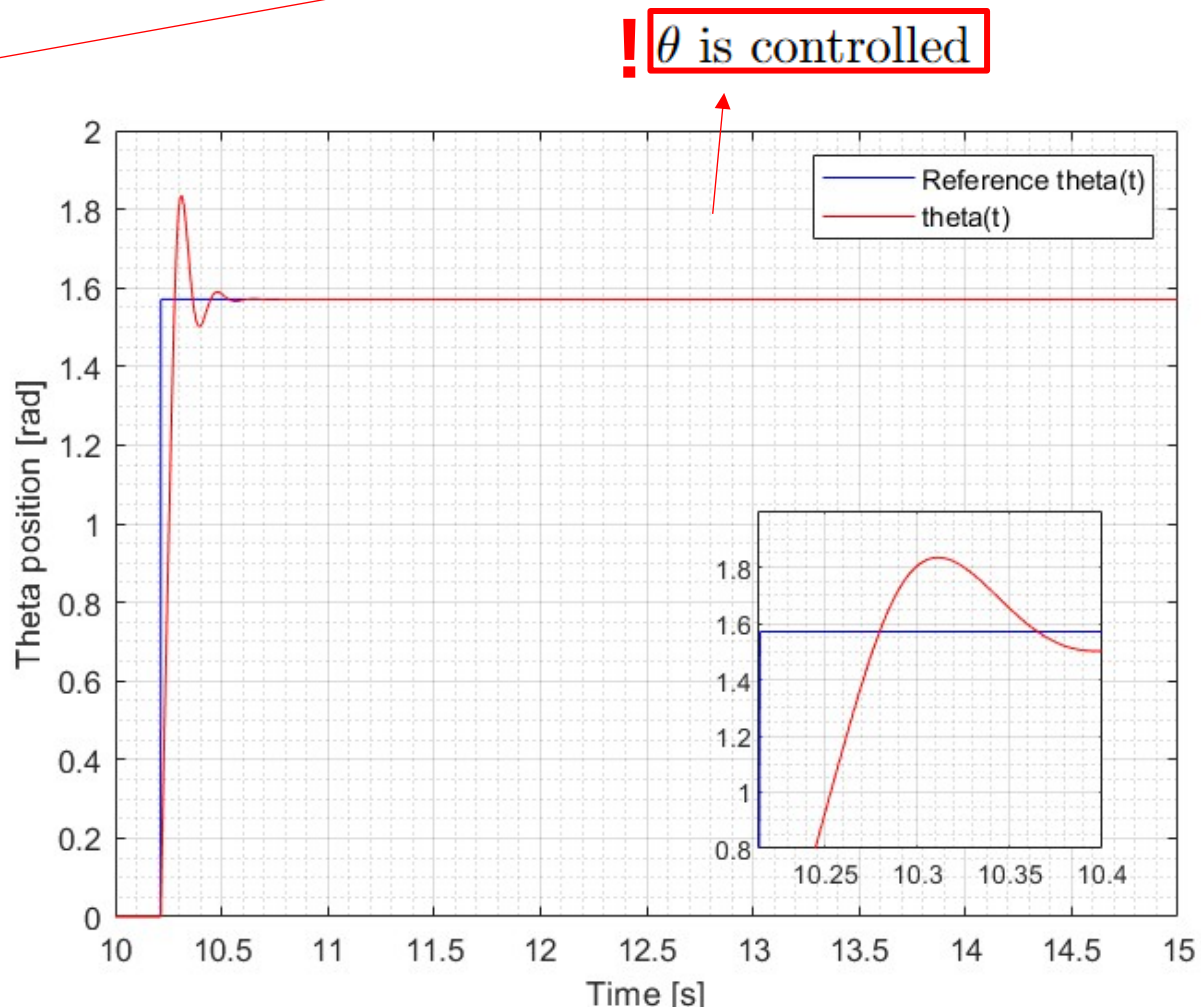
Lenght from the origin

Remember the first waypoint:

$$\rho = \sqrt{(270-125)^2 + (60-60)^2} = \sqrt{145^2 + 0^2} = 145$$

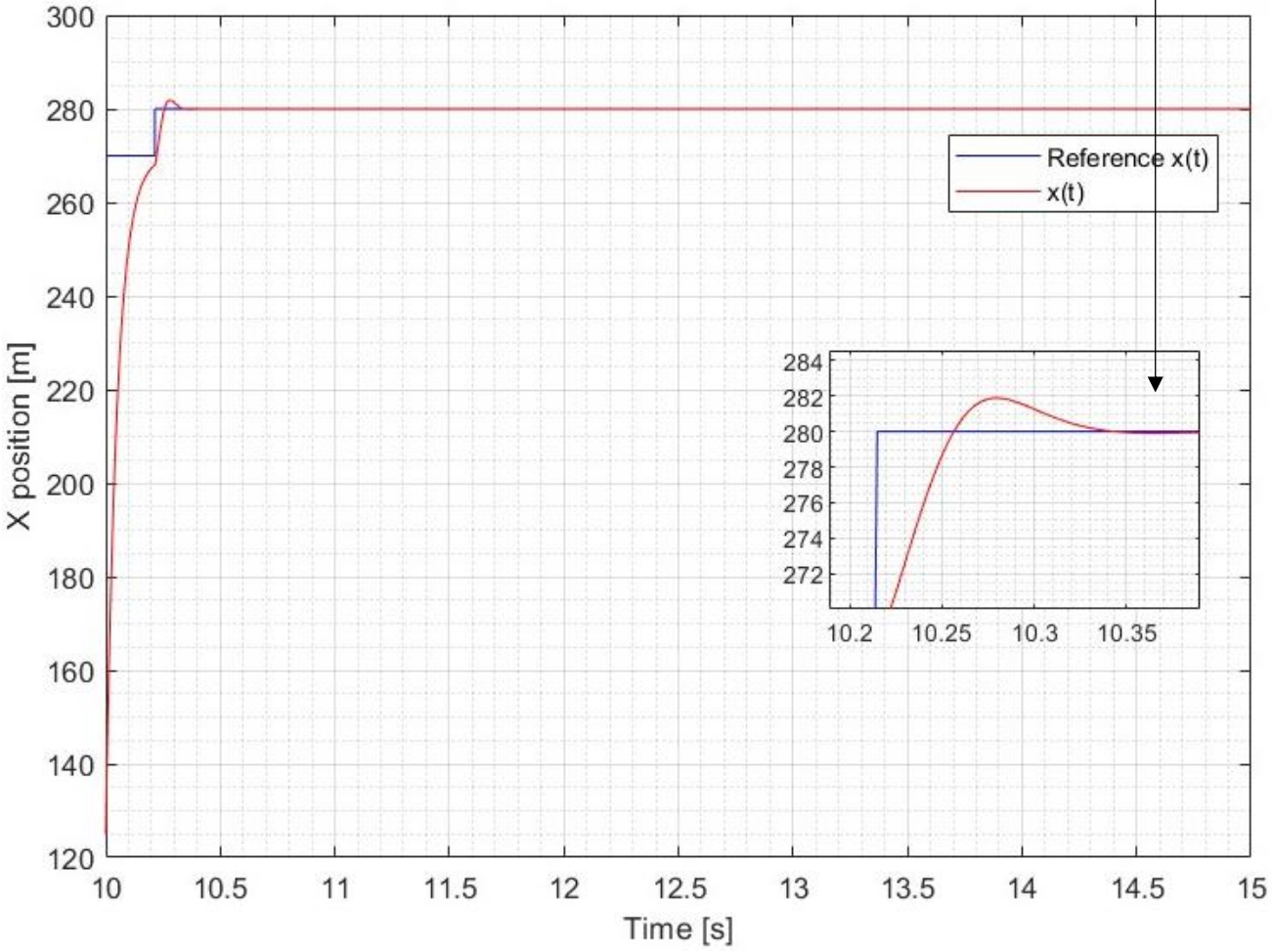


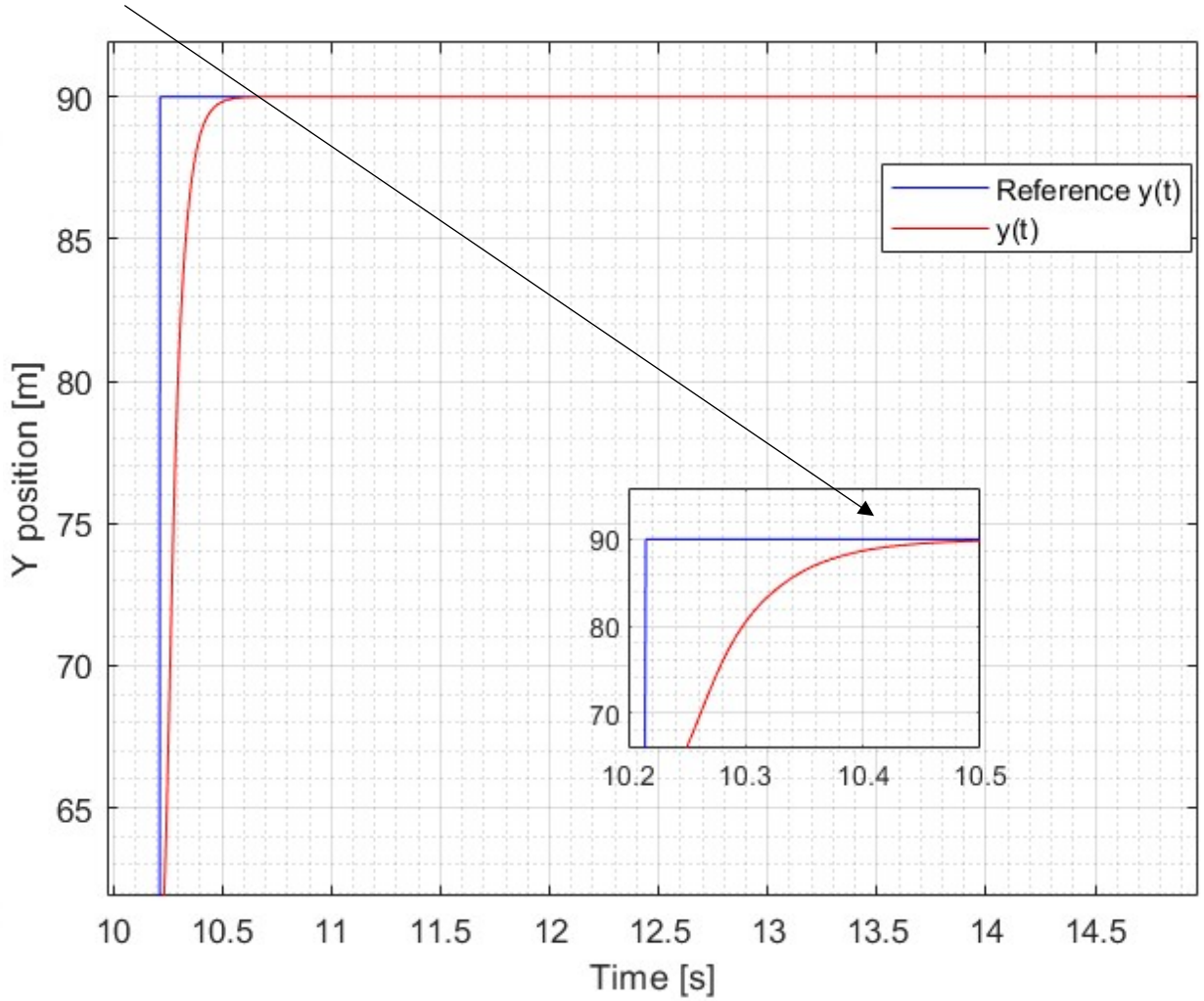
! θ is controlled



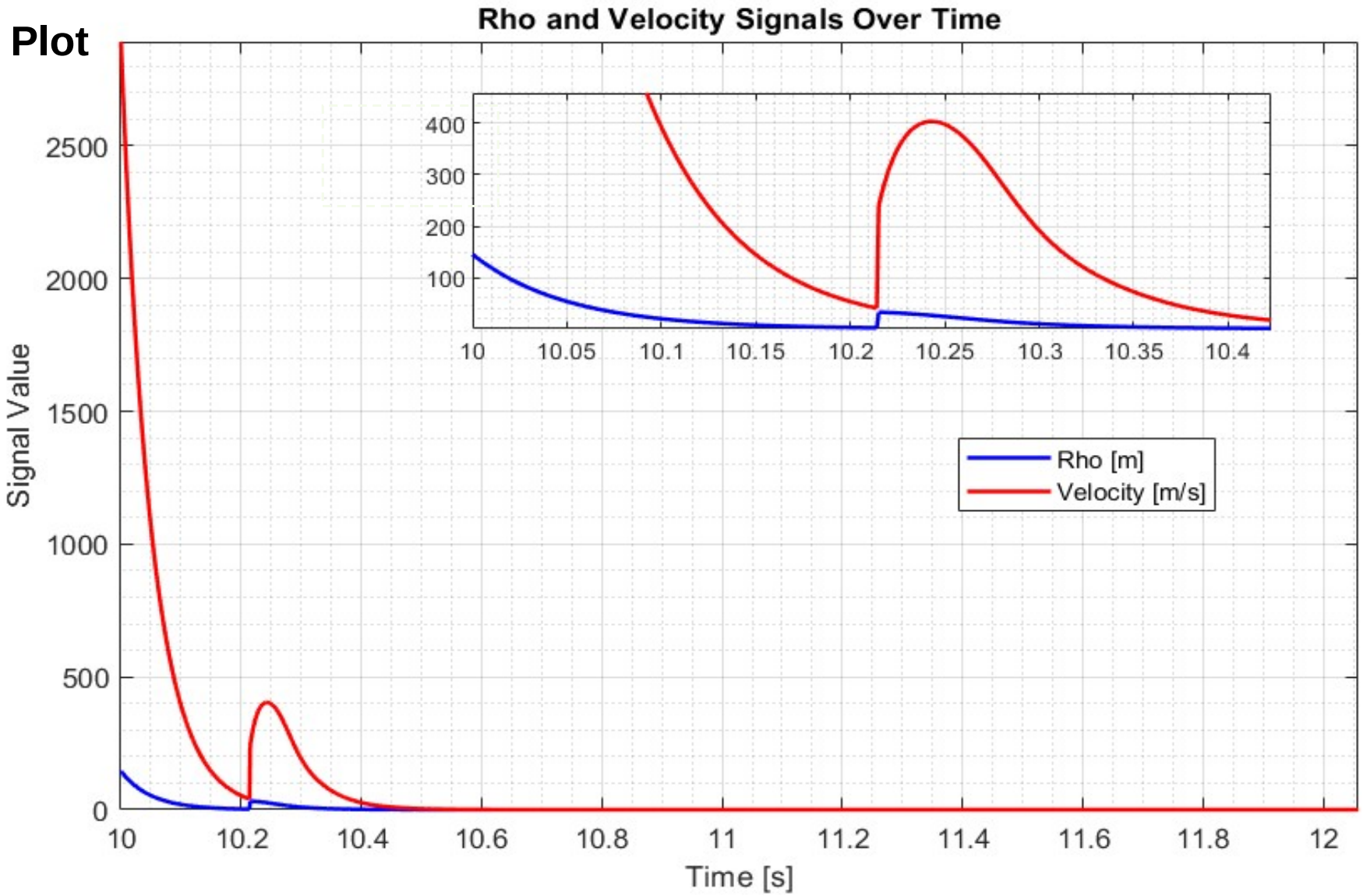
Plot

final position in 0.5s





Plot

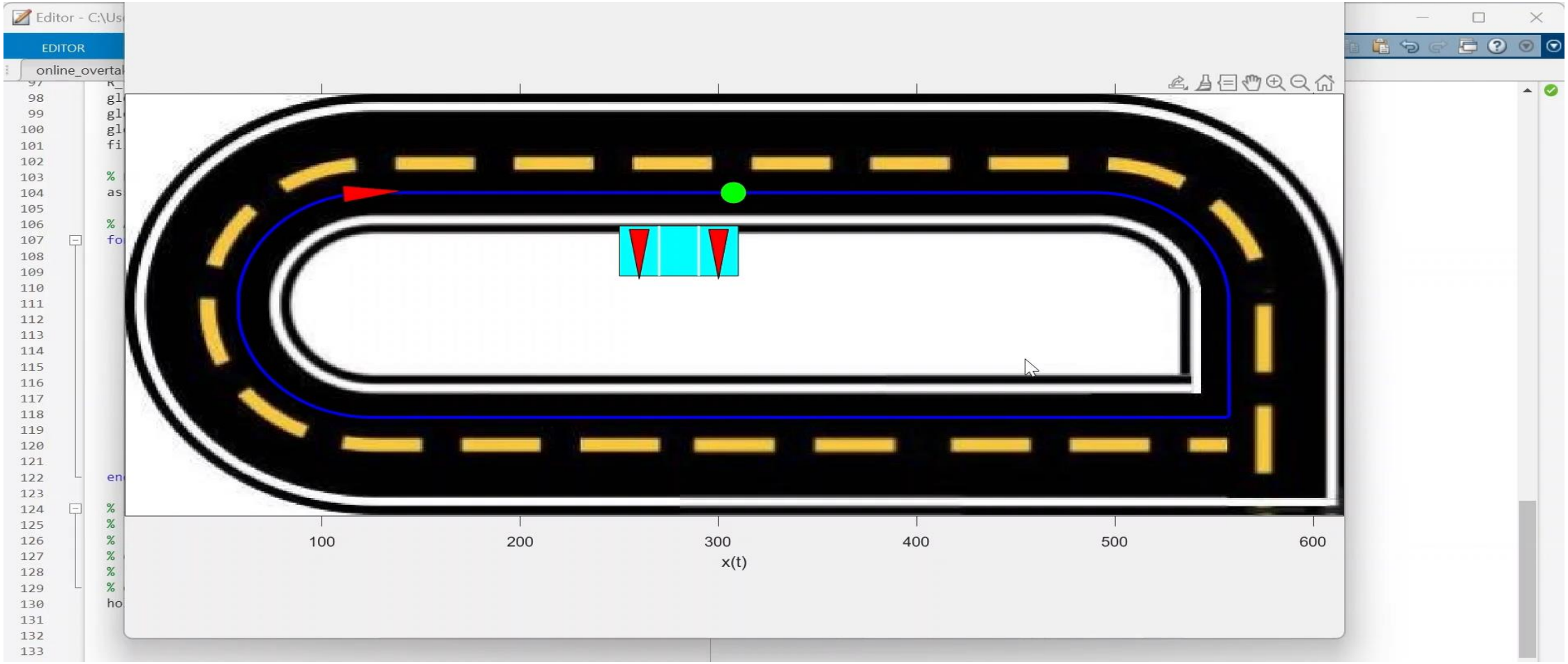


V is modulated based on the distance from the origin

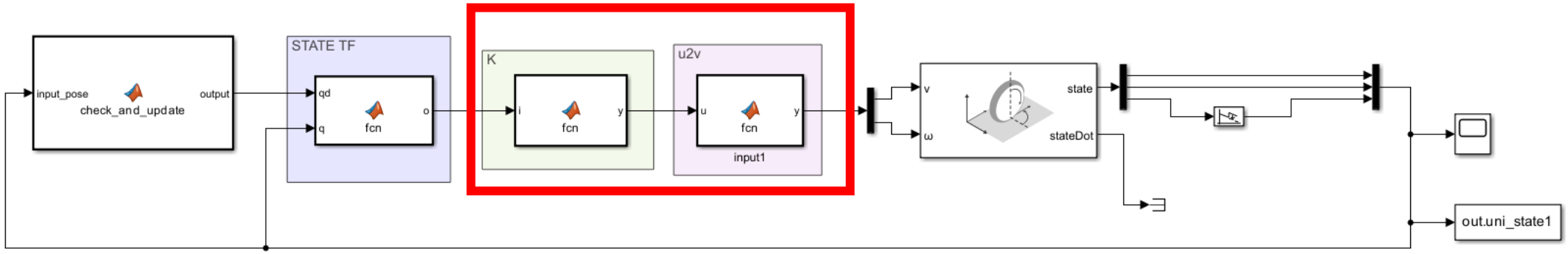
High speed far from the origin

Greater precision with lower speed when close to the origin.

Posture regulation(with singularity)



Posture regulation (2°version)



Drive the unicycle to a full pose

$$\mathbf{q}_{\text{des}} = \begin{bmatrix} P_{\text{des}} \\ \theta_{\text{des}} \end{bmatrix}$$

The final orientation is ~~not~~ controlled

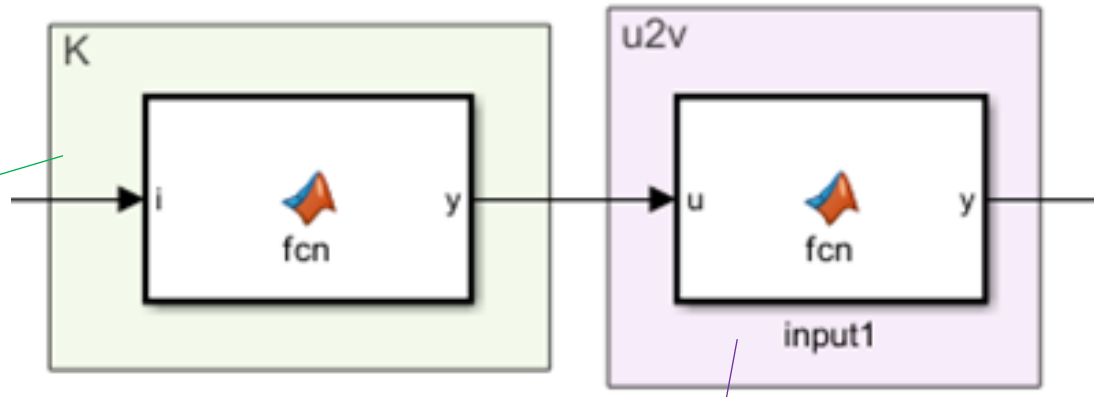
New:

Avoid singularity on ρ with change of input

$$\text{new variable } u = \frac{v}{\rho}$$

New Control law

```
function y = fcn(i)
    k1=10;
    k2=40;
    k3=8;

    rho=i(1);
    gamma=i(2);
    delta=i(3);

    u=k1*cos(gamma);

    if gamma > 0.0001
        w = k2 * gamma + k1 * sin(gamma) * cos(gamma) / gamma * (gamma + k3 * delta);
    else
        w=0;
    end

    y = [rho;u;w];
```

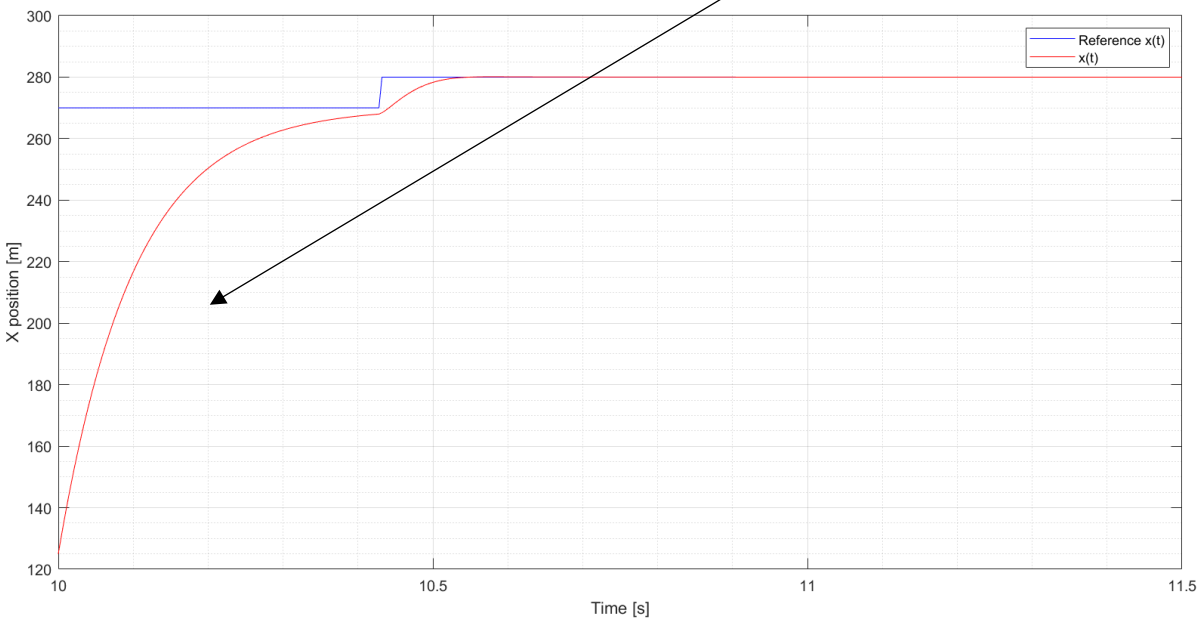


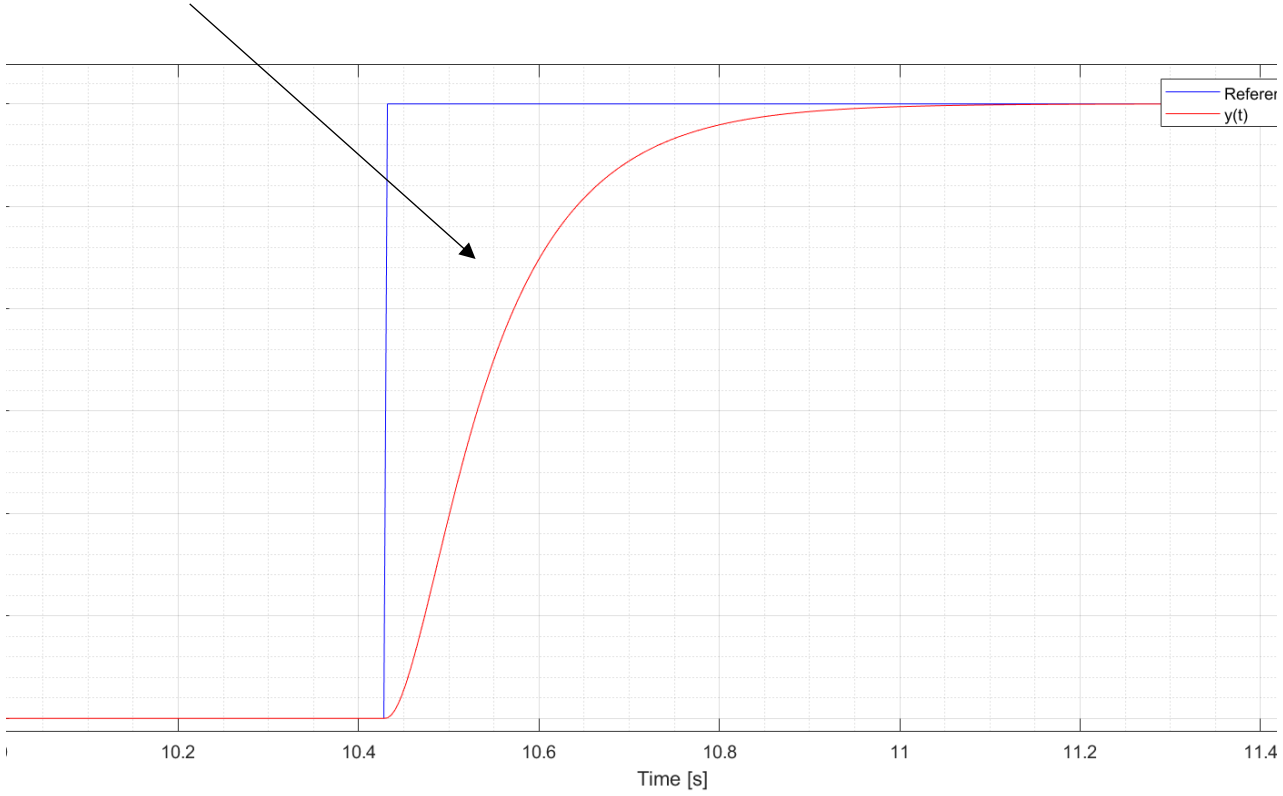
new variable $u = \frac{v}{\rho}$

```
function y = fcn(u)
    v=u(2)*u(1);
    w=u(3);
    y = [v;w];
```

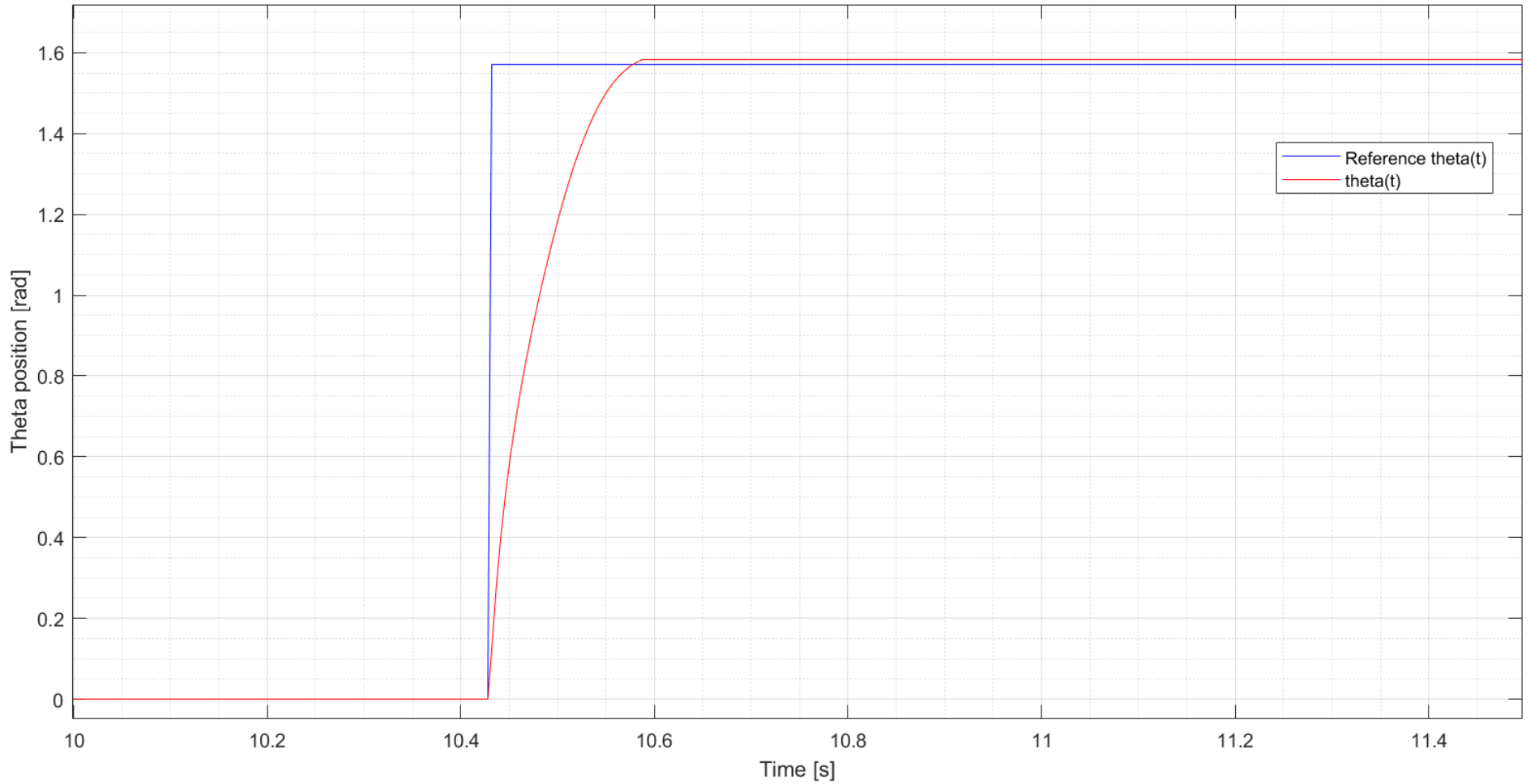
Plot

Reaches final position in 0.5 s
The concave shape is given by the fact that the linear velocity is proportional to roh

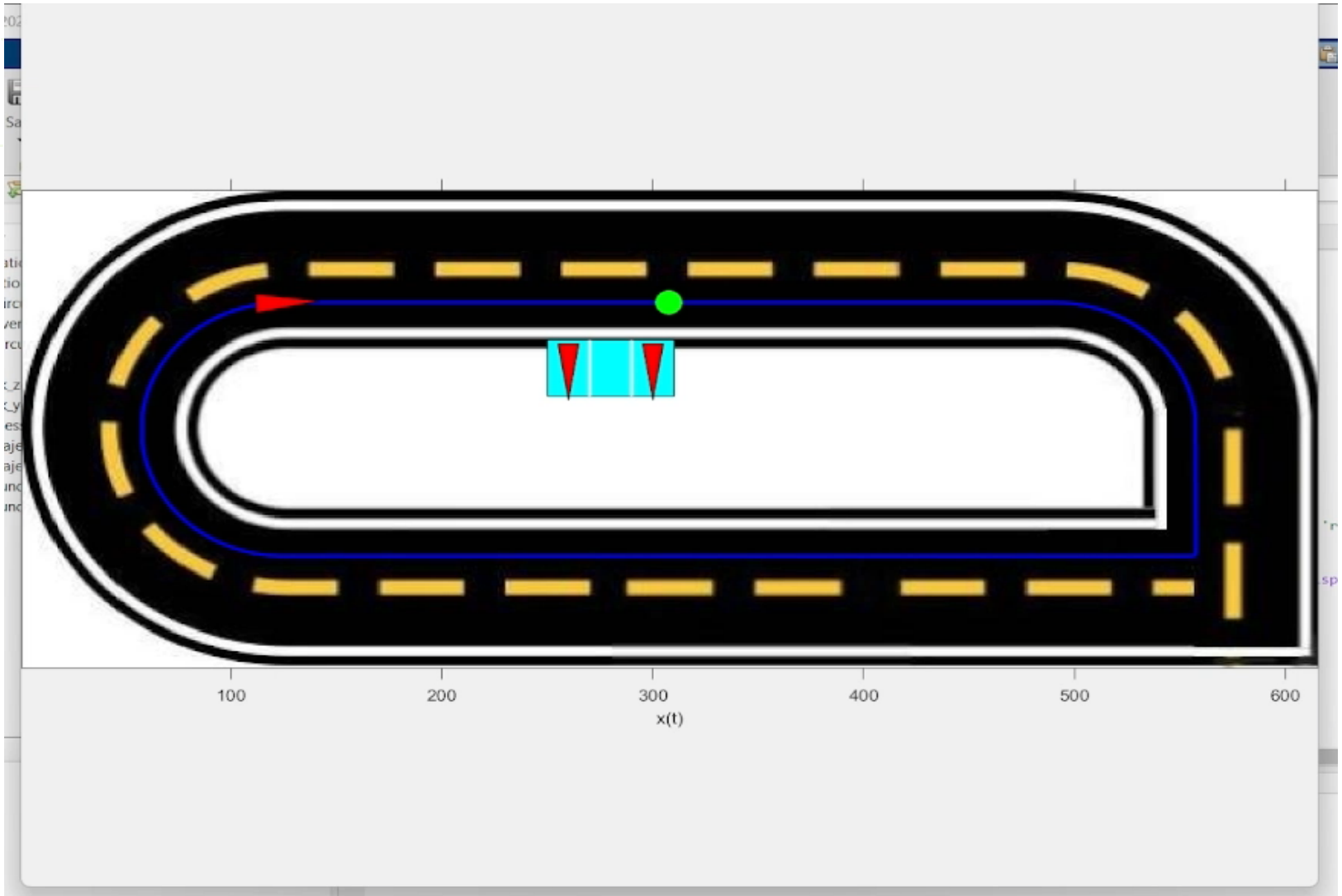




Plot


Reference theta(t)
theta(t)
Theta position [rad]
Time [s]
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
10
10.2
10.4
10.6
10.8
11
11.2
11.4

Posture regulation (without singularity)



Final considerations - tracking

BEST AMONG THE FOUR

State error feedback

- Works with the state (x,y,theta) → theta directly controlled
- Trajectory must be twice differentiable and persistent
- Differential Flatness block induces constraints on the trajectory
- System dynamic is coupled
- Controller design is more complicated than the others, gains are time varying and related to the velocities
- Convergence is difficult to guarantee (linear → only locally, nonlinear → only with bounded velocities and bounded derivatives)

Output error feedback , B-point

- Theta is not directly controlled, but coupled in the dynamics with x and y
- It works with any trajectories, given in terms of xb and yb, so no smooth constraints (as far as b!=0)
- The dynamic becomes linear,decoupled, SISO , single integrator
- Control design is easy, PD with gains TI and > 0

Output error feedback , further derivatives

- Same as diff flatn, trajectory must be smooth (differentiable) and persistent , v!=0 (singular mapping)
- System dynamic becomes decoupled, linear, double integrator
- Controller design is quite easy, gains TI and >0 (but there are 4 gains)

Final considerations - regulation

Cartesian regulation

- Theta non controlled
- v proportional to ep
- Controller design simple, 2 gains > 0
- No singularities issues

Posture regulation

- Theta is now controlled but indirectly (through gamma and delta)
- Controller design is more difficult, 3 gains and gamma at the denom (even if there is semplification in case)
- Singularity problem in rho=0 → solved by shifting singularity in input

Output error feedback (B-point) and posture regulation(without singularity)